Oracle9*i*

ケース・スタディ - XML アプリケーション

リリース 1（9.0.1）

2001 年 10 月

部品番号：J04169-01

ORACLE®

Oracle9*i* ケース・スタディ - XML アプリケーション , リリース 1（9.0.1）

部品番号：J04169-01

原本名：Oracle9*i* Case Studies - XML Applications, Release 1 (9.0.1)

原本部品番号：A88895-01

原本著者：Shelley Higgins

原本協力者：Valerie Moore, Sandeepan Banerjee, Robert Dell'immagine, Robert Hall, Karun K, Murali Krishnaprasad, Olivier LeDiouris, Paul Nock, Ami Parekh, Rajesh Raheja, Carol Roston, Frank Rovitto, Mark Scardina, Manh-Kiet (Allen) Yap, Ari Adler, Phil Bates, Catherine Bauer, Mark Bauer, Steve Cave, Steve Corbett, Claire Dessaux, Roger Ford, William Gietz, Andy Page, Rahul Pathak, Padmini Ranganathan, Jim Rawles

目次

2 Oracle XML アプリケーションのモデリングおよび設計問題

第 II 部　XML を使用したコンテンツおよびドキュメントの管理

3　Oracle9*i*AS Wireless Edition と XML

4 XML および XSQL を使用した表示のカスタマイズ : Flight Finder

第 III 部　XML データ交換

7　XSL を使用した Discoverer 4*i* Viewer のカスタマイズ

8 オンライン B2B XML アプリケーション : 手順

9 Service Delivery Platform（SDP）と XML

A XML の手引き

用語集

索引

はじめに

この章の内容は、次のとおりです。

このマニュアルの内容

このマニュアルでは、Oracle9*i* の XML 対応データベース・テクノロジを使用する事例とアプリケーションについて説明します。Oracle XML 対応テクノロジを使用して、データベースで XML データの格納、管理、問合せおよび交換を行うための様々な方法について説明します。

また、実際のビジネス・アプリケーションに基づく複数の使用例についても説明します。各事例は、主な機能に従って記載されています。つまり、次の高レベルな作業のうちいずれか一方で使用されるか、あるいは両方で使用されるかによって、異なる事例が示されます。

- XML ベースのコンテンツやドキュメントの管理。第 II 部「XML を使用したコンテンツおよびドキュメントの管理」を参照してください。
- Business-to-Business（B2B）、Business-to-Consumer（B2C）、アプリケーション間（A2A）または peer-to-peer（P2P）アプリケーションでの XML データ交換。第 III 部「XML データ交換」を参照してください。アプリケーションの詳細は、第 8 章「オンライン B2B XML アプリケーション : 手順」を参照してください。第 8 章では、XML の B2B データ交換およびカスタマイズされたプレゼンテーション・アプリケーションの構築方法を説明しています。

構成済み XML または分解済み（生成済み）XML

通常、XML 文書は次のうちいずれかの方法で処理されます。

- 構成済み XML 文書。LOB に格納されます。
- 分解済み XML 文書フラグメント。リレーショナル表に格納され、XML タグはデータベース表のそれぞれの列にマップされます。分解済みまたは断片化された XML 文書は、構成済み XML 文書へと再生成できます。

Oracle XML 対応テクノロジ

Oracle XML 対応テクノロジの主要コンポーネントは、XML Developer's Kit（XDK）です。これらのキットは、次の 4 つの言語による実装で使用可能です。

- **Java**。XDK for Java、XDK for JavaBeans および XML SQL Utility for Java。
- **PL/SQL**。XDK for PL/SQL および XML SQL Utility for PL/SQL。
- **C**。XDK for C。
- **C++**。XDK for C++。

関連項目：

- 『Oracle9*i* アプリケーション開発者ガイド - XML』
- 『Oracle9*i* XML リファレンス』

対象読者

このマニュアルは、Oracle9*i* で XML アプリケーションを構築する開発者を対象としています。

前提知識

このマニュアルを使用するには、XML と XSL の知識があれば役立ちますが、必須ではありません。詳細の情報源については、付録 A「XML の手引き」および『Oracle9*i* アプリケーション開発者ガイド - XML』の第 3 章の末尾にある FAQ の項を参照してください。

このマニュアルには、SQL、Java、PL/SQL、C または C++ による多数の例が記載されており、このうち 1 つ以上の言語に関して作業上の知識を持っていることが前提となります。

機能の範囲と可用性

このマニュアルの情報は、Oracle XML 対応テクノロジ・コンポーネントに関する現在の情報です。これらの情報は常に更新されています。最新情報を確認するには、次の Oracle Technology Network Japan（OTN Japan）の URL を参照してください。
http://otn.oracle.com/tech/xml/xml.html

このマニュアルの構成

このマニュアルは 3 部構成で、9 つの章、付録、索引および用語集が含まれています。

- 第 I 部「Oracle XML 対応テクノロジの概要」
  - 第 1 章「Oracle XML 対応テクノロジ」では、Oracle9*i* の XML コンポーネントの使用方法、データベースでの XML サポート、XMLType と URI 参照の使用方法、XML SQL Utility（XSU）、Oracle Text を適用して XML 文書から情報を検索し、取り出す方法の概要と基本情報について説明します。
  - 第 2 章「Oracle XML アプリケーションのモデリングおよび設計問題」では、XML をデータベースに格納する様々な方法、XML の設計上の問題および XML をデータベースにロードする方法について説明します。また、実際のビジネスでの使用例を示し、XDK や使用を検討できる他の Oracle XML 対応コンポーネントについても説明します。

- 第 II 部「XML を使用したコンテンツおよびドキュメントの管理」
  - 第 3 章「Oracle9iAS Wireless Edition と XML」では、Wireless Edition（Portal-to-Go）コンポーネントについて説明し、これらのコンポーネントを使用して Web サイトからコンテンツを抽出し、XML に変換し、どのデバイスでも表示できるようにデータを変換する方法について説明します。
  - 第 4 章「XML および XSQL を使用した表示のカスタマイズ : Flight Finder」では、Flight Finder アプリケーションを使用してデータベースとの間で XML を生成し、XSQL Servlet を使用して問合せを処理し、結果を XML として出力する方法について説明します。また、Flight Finder で XSL スタイルシートを使用して XML データをフォーマットする方法も説明します。このアプリケーションおよびデモは、Oracle Technology Network（OTN）でも入手できます。
  - 第 5 章「XML を使用したコンテンツのカスタマイズ : Dynamic News アプリケーション」では、Dynamic News アプリケーション、このアプリケーションで使用する 3 つのカスタム・サーブレット、Oracle9*i* から XML SQL Utility（XSU）を使用してニュース・コンテンツにアクセスする方法について説明します。このアプリケーションには、静的、半動的および動的という 3 つのユーザー・カスタマイズ・レベルが用意されています。この章では、表現のカスタマイズの詳細も一部説明します。
  - 第 6 章「Oracle Internet File System を使用した XML アプリケーションの作成」では、XML サポートに重点を置いて Oracle 9*i*FS を紹介します。
- 第 III 部「XML データ交換」
  - 第 7 章「XSL を使用した Discoverer 4i Viewer のカスタマイズ」では、Discoverer 4*i*（9*i*）を使用して Web アプリケーションをカスタマイズし、ビジネス・インテリジェンス、フォームおよびレポートを活用できるようにする方法について説明します。この章では、XSL スタイルシートを使用して表現をカスタマイズする方法の詳細も一部説明します。
  - 第 8 章「オンライン B2B XML アプリケーション : 手順」では、XSQL Servlet とアドバンスト・キューイング（AQ）を使用して、B2B オンライン・カタログ XML アプリケーションを構築し、実装する方法の 1 つを詳細に説明します。これには、生成された XML メッセージを様々なユーザー・デバイスに合せて変換するためのスクリプトが含まれています。このアプリケーションの拡張バージョンと簡略バージョンは、http://http:otn.oracle.com/tech/xml でも入手できます。
  - 第 9 章「Service Delivery Platform（SDP）と XML」では、Phone Number Portability アプリケーションについて説明します。この章では、XML メッセージ機能を各種通信製品およびシステムで使用する方法と、*i*Message Studio、Event Manager および Adapters で設計する方法の概要を説明します。

関連文書

詳細は、次の Oracle マニュアルを参照してください。

- 『Oracle9*i* アプリケーション開発者ガイド - XML』
- 『Oracle9*i* データベース新機能』。Oracle9*i* および Oracle9*i* Enterprise Edition の違いと、それぞれで使用可能な機能およびオプションについて説明しています。また、Oracle9*i* の新機能もすべて記載されています。
- JDeveloper のマニュアル
- 『Oracle9*i* アプリケーション開発者ガイド - 基礎編』
- 『Oracle8*i* アプリケーション開発者ガイド - アドバンスト・キューイング』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』
- 『Oracle9*i* XML リファレンス』

リリース・ノート、インストレーション・ガイド、ホワイト・ペーパーのダウンロード

リリース・ノート、インストレーション・マニュアル、ホワイト・ペーパーまたはその他の関連ドキュメントは、Oracle Technology Network Japan（OTN Japan）に接続すれば、無償でダウンロードできます。OTN を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

`http://otn.oracle.co.jp/membership/`

OTN のユーザー名とパスワードを取得済みであれば、次の OTN Web サイトのドキュメント・セクションに直接接続できます。

`http://otn.oracle.co.jp/document/`

このマニュアルにオンラインでアクセスする方法

このマニュアルのコピーは次の CD に収録されており、次の URL からダウンロードすることもできます。

- Oracle9*i* ソフトウェア CD に添付されているドキュメント CD。
- Oracle Technology Network Japan（OTN Japan）の URL である http://otn.oracle.co.jp/docs/index.htm の「Data Server」（または使用中の他の製品）。たとえば、「Oracle9*i*」>「General Documentation Release 1 (9.0.1)」（または、指定する必要のある他のセクション）を選択します。

表記規則

このマニュアル・セットの本文とコード例に使用されている表記規則について説明します。

- 本文中の表記規則
- コード例の表記規則

本文中の表記規則

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則と使用例を示しています。

表記規則	意味	例
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。	この句を指定する場合は、**索引構成表**を作成します。
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、RMAN キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージとメソッドの他、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	この句は、`NUMBER` 列に対してのみ指定できます。 `BACKUP` コマンドを使用すると、データベースのバックアップを作成できます。 `USER_TABLES` データ・ディクショナリ・ビューの `TABLE_NAME` 列を問い合せます。 `DBMS_STATS.GENERATE_STATS` プロシージャを使用します。

表記規則	意味	例
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびサンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネット・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニット、およびパラメータ値があります。 **注意：**一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。	`sqlplus` と入力して SQL*Plus をオープンします。 パスワードは `orapwd` ファイルに指定されています。 データ・ファイルと制御ファイルのバックアップを `/disk1/oracle/dbs` ディレクトリに作成します。 `department_id`、`department_name` および `location_id` の各列は、`hr.departments` 表にあります。 初期化パラメータ `QUERY_REWRITE_ENABLED` を `true` に設定します。 `oe` ユーザーで接続します。 これらのメソッドは `JRepUtil` クラスに実装されます。
固定幅フォントでイタリックの小文字	固定幅フォントでイタリックの小文字は、プレースホルダまたは変数を示します。	*`parallel_clause`* を指定できます。 `U`*`old_release`*`.SQL` を実行します。*`old_release`* は、アップグレード前にインストールしていたリリースです。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus またはその他のコマンドライン文を示します。次のように、固定幅フォントで、通常の本文とは区別して記載されています。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表は、コード列の記載上の表記規則と使用例を示します。

表記規則	意味	例
[ ]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体は入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のどちらかを示します。 ■ 例に直接関係のないコード部分が省略されていること。 ■ コードの一部が繰返し可能であること。	CREATE TABLE ... AS *subquery*; SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関係のない数行のコードが省略されていることを示します。	
その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、示されているとおりに入力してください。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック	イタリックの文字は、特定の値を指定する必要のあるプレースホルダまたは変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

表記規則	意味	例

大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、小文字でも入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名を示します。
	注意：一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Oracle XML 対応テクノロジの新機能

この章では、次のリリースの新機能について説明します。

- Oracle9i リリース 1（9.0.1）で導入された XML 機能
- Oracle8i リリース 8.1.7 で導入された XML 機能

Oracle9*i* リリース 1（9.0.1）で導入された XML 機能

Oracle9*i* リリース 1（9.0.1）の新しい XML 機能は、次のとおりです。

XDK for Java

- XML Schema Processor for Java
- XML Parser for Java － DOM 2.0 および SAX 2.0 のサポート
- XSLT パフォーマンスの向上
- XML Schema Class Generator for Java のサポート

  関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「XML Schema Processor for Java の使用」

- **XSQL**
  - データベース・バインド変数のサポート。パフォーマンス改善のために、字句置換変数とデータベース・バインド変数がサポートされるようになりました。
  - Apache FOP を使用した PDF 出力のサポート。XSQL Pages を Apache FOP プロセッサと組み合せて、どんな XML コンテンツからでも Adobe PDF 出力を生成できます。
  - XSLT スタイルシート用のトラステッド・ホストのサポート。新しいセキュリティ機能により、非トラステッド・ホストからはスタイルシートを実行できないことが保証されます。
  - 非 Oracle JDBC ドライバの全面的なサポート。Oracle JDBC ドライバと非 Oracle JDBC ドライバの両方で、すべての問合せ、挿入、更新および削除機能を使用できます。
  - 動的に作成された XSQL ページの処理。XSQLRequest API では、プログラムで作成された XSQL ページを処理できるようになりました。
  - カスタム Connection Manager の使用。独自の Connection Manager を実装して、必要な方法でデータベース接続を処理できるようになりました。
  - インライン XML Schema の生成。オプションで、XML の問合せ結果の構造を記述するインライン XML Schema を生成できるようになりました。
  - 問合せ用のデフォルトの日付書式の設定。日付書式マスクを指定して、日付データのデフォルト書式を変更できるようになりました。
  - カスタム・シリアライザの記述。XSQL ページ・プロセッサからクライアントに戻される内容とその方法を制御する、カスタムのシリアライザを作成して使用できます。
  - 動的スタイルシート割当て。パラメータや SQL 問合せの結果に基づいてスタイルシートを動的に割り当てます。

- 転送済み XML の更新または削除。XML の挿入のみでなく、更新と削除もサポートされます。
- ターゲット列のみの挿入または更新。挿入要求または更新要求に含める必要のある列のリストを明示的に指定できるようになりました。
- ページ要求範囲付きオブジェクト。アクション・ハンドラは、ページ要求のコンテキスト内でオブジェクトを取得および設定して、ページ内のアクション間で状態を共有できるようになりました。
- ServletContext へのアクセス。HttpRequest および HttpResponse オブジェクトのみでなく、ServletContext にもアクセスできます。

  関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「XSQL ページ・パブリッシング・フレームワーク」

- **XDK for JavaBeans**
  - DBViewer Bean（新規）。XSL スタイルシートを適用し、結果の HTML をスクロール可能なスイング・パネルに表示して、データベース問合せや XML を表示します。
  - DBAccess Bean（新規）。DB Access Bean は、複数の XML およびテキスト・ドキュメントを保持する CLOB 表をメンテナンスします。

    関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「XML Transviewer Beans の使用」

- **XDK for C**
  - XML Parser for C － DOM 1.0 および DOM CORE 2.0（DOM のサブセット）
  - XML Schema Processor for C
  - XSLT パフォーマンスの向上

    関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「XML Schema Processor for C の使用」

- **XDK for C++**
  - XML Parser for C++ － DOM 1.0 および DOM CORE 2.0（DOM のサブセット）
  - XML Schema Processor for C++
  - XSLT パフォーマンスの向上

    関連項目： 『Oracle9*i* アプリケーション開発者ガイド - XML』の「XML Schema Processor for C++ の使用」

- **XDK for PL/SQL**
  - XSLT パフォーマンスの向上

XML SQL Utility（XSU）の機能

- SQL 問合せで指定された XML Schema の生成機能。
- XMLType および URI 参照のサポート
- SAX2 コールバックのストリームとしての XML の生成機能。
- データベースから XML を生成する場合の XML 属性のサポート。これにより、特定の列または列グループを XML 要素のかわりに XML 属性にマップするように簡単に指定できます。

XSU は、XDK for Java および XDK for PL/SQL の一部ともみなされます。

関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「XSU」

データベースの XML 関連の機能拡張

Extensible Markup Language（XML）は、データが構造化されているかどうかに関係なく Web 上で表示するために、W3C（World Wide Web Consortium）によって開発された標準形式です。Web ページなどのリソースは、Web 上のどこにあっても Universal Resource Identifier（URI）で識別されます。Oracle では、XML および URI データを処理するための型と、データベース自体に格納されたデータにアクセスするための URI のクラスである DBURI 参照が提供されています。また、データベースに外部と内部の URI を格納してアクセスできるように、新しい型セットも提供されています。

- **XMLType**。このオラクル社が提供する新しい型を使用すると、XML データをデータベースに格納して問合せできます。XMLType のメンバー関数を使用すると、XPath 式を使用した XML データへのアクセス、抽出および問合せが可能です。XPath も、XML 文書を横断するために W3C の委員会によって開発された標準です。Oracle の XMLType 関数では、W3C の XPath 式のサブセットがサポートされます。また、既存のリレーショナル・データやオブジェクト・リレーショナル・データから XMLType 型の値を作成できるように、一連の SQL 関数（SYS_XMLGEN や SYS_XMLAGG など）と PL/SQL パッケージ（DBMS_XMLGEN など）も提供されています。

  XMLType はシステム定義型のため、ファンクションの引数として、あるいは表やビューの列のデータ型として使用できます。表に XMLType 型の列を作成すると、Oracle 内部では、この列に対応付けられた実際の XML データが CLOB を使用して格納されます。すべての CLOB データと同様に、XML 文書全体のみを更新できます。XMLType 型の列では、Oracle Text の索引や他のファンクション索引を作成できます。

- **URI データ型**。Oracle では、URI 型のファミリ、つまり、継承階層により関連付けられた UriType、DBUriType および HttpUriType が提供されています。UriType はオブジェクト型で、残りの 2 つは UriType のサブタイプです。

- HttpUriType を使用すると、URL を外部 Web ページやファイルに格納できます。これらのファイルにはハイパー・テキスト転送プロトコル（Hypertext Transfer Protocol: HTTP）を使用してアクセスします。
- DBUriType を使用すると、データベース内のデータを参照する DBURI 参照を格納できます。UriType はスーパータイプのため、この型の列を作成し、そこに DBUriType 型または HttpUriType 型のインスタンスを格納できます。これにより、データベース内外に格納されたデータを参照し、データに一貫した方法でアクセスできます。

  DBURI 参照では、XPath に似た表現を使用してデータベース内のデータが参照されます。データベースを XML ツリーと考えると、表、行および列は XML 文書の要素とみなすことができます。たとえば、サンプルの人事管理ユーザー hr には、次の XML ツリーが表示されます。

```
<HR>
  <EMPLOYEES>
    <ROW>
      <EMPLOYEE_ID>205</EMPLOYEE_ID>
      <LAST_NAME>Higgins</LAST_NAME>
      <SALARY>12000</SALARY>
      .. <!-- other columns -->
    </ROW>
    ... <!-- other rows -->
  </EMPLOYEES>
  <!-- other tables..-->
</HR>
<!-- other user schemas on which you have some privilege on..-->
```

  DBURI 参照は、この仮想 XML 文書では単なる XPath 式です。そのため、EMPLOYEES 表内で従業員番号が 205 の従業員の SALARY 値を参照する場合は、DBURI 参照を次のように記述できます。

```
/HR/EMPLOYEES/ROW[EMPLOYEE_ID=205]/SALARY
```

  このモデルを使用すると、CLOB 列や他の列に格納されているデータを参照し、それを URL として外部に公開できます。Oracle では、このような URL を解釈できる標準的な URI サーブレットが提供されています。このサーブレットは、Oracle Servlet Engine の下にインストールして実行できます。

- **UriFactoryType**。UriFactoryType は、**ファクトリ型**、つまり、他のオブジェクト型を作成して戻すことができる型です。URL 文字列を指定すると、UriFactoryType では UriTypes の各種サブタイプのインスタンスを作成できます。URL 文字列が分析され、URL の型（HTTP、DBUri など）が識別され、そのサブタイプのインスタンスが作成されます。

関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』

- 「XML に対するデータベース・サポート」
- 「DBURI 参照」
- 「Oracle Text を使用した XML データの検索」

アドバンスト・キューイング（AQ）機能

新しいアドバンスト・キューイング機能には、次のように拡張 XML メッセージング・オプションが組み込まれています。

- Internet-Data-Access-Presentation（iDAP）
- HTTP アクセス用の AQXMLServlet
- XMLType キュー
- XML AQ メッセージ変換

関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「Oracle AQ を使用した XML データの交換」

メタデータ API

メタデータ API（新規）は、次の作業を実行するための単純で柔軟な一元化された手段を提供します。

- XML または作成 DDL としてのデータベース・オブジェクト（メタデータ）の定義全体の抽出
- 業界標準の XSLT（XML Stylesheet Transformation language）を介したメタデータの変換
- SQL DDL の生成によるデータベース・オブジェクトの再作成

メタデータ API は、インスタンスが稼働していれば Oracle9*i* で常に使用できます。Oracle Lite では使用できません。この API には、DBMS_METADATA PL/SQL パッケージ（新規）が含まれています。

関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「メタデータ API の使用」

Oracle Text（*inter*Media Text/Context）機能

新しいセクション・グループ PATH_SECTION_GROUP を使用すると、より洗練された新しいセクション検索機能を XML 文書に使用できます。

新しい Oracle Text 演算子は、次のとおりです。

* HASPATH() 演算子
* INPATH() 演算子

Oracle Text の PATH_SECTION_GROUP 機能には、次のサポートが含まれています。

- 大 / 小文字区別
- 直接の親が保証される複数タグのパスの検索
- ワイルドカード・レベルのパスの検索
- 検索による最上位タグの参照
- 属性値に応じた検索、セクションの有無による検索

関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「Oracle Text を使用した XML データの検索」

Phone Number Portability（SDP）

- ADSL と無線を含む新規アプリケーションの拡張サポート

関連項目： 第 9 章「Service Delivery Platform（SDP）と XML」

Oracle8*i* リリース 8.1.7 で導入された XML 機能

Oracle8*i* リリース 8.1.7 で導入された新しい XML 機能は、次のコンポーネントの拡張および改善版です。

- **XDK for Java**
- **XDK for C**
- **XDK for C++**
- **XDK for PL/SQL**
- **XML SQL Utility**

第 I 部

Oracle XML 対応テクノロジの概要

第 I 部では、Oracle XML 対応テクノロジと機能の概要について説明します。内容は、次のとおりです。

- 第 1 章「Oracle XML 対応テクノロジ」
- 第 2 章「Oracle XML アプリケーションのモデリングおよび設計問題」

1

Oracle XML 対応テクノロジ

この章の内容は、次のとおりです。

XML の概要

XML、W3C の XML 勧告、HTML と XML の違いおよび他の XML 構文の概要については、付録 A「XML の手引き」を参照してください。付録 A では、情報交換のためのインターネット標準である XML がデータベース・アプリケーションに使用する言語として適切かつ必要である理由についても説明します。

Oracle XML 対応テクノロジ

XML モデルは、構造化されたデータと半構造化されたデータからなっています。Oracle9*i* では、これらのデータの他に、複合データと構造化されていないデータもサポートされます。Oracle9*i* は、XML データの格納、問合せ、表示および操作をシステム固有の方法で処理できるという点で、XML に対応しています。

Oracle XML コンポーネント

Oracle XML コンポーネントは、次のとおりです。

- データベースの XML サポート
  - `XMLType` ― XML 文書の格納、問合せおよび取出しに使用する新しいデータ型
  - `SYS_XMLGEN` ― XML 文書を作成する SQL 関数
  - `SYS_XMLAGG` ― 複数の XML 文書を集計する SQL 関数
  - `DBMS_XMLGEN` ― SQL 問合せから XML を作成するビルトイン・パッケージ
  - URI のサポート ― グローバル参照とデータベース内参照の格納および取出し
  - テキストのサポート ― `XMLType` とテキスト列の XPath のサポート
- XML Developer's Kit（XDK）for Java
  - XML Parser for Java および XSLT Processor
  - XML Schema Processor for Java
  - XML Class Generator for Java
  - XSQL Servlet
  - XML SQL Utility（XSU）for Java
- XDK for JavaBeans
  - XML Transviewer Beans
    - DOMBuilder Bean
    - XSLTransformer Bean

    * DBAccessBean
    * TreeViewer Bean
    * SourceViewer Bean
    * XMLTransformPanel Bean
    * DBViewer Bean
- XDK for C
  - XML Parser for C
  - XML Schema Processor for C
- XDK for C++
  - XML Parser for C++
  - XML Schema Processor for C++
  - XML Class Generator for C++
- XDK for PL/SQL
  - XML Parser for PL/SQL
  - XML SQL Utility（XSU）for PL/SQL

代表的な XML ベースのビジネス・ソリューションは、次のとおりです。

- XML を使用したビジネス・データ交換
  - バイヤー / サプライヤ間の透過的な取引の自動化
  - パートナとのシームレスな統合および HTTP ベースのデータ交換
  - データベース・インベントリへのアクセスおよび商業取引とフローの統合
  - Oracle *i*Procurement の使用など、セルフサービス調達
  - Oracle Discoverer 3*i* Viewer を使用したデータ・マイニングとレポート
  - Oracle Exchange とアプリケーション
  - Phone Number Portability
- XML を使用したコンテンツおよびドキュメント管理
  - パーソナライズされた公開およびポータル
  - カスタマイズされた表示、Dynamic News の事例、Portal-to-Go および Flight Finder

Oracle 開発ツールおよびフレームワーク

XML アプリケーションの構築には、XSQL Servlet、XSQL Pages、Oracle Internet File System（Oracle 9*i*FS）、JDeveloper、Business Components for Java（BC4J）、Oracle Portal（WebDB）、Oracle9*i*AS Reports Services および Oracle9*i* Dynamic Services をすべて使用できます。

注意： XSQL Servlet および XSQL Pages は、Oracle XDK for Java の一部です。

データベースと中間層

XML アプリケーションは、データベースに配置するか、あるいは Oracle9*i* Application Server、Apache Server または他の Java 対応 Web サーバーなどの中間層に配置できます。

データベースに格納されるデータ

データは、オブジェクト・ビューを利用するリレーショナル表として格納されるか、あるいは XML 文書として `XMLType` および CLOB 列に格納されます。Oracle Text（*inter*Media Text）を使用すると、`XMLType` または CLOB 列に格納された XML 文書を効率的に検索できます。

Oracle9*i* からの XML データの格納および取出し

XML は Web でのデータ交換の標準として登場しました。Oracle9*i* では XML を次の形式で格納、検索および取り出す XML 対応機能が提供されています。

- **分解済み XML 文書として。**XML 文書が構成フラグメント単位で格納される場合です。この場合、XML データはオブジェクト・リレーショナル形式で格納されます。XML SQL Utility（XSU）または SQL のファンクションおよびパッケージを使用すると、このようなオブジェクト・リレーショナル・インスタンスから（構成済みの）XML 文書全体を生成できます。

  また、XSU または `Extract()` や `TABLE` などの SQL 関数を使用して、XML をオブジェクト・リレーショナル（分解済み）形式に戻すこともできます。

- **構成済み、つまり XML 文書全体として。**XML データは `XMLType` または CLOB/BLOB 列に格納されます。このようなドキュメントの検索には、`Extract()` や `ExistsNode()` などの `XMLType` ファンクション、あるいは Oracle Text の索引付けを使用します。

データベース内の XML サポート

Oracle9*i* では、次の XML サポートが提供されています。

- **XML 文書の生成。**Oracle9*i* では、クライアント側またはサーバー側での XML 生成がサポートされます。既存のオブジェクト・リレーショナル・データを使用して、対応する XML を生成できます。XML は、問合せ結果から生成する方法と、XSU（XML SQL Utility）Java または PL/SQL API を使用して SQL 問合せ自体の一部として生成する方法があります。

  このリリースの Oracle では、サーバー側の XML サポートが拡張されています。

  – XML の生成および集計用の新規 SQL 関数の提供

  – サーバーにリンクされた XML SQL Utility の C バージョンの提供

- **XML 文書の格納、問合せおよび取出し。**以前のリリースでは、XML 文書の格納、問合せおよび取出しに XSU などを使用していました。このリリースでは、新しいデータ型である XMLType を使用できます。

  `XMLType` では、XML 文書はキャラクタ・ラージ・オブジェクト（Character Large Object: CLOB）として格納されます。Oracle Text（*inter*Media Text）の索引付けを使用すると、`XMLType` 列に索引を付け、`CONTAINS` 演算子と XPath に似た構文を使用して問合せできます。XMLType では、XML 文書からフラグメントを抽出するメンバー関数もサポートされます。

  関連項目： 『Oracle9*i* アプリケーション開発者ガイド - XML』の「XML に対するデータベース・サポート」の「XMLType 列の索引付け」

XML と URI のデータ型

Oracle9*i* には、XML データと URI データを処理できる新しい型が用意されています。Extensible Markup Language（XML）は、データが構造化されているかどうかに関係なく Web 上で表示するために、W3C（World Wide Web Consortium）によって開発された標準形式です。

Web ページなどのリソースは、Web 上のどこにあっても Universal Resource Identifier（URI）を使用して識別されます。Oracle9*i* では、データベース自体に格納されているデータにアクセスできるように、URI の新しいクラス DBURI 参照が提供されています。また、データベースに外部と内部の URI を格納してアクセスできるように、新しい型セットも提供されています。

XMLType

オラクル社が提供する型である `XMLType` を使用すると、XML データをデータベースに格納して問合せできます。`XMLType` には、XPath 式を使用して XML データにアクセスし、データを抽出して問い合せるためのメンバー関数があります。XPath も、XML 文書を横断するために W3C の委員会によって開発された標準です。Oracle9*i* の `XMLType` ファンクションでサ

ポートされるのは、XPath 式の限られたサブセットのみです。また、これらの XMLType 値を既存のリレーショナル・データやオブジェクト・リレーショナル・データから作成できるように、SYS_XMLGEN、SYS_XMLAGG などの SQL ファンクションのセットと他の PL/SQL パッケージ（DBMS_XMLGEN）が提供されています。

関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』

- 「XML に対するデータベース・サポート」
- 「Oracle Text を使用した XML データの検索」
- 「Oracle AQ を使用した XML データの交換」
- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』または Oracle アドバンスト・キューイングでの XMLType の使用方法に関する情報

URI のデータ型

Oracle9*i* では、次の Uri 型のファミリが提供されています。

- UriType
- DBUriType
- HttpUriType

この 3 つの型は、継承階層で関連付けられています。UriType は抽象型で、DBUriType と HttpUriType はこの型のサブタイプです。

- HttpUriType を使用すると、外部の Web ページやファイルへの URL を格納できます。これらのファイルにはハイパー・テキスト転送プロトコル（Hypertext Transfer Protocol: HTTP）を使用してアクセスします。
- DBUriType を使用すると、データベース内のデータを参照する DBURI 参照を格納できます。

UriType はスーパータイプであるため、この型の列を作成し、そこに DBUriType 型または HttpUriType 型のインスタンスを格納できます。これにより、データベース内外に格納されたデータを参照し、一貫した方法でアクセスできます。DBURI 参照では、XPath に似た表現を使用してデータベース内のデータが参照されます。このモデルを使用すると、CLOB 列や他の列に格納されているデータを参照し、それを URL として外部に公開できます。Oracle9*i* では、標準的なサーブレットが提供されており、URL などを解釈できる Oracle Servlet Engine の下にインストールして実行できます。

関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「DBURI 参照」

拡張性と XML

Oracle の拡張性により、セクション検索用の Oracle Text 索引、XML を処理する特別な演算子、XML の集計、XML が関係する問合せの特別な最適化など、XML に対する特別な索引付けを使用できます。

Oracle Text の検索機能

LOB に格納された XML テキストに、拡張索引付けインタフェースを使用して索引を付けることができます。Oracle9*i* では、XML テキスト内で部分文字列の一致を検索できるように、`CONTAINS` や `WITHIN` などの演算子が提供されています。

> **関連項目：** 『Oracle9*i* アプリケーション開発者ガイド - XML』の「Oracle Text を使用した XML データの検索」

Oracle ベースの XML アプリケーション

インターネット・アプリケーションでは、XML に様々な用途が考えられます。このマニュアルでは、Oracle の XML コンポーネントが適している次の 2 つのデータベース中心のアプリケーションの分野について重点的に説明します。

コンテンツおよびドキュメントの管理

コンテンツおよびドキュメントの管理には、データ表現のカスタマイズが含まれます。このようなアプリケーションでは、通常、ほとんど作成済みの XML 文書が処理されます。このマニュアルでは、複数の事例について説明します。

> **関連項目：** このマニュアルのコンテンツおよびドキュメントの管理に関する次の章を参照してください。
>
> - 「XML を使用したコンテンツのカスタマイズ : Dynamic News アプリケーション」
> - 「Oracle9iAS Wireless Edition と XML」
> - 「XML および XSQL を使用した表示のカスタマイズ : Flight Finder」

Business-to-Business（B2B）または Business-to-Consumer（B2C）メッセージ機能

B2B および B2C メッセージ機能では、データがビジネス・アプリケーション間で交換されます。通常、このようなアプリケーションでは、生成済みの XML 文書、あるいは生成済み XML 文書と構成済み XML 文書の組合せが処理されます。

関連項目： このマニュアルの次の章を参照してください。

- 「XSL を使用した Discoverer 4i Viewer のカスタマイズ」
- 「Service Delivery Platform（SDP）と XML」
- 「オンライン B2B XML アプリケーション : 手順」

このマニュアルの後続の章では、Oracle XML コンポーネントおよび Oracle 開発ツールの使用方法と、これらのツールを使用して Web ベースのデータベース・アプリケーションを構築する方法について説明します。

関連項目： 『Oracle9*i* アプリケーション開発者ガイド - XML』の「Oracle AQ を使用した XML データの交換」「OE での XML の使用」

Oracle XML 対応テクノロジ・コンポーネントおよび機能

Oracle9*i* は、XML データベース・アプリケーションの構築に適しています。Oracle XML 対応テクノロジの機能は、次のとおりです。

- Oracle Text（interMedia Text）を使用した XML 文書の索引付けと検索
- メッセージング・ハブおよび中間層コンポーネント
- バックエンド、データベース、フロントエンドの統合の問題
- Oracle XDK が提供する最も一般的な 2 つの API: DOM および SAX
- Oracle の統合ツールおよびコンポーネントのパッケージ
- Oracle XML のサンプルとデモ

Oracle Text（*inter*Media Text）を使用した XML 文書の索引付けと検索

Oracle Text（*inter*Media Text）では、CLOB および他のドキュメントに格納された XML 用の強力な検索および取出しオプションが提供されています。このオプションを使用すると、表の 1 列に格納されている、それぞれ 4GB 以内の XML 文書およびドキュメント・セクションに索引を付けて検索できます。

Oracle Text の XML 文書検索には、階層要素のコンテナ関係、ドキュメント・タイプの識別および XML 属性の検索が含まれます。このような XML 文書検索は、標準的な SQL 問合せ述語や、他の強力な字句検索および全文検索オプションと併用できます。

データベースのテキストの CLOB に保存された XML 文書やドキュメント・セクションには、Oracle Text のテキスト検索エンジンで索引を付けることができます。開発者は特定の XML 階層内でデータの絞り込み検索を行ったり、XML 要素の属性内で名前と値のペアを検索できます。

Oracle Text はデータベースおよび SQL 言語とシームレスに統合されているため、開発者は簡単に SQL を使用して、構造化データと索引付きドキュメント・セクションの両方が関連する問合せを実行できます。

関連項目：『Oracle Text リファレンス』

メッセージング・ハブおよび中間層コンポーネント

Oracle XML には、次のコンポーネントも含まれています。

- **XML 対応メッセージング・ハブ**。この種のハブは、Oracle 以外のシステムとのインタフェースが必要な B2B アプリケーションに不可欠です。
- **中間層システム**。XML 対応アプリケーション、Web または統合サーバー。Oracle9*i* Application Server などです。

Oracle JVM（Java Virtual Machine: Java VM）

Oracle マルチスレッド・サーバー（MTS）・アーキテクチャに基づいて構築された Oracle JVM は、Java 1.2 準拠の仮想マシンであり、データ・サーバーによりメモリー・アドレス空間が共有されます。機能は次のとおりです。

- Java および XML 処理コードはメモリー内で実行されるため、標準の JDBC インタフェースを使用してデータ・アクセスが高速化されます。
- システム固有の方法で Java バイト・コードがコンパイルされ、サーバー側の Java のパフォーマンスが向上し、数千の同時ユーザーへと線形的に拡張できます。Oracle XDK コンポーネントは事前にロードされ、システム固有の方法でコンパイルされます。

Oracle JVM では、システム固有の CORBA および EJB 標準と、Java ストアド・プロシージャがサポートされるため、Java を SQL および PL/SQL スクリプトと簡単に統合できます。

Oracle9*i* Application Server

Oracle9*i* Application Server（Oracle9*i*AS）は、インターネットとイントラネットの Web アプリケーションにサービスを提供します。Oracle9*i* と統合され、データ・キャッシュや Oracle Portal などの拡張サービスを提供します。他にも、Oracle Advanced Queueing、Oracle Message Broker、Oracle Workflow、Oracle9*i* Reports Services、Dynamic Services などのサービスも提供されています。

関連項目： http://otn.oracle.co.jp/products/

バックエンド、データベース、フロントエンドの統合の問題

開発上の主要な問題は、複数ベンダーのバックエンドの ERP および CRM システムを、サプライ・チェーン内のパートナのシステムや、カスタマイズされたデータ・ウェアハウスと統合することです。

様々なベンダーのリレーショナル・データベースやオブジェクト・リレーショナル・データベース間でのデータ交換は、XML を使用することで単純化されます。XML および AQ を使用したデータ交換の実装例の 1 つが第 8 章に提供されています。

Oracle XML テクノロジと Oracle XML 対応ツール、インタフェースおよびサーバーにより、ほとんどのデータとアプリケーションの統合という問題に対処する構築ブロックを提供されます。

高パフォーマンスの実装

このような構築ブロックを使用することにより、次の理由で高パフォーマンスの実装が得られます。

- データベースのデータと XML の処理が同一サーバー上で行われるため、データ・アクセスのためのネットワークの通信量を削減できます。
- Oracle9*i* の問合せエンジンと Oracle JVM、Oracle9*i* Application Server または OIS の速度を活用することで、データ・アクセスがさらに高速になります。
- XDK for C コンポーネントを使用すると、システム固有の XML 機能とパフォーマンスの向上が得られます。

これにより、開発者は XML ベースの Web ソリューションを構築し、Java データベースのデータと機能を様々な方法で統合できます。

Oracle XDK が提供する最も一般的な 2 つの API: DOM および SAX

Oracle XDK は、Java、C、C++ および PL/SQL という 4 つの言語で実装されます。Java バージョンは、Oracle JVM（Java VM）で直接動作します。XML 1.0 仕様がサポートされ、検証用パーサーまたは非検証用パーサーとして使用されます。

このパーサーでは、開発者が XML 文書の処理に必要とする最も一般的な 2 つの API が提供されています。

- **DOM 1.0 および 2.0:** W3C 勧告によるドキュメント・オブジェクト・モデル（Document Object Model: DOM）・インタフェースです。この API を使用すると、解析対象となるドキュメントの要素の内容に標準的な方法でアクセスして編集できます。
- **SAX 1.0 および 2.0:** Simple API for XML インタフェースです。

詳細は、このマニュアルの後続の章および『Oracle9*i* アプリケーション開発者ガイド - XML』を参照してください。

カスタムの XML アプリケーションの作成

Oracle9*i* 環境では、XML 文書を処理するカスタム・アプリケーションを作成する作業を簡素化できます。これにより、業界標準に準拠した移植性のあるアプリケーションとコンポーネントを選択した言語で記述し、必要な層に配置できます。

XML Parser は、Oracle9*i* が移植されるすべてのオペレーティング・システムで、Oracle9*i* プラットフォームの一部となります。

Oracle XML Parser は、PL/SQL でも実装されます。既存の PL/SQL アプリケーションは、Oracle XML テクノロジを活用するように拡張できます。

Oracle の統合ツールおよびコンポーネントのパッケージ

Oracle9*i* では、E-Business アプリケーションの構築用ツールおよびコンポーネントの統合パッケージが提供されています。

- Oracle JDeveloper および Oracle Business Components for Java（BC4J）
- Oracle Internet File System（Oracle 9iFS または 9iFS）
- Oracle Portal
- Oracle Exchange

このツール・パッケージを使用すると、アプリケーション開発のためのデータやドキュメント・オブジェクトの交換が簡素化され、複数のシリアル化が不要になることが保証されます。

Oracle JDeveloper および Oracle Business Components for Java（BC4J）

Oracle JDeveloper は、Oracle9*i* 上で Java および XML を活用するアプリケーションを構築、配置およびデバッグするための統合環境です。この環境では、CORBA、EJB および Java ストアド・プロシージャによる Java 1.1 または 1.2 での作業が簡素化されます。この環境では、次の作業が可能です。

- Oracle XML コンポーネントへの直接アクセスによる複数層アプリケーションの構築
- XML 情報を処理する Java サーブレットの迅速な作成およびデバッグ
- JDeveloper および BC4J コンポーネントを使用した、移植性の高いアプリケーション・ロジックの構築

Oracle JDeveloper を使用して構築されたアプリケーションの例を次に示します。

- Self-Service Web-Expensing を含む *i*Procurement（Self Service Applications）。
- PDA のコンテンツ配信。第 3 章「Oracle9iAS Wireless Edition と XML」を参照してください。

- オンライン・マーケットプレース。
- XML および AQ メッセージ機能を使用したリテーラ / サプライヤ間の取引。第 8 章「オンライン B2B XML アプリケーション : 手順」を参照してください。

JDeveloper を使用して XML アプリケーションを構築する方法の詳細は、『Oracle9*i* アプリケーション開発者ガイド - XML』の「JDeveloper を使用した Oracle の XML アプリケーションの作成」を参照してください。

Oracle Business Components for Java（BC4J） Java 対応のビジネス・コンポーネント（BC4J）は、ビジネス・ロジックを再利用可能な Java コンポーネント・ライブラリにカプセル化し、柔軟な SQL ベースの情報ビューを通じて再利用するための、Oracle9*i* アプリケーションのフレームワークです。

注意： Oracle JDeveloper と BC4J は、Oracle9*i* には組み込まれておらず、BC4J ランタイムのみが組み込まれています。

Oracle Internet File System（Oracle 9*i*FS または 9*i*FS）

Oracle Internet File System（Oracle 9*i*FS）へのアクセスにより、SMB、HTTP、FTP、SMTP および IMAP4 など、標準的な Windows およびインターネット・プロトコル経由でファイルおよびフォルダ・ベースのモデルを使用して、ドキュメントとデータを簡単に編成し、アクセスできます。

Oracle 9*i*FS により、Web ベース・アプリケーションの構築と管理が簡単になります。これは Java 用のアプリケーション・インタフェースであり、PowerPoint ファイルなどのドキュメントを Oracle 9*i* にロードし、Oracle9*i* Application Server や Apache Web Server などの Web サーバーからのドキュメントを表示できます。

> **関連項目：** 第 6 章「Oracle Internet File System を使用した XML アプリケーションの作成」

Oracle 9*i*FS を使用すると、開発者による XML での作業が簡素化されます。この場合、Oracle 9*i*FS は XML のリポジトリとして機能します。Oracle 9*i*FS では、XML が自動的に解析され、内容が表と列に格納されます。Oracle 9*i*FS は、ファイルが要求されると内容を変換し、Web 上などで選択情報を配信します。

詳細は、`http://otn.oracle.com/products/ifs/` を参照してください。

Oracle Portal

Oracle Portal では、XML ベースの Rich Site Summary（RSS）形式のドキュメント入力や、情報と XSL スタイルシートのマージなどができます。結果はブラウザに表示できます。この設計により、情報の表現と情報自体が効率的に分離され、データの整合性を損なわずに表示を簡単にカスタマイズできます。

Oracle Portal は、エンタープライズ・ポータル、つまり、E-Business 用の Web サイトを構築して配置するためのソフトウェアです。ブラウザ・インタフェースにより、各ユーザーが必要とするビジネス情報、Web コンテンツおよびアプリケーションが整理された形式で個別に表示されます。これには、WebDB 2.2 のサイト構築およびセルフサービス Web パブリッシング機能が含まれ、Single Sign-On、パーソナライズ機能およびコンテンツ分類などの新しいエンタープライズ・ポータル機能が追加されます。Oracle Portal では Oracle9*i* が使用され、Oracle 9*i* Application Server とともに配置およびパッケージ化されます。

ポートレット： ポートレットは、Web ベース・リソースへのアクセスを提供する再使用可能なインタフェース・コンポーネントです。ポートレット経由で Web ページ、アプリケーション、ビジネス・インテリジェンス・レポート、シンジケート・コンテンツ・フィード、管理対象ソフトウェア・サービスまたは他のリソースにアクセスでき、Oracle Portal のサービスとしてパーソナライズし、管理できます。企業は、独自のポートレットを作成するか、サード・パーティのポートレット・プロバイダによるポートレットを選択できます。Oracle では、開発者が PL/SQL、Java、HTML または XML を使用してポートレットを簡単に作成できるように、Portal Developer's Kit（PDK）が提供されています。

> **関連項目：** Oracle Portal の PDF および URL サービスの概要は、『Oracle9*i* アプリケーション開発者ガイド - XML』の「PDK を使用した Oracle Portal での XML データのビジュアル化」を参照してください。

Oracle Exchange

Oracle Exchange プラットフォームは Oracle9*i* ベースです。業界や企業のサプライ・チェーン全体をサポートするために必要なビジネス・トランザクションがすべて用意されています。Oracle Exchange は Oracle E-Business Suite に基づいており、このパッケージでは顧客候補からの初めての連絡、製造計画の作成と実施、販売後のサービスやサポートなど、サプライ・チェーン全体がサポートされます。

Oracle Exchange では、データ交換形式およびメッセージのペイロードとしての XML と、アドバンスト・キューイングが使用されます。

XML Gateway

XML Gateway は一連のサービスで、Oracle E-Business Suite と簡単に統合でき、ビジネス・イベントによりトリガーされる XML メッセージを作成および消費できます。また、XML Gateway は Oracle Advanced Queuing と統合され、メッセージをエンキューまたはデキューします。その後、メッセージは Oracle Message Broker などのメッセージ・トランスポート・サービスを通じてビジネス・パートナとの間で相互に通信が行われます。

メタデータ API

メタデータ API は、次の作業を実行するための単純で柔軟な一元化された手段を提供します。

- XML または作成 DDL としてのデータベース・オブジェクト（メタデータ）の定義全体の抽出
- 業界標準の XSLT（XML Stylesheet Transformation language）を介したメタデータの変換
- SQL DDL の生成によるデータベース・オブジェクトの再作成

メタデータ API は、インスタンスが稼働していれば Oracle9*i* で常に使用できます。Oracle Lite では使用できません。

> **関連項目：**『Oracle9*i* アプリケーション開発者ガイド - XML』の「メタデータ API の使用」

XML 関連のその他の取組み

前述のツールの他にも、次のような取組みが進行中です。

XML Metadata Interchange（XMI）：ツールおよびデータ・ウェアハウスのメタデータの管理と共有

オラクル社、IBM 社および Unisys 社は、XML Metadata Interchange（XMI）仕様のサポートを提案しています。これにより、オラクル社や他社のアプリケーション開発ツールとデータ・ウェアハウス・ツールが共通のメタデータを交換でき、アプリケーションやウェアハウスの設計を変更せずに必要なツールを選択できることが保証されます。

Oracle Advanced Queueing の XML サポート：信頼性の高い非同期メッセージ機能のためのインターネットの使用

Oracle Advanced Queuing（AQ）では、XML 文書、ドキュメント・セクション、またはペイロードとしてのフラグメントを含むメッセージなど、非同期メッセージを保護 HTTP 経由で確実に伝播できるようになりました。このため、動的な取引が可能になり、企業間やエージェント間のリンクの確立に伴う遅延や初期コストが不要になります。

> **関連項目：**『Oracle9*i* アプリケーション開発者ガイド - XML』の「Oracle AQ を使用した XML データの交換」

Oracle XML のサンプルとデモ

このマニュアルには、Oracle XML コンポーネントの使用例が記載されています。例が 1 つのスキーマに準拠しているわけではありません。例がダウンロードできる場合や、`$ORACLE_HOME/rdbms/demo` または `$ORACLE_HOME/xdk/.../sample` に用意されている場合は、そのことが記載されています。

Oracle XML コンポーネントの実行要件

Oracle8*i* 以上の場合は、Java や XML などのインターネット標準に固有のサポートが組み込まれています。Oracle XML コンポーネントと、それを使用して構築されたアプリケーションは、組込み JavaVM である Oracle JVM を使用してデータベース内で実行できます。

より小型のデータベースを必要とするデバイスやアプリケーションの場合、XML データの格納と取出しには Oracle Lite を使用します。

Oracle XML コンポーネントは、`http://otn.oracle.com/tech/xml` から無償でダウンロードできます。

XDK の要件

XDK for Java および XDK for PL/SQL の要件は、次のとおりです。

- XDK for Java には、JDK/JRE 1.1 以上の VM for Java が必要です。
- XDK for PL/SQL には、Oracle8.x 以上または PL/SQL カートリッジが必要です。

要件については、『Oracle9*i* アプリケーション開発者ガイド - XML』の XDK に関する章、第 19 ～ 29 章および付録 C ～ G も参照してください。

Oracle9*i* データベースおよび Oracle9*i* Application Server に含まれている XML コンポーネント

表 1-1 に、Oracle9*i* データベースおよび Oracle9*i* Application Server（Oracle9*i*AS）に含まれている XDK コンポーネントのバージョンを示します。

表 1-1 Oracle9*i* および Oracle9*i*AS の XDK コンポーネントで提供されるバージョン

XDK コンポーネント	Oracle9*i* データベース リリース 1（9.0.1）	Oracle9*i*AS リリース ー 予備バージョン 特に明記しない場合のみ
XDK for Java		
XML Parser for Java および XSLT Processor	9.0.1.0.0	9.0.1.0.0
XML Schema Processor for Java	9.0.1.0.0	9.0.1.0.0
XML Class Generator for Java	9.0.1.0.0	9.0.1.0.0
XSQL Servlet	9.0.1.0.0	9.0.1.0.0
XML SQL Utility（XSU）for Java	9.0.1.0.0	9.0.1.0.0
XDK for JavaBeans		
XML Transviewer Beans	9.0.1.0.0	9.0.1.0.0
XDK for C		
XML Parser for C および XSLT Processor	9.0.1.0.0	9.0.1.0.0
XML Schema Processor for C	9.0.1.0.0	9.0.1.0.0
XDK for C++		
XML Parser for C++ および XSLT Processor	9.0.1.0.0	9.0.1.0.0
XML Schema Processor for C++	9.0.1.0.0	9.0.1.0.0
XML Class Generator for C++	9.0.1.0.0	9.0.1.0.0
XDK for PL/SQL		
XML Parser for PL/SQL および XSLT Processor	9.0.1.0.0	9.0.1.0.0
XML SQL Utility（XSU）for PL/SQL	9.0.1.0.0	9.0.1.0.0

XML テクニカル・サポート

Oracle XML 対応テクノロジに関するテクニカル・サポートは、オラクル社カスタマ・サポート・センターなどの通常のサポート・チャネルの他に、Oracle Technology Network Japan（OTN Japan）の「フォーラム」オプションを通じて無償で利用できます。

```
http://otn.oracle.co.jp
```

OTN Japan の登録ユーザーでなくても、OTN Japan の「フォーラム」を通じて XML 関連の質問を転送したり、質問に対して回答できます。OTN Japan のフォーラムを利用する手順は、次のとおりです。

1. OTN Japan サイトの左側のナビゲーション・バーで「フォーラム」を選択します。
2. 「テクノロジー」セクションにスクロール・ダウンします。「XML」を選択します。
3. 質問、コメントなどを転送します。

OTN Japan からの最新ソフトウェアのダウンロード

次の URL で OTN Japan にアクセスし、Oracle XML コンポーネントの最新情報をダウンロードできます。

```
http://otn.oracle.co.jp/software/tech/xdk/xdk.html
```

2

Oracle XML アプリケーションのモデリングおよび設計問題

この章の内容は、次のとおりです。

生成される XML または構成済み XML として格納可能な XML データ

XML データを Oracle9*i* に格納する方法は、次のとおりです。

- **生成される XML**。この場合、XML データはデータベース内のオブジェクト・リレーショナル表またはビューにまたがって格納されます。このデータを必要に応じて動的に生成し、元の XML 形式に戻すことができます。
- **構成済み（作成済みまたはネイティブ）XML**。この場合、XML 文書はそのまま CLOB に格納されます。

生成される XML

XML をオブジェクト・リレーショナル表およびビューから生成できます。ここでは、オブジェクト・リレーショナル表およびビューを使用するメリットを、純粋なリレーショナル構造を使用する場合と比較して説明します。

生成される XML が使用されるのは、XML が交換フォーマットになっており、既存のビジネス・データが XML 構造（タグ）で囲まれている場合です。これは、データベースにおける XML の最も一般的な使用方法です。この場合、XML は交換処理自体にのみ一時的に使用されます。

生成される XML の例

生成される XML 文書の例には、注文書や請求書、フライト・スケジュールなどがあります。

Oracle にはオブジェクト・リレーショナル拡張機能があり、オブジェクト型、オブジェクト参照およびコレクションを使用してデータベース内のデータの構造を獲得できます。XML データの構造をオブジェクト・リレーショナル形式で格納および保持するには、次の 2 つのオプションがあります。

- 各要素の属性をリレーショナル表に格納し、XML 要素の構造を獲得するオブジェクト・ビューを定義します。
- 構造化された XML 要素をオブジェクト表に格納します。

オブジェクト・リレーショナル形式で格納されたデータは、必要に応じて SQL を使用し、簡単に更新、問合せ、再配置および再フォーマットできます。

オブジェクト・リレーショナル形式による生成される XML 文書の格納

複合 XML 文書をオブジェクト・リレーショナル・インスタンスとして格納し、効率的に索引を付けることができます。この種のインスタンスでは、XML のネストが完全に表現され、XML のセマンティクスがリストされます。Oracle の拡張インフラストラクチャでは、XML 文書を高速で検索できるように、パス索引など、新しいタイプの索引を作成できます。

XML SQL Utility（XSU）による XML の格納および SQL 問合せ結果から XML への変換

XML SQL Utility（XSU）では、XML 文書を、基礎となるオブジェクト・リレーショナル記憶域にマップして格納する手段が提供されており、逆にオブジェクト・リレーショナル・データを XML 文書として取り出すこともできます。

XSU では、問合せの別名や列名が要素のタグ名にマップされ、オブジェクト型のネストが保たれて、SQL 問合せの結果が XML に変換されます。変換結果には、テキスト形式とドキュメント・オブジェクト・モデル（Document Object Model: DOM）ツリー形式があります。後者を生成すると、テキスト解析によるオーバーヘッドを回避して、DOM ツリーを直接実現できます。

関連項目：『Oracle9*i* アプリケーション開発者ガイド - XML』の「XSU」

構成済み（作成済みまたはネイティブ）XML

Oracle8*i* 以上では、ラージ・オブジェクト（Large Object: LOB）をキャラクタ・ラージ・オブジェクト（Character Large Object: CLOB）、バイナリ・ラージ・オブジェクト（Binary Large Object: BLOB）または外部に格納されるバイナリ・ファイル（BFILE）として格納できます。LOB は、構成済み（作成済みまたはネイティブ）XML 文書の格納に使用します。

CLOB または BFILE への構成済み XML データの格納

受信する XML 文書が特定の構造に準拠していない場合は、CLOB に格納する方がよいことがあります。たとえば、XML メッセージング環境では、キューにある各 XML メッセージの構造が異なる場合があります。

CLOB には大きいキャラクタ・データが格納され、構成済み XML 文書の格納に役立ちます。

BFILE は外部ファイル参照であり、XML 文書の格納に使用することもできますが、アクセス頻度の低いマルチメディア・データに使用する方が役立ちます。この場合、XML は Oracle 外部で格納および管理されますが、サーバー側で問合せに使用できます。ドキュメントのメタデータは、索引付けとアクセスが高速になるように、サーバーのオブジェクト・リレーショナル表に格納できます。

XML 文書をそのまま CLOB または BLOB に格納する方法は、XML 文書に静的コンテンツが含まれており、ドキュメント全体の置換によってのみ更新される場合に適しています。

- 構成済み XML の例には、記事、広告、書籍、法的契約書などがあります。これらのドキュメントはドキュメント・セントリックと呼ばれ、ドキュメント全体がデータベースから配信されます。これらのドキュメントをそのまま Oracle に格納すると、実績あるデータベースのメリットとそのファイル・システム記憶域以上の信頼性を得られます。

- データベース外部への格納。XML 文書をデータベース外部に格納するように選択した場合も、Oracle 機能を使用して BFILE、URL およびテキスト・ベースの索引付けを行うと、ドキュメントの索引付け、問合せおよび効率的な取出しができます。

Oracle Text（*inter*Media Text）の索引付けによる XML 要素のコンテンツのファイングレイン検索

Oracle では、外部ドキュメントを指す URL のみでなく、LOB 列の Oracle Text（*inter*Media Text）索引を作成できます。この索引付けメカニズムは、XML データにも機能します。

Oracle8*i* および Oracle9*i* では、XML 要素のコンテンツ内で XML タグ、セクションおよびサブセクションのテキスト検索が認識されます。その結果、構造化されていないデータを対象とし、ドキュメント内の特定のセクションまたは要素に限定して問合せできます。

Oracle Text の例 : CONTAINS を使用したテキストおよび XML データの検索

次の Oracle Text（*inter*Media Text）の例は、すでに適切な索引を作成済みであることを想定しています。

```
SELECT *
FROM   purchaseXMLTab
WHERE  CONTAINS(po_xml,"street WITHIN addr") >= 1;
```

> **関連項目：** Oracle Text の詳細は、『Oracle9*i* アプリケーション開発者ガイド - XML』の「Oracle Text を使用した XML データの検索」を参照してください。

構成済み（作成済み）XML の格納のメリット

XML 文書の構造が不明または動的な場合は、CLOB 記憶域が最適です。

構成済み XML の格納のデメリット

オブジェクト・リレーショナル列に対するほとんどの SQL 機能は使用できません。更新など、特定の操作の並行性が低下する場合があります。ただし、このドキュメントの正確なコピーが保持されます。

ハイブリッドな XML 格納方法を使用したマッピングの細分化の改善

前項で説明した内容は、次のとおりです。

- 構造化された XML 文書（生成される XML）をオブジェクト・リレーショナル・インスタンスにマップする方法
- 構成済み XML 文書（作成済み）を LOB に格納する方法

ただし、多くの場合は、マッピングの最小単位をより厳密に制御する必要があります。

たとえば、書籍などのテキスト・ドキュメントを XML 形式でマッピングする場合に、個々の要素を 1 つずつ拡張し、オブジェクト・リレーショナル形式で格納するとは限りません。この種のドキュメントのフォントおよび段落情報をオブジェクト・リレーショナル形式で格納すると、問合せには役立たない場合があります。

一方、テキスト・ドキュメント全体を CLOB に格納すると、ドキュメント全体に対する SQL 問合せの効率が悪くなります。

ユーザー定義の格納の細分化を可能にするハイブリッドなアプローチ

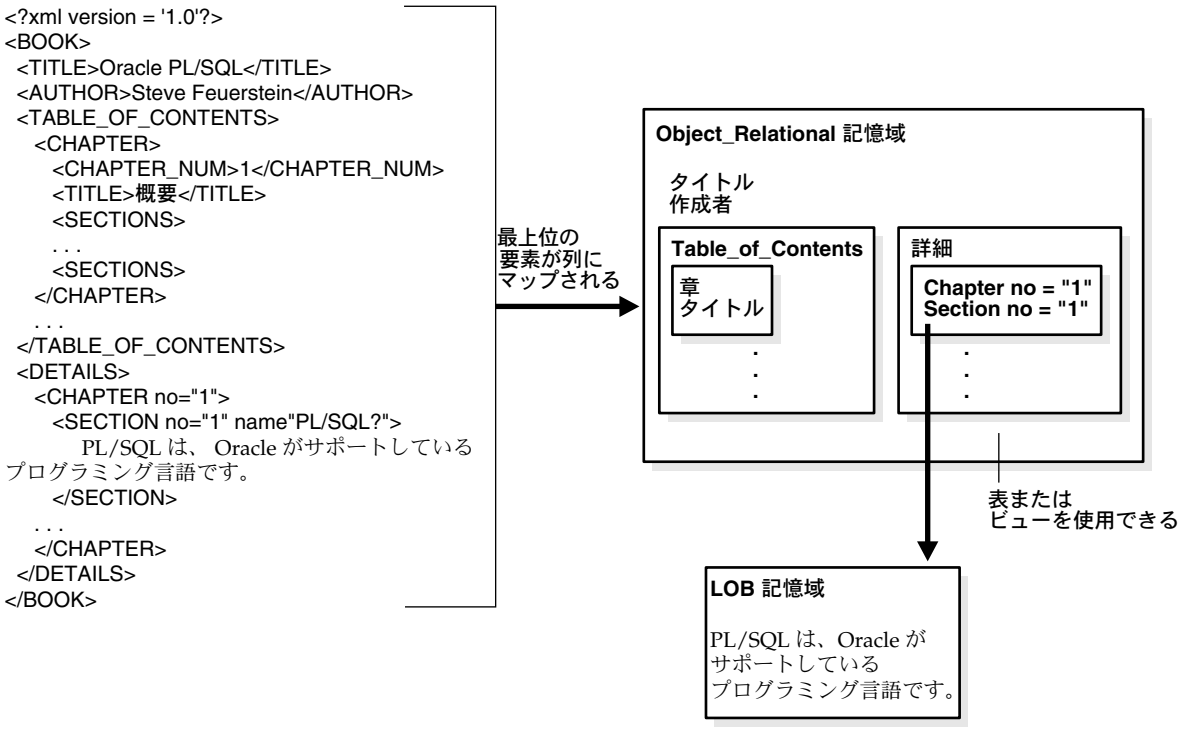
代替策は、この種の格納にはユーザー定義の細分化を使用することです。書籍の例では、次の作業が可能です。

- 章、項、タイトルなど、最上位の要素の問合せ。このような要素は、オブジェクト・リレーショナル表に格納できます。
- 書籍の各項の内容の問合せ。これらの項は CLOB に格納できます。

マッピングの最小単位は、表の定義時に指定できます。サーバーでは、各種のソースから XML を自動的に構成し、それに応じて問合せを生成できます。

図 2-1 に、このハイブリッド XML 記憶域のアプローチを示します。

図 2-1 ハイブリッド XML 記憶域のアプローチ : コンテンツが CLOB に格納されている場合のトップレベル要素の問合せ



ハイブリッドな格納のメリット

XML 文書をハイブリッドに格納するアプローチのメリットは、次のとおりです。

- ドキュメント全体を分解せずに、有効で問合せ可能な情報をオブジェクト・リレーショナル形式で柔軟に格納できます。
- ドキュメント全体が分割されないため、ドキュメントを再構成する時間を節約できます。
- LOB に格納されたドキュメント部分でテキスト検索ができます。

生成される XML の変換

データベースから生成される XML は標準的な形式になっており、各列は要素にマップされ、各オブジェクト型はネストされた要素にマップされています。ただし、アプリケーションでは状況に応じて異なる XML 文書表現を必要とする場合があります。

XML 文書の構造変換が必要な場合

XML 文書が構造化されており、その構造に、基礎となるデータベース・スキーマ構造との互換性がない場合は、データをデータベースに書き込む前に適切な形式に変換する必要があります。次の 3 つの方法があります。

- XSL スタイルシートや他のプログラミング・アプローチを使用します。
- データ中心の XML 文書をそのまま単一オブジェクトとして格納します。
- 各種の XML 文書構造に対応するオブジェクト・ビューを定義し、適切な変換を実行してベース・データを更新する INSTEAD-OF トリガーを定義します。

ビューを使用した XML 文書とデータの結合

構造化された XML データと構造化されていない XML データが組み合されていて、全体を表示および操作する必要がある場合は、Oracle のビューを使用できます。

ビューを使用すると、様々な方法で格納されている XML データを組み合せ、処理中にオブジェクトを作成できます。次の操作を実行できます。

- 従業員データや顧客データのように構造化されたデータを、オブジェクト・リレーショナル表の 1 箇所に格納します。
- 説明やコメントなど、関連付けられているが構造化されていないデータを CLOB に格納します。

データ全体を取り出す必要がある場合は、ビューの SELECT 文で型コンストラクタを使用して、データの各部分から構造を簡単に作成できます。これにより、XML SQL Utility では、ビューから作成されたデータを単一の XML 文書として取り出すことができます。

XSLT を使用した問合せ結果の変換

この操作では、オリジナルのドキュメントを問い合せて、結果をユーザーまたはアプリケーションが必要とする書式に変換します。たとえば、アプリケーションで WML を使用して携帯電話と対話している場合は、生成された XML を、WML または携帯電話との通信に適した他の類似する規格へ変換する必要がある場合があります。

そのためには、生成された XML 文書に XSLT 変換を適用します。XSLT 変換自体を生成フェーズの最適化とできます。データベース・サーバー内のスケーラブルな高性能 XSLT 変換エンジンでは、大量のデータを処理できます。

変換の索引付けおよび問合せ

XML 文書の変換後のビューで、索引の作成や問合せが必要になる場合があります。たとえば、XML メッセージング環境では、様々な形式の発注メッセージが存在する可能性があります。特定の問合せですべての発注メッセージを処理できるように、標準的な問合せを行う必要があります。

この場合、問合せはドキュメントの変換後のビューに適用されます。ファンクション索引を作成するか、標準ビューを使用できます。

索引付けの方法

`extract()` および `existsNode()` メンバー関数の固有の実装では、XML 文書が解析され、パス横断が実行されてからフラグメントが抽出されます。ただし、これはパフォーマンス強化や拡張性をもたらす解決策ではありません。

2 番目のアプローチは、Oracle Text（*inter*Media Text）の索引付けを使用することです。

関連項目：

- Oracle Text の詳細は、『Oracle9*i* アプリケーション開発者ガイド - XML』の「Oracle Text を使用した XML データの検索」を参照してください。
- 『Oracle Text リファレンス』

また、拡張索引付けインフラストラクチャを使用して、`XMLType` 列で独自の索引付けメカニズムを構築することもできます。

XML Schema およびドキュメントのマッピング

W3C は、現行の Document Type Definition（DTD）ベースのメカニズムを発展させ、構造化スキーマおよびデータ型に新しい XML ベースの表記法を提供するためのスキーマ・ワーキング・グループを設立しました。XML Schema の用途は、次のとおりです。

- XML Schema1: ドキュメント構造（要素、属性、ネームスペース）の制約
- XML Schema2: コンテンツ（データ型、エンティティ、表記法）の制約

データ型自体は、基本形（バイト、日付、整数、順序、間隔）でも、ユーザー定義（既存のデータ型から導出され、範囲、精度、長さ、マスクなど、ベース型の特定のプロパティを制約できるものなど）でもかまいません。アプリケーション固有の制約と記述が許容されます。

XML Schema は、要素、属性およびデータ型の定義が継承されます。構造体の意味を標準的な方法で明確に理解できるように、URI 参照のメカニズムが提供されています。また、埋込みドキュメントやコメント用のスキーマ言語も提供されています。

たとえば、次のように単純なデータ型を定義できます。

XML Schema の例 1: 単純なデータ型の定義

次の例に、XML Schema に単純なデータ型を定義する方法を示します。

```
<datatype name="positiveInteger"
            basetype="integer"/>
   <minExclusive> 0 </minExclusive>
</datatype>
```

この単純な例でも、XML Schema がベース型や最小値制約などの重要な多数の新しい構造体を提供することは明らかです。

動的データがデータベースから生成される場合、通常はデータベースの型システムで表されます。Oracle では、これは前述のオブジェクト・リレーショナル型システムです。この型システムは NULL、変数精度、NUMBER（7、2）、制約チェック、ユーザー定義型、継承、型間の参照、型のコレクションなど、データ型に関して豊富な機能を持っています。XML Schema ではスキーマ制約の多様な側面を獲得でき、生成されたドキュメントをデータの基礎となる型システムと一致させやすくなります。

XML Schema の例 2: XML Schema を使用した基礎となるスキーマへの生成された XML 文書のマップ

XML Schema で表される単純な発注書の型を考えてみます。

```
<type name="Address" >
   <element name="street" type="string" />
   <element name="city"   type="string" />
   <element name="state"  type="string" />
   <element name="zip"    type="string" />
</type>

<type name="Customer">
   <element name="custNo"
                    type="positiveInteger"/>
   <element name="custName" type="string" />
   <element name="custAddr" type="Address" />
</type>

<type name="Items">
   <element name="lineItem" minOccurs="0" maxOccurs="*">
    <type>
      <element name="lineItemNo" type="positiveInteger" />
      <element name="lineItemName" type="string" />
      <element name="lineItemPrice" type="number" />
      <element name="LineItemQuan">
```

```
        <datatype basetype="integer">
          <minExclusive>0</minExclusive>
        </datatype>
      </element>
    </type>
   </element>
</type>

<type name="PurchaseOrderType">
    <element name="purchaseNo"
                    type="positiveInteger" />
    <element name="purchaseDate"  type="date" />
    <element name="customer" type="Customer" />
    <element name="lineItemList"  type="Items" />
</type>
```

重要な点は、XML Schema で提案されている構造体と SQL: 1999 ベースの型システムが厳密に一致していることです。このように両者が厳密に一致していれば、XML Schema をデータベースのオブジェクト・リレーショナル・スキーマにマップし、前述のスキーマに従って有効なドキュメントをデータベース・スキーマの行オブジェクトにマップすることは比較的簡単です。実際に、DTD より XML Schema の方が表現が豊富なため、マッピングが大幅に簡単になります。

XML Schema で得られるスキーマ制約が適用されるのは、データ駆動アプリケーションのみではありません。動的な動作が禁止されているドキュメント駆動アプリケーションは多数存在します。

- 単純な例には、マークアップ・タグに基づいて様々にルート指定されるメモがあります。
- より高度な例は、航空会社の欧州路線に関するテクニカル・サービス・マニュアルです。マニュアルの作成者は、XML Schema が提供する複合制約に基づいて常に有効な部品番号を入力することが保証され、部品番号の妥当性が在庫レベル、需給の変動または規制の変化などの動的な考慮事項に依存することも保証されます。

データ交換アプリケーションにおける一般的な XML 設計の問題

この項では、データ交換アプリケーションに関する次の XML 設計の問題について説明します。

- データベースに格納された XML データからの Web フォームの生成
- Web フォームからデータベースへの XML データの送信

データベースに格納された XML データからの Web フォームの生成

Web フォームのインフラストラクチャを生成する手順は、次のとおりです。

1. XML SQL Utility を使用し、基礎となる問合せ対象の表のスキーマに基づいて DTD を生成します。
2. 生成された DTD を XML Java Class Generator への入力として使用し、DTD 要素に基づいてクラス・セットを生成させます。
3. これらのクラスを使用して Web ベース・フォームの基礎となるインフラストラクチャを生成する Java コードを作成します。このインフラストラクチャに基づいて、Web フォームでユーザー・データを獲得し、データベース・スキーマと互換性のある XML 文書を作成できます。このデータは、対応するデータベース表やオブジェクト・ビューに直接書き込むことができます。それ以外の処理は不要です。

Web フォームからデータベースへの XML データの送信

Web フォーム経由で取得したデータが基礎となるデータベース・スキーマにマップされることを保証する方法の 1 つは、Web フォームおよび基礎となる構造を、スキーマ互換 DTD に基づいて XML データを生成するように設計することです。この項では、XML SQL Utility と XML Parser for Java を使用して、この設計作業を行う方法について説明します。この使用例の流れは次のとおりです。

1. Java アプリケーションで XML SQL Utility を使用して、ターゲットとなるオブジェクト・ビューまたは表の形式と一致する DTD を生成します。
2. この DTD をアプリケーションから XML Class Generator for Java に送ると、ユーザーに対して表示する Web フォームの設定に使用できるクラスが構築されます。
3. 生成されたクラスを使用して、JavaServer Pages、Java サーブレットまたは他のコンポーネントにより Web フォームが動的に構築されます。
4. ユーザーがフォームに入力して送信すると、サーブレットはデータを適切な XML データ構造にマップし、XML SQL Utility はデータをデータベースに書き込みます。

XML SQL Utility の DTD 生成機能を使用すると、ターゲットとなるオブジェクト・ビューまたは表の XML 形式を判断できます。そのためには、オブジェクト・ビューまたは表の `SELECT * FROM` を実行して XML の結果を生成します。

この結果では、DTD 情報が別々のファイルとして含まれるか、XML ファイルの先頭に DOCTYPE タグで囲まれた状態で埋め込まれます。

この DTD を XML Class Generator への入力として使用し、DTD 要素に基づいてクラス・セットを生成させます。次に、これらのクラスを使用して Web ベース・フォームの基礎となるインフラストラクチャを生成する Java コードを作成します。その結果、Web フォーム経由で送られたデータは、データベースに書き込むことができる XML 文書に変換されます。

アプリケーション間の XML 文書の送信

アプリケーション間で XML 文書を送信するには、多数の方法があります。この項では、いくつかの一般的な方法について説明します。

次の場合について考えてみます。

- 送信側アプリケーションが XML 文書を送信する場合。
- 受信側アプリケーションが XML 文書を受信する場合。

ファイル転送。受信側アプリケーションは、FTP、NFS、SMB または他のファイル転送プロトコルを通じて、送信側アプリケーションからの XML 文書を要求します。ドキュメントは、受信側アプリケーションのファイル・システムにコピーされます。アプリケーションはファイルを読み込んで処理します。

HTTP。受信側アプリケーションは、サーブレットに HTTP 要求を送ります。サーブレットから XML 文書が受信側アプリケーションに戻され、そこで読み込まれて処理されます。

Web フォーム。送信側アプリケーションが Web フォームを表示します。ユーザーがフォームに入力し、ブラウザで実行される Java アプレットまたは Java スクリプト経由で情報を送ります。アプレットまたは Java スクリプトにより、ユーザーのフォームが XML 形式で受信側アプリケーションに送信され、そこで読み込まれて処理されます。受信側アプリケーションが最終的にデータをデータベースに書き込む場合、送信側アプリケーションは XML をデータベース互換の形式で作成する必要があります。この操作に Oracle XML 製品を使用する方法については、「Web フォームからデータベースへの XML データの送信」を参照してください。

アドバンスト・キューイング。Oracle データベースは、Oracle Advanced Queueing（AQ）を通じて、Net Services、HTTP および JDBC 経由で、XML 文書を 1 つ以上の受信側アプリケーションにメッセージとして送信します。受信側アプリケーションは、XML メッセージをデキューして処理します。

関連項目：

- 『Oracle9*i* アプリケーション開発者ガイド - XML』の「Oracle AQ を使用した XML データの交換」
- 第 8 章「オンライン B2B XML アプリケーション : 手順」
- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』

データベースへの XML のロード

XML データや DTD ファイルを Oracle9*i* にロードする方法は、次とおりです。

- `DBMS_LOB` など、LOB 用の PL/SQL ストアド・プロシージャの使用
- Java（Pro*C、C++）カスタム・コードの作成
- SQL*Loader の使用
- XML SQL Utility（XSU）

また、Oracle Internet File System（Oracle 9*i*FS）を使用して XML 文書をデータベースに入れることもできます。DTD はサポートされませんが、DTD にかわる標準である XML Schema はサポートされます。

SQL*Loader の使用

SQL*Loader を使用すると LOB をバルク・ロードできます。

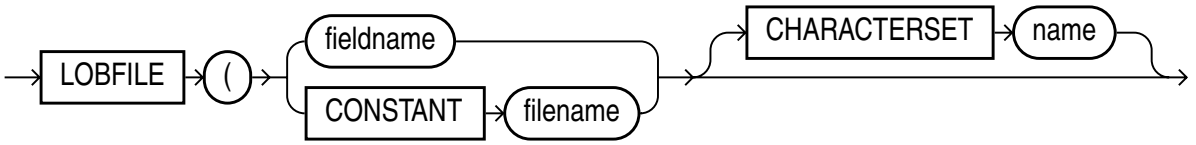
関連項目：

- SQL*Loader を使用して LOB をロードする方法の詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。
- SQL*Loader の使用方法の簡単な説明と使用例は、『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』の第 4 章「LOB の管理」の「LOB ロード時の SQL*Loader の使用」を参照してください。

SQL*Loader を使用した LOB への XML 文書のロード

LOB はきわめて大きい場合があるため、SQL*Loader では LOB データをメイン・データ・ファイル（データの残りはインライン）または LOBFILE からロードできます。図 2-2 に、LOBFILE の構文を示します。

図 2-2 LOBFILE の構文



LOB データが長すぎて、LOBFILE からロードする方がよい場合があります。LOBFILE では、LOB データ・インスタンスは引き続きフィールドにある（事前にサイズが決まっているデリミタ付きの長い値）とみなされますが、これらのフィールドはレコードに編成されません（LOBFILE にはレコードという概念は存在しません）。したがって、レコードの処理に伴うオーバーヘッドが回避されます。この種のデータ編成は、LOB のロードに適しています。

LOBFILE からの LOB をメモリーに格納する場合、要件はありません。SQL*Loader では、LOBFILE が 64KB のチャンク単位で読み込まれます。64KB を超える物理レコードをロードするには、`READSIZE` パラメータを使用して大きいサイズを指定できます。

CLOB に XMLType 列または XML データを含む列をロードする最善の方法は、LOBIFLE を使用することです。

- **XML が有効な場合**。LOBFILE 内の XML データが大きく、それが有効な XML であることがわかっている場合は、すべての XML 検証処理をバイパスするため、ダイレクト・パス・ロードを使用します。
- **XML の検証が必要な場合**。XML データの妥当性検証が必要な場合は、ダイレクト・パス・ロードほど有効ではないことを念頭に置いて、従来型パスによるロードを使用します。

従来型パスによるロードでは、SQL の `INSERT` 文が実行され、Oracle データベースに表が移入されます。ダイレクト・パス・ロードでは Oracle データ・ブロックがフォーマットされ、データ・ブロックがデータベース・ファイルに直接書き込まれることで、Oracle データベースのオーバーヘッドの大部分が排除されます。

ダイレクト・パス・ロードはデータベース・リソースの他のユーザーと競合しないため、通常はデータをほぼディスク速度でロードできます。制限、セキュリティおよびバックアップの含意など、ダイレクト・パス・ロードに伴う考慮事項は、『Oracle9*i* データベース・ユーティリティ』の第 9 章を参照してください。

図 2-3 に、SQL*Loader のダイレクト・パス・ロードと従来型パスによるロードを示します。

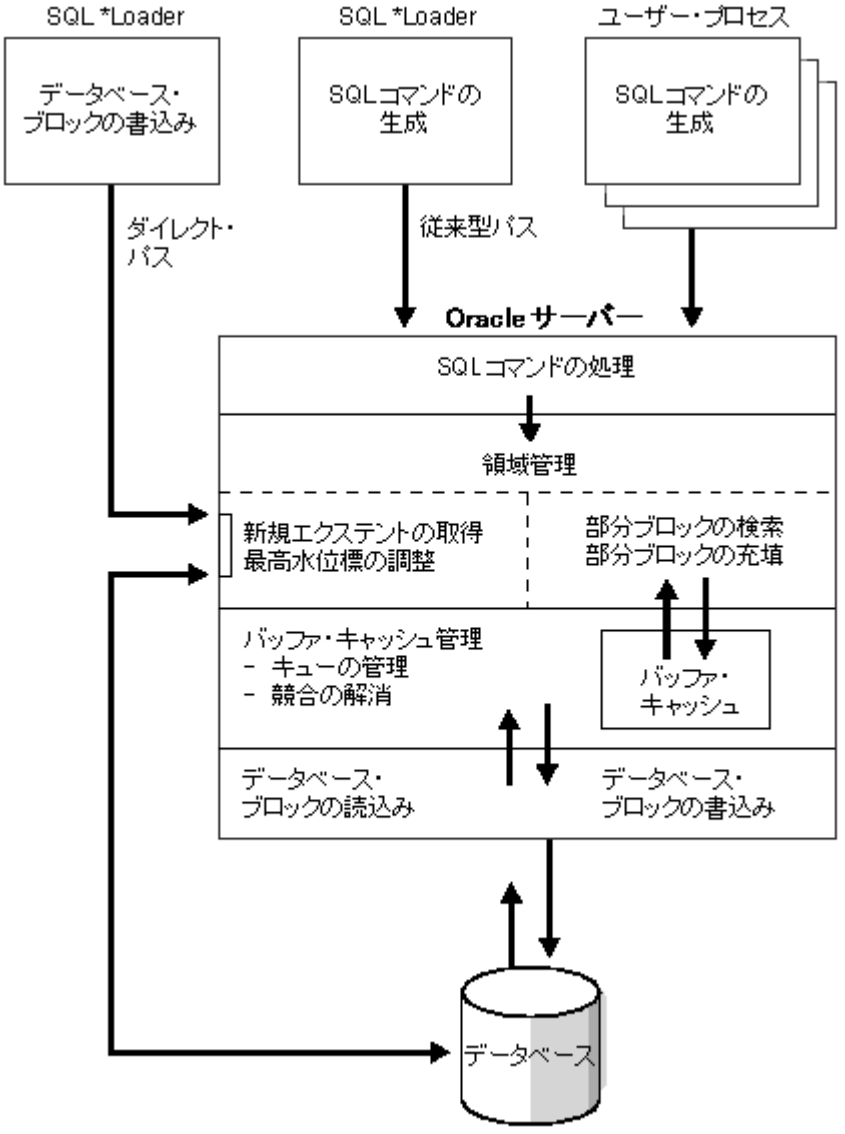
ロードする表は、データベースに存在する必要があります。SQL*Loader で表が作成されることはありません。すでにデータが含まれている既存の表または空の表がロードされます。

ロードするには、次の権限が必要です。

- ロードする表に対する INSERT 権限が必要です。
- REPLACE または TRUNCATE オプションを使用して表の古いデータを削除した後に新しいデータをロードする場合は、ロードする表に対する DELETE 権限が必要です。

関連項目： ロードの詳細と例は、『Oracle9*i* データベース・ユーティリティ』の第 7 および 9 章を参照してください。

図 2-3 SQL*Loader: ダイレクト・パス・ロードと従来型パスによるロード



Oracle XML 対応テクノロジを使用するアプリケーション

インターネット・アプリケーションでは、XML の様々な用途が考えられます。Oracle の XML コンポーネントは、次の 2 つのデータベース中心のアプリケーションの分野に適しています。

- 「Oracle の XML 対応テクノロジによるコンテンツおよびドキュメントの管理」。これには、データ表示のカスタマイズが含まれます。
- 「Business-to-Business（B2B）および Business-to-Consumer（B2C）メッセージ機能」。システム間またはシステム内アプリケーションでのデータ交換に使用します。

また、この 2 つを併用することもできます。この 2 つのアプリケーション分野については、それぞれ第 II 部「XML を使用したコンテンツおよびドキュメントの管理」と第 III 部「XML データ交換」で重点的に説明します。

この章では、この 2 つの分野におけるビジネス・アプリケーションの使用例について説明します。

Oracle の XML 対応テクノロジによるコンテンツおよびドキュメントの管理

データ表示のカスタマイズ

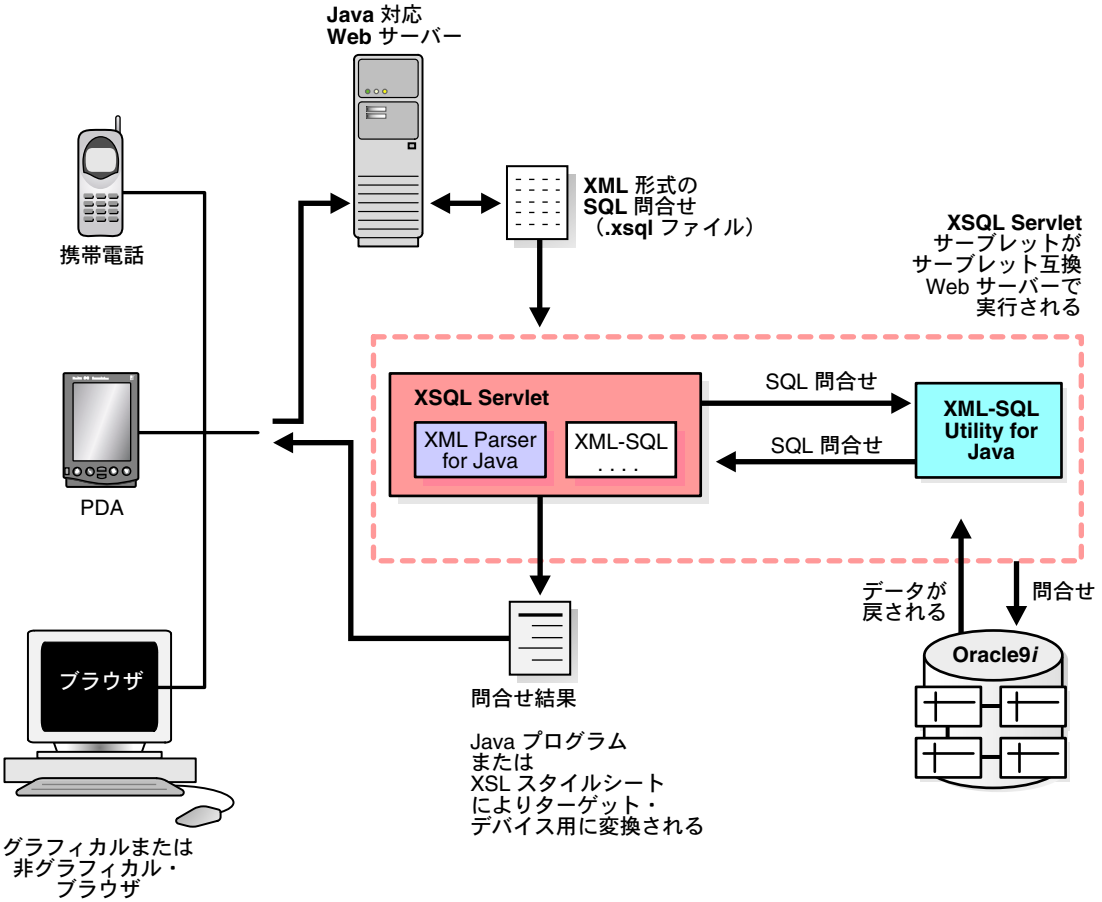
XML の普及により、様々なブラウザ、デバイスおよびユーザー向けにデータ表示をカスタマイズできます。XML 文書をクライアント、中間層またはサーバーで XSL スタイルシートと併用すると、次のように多様なクライアント・デバイスの個々のユーザーに合せて調整された XML データを変換、編成および表示できます。

- グラフィカルおよび非グラフィカルな Web ブラウザ
- Palm Pilot などの携帯情報端末（PDA）
- デジタル携帯電話およびポケットベル

これにより、ビジネス・アプリケーションの対象をビジネス操作に絞り、様々な出力デバイスに簡単に対処できます。

XML と XSL を使用すると、動的 Web サイトの作成と管理も簡単になります。XSL スタイルシートを変更するのみで表示を変更でき、基礎となるビジネス・ロジックやデータベース・コードを変更する必要はありません。新規ユーザーやデバイス用に、必要に応じて新しい XSL スタイルシートを設計するだけですみます。図 2-4 に、これを示します。

図 2-4 コンテンツの管理 : 表示のカスタマイズ



関連項目： 『Oracle9*i* アプリケーション開発者ガイド - XML』の「XML Parser for Java の使用」

Oracle の XML コンポーネントを使用したコンテンツ管理の次の使用例について考えてみます。

- 使用例 1. コンテンツおよびドキュメントの管理 : XML 対応 Oracle テクノロジを使用した複合ドキュメントの公開
- 使用例 2. コンテンツおよびドキュメントの管理 : Oracle の XML テクノロジを使用した個人情報の配信
- 使用例 3. コンテンツ管理 : Oracle の XML テクノロジを使用したデータ駆動のアプリケーションのカスタマイズ

使用例ごとに、ビジネス上の問題、解決策、主要な作業および使用する Oracle XML のコンポーネントについて簡単に説明します。

これらの使用例については、第 II 部「XML を使用したコンテンツおよびドキュメントの管理」の事例で具体的に説明します。

使用例 1. コンテンツおよびドキュメントの管理 : XML 対応 Oracle テクノロジを使用した複合ドキュメントの公開

問題

会社 X には、SGML および XML でマークアップされたテキスト・フラグメントの多数のドキュメント・リポジトリがあります。複合ドキュメントを動的に公開する必要があります。

解決策

最も重要なのは、データベース・アプリケーション設計を適切なデータベースの設計から始める必要があることです。つまり、会社 X は、まず適切なデータ・モデル化および設計ガイドラインを使用する必要があります。これにより、より短時間のうちにデータに対するオブジェクト・ビューを作成できます。

XMLType を使用して、リレーショナル・データを更新できる XML 形式でドキュメントを格納します。Oracle Internet File System（Oracle 9*i*FS）をデータ・リポジトリ・インタフェースとして使用します。Oracle 9*i*FS を使用すると、XML データ・リポジトリの管理作業を実装できます。

会社 X は、XSL スタイルシートを使用してドキュメント・セクションやフラグメントを組み合せ、複合ドキュメントをユーザーに電子的に配信できます。望ましい解決策の 1 つは、単一のソースおよび作成またはマルチチャネルによる公開には Arbortext および EPIC を使用することです。マルチチャネルによる公開を使用すると、同じドキュメントを HTML、PDF、Word、ASCII テキスト、SGML および FrameMaker など、様々な形式で生成できます。

関連項目： Arbortext および EPIC 製品の詳細は、
http://www.arbortext.com/jp を参照してください。

図 2-5 を参照してください。

関連する主要な作業

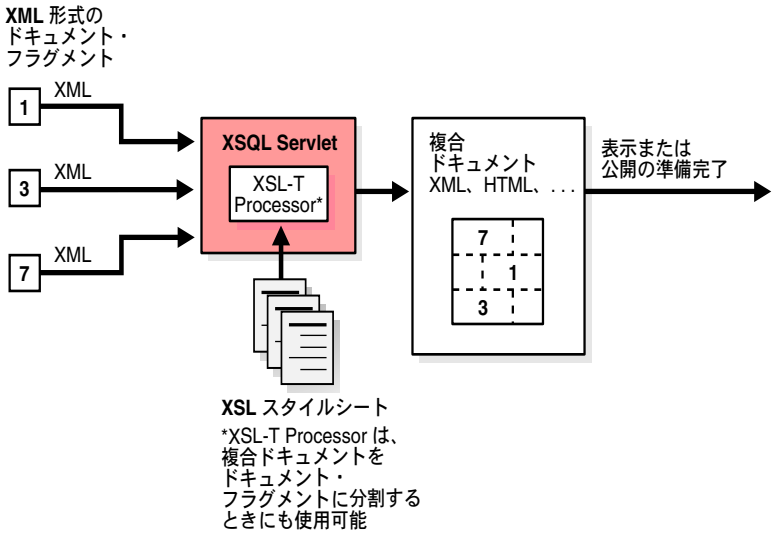
使用例 1 の解決策に関連する主要な作業は、次のとおりです。

1. データベースを慎重に設計します。使用する XML のタグと要素を決定します。
2. これらのセクションやフラグメントをデータベースの CLOB の XMLType 列に格納します。
3. セクションやフラグメントを完全なドキュメントとして表示するための XSL スタイルシートを作成します。

使用する Oracle XML コンポーネント

- XML Parser および XSLT
- セクションやフラグメントをデータベースに出し入れするための XSQL Servlet および XSU

図 2-5 使用例 1. XSL を使用した複合ドキュメントの作成と公開



使用例 2. コンテンツおよびドキュメントの管理 : Oracle の XML テクノロジを使用した個人情報の配信

問題

大規模なニュース配信プロバイダは、各種のニュース・ソースからデータを受信します。このデータをデータベースに格納し、すべての配信者とユーザーが、ニュース配信元との契約に従って、いつでもカスタマイズされた特定のニュースを参照できるように、必要に応じて送信する必要があります。配信元は、XSL を使用してデータを正規化し、データベースに格納します。格納されたデータは、複数の Web サイトとポータルの支援に使用されます。このような Web サイトやポータルは、各種の有線および無線クライアントから HTTP 要求を受信します。

解決策

XSL スタイルシートと XSQL Servlet を使用して、要求側サービスに適切な変換を動的に配信します。図 2-6 を参照してください。また、『Oracle9*i* アプリケーション開発者ガイド - XML』の Oracle9*i*AS Dynamic Services および Oracle Syndication Server（OSS）に関する章も参照してください。

関連項目：

- 第 3 章「Oracle9iAS Wireless Edition と XML」
- 第 5 章「XML を使用したコンテンツのカスタマイズ : Dynamic News アプリケーション」

関連する主要な作業

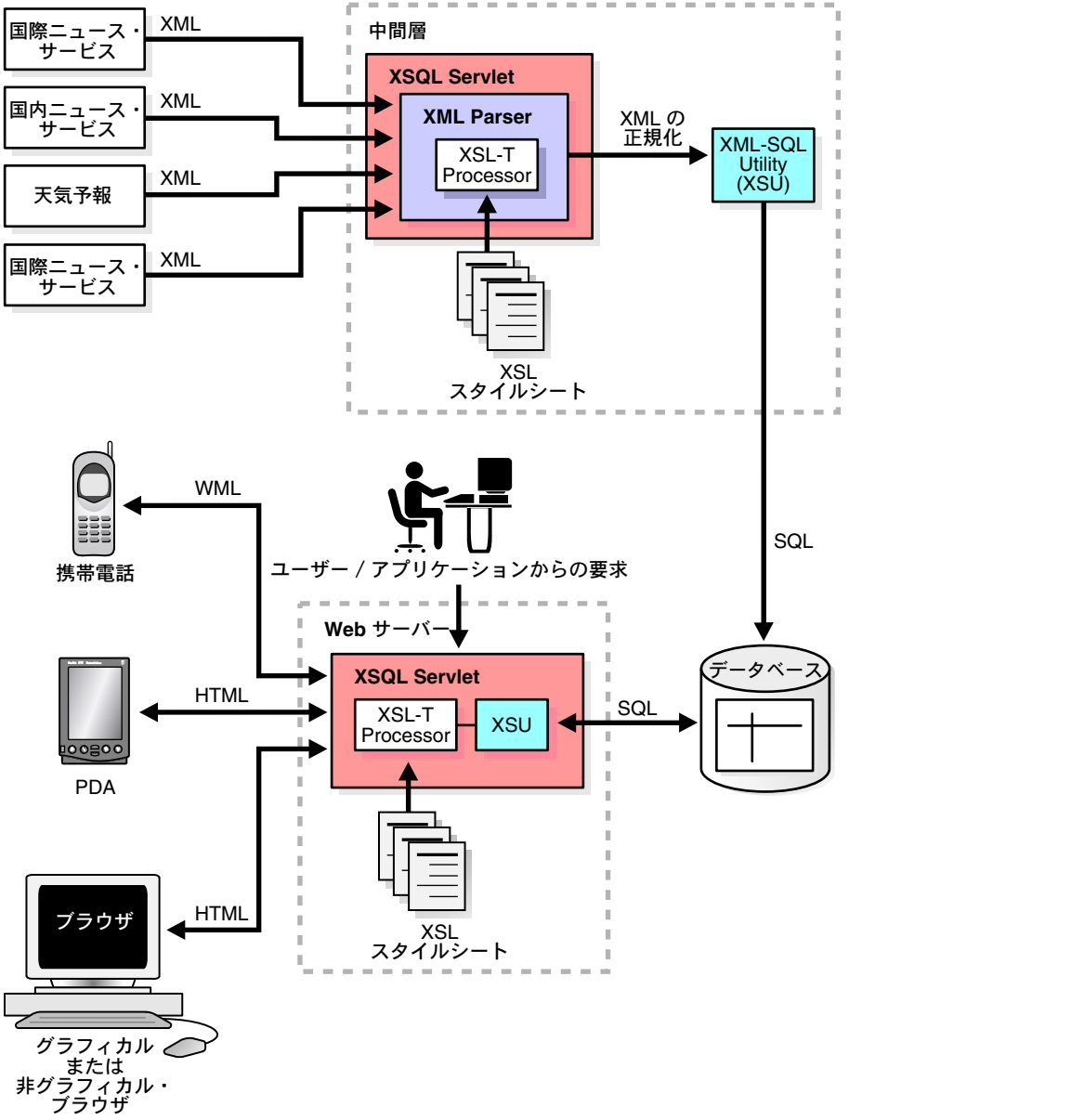
使用例 2 に関連する主要な作業は、次のとおりです。

1. データベース・スキーマのデータ・モデルを、最適の出力が得られるように設計します。
2. 情報ソースごとに、正規化された形式に変換するための XSL スタイルシートを作成し、データベースに格納します。
3. データを Web サイトに表示するために、XSQL ページとともに XSL スタイルシートを作成します。

使用する Oracle XML コンポーネント

- XML Parser for Java v2
- XML SQL Utility（XSU）
- XSQL Servlet

図 2-6 使用例 2. Oracle の XML コンポーネントによるカスタマイズされたニュース情報の配信



使用例 3. コンテンツ管理 : Oracle の XML テクノロジを使用したデータ駆動のアプリケーションのカスタマイズ

問題

会社 X は、データを Thin クライアントに対話形式で配信する必要があります。

解決策

問合せはクライアントからデータベースに送信され、データベースでは出力が 1 つ以上の XSL スタイルシートを通じて動的に表示され、クライアント・アプリケーションに送信されます。データは、リレーショナル・データベースに LOB で格納され、XML で実体化されます。

関連する主要な作業

第 7 章「XSL を使用した Discoverer 4i Viewer のカスタマイズ」を参照してください。また、『Oracle9*i* アプリケーション開発者ガイド - XML』の Reports9*i* の章も参照してください。

使用する Oracle XML コンポーネント

- XML Parser for Java および XSLT Processor

Business-to-Business（B2B）および Business-to-Consumer（B2C）メッセージ機能

ビジネス・アプリケーション開発者にとっての問題は、様々なベンダーや様々なアプリケーション・ドメインのアプリケーションで生成されたデータを結合することです。Oracle の XML 対応テクノロジでは、データとそのコンテキストに重点が置かれ、特定のネットワークや通信プロトコルに限定されないことで、このようなアプリケーション間でのデータ交換が簡単になります。

アプリケーションでは XML および XSL 変換を使用してデータを交換でき、独自のデータ形式や互換性のないデータ形式を管理したり解釈する必要がありません。

Oracle の XML コンポーネントを使用する次の Business-to-Business および Business-to-Consumer（B2B/B2C）メッセージ機能の使用例を考えてみます。

- 使用例 4. B2B メッセージ機能 : XML を使用した複数ベンダー用のオンライン・ショッピング・カート設計
- 使用例 5. B2B メッセージ機能 : オンライン在庫管理アプリケーションのための Oracle XML コンポーネントおよび AQ の使用
- 使用例 6. B2B メッセージ機能 : Oracle の XML 対応テクノロジおよび AQ を使用した複数アプリケーションの統合

それぞれの使用例では、問題、解決策、問題解決に使用する主要な作業および使用する Oracle XML コンポーネントについて簡単に説明します。

これらの使用例については、第 III 部「XML データ交換」の事例で具体的に説明します。

使用例 4. B2B メッセージ機能 : XML を使用した複数ベンダー用のオンライン・ショッピング・カート設計

問題

会社 X は、各種ベンダーから入荷する製品用にオンライン・ショッピング・カートを構築する必要があります。また、注文をオンラインで受信し、受注した製品に基づいて適切なベンダーに転送しようと考えています。

解決策

XML を使用して、統合されたオンライン購入申込書を配信します。ユーザーはコンピュータ・メーカーの Web サイトに直接アクセスし、最新モデル、構成オプションおよび価格をブラウズしながら、新規ハードウェアに関する新規の購買依頼を入力できます。ユーザーのサイトからベンダーの Web サイトに、購買依頼の参照番号と認証情報が送信されます。

ベンダー・サイトで、ユーザーがショッピング・カートに品目を追加し、ボタンをクリックしてショッピングを完了します。ベンダーは、ユーザーが選択した部品番号、数量および価格を含む XML ファイルとして、ショッピング・カートの内容を会社 X のアプリケーションに送信します。

ショッピング・カートからの品目は、新規購買依頼に明細項目として自動的に追加されます。

顧客の発注書（XML 形式）は、適切なベンダー・データベースに配信され、そこで処理されます。XSL は、ショッピング・カートの変換と転送のための分割に使用されます。データはリレーショナル・データベースに格納され、XML を使用して実体化されます。図 2-7 を参照してください。

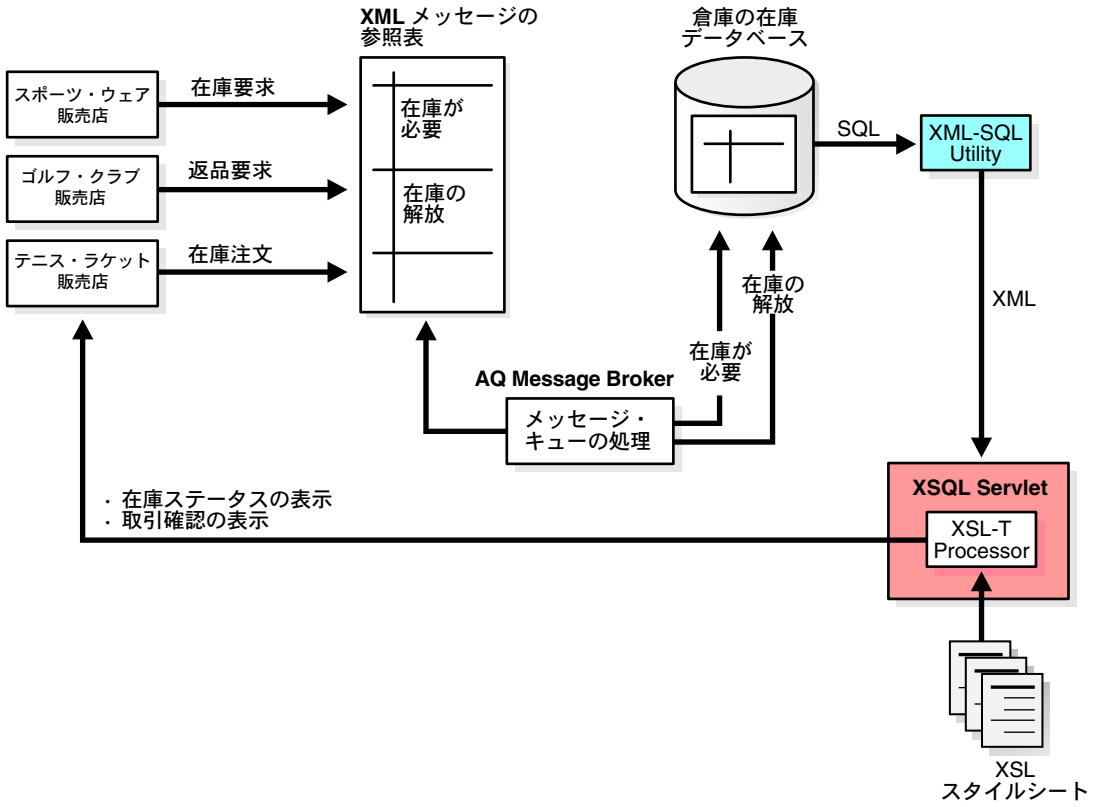
関連項目： より単純な実装例は、第 8 章「オンライン B2B XML アプリケーション : 手順」を参照してください。

- XML とアドバンスト・キューイングの併用に関する基本情報は、『Oracle9*i* アプリケーション開発者ガイド - XML』の第 9 章を参照してください。
- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』

使用する Oracle XML コンポーネント

- Oracle XML Parser
- XML SQL Utility
- XSQL Servlet

図 2-7 使用例 4. Oracle の XML コンポーネントを使用した複数ベンダー用のオンライン・ショッピング・カート



使用例 5. B2B メッセージ機能 : オンライン在庫管理アプリケーションのための Oracle XML コンポーネントおよび AQ の使用

問題

クライアント / サーバー・アプリケーションおよびサーバー間アプリケーションにより、データ・リソースとインベントリがデータベース・リポジトリに格納されます。このリポジトリは、会社間で共有されます。会社 X は、データ・リソースがアクセスされるたびにそれを知る必要があり、システム上のすべてのユーザーおよび顧客はデータがアクセスされる時期とアクセス場所を知る必要があります。

解決策

リソースがアクセスまたは解放されると、可用性に関する XML メッセージがトリガーされます。これにより、必要に応じてリソースが XSL を使用して複数のクライアントの形式に変換されます。逆に、あるクライアントがリソースを取得すると、そのリリースが削除されたことを示す XML メッセージが他のクライアントに送信されます。メッセージは LOB に格納されます。データはリレーショナル・データベースに格納され、XML を使用して実体化されます。図 2-8 を参照してください。

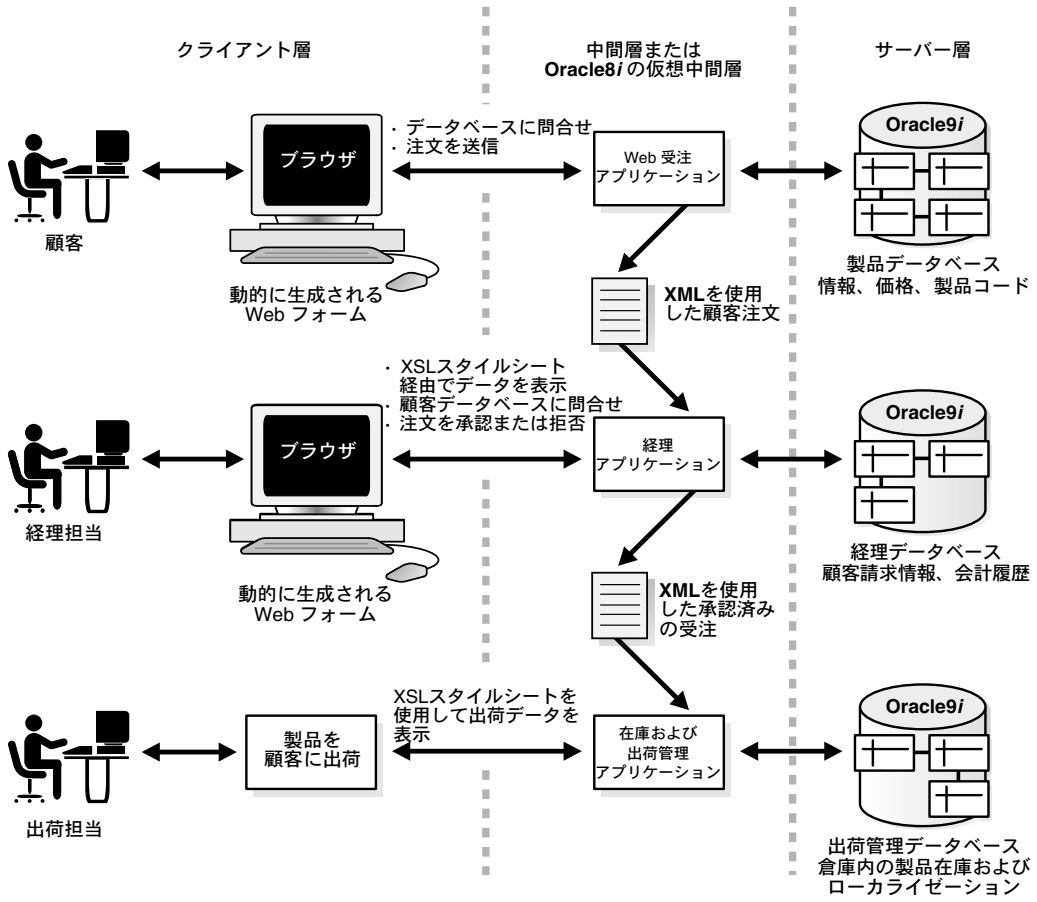
関連項目 : より単純な実装例は、第 8 章「オンライン B2B XML アプリケーション : 手順」を参照してください。

- XML とアドバンスト・キューイングの併用に関する基本情報は、『Oracle9*i* アプリケーション開発者ガイド - XML』の第 9 章を参照してください。
- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』

使用する Oracle XML コンポーネント

- XML Parser
- XSLT Processor

図 2-8 使用例 5. オンライン在庫管理アプリケーションのための Oracle XML コンポーネントおよび AQ の使用



使用例 6. B2B メッセージ機能 : Oracle の XML 対応テクノロジおよび AQ を使用した複数アプリケーションの統合

問題

会社 X は、複数のアプリケーションでデータを通信および共有して、ビジネス・ワークフローおよびプロセスを統合する必要があります。

解決策

XML をメッセージ・ペイロードとして使用します。これは XSLT Processor 経由で変換され、エンベロープに入れられ、適切にルート指定されます。XML メッセージは、AQ Broker データベースの LOB に格納されます。メッセージとデータのルーティングと変換が簡単になるように、Oracle Workflow を使用します。この解決策では、コンテンツ管理も利用します。この場合は、XSL スタイルシートを使用して表示をカスタマイズします。図 2-9 を参照してください。

関連する主要タスク

1. ユーザーまたはアプリケーションが要求を送信します。結果として生成されたデータは、XSU を使用して社内データベースから取り出されます。
2. データは XSLT Processor により変換され、AQ Broker に送信されます。
3. AQ Broker はこのメッセージを読み込み、それに従って必要なアクションを判断します。次に、適切な応答をアプリケーション 1、2 および 3 に対して発行し、処理を続行します。

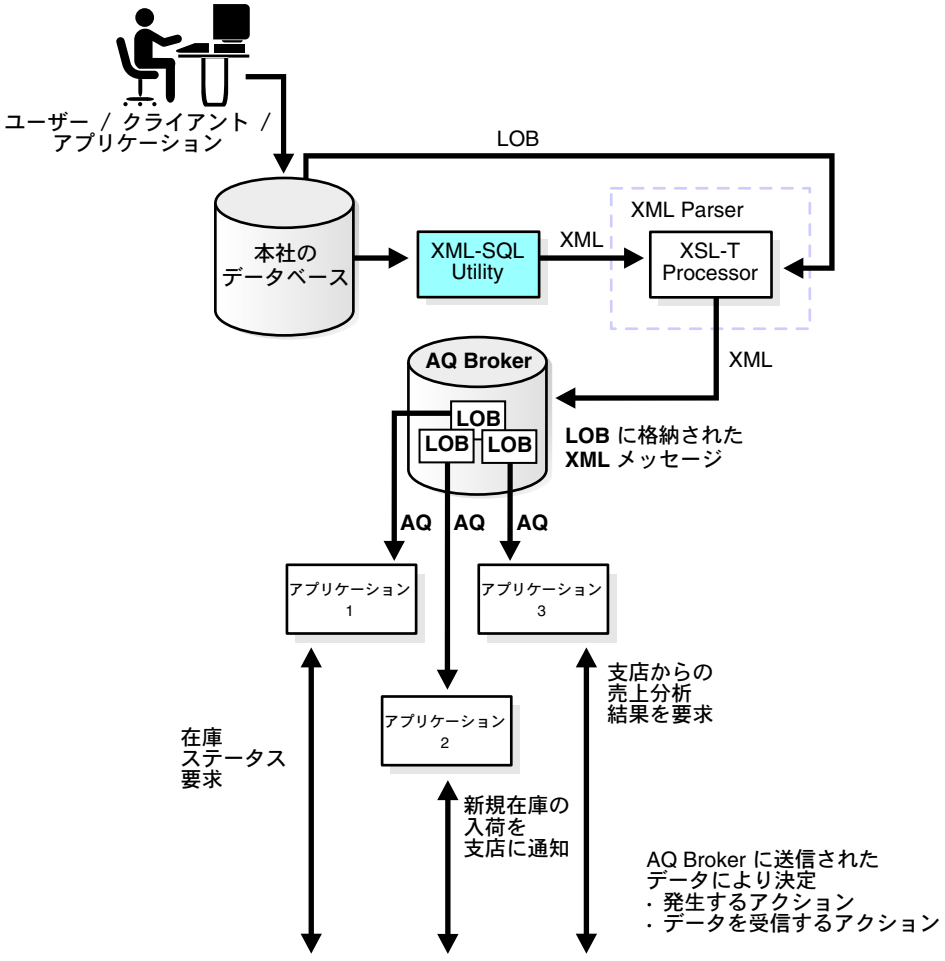
関連項目： より単純な実装例は、第 8 章「オンライン B2B XML アプリケーション : 手順」を参照してください。

- XML とアドバンスト・キューイングの併用に関する基本情報は、『Oracle9*i* アプリケーション開発者ガイド - XML』の第 9 章を参照してください。
- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』

使用する Oracle XML コンポーネント

- XML Parser
- XSLT Processor
- XML SQL Utility（XSU）

図 2-9 使用例 6. Oracle XML コンポーネントおよびアドバンスト・キューイングを使用した複数アプリケーションの統合



第 II 部

XML を使用したコンテンツおよびドキュメントの管理

第 II 部では、事例をあげて、XML ベースのコンテンツおよびドキュメント管理の実装方法について説明します。

第 II 部の内容は、次のとおりです。

3

Oracle9*i*AS Wireless Edition と XML

この章の内容は、次のとおりです。

Oracle9*i*AS Wireless Edition（Portal-to-Go）の概要

Oracle9*i* Application Server Wireless Edition（Oracle9*i*AS Wireless Edition）を使用すると、通信会社、企業およびインターネット会社は、Smartphone、ポケットベル、PDA など、あらゆるワイヤレス・インターネット・デバイス向けに、新規または既存のインターネット・アプリケーションやコンテンツをワイヤレスで使用可能にできます。

OracleMobile はオラクル社の 1 部門であり、Oracle9*i* Application Server Wireless Edition テクノロジに基づいてワイヤレスの Web サイトの作成サービスやホスティング・サービスを提供するワイヤレス・アプリケーション・サービス・プロバイダ（WASP）です。

Oracle9*i* Application Server Wireless Edition

コンテンツ・アダプタは、すべてのコンテンツを XML に変換します。コンテンツ・トランスフォーマは、XML をどんなデバイスでもサポートされるマークアップ言語（HTML、WML、HDML、VoiceXML、VoxML、SMS など）に変換します。Oracle9*i*AS Wireless Edition のオープンなアーキテクチャと XML テクノロジの使用により、現在および将来の標準のサポートが保証されます。Oracle9*i*AS Wireless Edition は、ユーザーおよびデバイスにロケーション・ベース・サービス、ワイヤレス・メッセージ機能、ワイヤレス M-Commerce および広範囲なパーソナライズを提供します。

ほとんどの Web クライアントは PC ですが、Meta Group は「2003 年までにはインターネット・アクセスの 50% 以上が PC 以外からのものになるだろう」と予測しています。

Oracle9*i*AS Wireless Edition（Portal-to-Go）により、次のサービスが使用可能になります。

- 事実上あらゆるワイヤレス・デバイスから、保護 E-Business アプリケーションなど、既存の Web またはデータベース・アプリケーションやコンテンツにアクセスできます。
- 無線通信業者は、広範囲の E-Commerce サービス・プロバイダになることができます。

Portal-to-Go は Oracle Internet Platform のコンポーネントであり、Web コンテンツを対応デバイスに配信するために必要なすべての機能を備えたサーバー製品です。Portal-to-Go は既存のコンテンツをデバイス固有の形式に変換し、エンド・ユーザーにポータル・インタフェースを提供します。

キーとなる XML

XML は、コンテンツ・プロバイダがモバイル・ユーザーに様々な形式でデータを配信するためのキーとなります。XML により、ソース・コンテンツの形式がターゲット・デバイスの形式から分離され、コンテンツ・プロバイダはあらゆるソースからデータを取り込んで、どんなターゲットにも配信できるようになります。このような XML ベースのテクニックは、次のようにデータをある形式から別の形式に変換するアプリケーションに使用します。

- エンタープライズ・アプリケーションの統合
- ユーザー・プロファイルに基づくコンテンツ配信のカスタマイズ
- 市場のサプライヤが交換に使用している形式によるコンテンツとサービスの集計

Portal-to-Go のコンポーネント

Oracle9*i*AS Wireless Edition ポータルのコンポーネントは、次のとおりです。

- コンテンツ・アダプタ ― すべてのコンテンツを独立した XML に変換します。
- コンテンツ・トランスフォーマ ― XML をデバイス固有のマークアップ言語に変換します。
- Service Designer ― Web サービスの設計方法と管理方法を指定します。サービスは、データをモバイル・デバイスに配信します。
- Personalized Portal ― ユーザーはアクセスするサービスをパーソナライズできます。
- Request Manager ― ユーザー・デバイスとサービス要求を認識します。

この章では、Oracle9*i*AS Wireless Edition で XML を使用して Web コンテンツをすべてのデバイスに使用可能にする方法について説明します。また、株式取引サービスを例にして、XML が中間的なデータ交換形式として果たす役割について説明します。

Oracle XML コンポーネント

Oracle9*i*AS Wireless Edition アダプタおよびトランスフォーマでは、XML Parser for Java v2 が使用されます。また、XML Parser for Java の XSLT パッケージも使用されます。

Oracle9*i*AS Wireless Edition（Portal-to-Go）の機能

Oracle9*i*AS Wireless Edition には、次のサポート機能が含まれています。

- Apache および Apache JServ
- すべてのモバイル・デバイス
- 出力変数名の明示的な設定など、デバイス出力のカスタマイズ
- シングルバイト、マルチバイトおよび固定幅のコード体系（および「$」などの特殊文字）の処理
- Oracle9*i*AS Wireless Edition に組み込まれている、情報をメッセージ機能対応デバイス（SMS、電子メールなど）にプッシュする、スケーラブルでユーザーによるカスタマイズが可能な通知エンジン
- ロケーション・ベース・サービスの開発に対する包括的なサポート
- 既存の全コンテンツの単純な自動フィルタリングおよび変換のためのフィルタリング・サービス
- ロケーション・マーク、ブックマークおよびオブジェクト制御など、ワイヤレス・ユーザーの作業を簡素化する、組込みの柔軟なポータルおよびパーソナライズ機能
- 拡張トランザクションおよびデータベースのロギング、イベント管理およびパフォーマンス監視

関連項目：

- リポジトリ・アップグレードの詳細は、『Oracle9*i* Application Server Wireless Edition コンフィギュレーション・ガイド』を参照してください。
- http://otn.oracle.co.jp/products/app_server/we/we.html

Oracle9*i*AS Wireless Edition の実行要件

Oracle9*i*AS Wireless Edition の実行要件は、次のとおりです。

- Oracle8*i* リリース 8.1.5 以上
- Oracle9*i* Application Server
- Java の構成要件
  - Service Designer。Oracle9*i*AS Wireless Edition Service Designer には JDK 1.2.2 が必要です。JDK 1.2.2 は、Oracle9*i*AS Wireless Edition の CD-ROM からインストールできます。
  - Web Integration Developer。Web Integration Developer には独自の Java Virtual Machine（Java VM）が組み込まれており、Java の設定は不要です。
  - サーバー・コンポーネント。Oracle9*i*AS Wireless Edition のサーバー・コンポーネントは、JDK 1.1 または 1.2 で動作します。JDK 1.2 はパフォーマンスが改善されています。

Oracle9*i*AS Wireless Edition: サポートされるデバイスとゲートウェイ

トランスフォーマ

Oracle9*i*AS Wireless Edition では、次のベンダーによる最新の WAP 準拠デバイス用のトランスフォーマが提供されています。

- Alcatel 社
- Ericsson 社（R320 を含む）
- Motorola 社（Timeport を含む）
- Neopoint 社（NP1000 を含む）
- Nokia 社（7100）
- Samsung 社

また、独自のトランスフォーマを作成し、Oracle9*i*AS Wireless Edition のサポート機能を他のデバイスへと拡張することもできます。

WAP ゲートウェイ

Oracle9*i*AS Wireless Edition は、次の WAP ゲートウェイでのテストを完了しています。

- Phone.com UP.link Gateway
- Nokia WAP Gateway
- Ericsson WAP Gateway
- Infinite Technologies WAPLite

Oracle9*i*AS Wireless Edition の動作

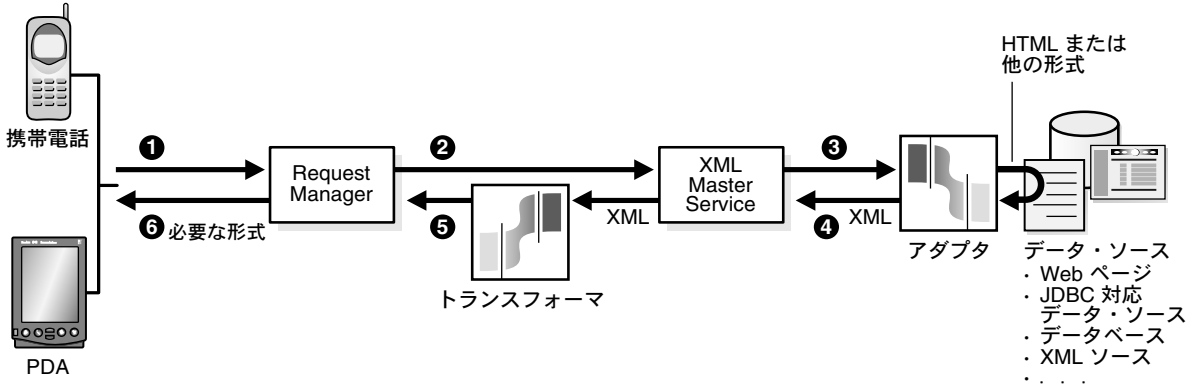
図 3-1 に、Oracle9*i*AS Wireless Edition（Portal-to-Go）の動作を示します。エンド・ユーザーが Oracle9*i*AS Wireless Edition サービスを要求すると、次の作業が実行されます。

1. Oracle9*i*AS Wireless Edition の Request Manager が、認証などのユーザー・レベルの前処理を実行します。
2. Request Manager が対応する Master Service に要求を送信します。
3. Master Service がアダプタを起動し、要求されたコンテンツを取り出します。
4. アダプタがコンテンツを XML 形式で戻します。
5. トランスフォーマが XML コンテンツをターゲット・デバイスに適した形式に変換します。
6. Request Manager が情報をデバイスに戻します。

XML および関連テクノロジは、次のように Oracle9*i*AS Wireless Edition の機能の中核となるものです。

- XML により、表示とコンテンツが分離されます。
- DTD により XML タグがユーザー・インタフェース（UI）要素にマップされます。
- XSL スタイルシートにより、結果のフォーマット、ソートおよびフィルタリングに関する規則が定義されます。

図 3-1 Oracle9*i*AS Wireless Edition の動作



Oracle9*i*AS Wireless Edition コンポーネント

Oracle9*i*AS Wireless Edition サービス

Oracle9*i*AS Wireless Edition サービスにより、Oracle9*i*AS Wireless Edition クライアントから要求された情報が 1 単位ごとにカプセル化され、クライアントに配信されます。次のようなサービスがあります。

- 株式取引
- ニュース
- 地図
- 電子メール

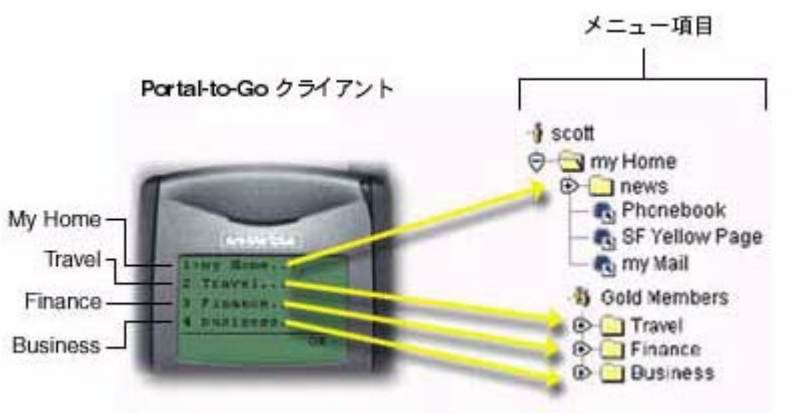
既存の Web サイト、データベースへの問合せまたは XML ソースからサービスを構築できます。

Master Service

Master Service は、サービスを実装して特定のアダプタを起動する Oracle9*i*AS Wireless Edition オブジェクトです。通常、エンド・ユーザーには、サービスが携帯電話のメニュー項目として表示されるか、Web ページ上のリンクとして表示されます。エンド・ユーザーは、デバイス・インタフェースでメニュー項目を選択し、Master Service を起動します。Master Service からは、次の種類のデータが戻されます。

- 映画のレビューなどの静的テキスト
- 航空チケット予約システムなどのアプリケーション

図 3-2 エンド・ユーザーにメニュー項目として表示されるサービス：メニュー項目の選択時に起動される Master Service



Master Service はアダプタをデバイスのトランスフォーマにマッピングし、Oracle9*i*AS Wireless Edition のコンテンツ・ソースを配信プラットフォームにリンクします。各 Master Service は 1 つのアダプタに基づいています。Master Service は、使用するアダプタ独自のインスタンスを作成します。したがって、複数のサービスが同じタイプのアダプタを使用し、それぞれがアダプタにサービス固有の引数値を渡すことができます。

Oracle9*i*AS Wireless Edition アダプタ

Oracle9*i*AS Wireless Edition アダプタは、外部ソースからデータを取り出して Oracle9*i*AS Wireless Edition の XML 形式で表示する Java アプリケーションです。アダプタは、Master Service により起動されると、サービスのコンテンツを含む XML 文書を戻します。アダプタは、Oracle9*i*AS Wireless Edition サーバーとコンテンツ・ソース間のインタフェースを提供します。

アダプタの機能は、次のとおりです。

- データ・ソースへの接続
- コンテンツの取出し
- Oracle9*i*AS Wireless Edition の XML へのコンテンツの変換

Oracle9*i*AS Wireless Edition には、Web ページや JDBC 対応データ・ソースなど、一般的なコンテンツ・ソース用のビルトイン・アダプタと、他のコンテンツ・ソースで動作するように変更できるアダプタがあります。

どのアダプタも、Oracle9*i*AS Wireless Edition の XML を生成する必要があります。これは、Oracle9*i*AS Wireless Edition の DTD に準拠する整形式の有効な XML 文書です。

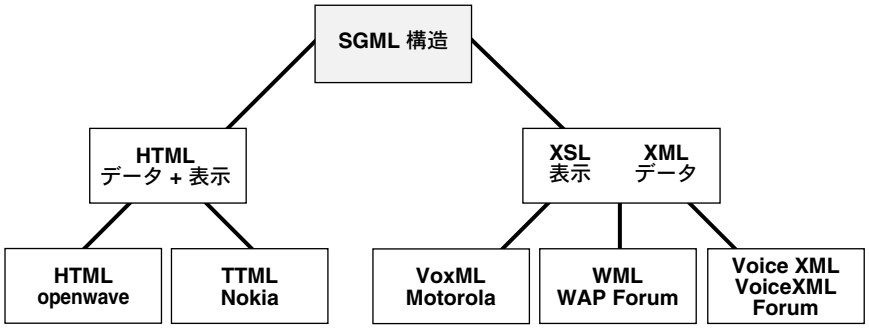
Oracle9*i*AS Wireless Edition トランスフォーマ

Oracle9*i*AS Wireless Edition トランスフォーマは、XML 文書をターゲット形式または別の Oracle9*i*AS Wireless Edition 形式に変換する、Java プログラムまたは XSL-T スタイルシートです。変換の他に、テキストの再配置、フィルタリングおよび追加もできます。トランスフォーマにより、コンテンツをターゲット・デバイスに最適の形式で表示できます。Oracle9*i*AS Wireless Edition では、次のマークアップ言語用のトランスフォーマが提供されています。

- WML 1.1 － WAP Forum により定義されたワイヤレス・マークアップ言語。
- Tiny HTML － Palm Pilots のような携帯デバイス（電話以外）に適した HTML のサブセット。
- VoxML － アプリケーションとの音声による対話を可能にする Motorola 社のマークアップ言語。
- TTML（Tagged Text Mark-up Language）－ Nokia 社が開発した HTML のサブセット。
- HDML（Handheld Devices Markup Language）－ 携帯デバイス専用に設計されたマークアップ言語。
- プレーン・テキスト － ショート・メッセージ・サービス対応のデバイスと電子メール・アプリケーション用にコンテンツを変換します。

図 3-3 に、これらのマークアップ言語とその導出関係を示します。

図 3-3 Oracle9*i*AS Wireless Edition でサポートされる複数の HTML および XML ベースのマークアップ言語



トランスフォーマを使用して、すべてのデバイスのコンテンツ表示を最適化し、新規デバイスのプラットフォームをサポートします。ほとんどの場合は、既存のトランスフォーマを変更するか再利用するだけですみます。

XML 経由のデータ交換 : Oracle9*i*AS Wireless Edition を使用した、ソースから XML または XML からターゲットへのデータ交換

XML を中間形式として使用すると、ソースとデバイスの種類を問わずデータを取り出して配信できます。株価とヘッドラインを提供する Web アプリケーションがあり、情報を携帯電話と PDA（Palm Pilot などの携帯情報端末）に配信する必要があるとします。

各デバイスにはコンテンツの形式に関して特定の要件があるため、各デバイスに同じデータは送信できません。そのための方法を考えてみます。Oracle9*i*AS Wireless Edition では、中間的なデータ形式が XML で定義されます。また、コンテンツ・プロバイダが次の作業を実行できるように、ツールも提供されています。

- ソース・コンテンツの抽出
- ソース・コンテンツから XML への変換
- XML から各デバイスのマークアップ言語への変換

コンテンツの抽出

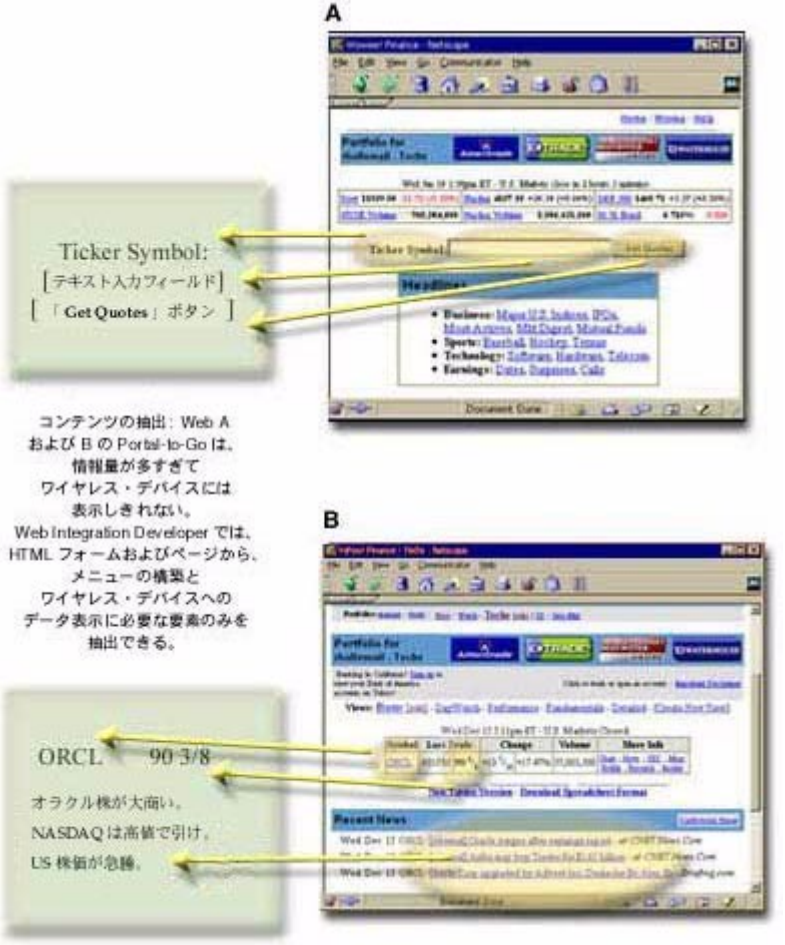
携帯デバイスにはデスクトップ・モニターほど多くの情報を表示できないため、情報を選択的に表示する必要があります。図 3-4 に、株式データ・アプリケーションからの 2 つの Web ページを意図的に判読不能にした状態で示します。

- A － 1 ページ目は、会社の証券コードを入力するフォームです。たとえば、ORCL は、オラクル社の証券コードです。
- B － 2 ページ目には、株価とその他の会社情報が表示されます。

どちらのページにも多数の広告、ボタン、ハイパーリンク、関連トピックなどが含まれています。最初の手順は、Web ページのうち、サービスからアクセス可能にする必要のある要素を識別することです。

次に、Web Integration アダプタを使用してコンテンツを XML に変換できます。

図 3-4 ワイヤレス・デバイスに表示する HTML ページ要素の抽出



XML への変換

Oracle9*i*AS Wireless Edition アダプタは、ソースからコンテンツを取り出します。前述の例では、Web ページから特定の株価とヘッドラインを取り出しています。その後、アダプタはコンテンツを XML に変換します。

中間的な XML 形式を使用する理由

ターゲット・デバイスの形式に直接変換しない理由について説明します。これには、柔軟性と拡張性という 2 つの理由があります。ソースからターゲットに直接変換するには、実際にはソースとターゲットのペアごとにアダプタとトランスフォーマを作成する必要があります。XML を中間形式として使用すると、ソースごとにアダプタを 1 つと、各デバイスごとにトランスフォーマを 1 つ定義するだけですみます。たとえば、2 つのコンテンツ・ソースと 3 つのターゲット・デバイスがあるとします。

- **XML を経由せずにソースからターゲットに変換する場合**。アダプタとトランスフォーマのペアが 6 つ、つまり、合計 12 のコンポーネントが必要になります。
- **XML 経由でソースからターゲットに変換する場合**。アダプタ 2 つとトランスフォーマ 3 つ、合計 5 つのコンポーネントだけですみます。

シンプル・リザルト DTD の使用

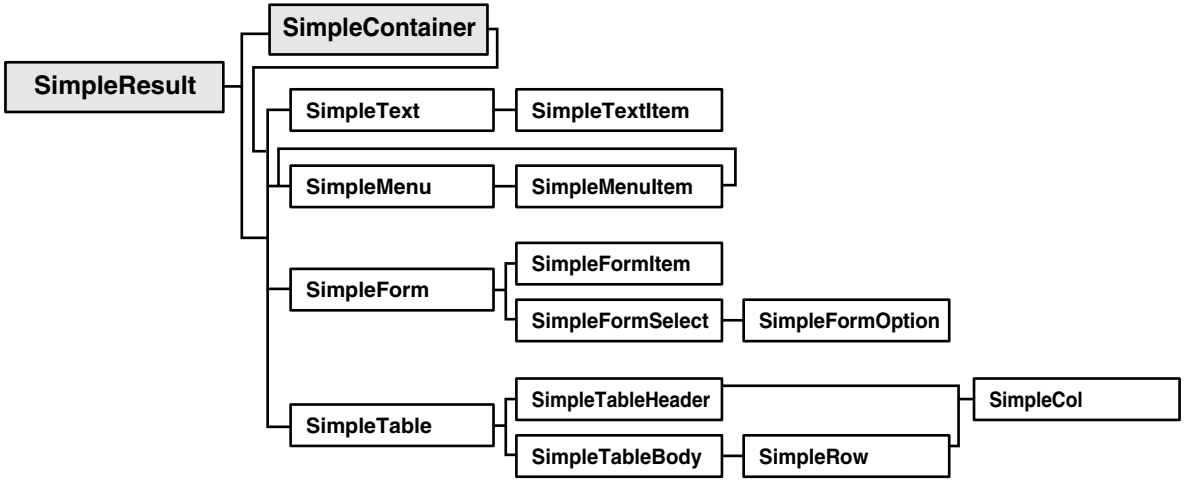
アダプタ出力を汎用的なものにするには、XML 形式にする必要があります。重要な点は、表示するデータ型やデバイスに関係なく表示できる XML 文書タイプを定義することです。ドキュメント・タイプは、Document Type Definition（DTD）で定義されます。DTD は、中に含めることのできる要素を記述して、XML 文書のクラスの文法を示すファイルです。

真に汎用的な中間データ形式を作成するために、Oracle9*i*AS Wireless Edition ではシンプル・リザルト DTD を使用しています。シンプル・リザルト DTD の各要素は、抽象ユーザー・インタフェースの各要素を表します。これには、次の要素が含まれます。

- テキスト項目
- メニュー
- フォーム
- 表

図 3-5 に、シンプル・リザルト DTD のコンテンツ・モデルを示します。

図 3-5 シンプル・リザルト DTD のコンテンツ・モデル



株価データの例に使用した要素を示す SimpleResult.dtd の一部を次に示します。

```
<!--
Entity:   "GENATTR" contains generic attributes for most elements.
Attribs:  "name" is the name of the element.
          "title" is the title of the element.
...
-->
<!ENTITY % GENATTR "
            name    CDATA #IMPLIED
            title   CDATA #IMPLIED
 ...        ">

<!--
Element:    "SimpleResult" is the result element.
Usage:      This element contains the result.
Children:   "SimpleText" is a text result.
 ...
-->

<!ELEMENT   SimpleResult ((SimpleContainer|SimpleText|SimpleMenu|
            SimpleForm|SimpleTable|SimpleImage|SimpleBreak)+)>
<!ATTLIST   SimpleResult %GENATTR;>
...
```

```
<!--
Element:    "SimpleText" for displaying one or more blocks of text.
Usage:      Used for plain text.
Children:   "SimpleTextItem" is a block of text.
-->

<!ELEMENT   SimpleText (SimpleTextItem+)>
<!ATTLIST   SimpleText %GENATTR;>

<!--
Element:    "SimpleTextItem" is a block of text
Usage:      Holds one block of text - normally a single paragraph.
Children:   "#PCDATA" is the actual text.
-->

<!ELEMENT   SimpleTextItem (#PCDATA)>
<!ATTLIST   SimpleTextItem %GENATTR;>

...

<!--
Element:    "SimpleForm" for displaying one or more input fields.
Usage:      As a data-entry form.
Children:   "SimpleFormItem" for each input field.
Attribs:    "target" is the link target for this form.
            "section" is the section identifier

***** WIDL アダプタの特殊なケース *****

-->
<!ELEMENT    SimpleForm ((SimpleFormItem|SimpleFormSelect)+)>
<!ATTLIST    SimpleForm %GENATTR;
             target  CDATA #REQUIRED
             section CDATA #IMPLIED>
<!--
Element:    "SimpleFormItem" is a single input item in a simple form.
Usage:      For getting input from a user.
Children:   "#PCDATA" contains pre-filled input from the server.

****** これにより、デフォルト属性がオーバーライドされます。*******

Attribs:    "default" provides a default value for optional fields.

*****  デフォルト値は、フィールドが空の場合にのみ使用する必要があります。
        「mandatory」はフォーム項目が必須であることを示します。
        「maxLength」では最大入力長を指定します。
```

```
-->
<!ELEMENT   SimpleFormItem (#PCDATA)>
<!ATTLIST   SimpleFormItem %GENATTR;
            default  CDATA #IMPLIED
            mandatory(yes|no) "no"
            maxLength CDATA #IMPLIED>

...
```

アダプタによるソース・コンテンツから DTD 要素へのマッピング

Oracle9*i*AS Wireless Edition アダプタでは、ソース・コンテンツが適切なシンプル・リザルト要素にマップされます。

- 入力バインドでは、フォームの <input> タグとサービス URL の変数を通じて、要求を完了するために必要なデータを指定します。
- 出力バインドは、リクエスタに戻される結果です。サービスとデバイス用に HTML の関連部分のみが選択されます。

たとえば、表 3-1 は、架空の StockData アダプタにより生成された入力フォームの XML（テキスト・ラベル、入力フィールドおよび送信ボタン）と結果ページ（証券コード、株価およびヘッドライン）を示しています。

表 3-1 StockData アダプタにより生成された入力ページと結果ページの XML

入力ページの XML	XML 結果ページ : 株価およびヘッドライン・ページ
<SimpleResult> <SimpleText> <SimpleTextItem name = "TickerField"> 証券コード : </SimpleTextItem> </SimpleText> <SimpleForm title="Input Form"> <SimpleFormItem name="Ticker"> </SimpleFormItem> <SimpleFormButton name="submitBtn"> 株価の取得 </SimpleFormButton> </SimpleForm> </SimpleResult>	<SimpleResult> <SimpleText title="Quote Results"> <SimpleTextItem name="Ticker"> ORCL </SimpleTextItem> <SimpleTextItem name="Price"> 90 3/8 </SimpleTextItem> </SimpleText> <SimpleText title="Headlines"> <SimpleTextItem name = "Headline1"> * オラクル株が大商い。 </SimpleTextItem> <SimpleTextItem name = "Headline2"> * NASDAQ は高値で引け。 </SimpleTextItem> <SimpleTextItem name = "Headline3"> * US 株が急騰。 </SimpleTextItem> </SimpleText> </SimpleResult>

サンプル・アダプタ・クラス

次のコード例に、Java によるアダプタの実装方法を示します。このコード例を使用して、カスタム・コンテンツ・ソース用の独自アダプタを作成できます。

「Oracle9iAS Wireless Edition アダプタの例 1: 初期メッセージへのユーザー名の表示」は単純ですが、初期メッセージにユーザー名が表示される完全なアダプタ実装です。

Oracle9*i*AS Wireless Edition アダプタの例 1: 初期メッセージへのユーザー名の表示

初期メッセージにユーザー名を表示するサービス用の単純なアダプタを考えます。入力は次のとおりです。

- 初期メッセージに使用する文字列を示す初期化パラメータ
- ユーザー名を示す入力パラメータ

例 2 のアダプタは、invoke メソッドと次のパッケージのメソッドを使用してシンプル・リザルト・ドキュメントを構築します。

- `org.w3c.dom.Element`
- `org.w3c.dom.Text`

invoke メソッドにより次の作業が実行されます。

1. ルート結果要素が作成されます。
2. SimpleText 要素が作成されます。タイトル属性が設定され、この要素がルート要素に追加されます。シンプル・リザルト DTD に定義されているように、SimpleTextItem は SimpleText の必須の子要素です。
3. 入力パラメータ値が取り出され、結果ドキュメントに追加されます。
4. 結果が戻されます。

このアダプタの実装は次のとおりです。

```
import org.w3c.dom.Element;
import org.w3c.dom.Text;
import oracle.panama.Argument;
import oracle.panama.Arguments;
import oracle.panama.ServiceRequest;
import oracle.panama.adapter.Adapter;
import oracle.panama.adapter.AdapterDefinition;
import oracle.panama.adapter.AdapterException;
import oracle.panama.adapter.AdapterHelper;

public class HelloAdapter implements Adapter {
   private boolean initialized = false;
```

```
 private String greeting = "Hello";
 public static final String GREETING = "greeting";
 public static final String NAME = "name";

 // Called once, when the adapter is instantiated.
 public void init (Arguments args) throws AdapterException {
    synchronized (this) {
       if(!initialized) {
          initialized = true;
          greeting = args.getInputValue( GREETING );
       }
    }
 }
public Element invoke (ServiceRequest sr)
                        throws AdapterException {
    Element result = XML.makeElement("SimpleResult");
    Element st = XML.makeElement("SimpleText");
    st.setAttribute ("title",
                "Oracle Portal-to-Go Server HelloAdapter Sample");
    result.appendChild (st);
    Element sti = XML.makeElement("SimpleTextItem");
    sti.setAttribute ("name", "message");
    sti.setAttribute ("title", "Portal-to-Go says:");
    st.appendChild (sti);
    // ServiceRequest sr contains input parameters (like NAME, below).
    String name = sr.getArguments().getInputValue(NAME);
    Text txt = XML.makeText( greeting + " " + name + "!");
    sti.appendChild (txt);
    return result;
 }
 // This method enables master services to determine
 // the initialization parameters used by the adapter.
 private AdapterDefinition initDef = null;
 public AdapterDefinition getInitDefinition() {
    if (initDef == null) {
       synchronized (this) {
          if (initDef == null) {
             initDef = AdapterHelper.createAdapterDefinition();
             initDef.createInit( Argument.SINGLE_LINE,
                                 GREETING,
                                 "Greeting phrase",
                                 null );
           }
       }
    }
    return initDef;
 }
```

```
   // This method defines the adapter&#146;s runtime input parameters.
   private AdapterDefinition adpDef = null;
   public AdapterDefinition getAdapterDefinition() throws AdapterException {
      if (adpDef == null ) {
         synchronized (this) {
            if (adpDef == null) {
                if (initDef == null)
                   throw new AdapterException ("Adapter is
                                not properly initialized");
                adpDef = initDef;
                adpDef.createInput( Argument.SINGLE_LINE,
                                    NAME,
                                    "Name to greet",
                                    null );
             }
         }
      }
      return adpDef;
   }
}
```

前述のアダプタが入力パラメータ「Dolly」で起動されると、次の XML の結果を戻します。

```
<SimpleResult>
   <SimpleText title="Oracle Portal-to-Go Server Hello Sample">
      <SimpleTextItem name="message" title="Portal-to-Go says:">
         Hello Dolly!
      </SimpleTextItem>
   </SimpleText>
</SimpleResult>
```

XML からターゲット・マークアップ言語への変換

Oracle9*i*AS Wireless Edition トランスフォーマでは、XML 文書がターゲット・デバイス用のマークアップ言語に変換されます。SimpleResult などの汎用的な内部 XML 形式を使用して情報を表現することで、各クライアント・デバイスの UI 機能を全面的に活用できます。

トランスフォーマでは、SimpleResult DTD を使用して抽象 UI 要素がターゲット形式にマップされます。トランスフォーマは、必要な処理に応じて Java または XSL-T で実装できます。

- **XSL -T。** XSL スタイルシートには、複雑なパターン一致ロジックと結果処理ロジックを組み込むことができます。通常は、ターゲット形式のマークアップ・タグなど、リテラルの結果要素が含まれます。Oracle9*i*AS Wireless Edition では、デフォルトで XSL スタイルシートが使用されます。「Oracle9iAS Wireless Edition の XSL スタイルシート・トランスフォーマの例 1: シンプル・リザルト・ドキュメントからプレーン・テキストへの変換」を参照してください。

- **Java**。Java を使用すると、VOX デバイス用の Repeat 機能など、画面に書き込むデバイスには不要なデバイス固有の動作を追加できます。「Oracle9iAS Wireless Edition の Java トランスフォーマの例 1: シンプル・リザルト要素から別の形式への変換」を参照してください。

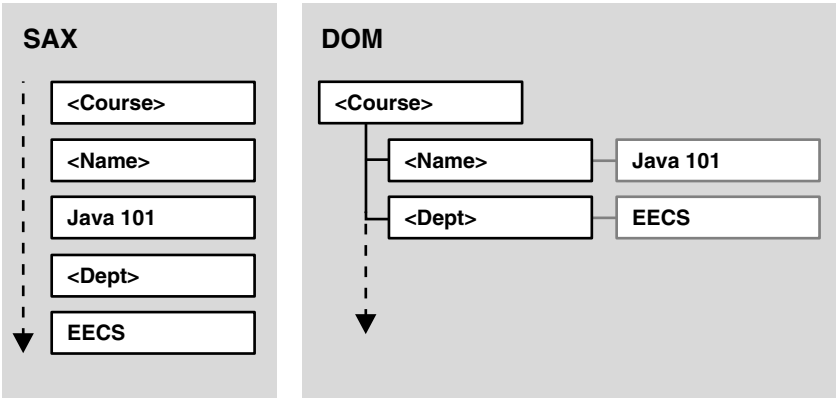
Oracle9*i*AS Wireless Edition: Java トランスフォーマ

次のどちらかのインタフェースを使用して Java トランスフォーマを作成できます。

- ドキュメント・オブジェクト・モデル（Document Object Model: DOM）・インタフェース。ツリー・ベースのドキュメント・オブジェクト・モデルを操作します。
- Simple API for XML（SAX）インタフェース。解析プロセス中にイベントと直接相互作用します。

図 3-6 に、この 2 つのインタフェースを示します。

図 3-6 DOM インタフェースと SAX インタフェース



Oracle9*i*AS Wireless Edition には、シンプル・リザルト・ドキュメントをプレーン・テキストに変換する Java トランスフォーマが組み込まれています。トランスフォーマは、変換後のドキュメント内でマークアップ・タグを作成するのではなく、改行やタブなどの単純なテキスト書式要素を適用します。

Oracle9*i*AS Wireless Edition の Java トランスフォーマの例 1: シンプル・リザルト要素から別の形式への変換

この例は単純ですが、シンプル・リザルト要素を別の形式に変換する方法を示しています。

```
package oracle.panama.core.xform;
import org.w3c.dom.NodeList;
import org.w3c.dom.Element;
import oracle.panama.PanamaException;
import oracle.panama.core.LogicalDevice;
import oracle.panama.core.Service;
import oracle.panama.Arguments;
import oracle.panama.core.parm.PanamaRequest;
import oracle.panama.core.parm.AbstractRequest;

public class SimpleResultToText implements Transform {
    public SimpleResultToText() {}

    private String format(Element el) {
        if (el == null) {
            return "";
        }
        StringBuffer buf = new StringBuffer();
        String name = el.getTagName();
        if (name != null && name.length() > 0) {
            buf.append(name);
            buf.append(": ");
        }
        buf.append(el.getNodeValue());
        return buf.toString();
    }

    public String transform(Element element, LogicalDevice device)
                                  throws PanamaException {
        PanamaRequest req = AbstractRequest.getCurrentRequest();
        Service service = req.getService();
        StringBuffer buf =
          new StringBuffer((service == null) ? "" : service.getName());
        NodeList list = element.getElementsByTagName("*");
        Element el;
        String tag;
        boolean newRow = false;
        for (int i = 0; i
            el = (Element)list.item(i);
            tag = el.getTagName();
            if (tag.equals("SimpleRow")) {
                newRow = true;
```

```
                buf.append("¥n");
            } else if (tag.equals("SimpleCol")) {
                if (!newRow) {
                    buf.append("¥t");
                } else {
                    newRow = false;
                }
                buf.append(format(el));
            } else if (tag.equals("SimpleText") ||
                       tag.equals("SimpleForm") ||
                       tag.equals("SimpleMenu")) {
                newRow = true;
                buf.append("¥n");
            } else if (tag.equals("SimpleTextItem") ||
                       tag.equals("SimpleFormItem") ||
                       tag.equals("SimpleMenuItem")) {
                if (!newRow) {
                    buf.append("¥n");
                } else {
                    newRow = false;
                }
                buf.append(format(el));
            }
        }
        return buf.toString();
    }
}
```

Oracle9*i*AS Wireless Edition: XSL スタイルシート・トランスフォーマ

XSL スタイルシートは、他の XML 文書の処理ルールを指定する XML 文書です。Java トランスフォーマと同様に、XSL スタイルシートは特定の DTD に固有のものであり、その DTD で宣言されたすべての要素を処理する必要があります。ソース・ドキュメント内で要素が見つかると、その要素に対して定義されているルールに従ってコンテンツがフォーマットされます。

Oracle9*i*AS Wireless Edition の XSL スタイルシート・トランスフォーマの例 1: シンプル・リザルト・ドキュメントからプレーン・テキストへの変換

この XSL トランスフォーマの例は、Oracle9*i*AS Wireless Edition の初期リポジトリに含まれており、前述の Java トランスフォーマの XSL バージョンです。このトランスフォーマでは、シンプル・リザルト・ドキュメントがプレーン・テキストに変換されます。

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">
   <xsl:template match="/">
      <xsl:apply-templates></xsl:apply-templates>
   </xsl:template>
      <xsl:template match="SimpleTextItem | SimpleFormItem | SimpleMenuItem">
      <xsl:text>
      </xsl:text>
      <xsl:value-of select="."></xsl:value-of>
   </xsl:template>
   <xsl:template match="SimpleRow">
      <xsl:text></xsl:text>
      <xsl:for-each select="./SimpleCol">
         <xsl:text></xsl:text>
         <xsl:value-of select="."></xsl:value-of>
      </xsl:for-each>
   </xsl:template>
</xsl:stylesheet>
```

この例の XSL スタイルシートでは、次の作業が実行されます。

1. パターン一致セマンティクスを使用してシンプル・リザルト要素が選択されます。たとえば、要素「/」はドキュメントのルート要素と一致します。
2. `apply-templates` を使用して、その要素のコンテンツが処理されます。
3. ソース要素ツリーを下位へたどって各副要素が選択され、処理されます。`value-of` や `for-each` などの文字命令では、一致する要素のコンテンツが操作されます。
   - `value-of` 要素では、その要素の実際のコンテンツが抽出されます。
   - `for-each` 要素では反復処理が適用されます。

一意のトランスフォーマを必要とする各マークアップ言語

一意のマークアップ言語には、それぞれ一意のトランスフォーマが必要です。株価データの例で、PDA と携帯電話に異なるマークアップ言語（Tiny HTML と WML）が使用されているため、2 つのトランスフォーマが必要であるとします。この 2 つのトランスフォーマを構築すると、シンプル・リザルト XML を生成する任意のアダプタからのコンテンツを処理できます。

表 3-2 に、アダプタの SimpleResult XML コードと、2 つのトランスフォーマにより生成されるマークアップ言語を示します。

- PDA 用の Tiny HTML。株価とヘッドラインの両方をフォーマットして表示できます。
- 携帯電話用の WML。株価のみ表示します。

表 3-2 一意のトランスフォーマを使用したアダプタのシンプル・リザルト XML の変換

アダプタのシンプル・リザルト XML	一意のトランスフォーマ
<SimpleResult> <SimpleText title="Quote"> <SimpleTextItem name="Ticker"> ORCL </SimpleTextItem> <SimpleTextItem name="Price"> 90 3/8 </SimpleTextItem> </SimpleText> <SimpleText title="Headlines"> <SimpleTextItem name = "Headline1"> * オラクル株が大商い。 </SimpleTextItem> <SimpleTextItem name = "Headline2"> * NASDAQ は高値で引け。 </SimpleTextItem> <SimpleTextItem name = "Headline3"> * US 株が急騰。 </SimpleTextItem> </SimpleText> </SimpleResult>	**PDA 用の Tiny HTML** <html> Quote Ticker: ORCL Price: 90 3/8 Headlines: * オラクル株が大商い。 * NASDAQ は高値で引け。 * US 株が急騰。 </html> **携帯電話用の WML** <?xml version "1.0"?> <!DOCTYPE WML PUBLIC "-//WAPFORUM/DTD WML 1.0//EN" "http://www.wapforum.org.DTD.wml.xml"> <WML> <CARD NAME="QUOTE_CARD" TITLE="Quote Card"> ORCL 90 3/8 </CARD> </WML>

Oracle9*i*AS Wireless Edition のスタイルシート・トランスフォーマの例 2: WML 1.1 トランスフォーマのスタイルシートのカスタマイズ

Phone.com ブラウザでの WML のブラウズ

Phone.com ブラウザを使用する場合は、先に進む前にナビゲーション・モデルで [Link] オプションを選択する必要があります。スタイルシートをカスタマイズすると、この動作を変更できます。たとえば、WML 1.1 トランスフォーマ・スタイルシートに次のコードを追加します。

```
| The SimpleForm Mapping
+-->
<xsl:template match="SimpleForm">
<p>
 <xsl:variable name="theTarget">
 <xsl:value-of select="@target"/>
 <xsl:for-each select="SimpleFormItem | SimpleFormSelect">
 <xsl:text>&#38;</xsl:text>
 <xsl:value-of select="@name"/>
 <xsl:text>=$(</xsl:text>
 <xsl:value-of select="@name"/>
 <xsl:text>)</xsl:text>
</xsl:for-each>
</xsl:variable>
<xsl:apply-templates/>

<!-- Ensure [Link] is selected -->
<select>
<option>
<onevent type="onpick">
<go href="{$theTarget}"/>
</onevent>
<xsl:choose>
<xsl:when test="boolean(@submit)">
<xsl:value-of select="@submit"/>
</xsl:when>
<xsl:otherwise>Submit</xsl:otherwise>
</xsl:choose>
</option>
</select>
</p>

<!-- Ensure [Link] is selected ends -->
<!--
<a href="{$theTarget}">
  <xsl:choose>
```

```
    <xsl:when test="boolean(@submit)">
    <xsl:value-of select="@submit"/>
    </xsl:when>
  <xsl:otherwise>Submit</xsl:otherwise>
  </xsl:choose>
</a>
<br/>
</p>
-->
</xsl:template>
```

Oracle9*i*AS Wireless Edition のスタイルシート・トランスフォーマの例 3: XSL Java 拡張機能

SimpleResult XML は、SimpleDB と呼ばれる新規要素の追加により拡張されています。この要素は、データベース上で INSERT、UPDATE および DELETE 文を実行したり、Oracle データベース上で PL/SQL を実行するために使用します。

たとえば、この機能を、コンテンツ・プロバイダがコストを定義する拡張請求システムに使用できます。また、すべてのデータベース標準機能（UTL_SMTP、UTL_FILE...）を使用して、マスター・サービス機能を拡張することもできます。

そのために、メソッド `processDB()` を含む新規の Java クラス `oracle.panama.core.xform.MyXslExtension` が作成されています。

この機能をテストするには、`MyXslExtension.class` を次のディレクトリにコピーします。

%ORACLE_HOME%/panama/server/classes/oracle/panama/core/xform

さらに、デバイス・トランスフォーマに次のヘッダーを追加します。

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

xmlns:p2g="http://www.oracle.com/XSL/Transform/java/oracle.panama.core.xform.XSLJava
" exclude-result-prefixes="p2g"

xmlns:myxsl="http://www.oracle.com/XSL/Transform/java/oracle.panama.core.xform.MyXsl
Extension">
```

次の文をトランスフォーマ本体に追加します。

```
<!-- Execute SimpleDB-->
<xsl:template match="SimpleDB">
      <xsl:value-of select="p2g:processDB(.)"/>
</xsl:template>
```

または次の文を追加します。

```
<!-- Bypass SimpleDB-->
<xsl:template match="SimpleDB">
      <xsl:apply-template/>
</xsl:template>
<xsl:template match="SimpleDBItem"/>
```

SimpleDB 要素を使用して結果トランスフォーマにより生成された SimpleResult の例を次に示します。

```
<SimpleResult>
<SimpleContainer>
  <SimpleText>
     <SimpleTextItem>Symbol : ORCL</SimpleTextItem>
     <SimpleTextItem>Value (USD): 76 15/32</SimpleTextItem>
     <SimpleTextItem>Value (HKD): 592.59</SimpleTextItem>
  </SimpleText>
  <SimpleDB jdbc="jdbc:oracle:thin:@hkpsnt3.hk.oracle.com:1521:orcl" user="scott"
password="tiger">
    <SimpleDBItem type="SQL">
       insert into quote values ('orcl','ORCL','76 15/32',null, null)
    </SimpleDBItem>
    <SimpleDBItem type="PLSQL">
      begin
       insert into quote values ('orcl','ORCL','76 15/32',null,sysdate);
      end;
    </SimpleDBItem>
  </SimpleDB>
</SimpleContainer>
</SimpleResult>
```

これは単なる一例であり、他の Java メソッドを作成して独自の拡張機能を追加できます。

MyXslExtension.java

```
// Allen M.K. YAP - Sales Consultant
// Oracle System Hong Kong Ltd.
// Email : Manh-Kiet.Yap@oracle.com

// File : MyXslExtension.java
// Date : 28/09/2000

package oracle.panama.core.xform;

import oracle.panama.core.xml.XML;
import oracle.panama.core.admin.L;
import org.w3c.dom.*;
```

```
import java.sql.*;

public class MyXslExtension
{

    private static String getTextValue(Element element)
    {
        Node node = element.getFirstChild();
        if(node != null && node.getNodeType() == 3)
            return node.getNodeValue();
        else
            return "";
    }

    public static String processDB (NodeList nlist) {

        Element element  = (Element) nlist.item(0);

        String out = new String();
        boolean fail = false;
        String jdbc_conn = element.getAttribute("jdbc");
        String jdbc_user = element.getAttribute("user");
        String jdbc_pass = element.getAttribute("password");
        try {
          Connection conn = DriverManager.getConnection (jdbc_conn,jdbc_user,jdbc_
pass );

          NodeList nodelist = element.getElementsByTagName("SimpleDBItem");
          for(int i = 0; i < nodelist.getLength(); i++)
          {
            Element st = (Element)nodelist.item(i);

            if ( (element.getAttribute("type")).equals("SQL")) {//SQL
              try {
                PreparedStatement pstmt=null;
                pstmt = conn.prepareStatement(getTextValue(st));
                pstmt.executeUpdate();
                pstmt.close();
              }
              catch (SQLException e) {
               L.e("SQL fails : "+e);
                out.concat(getTextValue(st)+"¥nSQL fails :¥n "+e+"¥n");
                fail=true;
              }
            }
            else { //PLSQL
              try {
```

```
                CallableStatement plstmt = conn.prepareCall (getTextValue(st));
              plstmt.execute();
              plstmt.close();
            }
            catch (SQLException e) {
                fail=true;
               L.e("PLSQL fails : "+e);
                out.concat(getTextValue(st)+"¥nPLSQL fails :¥n "+e+"¥n");
            }
           }
        }

        conn.commit();
        conn.close();
        if (fail) return (out);
        else return ("DB actions successfully completed.");
      }
      catch (SQLException e) {
        L.e(e);
        return ("DB actions failed");
      }

    }

    public MyXslExtension()
    {
    }
}
```

Oracle9*i*AS Wireless Edition の事例 1: オンライン・ドラッグストアのアクセスの拡張

オンライン・ドラッグストアで、Oracle9*i*AS Wireless Edition のワイヤレス・インターネット・ソフトウェアを使用して顧客へのアクセスを拡張し、携帯デバイス経由でオンライン・ドラッグストアに 24 時間いつでもアクセスできるようにしているとします。

Oracle9*i*AS Wireless Edition により、既存のインターネット・サイトが携帯ワイヤレス・デバイスへと拡張されます。この場合、Oracle9*i*AS Wireless Edition は、Oracle Internet Platform に構築されたオンライン・ストアと統合されます。このソリューションにより、顧客は事実上どこからでもドラッグストアの全製品ラインを購入できます。

Oracle9*i*AS Wireless Edition はインターネット・コンテンツ・デバイスに依存しないため、携帯情報端末（PDA）、ワイヤレス・アプリケーション・プロトコル（WAP）による電話またはポケットベルなど、インターネットに接続しているすべてのデバイスから、PC 用に設計された既存のコンテンツにアクセスできます。

Oracle9*i*AS Wireless Edition の事例 2: バンキング・サービスの拡張

銀行は、携帯電話を通じて顧客にオンライン・サービスを提供するようになり、オラクル社のワイヤレス・インターネット・サーバー製品である Oracle9*i*AS Wireless Edition を使用しています。

銀行の顧客は、WAP（ワイヤレス・アプリケーション・プロトコル）対応の電話または標準的な GSM 電話を通じて、金融情報、最寄りの支店を探すための検索機能、ローン返済額計算機能、イベント・カレンダおよび天気予報にアクセスできます。

また、銀行はワイヤレス・インターネット・サービスの内容にトランザクション型のバンキング・サービスを追加しつつあります。これにより、銀行の新たな WAP プラットフォームでは、顧客の携帯電話を通じて銀行のオンライン情報およびサービスにもアクセスできるようになります。

Oracle9*i*AS Wireless Edition の事例 3: オンライン・オークション・サイト

オンライン・オークション・サイトは、携帯電話、PDA または他のモバイル・デバイスからショッピングするためのオプションを提供することで、アクセス可能性と可用性を顧客へと拡張できます。

4

XML および XSQL を使用した表示のカスタマイズ : Flight Finder

この章の内容は、次のとおりです。

XML Flight Finder サンプル・アプリケーション : 概要

XML Flight Finder は、航空機のフライト・データをフェッチし、結果をクライアント・デバイス（PC、携帯電話、PDA など）に合せてカスタマイズします。このアプリケーションは Oracle9*i* に組み込まれており、Oracle XSQL Servlet を利用するため、SQL 問合せの発行と出力形式の定義に XML、XSL および XSQL テキスト・ファイルを使用できます。Java プログラミングは不要であり、コンパイルが必要なコードもありません。また、構築、カスタマイズおよびメンテナンスも簡単です。

XML Flight Finder のソース・コードをダウンロードし、調べて変更を加えてください。また、Flight Finder での Oracle XML 製品およびテクノロジの使用方法に関するトピックを読み、ページにあるリンクを使用してサイトにアクセスし、PC 上で携帯電話をシミュレートできるソフトウェアなどをダウンロードできます。

この情報とアプリケーションは、次の URL でダウンロードできます。

- http://otn.oracle.com/sample_code/index.htm

XML Flight Finder の実行要件

XML Flight Finder アプリケーションを構築して実行するための要件は、次のとおりです。

- Java 1.2 以上。
- Oracle8*i* リリース 8.1.5 以上。
- データベースと互換性のあるバージョンの SQL*Plus。
- Oracle XSQL Servlet（Web-to-Go personal Web server for Windows NT を含む）。最新バージョンを OTN Japan からダウンロードしてください。
- Flight Finder ファイル。fly.zip をダウンロードしてください。
- Web ブラウザ。最善の結果を得るには、XML を処理できるブラウザ（Internet Explorer 5 など）を使用してください。
- コンピュータ上で他のデバイス（携帯電話など）をシミュレートするソフトウェア（オプション）。
- Apache または Oracle9*i* Application Server（オプション）。Flight Finder は、Web-to-Go さえあれば Windows NT マシンで実行できますが、Apache または Oracle9*i* Application Server で実行することもできます。

Flight Finder の動作

Flight Finder は、都市間のフライトに関する情報をデータベースに問い合せて、エンド・ユーザーのデバイス用にカスタマイズされた形式で結果を戻します。Flight Finder は Oracle9*i* 上に構築されており、次の製品とテクノロジを使用します。

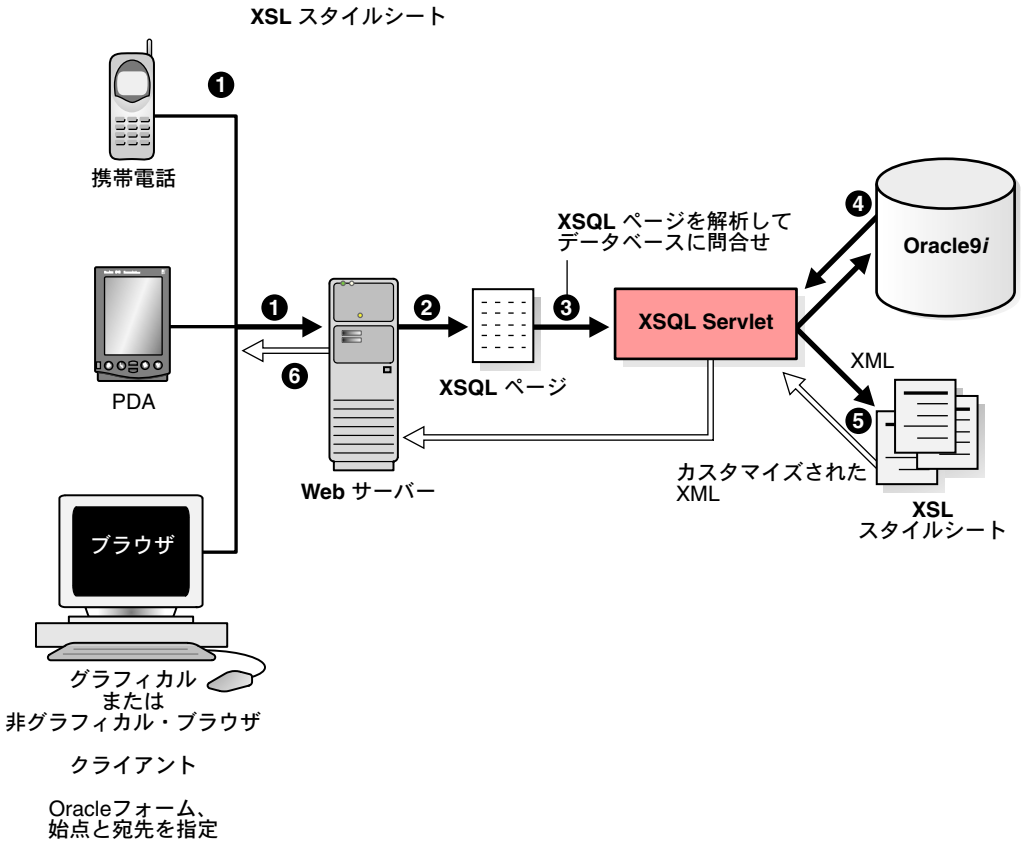
- SQL。ビジネス・データへのアクセスの標準です。
- Oracle XSQL Servlet。XSQL ページで定義されている問合せを処理します。XSQL ページは、SQL コードを含む XML 文書です。XSQL Servlet は結果セットを XML として出力します。
- XSLT。ターゲット・デバイスに合せて XML を変換するためのオープンな標準を定義します。

この章では、Flight Finder アプリケーションの実装方法について説明します。ここで説明するテクニックは、任意の Web ベース・アプリケーションで使用できます。Web ベース・アプリケーションでは、次の作業が実行されます。

- Web 上のクライアント・デバイスから要求を受信します。
- データベースのコンテンツを複数のデバイスに配信します。
- 複数のデバイスからの入力をデータベースに書き込みます。

図 4-1 に、Flight Finder の動作を示します。

図 4-1 XML Flight Finder



1. エンド・ユーザーが、サポートされているクライアント・デバイスを使用してフォームに入力し、始点と宛先を指定します。エンド・ユーザーがフォームを送信したときに実行される XSQL ページは、そのフォームのソース・コードで指定されています。
2. Web サーバーが XSQL ページで XSQL Servlet を起動します。
3. XSQL Servlet は、XSQL ページを解析してデータベースに問い合せます。
4. データベースは問合せ結果を戻し、XSQL Servlet はそれを XML 文書に変換します。
5. XSQL Servlet は、エンド・ユーザーのクライアント・デバイスに適した XSL スタイルシートを XML に適用して変換します。
6. Web サーバーが、カスタマイズされたドキュメントをクライアントに戻します。

Oracle9*i* では、Oracle XML コンポーネントと、それを使用してデータベース内で構築されたアプリケーションを実行できます。小型のデータベースを必要とするデバイスやアプリケーションの場合は、XML データの格納と取出しに Oracle9*i* Lite を使用できます。また、これらのコンポーネントは、Oracle9*i* Application Server などの中間層やクライアントでも実行できます。

Flight Finder によるデータベースの問合せ ー 結果から XML への変換

この項では、Flight Finder がデータベースに問い合せて、結果セットを XML 文書に変換する方法について説明します。Flight Finder アプリケーションは、次のように XSQL ページと XSL スタイルシートで構成されています。

- XSQL ページでは問合せが定義されます。
- XSL スタイルシートでは問合せの結果がフォーマットされます。

Flight Finder に Java コードは含まれておらず、処理は Oracle XSQL Servlet に委譲されます。

Flight Finder では、フライト・データが 2 つの表 AIRPORTS および FLIGHTS に格納されます。

- AIRPORTS では、CODE 列が主キーです。
- FLIGHTS では、CODE 列が主キーで、CODE_FROM および CODE_TO 列は AIRPORTS.CODE を参照する外部キーです。

次の SQL コードは、この 2 つの表の構造を示しています（太字の列名は主キーで、イタリックの列名は外部キーです）。

```
create table airports
(
 code varchar2(3),
 name varchar2(64)
 );

create table flights
(
code varchar2(6),
 code_from varchar2(3),
       code_to varchar2(3),
       schedule date,
       status varchar2(1),
        gate varchar2(2)
       );
```

XSQL Servlet を使用した問合せの処理と XML としての結果の出力

XSQL Servlet は SQL 問合せを処理し、結果セットを XML として出力します。

XSQL Servlet は Java サーブレットとして実装されており、入力として XSQL ページを使用します。これは、埋込み SQL 問合せを含む XML ファイルです。その多数の操作には、XML Parser for Java および XML- SQL Utility for Java が使用されます。

たとえば、次のコードは `fly.xsql` から抜粋したものです。これは XML であり、XSQL Servlet で解析される特殊な <xsql> タグが含まれています。

`flightFinderResult` タグでは、問合せのパラメータに値を割り当てる構造が定義されています。このタグでは、xsql キーワードを定義するためのネームスペースも識別され、XSQL Servlet に対して（事前定義済みの）データベース接続 fly を使用するように指定します。

このコードでは、`<xsql:query>` タグを使用して問合せを定義しています（XSQL Servlet のダウンロード内容には、各 XSQL タグの構文とオプションの説明に関するヘルプ・システムも含まれています）。また、問合せ文の本体では、他に 2 つのパラメータ（FROM および TO）を使用して、エンド・ユーザーが選択した都市名が格納されます。

注意： XSQL ページでは、XSLT 構文 {@param} を使用してパラメータが示されます。

図 4-2 に、Flight Finder のブラウザ・フォームと、このフォームを使用して FROM 情報（Los Angeles）と TO 情報（San Francisco）を入力する方法を示します。

図 4-2 XSQL Servlet を使用した問合せの処理と XML としての結果の出力 : Flight Finder ブラウザ・フォームへの FROM および TO の入力

```
<?xml version="1.0"?>
   ...
<flightFinderResult xmlns:xsql="urn:oracle-xsql" connection="fly" lang="english">
     <xsql:set-stylesheet-param name="lang" value="{@lang}"/>
     <xsql:query tag-case="upper">
       <![CDATA[
       select F.code, F.code_from, A1.name as "depart_airport",
           F.code_to, To_char(F.schedule, 'HH24:MI') as "Sched",
           A2.name as "arrive_airport",
           Decode(F.Status, 'A', 'Available', 'B', 'Full', 'Available')
              as "Stat",F.Gate
              from flights F, airports A1, airports A2
              where to_number(To_Char(F.schedule, 'HH24MI')) >
                    to_number(To_Char(sysdate, 'HH24MI')) and
                    F.code_from = '{@FROM}' and F.code_to = '{@TO}' and
                    F.code_from = A1.code and F.code_to = A2.code
         ]]>
     ...
     </xsql:query>
    ...
</flightFinderResult>
```

この後のリストに次の URL の処理により XSQL Servlet から戻される XML の一部を示します。この URL には大 / 小文字区別があります。

```
http://localhost:7070/fly.xsql?FROM=LAX&TO=SFO&xml-stylesheet=none
```

この URL はサーバーに対して XSQL Servlet を起動し、スタイルシートを適用せずにファイル `fly.xsql` を処理し、LAX（Los Angeles）から SFO（San Francisco）へのフライトを検索するように指示します（スタイルシートには、データベースからのエラー・メッセージなどの RAW XML コードが表示されるため、便利なデバッグ・テクニックです）。

結果は、結果セット内の行のデータを含む XML 文書です（次の抜粋は 1 行目のみを示しています）。

タグ ROWSET および ROW は、XSQL Servlet で定義されます。行セット内の各行のタグ（CODE、CODE_FROM および DEPART_AIRPORT など）は、データベース表の列名から取り込まれます。

```
<?xml version="1.0" ?>
       <flightFinderResult lang="english">
             <ROWSET>
                <ROW NUM="1">
                      <CODE>OA0307</CODE>
                      <CODE_FROM>LAX</CODE_FROM>
                      <DEPART_AIRPORT>Los Angeles</DEPART_AIRPORT>
                      <CODE_TO>SFO</CODE_TO>
                      <SCHED>12:04</SCHED>
```

```
            <ARRIVE_AIRPORT>San Francisco</ARRIVE_AIRPORT>
            <STAT>Available</STAT>
            <GATE>05</GATE>
          </ROW>
        ..
      </ROWSET>
     ...
  </flightFinderResult>
```

XML 文書には、データとそれを記述するタグが含まれていますが、データを表示用にフォーマットする方法に関する情報は含まれていません。これは一見すると制限のようですが、実際には機能であり、XML に柔軟性をもたらしています。XML 文書形式のデータは、必要な方法でフォーマットできます。

スタイルシートを使用した XML のフォーマット

Flight Finder は、XSLT 変換を適用し、XML の結果をエンド・ユーザーのクライアント・デバイスに適した形式で表示します。この項では、このプロセスについて説明します。

XML、XSLT および XSQL Servlet の関係の概要は、http://otn.oracle.com/tech/xml の URL を使用して Oracle Technology Network（OTN）の「XSQL Pages and XSQL Servlet Release Notes」を参照してください。

単一のスタイルシート、単一のターゲット・デバイス

Flight Finder では、XSL スタイルシートを使用して、問合せ結果を表す XML 文書がフォーマットされます。スタイルシート自体は、別の XML 文書のノードの処理方法を指定する XML 文書です。処理に関する指示は、テンプレートと呼ばれる構造で定義され、スタイルシートでは、選択されたノードにこれらのテンプレートが適用され、ドキュメントがフォーマットされます。

たとえば、前述の XML 文書には、ROWSET、ROW、CODE などのノードが含まれています。次のコード（flyHTMLdefault.xsl から抜粋）は、スタイルシートが ROWSET 内の ROW ごとに CODE、DEPART_AIRPORT および ARRIVE_AIRPORT の各ノードを選択し、テンプレートを適用して出力をフォーマットする方法を示しています。

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version='1.0'>
...
<xsl:template match="/">
<html>
...
  <xsl:for-each select="flightFinderResult/ROWSET/ROW">
    <tr>
      <td><xsl:apply-templates select="CODE"/></td>
       <td><xsl:apply-templates select="DEPART_AIRPORT"/></td>
```

```
        <td><xsl:apply-templates select="ARRIVE_AIRPORT"/></td>
         ...
     </tr>
     </xsl:for-each>
     ...
  </html>
 </xsl:template>
<xsl:template match="CODE">Fly Oracle Airlines <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="DEPART_AIRPORT">Leaving <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="ARRIVE_AIRPORT">
 for <xsl:value-of select="."/>
</xsl:template>
...
</xsl:stylesheet>
```

この例では、フォーマットは単純で、単に各ノードのコンテンツに文字列が付加されます。たとえば、XSLT プロセッサが CODE ノードにアクセスすると、そのノードの値に文字列「Fly Oracle Airlines」を付加します。その結果、HTML は次のようになります。

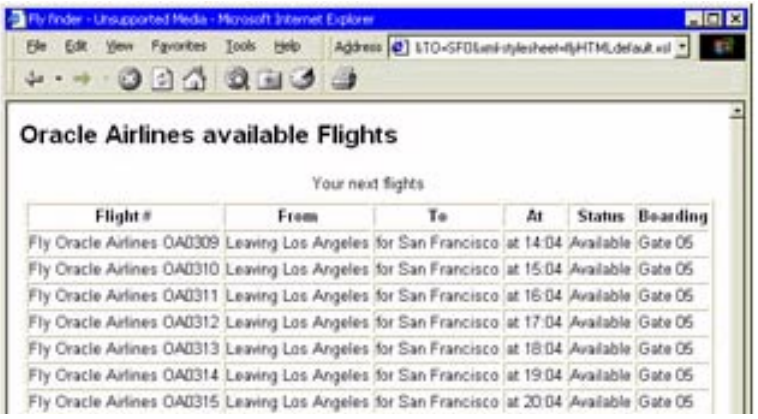
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
...
<TR>
 <TD>Fly Oracle Airlines OA0309</TD>
 <TD>Leaving Los Angeles</TD>
 <TD>for San Francisco</TD>
 ...
</TR>
...
</HTML>
```

ブラウザで、次の URL を入力します。

http://localhost:7070/fly/fly.xsql?FROM=LAX&TO=SFO&xml-stylesheet=flyHTMLdefault.xsl

図 4-3 に、XML にスタイルシートが適用された後のブラウザでの表示結果を示します。

図 4-3 Flight Finder: スタイルシートで XML をフォーマットした後の結果



多数のスタイルシート、多数のターゲット・デバイス

XSL スタイルシートは、複数のデバイス、言語およびユーザー・インタフェースにとってキーとなるものです。XSQL ページの最上部に複数の `<?xml-stylesheet?>` タグを挿入し、各タグでユーザー・エージェントを XSL スタイルシートと対応付けるメディアおよび href 属性を定義できます（HTTP 要求には、要求を出したデバイスを識別するユーザー・エージェント・ヘッダーが含まれています）。メディア属性のない処理指示は、すべてのユーザー・エージェントと一致するため、代替あるいはデフォルトとして使用できます。

たとえば、次の XML コードは fly.xsql から抜粋したものです。このコードには、スタイルシート `flyVox.xsl` を Motorola Voice Browser エージェントにマップするタグや、スタイルシート `flyPP.xsl` を HandHTTP（Palm Pilot）エージェントにマップするタグなど、複数の `<?xml-stylesheet?>` タグが含まれています。

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" media="MSIE 5.0" href="flyHTML.xsl"?>
<?xml-stylesheet type="text/xsl" media="Motorola Voice Browser" href="flyVox.xsl"?>
<?xml-stylesheet type="text/xsl" media="UP.Browser" href="flyWML.xsl"?>
<?xml-stylesheet type="text/xsl" media="HandHTTP" href="flyPP.xsl"?>
<?xml-stylesheet type="text/xsl" href="flyHTMLdefault.xsl"?>

<flightFinderResult xmlns:xsql="urn:oracle-xsql" connection="fly" lang="english">
<xsql:stylesheet-param name="lang" value="{@lang}"/>
<xsql:query tag-case="upper">
...
</xsql:query>
...
</flightFinderResult>
```

次の 2 つのリストに、Palm Pilot（flyPP.xsl）と音声ブラウザ・デバイス（flyVox.xsl）について、それぞれ結果セット行を 1 つずつフォーマットする XSLT コードを示します。

flyPP.xsl からの XSLT コード :

```
...
 <xsl:for-each select="flightFinderResult/ROWSET/ROW">
   <tr>
     <td>
       <a>
         <xsl:attribute name="href">
           #<xsl:value-of select="CODE"/>
         </xsl:attribute>
         <b><xsl:value-of select="CODE"/></b>
       </a>
     </td>
     <td><xsl:apply-templates select="SCHED"/></td>
     <td><xsl:apply-templates select="GATE"/></td>
   </tr>
 </xsl:for-each>
 ...
   <xsl:template match="CODE">
     <xsl:value-of select="."/>
   </xsl:template>
   <xsl:template match="SCHED">
     at <b><xsl:value-of select="."/></b>
   </xsl:template>
   <xsl:template match="GATE">
     gate <b><xsl:value-of select="."/></b>
   </xsl:template>
 ...
```

flyVox.xsl からの XSLT コード :

```
...
<xsl:for-each select="flightFinderResult/ROWSET/ROW">
 <step><xsl:attribute name="name">
 step<xsl:value-of select="position()"/>
  </xsl:attribute>
   <prompt>
     <xsl:apply-templates select="CODE"/>
     <xsl:apply-templates select="SCHED"/>,
     <xsl:text>Do you take that one?</xsl:text>
   </prompt>
   <input type="OPTIONLIST" name="FLIGHT">
   <xsl:choose>
```

```
    <xsl:when test="position() = @NUM">
     <option>
      <xsl:attribute name="next">
       #<xsl:value-of select="CODE"/>
      </xsl:attribute>
      <xsl:text>Yes</xsl:text>
     </option>
      <xsl:if test="position() &lt; last()">
     <option>
      <xsl:attribute name="next">#step<xsl:value-of select="position() + 1"/>
      </xsl:attribute>
      <xsl:text>Next</xsl:text>
     </option>
    </xsl:if>
    <xsl:if test="position() &gt; 1">
     <option>
      <xsl:attribute name="next">#step<xsl:value-of select="position() - 1"/>
      </xsl:attribute>
      <xsl:text>Previous</xsl:text>
     </option>
    </xsl:if>
   </xsl:when>
  </xsl:choose>
 </input>
</step>
</xsl:for-each>
...
```

出力のローカライズ

ポータル（index.html）を通じて Flight Finder を起動するときに、プロンプトとラベルの言語を選択できます。

Flight Finder では、英語、フランス語、スペイン語およびドイツ語がサポートされます。そのために、エンド・ユーザーの選択言語を識別するパラメータが HTML から XSQL へ、XSQL から XSL へと渡され、翻訳済みメッセージのファイルから適切なテキストが選択されます。たとえば、アプリケーションがユーザーの使用言語（フランス語）を追跡し、その言語によるラベルを選択する方法の概要を次に示します。

1. index.html（ユーザーがリンクをクリックして言語を選択）

   ```
   <a href="http://localhost:7070/xsql/fly/index.xsql?lang=french">Français</a>
   ```

2. index.xsql（XSQL ページがユーザーの選択肢をパラメータに格納）

   ```
   <xsql:set-stylesheet-param name="lang" value="{@lang}"/>
   ```

3. flyHTML.xsl （スタイルシートが言語選択パラメータを使用してメッセージ・ファイルからメッセージを選択）

```
<xsl:value-of select= "document('messages.xml')/messages/msg[@id=101 and
@lang=$lang]"/>
```

4. messages.xml（メッセージ・ファイルに翻訳済みメッセージを格納）

```
<msg id="101" lang="french">Prochains vols sur Oracle Airlines</msg>
```

次のリストに、コンテキスト内でのこのステップを示します。

index.html には、サポートされる各言語の URL で index.xsql を起動する HREF リンクが表示されます。

..

Web-to-Go の場合

```
<!-- Assumes default install to c:¥xsql and Flight Finder files in c:¥xsql¥fly -->
<ul>
 <li type="disc">
  <a href="http://localhost:7070/xsql/fly/index.xsql">English</a>
   </li>
   <li type="disc">
  <a
href="http://localhost:7070/xsql/fly/index.xsql?lang=french">Fran&ccedil;ais</a>
   </li>
   <li type="disc">
  <a
href="http://localhost:7070/xsql/fly/index.xsql?lang=spanish">Espa&ntilde;ol</a>
   </li>
   <li type="disc">
  <a href="http://localhost:7070/xsql/fly/index.xsql?lang=german">Deutsch</a>
   </li>
  </ul>
...
```

次に、ユーザーによる選択肢が URL から抽出され、index.xsql のパラメータにプラグインされます。URL で言語が指定されていない場合、次のコードの 1 行によりデフォルトで英語に設定されます。この XSQL ページでは、XSQL Servlet によりデータベースに送信される問合せも定義されています（この例では省略されています）。

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" media="Mozilla" href="indexHTML.xsl"?>
...
<index xmlns:xsql="urn:oracle-xsql" connection="fly" lang="english">
```

```
<xsql:set-stylesheet-param name="lang" value="{@lang}"/>
...
</index>
```

データベースが問合せ結果を戻すと、XSQL Servlet はそれに XSLT 変換を適用してフォーマットします。次のコードはスタイルシート flyHTML.xsl から抜粋したものです。このコードには、メッセージ・ファイル（messages.xml）を開いて指定された言語のメッセージ 101 を選択する 1 行が含まれています。

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version='1.0'>
    <xsl:output media-type="text/html" method="html"/>
      <xsl:param name="lang" select="@lang"/>
       <xsl:template match="/">
          <html>
...
          <body>
          ...
<!-- Next available flights -->
<xsl:value-of select=
 "document('messages.xml')/messages/msg[@id=101 and @lang=$lang]"/>
  ...
         </body>
       </html>
     </xsl:template>
     ...
</xsl:stylesheet>
```

次の XML コードは messages.xml から抜粋したものです。このファイルでは、メッセージは Flight Finder からクライアントに送信される情報（ラベルやプロンプトなど）を表します。メッセージは ID 番号で識別され、それぞれサポートされる各言語に翻訳されます。次のコードは、メッセージ 101 の 4 つの翻訳を示しています。翻訳には、メッセージのドイツ語バージョンのように各国語キャラクタ・セットのコードを含めることができる点に注意してください。この種の文字を表示するには、ブラウザの設定が必要になる場合があります。たとえば、Internet Explorer では、「表示」> 「エンコード」> 「その他」> 「西ヨーロッパ言語（Windows）」を選択します。

```
<?xml version="1.0"?>
<messages>
...
  <msg id="101" lang="english">Oracle Airlines available flights</msg>
  <msg id="101" lang="french">Prochains vols sur Oracle Airlines</msg>
  <msg id="101" lang="spanish">Proximos vuelos sobre Oracle Airlines</msg>
  <msg id="101" lang="german">M&#246;gliche Fl&#252;ge mit Oracle Airlines</msg>
...
</messages>
```

XML のデータベースへの書込み

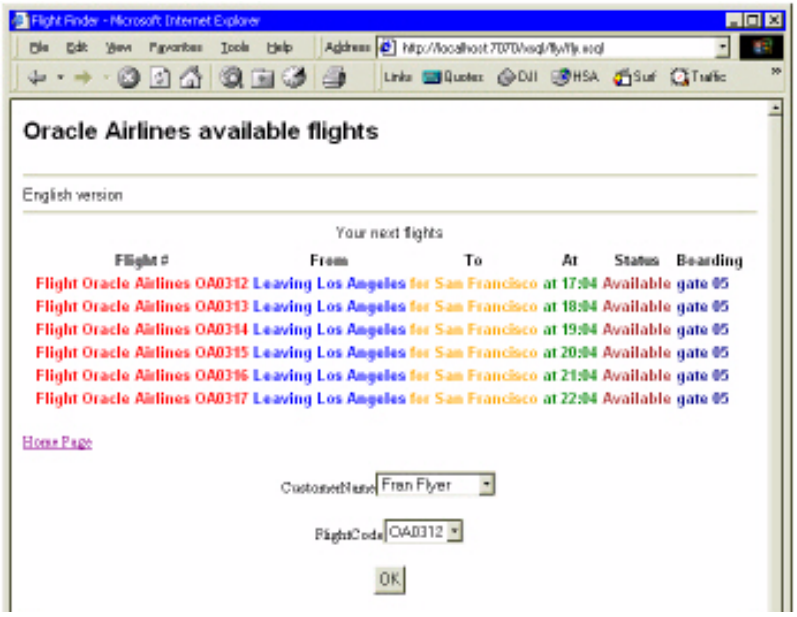
この項では、Flight Finder がユーザーからの入力を取得し、それを XML に変換してデータベースに書き込む方法について説明します。

1 ユーザー入力の取得

最初の手順は、ユーザー入力を取得することです。

図 4-4 に、HTML フォームを示します。このフォームには、Los Angeles から San Francisco へのフライトに関する問合せ結果と、顧客名およびフライト・コードのドロップダウン・リストが表示されます。ユーザーが氏名とコードを選択し、「OK」ボタンをクリックして、入力した顧客名で指定のフライトを予約すると、アプリケーションはその情報をデータベースに書き込みます。アプリケーションのこの部分は、HTML および英語に対してのみ実装されています。

図 4-4 Flight Finder: Los Angeles から San Francisco へのフライトに関する問合せ結果を表示する HTML フォーム



ドロップダウン・リスト CustomerName および FlightCode にデータベースからの値を移入する `fly.xsql` からのコードを次に示します。<form> タグには、ユーザーからフォームが送信されたときに値を処理するために実行するファイルとして、bookres.xsql を指定するアクション属性が含まれています。

ファイル flyHTML.xsl（このリストでは省略されています）には、前述の図に示したフォームをフォーマットするための XSLT 指示があります。

```
...
<form action="bookres.xsql" method="post">
   <field name="CustomerName">
      <xsql:query rowset-element="dropDownList"
          row-element="listElem">
          <![CDATA[
             select unique name as "listItem"
               from customers
               order by name
               ]]>
           </xsql:query>
         </field>
          <field name="FlightCode">
           <xsql:query rowset-element="dropDownList"
                       row-element="listElem">
            <![CDATA[
             select F.code as "listItem",
             F.code as "itemId",
             A1.name as "depart_airport",
             A2.name as "arrive_airport"
               from flights F,
               airports A1,
               airports A2
               where to_number(To_Char(F.schedule, 'HH24MI')) >
                     to_number(To_Char(sysdate, 'HH24MI')) and
                     F.code_from = '{@FROM}' and
                     F.code_to = '{@TO}' and
                     F.code_from = A1.code and
                     F.code_to = A2.code
               ]]>
            </xsql:query>
           </field>
           <sendRequest type="button" label="OK"/>
         </form>
       ...
```

2 ユーザーから取得した値のコード・パラメータへの割当て

ユーザーから値を取得した後の手順は、その値をコード内のパラメータに割り当てることです。次のコードは、bookres.xsql から抜粋したものです。

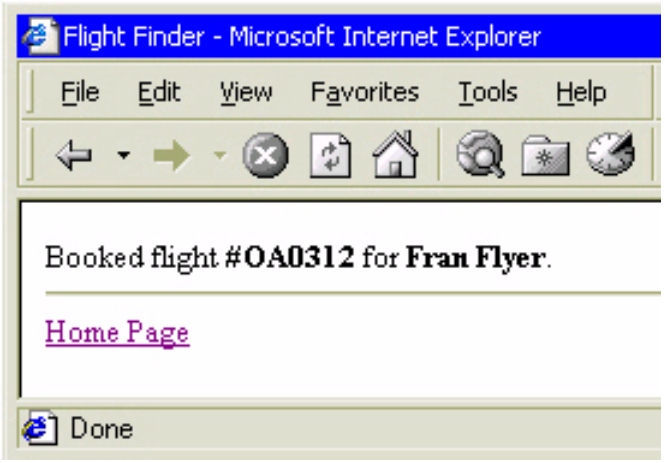
ユーザーの選択肢はパラメータ CustomerName および FlightCode に格納され、値を XSLT スタイルシートに渡せるように、パラメータ cust および code が定義されます。また、<xsql:dml> タグを使用して、CUSTOMERS 表に 1 行を挿入する SQL 文が定義されます。

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" media="Mozilla" href="bookresHTML.xsl"?>
<?xml-stylesheet type="text/xsl" media="MSIE 5.0" href="bookresHTML.xsl"?>
   <bookFlight xmlns:xsql="urn:oracle-xsql" connection="fly">
        <xsql:set-stylesheet-param name="cust" value="{@CustomerName}"/>
        <xsql:set-stylesheet-param name="code" value="{@FlightCode}"/>
        <xsql:dml>
        <![CDATA[
           insert into customers values
            ('{@CustomerName}', tripseq.NEXTVAL, '{@FlightCode}')
         ]]>
        </xsql:dml>
       ...
  </bookFlight>
```

3 ユーザーに対する操作の成功の通知

最後の手順は、操作が成功したかどうかをユーザーに通知することです。この場合は、次のように、フライトが予約されたかどうかを通知します。

図 4-5 Flight Finder: ユーザーに対するフライト予約成功の通知



次のコードは、bookresHTML.xsl から抜粋したものです。

パラメータ cust および code が宣言されており、bookres.xsql から渡された値が格納され、各パラメータを使用してユーザーにメッセージが表示されます。このようなパラメータの XSLT 構文は、$param です。

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output media-type="text/html"/>
     <xsl:param name="cust"/>
       <xsl:param name="code"/>
        <xsl:template match="/">
          <html>
            <head>
            <title>Flight Finder</title>
            </head>
            <body>
              Booked flight #<b><xsl:value-of select="$code"/></b>
              for <b><xsl:value-of select='$cust'/></b>.
              <hr/>
             <xsl:apply-templates select="bookFlight/returnHome"/>
            </body>
           </html>
        </xsl:template>
       ...
</xsl:stylesheet>
```

Oracle9*i* Application Server Wireless Edition（Portal-to-Go）の使用

独自の XSQL コードと XSL コードを記述するかわりに、Oracle9*i* Application Server Wireless Edition（Oracle Portal-to-Go）を使用できます。

Oracle Internet Platform のコンポーネントである Oracle9*i*AS Wireless Edition では、あらゆる対応デバイスに Web コンテンツを配信するために必要な機能がすべて提供されます。既存のコンテンツをデバイス固有の形式に変換し、エンド・ユーザー用のポータル・インタフェースを提供します。また、Oracle JDeveloper で開発できます。

Oracle9*i*AS Wireless Edition では、XML を使用してコンテンツの取得と配信が分離されます。

Oracle9*i*AS Wireless Edition ポータルには、次のコンポーネントが含まれています。

- データをモバイル・デバイスに配信するサービス
- HTML および RDBMS のコンテンツを XML に変換するアダプタ
- XML を、HTML、WML、TinyHTML および音声マークアップ言語（VoxML）など適切なマークアップ言語に変換するトランスフォーマ

ホワイト・ペーパー、製品マニュアルおよび無償でダウンロードできるソフトウェア・バージョンなどの詳細は、http://otn.oracle.co.jp/products/app_server/we/we.html の URL で OTN Japan の Oracle9*i*AS Wireless Edition ページにアクセスしてください。

関連項目： 第 3 章「Oracle9iAS Wireless Edition と XML」

5

XML を使用したコンテンツのカスタマイズ : Dynamic News アプリケーション

この章の内容は、次のとおりです。

Dynamic News アプリケーションの概要

Dynamic News アプリケーションでは、Oracle XML プラットフォーム・コンポーネントと Oracle9*i* データベースを併用して、Web ベースのニュース・サービスが構築されます。また、Java、XML、XSL、HTML および Oracle9*i* が併用されます。

- データベース内のニュース項目を使用すると、ユーザー入力に基づいて問合せを実行し、コンテンツをパーソナライズできます。
- XML、XSL および HTML を使用すると、複数プラットフォーム用に表示をカスタマイズできます。
- Dynamic News アプリケーションでは、パフォーマンスを改善できる場合に、XML 文書が事前に生成されます。

問題 : ユーザー要求に従って、ブラウザで受信するニュースをカスタマイズする必要があります。

解決策 : この問題を解決するには、Oracle XML コンポーネント、Oracle9*i* データベースおよびカスタム・サーブレットを使用します。この章では、この解決策について説明します。

使用する Oracle XML コンポーネント : XML Parser for Java、XML SQL Utility（XSU）for Java

---

Dynamic News の主要な作業

Dynamic News アプリケーションは、次の作業の実行方法を示します。

- データベースへの新規ヘッドラインの格納
- XML によるニュースの出力
- XSL スタイルシートの適用による新規ヘッドラインのフォーマット

Dynamic News アプリケーションの概要

Dynamic News は、データベースからニュース項目（ヘッドライン）を取り出して HTML ページを構築します。HTML ページは、ユーザー設定項目に従ってカスタマイズされます。

各ページには項目リストが表示され、各項目には詳細なトピックへのハイパーリンクが付いています。ニュース項目には、それぞれ次のような属性があります。

- Sports や Technology などのカテゴリ
- Baseball や Software などのサブカテゴリ
- Feature や Review などのタイプ

カスタマイズの 3 つのレベル : 静的、半動的および動的

Dynamic News では、これらの属性を使用して次の 3 つのカスタマイズ・レベルが提供されます。

- 静的
- 半動的
- 動的

表 5-1 に、これらの選択肢を示します。

表 5-1 Dynamic News: 3 つのカスタマイズ・レベル

カスタマイズ・レベル	説明
静的	静的ページはカスタマイズされません。 このレベルのエンド・ユーザーには、各カテゴリ、サブカテゴリおよびタイプのすべての項目を示すページが表示されます。 ニュース・システムの管理者は、管理サーブレットを使用して、静的な XML 文書を定期的（1 時間ごとなど）に生成します。 アプリケーションではこの種のページを必要に応じて構築できますが、要求したユーザーごとに問合せを実行して同一ページを構築するよりも、事前に生成されたページを提供する方が高速です。
半動的	半動的ページでは、事前に生成された項目リストが結合されます。 エンド・ユーザーが 1 つ以上のカテゴリを選択すると、Dynamic News により各カテゴリの項目を示すページが構築されます。ニュース管理者は、定期的に管理サーブレットを使用して、各カテゴリの項目のリストを事前に生成します。 半動的ページは、静的ページと同様に、パフォーマンス改善のために事前に生成されたドキュメントから構築されます。
動的	動的ページは、エンド・ユーザーからの要求時に構築されます。コンテンツはデータベースから直接取り込まれ、事前に生成されるものはありません。 最初に、エンド・ユーザーがサーブレットを起動し、カテゴリ、サブカテゴリおよびタイプを選択します。次に、Dynamic News が、その条件と一致する項目をデータベースに問い合せ、結果セットを使用して XML 文書を構築します。その後、静的ページや半動的ページの場合と同様に、XSLT 変換が適用されて HTML が生成されます。

注意 : 「動的」および「静的」という用語は、ページの動作ではなくコンテンツを指します。

Dynamic News の SQL の例 1: 項目スキーマ nisetup.sql

nisetup.sql から抜粋した、ニュース項目の構造を定義する SQL を次に示します。

```
CREATE TABLE news.NEWS_ITEMS
( ID   NUMBER NOT NULL,
  TITLE           VARCHAR2(200),
  URL             VARCHAR2(200),
  DESCRIPTION     VARCHAR2(2000),
  ENTRY_DATE      DATE,
  CATEGORY_ID     NUMBER,
  SUB_CATEGORY_ID NUMBER,
  TYPE_ID         NUMBER,
  SUBMITTED_BY_ID NUMBER,
  EXPIRATION_DATE DATE,  A
  APPROVED_FLAG   VARCHAR2(1)
);
```

Dynamic News サーブレット

表 5-2 に、Dynamic News アプリケーションで使用されるサーブレットを示します。これらのサーブレットは、アプリケーション・ロジックへのエントリ・ポイントを提供します。

表 5-2 Dynamic News サーブレット

サーブレット	説明	ファイル名
管理	■ データベースにニュース項目を追加します。 ■ ユーザー、タイプおよびカテゴリのリストをメンテナンスします。 ■ 静的（カスタマイズされない）ニュース・ページの XML および HTML を生成します。	xmlnews/admin/AdminServlet.java
半動的	エンド・ユーザーが選択したカテゴリのニュース項目のリストを生成します。	xmlnews/dynamic/SemiDynamicServlet.java
動的	データベースからニュース項目を取り出し、エンド・ユーザーの設定項目に基づいてカスタム・ページを生成します。	xmlnews/dynamic/DynamicServlet.java

Dynamic News の動作 : 概要

XML 文書の生成による HTML ページの構築

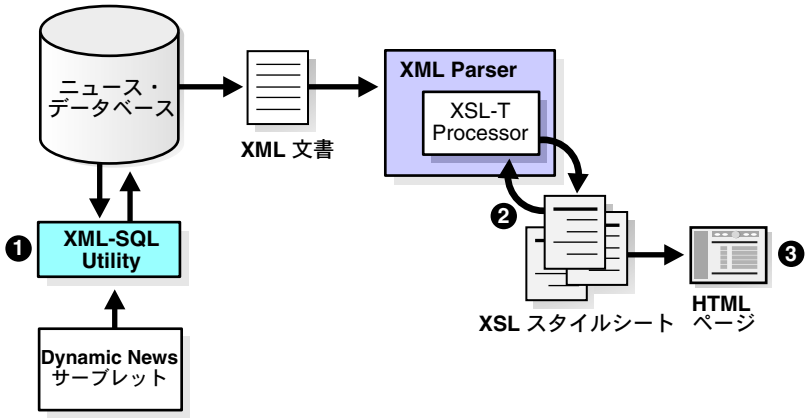
Dynamic News は、XML 文書を生成して HTML ページを構築します。

- 静的ページ。ニュース・システム管理者が設定した間隔で、事前に生成された XML 文書から構築されます。
- 半動的ページ。ユーザーが選択したカテゴリの項目を表示する、事前に生成された XML 文書から構築されます。
- 動的ページ。ユーザーが選択したカテゴリ、サブカテゴリおよびタイプ別の項目リストを表示する XML 文書から、必要に応じて構築されます。

図 5-1 に、これらのステップを Dynamic News が実行する方法の概要を示します。

1. Oracle XML SQL Utility （XSU）をコールします。これにより、データベースにニュース項目を問い合せ、結果を XML 文書に書き込みます。実行モードは次のとおりです。
   - 静的ページの場合はバッチ・モード
   - 半動的ページの場合はバッチ・モード
   - 動的ページの場合は要求時
2. Oracle XML Parser for Java の XSL-T Processor を使用し、3 つの XSL スタイルシートのうち 1 つを使用して XML を HTML に変換します。Netscape Navigator 用に 1 つ、Internet Explorer 用に 1 つ、他のすべてのブラウザ用の汎用スタイルシートが 1 つあります。
3. HTML ページを Web サーバー経由でユーザーに配信します。

図 5-1 Dynamic News



静的ページ

Dynamic News は、使用可能なニュース項目をすべて表示する静的ページを生成します。この種のページは、1 時間ごとなど、ニュース・システム管理者が設定した間隔で構築されます。それ以外の場合は変更されません。

静的ページを使用する場合

静的ページは、データがほとんど変化しないアプリケーションすべてで役立ちます。たとえば、ERP や顧客アプリケーションからの日次サマリーを公開する場合などです。このコンテンツは静的であるため、要求したユーザーごとに個別に構築するよりも、1 ページを事前に生成する方が効率的です。

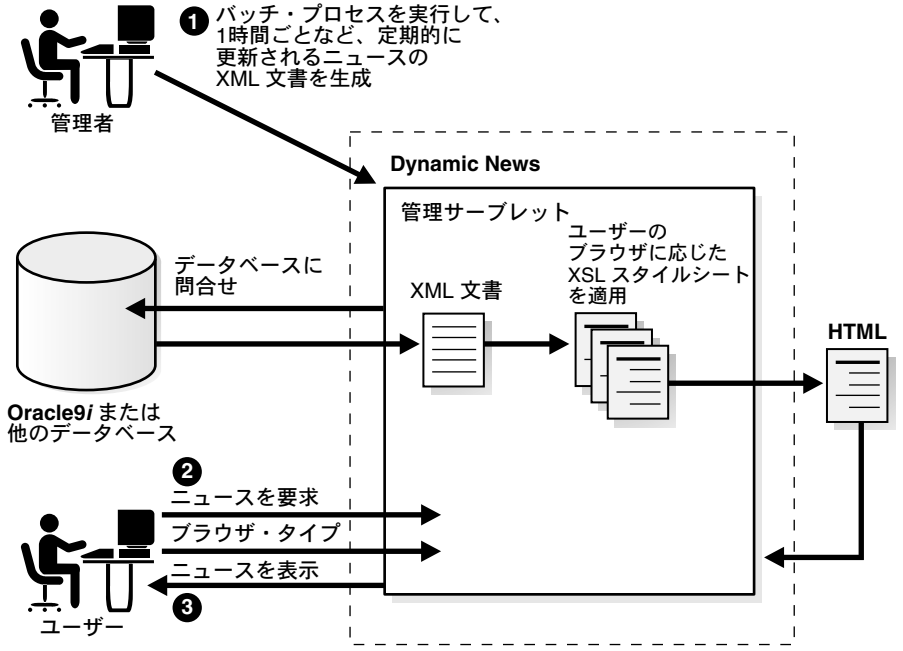
静的ページの動作

管理者が管理サーブレットから実装されたバッチ・プロセスを実行すると、データベースが問合せされ、XML 文書が生成されます。エンド・ユーザーが Dynamic News を起動してすべてのニュースを表示すると、サーブレットは HTTP 要求のユーザー・エージェント・ヘッダーからブラウザ・タイプを取得し、XML 文書を読み込んで、適切な XSL スタイルシートを適用します。

最後に、図 5-2 のように、エンド・ユーザーのブラウザ用にフォーマットされた HTML ページが戻されます。

もう 1 つのアプローチは、バッチ・プロセスの一部として XSL スタイルシートを適用し、スタイルシートごとに 1 つずつ HTML ページを生成する方法です。この場合、管理するファイル数は増えますが、ランタイム・サーブレットは小型になります。

図 5-2 Dynamic News: 静的ページ ー XML 文書の生成



半動的ページ

アプリケーションは、事前に生成されたリストを組み合せて半動的ページを構築します。カテゴリ別の項目リストは管理者によって事前に生成されますが（カテゴリごとに 1 つの XML ファイル）、それを含むページはユーザーごとにカスタマイズされます。エンド・ユーザーは、Sports、Business および Entertainment などのカテゴリを選択します。

半動的ページを使用する場合

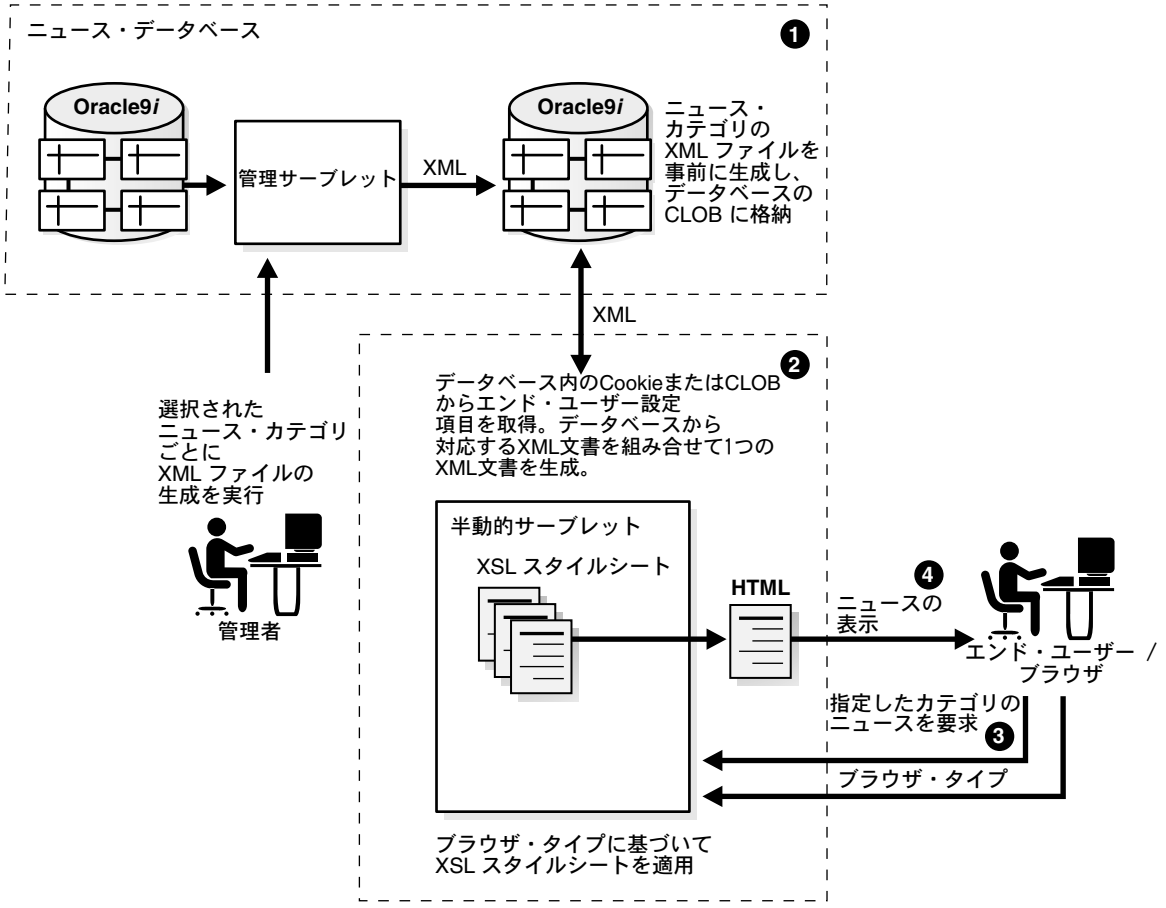
半動的アプローチが役立つのは、データがほとんど変更されず、エンド・ユーザーに提供する必要のある選択肢が比較的少数の場合です。提供する選択肢が多いアプリケーションほど多数のドキュメントを事前に生成する必要があり、それに比例してメリットも乏しくなります。

半動的ページの動作

図 5-3 に、半動的な生成の動作を示します。この動作は次の 2 つのフェーズに分かれています。

- **フェーズ 1 ― 静的処理フェーズ**: 管理者が定期的に管理サーブレットを使用して XML ファイルを事前に生成し、それをデータベース内の CLOB に格納します。XML ファイルは、単純なフラット・ファイル・システムに格納することもできますが、データベースにとっては潜在的なパフォーマンス改善というメリットが得られなくなります。
- **フェーズ 2 ― 動的処理フェーズ**: このフェーズは、エンド・ユーザーが、指定したカテゴリのニュース項目を要求した時点で開始されます。サーブレットは、データベースから CLOB を取り出し、結合して 1 つの XML 文書にします。ユーザー設定項目はデータベースとクライアント側 Cookie の両方に格納され、パフォーマンスを改善できる場合に Cookie から読み込まれます。次に、エンド・ユーザーのブラウザと一致する XSL スタイルシートを使用して、XML 文書が HTML ページに変換されます。静的ページの場合と同様に、サーブレットは HTTP 要求のユーザー・エージェント・ヘッダーからブラウザ・タイプを取得します。

図 5-3 Dynamic News: 半動的ページ ー XML 文書の生成



動的ページ

アプリケーションは、項目をデータベースから直接取り出して、必要に応じて動的ページを構築します。エンド・ユーザーは「Create/Edit User Preference Page」にアクセスし、カテゴリ、サブカテゴリおよびタイプ（「Entertainment」－「Movies」－「Review」など）を選択します。

動的ページを使用する場合

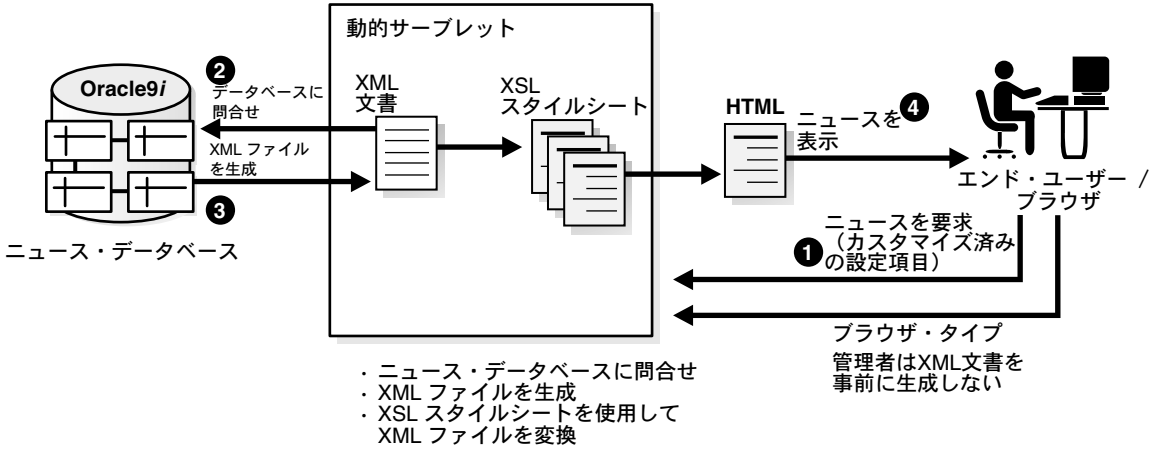
動的ページが役立つのは、Breaking News など、最新情報を配信する場合です。また、指定した株式に関する過去 10 年間のいずれかの日付の終値など、履歴データを配信する場合にも役立ちます。考えられる要求ごとにドキュメントを事前に生成するのは不可能ですが、データベースからデータを取り出すのは簡単で効率的です。

動的ページの動作

図 5-4 に、動的な生成の動作を示します。他のランタイム・モデルとは異なり、管理者は XML 文書を事前に生成しません。かわりに、動的サーブレットはエンド・ユーザーが選択したカスタマイズに基づいて、ニュース項目をデータベースに問い合せます。

ユーザー設定項目はデータベースとクライアント側 Cookie の両方に格納され、パフォーマンスを改善できる場合に Cookie から読み込まれます。問合せの結果を使用して XML ファイルが生成され、XSL スタイルシートを使用してユーザーのブラウザに適した HTML ページに変換されます。他のアプローチの場合と同様に、アプリケーションは HTTP 要求のユーザー・エージェント・ヘッダーからブラウザ・タイプを取得します。

図 5-4 Dynamic News: 動的ページ － XML 文書の生成



コンテンツのパーソナライズ

Oracle9*i* により、Dynamic News に柔軟性が加わります。ニュース項目はデータベースに格納されるため、Dynamic News により必要に応じてコンテンツをカスタマイズできます。この項のコード例は、エンド・ユーザーが指定したカテゴリのニュース項目をアプリケーションが取り出し、ページをパーソナライズする方法を示しています。主要な作業は次のとおりです。

1. エンド・ユーザー設定項目を取得します。
2. データベースからニュース項目を取り出します。
3. ニュース項目を結合してドキュメントを構築します。
4. パーソナライズされたコンテンツに編集した後に、アプリケーションは後述のようにエンド・ユーザーのブラウザに合せてページをフォーマットし、表示をカスタマイズします。

1 エンド・ユーザー設定項目の取得

設定項目を処理するロジックはアプリケーション全体に分散されており、データはデータベースとクライアント側 Cookie の両方に格納されます。アプリケーションは、パフォーマンスを改善できる場合に Cookie から設定項目データを読み込みます。Cookie からデータを取得できない場合（エンド・ユーザーがサイトに初めてアクセスした場合や、そのブラウザが Cookie を受け入れない場合など）、アプリケーションは設定項目データをデータベースから読み込みます。

クライアント側 Cookie から

次の 2 つの方法は、アプリケーションが Cookie に格納された設定項目データを処理する方法を示しています。どちらの方法も xmlnews.common.UserPreference からの抜粋で、サンプル Cookie は次のとおりです。

```
DynamicServlet=3$0$0#4$2$1***242
```

この Cookie は、ドル記号を使用して設定項目の値を区切り、シャープ記号を使用してカテゴリを区切り、3 個のアスタリスクを個々のユーザー ID および設定項目データへのトークンとして使用しています。前述のサンプル Cookie は、ユーザー 242 がカテゴリ 3 および 4 の項目を要求していることを示します。カテゴリ 3 で、ユーザーは全サブカテゴリの全タイプの項目を要求しています（値 0 の場合は、すべての項目が選択されます）。カテゴリ 4 では、ユーザーはサブカテゴリ 2 の項目のうち、タイプ 1 の項目のみを要求しています。

サンプル・アプリケーションでは、このような Cookie は次の 2 段階で処理されます。

1. 最初に、`getNewsCookie` が HTTP 要求を発行したブラウザから「DynamicServlet」Cookie を取得します。
2. 次に、それを `loadPreferenceFromCookie` が解析し、ユーザーの ID と設定項目を含む文字列を取得します。

```
public Cookie getNewsCookie(HttpServletRequest request)
              throws Exception {
      Cookie c[] = request.getCookies();
      Cookie l_returnCookie = null;
              for (int i = 0; (c!= null) && (i < c.length); i++) {
              if (c[i].getName().equals("DynamicServlet")) {
                  l_returnCookie = c[i];
                  }
                }
                return l_returnCookie;
              }
public Vector loadPreferenceFromCookie(Cookie p_cookie) throws Exception {
      Vector l_prefId = new Vector(2);
      String l_Preferences = p_cookie.getValue();
      StringTokenizer l_stToken = new StringTokenizer(l_Preferences, "***");
      String l_userId = "";
      while (l_stToken.hasMoreTokens()) {
         // First Token is User Preference.
         l_Preferences = l_stToken.nextToken();
         // Second Token is User ID.
         l_userId = l_stToken.nextToken();
      }
      l_prefId.addElement(l_Preferences);
      l_prefId.addElement(l_userId);
      return l_prefId;
   }
```

データベースの問合せ

Cookie から設定項目を読み込めない場合、アプリケーションはデータベースに問合せします。クラス xmlnews.common.GenUtility では、データベースに接続してニュースのカテゴリ、サブカテゴリおよびタイプをフェッチするメソッドが実装されます。

半動的サーブレットと動的サーブレットは、これらのメソッドとメソッド loadInitalPreference および constructUserPreference をコールします。どちらも、xmlnews/common/UserPreference.java に実装されます。

メソッド loadInitalPreference は getSubCategories をコールし、結果セット全体をループし、カテゴリ値を区切り文字と結合して設定項目文字列を構築します。

```
public String loadInitialPreference(Vector p_category, Vector p_subcategory,
       Vector p_types, Connection p_con)
throws Exception {
GenUtility m_general = new GenUtility();
...
for (int i = 0; i < p_category.size(); i++) {
       String l_cat[] = (String []) p_category.elementAt(i);
       l_category = l_cat[0];
```

```
        Vector l_subcategory = m_general.getSubCategories(p_con,l_cat[0]);

        for(int l_j = 0, l_k = 0; l_j < l_subcategory.size(); l_j++, l_k++)
 {
...
// Append the next preferences to the constructed string
   l_userPref = l_userPref+"#"+l_category+"$"+l_subCat+"$"+l_typeStr;
          }
   }
...
    return l_userPref;
   }

public static Vector getSubCategories(Connection p_conn, String p_categoryId)
    throws Exception {
    Vector l_subCats = new Vector();

PreparedStatement l_pstmt = p_conn.prepareStatement(
   "Select id, name from sub_categories where category_id = ? ");
    l_pstmt.setString(1, p_categoryId);
    ResultSet l_rset = l_pstmt.executeQuery();

while (l_rset.next()) {
     String[] l_subCat = new String[2];
     l_subCat[0] = new String(l_rset.getString(1));
     l_subCat[1] = new String(l_rset.getString(2));
     l_subCats.addElement(l_subCat);
     }
l_pstmt.close();
return l_subCats;
}
```

たとえば、次のコードは xmlnews.dynamic.DynamicServlet.service から抜粋したものです。

このコードは、前述のメソッドをコールしてデータベースからエンド・ユーザー設定項目を読み込み、それを使用して HTML ページを構築します。

```
public void service(HttpServletRequest p_request,
      HttpServletResponse p_  response)
      throws ServletException {

   // The following are declared elsewhere as class variables
   // and initialized in the servlet's init method.
   // GenUtility m_general = null;

   // m_general = new GenUtility();
   // UserPreference m_userPreference = null;
```

```
   // m_userPreference = new UserPreference();
   ...

   // If the database connection has been closed, reopen it.
      if (m_connection == null || m_connection.isClosed())
          m_connection = m_general.dbConnection();
   ...
      String l_preference = m_userPreference.loadInitialPreference(
      m_general.getCategories(m_connection),
      null, m_general.getTypes(m_connection),
                               m_connection);

m_userPreference = m_userPreference.constructUserPreference
   ( l_preference,m_status);

   // Display the Dynamic Page
      this.sendDynamicPage(l_browserType, p_response,
      l_userName, m_userPreference,
      m_servletPath + "?REQUEST_TYPE=SET_ADVANCED_USER_PREFS",
      m_servletPath + "?REQUEST_TYPE=LOGIN_REQUEST",
      m_servletPath + "?REQUEST_TYPE=LOG_OUT_REQUEST",
      m_servletPath);
    ...
 }
```

2 データベースからのニュース項目の取出し

次のコードは xmlnews.admin.AdminServlet.performGeneration および xmlnews.admin.AdminServlet.staticProcessingHtml から抜粋したもので、アプリケーションが使用可能な各カテゴリのニュース項目をデータベースに問い合せ、それぞれの結果セットを XML 文書に変換する方法を示しています。

データベースには各カテゴリの XML がキャラクタ・ラージ・オブジェクト（Character Large OBject: CLOB）として格納されるため、アプリケーションは極端に長いリストでも処理できます。

```
public void performGeneration(String p_user, String p_genType,
      HttpServletResponse p_response)
      throws ServletException, IOException {
...
      try {
      String l_fileSep = System.getProperty("file.separator");
      String l_message = ""; // Holds status message

      if (p_genType.equals("BATCH_GEN")) { // Batch Generation
      String l_htmlFile = "BatchGeneration";
      String l_xslFile  = "BatchGeneration";
```

```
      String l_xmlFile  = "BatchGeneration";

      // Generate the XML and HTML content and save it in a file
         this.staticProcessingHtml(
         m_dynNewsEnv.m_dynNewsHome+l_fileSep+l_htmlFile+".html",
         m_dynNewsEnv.m_dynNewsHome+l_fileSep+m_dynNewsEnv.m_batchGenXSL,
         m_dynNewsEnv.m_dynNewsHome+l_fileSep+l_xmlFile+".xml"
      );
      ...
    }
 ...
 }
```

メソッド xmlnews.admin.AdminServlet.staticProcessingHtml は、ニュース項目をフェッチする問合せを定義して実行します。次に、Oracle XML SQL Utility（XSU）を使用して結果セットから XML 文書を構築し、XSLT 変換を適用して HTML ページを作成します。

```
public void staticProcessingHtml(String p_htmlFile,String p_xslfile,
   String p_xmlfile) throws Exception {
   String l_query = "select a.id, a.title, a.URL, a.DESCRIPTION, " +
   " to_char(a.ENTRY_DATE, 'DD-MON-YYYY'), a.CATEGORY_ID, b.name,
                  a.SUB_CATEGORY_ID, c.name, a.Type_Id, d.name, " +
" a.Submitted_By_Id, e.name, to_char(a.expiration_date, 'DD-MON-YYYY'),
                                                        a.approved_flag " +
" from news_items a, categories b, sub_categories c, types d, users e where " +
" a.category_id is not null and a.sub_category_id is not null and "+
" a.type_id is not null and a.EXPIRATION_DATE is not null and "+
" a.category_id = b.id  AND a.SUB_CATEGORY_ID = c.id AND a.Type_ID = d.id
                         AND " +
" a.SUBMITTED_BY_ID = e.id AND "+
" a.EXPIRATION_DATE > SYSDATE AND "+
" a.APPROVED_FLAG = ¥'A¥' ORDER BY b.name, c.name ";

 Statement l_stmt = m_connection.createStatement();
 ResultSet l_result = l_stmt.executeQuery(l_query);
// Construct the XML Document using Oracle XML SQL Utility
   XMLDocument l_xmlDocument = m_xmlHandler.constructXMLDoc(l_result);
   l_stmt.close();

// Get the HTML String by applying corresponding XSL to XML.
   String l_htmlString = m_xmlHandler.applyXSLtoXML(l_xmlDocument,p_xslfile);

 File l_file = new File(p_htmlFile);
 FileOutputStream l_fileout = new FileOutputStream(l_file);
 FileOutputStream l_xmlfileout = new FileOutputStream(new File(p_xmlfile));.
 l_fileout.write(l_htmlString.getBytes());
```

```
 l_xmlDocument.print(l_xmlfileout);

 l_fileout.close();
 l_xmlfileout.close();
}
```

3 ニュース項目の結合によるドキュメントの構築

コンテンツのパーソナライズの最終ステップは、エンド・ユーザー設定項目に従って XML 文書を HTML ページに変換することです。

次のコードは xmlnews.generation.SemiDynamicGenerate.dynamicProcessing から抜粋したものです。

このコードはユーザーが選択したカテゴリに対応する CLOB を取り出し、各 CLOB を XML 文書に変換してから、結合して 1 つの XML 文書にします。XML 文書を HTML ページに変換する処理については、次の項を参照してください。

```
public XMLDocument semiDynamicProcessingXML(Connection p_conn, UserPreference p_
prefs)
        throws Exception
 {
     String l_htmlString = null ;
     XMLDocument l_combinedXMLDocument = null ;
     XMLDocument[] l_XMLArray = new XMLDocument[p_prefs.m_categories.size()];
     int l_arrayIndex = 0 ;

     PreparedStatement l_selectStmt = p_conn.prepareStatement(
                  " SELECT PREGEN_XML FROM CATEGORIES_CLOB WHERE CATEGORY_ID = ?");
     // Process each preference.
     for ( ; l_arrayIndex < p_prefs.m_categories.size(); ++l_arrayIndex ){
       l_selectStmt.setString(1, p_prefs.m_categories.elementAt(l_
arrayIndex).toString());
       OracleResultSet l_selectRst = (OracleResultSet)l_selectStmt.executeQuery();
       if (l_selectRst.next()) {
          CLOB l_clob = l_selectRst.getCLOB(1);
          l_XMLArray[l_arrayIndex] = convertFileToXML(l_clob.getAsciiStream());
       } else
          l_XMLArray[l_arrayIndex] = null ;
     }
     l_selectStmt.close();

     XMLDocHandler l_xmlHandler = new XMLDocHandler();
     l_combinedXMLDocument = l_xmlHandler.combineXMLDocunemts(l_XMLArray );
     return l_combinedXMLDocument ;
   }
```

4 表示のカスタマイズ

データベースからニュース項目をフェッチした後に、Dynamic News はそれを XML 文書に変換します。XML ではコンテンツが表示から分離されており、カスタム HTML ページを簡単に構築できます。

Dynamic News は、様々な XSL スタイルシートを使用して、XML 文書を各種ブラウザ用にカスタマイズされた HTML ページに変換します。

- Netscape Navigator 用に 1 つ
- Microsoft Internet Explorer 用に 1 つ
- 他のブラウザ用の汎用スタイルシート 1 つ

このプロセスは、次の 4 つのステップに分かれています。

1. ユーザーのブラウザ・タイプの取得
2. ニュース項目の取得
3. XML 文書の構築
4. XML から HTML への変換

アプリケーションは、HTTP 要求を受信するたびにユーザー・エージェント・ヘッダーを検査し、要求を出したブラウザの種類を調べます。次のコード行は xmlnews.dynamic.DynamicServlet.service から抜粋したもので、サーブレットが RequestHandler オブジェクト（xmlnews/common/RequestHandler.java に実装されています）を作成し、要求を解析してブラウザ・タイプを取得する方法を示しています。次に、サーブレットはこの情報を使用し、エンド・ユーザー設定項目とブラウザ・タイプに基づいて HTML ページを戻します。

```
public void service(HttpServletRequest p_request, HttpServletResponse p_response)
throws ServletException {
        ...
       // Instantiate a Request Handler (declared elsewhere)
        m_reqHandler = new RequestHandler(m_userPreference, m_general,m_status);
        RequestParams l_reqParams = m_reqHandler.parseRequest(p_request, m_
connection);
        String l_browserType = l_reqParams.m_browserType;
        ...
        // Display the Dynamic Page
        this.sendDynamicPage(l_browserType,p_response,l_userName,m_userPreference,
                             m_servletPath+"?REQUEST_TYPE=SET_ADVANCED_USER_PREFS",
              m_servletPath+"?REQUEST_TYPE=LOGIN_REQUEST",
              m_servletPath+"?REQUEST_TYPE=LOG_OUT_REQUEST",
              m_servletPath);
        ...
    }
```

xmlnews.common.GenUtility.getBrowserType にあるユーザー・エージェント・ヘッダーから実際にブラウザ・タイプを抽出するコードは、次のとおりです。

```
        public String getBrowserType(HttpServletRequest p_request) throws Exception
{

          // Get all the Header Names associated with the Request
          Enumeration l_enum = p_request.getHeaderNames();

          String l_Version     = null;
          String l_browValue   = null;
          String l_browserType = null;

          while (l_enum.hasMoreElements()) {
             String l_name = (String)l_enum.nextElement();
             if (l_name.equalsIgnoreCase("user-agent"))
                 l_browValue = p_request.getHeader(l_name);
          }

          // If the value contains a String "MSIE" then it is Internet Explorer
          if (l_browValue.indexOf("MSIE") > 0 ) {
              StringTokenizer l_st = new StringTokenizer(l_browValue, ";");
              // Parse the Header to get the browser version.
              l_browserType = "IE";
              while (l_st.hasMoreTokens()) {
                 String l_tempStr = l_st.nextToken();
                 if (l_tempStr.indexOf("MSIE") > 0 ) {
                    StringTokenizer l_st1 = new StringTokenizer(l_tempStr, " ");
                    l_st1.nextToken();
                    l_Version = l_st1.nextToken();
                 }
              }
          // If the value contains a String "en" then it is Netscape
          } else if (l_browValue.indexOf("en") > 0) {
              l_browserType = "NET";
              String l_tVersion = l_browValue.substring(8);
              int l_tempInd  = l_tVersion.indexOf("[");
              l_Version = l_tVersion.substring(0, l_tempInd);
          }

          // Return the Browser Type and Version after concatenating
          return l_browserType + l_Version;
        }
```

エンド・ユーザーのブラウザ・タイプを取得した後に、DynamicServlet のサービス・メソッドはそれを xmlnews.dynamic.DynamicServlet.sendDynamicPage に渡します。

このメソッドは、データベースから XML 文書をフェッチし、エンド・ユーザーのブラウザ・タイプに適した XSL スタイルシートを適用して HTML に変換し、HTML を生成します。

```
public void sendDynamicPage(String p_browserType,HttpServletResponse p_response,
  String p_userName,UserPreference p_pref,String p_userPrefURL,
  String p_signOnURL,String p_logout,
  String p_servletPath) throws Exception {
  String l_finalHTML = ""; // Holds the html
     if (p_browserType.startsWith("IE4") || (p_browserType.startsWith("IE5"))) {
         // Send the XML and XSL as parameters to get the HTML string.
         l_finalHTML = m_handler.applyXSLtoXML(
                              this.dynamicProcessingXML(m_connection, p_pref),
                              m_dyEnv.m_dynNewsHome + "/DynamicIE.xsl"
                              );
  String l_thisbit = m_general.postProcessing(l_finalHTML,p_userName,
         p_userPrefURL,p_signOnURL,p_logout,p_servletPath);
         PrintWriter l_output = p_response.getWriter();
         l_output.print(l_thisbit);
         l_output.close();
     }
      else if (p_browserType.startsWith("NET4") ||
         (p_browserType.startsWith("NET5"))) {
         // Do the same thing, but apply the stylesheet "/DynamicNS.xsl"
         ...
      // When the Browser is other than IE or Netscape.
      } else {
      // Do the same thing, but apply the stylesheet "/Dynamic.xsl"
      ...
      }
   }
```

主なメソッドは、次のとおりです。

- `xmlnews.dynamic.DynamicServlet.dynamicProcessingXML`

  このメソッドは、エンド・ユーザーの設定項目と一致するニュース項目をデータベースに問い合せます。`xmlnews.common.XMLDocHandler.constructXMLDoc` をコールして、問合せの結果セットを XML 文書に変換します。

- `xmlnews.common.XMLDocHandler.applyXSLtoXML`

  このメソッドは、指定された XSL スタイルシートを使用して、XML 文書を HTML に変換します。この変換には、Oracle XML Parser バージョン 2.0 の XSL 変換機能が使用されます。特に、ドキュメント・オブジェクト・モデル（Document Object Model: DOM）・パーサーを使用して、XML 文書の構造を表すツリーを作成します。最終的な HTML 文字列を構築するために、ツリーのルートとして機能する要素を作成してから、解析済みの DOM ドキュメントを追加します。

ニュース項目のインポートとエクスポート

Dynamic News では、Resource description framework Site Summary（RSS）標準に準拠する XML 文書をインポートおよびエクスポートすることもできます。RSS はデータ・チャネルを共有する手段として Netscape 社が開発した標準で、my.netscape.com や slashdot.org などの Web サイトで使用されています。

アプリケーションは、RSS を使用してニュース・ページをシンジケートし（RSS ホストで使用可能にし）、他の RSS サイトからのニュースを集計できます。たとえば、Dynamic News には `xmlnews.admin.RSSHandler` クラスが組み込まれています。このクラスでは、指定された DTD を使用して指定のファイルが解析され、ニュース項目が抽出されてから、項目がハッシュ表に格納されます。また、このクラスには、そのハッシュ表の要素を戻すメソッドも用意されています。

6

Oracle Internet File System を使用した XML アプリケーションの作成

この章の内容は、次のとおりです。

Oracle Internet File System（Oracle 9*i*FS）の概要

Oracle Internet File System（Oracle 9*i*FS）により、SMB、HTTP、FTP、SMTP および IMAP4 など、標準的な Windows およびインターネット・プロトコル経由でファイル・ベースおよびフォルダ・ベースの隠喩を使用して、ドキュメントとデータを簡単に編成し、アクセスできます。

9*i*FS により、Web ベース・アプリケーションの構築と管理が簡単になります。これは Java 用のアプリケーション・インタフェースであり、PowerPoint.PPT ファイルなどのドキュメントを Oracle9*i* にロードし、Oracle9*i* Application Server などの Web サーバーからドキュメントを表示できます。

Oracle 9*i*FS での XML の処理

9*i*FS を使用すると、開発者による XML の処理が簡素化されます。この場合、Oracle 9*i*FS は XML のリポジトリとして機能します。Oracle 9*i*FS では、XML 文書に対して次の作業を実行できます。

- XML を自動的に解析し、コンテンツを表および列に格納します。
- XML ファイルのコンテンツを表示します。

ファイルが要求された場合に、Web などで選択情報を配信します。

また、Oracle 9*i*FS では、新規のファイル・タイプを定義する拡張可能な方法がサポートされ、XML 文書であるファイル・タイプの定義、解析および表示用の組込みサポートが提供されています。

Oracle 9*i*FS では、XML データを格納するのみでなく、ビジネス・インテリジェンスの実装、動的コンテンツの生成およびアプリケーション間でのデータ共有のために、XML のあらゆる潜在的機能を活用できるようにします。

ドキュメント記述子の指定

Oracle 9*i*FS では、XML ベースのファイル・タイプを登録するときに、ドキュメント記述子で次の情報を指定します。

- ファイル・タイプの XML 文書構造
- データベースへの格納方法

たとえば、ドキュメントを完全な書式でデータベース内のラージ・オブジェクト（Large Object: LOB）に保存できます。

9*i*FS のドキュメント記述子では、XML ベースの構文を使用して、XML ベースのファイル・タイプの構造（またはスキーマ）が記述されます。

ファイルを Oracle 9*i*FS に保存または送信すると、ドキュメントはファイル・タイプの 1 つとして認識され、その XML が解析されて、データはドキュメント記述子で指定した表に格納されます。

これと同じ情報は、Oracle 9*i*FS でサポートされるいずれかのプロトコル経由でファイル・タイプの特定のインスタンスが要求された場合に、XML 文書の表示、つまり配信のための再編集に使用されます。

関連項目：

- http://otn.oracle.com/products/ifs/
- 『Oracle Internet File System セットアップおよび管理ガイド』

Oracle 9*i*FS パーサーの使用

Oracle 9*i*FS で認識される構造を持った新規 XML ファイルをフォルダに入れると、Oracle XML Parser for Java は XML ファイルを分析し、個々の属性を Oracle 9*i*FS スキーマに格納できます。Oracle 9*i*FS のサブクラスを定義することで、どの属性をどの列に格納するかを指定します。また、Oracle 9*i*FS に属性と列のデフォルト・マッピングを作成させることもできます。

最初はファイル内にあった属性は、解析後はファイルの属性となります。この余分なメタデータを編集し、ファイル・システムで検索条件として使用できます。

XML ベースの会社の標準的な保険料請求フォームを考えてみます。Oracle 9*i*FS に対して、各ファイルから属性タグ情報を抽出し、これを表に個別に格納し、XML 保険料請求ファイルを解析するように指示できます。

これにより、ファイル内の他の属性と同様に、地域や代理店などの XML 属性を検索できます。また、データは、保険業界の分析ツールなど、リレーショナル・アプリケーションでも使用可能になります。

XML ファイルは解析済みのため、これがそのファイルの終わりであることを意味しているわけではありません。オリジナルの XML ファイルを開くと、XML レンダラにより再構成されます。6-7 ページの「Oracle 9iFS での XML の表示」を参照してください。

Oracle 9*i*FS の標準パーサーとカスタム・パーサー

Oracle 9*i*FS では、XML アプリケーションにより生成されるドキュメント形式に必要な場合は、カスタム・パーサーが必要です。

Oracle 9*i*FS パーサーを明示的に起動する必要がある場合

Oracle 9*i*FS パーサーの起動方法は、アプリケーションにより生成されたドキュメントが Oracle 9*i*FS に入力される方法によって異なります。

- XML 文書がプロトコル・サーバーを使用してアップロードされる場合は、Oracle 9*i*FS の XML Parser がプロトコル・サーバーにより自動的に起動されます。
- XML 文書がアプリケーションによりアップロードされる場合は、Oracle 9*i*FS の XML Parser またはカスタム・パーサーを明示的に起動する必要があります。それ以外の場合、XML 文書はオブジェクトへと解析されるかわりに RAW データとして読み込まれます。
- XML 以外の形式でドキュメントを生成するカスタム・クラスがアプリケーションで定義される場合は、Oracle 9*i*FS の Java API の一部として提供されるクラスとメソッドを使用して、カスタム・パーサーを作成してください。

  カスタム・パーサーにより、カスタム・クラスの Oracle 9*i*FS リポジトリ・オブジェクトが作成されます。

  たとえば、Document クラスのサブクラスとなる Memo クラスを定義しているとします。Memo クラスには、カスタム属性 To、From、Date および Text（メモの内容）が含まれています。

  Memo オブジェクトを Oracle 9*i*FS に格納するには、パーサーが必要です。

  - Memo ドキュメントが XML 形式の場合、Oracle 9*i*FS の `SimpleXmlParser()` を使用して属性を抽出できます。
  - Memo ドキュメントが特別な形式で格納されている場合は、カスタム・パーサーを作成し、属性の抽出方法を指定します。

Oracle 9*i*FS の標準パーサーの使用

Oracle 9*i*FS では、アプリケーションを作成できるように複数の標準パーサーが提供されています。表 6-1 に、Oracle 9*i*FS の標準パーサー・クラスを示します。

表 6-1 Oracle 9*i*FS の標準パーサー・クラス

クラス	説明
SimpleXmlParser	Oracle 9*i*FS リポジトリ内で XML 文書本体からオブジェクトを作成します。このクラスは、Oracle 9*i*FS に格納されている全 XML 文書のデフォルト・パーサーとして使用されます。SimpleXmlParser により XmlParser が拡張されます。
XmlParser	カスタム XML Parser 開発用のベース・クラスです。
SimpleTextParser	カスタム・パーサーの作成を必要とする開発者にとっての開始点となります。SimpleTextParser では、simple name=value 構文が使用されます。
ClassSelectionParser	指定された形式のすべてのファイルにカスタム属性を追加します。実際の解析は実行しません。

解析オプション

Oracle 9*i*FS では、次の解析オプションが提供されています。

- **SimpleXmlParser**。最小限のカスタマイズ用です。ドラッグ・アンド・ドロップによるアップロードを使用して、FTP、SMB、Windows インタフェースおよび Oracle 9*i*FS Web ユーザー・インタフェース用に機能します。
- **ClassSelectionParser**。実際の解析は実行しません。ファイルをリポジトリに格納する前に、すべての .doc ファイルなど、特定のファイル拡張子を持つファイルに 1 つ以上のカスタム属性を追加できます。クラスを特定のファイル形式にマップします。

Oracle 9*i*FS のカスタム・パーサーの使用

.doc または .xls ドキュメントなど、XML 以外のドキュメントを解析する場合や、Vcards を表す .vcf のようなカスタム・タイプを定義している場合は、カスタム・パーサーを記述して、この種のドキュメントからデータベース・オブジェクトを作成する必要があります。カスタム・パーサーを作成するには、既存の Oracle 9*i*FS パーサーをサブクラス化する方法と、新規のカスタム・クラスを作成して `oracle.9iFS.beans.parsers.Parser` インタフェースを実装する方法があります。

Oracle 9*i*FS の XML 解析動作

Oracle 9*i*FS に XML で記述された文書を置くと、`SimpleXmlParser()` がコールされてドキュメント・オブジェクトが作成されます。

`gking.vcf` ドキュメント・インスタンスが XML ファイル `gking.xml` に変換されており、エンド・ユーザーが Oracle 9*i*FS の Windows インタフェースを使用しているとします。

1. gking.xml など、XML 文書のインスタンスを Oracle 9*i*FS のフォルダ /iFS/system/vcards にドラッグするとします。
2. SMB で、ファイル拡張子 .xml に基づいてパーサー検索を実行します。SMB は、Microsoft Windows 95、Windows 98 および Windows NT クライアント経由で Oracle 9*i*FS にアクセスするためのプロトコルです。ファイルを Oracle 9*i*FS との間でドラッグしたり、Oracle 9*i*FS 内で直接編集できます。
3. パーサー検索によって一致する XML ファイルが見つかり、`SimpleXmlParser()` がコールされます。
4. Vcard カスタム・クラス定義ファイルが以前に Oracle 9*i*FS に格納されているため、`SimpleXmlParser()` では `gking.xml` が Vcard ドキュメントとして認識されます。
5. `SimpleXmlParser()` が `gking.xml` を解析し、Vcard オブジェクト gking を作成します。

   gking.xml のかわりに gking.vcf ドキュメント・インスタンスが使用された場合、ステップ 2 のパーサー検索では SMB によりカスタム・パーサー VcardParser がコールされます。

パーサー・アプリケーションの作成

パーサー・アプリケーションを作成する手順は、次のとおりです。

1. パーサー・クラスの記述
2. パーサーの配置
3. パーサーのコール（パーサー・アプリケーション内）
4. ParserCallback の記述（オプション）

関連項目： http://otn.oracle.com/products/ifs

パーサーの詳細は、Oracle 9*i*FS の Javadoc の次のクラスを参照してください。

```
oracle.ifs.beans.parsers.SimpleXmlParser
oracle.ifs.beans.parsers.XmlParser
oracle.ifs.beans.parsers.SimpleTextParser
```

Oracle 9*i*FS での XML の表示

デフォルトでは、Oracle 9*i*FS の XML レンダラでは、XML 属性タグを含むオリジナル・ファイルが再構成されます。ファイルをダブルクリックすると、Oracle 9*i*FS により再構成され、使用する XML エディタで開かれます。

ファイルを再構成できるようにすることが重要ですが、ファイルをさらに加工することもできます。たとえば、XML ベースの保険料請求を解析する場合、顧客が Web ベースのセルフサービス・アプリケーションを通じてアクセスするときに、オリジナル・ファイルの一部のセクションのみを表示する必要があります。また、ファイルの内容から総額、件数および平均などの値を計算し、計算結果をファイルに追加することも必要です。さらに、ファイルの形式を変更し、読みにくい XML のかわりに RTF や HTML として表示できます。

このような作業はいずれも、Oracle 9*i*FS レンダラを通じて実行できます。XML ファイルの解析後のコンテンツは Oracle 9*i*FS で動的に再編集されてから、ユーザーまたはアプリケーション用に必要な情報の範囲、形式またはタイプが表示されます。特定のクラスの XML ファイル用に新規のレンダラを作成して登録すると、この種のファイルの Oracle 9*i*FS での表示方法を変更できます。

関連項目： http://otn.oracle.com/products/iFS。Oracle 9*i*FS カスタム・レンダラの作成方法の詳細は、「The XSL Custom Renderer Sample Application」と「technical brief」を順に選択してください。

XML およびビジネス・インテリジェンス

解析後の XML ファイルでは、各属性のデータが Oracle 9*i*FS スキーマ内で識別可能な別個の列に格納されるため、Oracle 9*i*FS では XML ファイル内に以前に格納されているビジネス・インテリジェンスをデータ・マイニング・アプリケーションで利用できます。

保険料請求の例に戻ると、ビジネス・インテリジェンスは請求ごとに格納され、システムに格納されている全請求の全情報の集計値は各部の合計を超えています。代理店による請求ファイルの挿入または更新の動向を、管理者がリアルタイムで追跡できるかどうかを考えます。Oracle 9*i*FS ファイルを使用してファイルを解析し、解析後のファイルの内容をデータ・マイニング・ツールに送れば、リアルタイムで追跡できます。

XML ファイルを使用した Oracle 9*i*FS の構成

Oracle 9*i*FS を構成するには、ファイル・システムをカスタマイズする XML を記述できます。たとえば、Oracle 9*i*FS の SDK には、ファイルとフォルダをサブクラス化する機能があります。XML ベースの発注書をアプリケーション用に別のファイル・タイプとして識別する必要がある場合は、Oracle 9*i*FS のサブクラスを定義できます。このサブクラスは、次の手順で簡単に作成できます。

1. サブクラスの定義に関する Oracle 9*i*FS の構文に従って XML ファイルを記述します。
2. この XML ファイルを、Oracle 9*i*FS 内の必要なフォルダにドラッグ・アンド・ドロップします。

この種の構成ファイルを使用すると、アプリケーションを配置するために複雑なスクリプトを記述する必要がなくなります。かわりに、XML 構成ファイルなど、必要なすべてのファイルを新しい Oracle 9*i*FS インスタンスにドラッグ・アンド・ドロップします。

第 III 部

XML データ交換

第 III 部では、XML データ交換の実装方法を示す事例について説明します。

第 III 部の内容は、次のとおりです。

- 第 7 章「XSL を使用した Discoverer 4i Viewer のカスタマイズ」
- 第 8 章「オンライン B2B XML アプリケーション : 手順」
- 第 9 章「Service Delivery Platform（SDP）と XML」

その他の例、アプリケーションおよび FAQ については、『Oracle9*i* アプリケーション開発者ガイド - XML』も参照してください。

7

XSL を使用した Discoverer 4*i* Viewer のカスタマイズ

この章の内容は、次のとおりです。

Discoverer 4*i* Viewer: 概要

使用する XML コンポーネント : Oracle XML Parser for Java、バージョン 2

Discoverer の概要

Discoverer Business Intelligence ソリューションは、組織のデータを情報へと変換します。Oracle Discoverer for the Web では、この情報に Web ブラウザ・インタフェースを使用してアクセスできます。

Discoverer 4*i* Viewer

Oracle Discoverer 4*i* Viewer は、情報をインターネットやイントラネット上のどこからでも使用できるようにして、Web ページへの透過的な埋込みや企業ポータルからのアクセスを可能にします。また、Oracle Discoverer 4*i* Viewer は、どんな Web サイトにも適合するように、XML や XSL などの標準的な Web テクノロジを使用してカスタマイズできます。

Oracle Discoverer を使用すると問合せができ、Oracle Reports を使用すると、HTML、Adobe 社の Portable Document Format（PDF）および XML のように様々な形式でレポートを公開できます。

Oracle Discoverer 4*i* Viewer のカスタマイズ

この章では、カスタマイズの例を示し、Discoverer 4*i* Viewer の使用方法について説明します。

- XML と XSL: Discoverer 4*i* Viewer では、業界標準の XML を使用してデータやアプリケーションの状態が表示され、XSL スタイルシート言語を使用してユーザー・インタフェースがフォーマットされます。標準的な XSL ツールを使用して、ユーザー・インタフェースをカスタマイズしたり、完全な埋込みビジネス・インテリジェンス・アプリケーションを生成できます。
- HTML: 単一のカスタマイズ・ファイル内で HTML フォーマット属性を指定できます。特に、HTML フォーマットに慣れている場合、フォント、色およびグラフィックスを簡単に変更できます。

Discoverer 4*i* Viewer は、中間層の B2B アプリケーションで駆動してアクセスできます。

Discoverer 4*i* Viewer の詳細情報

Discoverer 4*i* Viewer の詳細情報は、次の URL を参照してください。

http://otn.oracle.co.jp/document/products/bitools/discoverer.html

Discoverer 4*i* Viewer: 機能

Discoverer 4*i* Viewer では、次の作業を実行できます。

- **レポートの実行**
  - データベースに保存されたレポートの動的な実行。
  - 実行時のパラメータ入力。
  - スケジューリングされたレポートの実行。
  - 問合せの取消し。
  - ページ軸上のディメンション値間のページング。
  - ワークブック・オプションとデータベース・オプションの変更。
  - レポートの出力。
  - HTML、Excel および他の PC ファイル形式など、各種ファイル形式によるレポートのエクスポート。
- **データの分析**
  - ドリルダウン分析の実行。
  - 様々なレベルのサマリー・データへのドリル。
  - Web ページ、Microsoft Word ドキュメントなど、他のアプリケーションに保持されているデータへのドリル・アウト。
  - 軸間でのデータのピボットによる有効な分析および比較のための配置。
- **問合せの制御**
  - 問合せの実行時間の制御。時間のしきい値に達しても問合せが実行中の場合は、自動的に終了します。
  - 問合せ見積りの表示。問合せが事前定義済みの時間しきい値より長時間かかると予測される場合、Discoverer Viewer では問合せを実行するかどうかを判断できるように警告が表示されます。
  - サマリー表への問合せの自動的なリダイレクト。サマリー・データ要求は、事前に集計済みのデータを含むサマリー表に自動的にリダイレクトされます。
- **データ・アクセスの保護**
  - Web サーバーおよびデータベースのセキュリティ機能の活用。
  - 複数のファイアウォールの通過。
  - SSL、x.509 および他の標準的な Web セキュリティ・プロトコルのサポート。
  - データへのモバイル・アクセスのサポート。

- **ブラウザ・オプションの使用**
  - よく使用するレポートへのブックマークの設定。
  - 他の Web ページへのレポートの埋込み。
  - ブラウザ・オプションの変更によるフォント・サイズとリンク・スタイルの変更。
  - 詳細なスプレッドシート分析のための、Excel のような他の形式への出力のエクスポート。
  - レポートのオフライン表示（Internet Explorer 5）。

  Discoverer 4*i* Viewer では Java もフレームも使用しないため、低スペックのブラウザでも使用できます。JavaScript が使用可能な場合、Discoverer ではユーザー・インタフェースの拡張に使用されます。ただし、JavaScript は必須ではありません。使用可能でない場合、ユーザー・インタフェースのパフォーマンスはやや低下します。
- Discoverer レポートの埋込み
  - 既存の Web ページでは、組み込むワークブックとワークシートを定義する URL を指定します。リンクをクリックすると、データベースが問合せされ、最新データが HTML 形式で表示されます。
  - Oracle Portal（または *i*Portal、以前の WebDB）などのポータルでは、ホスティング・ポータルの表示を使用できます。
- 完全なカスタム Web アプリケーションの構築や、他の中間層 Web システムへのデータ配信に使用

Discoverer 4*i* Viewer には、次の用途があります。

- スタンドアロンのビジネス・インテリジェンス・ツールとして使用します。
- データベースの出力を Web サイトおよびポータルと統合します。
- Web サイトの表示に合せてカスタマイズし、会社のロゴなどを取り込んだり、Web 用にカスタムの Discoverer アプリケーションを構築します。

Discoverer 4*i* Viewer: アーキテクチャ

図 7-1 に、Discoverer 4*i* Viewer のアーキテクチャを示します。

Discoverer 4*i* Viewer のコンポーネントは、次のとおりです。

- Oracle Discoverer Application Server。Discoverer Web ソリューション用のエンジンです。
- Apache や Apache JServ（JVM）などの Web サーバーとサーブレット・コンテナ。
- Oracle XML Parser for Java v2。これには XSLT Processor が含まれます。
- Discoverer 4*i* Viewer Servlet。
- Discoverer Server Interface、Java モジュール。
- Oracle9*i* データベース。

Discoverer 4*i* Viewer の動作

Discoverer 4*i* Viewer の動作は、図 7-1 を参照してください。

1. Discoverer 4*i* Viewer が、他の Web サイトと同様に、標準的な Web ブラウザから URL 経由で起動されます。この URL は、Web サーバー上で実行される Discoverer 4*i* Viewer Servlet により処理されます。

   このサーブレットは、Discoverer Server Interface（モデル）を使用して Discoverer Application Server と通信します。Discoverer Server Interface と Discoverer Application Server は、どちらも Discoverer Plus で使用されます。

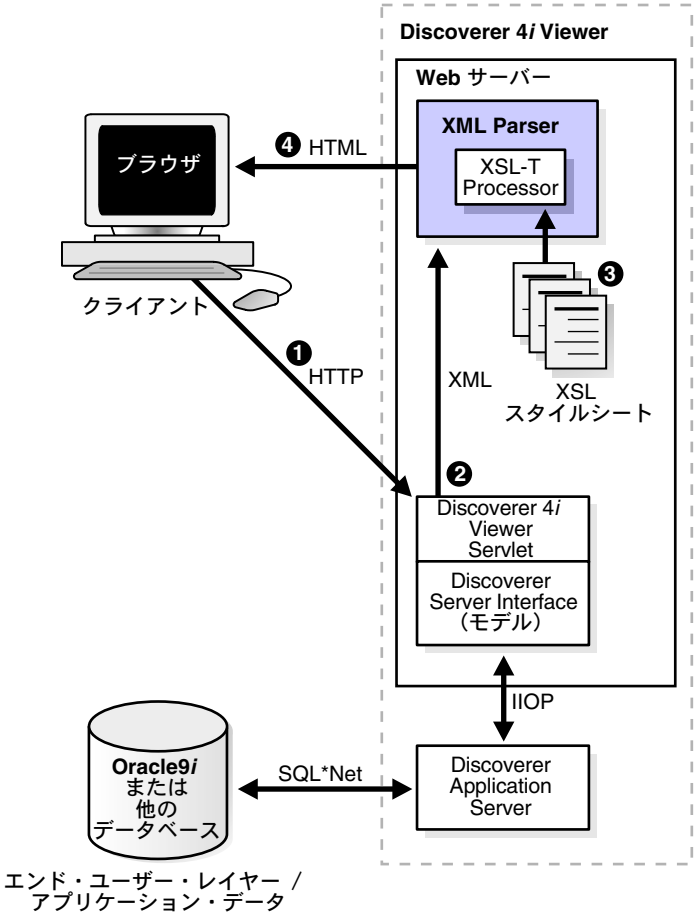
   Discoverer Server Interface。これはアプレットですが、ここでは Discoverer Plus のようにクライアントの JVM で実行されるのではなく、Web サーバー上で実行されています。 Discoverer 4*i* Viewer Servlet は、CORBA IIOP プロトコルを使用して Discoverer Application Server と通信します。

   Discoverer 4*i* Viewer と Discoverer Plus は、同じ Discoverer Application Server を使用します。

2. Discoverer 4*i* Viewer Servlet はクライアント・ブラウザからの HTTP 要求を解析し、Discoverer Application Server に対して必要なコールを行います。サーバー応答は、サーブレットにより生成された XML で表され、XML/XSL プロセッサ（XSLT Processor）に送信されます。
3. これにより、XML はユーザー・インタフェースの表示を定義する XSL 構成ファイルと結合されます。
4. XSLT Processor により HTML が生成され、元のブラウザに送信されます。

   これは、Discoverer 4*i* Viewer のユーザー・インタフェースを個々のサイト用にカスタマイズするための XSL ファイルです。

図 7-1 Discoverer 4*i* Viewer のアーキテクチャ



Discoverer Application Server のレプリケート

Web サーバーと Discoverer 4*i* Viewer Servlet コンテナは、標準的な Web フレーミングおよび仮想ホスティング・テクニックを使用してレプリケートできます。

実際のシステムでは、多数のユーザーが各 Web サーバーおよびアプリケーション・サーバーを使用しています。Discoverer では、使用可能なマシン間で負荷を分散させる方法を正確に決定できます。

カスタマイズされた Web アプリケーションへの Discoverer 4*i* Viewer の使用

Discoverer 4*i* Viewer では、次の XML コンポーネントを使用して HTML が生成されます。

- XML。使用可能な情報を記述します。
- XSLT Processor および XSL スタイルシート。情報を HTML で表示する方法を定義します。

XSL 構成ファイル（スタイルシート）では、使用するフォントや色など、単純な属性が定義されるのみでなく、各ページのレイアウトやユーザーとの相互作用も定義されます。XSL スタイルシートをカスタマイズすると、特定の Discoverer アプリケーションを構築して Web 上で配信できます。

注意： ここで説明するアプリケーションは、Internet Explorer 5.x ブラウザで動作します。

ステップ 1: ブラウザからの URL の送信

ログイン後に Discoverer Viewer から、ビジネス分析のためにオープンできるワークブックのリストを要求したとします。発行される URL は、http://ukp14910.uk.oracle.com/disco/disco4iv?us=video&db=Disco です。

この URL では、サーブレットがインストールされているマシン、ユーザー名および使用するデータベース接続文字列を指定します。通常は、セキュリティのためにパスワードは URL に表示されません。

ステップ 2: サーブレットによる XML の生成

Discoverer 4*i* Viewer Servlet が URL を処理します。Discoverer Application Server に対して、このユーザーのセキュリティ設定をチェックし、このユーザーがアクセスを許可されているワークブックの詳細を戻すように指示します。

セキュリティ設定は、データベースのエンド・ユーザー・レイヤー表に保持されています。この情報が Discoverer Application Server から戻されると、サーブレットは、戻された 3 つのワークブックに関する情報を表示できるように、次の XML を生成します。

- Store and Band Analysis ワークブック
- Video Sales Analysis ワークブック
- Annual Sales Report ワークブック

Discoverer XML の例 1: 3 つのワークブックのレポート・データ

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="example1.xsl"?>
<discoverer version="3.5.8.12" login_method="discoverer">
 <account name="myname@mydatabase" ref="MYNAME%40mydatabase">
  <user>MYNAME</user>
   <database>mydatabase</database>
   <eul default="true" name="myeul">
   <workbook name="Store and Band Analysis" ref="Store~20and~20Band~20Analysis">
      <description>Shows sales by Store, broken into sales bands</description>
   </workbook>
   <workbook name="Video Sales Analysis" ref="Video~20Sales~20Analysis">
      <description>General purpose analysis of the business</description>
   </workbook>
   <workbook name="Annual Sales Report" ref="Annual~20Sales~20Report">
      <description>Shows yearly and quarterly sales of products</description>
   </workbook>
   </eul>
 </account>
</discoverer>
```

注意： これらのワークブック名と記述がユーザーに表示される方法に関する情報は、XML に含まれていません。これは、XSL ファイルの機能です。

ステップ 3: XSLT Processor による XSL スタイルシートの適用

XSL は、W3C により定義された業界標準のスタイルシート言語です。XSL を使用すると、XML ファイルから要素を選択して HTML テンプレートと結合し、Web ブラウザ用の HTML 出力を生成できます。

Discoverer 4*i* Viewer のユーザー・インタフェース全体が XSL で定義されています。つまり、このユーザー・インタフェース（UI）をカスタマイズしてコピーし、HTML エディタや XSL エディタなどの標準的な Web 開発ツール、あるいは単純なテキスト・エディタを使用して、ユーザー・インタフェースの代替スタイルを定義できます。

ステップ 4: XSLT Processor による HTML の生成

XSL と XML を使用します。

ステップ 2 で生成された XML と、標準的な Discoverer 4*i* Viewer の XSL 構成ファイル（スタイルシート）を使用して、この 2 種類のファイルが XML Parser for Java v2 の XSLT Processor 内で結合されます。これにより、XML 文書の HTML バージョンが生成されます。

この HTML が、初期 URL への応答として元のブラウザに送信されます。

Discoverer 4*i* Viewer の場合、生成された HTML ではフレームが使用されないため、ブラウザやインターネット・デバイスの負荷は最小限になります。したがって、他の Web アプリケーションやポータルと簡単に統合できます。JavaScript が使用可能な場合、Discoverer ではユーザー・インタフェースの拡張に使用されます。ただし、JavaScript は必須ではありません。使用可能でない場合、ユーザー・インタフェースのパフォーマンスはやや低下します。

XSL スタイルシート・ファイルの変更によるスタイルのカスタマイズ : style.xsl

社内の標準に合せてフォントや色を簡単に変更したり、会社のロゴを表示してブランド・イメージを持たせることができるようにする必要があります。このようなグローバル変更は、変更できるスタイルごとに特別なタグを定義する単一の XSL スタイルシート・ファイル style.xsl で実行できます。次に例を示します。

- **ロゴの挿入**。ロゴを挿入するには、次の行を変更します。

  ```
  <xsl:variable name="logo_src"> </xsl:variable name>
  ```

  変更後の行は、次のようになります。

  ```
  <xsl:variable name="logo_src"> http:www.mycompany.com/images/mylogo.gif
  </xsl:variable name>
  ```

- **文字の色の変更**。文字の色を変更するには、次の行を変更し、適切なカラー・コードを追加します。

  ```
  <xsl:variable name="text_color">#000000</xsl:variable>
  ```

この方法で多数のスタイルを一括変更できますが、ユーザー・インタフェース全体の操作は変更されません。これは、Discoverer 4*i* Viewer をカスタマイズする唯一の方法です。実際には、XSL を使用すると、次の例に示すように完全にカスタマイズされたアプリケーションを作成できます。

Discoverer 4*i* Viewer: XML および XSL を使用したカスタマイズの例

次の XML および XSL フラグメントを使用して、Web ブラウザでカスタマイズを試行できます。

ステップ 1: XML ファイル

このデータは、前述の「Discoverer XML の例 1」と同様に標準的な XML ファイルです。

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="example1.xsl"?>
<discoverer version="3.5.8.12" login_method="discoverer">
 <account name="myname@mydatabase" ref="MYNAME%40mydatabase">
  <user>MYNAME</user>
   <database>mydatabase</database>
   <eul default="true" name="myeul">
   <workbook name="Store and Band Analysis" ref="Store~20and~20Band~20Analysis">
      <description>Shows sales by Store, broken into sales bands</description>
   </workbook>
   <workbook name="Video Sales Analysis" ref="Video~20Sales~20Analysis">
      <description>General purpose analysis of the business</description>
   </workbook>
   <workbook name="Annual Sales Report" ref="Annual~20Sales~20Report">
      <description>Shows yearly and quarterly sales of products</description>
   </workbook>
   </eul>
 </account>
</discoverer>
```

この XML ファイルは XML バージョンの指定から始まります。2 行目で、データの処理に適用する XSL ファイル example1.xsl と、このファイルの残りの部分が Discoverer 4*i* Viewer から生成されるように指定しています。

この例に最初の 2 行が追加されているため、テキスト・エディタを使用してファイルにテキストを入力し、それを Web ブラウザで開いて、XSL の変更結果を視覚的に確認できます。この操作が必要な場合は、拡張子「xml」を指定してファイルを保存してください。

ステップ 2: XSL ファイル example1.xsl

XSL ファイル example1.xsl の内容は、次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html>
    <body bgcolor="#ffffff" link="#663300" text="#000000">
       <b><i>Choose a Workbook:</i></b>
```

```
        <br/>
        <table border="2">
          <xsl:for-each select="/discoverer/account/eul/workbook">
           <tr>
             <td width="242">
               <font face="sans-serif">
                 <xsl:value-of select="@name"/>
               </font>
             </td>
             <td>
                <xsl:value-of select="description"/>
             </td>
           </tr>
         </xsl:for-each>
        </table>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

ステップ 3: XML+XSL = HTML

図 7-2 に、XML ファイルをブラウザで開いたときに、ブラウザが XSL スタイルシート（example1.xsl）に読み込んで HTML を生成する内容を示します。

図 7-2 ブラウザに表示されるワークブックのリスト、XML+ example1.xsl=HTML ― 変更前

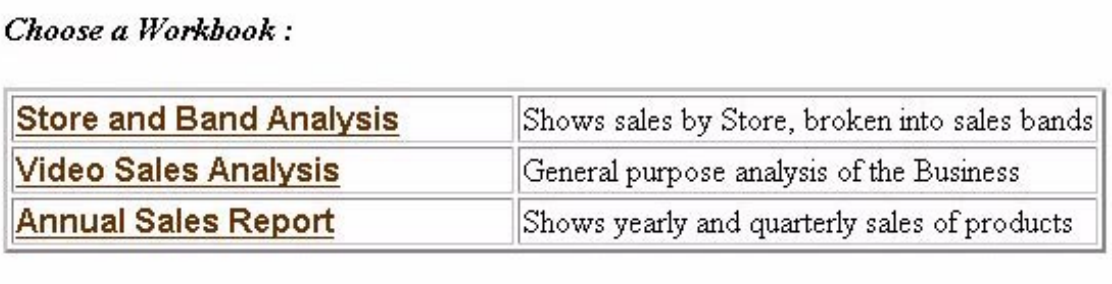


表 7-1 に、XSL ファイル example1.xsl の 5 行目以降を示します。ここでは、HTML の生成方法が記述されています。このファイルは XML バージョンの指定から始まります。2 行目は、このファイルがスタイルシートであることを示しています。HTML テンプレートは 4 行目の <HTML> タグから始まります。

表 7-1 example1.xsl の検査 ー XSL ファイルの変更前

example1.xsl のコード	コードの意味
<body bgcolor="#ffffff" link="#663300" text="#000000">	この行では、使用する色を定義します。
Choose a Workbook :	これは単なる HTML ではなく、太字でイタリックのフォントを設定し、テキスト「Choose a workbook :」を挿入しています。
<table border="2">	HTML の表が、二重線の枠付きで始まります。
<xsl:for-each select="discoverer/account/eul/workbook">	これが実際の最初の XSL コードで、意味は次のとおりです。 XML データ・ファイルにアクセスし、workbookinfo タグごとに終了タグ </xsl:for-each> に達するまで、以降のすべてのステップを実行します。 XML ファイルに表示されるワークブックごとに次の XSL が処理され、見つかったワークブックごとに HTML の表に 1 行が挿入されます。
<tr> <td width="242"> <xsl:value-of select="@name"/> </td>	<tr> で、表の新規行が始まります。 <td> は、1 列目に挿入する表データを定義します。列の幅は 242 ピクセル、フォントは Sans-Serif に設定されています。 XSL 行により、各 workbookinfo セクションの下の <NAME> タグに XML ファイルからのテキストが挿入されます。
<td> <xsl:value-of select="description"/> </td> </tr>	これらの行では、HTML の表の 2 列目が定義され、XML ファイル内の <DESCRIPTION> タグを使用してワークブックの記述テキストが挿入されます。HTML の表の各行には、クリックできるようにリンクになっているワークブック名と、テキストで表示されたワークブックの記述が含まれます。XML ファイルには 3 つのワークブックがあるため、表は 3 行になります。

注意：

- この例は、Discoverer 4*i* Viewer によるワークブック・リストの表示方法を正確に示しているわけではありません。この例はわかりやすいように単純化されていますが、XSL スタイルシートにより出力の表示がどのように制御されるかを示しています。代表的な表示については、図 7-4 を参照してください。
- Discoverer 4*i* Viewer では、XML と XSL は Web ブラウザではなく中間層の XSLT Processor 内で結合されます。

ステップ 4: XSL スタイルシートのカスタマイズ（example2.xsl）

XSL スタイルシートは次のように変更されています。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
 <html>
  <body bgcolor="#ffffff" link="#663300" text="#000000">
   <table border="0">
      <tr>
        <td>width="500" height="100" background="disco_banner.gif">
          <font face="sans-serif">
            <b>Performance Reports</b>
          </font>
        </td>
      </tr>
   </table>
   <table border="0">
      <xsl:for-each select="/discoverer/account/eul/workbook">
         <tr>
           <td>
             <font face="sans-serif">
               <b>
                 <a href="link.htm">
                    <img src="button2.gif">
                      <xsl:attribute name="alt">
                      <xsl:value-of select="description"/>
                      </xsl:attribute>
                    </img>
                 </a>
               </b>
             </font>
           </td>
           <td>
```

```
            <font face="sans-serif">
               <xsl:value-of select="@name"/>
            </font>
          </td>
        </tr>
      </xsl:for-each>
   </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

このスタイルシートを同じ XML と結合すると、図 7-3 のように表示されます。

図 7-3 同じ XML と変更後の XSL スタイルシートを使用して表示されたワークブック・リスト

この場合、ユーザー・インタフェースの表示がまったく異なっており、より多くのグラフィックが使用されています。テキスト・リンクのかわりに、レポート実行用のグラフィカ

ルなボタンがあり、各ボタンにはマウスを置いたときに表示される動的なツール・ヒントが付いています。

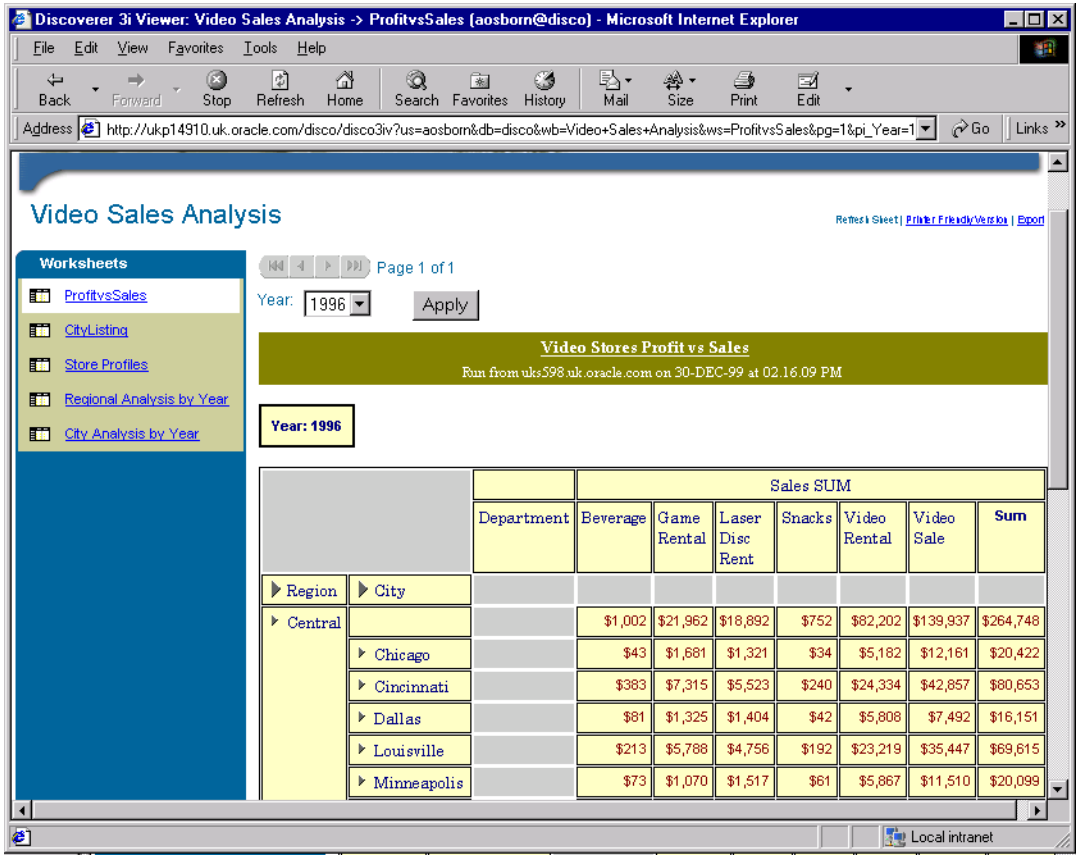
変更後の XSL ファイルについては、表 7-2 を参照してください。

図 7-4 に、このサンプル・アプリケーションの代表的な Web ベースの表示を示します。

表 7-2 example2.xsl の検査 ー XSL ファイルの変更後

example2.xsl のコード	コードの意味
<table border="0"> <tr> <td width="500" height="100" background="disco_banner.gif"> Performance reports </td> </tr> </table>	これらの行では、表を作成し、グラフィックと見出し「Performance Reports」を挿入します。
<table border="0"> <xsl:for-each select="discoverer/account/eul/workbook">	前述の例と同様に、ワークブック名が表示されるメイン表がここから始まりますが、表の枠線がなく、行の定義が異なります。
<tr> <td> <img src="button2.gif"> <xsl:attribute name="alt"> <xsl:value-of select="description"/> </xsl:attribute> </img> 	最初の表のデータ列もハイパーリンクとして定義されていますが、この場合はイメージ button2.gif がテキスト・リンクではなくイメージとして使用されています。使用フォントは Sans-Serif です。 イメージ上にツール・ヒントを表示するために、HTML の alt 属性が使用されています。 通常、alt 属性は次のように単純なテキスト文字列に使用します。 <img src="button2.gif" alt="Tooltip text to appear when a mouse is over the image"> ただし、ここでは動的なツール・ヒントが必要なため、XML ファイルの <description> タグからテキストを取得して、alt タグを生成しています。そのために、<xsl:attribute> タグが使用されています。 <xsl:value-of select="description"/>。2 番目のデータ列では、前述の例と同様に XSL を使用して XML ファイルから取得し、表示するワークブックの名前を選択しています。

図 7-4 Discoverer 4*i* Viewer: ビジネス・ソリューションとして代表的な Web ベースの表示



FAQ: Discoverer 4*i* Viewer

サーブレットの説明

質問

サーブレットとは何ですか。

回答

サーブレットは、サーバー・アプリケーション上で実行され、クライアントからの要求に応答する Java コードのモジュールです（このため「サーブレット」と呼ばれます。同様に、クライアント側では「アプレット」と呼ばれます）。サーブレットは特定のクライアント / サーバー・プロトコルには連結されておらず、HTTP で最も一般的に使用されます。また、通常、「サーブレット」という用語は「HTTP サーブレット」を意味します。

サーブレットにより、パッケージ javax.servlet（基本的なサーブレット・フレームワーク）および javax.servlet.http（HTTP 要求に応答するサーブレット用のサーブレット・フレームワークの拡張）内の Java 標準拡張クラスが使用可能になります。サーブレットは移植性の高い Java 言語で記述され、標準的なフレームワークに準拠しているため、サーバーやオペレーティング・システムに依存しない方法で、高度なサーバー拡張機能を作成できます。

HTTP サーブレットの代表的な用途は、次のとおりです。

- HTML フォームから送信されたデータの処理と格納。
- 動的コンテンツの提供。たとえば、データベースの問合せ結果をクライアントに戻すなどの処理です。
- ステートレス HTTP の最上位での状態情報の管理。たとえば、多数の顧客のショッピング・カートを同時に管理し、それぞれの要求を適切な顧客にマップする、オンライン・ショッピング・カート・システムなどです。

Discoverer 4*i* Viewer とブラウザ間の通信

質問

Discoverer 4*i* Viewer はユーザーのブラウザとの通信に何を使用しますか。

回答

HTTP と HTML を使用します。

Discoverer 4*i* Viewer と XML

質問

Discoverer 4*i* Viewer では XML がどのように使用されますか。

回答

XML は中間層で生成され、アプリケーションの状態を表します。Discoverer 4*i* Viewer Servlet はユーザーのブラウザからの HTTP 要求を解析し、Discoverer Server に対して必要なコールを行います。

サーバーからの応答は、サーブレットにより生成された XML で表示されます。この XML に XSL が適用され、ユーザーのブラウザに表示される HTML が生成されます。

XML と XSL を使用すると、基礎となるデータや表示をカスタマイズしやすいように分離できます。

disco4iv.xml

質問

disco4iv.xml ファイルの役割は何ですか。

回答

disco4iv.xml ファイルを使用すると、各種オプションを構成して、Discoverer 4*i* Viewer に必要な動作を指定できます。たとえば、接続先の Discoverer セッションを指定できます。

Discoverer 4*i* と XSL

質問

Discoverer 4*i* Viewer では XSL がどのように使用されますか。

回答

Discoverer 4*i* Viewer では、XSL（または XSLT）を使用して、中間層で生成された XML がユーザーのブラウザに送信される HTML に変換されます。XSL ファイルを編集すると、UI のスタイルと表示を全面的に制御できます。

サポートされる XSLT プロセッサ

質問

Discoverer 4*i* Viewer ではどんな XSL プロセッサが使用されますか。

回答

Discoverer 4*i* Viewer は、Oracle XSLTProcessor を使用するように構成できます。これはデフォルトで、XDK for Java に付属しています。

XSL エディタ

質問

XSL スタイルシートの編集には、どんなツールを使用できますか。

回答

どんなテキスト・エディタでも XSL ファイルは編集できますが、次のアプリケーションは XSL の編集専用に設計されています。

- eXcelon Stylus
- IBM XSL Editor
- XML Spy

スタイルシートのカスタマイズ

質問

通常、スタイルシートをカスタマイズするには何を変更しますか。

回答

スタイルシートをカスタマイズするには、次の項目を編集します。

- **disco4iv.xsl**。HTML ページのタイプと、その表示規則を定義します。
- **page_layouts.xsl**。各種 HTML ページ全体のレイアウトを定義します。
- **gui_components.xsl**。ページ・レイアウトに使用する各 GUI コンポーネントの表示を定義します。
- **style.xsl**。Discoverer 4*i* Viewer で使用する各種フォントのスタイルを定義します。

- **errors.xsl**。独自のカスタム・エラー・メッセージを作成します。
- **functions.xsl**（お薦めしません）。他の XSL ファイルで使用される中心的な機能の動作を変更します。
- **scripts.xsl**（お薦めしません）。UI の拡張に使用する JavaScript を変更します。
- **render_table.xsl**（お薦めしません）。表 / クロス集計の表示方法を変更します。

スタイルシートの変更結果の確認

質問

独自の XSL スタイルシートをカスタマイズする場合に、変更結果を確認できないのはなぜですか。

回答

デフォルトでは、XSLT Processor はパフォーマンスを向上させるために XSL ファイルをメモリーにキャッシュします。変更結果を確認するには、次の 2 つの方法があります。

- 新たな変更結果の確認が必要になるたびに、サーブレットを（Web サーバーを再起動して）再起動します。これにより、サーブレットでは XSL ファイルがディスクから再び読み込まれます。
- XSL キャッシングを無効化します。disco4iv.xml ファイルの <document> セクションに次の行を追加し、Web サーバーを再起動します。

```
<argument name="xsl_cache">false</argument>
```

ブラウザに何も表示されない場合

質問

ブラウザに何も表示されないのはなぜですか。

回答

通常、これは次のどちらかの操作が原因です。

- 存在しない XSL テンプレートをコールした場合。
- 存在しない変数の使用を試みた場合。

XML と XSL の詳細情報

質問

XML と XSL の詳細情報は、どこで提供されていますか。

回答

- http://java.sun.com/docs/books/tutorial/servlets/l
- http://www.w3.org/Style/XSL/
- http://www.w3.org/XML/
- http://www.builder.com/Authoring/XmlSpot/?tag=st.cn.sr1.ssr.bl_xml
- http://zvon.vscht.cz/HTMLonly/XSLTutorial/Books/Book1/bookInOne.html
- http://www.arbortext.com/Think_Tank/Norm_s_Column/Issue_One/Issue_One.html
- http://www.dpawson.co.uk/xsl/sect21.html

Discoverer Viewer XML の DTD

質問

Discoverer Viewer XML はどんな構造を持っていますか。

回答

Discoverer Viewer で生成される XML 文書は、次の DTD に従っています。

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT account (error*, user?, database?, connect?, role*, eul*, option*,
version*)>
<!ATTLIST account
    name CDATA #IMPLIED
    mv_summaries_supported (true | false) "true"
    ref CDATA #IMPLIED
>

<!ELEMENT axes (axis)*>

<!ELEMENT axis (item*)>
<!ATTLIST axis
    position (m | x | y | z) #REQUIRED
>
```

```
<!ELEMENT background_color EMPTY>
<!ATTLIST background_color
    red CDATA #REQUIRED
    green CDATA #REQUIRED
    blue CDATA #REQUIRED
>

<!ELEMENT cell EMPTY>
<!ATTLIST cell
    result CDATA #REQUIRED
>

<!ELEMENT chart (page_item*, dim:image_map)>
<!ATTLIST chart
    name CDATA #REQUIRED
    height CDATA #REQUIRED
    width CDATA #REQUIRED
>

<!ELEMENT command (#PCDATA)>
<!ATTLIST command
    name CDATA #REQUIRED
    ref CDATA #IMPLIED
    implied (true | false) "false"
    valid (true | false) "true"
>

<!ELEMENT connect (#PCDATA)>

<!ELEMENT data (value, qdr?)>

<!ELEMENT database (#PCDATA)>

<!ELEMENT date (#PCDATA)>
<!ATTLIST date
    ref CDATA #IMPLIED
>

<!ELEMENT description (#PCDATA)>
<!ELEMENT error (#PCDATA | command)*>
<!ATTLIST error
code CDATA #REQUIRED
severity CDATA #IMPLIED
>

<!ELEMENT discoverer (session, request, account*, export*, locale?, version*)>
<!ATTLIST discoverer
```

```
    login_method (application | discoverer) "discoverer"
>

<!ELEMENT drill (#PCDATA)>
<!ATTLIST drill
    ref CDATA #IMPLIED
>

<!ELEMENT drill_path (#PCDATA)>
<!ATTLIST drill_path
    name CDATA #REQUIRED
    hierarchy_name CDATA #IMPLIED
    direction (collapse | up | down) #REQUIRED
    level CDATA #REQUIRED
    ref CDATA #REQUIRED
>

<!ELEMENT edge (item*, edge_row*)>
<!ATTLIST edge
    placement (page | side | top) #REQUIRED
>

<!ELEMENT edge_row (value*)>

<!ELEMENT eul (workbook*, version*)>
<!ATTLIST eul
    name CDATA #REQUIRED
    default (true | false) "false"
    ref CDATA #IMPLIED
>

<!ELEMENT export (#PCDATA)>
<!ATTLIST export
    name CDATA #REQUIRED
    ref CDATA #IMPLIED
    format CDATA #REQUIRED
>

<!ELEMENT font EMPTY>
<!ATTLIST font
    name CDATA #REQUIRED
    size CDATA #REQUIRED
    bold (true | false) "false"
    italic (true | false) "false"
    strikeout (true | false) "false"
    underline (true | false) "false"
>
```

```
<!ELEMENT foreground_color EMPTY>
<!ATTLIST foreground_color
    red CDATA #REQUIRED
    green CDATA #REQUIRED
    blue CDATA #REQUIRED
>

<!ELEMENT format (background_color, foreground_color, graphic_bar_color?, font)>
<!ATTLIST format
    id CDATA #REQUIRED
    description CDATA #IMPLIED
    display_name CDATA #IMPLIED
    horizontal_alignment (left | center | default | right) #REQUIRED
    vertical_alignment (bottom | center | top | lower_bound | upper_bound) #REQUIRED
    graphic_bar_visible (true | false) "false"
    word_wrap (true | false) "false"
>

<!ELEMENT format_map (format*)>

<!ELEMENT graphic_bar_color EMPTY>
<!ATTLIST graphic_bar_color
    red CDATA #REQUIRED
    green CDATA #REQUIRED
    blue CDATA #REQUIRED
>

<!ELEMENT group (value*, group*, data*)>

<!ELEMENT item (drill_path*, sort*)>
<!ATTLIST item
    name CDATA #REQUIRED
    key CDATA #REQUIRED
    id CDATA #IMPLIED
    format_class CDATA #IMPLIED
    heading CDATA #IMPLIED
>

<!ELEMENT layout (row*)>

<!ELEMENT locale (#PCDATA)>
<!ATTLIST locale
    language CDATA #REQUIRED
    country CDATA #REQUIRED
>
```

```
<!ELEMENT measure_edge (item+)>
<!ATTLIST measure_edge
    placement CDATA #REQUIRED
    level CDATA #REQUIRED
>

<!ELEMENT option (#PCDATA)>
<!ATTLIST option
    name (aq | ftd | msa | nad | nv | qif | qll | qpw | qrl | qtl | rpp | usd)
#REQUIRED
    enable (true | false) "false"
>

<!ELEMENT page_item (drill_path*, sort*, value+)>
<!ATTLIST page_item
    name CDATA #REQUIRED
    ref CDATA #IMPLIED
    key CDATA #IMPLIED
    id CDATA #IMPLIED
    format_class CDATA #IMPLIED
    heading CDATA #IMPLIED
>

<!ELEMENT parameter (value, prompt)>
<!ATTLIST parameter
    name CDATA #REQUIRED
    ref CDATA #IMPLIED
    description CDATA #IMPLIED
    lov_exists (true | false) "false"
    multivalued (true | false) "false"
    wildcard_supported (true | false) "false"
    type CDATA #REQUIRED
>

<!ELEMENT prompt (#PCDATA)>

<!ELEMENT qdr (#PCDATA)>

<!ELEMENT query (parameter*, axes, sheet_data?, chart?, drill?)>
<!ATTLIST query
    version CDATA #REQUIRED
    status CDATA #REQUIRED
    step CDATA #REQUIRED
    elapsed CDATA #IMPLIED
    estimate CDATA #REQUIRED
>
```

```
<!ELEMENT request (error*, command*)>
<!ATTLIST request
    source CDATA #REQUIRED
    parameters CDATA #IMPLIED
>

<!ELEMENT role (#PCDATA | security_group)*>
<!ATTLIST role
    name CDATA #REQUIRED
    ref CDATA #IMPLIED
    current (true | false) "false"
>

<!ELEMENT row (cell*)>

<!ELEMENT security_group (#PCDATA)>

<!ELEMENT session EMPTY>
<!ATTLIST session
    id CDATA #REQUIRED
>

<!ELEMENT sheet_data (page_item*, format_map?, edge*, measure_edge?, group*,
error*)>
<!ATTLIST sheet_data
    name CDATA #REQUIRED
    row_range_begin CDATA #REQUIRED
    row_range_end CDATA #REQUIRED
    total_rows CDATA #REQUIRED
    mode (inline | outline) #REQUIRED
>

<!ELEMENT sheet_layout (axis+)>

<!ELEMENT sort EMPTY>
<!ATTLIST sort
    type (none | group | hidden | page) #REQUIRED
    direction (hi_lo | lo_hi) #REQUIRED
    line_width CDATA #IMPLIED
    spaces CDATA #IMPLIED
    level CDATA #IMPLIED
>

<!ELEMENT time (#PCDATA)>
<!ATTLIST time
    ref CDATA #IMPLIED
>
```

```
<!ELEMENT title (#PCDATA)>

<!ELEMENT user (#PCDATA)>

<!ELEMENT value (#PCDATA | drill_path)*>
<!ATTLIST value
    current (true | false) "false"
    default (true | false) "false"
    wildcard (true | false) "false"
    ref CDATA #IMPLIED
    format_class CDATA #IMPLIED
    item_class CDATA #IMPLIED
    type (item | spacing | total) #IMPLIED
    id CDATA #IMPLIED
    data CDATA #IMPLIED
    label CDATA #IMPLIED
>

<!ELEMENT version EMPTY>
<!ATTLIST version
    component CDATA #REQUIRED
    product CDATA #REQUIRED
    version CDATA #REQUIRED
>

<!ELEMENT workbook (description?, worksheet*, date?, time?)>
<!ATTLIST workbook
    name CDATA #REQUIRED
    ref CDATA #IMPLIED
>

<!ELEMENT worksheet (description?, sheet_layout?, title?, layout?, query*)>
<!ATTLIST worksheet
    name CDATA #REQUIRED
    ref CDATA #REQUIRED
>

<!ELEMENT dim:image_map (GraphMap)>
<!ATTLIST dim:image_map
    xmlns:dim CDATA #REQUIRED
>

<!--
The GraphMap and related entities are provided by the BI Beans team.
-->
<!ELEMENT GraphMap (DataLine | DataMarker | TwoDMarker | StockMarker | AreaMarker |
```

```
ThreeDMarker | LegendMarker |
    LegendText | MarkerText | PieLabel | Slice | SliceLabel | O1TickLabel |
X1TickLabel | O1Title | X1Title |
    Y1TickLabel | Y1Title | Y2TickLabel | Y2Title | ZTickLabel | ZTitle | Title |
Subtitle | Footnote)*>

<!ELEMENT DataLine (Group, Series, Tooltip?, Geometry)>

<!ELEMENT DataMarker (Group, Series, Tooltip?, Geometry)>
<!ELEMENT TwoDMarker (Group, Series, Tooltip?, Geometry)>
<!ELEMENT StockMarker (Group, Series, Tooltip?, Geometry)>
<!ELEMENT AreaMarker (Group, Series, Tooltip?, Geometry)>
<!ELEMENT ThreeDMarker (Group, Series, Tooltip?,Geometry)>
<!ELEMENT LegendMarker (Series, Geometry)>
<!ELEMENT LegendText (Series, Geometry)>
<!ELEMENT MarkerText (Group, Series, Geometry)>
<!ELEMENT PieLabel (Group, Geometry)>
<!ELEMENT Slice (Group, Series, Tooltip?, Geometry)>
<!ELEMENT SliceLabel (Group, Series, Geometry)>
<!ELEMENT O1TickLabel (Group, Geometry)>
<!ELEMENT X1TickLabel (Geometry)>
<!ELEMENT O1Title (Geometry)>
<!ELEMENT X1Title (Geometry)>
<!ELEMENT Y1TickLabel (Geometry)>
<!ELEMENT Y1Title (Geometry)>
<!ELEMENT Y2TickLabel (Geometry)>
<!ELEMENT Y2Title (Geometry)>
<!ELEMENT ZTickLabel (Series, Geometry)>
<!ELEMENT ZTitle (Series, Geometry)>
<!ELEMENT Title (Geometry)>
<!ELEMENT Subtitle (Geometry)>
<!ELEMENT Footnote (Geometry)>
<!ELEMENT Group (#PCDATA)>
<!ELEMENT Series (#PCDATA)>
<!ELEMENT Tooltip (Line)*>
<!ELEMENT Line (#PCDATA)>
<!ELEMENT Geometry (Vertex)*>
<!ELEMENT Vertex EMPTY>
<!ATTLIST Vertex x CDATA #REQUIRED>
<!ATTLIST Vertex y CDATA #REQUIRED>
```

8

オンライン B2B XML アプリケーション : 手順

この章の内容は、次のとおりです。

- 手順 3.「Validate」によるトランザクションのコミット、リテーラ・アプリケーションによる XML 注文書の生成
- 手順 4. AQ Broker - トランスフォーマによるサプライヤの形式に従った XML 文書の変換
- 手順 5. サプライヤ・アプリケーションによる XML 文書の解析およびサプライヤ・データベースへの注文の挿入
- 手順 6a. サプライヤ・アプリケーションからサプライヤに対する注文保留を示すアラート
- 手順 7. AQ Broker - トランスフォーマによるリテーラの形式への XML 注文書の変換
- 手順 8. リテーラ・アプリケーションによる Ord 表の更新とリテーラに対する新規オーダー・ステータスの表示

- Java の例 ー コール順序
- XSL および XSL 管理スクリプト
- XML プロセスおよび管理スクリプト
- B2B XML アプリケーションで使用されるその他のスクリプト
- リテーラ・スクリプト
- AQ Broker - トランスフォーマおよびアドバンスト・キューイングのスクリプト
- サプライヤ・スクリプト

オンライン B2B XML アプリケーションの概要

この章では、オンライン B2B XML アプリケーションの構築に必要なすべての手順とスクリプトについて説明します。

このアプリケーションの変更済みバージョンは、Oracle Technology Network（OTN）の URL、http://otn.oracle.com/tech/xml の「WebStore B2B Demo」で入手できます。この変更済みバージョンでは、複数のサプライヤおよびリテーラのサポートが追加されており、この章で説明するよりも大規模な B2B オンライン・データ交換用に拡張できます。また、この OTN サイトからはスクリプトもダウンロードできます。

オンライン B2B XML アプリケーションの実行要件

ここでは、オンライン B2B XML アプリケーションを構築して実行するための要件について説明します。

- クライアント
  - オペレーティング・システム : Windows NT。このアプリケーションで使用される 3 つの .bat ファイルは Windows 固有です。ただし、UNIX システム用などに、シェル・スクリプトでリライトすることもできます。
  - ツール : JDeveloper 3.1 以上、208MB。
  - XML および XSL エディタ : 任意のエディタ。テキスト・エディタも使用できます。
  - ブラウザ : IE5.0 以上、Netscape 5 以上および HandWeb のような PDA ブラウザなど。
- 中間層
  - 開発環境には 513KB が必要です。
  - ランタイム環境の場合は 135KB のみ必要です。
  - XML Parser for Java および XSU for Java を含む XSQL Servlet。
  - HTTP リスナー。
- サーバー
  - Oracle8*i* リリース 8.1.7 以上など、Oracle および Java 対応サーバー。

オンライン B2B XML アプリケーションの構築 : 概要

この XML アプリケーションおよびデモは、コンテンツの管理と B2B メッセージ機能の実装を示しています。このアプリケーションでの主なトランザクションは、次のとおりです。

- リテーラ（R）が、ブラウザ、携帯電話または PDA（携帯情報端末）などのデバイスから発注します。
- サプライヤ（S）が受注を示すアラートを受け取ります。在庫の確認とリテーラの信用照会の後に、サプライヤが「Ship」ボタンをクリックします。
- リテーラとサプライヤは、この注文の出荷状態をどのデバイスからでも確認できます。

問題

リテーラ（R）は、複数のサプライヤ（サプライヤ（S））に対する商品の発注を自動化し、どのデバイスからも発注でき、そのオーダー・ステータスを表示できるようにする必要があります。

解決策

この解決策では、次を実装します。

- **Oracle XML コンポーネント**。リテーラの Web サイトから受信した HTML 形式（または他の形式）の発注データを、XML 文書に変換します。
- **Oracle8*i* 以上の データベース**。この解決策では、リテーラとサプライヤがそれぞれデータを Oracle8*i* 以上のデータベースに格納している場合を想定します。
- **AQ Broker ― トランスフォーマ**。この AQ アプリケーションでは、リテーラとサプライヤ間の注文の流れを管理します。リテーラは注文を AQ キューに送ります。関心のあるサプライヤは、キューから注文を取り出します（読み込む、つまりデキューします）。AQ は、注文の流れに関するインテリジェンスの抽出にも使用されます。各注文書は XML メッセージです。このメッセージは、リテーラとサプライヤの両方が認識できる形式に変換されます。

識別される作業

図 8-2 に主要な作業を示します。

1. リテーラが、ブラウザ、携帯情報端末（PDA）または携帯電話から注文を入力します。
2. リテーラが確認すると、その注文は XSQL Servlet を使用して XML に変換されます。
3. リテーラのアプリケーションから AQ Broker に XML 形式の注文書が送信されます。
4. XML 形式の注文書データの送信には、AQ メッセージ機能が使用されます。リテーラには、オーダー・ステータスが「Pending」として表示されます。AQ Broker により、XML 形式の注文書がサプライヤ側で認識できる形式に変換されます。

5. サプライヤのアプリケーションにより、注文がサプライヤ・データベースに挿入されます。
6. サプライヤのアプリケーションにより注文が解析され、注文を受信済みで処理を待機中であることを示すアラートがサプライヤに送信されます。
7. サプライヤが画面上で「Shipped」をクリックすると、AQ メッセージ機能を使用して XML 形式のオーダー・ステータス・データが AQ Broker に戻されます。戻された XML オーダー・ステータスは、AQ Broker によりリテーラ側で認識できる形式に変換されます。
8. サプライヤが再フォーマットされた XML 形式のオーダー・ステータス・メッセージを受信します。リテーラのアプリケーションにより、リテーラ・データベースが新規オーダー・ステータスで更新されます。リテーラがオーダー・ステータスを確認すると「Shipped」と表示されます。

関連する作業の詳細、表示される画面および使用するスクリプトは、「B2B XML アプリケーションの実行 : 手順の詳細」および図 8-2「オンライン B2B XML アプリケーション : 主要コンポーネント」を参照してください。

使用する XML および Oracle コンポーネント

- XSQL Servlet。XML-SQL Utility（XSU）、XML Parser for Java バージョン 2 および XSLT Processor などです。
- Oracle8*i* リリース 8.1.7 以上または Oracle9*i*。

使用するツール

JDeveloper

> **注意 :** この B2B XML アプリケーションでは、事前作成済みの（静的または構成済み）XML 文書は使用しません。このアプリケーションのすべての XML 文書は、データベースから動的に生成（分解）されます。

データを XML に変換する理由

リテーラとサプライヤは様々な形式を使用しています。

リテーラは様々な注文フォーム形式を使用しているため、その注文データはサプライヤが認識して処理できるように XML に変換されます。

サプライヤは、オーダー・ステータスと受信確認データに異なる形式を使用しています。このデータは、リテーラがオーダー・ステータスと受信確認を認識できるように XML に変換されます。

注意： この解決策では、2 つの事前決定済みの顧客注文ドキュメント形式による限定的なセットを使用します。

- **リテーラの注文データ**。注文データはリテーラ・データベース R に格納され、AQ Broker により XSL スタイルシートを使用して、特定のサプライヤが認識できる形式に変換されます。
- **サプライヤのオーダー・ステータス・データ**。このデータは、AQ Broker により XSL スタイルシートを使用して、特定のリテーラが認識できる形式に変換されます。

注意： トランスフォーマ API および関連する表は、リテーラ・データベースまたはサプライヤ・データベースなど、どこでも存在が可能です。

図 8-1 に、リテーラ / サプライヤ間のトランザクション・フロー全体を示します。リテーラが注文を入力します。

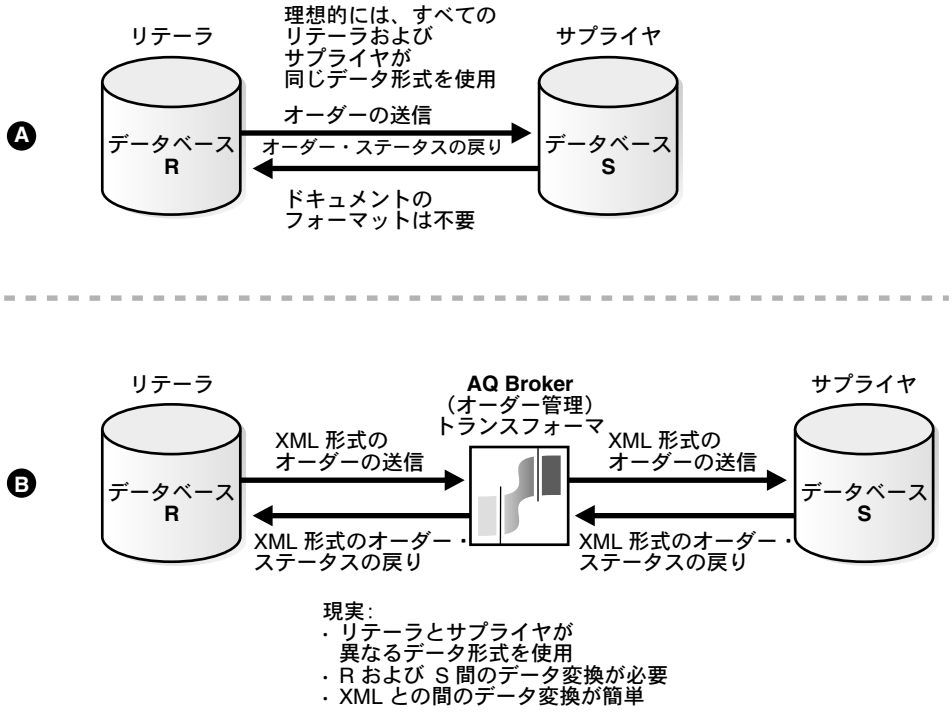
- 理想的には、すべてのリテーラおよびサプライヤが同じ注文書ドキュメント形式を使用していれば、プロセスは単に A のようになります。
- 現実には、通常、リテーラおよびサプライヤは、それぞれ異なる注文書ドキュメント形式を使用しています。データを XML に変換することで、このようなトランザクションをできるだけシームレスにします。XSL スタイルシートを適用すると、データ形式をカスタマイズし、どんなデバイスでも複数の形式で表示できます。

アドバンスト・キューイング（AQ）を使用する理由

このアプリケーションで AQ を使用すると、次のメリットが得られます。

- AQ により、リテーラからサプライヤへの注文の流れとオーダー・ステータス更新、およびサプライヤからリテーラへの受信確認の流れが管理されます。
- AQ により、リテーラがサプライヤから分離されるため、どのリテーラも注文を同じキューに入れ、どのサプライヤもそのキューから簡単に注文を取り出すことができます。つまり、多対多の使用例の単純な実装が容易になります。
- また、AQ では、処理対象となる注文に関するインテリジェンスが抽出されます。

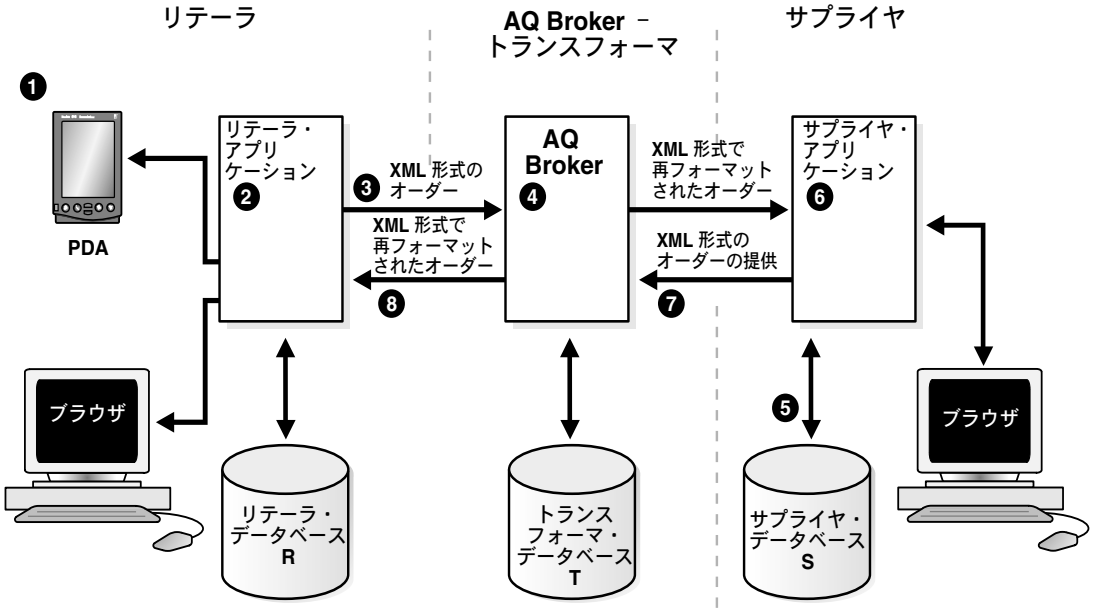
図 8-1 データを XML に変換する理由：リテーラの注文データをすべてのサプライヤが認識可能 ー サプライヤのオーダー・ステータスと受信確認をすべてのリテーラが認識可能



オンライン B2B XML アプリケーション : 主要コンポーネント

図 8-2 に、このオンライン B2B XML アプリケーションに使用される主要コンポーネントを示します。リテーラはサプライヤの商品を発注し、サプライヤから商品が出荷されたことを示す確認を受信します。図 8-5 に、このプロセスの詳細なトランザクション・ダイアグラムを示します。

図 8-2 オンライン B2B XML アプリケーション : 主要コンポーネント



オンライン B2B XML アプリケーションを実行するための作業の概要

図 8-3 に、構築する B2B XML アプリケーションで使用されるスキーマを示します。

B2B XML アプリケーションを実行するには、次のタスクを実行します。

- 作業 1. オンライン B2B XML アプリケーションの実行環境のセットアップ
- 作業 2. B2B アプリケーションの実行
- 作業 3. B2B アプリケーション・セッションの終了

ブラウザに表示される内容など、B2B XML アプリケーションの実行の詳細は、8-35 ページの「B2B XML アプリケーションの実行 : 手順の詳細」を参照してください。また、リテーラ側とサプライヤ側に表示される典型的なスクリーンショットも掲載されています。

図 8-3 B2B XML のリテーラ（Customers）およびサプライヤ（Supplier）スキーマ

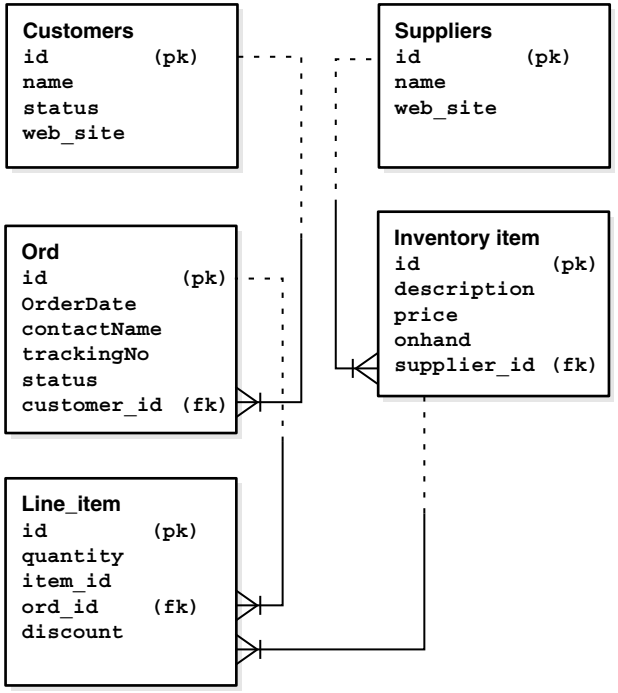
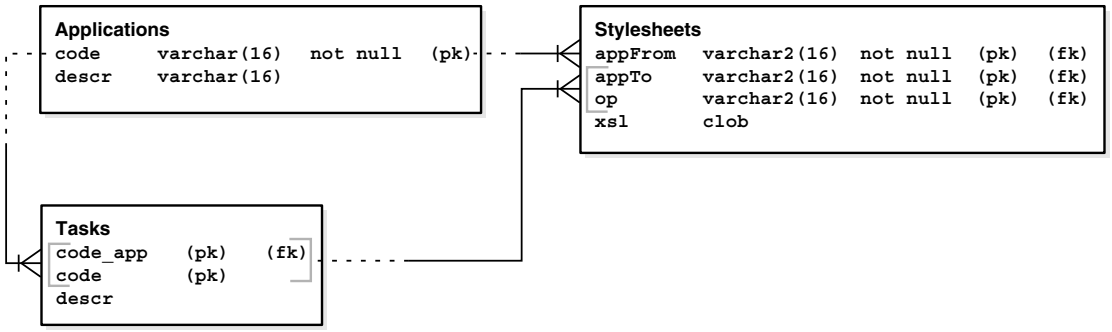


図 8-4 B2B XML AQ Broker スキーマ : Stylesheets



作業 1. オンライン B2B XML アプリケーションの実行環境のセットアップ

1. Apache または他の Web サーバーを起動します。
2. IE5 などのブラウザを起動します。
3. ログインします。
4. B2B XML アプリケーションの実行に必要なスキーマをすべて設定するには、次の手順で操作します。

   Retailer および Supplier スキーマを作成します。「オンライン B2B XML アプリケーション : 主要コンポーネント」を参照してください。

   - 必要な方法でデータベースに接続します。
   - **buildAll.sql** を実行します。このスクリプトでは、要求したユーザーを作成するために、システム・パスワードの入力を求めるプロンプトが表示されます。

5. AQ スキーマを作成します。
   - 目的に合ったマシンで SQL スクリプト **mkAQUser.sql** を実行します。
   - aqMessBrok/aqMessBrok で接続し、スクリプト **mkQ.sql** を実行します。

6. XSL の表を作成します。
   - 接続したままでスクリプト **mkSSTables.sql** を実行します。
   - **setup.sql** を実行して XSL スタイルシートをデータベースにインストールします。
   - 次の手順に従って接続を変更した後に、GUIStylesheet java クラスを実行してテストします。
7. 接続を変更します。
   - 次のファイル内で JDBC 接続パラメータを変更します。
     * AppCste.java
     * retail.bat
     * supplier.bat
     * PlaceOrder.xsql
   - 最後に、XSQLConfig.xml を変更して、retailer/retailer 上に接続 retail を作成します。
   - ステップ 8 に進む前に、すべてのファイルを再コンパイルします。
8. B2B XML アプリケーションを実行する前に、スクリプト **reset.sql** を実行して AQ 環境をリセットします。
9. Broker、Supplier および Retailer の 3 つの bat ファイルを変更して実行します。
   - .bat ファイルを変更します。3 つの main が使用されており、それぞれ次の .bat ファイルから起動されます。
     * message broker 用の Broker.bat
     * supplier 用の Supplier.bat
     * retailer 用の Retail.bat

     最初に、環境に合った .bat ファイルを次のように変更します。
     * **verbose**。「true」に設定すると、受信したメッセージに関する詳細が多数表示されます。
     * **step**。true に設定すると、処理ステップが終わるたびに [Enter] キーを押すように要求されます。step に数値を指定すると、先に進む前にステップ間で待機するミリ秒数とみなされます。

     `Retail.bat` および `Supplier.bat` には、-dbURL パラメータも使用できます。このパラメータには、問題のデータベースへの接続に使用する URL を記述します。デフォルトの URL は、jdbc:oracle:thin:@localhost:1521:ORCL です。

作業 2. B2B アプリケーションの実行

1. **broker.bat**、**supplier.bat** および **retailer.bat** を実行します。
2. **GUIStylesheet.class** を実行して StyleSheet ユーティリティをチェックします。

これらのスタイルシートは、ブローカで受信したドキュメントの処理に使用されます。

ブラウザに表示される内容など、B2B XML アプリケーションの実行の詳細は、「B2B XML アプリケーションの実行 : 手順の詳細」を参照してください。

作業 3. B2B アプリケーション・セッションの終了

1. B2B XML アプリケーションを終了します。

   Java クラス **stopAllQueues** またはスクリプト **stopQ.bat** を実行します。
2. Apache または Web サーバーを停止します。

オンライン B2B XML アプリケーション : データベース・スキーマの設定

次に示すスキーマ・スクリプトを順番どおりに実行する必要があります。

- Retailer スキーマと Supplier スキーマの作成および構築
  - SQL の例 1: Retailer および Supplier 環境のセットアップ ― BuildAll.sql
  - これにより、SQL の例 2: Retailer および Supplier スキーマの作成と移入 ― BuildSchema.sql がコールされます。
- AQ 環境およびキュー表の作成
  - SQL の例 3: AQ 用の環境のセットアップ ― mkAQUser.sql
  - SQL の例 4: AQ キュー作成スクリプトのコール ― mkQ.sql. これにより、次のスクリプトがコールされます。
    - SQL（PL/SQL）の例 5: 表 AppOne_QTab の作成 ― mkQueueTableApp1.sql
    - SQL（PL/SQL）の例 6: 表 AppTwo_QTab の作成 ― mkQueueTableApp2.sql
    - SQL（PL/SQL）の例 7: 表 AppThree_QTab の作成 ― mkQueueTableApp3.sql
    - SQL（PL/SQL）の例 8: 表 AppFour_QTab の作成 ― mkQueueTableApp4.sql
- XSL スタイルシート表を含む Broker スキーマの作成
  - SQL の例 9: Broker スキーマの作成 ― mkSSTables.sql.

これにより、次のスクリプトがコールされます。

- SQL（PL/SQL）の例 10: CLOB への XSL データの入力、Broker スキーマへの移入 ― setup.sql

- 環境のクリーン・アップとアプリケーションを再実行する準備
  - SQL の例 11: キュー・アプリケーションの停止と削除、キュー・アプリケーションの起動 ― reset.sql

SQL コードのコール順序

SQL サンプル・コードのコール順序を次に示します。各ファイルの .sql 拡張子は省略されています。表記「<---」は「コール」を意味します。たとえば、BuildAll.sql <----- BuildSchema は、BuildAll.sql により BuildSchema がコールされることを示します。

- BuildAll.sql <---- BuildSchema.sql
- mkAQuser.sql
- mkQ.sql
  - <---- mkQueueTableApp1
  - <---- mkQueueTableApp2
  - <---- mkQueueTableApp3
  - <---- mkQueueTableApp4
- mkSSTables.sql <---- setup.sql
- reset.sql
  - <---- stopQueueApp1
  - <---- stopQueueApp2
  - <---- stopQueueApp3
  - <---- stopQueueApp4
  - <---- dropQueueApp1
  - <---- dropQueueApp2
  - <---- dropQueueApp3
  - <---- dropQueueApp4
  - <---- createQueueApp1
  - <---- createQueueApp2
  - <---- createQueueApp3

- <---- createQueueApp4
- <---- startQueueApp1
- <---- startQueueApp2
- <---- startQueueApp3
- <---- startQueueApp4

Retailer スキーマと Supplier スキーマの作成および構築

次のスキーマ・スクリプトにより、Retailer および Supplier 環境、ユーザー、表領域、割当て制限などがセットアップされます。また、スキーマも作成され、移入されます。

- SQL の例 1: Retailer および Supplier 環境のセットアップ ― BuildAll.sql. これにより、次のスクリプトがコールされます。
- SQL の例 2: Retailer および Supplier スキーマの作成と移入 ― BuildSchema.sql

SQL の例 1: Retailer および Supplier 環境のセットアップ ― BuildAll.sql

`BuildAll.sql` により、Retailer および Supplier スキーマの環境がセットアップされます。`BuildSchema.sql` がコールされ、Retailer および Supplier スキーマが作成されてから、各スキーマにデータが移入されます。

```
--
-- buildall.sql builds all the schemas
--
accept sysPswd prompt 'Enter the system password
> ' hide
accept cStr    prompt 'Enter the connect string if any, including ''@'' sign (ie
@atp-1) > '
connect system/&sysPswd&cStr
drop user retailer cascade
/
drop user supplier cascade
/
col tablespace_name head "Available Tablespaces"
select tablespace_name from dba_tablespaces
/
prompt
accept userTbsp prompt 'What is the DEFAULT Tablespace name ?   > '
accept tempTbsp prompt 'What is the TEMPORARY Tablespace name ? > '
prompt

create user retailer identified by retailer
default tablespace &userTbsp
```

```
temporary tablespace &tempTbsp
quota unlimited on &userTbsp
/
grant connect, resource, create any directory to retailer
/
create user supplier identified by supplier
default tablespace &userTbsp
temporary tablespace &tempTbsp
quota unlimited on &userTbsp
/
grant connect, resource, create any directory to supplier
/
prompt Now populating Supplier, hit [Return]
pause
connect supplier/supplier&cStr
@buildSchema
prompt Now populating Retailer, hit [Return]
pause
connect retailer/retailer&cStr
@buildSchema
prompt done !
```

SQL の例 2: Retailer および Supplier スキーマの作成と移入 — BuildSchema.sql

`BuildSchema.sql` は `BuildAll.sql` からコールされます。このスクリプトにより、Retailer スキーマと Supplier スキーマが作成、移入および構築されます。

このスクリプトでは、次の 5 つの表が作成され、移入されます。

- Customers
- Suppliers
- Inventory_item
- Ord
- Line_item

このスキーマの内容は、図 8-3 を参照してください。

```
--
-- buildSchema.sql drops then creates all the tables for the B2B XML Application
--
drop trigger line_item_insert_trigger;
drop table line_item;
drop table ord;
drop table customer;
drop table inventory_item;
```

```
drop table supplier;
drop sequence ord_seq;
drop sequence customer_seq;
drop sequence line_item_seq;
drop sequence supplier_seq;
drop sequence inventory_item_seq;

prompt
prompt Creating sequences...
prompt
prompt
prompt Creating sequence ORD_SEQ
create sequence ord_seq start with 101;

prompt Creating sequence CUSTOMER_SEQ
create sequence customer_seq start with 201;

prompt Creating sequence LINE_ITEM_SEQ
create sequence line_item_seq start with 1001;

prompt Creating sequence SUPPLIER_SEQ
create sequence supplier_seq start with 301;

prompt Creating sequence INVENTORY_ITEM_SEQ
create sequence inventory_item_seq start with 401;

prompt
prompt
prompt Creating tables...
prompt
prompt
--
-- ***** Create table CUSTOMERS ******
--
prompt Creating table CUSTOMER
create table customer(
  id         number,
  name       varchar2(30),
  status     varchar2(8),
  web_site   varchar2(40),
  constraint
     customer_pk
     primary key (id)
);
--
-- ***** Create table SUPPLIERS ******
--
```

```
prompt Creating table SUPPLIER
create table supplier(
  id         number,
  name       varchar2(30),
  web_site   varchar2(40),
  constraint
     supplier_pk
     primary key (id)
);
--
-- ***** Create table INVENTORY_ITEM ******
--
prompt Creating table INVENTORY_ITEM
create table inventory_item(
  id             number,
  description    varchar2(30),
  price          number(8,2),
  onhand         number,
  supplier_id    number,
  constraint
     inventory_item_pk
     primary key (id),
  constraint
     supplied_by
     foreign key (supplier_id) references supplier
);
--
-- ***** Create table ORD ******
--
prompt Creating table ORD
create table ord (
  id             number,
  orderDate      date,
  contactName    varchar2(30),
  trackingNo     varchar2(20),
  status         varchar2(10),
  customer_id    number,
  constraint
     ord_pk
     primary key (id),
  constraint
     order_placed_by
     foreign key (customer_id) references customer
);

prompt Creating table LINE_ITEM
create table line_item(
```

```
  id             number,
  quantity       number,
  item_id        number,
  ord_id         number,
  discount       number,
  constraint
     line_item_pk
     primary key (id),
  constraint
     item_ordered_on
     foreign key (ord_id) references ord,
  constraint
     order_for_item
     foreign key (item_id) references inventory_item
);

prompt
prompt
prompt Inserting data...
prompt
prompt
prompt Inserting values into SUPPLIER and INVENTORY_ITEM
prompt

insert into supplier values( supplier_seq.nextval,'DELL','http://dell.com');
insert into inventory_item values( inventory_item_seq.nextval,'Optiplex GXPro',
1500, 27, supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval,'Inspiron 7000', 2500,
49, supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval,'PowerEdge 6300',
7500, 16, supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval,'Inspiron 3000', 2500,
0, supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval,'Inspiron 2000', 2500,
0, supplier_seq.currval );

insert into supplier values( supplier_seq.nextval, 'HP', 'http://hp.com');
insert into inventory_item values( inventory_item_seq.nextval, 'LaserJet 6MP', 899,
123, supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval, 'Jornada 2000', 450,
1198, supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval, 'HP 12C', 69, 801,
supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval, 'LaserJet 2', 69, 3,
supplier_seq.currval );
```

```
insert into inventory_item values( inventory_item_seq.nextval,'Jaz PCMCIA adapter',
125, 54, supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval,'8860 Digital phone',
499, 12, supplier_seq.currval );
insert into inventory_item values( inventory_item_seq.nextval,'Jaz carrying bag',
20, 66, supplier_seq.currval );

insert into supplier values(supplier_seq.nextval,'Intel', 'http://www.intel.com');
prompt Inserting values into CUSTOMER
prompt

insert into ord values(ord_seq.nextval,sysdate,'George','AX'||ord_seq.currval,
'Pending', 201);
insert into line_item values (line_item_seq.nextval, 2, 410,ord_seq.currval, 0);
insert into line_item values (line_item_seq.nextval, 1, 402,ord_seq.currval, 0);
insert into line_item values (line_item_seq.nextval, 1, 406,ord_seq.currval, 0);

insert into ord values(ord_seq.nextval,sysdate,'Elaine','AX'||ord_seq.currval,
 'BackOrdered', 0);
create trigger line_item_insert_trigger
  before insert on line_item for each row
  begin
     select line_item_seq.nextval into :new.id from dual ;
  end;
/

commit;
```

AQ 環境およびキュー表の作成

AQ スキーマ・スクリプトを次のように実行します。

- SQL の例 3: AQ 用の環境のセットアップ ― mkAQUser.sql
- SQL の例 4: AQ キュー作成スクリプトのコール ― mkQ.sql. これにより、次のスクリプトがコールされます。
  - SQL（PL/SQL）の例 5: 表 AppOne_QTab の作成 ― mkQueueTableApp1.sql
  - SQL（PL/SQL）の例 6: 表 AppTwo_QTab の作成 ― mkQueueTableApp2.sql
  - SQL（PL/SQL）の例 7: 表 AppThree_QTab の作成 ― mkQueueTableApp3.sql
  - SQL（PL/SQL）の例 8: 表 AppFour_QTab の作成 ― mkQueueTableApp4.sql

SQL の例 3: AQ 用の環境のセットアップ ー mkAQUser.sql

次の SQL スクリプトにより、AQ を使用できるように環境がセットアップされ、ユーザー aqMessBrok が作成され、デフォルト表領域と一時表領域が作成され、AQ の PL/SQL パッケージ dbms_aqadm および dbms_aq to aqMessBrok の実行権限が付与されます。

```
set ver off
set scan on
prompt Creating environment for Advanced Queuing
accept mgrPsw prompt 'Please enter the SYSTEM password
> ' hide
accept cStr   prompt 'Please enter the the DB Alias if any, WITH the @ sign (ie
@Ora8i)> '
connect system/&mgrPsw&cStr

col tablespace_name head "Available Tablespaces"
select tablespace_name from dba_tablespaces
/
Prompt
accept userTbsp prompt 'What is the DEFAULT Tablespace name ?   > '
accept tempTbsp prompt 'What is the TEMPORARY Tablespace name ? > '
prompt

prompt Creating aqMessBrok
create user aqMessBrok identified by aqMessBrok
default tablespace &userTbsp
temporary tablespace &tempTbsp
quota unlimited on &userTbsp
/
grant connect, resource, aq_administrator_role, create any directory to aqMessBrok
/
grant execute on dbms_aqadm to aqMessBrok
/
grant execute on dbms_aq to aqMessBrok
/
```

SQL の例 4: AQ キュー作成スクリプトのコール ー mkQ.sql

このスクリプトにより、次の 4 つのスクリプトがコールされ、AQ のキュー表が作成されます。

```
@mkQueueTableApp1
@mkQueueTableApp2
@mkQueueTableApp3
@mkQueueTableApp4
```

SQL（PL/SQL）の例 5: 表 AppOne_QTab の作成 ー mkQueueTableApp1.sql

このスクリプトは mkQ.sql からコールされ、dbms_aqadm.create_queue_table プロシージャをコールして、キュー表 1 の AppOne_QTab を作成します。

```
execute dbms_aqadm.create_queue_table (queue_table => 'AppOne_QTab', queue_payload_
type => 'RAW');
```

SQL（PL/SQL）の例 6: 表 AppTwo_QTab の作成 ー mkQueueTableApp2.sql

このスクリプトは mkQ.sql からコールされ、dbms_aqadm.create_queue_table プロシージャをコールして、キュー表 2 の AppTwo_QTab を作成します。

```
execute dbms_aqadm.create_queue_table (queue_table => 'AppTwo_QTab', queue_payload_
type => 'RAW');
```

SQL（PL/SQL）の例 7: 表 AppThree_QTab の作成 ー mkQueueTableApp3.sql

このスクリプトは mkQ.sql からコールされ、dbms_aqadm.create_queue_table プロシージャをコールして、キュー表 3 の AppThree_QTab を作成します。

```
execute dbms_aqadm.create_queue_table (queue_table => 'AppThree_QTab', queue_
payload_type => 'RAW');
```

SQL（PL/SQL）の例 8: 表 AppFour_QTab の作成 ー mkQueueTableApp4.sql

このスクリプトは mkQ.sql からコールされ、dbms_aqadm.create_queue_table プロシージャをコールして、キュー表 4 の AppFour_QTab を作成します。

```
execute dbms_aqadm.create_queue_table (queue_table => 'AppFour_QTab', queue_payload_
type => 'RAW');
```

XSL スタイルシート表を含む Broker スキーマの作成

次のスクリプトを実行して、スタイルシート、タスクおよびアプリケーションの表を作成し、移入します。

- SQL の例 9: Broker スキーマの作成 ー mkSSTables.sql.

  これにより、次のスクリプトがコールされます。

- SQL（PL/SQL）の例 10: CLOB への XSL データの入力、Broker スキーマへの移入 ー setup.sql

SQL の例 9: Broker スキーマの作成 ー mkSSTables.sql

`mkSSTables.sql` を実行して Broker スキーマを作成します。このスクリプトにより、次の 3 つの表が作成され、移入されます。

- Stylesheets
- Tasks
- Applications

このスキーマの内容は、図 8-4 を参照してください。このスクリプトにより、`setup.sql` がコールされます。

```
prompt Building Stylesheets management tables.
prompt Must be connected as aqMessBrok (like the borker)
accept cStr prompt 'ConnectString (WITH @ sign, like @Ora8i) > '
connect aqMessBrok/aqMessBrok&cStr

drop table styleSheets
/
drop table tasks
/

drop table applications
/
create table applications
(
  code  varchar2(16) not null,
  descr varchar2(256)
)
/
alter table applications
  add constraint PK_APP
  primary key (code)
/
create table tasks
(
```

```
  code_app varchar2(16) not null,
  code     varchar2(16) not null,
  descr    varchar2(256)
)
/
alter table tasks
  add constraint PK_TASKS
  primary key (code_app,code)
/
alter table tasks
  add constraint TASK_FK_APP
  foreign key (code_app)
  references applications(code) on delete cascade
/
create table styleSheets
(
  appFrom varchar2(16) not null,
  appTo   varchar2(16) not null,
  op      varchar2(16) not null,
  xsl     clob
)
/
alter table styleSheets
  add constraint PK_SS
  primary key (appFrom,appTo,op)
/
alter table styleSheets
  add constraint SS_FK_FROM
  foreign key (appFrom)
  references applications(code)
/
alter table styleSheets
  add constraints SS_FK_TASK
  foreign key (appTo,op)
  references tasks(code_app,code)
/
@setup
```

SQL（PL/SQL）の例 10: CLOB への XSL データの入力、Broker スキーマへの移入 ー setup.sql

setup.sql により、スタイルシート・データがスタイルシート表の XSL 列（CLOB）にインストールされます。このスクリプトにより、プロシージャ loadlob が作成されます。また、PL/SQL パッケージ dbms_lob および dbms_output も使用されます。

```
prompt Installing the stylesheets
-- accept cStr prompt 'ConnectString (WITH @ sign, like @Ora8i) > '
-- connect aqMessBrok/aqMessBrok&cStr
prompt Creating LoadLob procedure
create or replace procedure loadLob (imgDir in varchar2,
                                     fname in varchar2,
                                     app_From in varchar2,
                                     app_To in varchar2,
                                     oper in varchar2) as
  tempClob  CLOB;
  fileOnOS  BFILE := bfilename(imgDir, fname);
  ignore    INTEGER;
begin
  dbms_lob.fileopen(fileOnOS, dbms_lob.file_readonly);
  select xsl
  into tempClob
  from StyleSheets S
  where s.APPFROM = app_From and
        s.APPTO = app_To and
        s.OP = oper
  for UPDATE;
dbms_output.put_line('External file size is: ' || dbms_lob.getlength(fileOnOS));
dbms_lob.loadfromfile(tempClob, fileOnOS, dbms_lob.getlength(fileOnOS));
dbms_lob.fileclose(fileOnOS);
dbms_output.put_line('Internal CLOB size is: '|| dbms_lob.getlength(tempClob));
exception
  When Others then
    dbms_output.put_line('Oooops : ' || SQLERRM);
end LoadLob;
/
show errors
set scan off

create or replace directory "LOB_DIR" as 'D:¥xml817¥references¥olivier_new'
/
insert into applications values ('RETAIL', 'Origin')
/
insert into applications values ('SUPPLY', 'Destination')
/
insert into tasks values ('SUPPLY', 'NEW ORDER', 'Insert a new Order')
```

```
/
insert into tasks values ('RETAIL', 'UPDATE ORDER', 'Update an Order Status')
/

set serveroutput on
begin
 insert into StyleSheets values ('RETAIL','SUPPLY','NEW ORDER',EMPTY_CLOB());
 loadLob('LOB_DIR', 'one.xsl', 'RETAIL','SUPPLY','NEW ORDER');
 insert into StyleSheets values ('SUPPLY','RETAIL','UPDATE ORDER',EMPTY_CLOB());
 loadLob('LOB_DIR', 'two.xsl', 'SUPPLY','RETAIL','UPDATE ORDER');
exception
  when others then
  dbms_output.put_line('Error Occurred : ' || chr(10) || SQLERRM);
end;
/
commit
/
```

環境のクリーン・アップとアプリケーションを再実行する準備

`reset.sql` を実行して環境をクリーン・アップし、このアプリケーションを再実行します。

- SQL の例 11: キュー・アプリケーションの停止と削除、キュー・アプリケーションの起動 ― reset.sql

  このスクリプトにより、次の 16 の PL/SQL スクリプトがコールされます。

  - stopQueueApp1.sql
  - stopQueueApp2.sql
  - stopQueueApp3.sql
  - stopQueueApp4.sql
  - dropQueueApp1.sql
  - dropQueueApp2.sql
  - dropQueueApp3.sql
  - dropQueueApp4.sql
  - createQueueApp1.sql
  - createQueueApp2.sql
  - createQueueApp3.sql
  - createQueueApp4.sql

- startQueueApp1.sql
- startQueueApp2.sql
- startQueueApp3.sql
- startQueueApp4.sql

SQL の例 11: キュー・アプリケーションの停止と削除、キュー・アプリケーションの起動 ー reset.sql

reset.sql スクリプトでは、最初に stopQueueApp1 ～ 4 のコールにより 4 つのキュー・アプリケーションがすべて停止されてから、dropQueueApp1 ～ 4 のコールにより削除され、startQueueApp1 ～ 4 のコールにより再起動されます。

また、[Enter] キーを押して終了するように求めるプロンプトが表示されます。

```
connect aqMessBrok/aqMessBrok
start stopQueueApp1
start stopQueueApp2
start stopQueueApp3
start stopQueueApp4
start dropQueueApp1
start dropQueueApp2
start dropQueueApp3
start dropQueueApp4
start createQueueApp1
start createQueueApp2
start createQueueApp3
start createQueueApp4
start startQueueApp1
start startQueueApp2
start startQueueApp3
start startQueueApp4
prompt Press [Return] to exit !
pause
exit
```

キュー停止 SQL スクリプト

次の 4 つのスクリプトは reset.sql からコールされます。各スクリプトでは、PL/SQL プロシージャ dbms_aqadm.stop_queue を使用してキューが停止されます。

stopQueueApp1.sql

```
execute dbms_aqadm.stop_queue(queue_name=>'AppOneMsgQueue');
```

stopQueueApp2.sql

```
execute dbms_aqadm.stop_queue(queue_name=>'AppTwoMsgQueue');
```

stopQueueApp3.sql

```
execute dbms_aqadm.stop_queue(queue_name=>'AppThreeMsgQueue');
```

stopQueueApp4.sql

```
execute dbms_aqadm.stop_queue(queue_name=>'AppFourMsgQueue');
```

キュー削除 SQL スクリプト

次の 4 つのスクリプトは reset.sql からコールされます。各スクリプトでは、PL/SQL プロシージャ dbms_aqadm.drop_queue を使用してキューが削除されます。

dropQueueApp1.sql

```
execute dbms_aqadm.drop_queue (queue_name=>'AppOneMsgQueue');
```

dropQueueApp2.sql

```
execute dbms_aqadm.drop_queue (queue_name=>'AppTwoMsgQueue');
```

dropQueueApp3.sql

```
execute dbms_aqadm.drop_queue (queue_name=>'AppThreeMsgQueue');
```

dropQueueApp4.sql

```
execute dbms_aqadm.drop_queue (queue_name=>'AppFourMsgQueue');
```

キュー作成 SQL スクリプト

次の 4 つのスクリプトは reset.sql からコールされます。各スクリプトでは、PL/SQL プロシージャ dbms_aqadm.create_queue を使用してキューが作成されます。

createQueueApp1.sql

```
execute dbms_aqadm.create_queue (queue_name=>'AppOneMsgQueue', queue_table=>'AppOne_
QTab');
```

createQueueApp2.sql

```
execute dbms_aqadm.create_queue (queue_name=>'AppTwoMsgQueue', queue_table=>'AppTwo_
QTab');
```

createQueueApp3.sql

```
execute dbms_aqadm.create_queue (queue_name=>'AppThreeMsgQueue', queue_
table=>'AppThree_QTab');
```

createQueueApp4.sql

```
execute dbms_aqadm.create_queue (queue_name=>'AppFourMsgQueue', queue_
table=>'AppFour_QTab');
```

キュー起動 SQL スクリプト

次の 4 つのスクリプトは reset.sql からコールされます。各スクリプトでは、PL/SQL プロシージャ dbms_aqadm.start_queue を使用してキューが起動されます。

startQueueApp1.sql

```
execute dbms_aqadm.start_queue(queue_name=>'AppOneMsgQueue');
```

startQueueApp2.sql

```
execute dbms_aqadm.start_queue (queue_name=>'AppTwoMsgQueue');
```

startQueueApp3.sql

```
execute dbms_aqadm.start_queue (queue_name=>'AppThreeMsgQueue');
```

startQueueApp4.sql

```
execute dbms_aqadm.start_queue (queue_name=>'AppFourMsgQueue');
```

dropOrder.sql

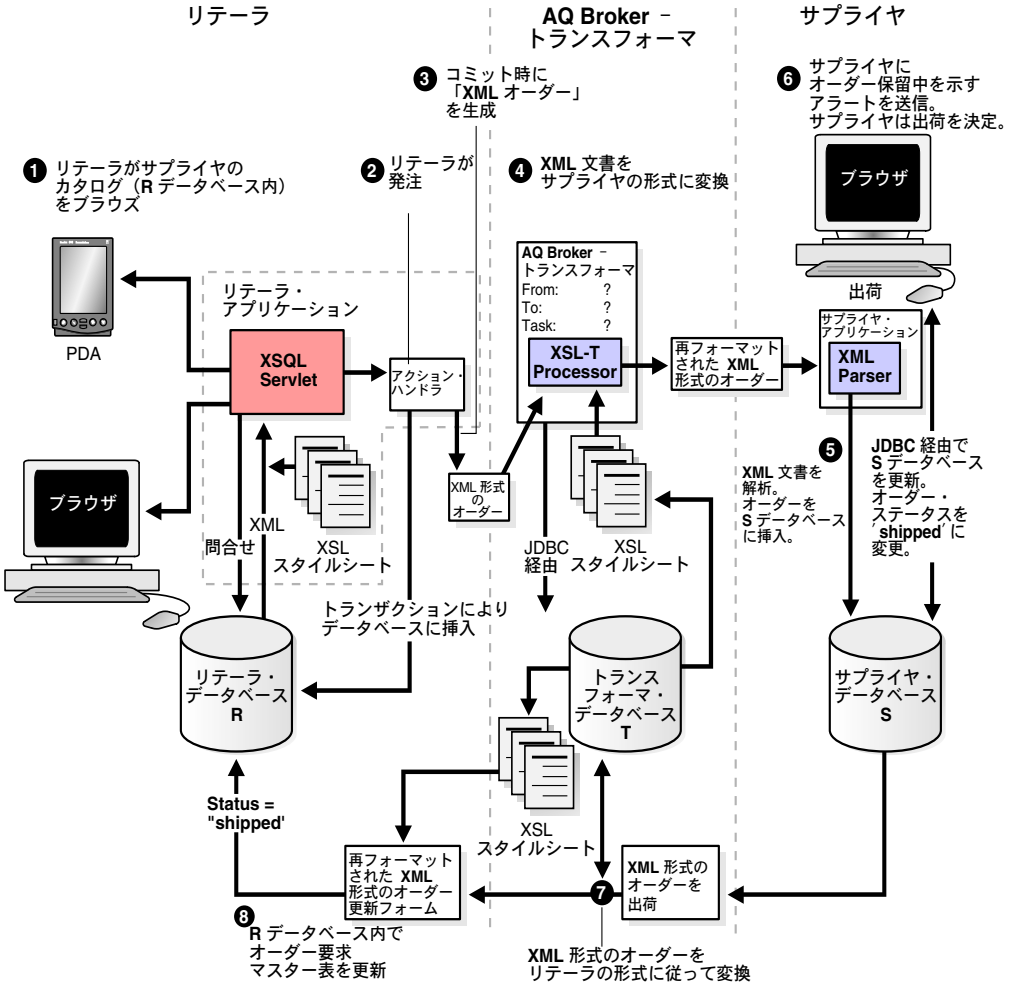
このSQLスクリプトでは、Retailer-Supplierデータベースの Customers 表から、顧客IDに従って注文が削除されます。

```
set ver off
accept CustName prompt 'Drop all for customer named > '

Delete LINE_ITEM I
Where I.ORD_ID in
(Select O.ID
 From ORD O
 Where O.CUSTOMER_ID in
 (Select C.ID
  From CUSTOMER C
  Where Upper(C.NAME) = Upper('&CustName')))
/
Delete ORD O
Where O.CUSTOMER_ID in
(Select C.ID
 From CUSTOMER C
 Where Upper(C.NAME) = Upper('&CustName'))
/
```

オンライン B2B XML アプリケーション : データ交換フロー

図 8-5 に、リテーラがサプライヤの商品を発注し、サプライヤから商品を出荷したという確認を受信する場合の、プロセスの詳細なトランザクション・ダイアグラムを示します。

図 8-5 ビジネス間データ交換 : XML および AQ を使用した、サプライヤに対するリテーラの注文送信と、サプライヤからのオーダー・ステータスと受信確認の受信



リテーラ / サプライヤ間のトランザクション

図 8-5 に、リテーラ / サプライヤ間のトランザクションのビジネス・フローを示します。これらのトランザクションの概要は、次のとおりです。

- 手順 1. リテーラによるサプライヤのオンライン「Hi-Tech Mall」カタログのブラウズ
- 手順 2. リテーラによる発注
- 手順 3. リテーラによる注文の確認と送信のためのコミット
- 手順 4. AQ Broker - トランスフォーマによるサプライヤの形式に従った XML 文書の変換
- 手順 5. サプライヤ・アプリケーションによる受信した再フォーマット済みの XML 注文書の解析およびサプライヤ・データベースへの注文の挿入
- 手順 6. サプライヤ・アプリケーションからサプライヤに対する注文保留を示すアラート
- 手順 7. AQ Broker - トランスフォーマによるリテーラの形式に従った XML 注文書の変換
- 手順 8. リテーラ・アプリケーションによる Ord および Line_Item 表の更新

トランザクションの詳細と B2B XML アプリケーションの実行方法は、8-35 ページの「B2B XML アプリケーションの実行 : 手順の詳細」を参照してください。

手順 1. リテーラによるサプライヤのオンライン「Hi-Tech Mall」カタログのブラウズ

次のリテーラ・トランザクションが発生します。

1. リテーラが XSQL を使用して Web サイトにログインします。
2. リテーラがサプライヤのオンライン・カタログをブラウズし、製品と数量を選択します。

手順 2. リテーラによる発注

リテーラは、発注時に「Place Order」をクリックして注文とコストを確認するか、「Give Up」をクリックして注文を取り消す必要があります。

手順 3. リテーラによる注文の確認と送信のためのコミット

リテーラが「Place Order」をクリックして注文を確認すると、注文データを含む XML 文書が生成されます。リテーラのアプリケーションでは、AQ のブローカ / トランスフォーマ・アプリケーションを使用して、この XML 形式の注文書がサプライヤに送信されます。

XSQL Servlet のアクション・ハンドラ「XSQL スクリプトの例 5: B2B プロセスの起動 ― placeorder.xsql」は、プロセス全体の主要コンポーネントです。これにより、トランザクションがリテーラ・データベース表 Ord に挿入されることが保証されます。

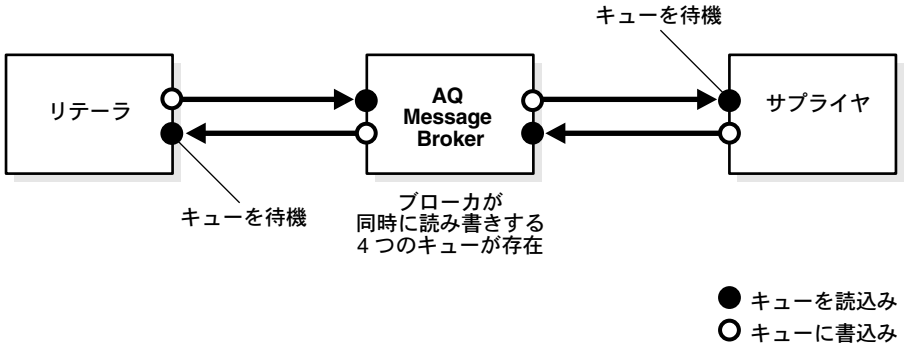
また、このアクション・ハンドラにより、XML 形式の注文が AQ Broker - トランスフォーマに送信されます。

手順 4. AQ Broker - トランスフォーマによるサプライヤの形式に従った XML 文書の変換

AQ Broker - トランスフォーマが XML 文書を受信すると、次のアクションが発生します。

1. AQ Broker - トランスフォーマは、リテーラからの注文書送信済みを示すキュー [READS] を待機します。図 8-6 を参照してください。

図 8-6 オンライン B2B XML アプリケーション : AQ メッセージ機能の流れ



2. AQ Broker は XML 文書による注文書メッセージを受信し、メッセージから次の情報を判断します。
   - FROM。メッセージの発信元（リテーラ）
   - TO。メッセージの送信先（サプライヤ）
   - OPERATION または TASK。このメッセージの処理に必要な操作

3. AQ Broker - トランスフォーマは Stylesheets 表を参照し、From、To および Task 条件に従って適切な XSL スタイルシートを選択します。スタイルシートは、Stylesheets 表の XSL 列の CLOB に格納されます。AQ Broker - トランスフォーマは、JDBC を使用してデータベースおよびスタイルシートにアクセスします。
4. XSLT Processor は、AQ Broker - トランスフォーマ・アプリケーションから通知を受け取って、選択され取り出された XSL スタイルシートを、注文データを含む XML 文書に適用します。XSLT Processor は、再フォーマットした XML 形式の注文書を出力します。
5. AQ Broker - トランスフォーマは、AQ を使用して、変換済みの XML 文書を「TO」宛先サプライヤに送信 [WRITE] します。

> **注意：** DTD（XML Schema）を使用すると、AQ Broker フェーズでの処理の前に適用されます。この例では、単純化するために、ドキュメントは常に同じ形式で送信されるものと想定しています。

AQ Broker - トランスフォーマで使用されるスキーマは、図 8-4 を参照してください。

手順 5. サプライヤ・アプリケーションによる受信した再フォーマット済みの XML 注文書の解析およびサプライヤ・データベースへの注文の挿入

サプライヤが AQ Broker - トランスフォーマから再フォーマット済みの XML 注文書を受信すると、次のプロトコルが発生します。

1. サプライヤは、リテーラからの注文が保留中であることを示す、AQ Broker - トランスフォーマからのキューを待機します。サプライヤは、この AQ メッセージをデキューします。
2. サプライヤは XML 文書を解析し、JDBC を使用して注文をサプライヤ・データベースに INSERT します。

手順 6. サプライヤ・アプリケーションからサプライヤに対する注文保留を示すアラート

サプライヤ・アプリケーションにより XML 文書がサプライヤ・データベースに挿入されると、次のアクションが発生します。

1. サプライヤ・アプリケーションがサプライヤに対して注文に関するアラートを発行します。オーダー・ステータスは「pending」のままです。
2. サプライヤは、受注した製品が在庫にあるかどうかと、リテーラの与信をチェックした後に、製品の出荷を決定し、「Ship」をクリックします。
3. サプライヤ・アプリケーションにより、サプライヤ・データベースの Ord 表のステータス列が「shipped」に更新されます。

手順 7. AQ Broker - トランスフォーマによるリテーラの形式に従った XML 注文書の変換

1. AQ Broker - トランスフォーマは、サプライヤからのキューを待機 [READS] します。
2. XML 注文出荷済みドキュメントを受信すると、AQ Broker - トランスフォーマはトランスフォーマ・データベース内の Stylesheets 表を参照し、From、To および Task 条件に従って適切な XSL スタイルシートを選択します。スタイルシートは、Stylesheets 表の XSL 列の CLOB に格納されます。AQ Broker - トランスフォーマは、JDBC を使用してデータベースおよびスタイルシートにアクセスします。
3. AQ Broker - トランスフォーマは、AQ を使用して、再フォーマット済みの XML 注文更新書を「TO」宛先リテーラに送信 [WRITE] します。

手順 8. リテーラ・アプリケーションによる Ord および Line_Item 表の更新

1. リテーラのアプリケーションにより、リテーラ・データベースが「shipped」オーダー・ステータス情報で更新されます。Ord 表が更新されます。
2. リテーラは、この情報をどのデバイスからでも表示できます。ステータスは「Shipped」と表示されます。

B2B XML アプリケーションの実行 : 手順の詳細

B2B XML アプリケーションのトランザクションとフローの詳細は、図 8-5 を参照してください。XML 注文書は、AQ Broker - トランスフォーマを介してリテーラとサプライヤの間でやりとりされます。

B2B XML アプリケーションを実行する前に、「オンライン B2B XML アプリケーションを実行するための作業の概要」で説明したスキーマ作成スクリプトの実行を完了していることを確認してください。

次の手順では、処理とこのアプリケーションの実行方法について説明します。

- 手順 1. リテーラによるサプライヤのオンライン「Hi-Tech Mall」カタログのブラウズ
- 手順 2. リテーラによる発注
- 手順 3. 「Validate」によるトランザクションのコミット、リテーラ・アプリケーションによる XML 注文書の生成
- 手順 4. AQ Broker - トランスフォーマによるサプライヤの形式に従った XML 文書の変換
- 手順 5. サプライヤ・アプリケーションによる XML 文書の解析およびサプライヤ・データベースへの注文の挿入
- 手順 6a. サプライヤ・アプリケーションからサプライヤに対する注文保留を示すアラート
- 手順 7. AQ Broker - トランスフォーマによるリテーラの形式への XML 注文書の変換
- 手順 8. リテーラ・アプリケーションによる Ord 表の更新とリテーラに対する新規オーダー・ステータスの表示

手順 1. リテーラによるサプライヤのオンライン「Hi-Tech Mall」カタログのブラウズ

B2B XML アプリケーションのプロシージャ・フローの詳細は、図 8-5 を参照してください。

注意： ここでは、サプライヤのカタログのコピーがリテーラのデータベースに格納されているとします。

---

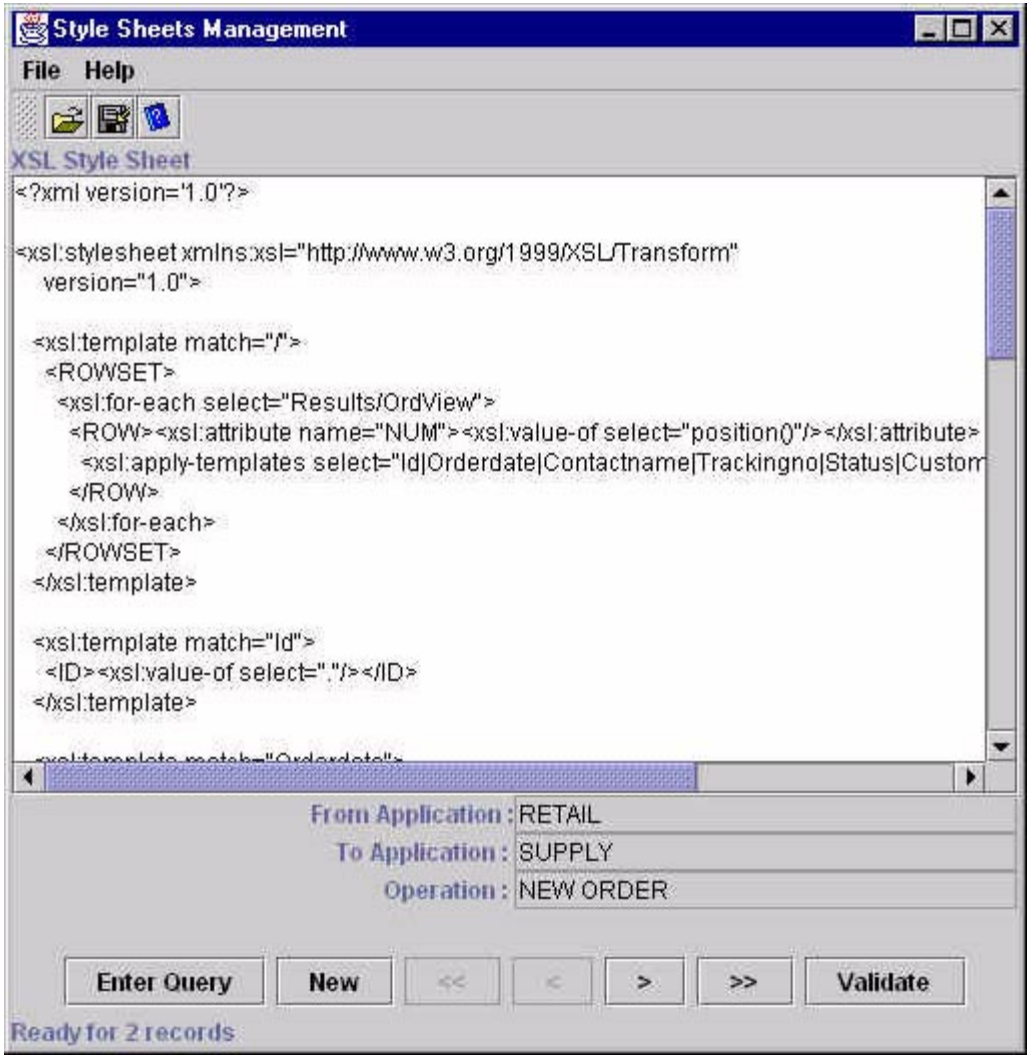
1. `SS.bat` を起動して StyleSheet ユーティリティをチェックし、機能しているかどうかを確認します。

   スタイルシート・バッチ・ファイル : SS.bat

```
@echo off
@echo Stylesheet Util
D:¥jdev31¥java¥bin¥java -mx50m -classpath "D:¥xml817¥references¥olivier_new;
D:¥jdev31¥lib¥jdev-rt.zip;
D:¥jdev31¥jdbc¥lib¥oracle8.1.6¥classes111.zip;
D:¥jdev31¥lib¥connectionmanager.zip;
D:¥jdev31¥lib;
D:¥jdev31¥lib¥oraclexsql.jar;
D:¥jdev31¥lib¥oraclexmlsql.jar;
D:¥jdev31¥lib¥xmlparserv2_2027.jar;
D:¥jdev31¥jfc¥lib¥swingall.jar;
D:¥jdev31¥jswdk-1.0.1¥lib¥servlet.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi11.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi.jar;
D:¥XMLWorkshop¥xmlcomp.jar;
D:¥jdev31¥java¥lib¥classes.zip"  B2BDemo.StyleSheetUtil.GUIStylesheet
```

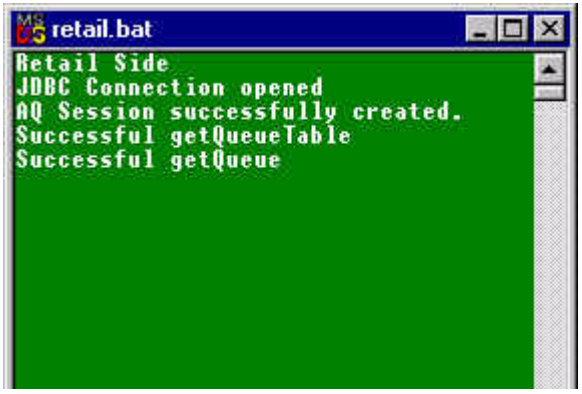
   このユーティリティを使用すると、スタイルシートが格納されている実際の `Stylesheets` 表をブラウズできます。これらのスタイルシートは、AQ Broker - トランスフォーマで受信したドキュメントの処理に使用されます。図 8-7 を参照してください。

図 8-7 スタイルシート・ユーティリティのチェック



2. retailer.bat を実行してリテーラ・アプリケーションを起動します。図 8-8 を参照してください。

図 8-8 リテーラ・アプリケーションの起動



3. broker.bat を実行して AQ Broker - トランスフォーマ・アプリケーションを起動します。図 8-9 を参照してください。

図 8-9 AQ Broker - トランスフォーマ・アプリケーションの起動

4. supplier.bat を実行してサプライヤ・アプリケーションを起動します。図 8-10 を参照してください。

図 8-10 サプライヤ・アプリケーションの起動 :「Supplier Watcher」

リテーラ、AQ Broker - トランスフォーマ（Broker）およびサプライヤ・アプリケーション用の 3 つのバッチ・ファイルを次に示します。

retailer.bat

```
@echo off
@echo Retail Side
D:¥jdev31¥java¥bin¥java -mx50m -classpath "D:¥xml817¥references¥Ora817DevGuide;
D:¥jdev31¥lib¥jdev-rt.zip;
D:¥jdev31¥jdbc¥lib¥oracle8.1.6¥classes111.zip;
D:¥jdev31¥lib¥connectionmanager.zip;
D:¥jdev31¥lib;
D:¥jdev31¥lib¥oraclexsql.jar;
D:¥jdev31¥lib¥oraclexmlsql.jar;
D:¥jdev31¥lib¥xmlparserv2_2027.jar;
D:¥jdev31¥jfc¥lib¥swingall.jar;
D:¥jdev31¥jswdk-1.0.1¥lib¥servlet.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi11.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi.jar;
D:¥XMLWorkshop¥xmlcomp.jar;
D:¥jdev31¥java¥lib¥classes.zip"  B2BDemo.Retailer.UpdateMaster -step=1000
-verbose=y -dbURL=jdbc:oracle:thin:@atp-1.us.oracle.com:1521:ORCL
```

broker.bat

```
@echo off
@echo Broker
D:¥jdev31¥java¥bin¥java -mx50m -classpath "D:¥xml817¥references¥Ora817DevGuide;
D:¥jdev31¥lib¥jdev-rt.zip;
D:¥jdev31¥jdbc¥lib¥oracle8.1.6¥classes111.zip;
D:¥jdev31¥lib¥connectionmanager.zip;
D:¥jdev31¥lib;D:¥jdev31¥lib¥oraclexsql.jar;
D:¥jdev31¥lib¥oraclexmlsql.jar;
D:¥jdev31¥lib¥xmlparserv2_2027.jar;
D:¥jdev31¥jfc¥lib¥swingall.jar;
D:¥jdev31¥jswdk-1.0.1¥lib¥servlet.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi11.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi.jar;
D:¥XMLWorkshop¥xmlcomp.jar;
D:¥jdev31¥java¥lib¥classes.zip"  B2BDemo.Broker.MessageBroker -step=1000
-verbose=y
```

supplier.bat

```
@echo off
@echo Supplier
D:¥jdev31¥java¥bin¥java -mx50m -classpath "D:¥xml817¥references¥Ora817DevGuide;
D:¥jdev31¥lib¥jdev-rt.zip;
D:¥jdev31¥jdbc¥lib¥oracle8.1.6¥classes111.zip;
D:¥jdev31¥lib¥connectionmanager.zip;
D:¥jdev31¥lib;D:¥jdev31¥lib¥oraclexsql.jar;
D:¥jdev31¥lib¥oraclexmlsql.jar;
D:¥jdev31¥lib¥xmlparserv2_2027.jar;
D:¥jdev31¥jfc¥lib¥swingall.jar;
D:¥jdev31¥jswdk-1.0.1¥lib¥servlet.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi11.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi.jar;
D:¥XMLWorkshop¥xmlcomp.jar;
D:¥jdev31¥java¥lib¥classes.zip"  B2BDemo.Supplier.SupplierWatcher -step=1000
-verbose=y -dbURL=jdbc:oracle:thin:@atp-1.us.oracle.com:1521:ORCL
```

5. 最後に、ブラウザ、Palm Pilot などの PDA、携帯電話または他のデバイスからクライアントを起動します。
6. [リテーラ]: ログインします。「Welcome!」画面が表示されます。図 8-11 を参照してください。

XSQL スクリプトの例 1: ログイン・ユーザーの ID のチェック : getlogged.xsql

```
<?xml version="1.0"?>

<!--
 | Second script to be called.
 | Check if the user in known in the database.
 | $Author: olediour@us $
 | $Revision: 1.1 $
 +-->
<?xml-stylesheet type="text/xsl" media="HandHTTP"     href="PP.xsl"?>
<?xml-stylesheet type="text/xsl" media="Mozilla"      href="HTML.xsl"?>

<loginResult xmlns:xsql="urn:oracle-xsql"
             connection="retail"
             custName="XXX">
  <pageTitle>Hi-Tech Mall</pageTitle>
  <xsql:query tag-case="upper">
    <![CDATA[
    select C.ID, C.NAME
    from CUSTOMER C
    where Upper(C.NAME) = Upper('{@custName}')
    ]]>
    <xsql:no-rows-query>
      Select '{@custName}' as "unknown" from dual
    </xsql:no-rows-query>
  </xsql:query>
  <nextStep>inventory.xsql</nextStep>
  <returnHome>index.xsql</returnHome>

</loginResult>
```

この XSQL スクリプトでは、次の XSL スクリプトがコールされます。

- pp.xsl。「XSL スタイルシートの例 1: HTML への結果の変換 ― html.xsl」を参照してください。
- html.xsl。「XSL スタイルシートの例 2: Palm Pilot ブラウザ用の結果の変換 ― pp.xsl」を参照してください。

図 8-11 [リテーラ]: ブラウザまたは PDA からのログイン

7. [リテーラ]: 「Please Enter the Mall」をクリックします。

XSQL スクリプトの例 2: 初期「Hi-Tech Mall」画面の表示 ー index.xsql

```
<?xml version="1.0"?>

<!--
 | This is the entry point in the application.
 | Notice that this script does not access the database.
 | $Author: olediour@us $
 | $Revision: 1.1 $
 +-->
<?xml-stylesheet type="text/xsl" media="HandHTTP"     href="PP.xsl"?>
<?xml-stylesheet type="text/xsl" media="Mozilla"      href="HTML.xsl"?>

<index xmlns:xsql="urn:oracle-xsql">
  <pageTitle>Hi-Tech Mall</pageTitle>
  <form action="getLogged.xsql" method="post">
    <field type="text" name="custName" prompt="Your ID"/>
    <button type="submit" label="Log In"/>
  </form>
</index>
```

8. [リテーラ]: 画面に「Hi-Tech Mall Catalog」の製品リストが表示されます。必要な製品をクリックします。図 8-12 を参照してください。

XSQL スクリプトの例 3: カタログ製品のリスト表示 ー inventory.xsql

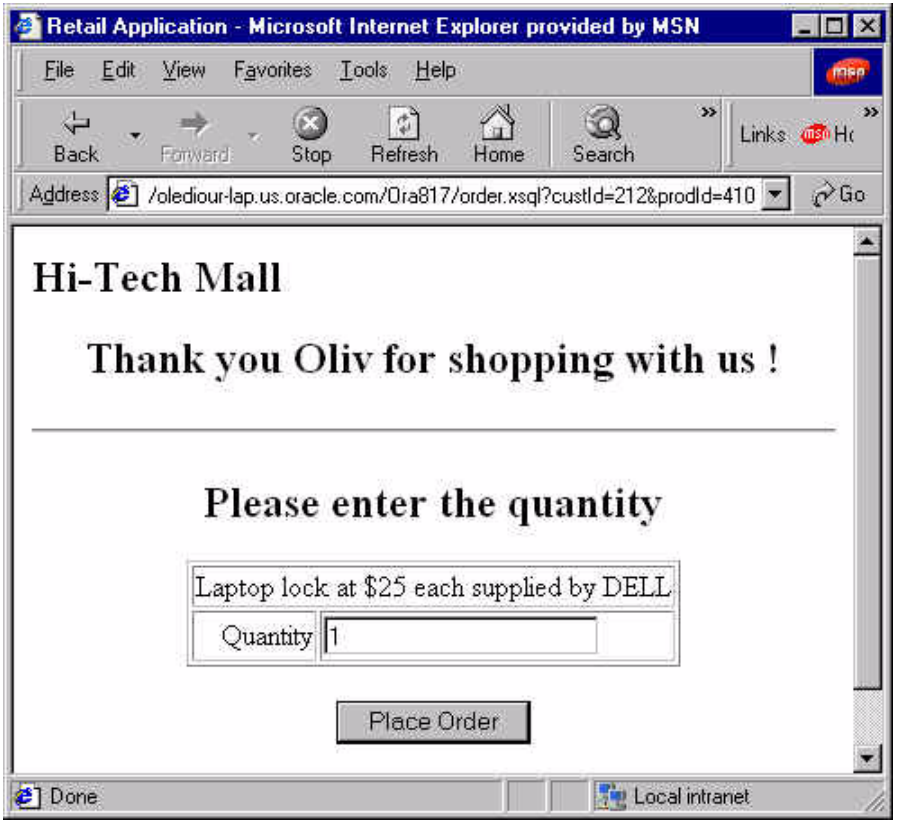
```
<?xml version="1.0"?>

<!--
 | This is the third script called.
 | It produces the catalog from the Retailer's database.
 |
 | $Author: olediour@us $
 | $Revision: 1.1 $
 +-->
<?xml-stylesheet type="text/xsl" media="HandHTTP"     href="PP.xsl"?>
<?xml-stylesheet type="text/xsl" media="Mozilla"      href="HTML.xsl"?>

<inventory xmlns:xsql="urn:oracle-xsql"
           connection="retail"
           custId="000">
  <pageTitle>Hi-Tech Mall</pageTitle>
  <form action="order.xsql" method="post">
    <hiddenFields>
      <xsql:include-param name="custId"/>
    </hiddenFields>
```

```
    <theMart>
      <xsql:query tag-case="upper">
        <![CDATA[
        select I.ID,
               I.DESCRIPTION,
               I.PRICE,
               S.NAME
        from INVENTORY_ITEM I,
             SUPPLIER S
        where I.SUPPLIER_ID = S.ID
        ]]>
        <xsql:no-rows-query>
          Select 'No items !' as "Wow" from dual
        </xsql:no-rows-query>
      </xsql:query>
    </theMart>
  </form>
  <returnHome>index.xsql</returnHome>

</inventory>
```

図 8-12 [リテーラ]: 「Hi-Tech Mall (Mart) Catalog」へのアクセス

9. [リテーラ]: 必要な数量を入力して「Place Order」ボタンをクリックします。図 8-13 を参照してください。

図 8-13 [リテーラ]: 数量の入力と「Place Order」のクリック



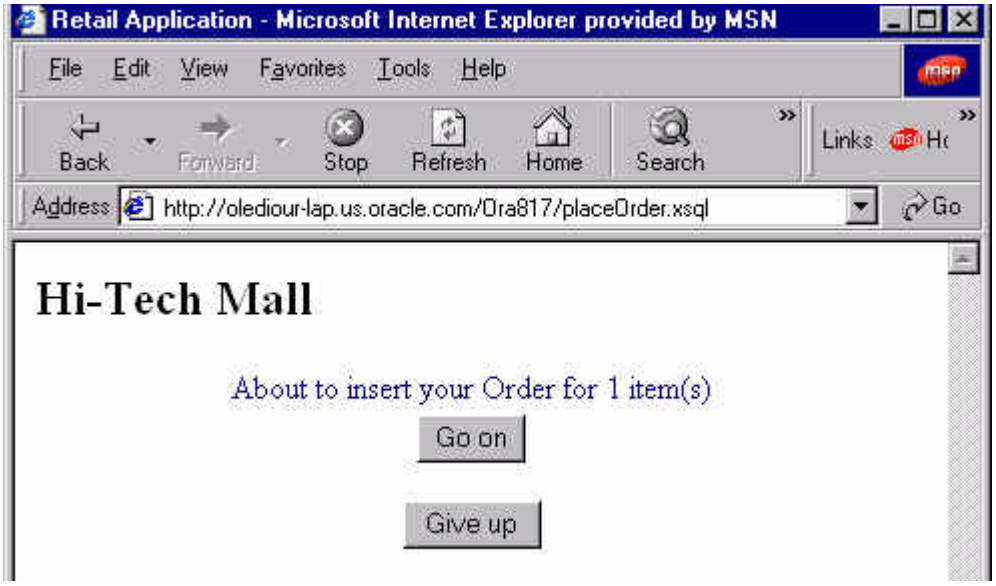
10. [リテーラ]: 「Go On」または「Give Up」をクリックします。図 8-14 を参照してください。

XSQL スクリプトの例 4: 数量の入力 ー order.xsql

```
<?xml version="1.0"?>
<!--
 | This is the fourth script called.
 | It prompts you to enter a quantity.
 |
 | $Author: olediour@us $
 | $Revision: 1.1 $
 +-->
<?xml-stylesheet type="text/xsl" media="HandHTTP"     href="PP.xsl"?>
<?xml-stylesheet type="text/xsl" media="Mozilla"      href="HTML.xsl"?>

<order xmlns:xsql="urn:oracle-xsql"
       connection="retail"
       custId="000"
       prodId="000">
  <pageTitle>Hi-Tech Mall</pageTitle>
  <xsql:query tag-case      = "upper"
              rowset-element= ""
              row-element   = "cust">
    <![CDATA[
    select C.ID,
           C.NAME
    from CUSTOMER C
    where C.ID = '{@custId}'
    ]]>
    <xsql:no-rows-query>
      Select '{@custId}' as "unknown" from dual
    </xsql:no-rows-query>
  </xsql:query>

  <xsql:query tag-case="upper"
       rowset-element=""
       row-element="prod">
    <![CDATA[
        select I.ID,
               I.DESCRIPTION,
               I.PRICE,
               S.NAME
        from INVENTORY_ITEM I,
             SUPPLIER S
        where I.SUPPLIER_ID = S.ID and
              I.ID = '{@prodId}'
    ]]>
    <xsql:no-rows-query>
      Select '{@prodId}' as "unknown" from dual
```

```
    </xsql:no-rows-query>
  </xsql:query>

  <returnHome>index.xsql</returnHome>
</order>
```

図 8-14 [リテーラ]: 「Go On」のクリック



手順 2. リテーラによる発注

リテーラが「Go On」をクリックすると、アプリケーションにより注文がチェックされ、リテーラの与信履歴もチェックされる場合があります。その後、「Validate」をクリックして注文を確認します。図 8-15 および図 8-16 を参照してください。

図 8-15 [リテーラ]: 「Validate」のクリック

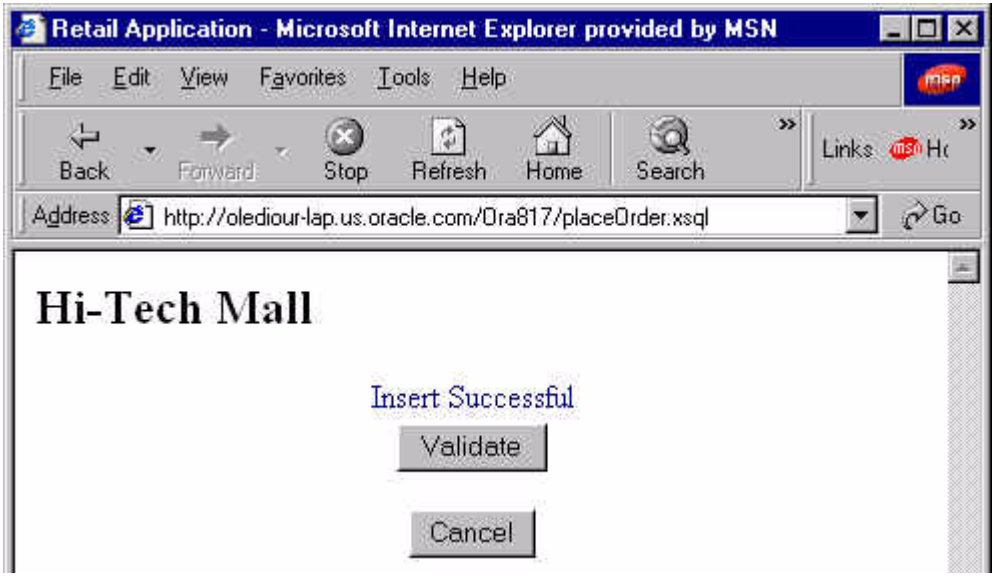
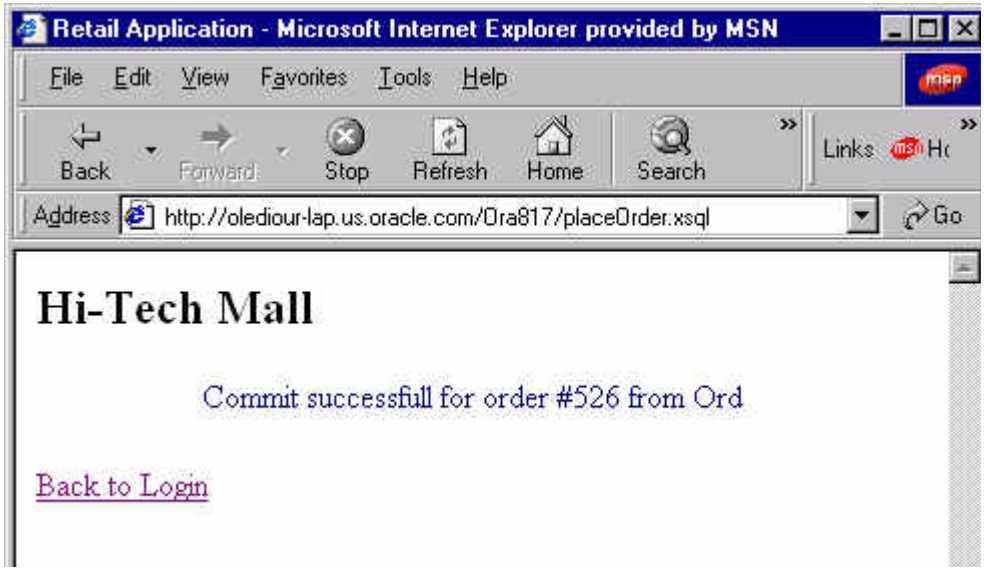


図 8-16 [リテーラ]: コミットの成功および表 Ord の更新完了



手順 3.「Validate」によるトランザクションのコミット、リテーラ・アプリケーションによる XML 注文書の生成

1. 「Validate」をクリックすると、XSQL Servlet のアクション・ヘッダーを使用してメイン B2B プロセスが実行されます。これでクライアントによる介入は終了です。

   次の各スクリプトは B2B アプリケーション（demo）により実行されます。

   - XSQL スクリプトの例 5: B2B プロセスの起動 ー placeorder.xsql
   - Java の例 1: placeOrder.xsql によるアクション・ハンドラのコール ー RetailActionHandler.java
   - Java の例 2: RetailActionHandler.java のセッション・コンテキストのメンテナンス ー SessionHolder.java

XSQL スクリプトの例 5: B2B プロセスの起動 ー placeorder.xsql

```
<?xml version="1.0"?>
<!--
 | This is the fifth and last, but not least, script called.
 | This script actually fires the whole B2B process.
 | It uses the Action Handler facility of XSQL Servlet.
 |
 | $Author: olediour@us $
 | $Revision: 1.1 $
 +-->
<?xml-stylesheet type="text/xsl" media="HandHTTP"     href="PP.xsl"?>
<?xml-stylesheet type="text/xsl" media="Mozilla"      href="HTML.xsl"?>

<placeOrder xmlns:xsql="urn:oracle-xsql"
            connection="retail"
            dbUrl     ="jdbc:oracle:thin:@atp-1.us.oracle.com:1521:ORCL"
            username  ="retailer"
            password  ="retailer"
            entity    ="Ord"
            operation ="insert"
            custId    =""
            ordId     =""
            prodId    =""
            qty       =""><
  <xsql:include-request-params/>
  <pageTitle>Hi-Tech Mall</pageTitle>
  <pageSeparator/>

  <xsql:action handler    ="B2BDemo.XSQLActionHandler.RetailActionHandler"
               dbUrl      ="{@dbUrl}"
```

```
                username   ="{@username}"
                password   ="{@password}"
                entity     ="{@entity}"
                operation  ="{@operation}"
                custId     ="{@custId}"
                ordId      ="{@ordId}"
                prodId     ="{@prodId}"
                qty        ="{@qty}"/>
    <pageSeparator/>
    <bottomLinks>
      <aLink href="placeOrder.xsql?operation=rollback">Rollback</aLink>
    </bottomLinks>
    <returnHome>index.xsql</returnHome>
  </placeOrder>
```

Java の例 1: placeOrder.xsql によるアクション・ハンドラのコール ー RetailActionHandler.java

注意： 次の例は、ほぼ 20 ページあります。

```
package B2BDemo.XSQLActionHandler;
/**
 * Action Handler called by the placeOrder.xsql script.
 * Actually fires the B2B process itself.
 * Uses SessionHolder to maintain transaction state.
 *
 * @see SessionHolder
 * @see placeOrder.xsql
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Corp.
 */
import oracle.xml.xsql.*;
import oracle.xml.xsql.actions.XSQLIncludeXSQLHandler;
import javax.servlet.http.*;
import javax.servlet.*;
import org.w3c.dom.*;

import java.sql.*;
import java.io.*;

import oracle.xml.parser.v2.*;

import B2BDemo.AQUtil.*;
import B2BDemo.*;
```

```
import B2BDemo.XMLUtil.*;

public class RetailActionHandler extends XSQLActionHandlerImpl
{
  private static final boolean verbose   = false;
  private static final boolean debugFile = false;

  private Connection actionConnection = null;

  private String appUrl      = "";
  private String appUser     = "";
  private String appPassword = "";

  public static final String DBURL       = "dbUrl";
  public static final String USERNAME    = "username";
  public static final String PASSWORD    = "password";

  public static final String OPERATION   = "operation";

  public static final String ENTITY      = "entity";

  public static final String ORDID       = "ordId";
  public static final String ORDERDATE   = "orderDate";
  public static final String CONTACTNAME = "contactName";
  public static final String TRACKINGNO  = "trackingNo";
  public static final String STATUS      = "status";
  public static final String CUSTID      = "custId";

  public static final String QTY         = "qty";
  public static final String PRODID      = "prodId";

  public static final String SELECT      = "select";
  public static final String INSERT      = "insert";
  public static final String BEGIN       = "begin";
  public static final String COMMIT      = "commit";
  public static final String ROLLBACK    = "rollback";

  XSQLActionHandler nestedHandler = null;
  String operation = null;

  String entity       = null;
  String ordId        = null;
  String orderDate    = null;
  String contactName  = null;
  String trackingNo   = null;
  String status       = null;
  String custId       = null;
```

```
String qty          = null;
String prodId       = null;

HttpServletRequest     request  = null;
HttpServletResponse    response = null;
HttpSession            session  = null;

public void init(XSQLPageRequest xspRequest, Element action)
{
  super.init(xspRequest, action);
  // Retrieve the parameters

  if (verbose)
    System.out.println("init Action Handler...................");

  appUrl      = getAttributeAllowingParam(DBURL,    action);
  appUser     = getAttributeAllowingParam(USERNAME, action);
  appPassword = getAttributeAllowingParam(PASSWORD, action);

  operation = getAttributeAllowingParam(OPERATION, action);
  entity    = getAttributeAllowingParam(ENTITY, action);

  ordId       = getAttributeAllowingParam(ORDID,       action);
  orderDate   = getAttributeAllowingParam(ORDERDATE,   action);
  contactName = getAttributeAllowingParam(CONTACTNAME, action);
  trackingNo  = getAttributeAllowingParam(TRACKINGNO,  action);
  status      = getAttributeAllowingParam(STATUS,      action);
  custId      = getAttributeAllowingParam(CUSTID,      action);
  prodId      = getAttributeAllowingParam(PRODID,      action);
  qty         = getAttributeAllowingParam(QTY,         action);
  //
  if (verbose)
  {
    System.out.println("OrdID  > " + ordId);
    System.out.println("CustID > " + custId);
    System.out.println("ProdID > " + prodId);
  }

  final String HOLDER_NAME = "XSQLActionHandler.connection";
  try
  {
    if (xspRequest.getRequestType().equals("Servlet"))
    {
      XSQLServletPageRequest xspr = (XSQLServletPageRequest)xspRequest;
      HttpServletRequest req      = xspr.getHttpServletRequest();
      session = req.getSession(true); // true : Create if missing !!!
      if (verbose)
```

```
        System.out.println("Session Id = " + session.getId() + " - new : " +
                                                              session.isNew());
      SessionHolder sh = (SessionHolder) session.getValue(HOLDER_NAME);
      if (sh == null)
      {
        if (verbose)
          System.out.println("New SessionHandler > Getting connected at " +
                                                    (new java.util.Date()));
        actionConnection = getConnected(appUrl, appUser, appPassword);
        sh = new SessionHolder(actionConnection);
        session.putValue(HOLDER_NAME, sh);
      }
      actionConnection = sh.getConnection();
      if (verbose)
      {
        System.out.println("Reusing Connection at " + (new java.util.Date()) +
                           " - Opened at " + sh.getOpenDate().toString());
        System.out.println("Driver     : " +
                          actionConnection.getMetaData().getDriverName());
        System.out.println("SessionId  : " + session.getId());
        System.out.println("AutoCommit : " +
                                      actionConnection.getAutoCommit());
      }
    }
  }
  catch (Exception e)
  {
    System.err.println("Error in retrieving session context ¥n" + e);
    e.printStackTrace();
  }
}

// The result is the out parameter
public void handleAction(Node result) throws SQLException
{
  XSQLPageRequest xpr = getPageRequest();
  if (xpr.getRequestType().equals("Servlet"))
  {
    // Get the servlet context and components
    XSQLServletPageRequest xspr = (XSQLServletPageRequest)xpr;
    request  = xspr.getHttpServletRequest();
    response = xspr.getHttpServletResponse();

    Document doc = null;

    // Display CLASSPATH
    XMLDocument myDoc = new XMLDocument();
```

```
try
{
  Element root = myDoc.createElement("root");
  myDoc.appendChild(root);

  Element cp = myDoc.createElement("ClassPath");
  root.appendChild(cp);

  // The text is a descendant of its node
  Node cpTxt = myDoc.createTextNode("text#");
  cpTxt.setNodeValue(System.getProperty("java.class.path"));
  cp.appendChild(cpTxt);

  Element e = myDoc.getDocumentElement();
  e.getParentNode().removeChild(e);
  result.appendChild(e);  // Append child to result before returning it.
}
catch (Exception e)
{
  System.err.println("Building XMLDoc");
  e.printStackTrace();
}
try
{
  // Add a node to hold operation value
  XMLDocument xmlDoc = new XMLDocument();
  Element elmt = xmlDoc.createElement("requiredOperation");
  xmlDoc.appendChild(elmt);
  Node theText = xmlDoc.createTextNode("text#");
  theText.setNodeValue(operation);
  elmt.appendChild(theText);
  // Append to result
  Element e = xmlDoc.getDocumentElement();
  e.getParentNode().removeChild(e);
  result.appendChild(e);  // Append child to result before returning it.
}
catch (Exception e)
{
  System.err.println("Building XMLDoc (2)");
  e.printStackTrace();
}

try
{
  // Dispatch
  if (operation.equals(SELECT))
 /* doc = manageSelect() */;
```

```
  else if (operation.equals(INSERT))
    doc = manageInsert();
  else if (operation.equals(BEGIN))
    doc = doBegin();
  else if (operation.equals(COMMIT))
    doc = doCommit();
  else if (operation.equals(ROLLBACK))
    doc = doRollback();
  else // Wrong operation
  {
    XMLDocument xmlDoc = new XMLDocument();
    Element elmt = xmlDoc.createElement("unknownOperation");
    xmlDoc.appendChild(elmt);
    Node theText = xmlDoc.createTextNode("text#");
    theText.setNodeValue(operation);
    elmt.appendChild(theText);
    // Append to result
    Element e = xmlDoc.getDocumentElement();
    e.getParentNode().removeChild(e);
    result.appendChild(e);  // Append child to result before returning it.
  }
}
catch (Exception ex)
{
// file://this.reportError(e);
  XMLDocument xmlDoc = new XMLDocument();
  Element elmt = xmlDoc.createElement("operationProblem");
  xmlDoc.appendChild(elmt);
  Node theText = xmlDoc.createTextNode("text#");
  theText.setNodeValue(ex.toString());
  elmt.appendChild(theText);
  // Append to result
  Element e = xmlDoc.getDocumentElement();
  e.getParentNode().removeChild(e);
  result.appendChild(e);  // Append child to result before returning it.
}

try
{
  if (doc != null)
  {
    Element e = doc.getDocumentElement();
    e.getParentNode().removeChild(e);
    result.appendChild(e);  // Append child to result before returning it.
  }
}
catch (Exception e)
```

```
      {
        try
        {
          ServletOutputStream out = response.getOutputStream();
          out.println(e.toString());
        }
        catch (Exception ex) {}
      }
    }
    else  // Command line ?
    {
      System.out.println("Request type is [" + xpr.getRequestType() + "]");
    }
  }

/**
 * Removed because uselezss in this demo.
 *
  private Document manageSelect() throws Exception
  {
    Document doc = null;
    String cStmt = "";

    if (custId != null && custId.length() > 0)
      vo.setWhereClause("Customer_Id = '" + custId + "'");
    else
      vo.setWhereClause(null);
    vo.executeQuery();
    doc = data.getXMLDocument(); // Query implicitly executed !
    return doc;
  }
*/
  private Document manageInsert() throws Exception
  {
    Document doc = null;

    if (entity.equals("Ord"))
      doc = insertInOrd();
    else if (entity.equals("LineItem"))
      doc = insertInLine();
    else
    {
      doc = new XMLDocument();
      Element elmt = doc.createElement("operationQuestion");
      Attr attr = doc.createAttribute("opType");
      attr.setValue("insert");
      elmt.setAttributeNode(attr);
```

```
    doc.appendChild(elmt);
    Node txt = doc.createTextNode("text#");
    elmt.appendChild(txt);
    txt.setNodeValue("Don't know what to do with " + entity);
  }
  return doc;
}

private Document insertInOrd()
{
  Document doc = null;
  if (custId == null || custId.length() == 0)
  {
    doc = new XMLDocument();
    Element elmt = doc.createElement("operationProblem");
    Attr attr = doc.createAttribute("opType");
    attr.setValue("OrdInsert");
    elmt.setAttributeNode(attr);
    doc.appendChild(elmt);
    Node txt = doc.createTextNode("text#");
    elmt.appendChild(txt);
    txt.setNodeValue("Some element(s) missing for ord insert (custId)");
  }
  else
  {
    String seqStmt = "select Ord_Seq.nextVal from dual";
    String seqVal = "";
    try
    {
      Statement stmt = actionConnection.createStatement();
      ResultSet rSet = stmt.executeQuery(seqStmt);
      while (rSet.next())
        seqVal = rSet.getString(1);
      rSet.close();
      stmt.close();
    }
    catch (SQLException e)
    {
      System.err.println("Error reading ORD_SEQ Sequence : " + e.toString());
    }
    //                                  1          2          3             4
    String cStmt = "insert into ORD values (?, sysdate, ?, 'AX' || ?,
                                                          'Pending', ?)";
    try
    {
      if (verbose)
        System.out.println("Inserting Order # " + seqVal);
```

```
  PreparedStatement pStmt = actionConnection.prepareStatement(cStmt);
  pStmt.setString(1, seqVal);
  pStmt.setString(2, "Ora817");  // Default value !
  pStmt.setString(3, seqVal);
  pStmt.setString(4, custId);
  pStmt.execute();
  pStmt.close();
/**
  try
  {
    Statement stmt = actionConnection.createStatement();
    ResultSet rSet = stmt.executeQuery("SELECT * FROM ORD WHERE ID = " +
                                                                  seqVal);
    int i = 0;
    while (rSet.next())
      i++;
    if (verbose)
      System.out.println(i + " record found for " + seqVal);
    rSet.close();
    stmt.close();
  }
  catch (SQLException e)
  {
    System.err.println("Error : " + e.toString());
  }
 */
  doc = new XMLDocument();
  Element elmt = doc.createElement("operationResult");
  Attr attr = doc.createAttribute("opType");
  attr.setValue("insert");
  elmt.setAttributeNode(attr);

  attr = doc.createAttribute("Step");
  attr.setValue(entity);
  elmt.setAttributeNode(attr);

  doc.appendChild(elmt);
  Node txt = doc.createTextNode("text#");
  elmt.appendChild(txt);
  txt.setNodeValue("About to insert your Order for " + qty + " item(s)");

  Element nextElmt = doc.createElement("nextStep");
  elmt.appendChild(nextElmt);
  attr = doc.createAttribute("Label");
  attr.setValue("Go on");
  nextElmt.setAttributeNode(attr);
```

```
attr = doc.createAttribute("Action");
nextElmt.setAttributeNode(attr);
attr.setValue("placeOrder.xsql");
Element pList = doc.createElement("prmList");
nextElmt.appendChild(pList);
// viewobject
Element prm = doc.createElement("prm");
pList.appendChild(prm);
attr = doc.createAttribute("name");
attr.setValue("entity");
prm.setAttributeNode(attr);
attr = doc.createAttribute("value");
attr.setValue("LineItem");
prm.setAttributeNode(attr);
// custId
prm = doc.createElement("prm");
pList.appendChild(prm);
attr = doc.createAttribute("name");
attr.setValue("custId");
prm.setAttributeNode(attr);
attr = doc.createAttribute("value");
attr.setValue(custId);
prm.setAttributeNode(attr);
// prodId
prm = doc.createElement("prm");
pList.appendChild(prm);
attr = doc.createAttribute("name");
attr.setValue("prodId");
prm.setAttributeNode(attr);
attr = doc.createAttribute("value");
attr.setValue(prodId);
prm.setAttributeNode(attr);
// qty
prm = doc.createElement("prm");
pList.appendChild(prm);
attr = doc.createAttribute("name");
attr.setValue("qty");
prm.setAttributeNode(attr);
attr = doc.createAttribute("value");
attr.setValue(qty);
prm.setAttributeNode(attr);
// ordId
prm = doc.createElement("prm");
pList.appendChild(prm);
attr = doc.createAttribute("name");
attr.setValue("ordId");
prm.setAttributeNode(attr);
```

```
  attr = doc.createAttribute("value");
  attr.setValue(seqVal);
  prm.setAttributeNode(attr);

  nextElmt = doc.createElement("nextStep");
  elmt.appendChild(nextElmt);
  attr = doc.createAttribute("Label");
  attr.setValue("Give up");
  nextElmt.setAttributeNode(attr);

  attr = doc.createAttribute("Action");
  nextElmt.setAttributeNode(attr);
  attr.setValue("placeOrder.xsql");
  pList = doc.createElement("prmList");
  nextElmt.appendChild(pList);
  // viewobject
  prm = doc.createElement("prm");
  pList.appendChild(prm);
  attr = doc.createAttribute("name");
  attr.setValue("operation");
  prm.setAttributeNode(attr);
  attr = doc.createAttribute("value");
  attr.setValue("rollback");
  prm.setAttributeNode(attr);
}
catch (Exception e)
{
  doc = new XMLDocument();
  Element elmt = doc.createElement("operationProblem");
  Attr attr = doc.createAttribute("opType");
  attr.setValue("insert");
  elmt.setAttributeNode(attr);

  attr = doc.createAttribute("Step");
  attr.setValue(entity);
  elmt.setAttributeNode(attr);

  doc.appendChild(elmt);
  Node txt = doc.createTextNode("text#");
  elmt.appendChild(txt);
  txt.setNodeValue(e.toString());
  if (verbose)
    System.out.println("Error : " + e.toString());
  Element prm = doc.createElement("parameters");
  elmt.appendChild(prm);
  // ID
  Element prmVal = doc.createElement("ID");
```

```
      prm.appendChild(prmVal);
      txt = doc.createTextNode("text#");
      prmVal.appendChild(txt);
      txt.setNodeValue(ordId);
      // CUSTOMER_ID
      prmVal = doc.createElement("CUSTOMER_ID");
      prm.appendChild(prmVal);
      txt = doc.createTextNode("text#");
      prmVal.appendChild(txt);
      txt.setNodeValue(custId);
    }
  }
  return doc;
}

private Document insertInLine()
{
  Document doc = null;
  if (custId == null || custId.length() == 0 ||
      qty == null || qty.length() == 0 ||
      prodId == null || prodId.length() == 0 ||
      ordId == null || ordId.length() == 0)
  {
    doc = new XMLDocument();
    Element elmt = doc.createElement("operationProblem");
    Attr attr = doc.createAttribute("opType");
    attr.setValue("lineInsert");
    elmt.setAttributeNode(attr);
    doc.appendChild(elmt);
    Node txt = doc.createTextNode("text#");
    elmt.appendChild(txt);
    txt.setNodeValue("Some element(s) missing for line insert (" +
                     ((custId == null || custId.length() == 0)?"custId ":"") +
                     ((qty == null || qty.length() == 0)?"qty ":"") +
                     ((prodId == null || prodId.length() == 0)?"prodId ":"") +
                     ((ordId == null || ordId.length() == 0)?"ordId ":"") +")"
                     );

    Element subElmt = doc.createElement("custId");
    elmt.appendChild(subElmt);
    txt = doc.createTextNode("text#");
    subElmt.appendChild(txt);
    txt.setNodeValue(custId);

    subElmt = doc.createElement("qty");
    elmt.appendChild(subElmt);
    txt = doc.createTextNode("text#");
```

```
  subElmt.appendChild(txt);
  txt.setNodeValue(qty);

  subElmt = doc.createElement("prodId");
  elmt.appendChild(subElmt);
  txt = doc.createTextNode("text#");
  subElmt.appendChild(txt);
  txt.setNodeValue(prodId);

  subElmt = doc.createElement("ordId");
  elmt.appendChild(subElmt);
  txt = doc.createTextNode("text#");
  subElmt.appendChild(txt);
  txt.setNodeValue(ordId);
}
else
{
  if (verbose)
    System.out.println("Inserting line : Ord>" + ordId + ", Prod>" + prodId
                                                    + ", Qty>" + qty);
/**
  try
  {
    Statement stmt = actionConnection.createStatement();
    ResultSet rSet = stmt.executeQuery("SELECT * FROM ORD WHERE ID = " +
                                                          ordId);
    int i = 0;
    while (rSet.next())
      i++;
    System.out.println(i + " record found for " + ordId);
    rSet.close();
    stmt.close();
  }
  catch (SQLException e)
  {
    System.err.println("Error : " + e.toString());
  }
 */
  String cStmt = "insert into line_item values (Line_item_seq.nextVal, ?, ?,
                                                        ?, 0)";
  try
  {
    PreparedStatement pStmt = actionConnection.prepareStatement(cStmt);
    pStmt.setString(1, qty);
    pStmt.setString(2, prodId);
    pStmt.setString(3, ordId);
    pStmt.execute();
```

```
pStmt.close();

doc = new XMLDocument();
Element elmt = doc.createElement("operationResult");
Attr attr = doc.createAttribute("opType");
attr.setValue("insert");
elmt.setAttributeNode(attr);

attr = doc.createAttribute("Step");
attr.setValue(entity);
elmt.setAttributeNode(attr);

doc.appendChild(elmt);
Node txt = doc.createTextNode("text#");
elmt.appendChild(txt);
txt.setNodeValue("Insert Successful");

Element nextElmt = doc.createElement("nextStep");
elmt.appendChild(nextElmt);
attr = doc.createAttribute("Label");
attr.setValue("Validate");
nextElmt.setAttributeNode(attr);

attr = doc.createAttribute("Action");
nextElmt.setAttributeNode(attr);
attr.setValue("placeOrder.xsql");
Element pList = doc.createElement("prmList");
nextElmt.appendChild(pList);
// operation
Element prm = doc.createElement("prm");
pList.appendChild(prm);
attr = doc.createAttribute("name");
attr.setValue("operation");
prm.setAttributeNode(attr);
attr = doc.createAttribute("value");
attr.setValue("commit");
prm.setAttributeNode(attr);
// ordId
prm = doc.createElement("prm");
pList.appendChild(prm);
attr = doc.createAttribute("name");
attr.setValue("ordId");
prm.setAttributeNode(attr);
attr = doc.createAttribute("value");
attr.setValue(ordId);
prm.setAttributeNode(attr);
```

```
  nextElmt = doc.createElement("nextStep");
  elmt.appendChild(nextElmt);
  attr = doc.createAttribute("Label");
  attr.setValue("Cancel");
  nextElmt.setAttributeNode(attr);

  attr = doc.createAttribute("Action");
  nextElmt.setAttributeNode(attr);
  attr.setValue("placeOrder.xsql");
  pList = doc.createElement("prmList");
  nextElmt.appendChild(pList);
  // operation
  prm = doc.createElement("prm");
  pList.appendChild(prm);
  attr = doc.createAttribute("name");
  attr.setValue("operation");
  prm.setAttributeNode(attr);
  attr = doc.createAttribute("value");
  attr.setValue("rollback");
  prm.setAttributeNode(attr);
}
catch (Exception e)
{
  if (verbose)
    System.out.println("Error when inserting " + e.toString());

  doc = new XMLDocument();
  Element elmt = doc.createElement("operationProblem");
  Attr attr = doc.createAttribute("opType");
  attr.setValue("insert");
  elmt.setAttributeNode(attr);

  attr = doc.createAttribute("Step");
  attr.setValue(entity);
  elmt.setAttributeNode(attr);
  doc.appendChild(elmt);

  Node txt = doc.createTextNode("text#");
  elmt.appendChild(txt);
  txt.setNodeValue(e.toString());

  Element prm = doc.createElement("parameters");
  elmt.appendChild(prm);
  // ID
  Element prmVal = doc.createElement("ORD_ID");
  prm.appendChild(prmVal);
  txt = doc.createTextNode("text#");
```

```
      prmVal.appendChild(txt);
      txt.setNodeValue(ordId);
      // QTY
      prmVal = doc.createElement("QTY");
      prm.appendChild(prmVal);
      txt = doc.createTextNode("text#");
      prmVal.appendChild(txt);
      txt.setNodeValue(qty);
      // ITEM_ID
      prmVal = doc.createElement("ITEM_ID");
      prm.appendChild(prmVal);
      txt = doc.createTextNode("text#");
      prmVal.appendChild(txt);
      txt.setNodeValue(prodId);
    }
  }
  return doc;
}

private Document doCommit() throws Exception
{
  Document doc = null;
  actionConnection.commit();

  doc = new XMLDocument();
  Element elmt = doc.createElement("operationResult");
  Attr attr = doc.createAttribute("opType");
  attr.setValue("commit");
  elmt.setAttributeNode(attr);
  doc.appendChild(elmt);
  Node txt = doc.createTextNode("dummy");
  elmt.appendChild(txt);
  txt.setNodeValue("Commit successfull for order #" + ordId + " from " + entity);

  if (ordId != null && ordId.length() > 0)
  {
    // Generate XML Document to send to AQ
    // Start from Ord with OrdId value -

    AQWriter aqw = null;

    aqw = new AQWriter(AppCste.AQuser,
                       AppCste.AQpswd,
                       AppCste.AQDBUrl,
                       "AppOne_QTab",
                       "AppOneMsgQueue");
```

```
    String doc2send = XMLGen.returnDocument(actionConnection, ordId);
    // sending XMLDoc in the Queue
    try
    {
      if (verbose)
        System.out.println("Doc : " + doc2send);
      if (debugFile)
      {
        BufferedWriter bw = new BufferedWriter(new FileWriter("debug.txt"));
        bw.write("Rows in " + entity);
        bw.write(doc2send);
        bw.flush();
        bw.close();
      }
    }
    catch (Exception ex) {}

    aqw.writeQ(new B2BMessage(MessageHeaders.APP_A,
                              MessageHeaders.APP_B,
                              MessageHeaders.NEW_ORDER,
                              doc2send));
    aqw.flushQ();  // Commit !
  }

  return doc;
}

private Document doRollback() throws Exception
{
  Document doc = null;
  actionConnection.rollback();

  doc = new XMLDocument();
  Element elmt = doc.createElement("operationResult");
  Attr attr = doc.createAttribute("opType");
  attr.setValue("rollback");
  elmt.setAttributeNode(attr);
  doc.appendChild(elmt);
  Node txt = doc.createTextNode("dummy");
  elmt.appendChild(txt);
  txt.setNodeValue("Rollback successfull");

  return doc;
}

private Document doBegin() throws Exception
{
```

```
    Document doc = null;
    actionConnection.setAutoCommit(false);

    doc = new XMLDocument();
    Element elmt = doc.createElement("operationResult");
    Attr attr = doc.createAttribute("opType");
    attr.setValue("begin");
    elmt.setAttributeNode(attr);
    doc.appendChild(elmt);
    Node txt = doc.createTextNode("dummy");
    elmt.appendChild(txt);
    txt.setNodeValue("Begin successfull");

    return doc;
  }

  private static Connection getConnected(String connURL,
                                         String userName,
                                         String password)
  {
    Connection conn = null;
    try
    {
      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      conn = DriverManager.getConnection(connURL, userName, password);
      conn.setAutoCommit(false);
    }
    catch (Exception e)
    {
      System.err.println(e);
      System.exit(1);
    }
    return conn;
  }
}
```

Java の例 2: RetailActionHandler.java のセッション・コンテキストのメンテナンス ー SessionHolder.java

```
// Copyright (c) 2000 Oracle Corporation
package B2BDemo.XSQLActionHandler;
/**
 * Used to maintain the connection context from the XSQL Action Handler.
 * Also closes the connection when servlet expires.
 *
 * @see RetailActionHandler
 */
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class SessionHolder implements HttpSessionBindingListener
{
  private Connection c;
  private java.util.Date d = null;

  public SessionHolder(Connection conn)
  {
    System.out.println("New SessionHandler");
    this.c = conn;
    this.d = new java.util.Date();
  }

  public Connection getConnection()
  {
    return this.c;
  }

  public java.util.Date getOpenDate()
  {
    return this.d;
  }

  public void valueBound(HttpSessionBindingEvent event)
  {
    System.out.println("¥nvalueBound ! " + event.getName() + "¥nat " + (new
          java.util.Date()) + "¥nfor " + event.getSession().getId());
  }

  public void valueUnbound(HttpSessionBindingEvent event)
  {
    System.out.println("¥nvalueUnbound ! " + event.getName() + "¥nat " + (new
                  java.util.Date()) + "¥nfor " + event.getSession().getId());
```

```
    event.getSession().removeValue("XSQLActionHandler.connection");
    if (this.c != null)
    {
      try { this.c.close(); }
      catch (Exception e)
      {
        System.out.println("Problem when closing the connection from " +
                           event.getName() +
                           " for " +
                           event.getSession().getId() +
                           " :¥n" +
                           e);
      }
    }
  }
}
```

手順 4. AQ Broker - トランスフォーマによるサプライヤの形式に従った XML 文書の変換

1. AQ Broker - トランスフォーマ・アプリケーションは、XML 注文書が保留されていることを示すアラートを受信します。
2. 注文の詳細を示す XML 文書は、XML-SQL Utility を使用して生成されています。このドキュメントは、伝播のためにアドバンスト・キューイングを使用して AQ Broker - トランスフォーマに送信されたものです。

   AQ Broker アプリケーションでは、Stylesheet 表に基づいて次の情報が認識されます。

   - 発信元 : リテーラ
   - 送信先 : サプライヤ
   - 内容 : NEW ORDER

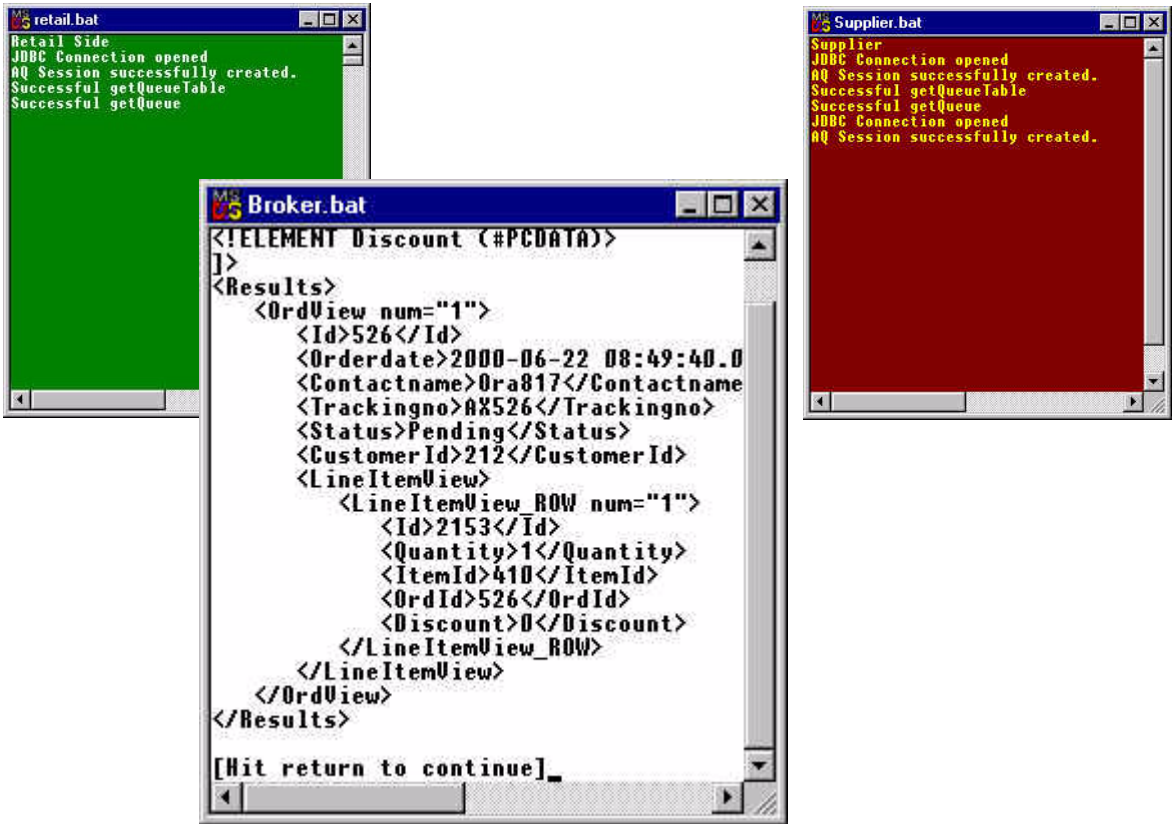
   これらの要素を使用して、Stylesheet 表から適切なスタイルシートが選択されます。XSLT Processor により変換が処理されます。図 8-17 を参照してください。

スクリプト

- MessageBroker. java は BrokerThread.java をコールし、BrokerThread.java は次のスクリプトをコールします。
  - BrokerThread.java は AQReader.java および AQWriter.java をコールします。

    AQReader.java および AQWriter.java では、どちらもメッセージ構造に B2BMessages.java が使用されます。

図 8-17 [AQ Broker]: retailer.bat、broker.bat および supplier.bat のコンソールの表示（1/3）



3. AQ Broker のコンソールで [Enter] キーを押します。

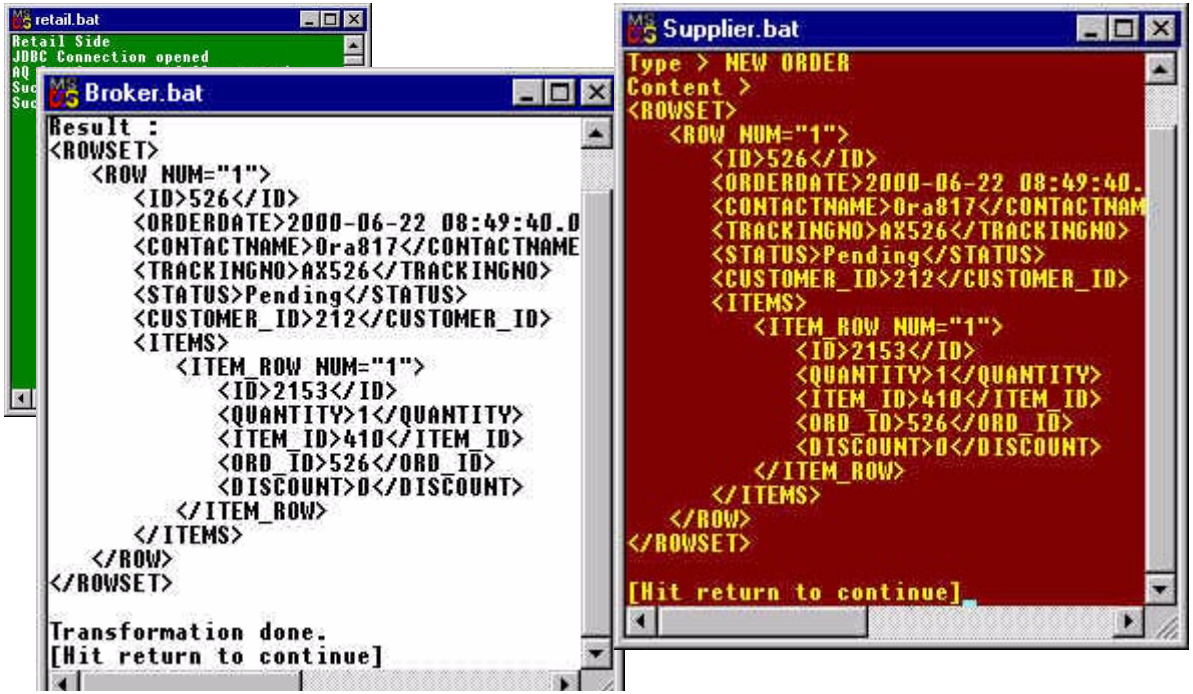
4. AppFrom、AppTo および Op 列の内容に従って、Stylesheets 表内で適切なスタイルシートが検索されます。XSL 変換は、選択されたスタイルシートを使用して進行します。これで、再フォーマット済みの XML 文書をサプライヤに送信する準備が完了しました。

> **注意：** この場合、XML + XSL = XML です。

5. AQ Broker のコンソールでもう一度 [Enter] キーを押します。図 8-18 を参照してください。「broker.bat」画面が変換処理の進行につれて変化します。結果は、XSLT 変換後に取得されます。

AQ Broker - トランスフォーマおよびアドバンスト・キューイング・プロセスを実行するコードのリストは、「AQ Broker - トランスフォーマおよびアドバンスト・キューイングのスクリプト」を参照してください。

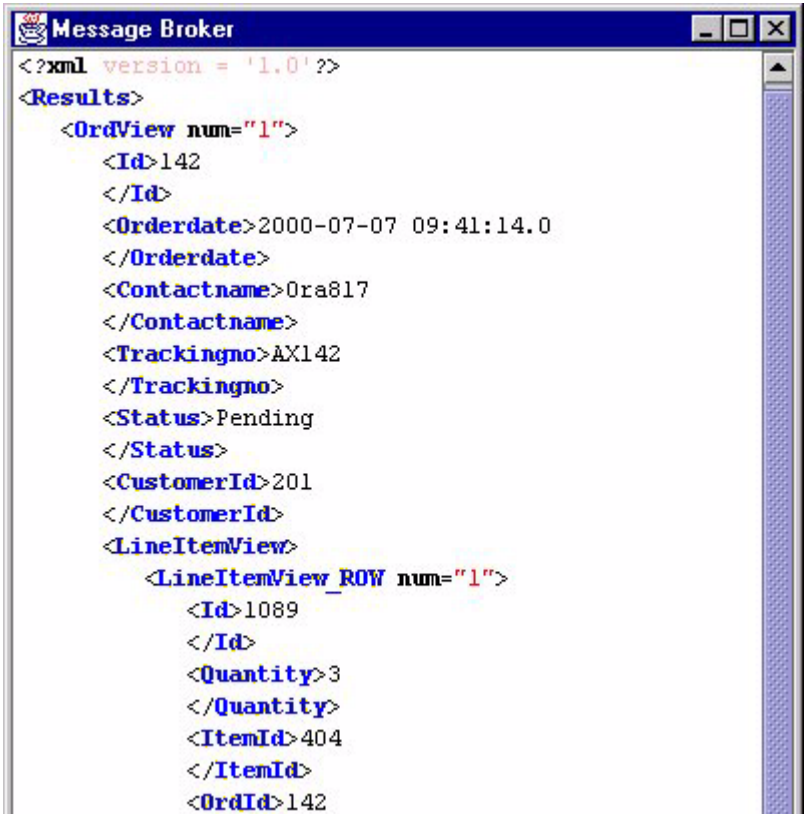
図 8-18 [AQ Broker]: retailer.bat、broker.bat および supplier.bat のコンソールの表示（2/3）



新規に再フォーマットされた XML 文書が、アドバンスト・キューイングを使用してサプライヤに送信されます [WRITE]。

注意： この時点では、どちらのアプリケーションでも同じ XML 文書を処理しているため、AQ Broker とサプライヤの .bat 画面は同じ内容になります。

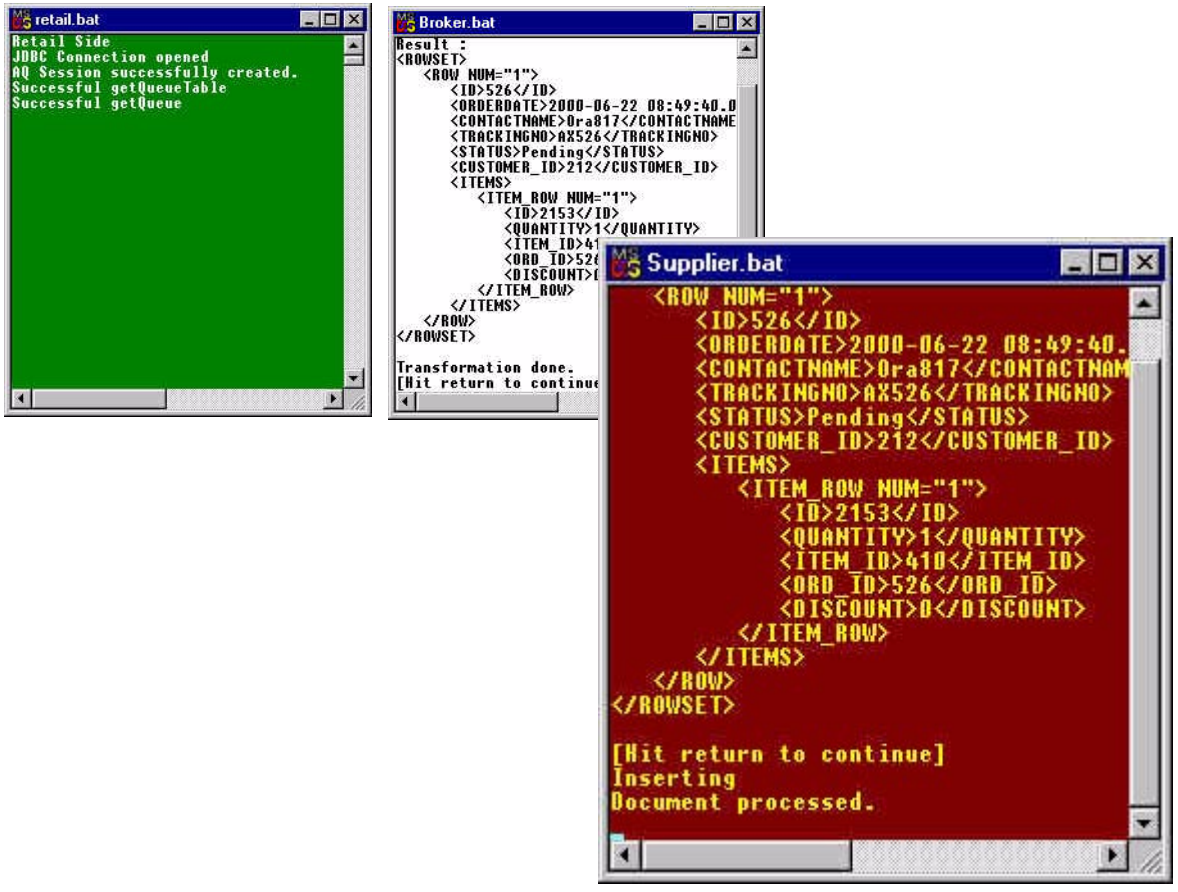
図 8-19 AQ Broker - トランスフォーマからのサンプル XML 文書出力



手順 5. サプライヤ・アプリケーションによる XML 文書の解析およびサプライヤ・データベースへの注文の挿入

1. サプライヤ・アプリケーションが XML 文書を受信します。この XML 文書に含まれているデータを解析する必要があり、このデータはデータベースに挿入されます。
2. サプライヤのコンソールで [Enter] キーを押します。図 8-20 を参照してください。

図 8-20 [AQ Broker]: retailer.bat、broker.bat および supplier.bat のコンソールの表示（3/3）



手順 6a. サプライヤ・アプリケーションからサプライヤに対する注文保留を示すアラート

1. ドキュメントが処理され、データが挿入されます。サプライヤ・アプリケーションの Watcher プログラムは、注文が保留中であることを示す Wake Up メッセージを送信します。図 8-21 を参照してください。
2. 「Supplier Watcher」ダイアログ・ボックスで「OK」をクリックします。図 8-22 を参照してください。

スクリプト

- `SupplierWatcher.java` が `SupplierFramer.java` をコールします。

図 8-21 サプライヤ・アプリケーションからサプライヤに対する注文保留を示すアラート：「Wake Up!」

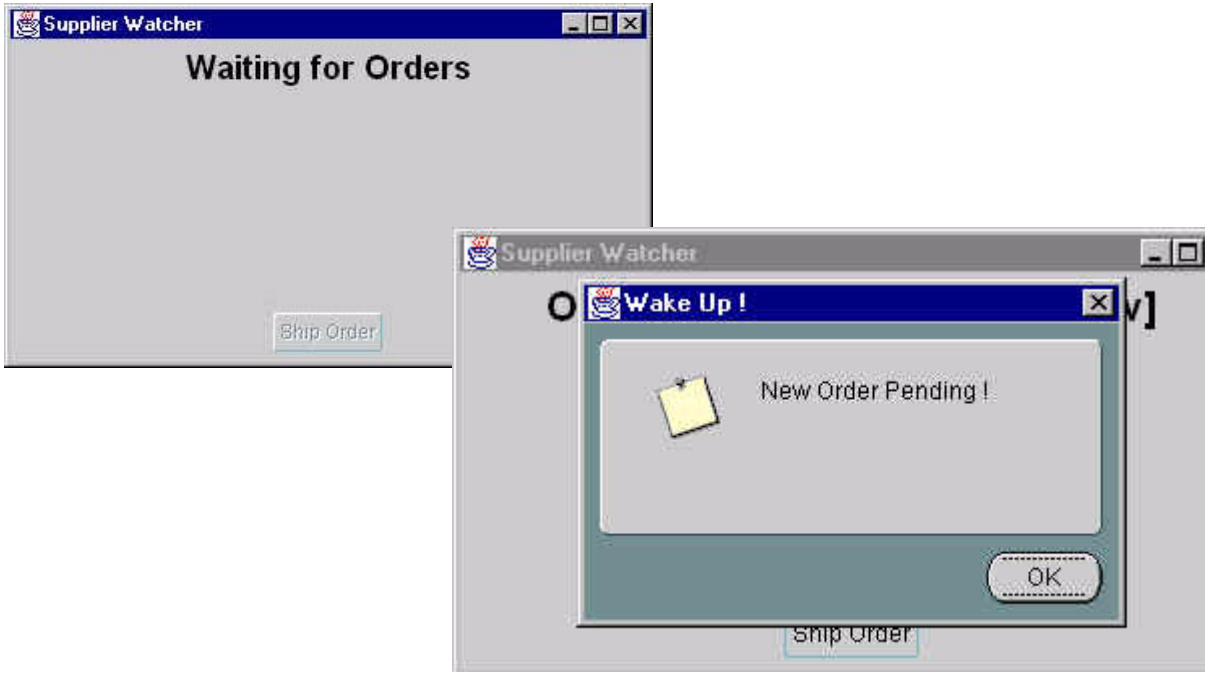
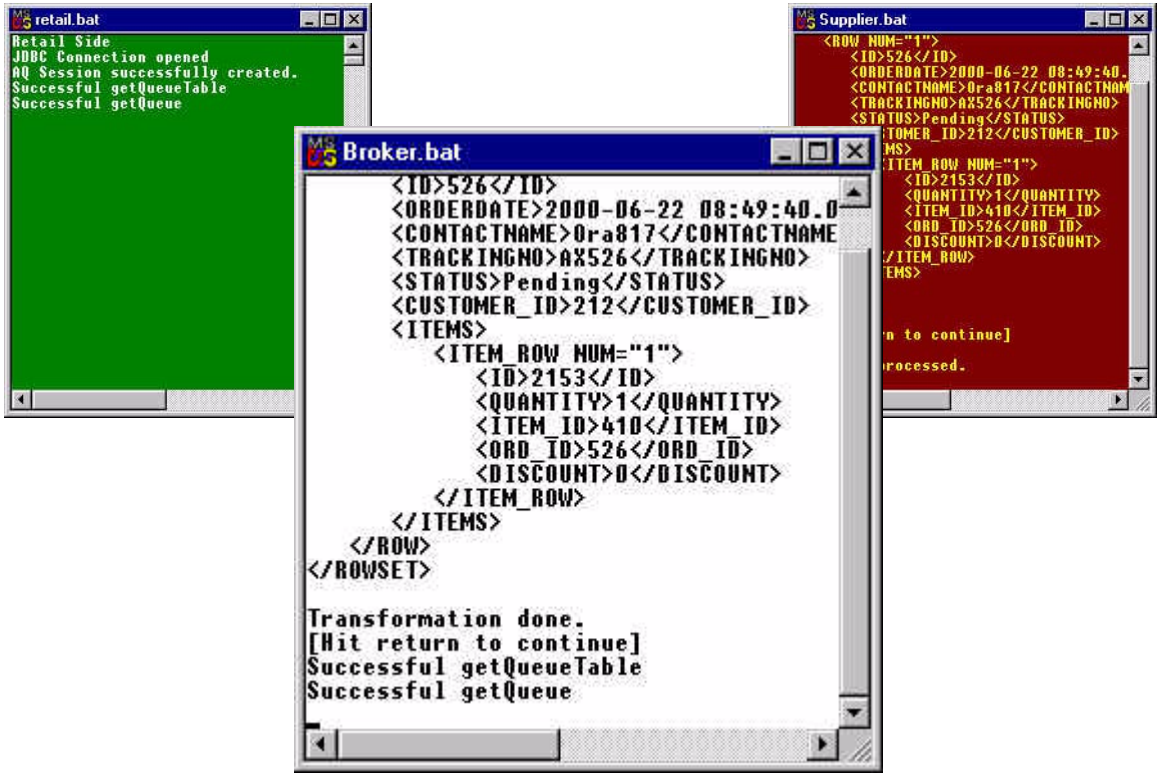


図 8-22 [サプライヤ]: retail.bat、broker.bat および supplier.bat のコンソール : 「Wake Up」に対して「OK」をクリックした後



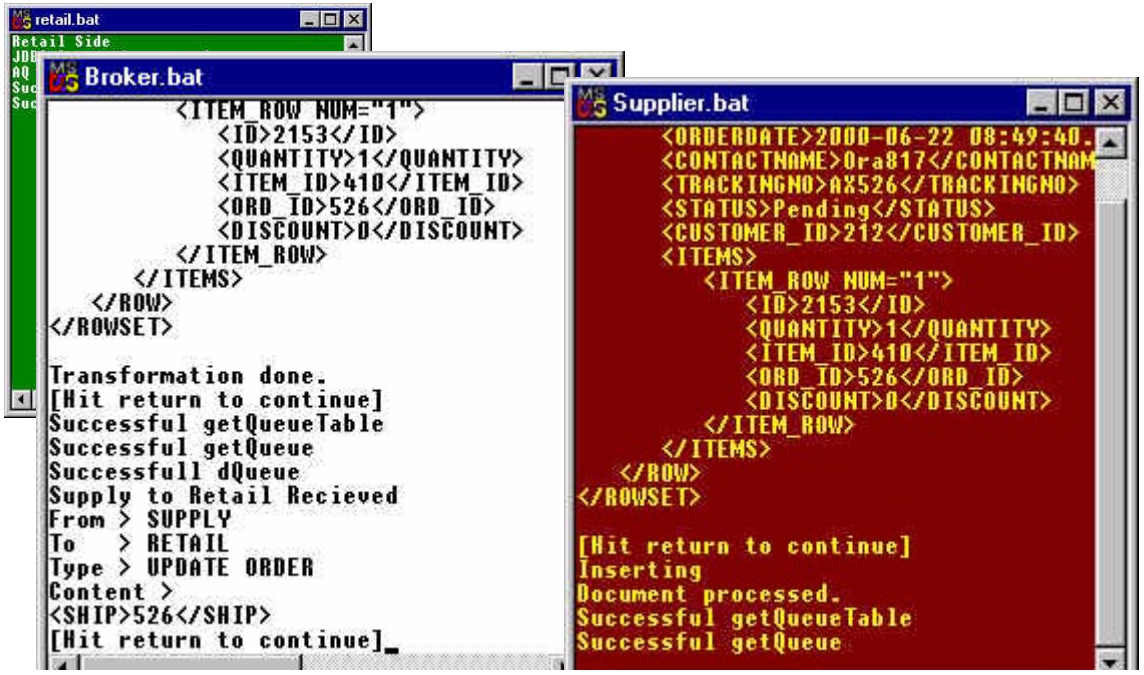
手順 6b. サプライヤによるリテーラへの製品出荷の決定

1. サプライヤが、この注文分の出荷を決定し、ダイアログ・ボックスで「Ship Order」をクリックします。図 8-23 および図 8-24 を参照してください。

スクリプト : 引き続き `SupplierWatcher.java` を使用します。

図 8-23 [サプライヤ]: 注文分の出荷の決定

図 8-24 [サプライヤ]: 「Ship Order」をクリックしたときの retailer.bat、broker.bat および supplier.bat のコンソールの表示



手順 6c. サプライヤ・アプリケーションによる新規 XML メッセージの生成と AQ Broker への送信

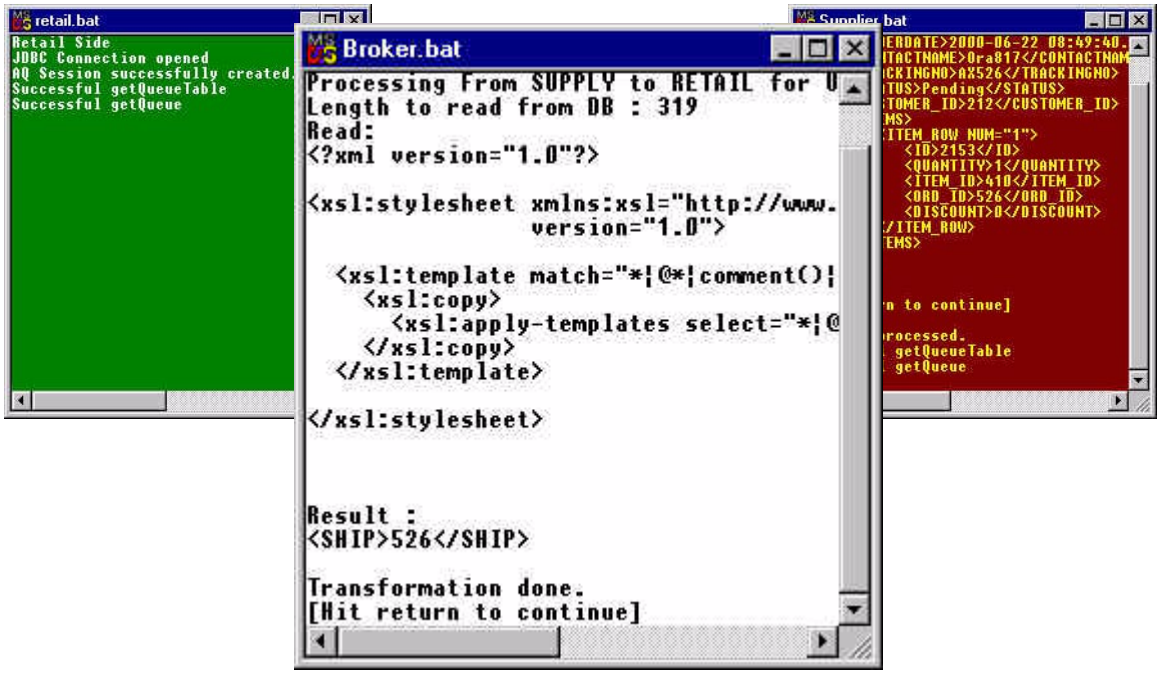
1. 注文を出荷するサプライヤ・アプリケーションが、ブローカへの送信用に新規メッセージを生成します。
2. ブローカのコンソールで [Enter] キーを押します。図 8-25 を参照してください。

図 8-25 [サプライヤ] retailer.bat、broker.bat および supplier.bat のコンソール ― フォームの新規 XML ドキュメント

手順 7. AQ Broker - トランスフォーマによるリテーラの形式への XML 注文書の変換

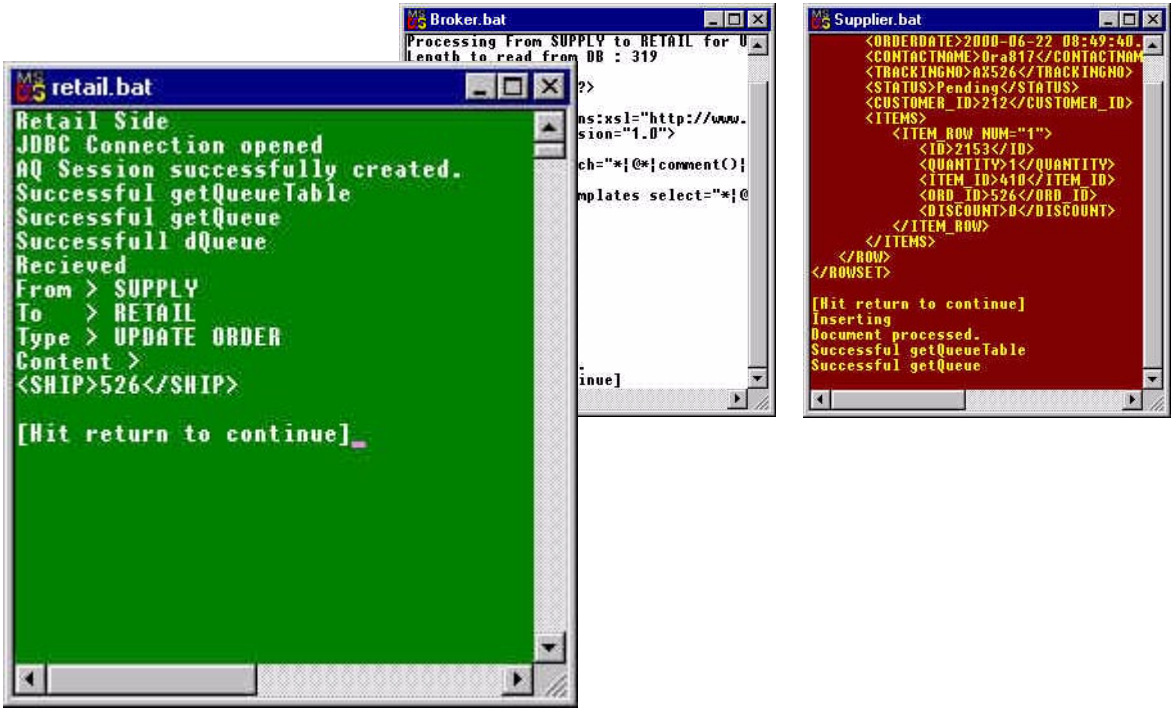
1. 手順 4 と同様に、AQ Broker - トランスフォーマのデータベースからスタイルシートが選択され、XML 注文書に適用されて、再フォーマット済みの XML 文書が生成されます。
2. ブローカのコンソールで [Enter] キーを押します。図 8-26 を参照してください。

図 8-26 [AQ Broker]: retailer.bat、broker.bat および supplier.bat のコンソール ― XML 文書の再フォーマット



3. ドキュメントがリテーラ・アプリケーションに送信されます。
4. リテーラのコンソールで [Enter] キーを押します。図 8-27 を参照してください。これにより、XML 注文書が解析されます。

図 8-27 [AQ Broker]: retailer.bat、broker.bat および supplier.bat のコンソール ー XML メッセージの送信



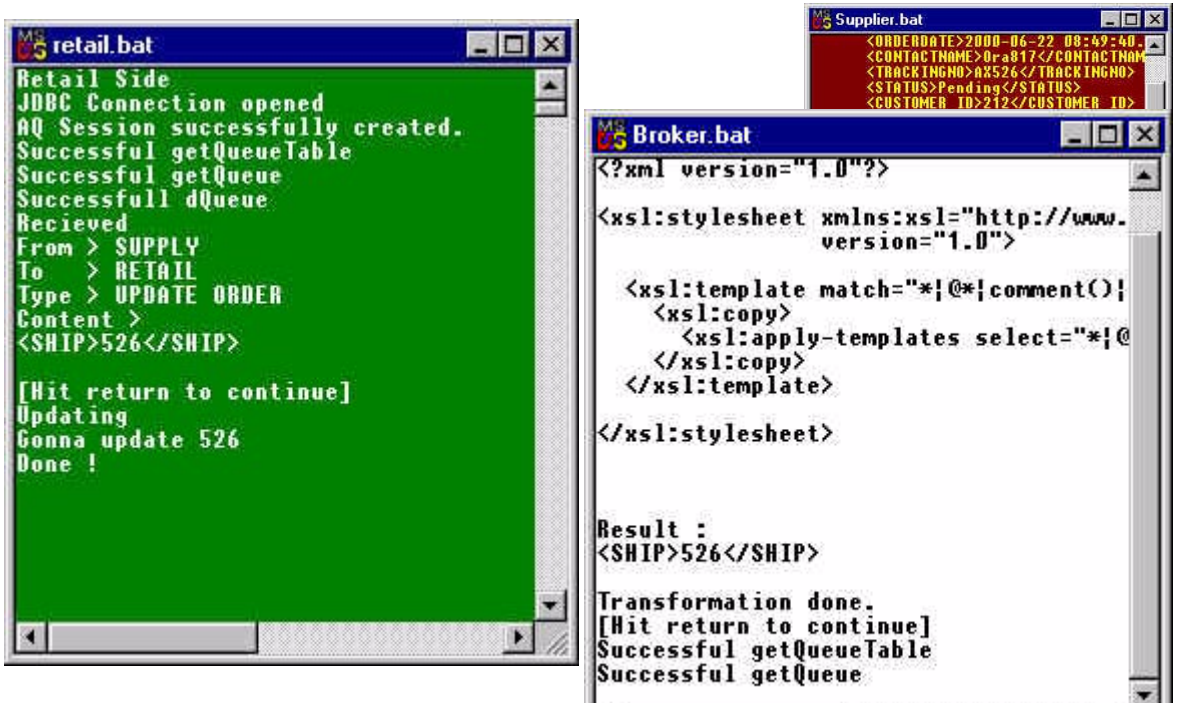
手順 8. リテーラ・アプリケーションによる Ord 表の更新とリテーラに対する新規オーダー・ステータスの表示

1. リテーラ・アプリケーションにより、リテーラ・データベースの「Pending」ステータスが新規の「shipped」オーダー・ステータス情報で更新されます。Ord 表が更新されます。
2. リテーラは、この情報をどのデバイスからでも表示できます。ステータスは「Shipped」と表示されます。図 8-28 を参照してください。

スクリプト

`UpdateMaster.java`。このスクリプトにより、メッセージが受信され、解析されます。

図 8-28 [リテーラ]: retailer.bat、broker.bat および supplier.bat のコンソール ー Shipped ステータスへの更新



これで処理は完了です。

B2B XML アプリケーションの停止

B2B XML アプリケーション（demo）を停止するには、「Java の例 3: stopQ.bat」を実行します。

Java の例 3: stopQ.bat

```
@echo off
@echo stopping all Qs
D:¥jdev31¥java¥bin¥java -mx50m -classpath
"D:¥xml817¥references¥Ora817DevGuide;
D:¥jdev31¥lib¥jdev-rt.zip;
D:¥jdev31¥jdbc¥lib¥oracle8.1.6¥classes111.zip;
D:¥jdev31¥lib¥connectionmanager.zip;
D:¥jdev31¥lib;D:¥jdev31¥lib¥oraclexsql.jar;
D:¥jdev31¥lib¥oraclexmlsql.jar;
D:¥jdev31¥lib¥xmlparserv2_2027.jar;
D:¥jdev31¥jfc¥lib¥swingall.jar;
D:¥jdev31¥jswdk-1.0.1¥lib¥servlet.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi11.jar;
D:¥Ora8i¥rdbms¥jlib¥aqapi.jar;
D:¥XMLWorkshop¥xmlcomp.jar;
D:¥jdev31¥java¥lib¥classes.zip"  B2BDemo.AQUtil.StopAllQueues
```

vieworder.sql を使用したオーダー・ステータスの直接的なチェック

オーダー・ステータスをデータベースから直接チェックするには、次の SQL スクリプトを実行します。

```
set ver off
select O.ID as "Order#",
   O.OrderDate as "Order Date",
   O.Status as "Status"
   From ORD O,
     CUSTOMER C
   Where O.CUSTOMER_ID = C.ID and
     Upper(C.NAME) = Upper('&CustName');
```

Java の例 ー コール順序

Java の例のコール順序を次に示します。各ファイルの .java 拡張子は省略されています。表記法「<---」は「コール」を示します。たとえば、AQReader <----- B2BMessage は、AQReader が B2BMessage をコールすることを示します。

- AQReader <---- B2BMessage
- AQWriter <---- B2BMessage
- UpdateMaster
  - <---- AQReader <----B2BMessage
  - <---- B2BMessage
  - <----- MessageHeaders
  - XMLFrame
- SupplierWatcher
  - <---- SupplierFrame
    - <---- AQReader <---- B2BMessage
    - <----XML2DMLv2 <---- TableInDocument
    - <---- TableInDocument
    - <---- AQWriter <---- B2BMessage
    - <---- B2BMessage
    - <---- MessageHeaders
  - <---- XMLFrame
- MessageBroker
  - <---- AppCste
  - <---- BrokerThread
    - <---- XSLTWrapper
    - <---- AQWriter <---- B2BMessage
    - <---- AQReader <---- B2BMessage
  - <---- AQReader <---- B2BMessage
  - <---- AQWriter <---- B2BMessage
  - <---- XMLFrame（MessageBroker によりコール）
- RetailActionHandler <---- SessionHolder

XSL および XSL 管理スクリプト

「B2B XML アプリケーションの実行 : 手順の詳細」の例のリスト表示が複雑になりすぎないように、XSL の例を別個に示します。

- XSL スタイルシートの例 1: HTML への結果の変換 ー html.xsl
- XSL スタイルシートの例 2: Palm Pilot ブラウザ用の結果の変換 ー pp.xsl
- Java の例 3: スタイルシートの管理 ー GUIInterface.java
- Java の例 4: GUIInterface_AboutBoxPanel.java
- Java の例 5: GUIStylesheet.java

XSL スタイルシートの例 1: HTML への結果の変換 ー html.xsl

```
<?xml version="1.0"?>
<!--
| $Author: olediour@us $
| $Date: 04-May-2000
| xsl for html
| $Revision: 1.1 $
+-->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:output media-type="text/html" method="html" encoding="ISO-8859-1"/>

  <xsl:template match="/">
   <html>
    <head>
      <title>Retail Application</title>
    </head>
    <body>
      <xsl:if test="//pageTitle">
        <h2><xsl:value-of select="//pageTitle"/></h2>
      </xsl:if>
      <xsl:choose>
        <xsl:when test="loginResult">
          <xsl:apply-templates select="loginResult"/>
        </xsl:when>
        <xsl:when test="index">
          <xsl:apply-templates select="index"/>
        </xsl:when>
        <xsl:when test="inventory">
          <xsl:apply-templates select="inventory"/>
        </xsl:when>
```

```
        <xsl:when test="order">
          <xsl:apply-templates select="order"/>
        </xsl:when>
        <xsl:when test="placeOrder">
          <xsl:apply-templates select="placeOrder"/>
        </xsl:when>
        <xsl:otherwise>
           <p align="center">
             <h3>This kind of XML Document cannot be processed...</h3>
           </p>
        </xsl:otherwise>
      </xsl:choose>
    </body>
  </html>
</xsl:template>

<xsl:template match="loginResult">
 <xsl:if test="ROWSET/ROW/unknown">
  <table width="98%">
  <tr>
  <td bgcolor="yellow" align="center">
  <xsl:value-of select="ROWSET/ROW/unknown"/> is not allowed to log in !</td>
  </tr>
  </table>
  </xsl:if>
  <xsl:if test="ROWSET/ROW/NAME">
   <p align="center">
    <h2>Welcome <xsl:value-of select="ROWSET/ROW/NAME"/> !</h2>
   </p>
   <p align="center">
    <a>
    <xsl:attribute name="href">
<xsl:value-of select="nextStep"/>?custId=<xsl:value-of select="ROWSET/ROW/ID"/>
</xsl:attribute>
        Please enter the Mall !
      </a>
    </p>
  </xsl:if>
  <p>
    <a><xsl:attribute name="href"><xsl:value-of
select="returnHome"/></xsl:attribute>Back to Login</a>
  </p>
</xsl:template>

<xsl:template match="index">
  <xsl:for-each select="form">
    <center>
```

```
     <form>
<xsl:attribute name="action"><xsl:value-of select="./@action"/></xsl:attribute>
<xsl:attribute name="method"><xsl:value-of select="./@method"/></xsl:attribute>
     <xsl:if test="./field">
       <table width="98%" border="1">
         <xsl:for-each select="./field">
           <tr>
             <td align="right"><xsl:value-of select="./@prompt"/></td>
             <td>
              <input>
                <xsl:choose>
                <xsl:when test="./@type = 'text'">
                <xsl:attribute name="type">text</xsl:attribute>
                </xsl:when>
                </xsl:choose>
                <xsl:attribute name="name">
                <xsl:value-of select="./@name"/></xsl:attribute>
                </input>
                </td>
                </tr>
                </xsl:for-each>
       </table>
     </xsl:if>
   <xsl:if test="./button">
   <p>
   <xsl:for-each select="./button">
     <input>
       <xsl:choose>
        <xsl:when test="./@type = 'submit'">
         <xsl:attribute name="type">submit</xsl:attribute>
        </xsl:when>
       </xsl:choose>
      <xsl:attribute name="value">
      <xsl:value-of select="./@label"/>
     </xsl:attribute>
    </input>
   </xsl:for-each>
   </p>
   </xsl:if>
</form>
     </center>
   </xsl:for-each>
 </xsl:template>

 <xsl:template match="inventory">
   <h2>This is the Mart content</h2>
   <table>
```

```
      <tr>
        <th>Prod #</th>
        <th>Product</th>
        <th>Price</th>
        <th>Supplied by</th>
      </tr>
      <xsl:for-each select="form/theMart/ROWSET/ROW">
        <tr>
          <td><xsl:value-of select="ID"/></td>
          <td>
            <a>
              <xsl:attribute name="href">
                <xsl:value-of
select="../../../../form/@action"/>?custId=<xsl:value-of
select="../../../../form/hiddenFields/custId"/>&amp;prodId=<xsl:value-of
select="ID"/>
              </xsl:attribute>
              <xsl:value-of select="DESCRIPTION"/>
            </a>
          </td>
          <td><xsl:value-of select="PRICE"/></td>
          <td><xsl:value-of select="NAME"/></td>
        </tr>
      </xsl:for-each>
    </table>
    <p>
      <a><xsl:attribute name="href"><xsl:value-of
select="returnHome"/></xsl:attribute>Back to Login</a>
    </p>
  </xsl:template>

  <xsl:template match="order">
    <center>
      <h2>Thank you <xsl:value-of select="CUST/NAME"/> for shopping with us !</h2>
      <hr/>
      <h2>Please enter the quantity</h2>
      <form action="placeOrder.xsql" method="post">
        <input type="hidden" name="prodId">
          <xsl:attribute name="value">
          <xsl:value-of select="PROD/ID"/>
        </xsl:attribute>
        </input>
        <input type="hidden" name="custId">
          <xsl:attribute name="value">
          <xsl:value-of select="CUST/ID"/></xsl:attribute>
        </input>
        <table border="1">
```

```
          <tr>
            <td colspan="2"><xsl:value-of select="PROD/DESCRIPTION"/>
                at $<xsl:value-of select="PROD/PRICE"/> each
                supplied by <xsl:value-of select="PROD/NAME"/></td>
          </tr>
          <tr>
            <td align="right">Quantity</td>
            <td><input type="text" name="qty"/></td>
          </tr>
        </table>
        <p><input type="submit" value="Place Order"/></p>
      </form>
    </center>
    <p>
      <a><xsl:attribute name="href">
      <xsl:value-of select="returnHome"/>
     </xsl:attribute>Back to Login</a>
    </p>
  </xsl:template>

  <xsl:template match="placeOrder">
    <xsl:if test="operationResult">
      <table width="98%">
        <tr><td align="center">
        <font color="navy">
        <xsl:value-of select="operationResult/text()"/>
        </font></td></tr>
        <tr>
          <td align="center">
            <xsl:for-each select="operationResult/nextStep">
              <form method="post">
                <xsl:attribute name="action"><xsl:value-of
select="./@Action"/></xsl:attribute>
                <xsl:if test="prmList">
                  <xsl:for-each select="prmList/prm">
                    <input type="hidden">
                      <xsl:attribute name="name"><xsl:value-of
select="./@name"/></xsl:attribute>
                      <xsl:attribute name="value"><xsl:value-of
select="./@value"/></xsl:attribute>
                    </input>
                  </xsl:for-each>
                </xsl:if>
                <input type="submit">
                  <xsl:attribute name="value"><xsl:value-of
select="./@Label"/></xsl:attribute>
                </input>
```

```
              </form>
            </xsl:for-each>
          </td>
        </tr>
      </table>
    </xsl:if>
    <xsl:if test="xsql-error">
      <table width="98%">
        <tr><td><xsl:value-of select="xsql-error/@action"/></td></tr>
        <tr><td><xsl:value-of select="xsql-error/statement"/></td></tr>
        <tr><td><xsl:value-of select="xsql-error/message"/></td></tr>
      </table>
    </xsl:if>
    <xsl:if test="operationProblem">
      <table width="98%">
        <tr>
          <td colspan="2" align="center">
            <font color="red"><b><xsl:value-of
select="operationProblem/text()"/></b></font>
          </td>
        </tr>
        <xsl:for-each select="operationProblem/parameters/*">
          <tr>
            <td align="right"><xsl:value-of select="name()"/></td>
            <td align="left"><xsl:value-of select="."/></td>
          </tr>
        </xsl:for-each>
      </table>
    </xsl:if>
    <xsl:if test="bottomLinks">
      <xsl:choose>
        <xsl:when test="operationResult">
        </xsl:when>
        <xsl:otherwise>
          <p align="center">
            <xsl:for-each select="bottomLinks/aLink">
              [<a><xsl:attribute name="href"><xsl:value-of
select="./@href"/></xsl:attribute><xsl:value-of select="."/></a>]
            </xsl:for-each>
          </p>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:if>
    <xsl:choose>
      <xsl:when test="operationResult/nextStep">
      </xsl:when>
      <xsl:otherwise>
```

```
        <xsl:if test="returnHome">
          <p>
            <a><xsl:attribute name="href"><xsl:value-of
select="returnHome"/></xsl:attribute>Back to Login</a>
          </p>
        </xsl:if>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

</xsl:stylesheet>
```

XSL スタイルシートの例 2: Palm Pilot ブラウザ用の結果の変換 ー pp.xsl

```
<?xml version="1.0"?>
<!--
| $Author: olediour@us $
| $Date: 04-May-2000
| xsl for html (Palm Pilot, HandWeb browser)
| $Revision: 1.1 $
+-->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:output media-type="text/html" method="html" encoding="ISO-8859-1"/>

  <xsl:template match="/">
    <html>
      <head>
        <title>Retail Application</title>
      </head>
      <body>
        <xsl:if test="//pageTitle">
          <h2><xsl:value-of select="//pageTitle"/></h2>
        </xsl:if>
        <xsl:choose>
          <xsl:when test="loginResult">
            <xsl:apply-templates select="loginResult"/>
          </xsl:when>
          <xsl:when test="index">
            <xsl:apply-templates select="index"/>
          </xsl:when>
          <xsl:when test="inventory">
            <xsl:apply-templates select="inventory"/>
          </xsl:when>
          <xsl:when test="order">
```

```
            <xsl:apply-templates select="order"/>
          </xsl:when>
          <xsl:when test="placeOrder">
            <xsl:apply-templates select="placeOrder"/>
          </xsl:when>
          <xsl:otherwise>
             <p align="center">
               <h3>This kind of XML Document cannot be processed...</h3>
             </p>
          </xsl:otherwise>
        </xsl:choose>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="loginResult">
    <xsl:if test="ROWSET/ROW/unknown">
      <table width="98%">
        <tr><td bgcolor="yellow" align="center"><xsl:value-of
select="ROWSET/ROW/unknown"/> is not allowed to log in !</td></tr>
      </table>
    </xsl:if>
    <xsl:if test="ROWSET/ROW/NAME">
      <p align="center">
        <h2>Welcome <xsl:value-of select="ROWSET/ROW/NAME"/> !</h2>
      </p>
      <p align="center">
        <a>
          <xsl:attribute name="href"><xsl:value-of
select="nextStep"/>?custId=<xsl:value-of select="ROWSET/ROW/ID"/></xsl:attribute>
          Please enter the Mall !
        </a>
      </p>
    </xsl:if>
    <p>
      <a><xsl:attribute name="href"><xsl:value-of
select="returnHome"/></xsl:attribute>Back to Login</a>
    </p>
  </xsl:template>

  <xsl:template match="index">
    <xsl:for-each select="form">
      <center>
        <form>
          <xsl:attribute name="action"><xsl:value-of
select="./@action"/></xsl:attribute>
          <xsl:attribute name="method"><xsl:value-of
```

```
select="./@method"/></xsl:attribute>
          <xsl:if test="./field">
            <table width="98%" border="1">
              <xsl:for-each select="./field">
                <tr>
                  <td align="right"><xsl:value-of select="./@prompt"/></td>
                  <td>
                    <input>
                      <xsl:choose>
                        <xsl:when test="./@type = 'text'">
                          <xsl:attribute name="type">text</xsl:attribute>
                       </xsl:when>
                       </xsl:choose>
                      <xsl:attribute name="name"><xsl:value-of
select="./@name"/></xsl:attribute>
                    </input>
                  </td>
                </tr>
              </xsl:for-each>
            </table>
          </xsl:if>
          <xsl:if test="./button">
            <p>
              <xsl:for-each select="./button">
                <input>
                  <xsl:choose>
                    <xsl:when test="./@type = 'submit'">
                      <xsl:attribute name="type">submit</xsl:attribute>
                    </xsl:when>
                  </xsl:choose>
                  <xsl:attribute name="value"><xsl:value-of
select="./@label"/></xsl:attribute>
                </input>
              </xsl:for-each>
            </p>
          </xsl:if>
        </form>
      </center>
    </xsl:for-each>
  </xsl:template>

  <xsl:template match="inventory">
    <h2>This is the Mart content</h2>
    <xsl:for-each select="form/theMart/ROWSET/ROW">
      <xsl:value-of select="ID"/>
      <xsl:text> </xsl:text>
      <form method="post">
```

```
        <xsl:attribute name="action">
          <xsl:value-of select="../../../../form/@action"/>
        </xsl:attribute>
        <input type="hidden" name="custId">
          <xsl:attribute name="value"><xsl:value-of
select="../../../../form/hiddenFields/custId"/></xsl:attribute>
        </input>
        <input type="hidden" name="prodId">
          <xsl:attribute name="value"><xsl:value-of select="ID"/></xsl:attribute>
        </input>
        <input type="submit">
          <xsl:attribute name="value"><xsl:value-of
select="DESCRIPTION"/></xsl:attribute>
        </input>
      </form>
      <xsl:text> @ $</xsl:text><xsl:value-of select="PRICE"/><xsl:text>
each</xsl:text>
      <xsl:text> Supplied by </xsl:text><xsl:value-of select="NAME"/>
      <br/>
    </xsl:for-each>
    <p>
      <a><xsl:attribute name="href"><xsl:value-of
select="returnHome"/></xsl:attribute>Back to Login</a>
    </p>
  </xsl:template>

  <xsl:template match="order">
    <center>
      <h2>Thank you <xsl:value-of select="CUST/NAME"/> for shopping with us !</h2>
      <hr/>
      <h2>Please enter the quantity</h2>
      <form action="placeOrder.xsql" method="post">
        <input type="hidden" name="prodId">
          <xsl:attribute name="value"><xsl:value-of
select="PROD/ID"/></xsl:attribute>
        </input>
        <input type="hidden" name="custId">
          <xsl:attribute name="value"><xsl:value-of
select="CUST/ID"/></xsl:attribute>
        </input>
        <p>
          <xsl:value-of select="PROD/DESCRIPTION"/>
                at $<xsl:value-of select="PROD/PRICE"/> each
                supplied by <xsl:value-of select="PROD/NAME"/>
          <br/>
            Quantity :
          <br/>
```

```
          <input type="text" name="qty"/>
        </p>
        <p><input type="submit" value="Place Order"/></p>
      </form>
    </center>
    <p>
      <a><xsl:attribute name="href"><xsl:value-of
select="returnHome"/></xsl:attribute>Back to Login</a>
    </p>
  </xsl:template>

  <xsl:template match="placeOrder">
    <xsl:if test="operationResult">
      <center>
        <xsl:value-of select="operationResult/text()"/>
        <br/>
        <xsl:for-each select="operationResult/nextStep">
          <form method="post">
            <xsl:attribute name="action"><xsl:value-of
select="./@Action"/></xsl:attribute>
            <xsl:if test="prmList">
              <xsl:for-each select="prmList/prm">
                <input type="hidden">
                  <xsl:attribute name="name"><xsl:value-of
select="./@name"/></xsl:attribute>
                  <xsl:attribute name="value"><xsl:value-of
select="./@value"/></xsl:attribute>
                </input>
              </xsl:for-each>
            </xsl:if>
            <input type="submit">
              <xsl:attribute name="value"><xsl:value-of
select="./@Label"/></xsl:attribute>
            </input>
          </form>
        </xsl:for-each>
      </center>
    </xsl:if>
    <xsl:if test="operationProblem">
      <table width="98%">
        <tr><td align="center"><font color="red"><xsl:value-of
select="operationProblem"/></font></td></tr>
      </table>
    </xsl:if>
    <xsl:if test="bottomLinks">
      <xsl:choose>
        <xsl:when test="operationResult">
```

```
        </xsl:when>
        <xsl:otherwise>
          <p align="center">
            <xsl:for-each select="bottomLinks/aLink">
              [<a><xsl:attribute name="href"><xsl:value-of
select="./@href"/></xsl:attribute><xsl:value-of select="."/></a>]
            </xsl:for-each>
          </p>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:if>
    <xsl:choose>
      <xsl:when test="operationResult/nextStep">
      </xsl:when>
      <xsl:otherwise>
        <xsl:if test="returnHome">
          <p>
            <a><xsl:attribute name="href"><xsl:value-of
select="returnHome"/></xsl:attribute>Back to Login</a>
          </p>
        </xsl:if>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

</xsl:stylesheet>
```

Java の例 3: スタイルシートの管理 ― GUIInterface.java

次のスクリプトでは、B2B XML アプリケーションに使用する GUI とスタイルシートを作成および管理します。

```
package B2BDemo.StyleSheetUtil;
/**
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

import java.sql.*;
import java.util.*;
// needed for new CLOB and BLOB classes
import oracle.sql.*;
import oracle.jdbc.driver.*;
import java.beans.*;
```

```
import javax.swing.event.*;

import B2BDemo.*;
import B2BDemo.XMLUtil.*;

public class GUInterface extends JFrame
{
  private boolean lite = false;  // Use O8iLite
  private boolean inserting = false;

  private final static int UPDATE = 1;
  private final static int INSERT = 2;

  private final static int ENTER_QUERY = 1;
  private final static int EXEC_QUERY  = 2;

  int queryState = ENTER_QUERY;

  String sqlStmt  = "Select APPFROM, " +
                    "       APPTO, " +
                    "       OP, " +
                    "       XSL " +
                    "From styleSheets";

  private static String connURL  = AppCste.AQDBUrl;
  private static String userName = AppCste.AQuser;
  private static String password = AppCste.AQpswd;
  private Connection conn = null;

  private Vector recVect = null;
  int currRec = 0;
  XslRecord thisRecord = null;

  BorderLayout borderLayout1 = new BorderLayout();
  JPanel jPanel1 = new JPanel();
  JMenuBar menuBar1 = new JMenuBar();
  JMenu menuFile = new JMenu();
  JMenuItem menuFileExit = new JMenuItem();
  JMenu menuHelp = new JMenu();
  JMenuItem menuHelpAbout = new JMenuItem();
  JLabel statusBar = new JLabel();
  JToolBar toolBar = new JToolBar();
  JButton buttonOpen = new JButton();
  JButton buttonClose = new JButton();
  JButton buttonHelp = new JButton();
  ImageIcon imageOpen;
  ImageIcon imageClose;
```

```
ImageIcon imageHelp;
JPanel jPanel2 = new JPanel();
BorderLayout borderLayout2 = new BorderLayout();
JButton firstButton = new JButton();
JPanel jPanel3 = new JPanel();
JPanel jPanel4 = new JPanel();
BorderLayout borderLayout3 = new BorderLayout();
BorderLayout borderLayout4 = new BorderLayout();
JPanel jPanel5 = new JPanel();
JTextField fromAppValue = new JTextField();
JLabel fromApp = new JLabel();
JPanel jPanel6 = new JPanel();
BorderLayout borderLayout5 = new BorderLayout();
JLabel jLabel2 = new JLabel();
JScrollPane jScrollPane1 = new JScrollPane();
JTextArea XSLStyleSheet = new JTextArea();
JButton previousButton = new JButton();
JButton nextButton = new JButton();
JButton lastButton = new JButton();
JButton validateButton = new JButton();
GridLayout gridLayout1 = new GridLayout();
JLabel toApp = new JLabel();
JTextField toAppValue = new JTextField();
JLabel operationLabel = new JLabel();
JTextField opValue = new JTextField();
JButton newButton = new JButton();
JButton deleteButton = new JButton();
JButton queryButton = new JButton();

public GUInterface()
{
  super();
  try
  {
    jbInit();
    buttonOpen.setIcon(imageOpen);
    buttonClose.setIcon(imageClose);
    buttonHelp.setIcon(imageHelp);
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
}

private void getConnected() throws Exception
{
```

```
    try
    {
      if (lite)
      {
        Class.forName("oracle.lite.poljdbc.POLJDBCDriver");
        conn = DriverManager.getConnection("jdbc:Polite:POLite", "system",
"manager");
      }
      else
      {
        Class.forName ("oracle.jdbc.driver.OracleDriver");
        conn = DriverManager.getConnection (connURL, userName, password);
      }
    }
    catch (Exception e)
    {
      System.err.println("Get connected failed : " + e);
      throw e;
    }
  }

  private void jbInit() throws Exception
  {
    if (conn == null)
    {
      try { getConnected(); }
      catch (Exception e)
      {
        JOptionPane.showMessageDialog(null, e.toString(),
                                      "Connection",
                                      JOptionPane.ERROR_MESSAGE);
        System.exit(1);
      }
    }
    imageOpen = new ImageIcon(GUInterface.class.getResource("openfile.gif"));
    imageClose = new ImageIcon(GUInterface.class.getResource("closefile.gif"));
    imageHelp = new ImageIcon(GUInterface.class.getResource("help.gif"));
    this.setTitle("Style Sheets Management");
    this.getContentPane().setLayout(borderLayout1);
    this.setSize(new Dimension(511, 526));
    jPanel1.setLayout(borderLayout2);
    menuFile.setText("File");
    menuFileExit.setText("Exit");
    menuFileExit.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        fileExit_ActionPerformed(e);
      }
```

```
});
menuHelp.setText("Help");
menuHelpAbout.setText("About");
menuHelpAbout.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    helpAbout_ActionPerformed(e);
  }
});
statusBar.setText("Initializing...");
buttonOpen.setToolTipText("Open File");
buttonClose.setToolTipText("Validate modifications");
buttonHelp.setToolTipText("About Style Sheet Manager");
firstButton.setText("<<");
jPanel5.setLayout(gridLayout1);
fromApp.setText("From Application :");
fromApp.setHorizontalAlignment(SwingConstants.RIGHT);
jLabel2.setText("XSL Style Sheet");
previousButton.setText("<");
nextButton.setText(">");
lastButton.setText(">>");
validateButton.setText("Validate");
gridLayout1.setRows(4);
toApp.setText("To Application : ");
toApp.setHorizontalAlignment(SwingConstants.RIGHT);
operationLabel.setText("Operation : ");
operationLabel.setHorizontalAlignment(SwingConstants.RIGHT);
jPanel6.setLayout(borderLayout5);
jPanel4.setLayout(borderLayout4);
jPanel3.setLayout(borderLayout3);
menuFile.add(menuFileExit);
menuBar1.add(menuFile);
menuHelp.add(menuHelpAbout);
menuBar1.add(menuHelp);
this.setJMenuBar(menuBar1);
this.getContentPane().add(statusBar, BorderLayout.SOUTH);
toolBar.add(buttonOpen);
toolBar.add(buttonClose);
toolBar.add(buttonHelp);
this.getContentPane().add(toolBar, BorderLayout.NORTH);
this.getContentPane().add(jPanel1, BorderLayout.CENTER);
jPanel1.add(jPanel2, BorderLayout.SOUTH);
jPanel2.add(queryButton, null);
jPanel2.add(newButton, null);
jPanel2.add(firstButton, null);
jPanel2.add(previousButton, null);
jPanel2.add(nextButton, null);
jPanel2.add(lastButton, null);
```

```
jPanel2.add(validateButton, null);
jPanel2.add(deleteButton, null);
jPanel1.add(jPanel3, BorderLayout.CENTER);
jPanel3.add(jPanel4, BorderLayout.NORTH);
jPanel3.add(jPanel5, BorderLayout.SOUTH);
jPanel5.add(fromApp, null);
jPanel5.add(fromAppValue, null);
jPanel5.add(toApp, null);
jPanel5.add(toAppValue, null);
jPanel5.add(operationLabel, null);
jPanel5.add(opValue, null);
jPanel3.add(jPanel6, BorderLayout.CENTER);
jPanel6.add(jLabel2, BorderLayout.NORTH);
jPanel6.add(jScrollPane1, BorderLayout.CENTER);
jScrollPane1.getViewport().add(XSLStyleSheet, null);

//
statusBar.setText("Connected...");
// Building Vector of record.
queryButton.setText("Enter Query");
queryButton.setActionCommand("query");
queryButton.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    queryButton_actionPerformed(e);
  }
});
buttonClose.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    buttonClose_actionPerformed(e);
  }
});
deleteButton.setText("Delete");
deleteButton.setToolTipText("Delete the current record");
deleteButton.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    deleteButton_actionPerformed(e);
  }
});
newButton.setText("New");
newButton.setToolTipText("Create a new record");
newButton.addActionListener(new java.awt.event.ActionListener()
```

```
{
  public void actionPerformed(ActionEvent e)
  {
    newButton_actionPerformed(e);
  }
});
validateButton.setToolTipText("Validate your modifications");
opValue.setEditable(false);
toAppValue.setEditable(false);
fromAppValue.setEditable(false);
validateButton.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    validateButton_actionPerformed(e);
  }
});
lastButton.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    lastButton_actionPerformed(e);
  }
});
firstButton.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    firstButton_actionPerformed(e);
  }
});
previousButton.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    previousButton_actionPerformed(e);
  }
});
nextButton.addActionListener(new java.awt.event.ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    nextButton_actionPerformed(e);
  }
});
lastButton.setActionCommand("last");
lastButton.setToolTipText("Last record");
```

```
  nextButton.setActionCommand("next");
  nextButton.setToolTipText("Next record");
  previousButton.setActionCommand("previous");
  previousButton.setToolTipText("Previous record");
  firstButton.setActionCommand("first");
  firstButton.setToolTipText("First record");

  // Execute query and build vector
  executeQuery(sqlStmt);

  updateStatusBar();
}

void executeQuery(String theSqlStmt)
{
  recVect = new Vector();
  try
  {
    Statement stmt = conn.createStatement();
    ResultSet rSet = stmt.executeQuery(theSqlStmt);
    CLOB clob = null;
    while (rSet.next())
    {
      clob = ((OracleResultSet)rSet).getCLOB(4);
      String strLob = dumpClob(conn, clob);
      XslRecord xslRecord = new XslRecord(rSet.getString(1),
                                          rSet.getString(2),
                                          rSet.getString(3),
                                          strLob);
      recVect.addElement(xslRecord);
    }
    rSet.close();
    stmt.close();
    // Populate form with first record
    firstButton.setEnabled(false);
    previousButton.setEnabled(false);
    nextButton.setEnabled(false);
    lastButton.setEnabled(false);
    if (recVect.size() > 0)
    {
      currRec = 1;
      displayRecord(currRec);
    }
    if (recVect.size() > 1)
    {
      nextButton.setEnabled(true);
      lastButton.setEnabled(true);
```

```
    }
  }
  catch (Exception e)
  {
    JOptionPane.showMessageDialog(null, e.toString(),
                                  "Executing request",
                                  JOptionPane.ERROR_MESSAGE);
    System.exit(1);
  }
}

void displayRecord(int rnk)
{
  XslRecord xslRecord = (XslRecord)recVect.elementAt(rnk-1);
  thisRecord = new XslRecord(xslRecord.FROM,
                             xslRecord.TO,
                             xslRecord.TASK,
                             xslRecord.XSL);
  XSLStyleSheet.setText(xslRecord.XSL);
  fromAppValue.setText(xslRecord.FROM);
  toAppValue.setText(xslRecord.TO);
  opValue.setText(xslRecord.TASK);

  XSLStyleSheet.requestFocus();
  XSLStyleSheet.setCaretPosition(0);

  // Buttons
  firstButton.setEnabled(false);
  previousButton.setEnabled(false);
  nextButton.setEnabled(false);
  lastButton.setEnabled(false);
  if (rnk > 1)
  {
    firstButton.setEnabled(true);
    previousButton.setEnabled(true);
  }
  if (rnk < recVect.size())
  {
    nextButton.setEnabled(true);
    lastButton.setEnabled(true);
  }
}

void updateStatusBar()
{
  statusBar.setText("Ready for " + recVect.size() + " records");
}
```

```
  private static String dumpClob(Connection conn, CLOB clob) throws Exception
  {
    String returnString = "";

    OracleCallableStatement cStmt1 = (OracleCallableStatement) conn.prepareCall
("begin ? := dbms_lob.getLength (?); end;");
    OracleCallableStatement cStmt2 = (OracleCallableStatement) conn.prepareCall
("begin dbms_lob.read (?, ?, ?, ?); end;");

    cStmt1.registerOutParameter (1, Types.NUMERIC);
    cStmt1.setCLOB (2, clob);
    cStmt1.execute ();

    long length = cStmt1.getLong (1);
    long i = 0;
    int chunk = 80;

    while (i < length)
    {
      cStmt2.setCLOB (1, clob);
      cStmt2.setLong (2, chunk);
      cStmt2.registerOutParameter (2, Types.NUMERIC);
      cStmt2.setLong (3, i + 1);
      cStmt2.registerOutParameter (4, Types.VARCHAR);
      cStmt2.execute ();

      long read_this_time = cStmt2.getLong (2);
      returnString += cStmt2.getString (4);
   // System.out.print ("Read " + read_this_time + " chars: ");
   // System.out.println (string_this_time);
      i += read_this_time;
    }
    cStmt1.close ();
    cStmt2.close ();
    return returnString;
  }

  static void fillClob (Connection conn, CLOB clob, String str) throws SQLException
  {
    OracleCallableStatement cStmt =
      (OracleCallableStatement) conn.prepareCall ("begin dbms_lob.write (?, ?, ?,
?); end;");

    int i = 0;
    int chunk = 80;
    int length = str.length();
```

```
      long c, ii;

      System.out.println("Length: " + length + "¥n" + str);
      while (i < length)
      {
        cStmt.setClob (1, clob);
        c = chunk;
        cStmt.setLong (2, c);
        ii = i + 1;
        cStmt.setLong (3, ii);
        cStmt.setString (4, str.substring(i, i + chunk));
        cStmt.execute ();
        i += chunk;
        if (length - i < chunk)
          chunk = length - i;
      }
      cStmt.close ();
    }

    void fileExit_ActionPerformed(ActionEvent e)
    {
      if (conn != null)
      {
        try { conn.close(); } catch (Exception ex) {}
      }
      System.exit(0);
    }

    void helpAbout_ActionPerformed(ActionEvent e)
    {
      JOptionPane.showMessageDialog(this, new GUInterface_AboutBoxPanel1(), "About",
JOptionPane.PLAIN_MESSAGE);
    }

    void nextButton_actionPerformed(ActionEvent e)
    {
      checkRecordChange();
      currRec++;
      displayRecord(currRec);
    }

    void previousButton_actionPerformed(ActionEvent e)
    {
      checkRecordChange();
      currRec--;
      displayRecord(currRec);
    }
```

```
void firstButton_actionPerformed(ActionEvent e)
{
  checkRecordChange();
  currRec = 1;
  displayRecord(currRec);
}

void lastButton_actionPerformed(ActionEvent e)
{
  checkRecordChange();
  currRec = recVect.size();
  displayRecord(currRec);
}

void validateButton_actionPerformed(ActionEvent e)
{
  validateRec();
}

void validateRec()
{
  thisRecord = new XslRecord(fromAppValue.getText(),
                             toAppValue.getText(),
                             opValue.getText(),
                             XSLStyleSheet.getText());
  if (saveChanges(thisRecord, (inserting?INSERT:UPDATE)))
    JOptionPane.showMessageDialog(null, "All right!");
}

void deleteRec()
{
  thisRecord = new XslRecord(fromAppValue.getText(),
                             toAppValue.getText(),
                             opValue.getText(),
                             XSLStyleSheet.getText());
  String sqlStmt  = "delete styleSheets where fromApp = ? and " +
                    "                         toApp   = ? and " +
                    "                         op      = ?";
  try
  {
    PreparedStatement pStmt = conn.prepareStatement(sqlStmt);
    pStmt.setString(1, thisRecord.FROM);
    pStmt.setString(2, thisRecord.TO);
    pStmt.setString(3, thisRecord.TASK);
    pStmt.execute();
    conn.commit();
```

```
      System.out.println("Deleted !");
      pStmt.close();
      // Delete from vector...
      recVect.removeElementAt(currRec - 1);
      updateStatusBar();
      if (currRec >= recVect.size())
        currRec--;
      displayRecord(currRec);
      JOptionPane.showMessageDialog(null, "All right!");
    }
    catch (SQLException sqlE)
    {
      JOptionPane.showMessageDialog(null, sqlE.toString(),
                                    "Deleting record",
                                    JOptionPane.ERROR_MESSAGE);
    }
    catch (Exception e)
    {
      JOptionPane.showMessageDialog(null, e.toString(),
                                    "Deleting record",
                                    JOptionPane.ERROR_MESSAGE);
    }
  }

  void checkRecordChange()
  {
    thisRecord = new XslRecord(fromAppValue.getText(),
                               toAppValue.getText(),
                               opValue.getText(),
                               XSLStyleSheet.getText());
    if (!thisRecord.equals((XslRecord)recVect.elementAt(currRec-1)))
    {
      int result = JOptionPane.showConfirmDialog(null, "Record has changed¥nDo you
want to save the modifications ?");
      if (result == JOptionPane.YES_OPTION)
      {
        saveChanges(thisRecord, UPDATE);
        JOptionPane.showMessageDialog(null, "All right!");
      }
    }
  }

  boolean saveChanges(XslRecord rec,
                      int operation)
  {
    boolean ret = true;
    if (operation == this.UPDATE)
```

```
    {
      String theSqlStmt = "update styleSheets set xsl = ? where appFrom = ? and
appTo = ? and op = ?";
      try
      {
        PreparedStatement pStmt = conn.prepareStatement(theSqlStmt);
        pStmt.setString(1, rec.XSL);
        pStmt.setString(2, rec.FROM);
        pStmt.setString(3, rec.TO);
          pStmt.setString(4, rec.TASK);
        pStmt.execute();
        conn.commit();
        System.out.println("Updated !");
        pStmt.close();
        // Reinsert in vector...
        recVect.setElementAt(rec, currRec - 1);
      }
      catch (SQLException sqlE)
      {
        JOptionPane.showMessageDialog(null, sqlE.toString(),
                                      "Saving record",
                                      JOptionPane.ERROR_MESSAGE);
        ret = false;
      }
    }
    else
    {
      System.out.println("Inserting new record");
      String sqlStmt  = "insert into styleSheets " +
                        "           ( appFrom,   " +
                        "             appTo,     " +
                        "             op,        " +
                        "             xsl        " +
                        "           ) values     " +
                        "           (?, ?, ?, ?)";
      String sqlGetLob = "select xsl from styleSheets " +
                         "where appFrom = ? and " +
                         "      appTo   = ? and " +
                         "      op      = ?";
      try
      {
        PreparedStatement pStmt = conn.prepareStatement(sqlStmt);
        pStmt.setString(1, rec.FROM);
        pStmt.setString(2, rec.TO);
        pStmt.setString(3, rec.TASK);
        pStmt.setString(4, ""); // Null in the LOB, will be filled later
        pStmt.execute();
```

```
      System.out.println("Inserted !");
      pStmt.close();

      PreparedStatement fillLOBStmt = conn.prepareStatement(sqlGetLob);
      fillLOBStmt.setString(1, rec.FROM);
      fillLOBStmt.setString(2, rec.TO);
      fillLOBStmt.setString(3, rec.TASK);
      ResultSet lobRSet = fillLOBStmt.executeQuery();
      while (lobRSet.next())
      {
        CLOB clob = ((OracleResultSet)lobRSet).getCLOB(1);
        fillClob(conn, clob, rec.XSL);
      }
      conn.commit();

      // Add in vector...
      recVect.addElement(rec);
      currRec = recVect.size();
      displayRecord(currRec);
    }
    catch (SQLException sqlE)
    {
      JOptionPane.showMessageDialog(null, sqlE.toString(),
                                    "Inserting record",
                                    JOptionPane.ERROR_MESSAGE);
      ret = false;
    }

    inserting = false;

    fromAppValue.setEditable(false);
    toAppValue.setEditable(false);
    opValue.setEditable(false);
  }
  updateStatusBar();
  return ret;
}

void buttonClose_actionPerformed(ActionEvent e)
{
  validateRec();
}

void newButton_actionPerformed(ActionEvent e)
{
  fromAppValue.setEditable(true);
  toAppValue.setEditable(true);
```

```
  opValue.setEditable(true);
  inserting = true;
  XSLStyleSheet.setText("");
  fromAppValue.setText("");
  toAppValue.setText("");
  opValue.setText("");
}

void deleteButton_actionPerformed(ActionEvent e)
{
  deleteRec();
}

void queryButton_actionPerformed(ActionEvent e)
{
  if (queryState == ENTER_QUERY)
  {
    queryState = EXEC_QUERY;
    queryButton.setText("Execute Query");
    fromAppValue.setEditable(true);
    toAppValue.setEditable(true);
    opValue.setEditable(true);

    XSLStyleSheet.setEditable(false);
    statusBar.setText("Entering query");
    XSLStyleSheet.setText("");
    fromAppValue.setText("");
    toAppValue.setText("");
    opValue.setText("");

    newButton.setEnabled(false);
    firstButton.setEnabled(false);
    previousButton.setEnabled(false);
    nextButton.setEnabled(false);
    lastButton.setEnabled(false);
    validateButton.setEnabled(false);
    deleteButton.setEnabled(false);
  }
  else
  {
    queryState = ENTER_QUERY;
    queryButton.setText("Enter Query");
    statusBar.setText("Executing query");

    fromAppValue.setEditable(false);
    toAppValue.setEditable(false);
    opValue.setEditable(false);
```

```
      XSLStyleSheet.setEditable(true);

      newButton.setEnabled(true);
      firstButton.setEnabled(true);
      previousButton.setEnabled(true);
      nextButton.setEnabled(true);
      lastButton.setEnabled(true);
      validateButton.setEnabled(true);
      deleteButton.setEnabled(true);

      // Execute query
      String stmt = sqlStmt;
      boolean firstCondition = true;
      if (fromAppValue.getText().length() > 0)
      {
        stmt += ((firstCondition?" where ":" and ") + "fromApp like '" +
fromAppValue.getText() + "' ");
        firstCondition = false;
      }
      if (toAppValue.getText().length() > 0)
      {
        stmt += ((firstCondition?" where ":" and ") + "toApp like '" +
toAppValue.getText() + "' ");
        firstCondition = false;
      }
      if (opValue.getText().length() > 0)
      {
        stmt += ((firstCondition?" where ":" and ") + "op like '" +
opValue.getText() + "' ");
        firstCondition = false;
      }
      executeQuery(stmt);
      updateStatusBar();
      displayRecord(currRec);
    }
  }
}
```

Java の例 4: GUIInterface_AboutBoxPanel.java

```
package B2BDemo.StyleSheetUtil;
/**
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import oracle.jdeveloper.layout.*;

public class GUInterface_AboutBoxPanel1 extends JPanel
{
  JLabel jLabel1 = new JLabel();
  JLabel jLabel2 = new JLabel();
  JLabel jLabel3 = new JLabel();
  JLabel jLabel4 = new JLabel();
  GridBagLayout gridBagLayout1 = new GridBagLayout();
  Border border1 = new EtchedBorder();


  public GUInterface_AboutBoxPanel1()
  {
    try
    {
      jbInit();
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
  }

  private void jbInit() throws Exception
  {
   jLabel1.setText("Stored Style Sheets management.");
   jLabel2.setText("Olivier LE DIOURIS");
   jLabel3.setText("Copyright (c) 1999");
   jLabel4.setText("Oracle Corp.");
   this.setLayout(gridBagLayout1);
   this.setBorder(border1);
   this.add(jLabel1, new GridBagConstraints2(0, 0, 1, 1, 0.0, 0.0,
GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(5,5,0,5),0,0));
    this.add(jLabel2, new GridBagConstraints2(0, 1, 1, 1, 0.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0,5,0,5),0,0));
    this.add(jLabel3, new GridBagConstraints2(0, 2, 1, 1, 0.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0,5,0,5),0,0));
```

```
    this.add(jLabel4, new GridBagConstraints2(0, 3, 1, 1, 0.0, 0.0,
    GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0,5,5,5),0,0));
  }
}
```

Java の例 5: GUIStylesheet.java

```
package B2BDemo.StyleSheetUtil;
/**
 * A grapical utility to manipulate the stylesheets stored in the database,
 * in the AQ Schema. The stylsheets will be used to transform the incoming
 * document into the outgoing one.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
//import oracle.bali.ewt.border.UIBorderFactory;
//import oracle.bali.ewt.olaf.OracleLookAndFeel;

public class GUIStylesheet
{
  private static final boolean useBali = false;

  public GUIStylesheet()
  {
    Frame frame = new GUInterface();
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
    {
      frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width)
    {
      frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width)/2, (screenSize.height -
frameSize.height)/2);
    frame.addWindowListener(new WindowAdapter() { public void
windowClosing(WindowEvent e) { System.exit(0); } });
    frame.setVisible(true);
  }
```

```
  public static void main(String[] args)
  {
    new GUIStylesheet();
  }
}
```

XML プロセスおよび管理スクリプト

B2B XML アプリケーションで使用される XML プロセスおよび管理スクリプトは、次のとおりです。

- Java の例 6: Main4XMLtoDMLv2.java
- Java の例 7: ParserTest.java
- Java の例 8: TableInDocument.java
- Java の例 9: XMLFrame.java
- Java の例 10: XMLProducer.java
- Java の例 11: XMLtoDMLv2.java
- Java の例 12: XMLGen.java
- Java の例 13: XMLUtil.java
- Java の例 14: XSLTWrapper.java

Java の例 6: Main4XMLtoDMLv2.java

```
package B2BDemo.XMLUtil;
/**
 * A main for tests
 * The XMLtoDMLv2 utility takes an XML document that can contain
 * data to be inserted in several tables.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.io.*;
import java.net.*;

public class Main4XMLtoDMLv2 extends Object
{
  // Manage user input...
  private static BufferedReader _stdin = new BufferedReader(new
InputStreamReader(System.in));
  private static String _buf = "";
```

```
  private static String _userInput(String prompt) throws Exception
  {
    String retString;
    System.out.print(prompt);
    try { retString = _stdin.readLine(); }
    catch (Exception e)
    {
      System.out.println(e);
      throw(e);
    }
    return retString;
  }
  // for tests
  public static void main(String args[])
  {
    XMLtoDMLv2 x2d = new XMLtoDMLv2("scott", "tiger",

"jdbc:oracle:thin:@olediour-lap.us.oracle.com:1521:Ora8i");

    String xmldocname = "";
    try { xmldocname = userInput("XML file name > "); }
    catch (Exception e) {}
    String xmldoc = readURL(createURL(xmldocname));

    TableInDocument d[] = new TableInDocument[2];
    d[0] = new TableInDocument("ROWSET", "ROW", "DEPT");
    d[1] = new TableInDocument("EMP", "EMP_ROW", "EMP");

    try
    {
      x2d.insertFromXML(d, xmldoc);
      System.out.println(xmldocname + " processed.");
    }
    catch (Exception e)
    {
      System.err.println("Ooops:¥n" + e);
    }

    try { _buf = _userInput("End of task..."); } catch (Exception ioe) {}
  }

  public static URL createURL(String fileName)
  {
    URL url = null;
    try
    {
      url = new URL(fileName);
```

```
  }
  catch (MalformedURLException ex)
  {
    File f = new File(fileName);
    try
    {
      String path = f.getAbsolutePath();
      // This is a bunch of weird code that is required to
      // make a valid URL on the Windows platform, due
      // to inconsistencies in what getAbsolutePath returns.
      String fs = System.getProperty("file.separator");
      if (fs.length() == 1)
      {
        char sep = fs.charAt(0);
        if (sep != '/')
        path = path.replace(sep, '/');
        if (path.charAt(0) != '/')
        path = '/' + path;
      }
      path = "file://" + path;
      url = new URL(path);
    }
    catch (MalformedURLException e)
    {
      System.err.println("Cannot create url for: " + fileName);
      System.exit(0);
    }
  }
  return url;
}

public static String readURL(URL url)
{
  URLConnection newURLConn;
  BufferedInputStream newBuff;
  int nBytes;
  byte aByte[];
  String resultBuff = "";

  aByte = new byte[2];
  try
  {
//  System.out.println("Calling " + url.toString());
    try
    {
      newURLConn = url.openConnection();
      newBuff = new BufferedInputStream(newURLConn.getInputStream());
```

```
        resultBuff = "";
        while (( nBytes = newBuff.read(aByte, 0, 1)) != -1)
          resultBuff = resultBuff + (char)aByte[0];
      }
      catch (IOException e)
      {
        System.err.println(url.toString() + "¥n : newURLConn failed :¥n" + e);
      }
    }
    catch (Exception e) {}
    return resultBuff;
  }

  private static String userInput(String prompt) throws Exception
  {
    String retString;
    System.out.print(prompt);
    try { retString = _stdin.readLine(); }
    catch (Exception e)
    {
      System.out.println(e);
      throw(e);
    }
    return retString;
  }
}
```

Java の例 7: ParserTest.java

```
package B2BDemo.XMLUtil;

import org.xml.sax.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.sql.*;

import oracle.xml.sql.query.*;
import oracle.xml.sql.dml.*;

import org.w3c.dom.*;
import oracle.xml.parser.v2.*;
import org.xml.sax.*;
/**
 * Just a main for tests.
```

```
 * Show how to retrieve the ID and CUSTOMER_ID fro an XML document
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class ParserTest extends Object
{

  static DOMParser parser = new DOMParser();

  static String XMLDoc =
"<ROWSET>" +
"   <ROW NUM=¥"1¥">" +
"      <ID>23321</ID>" +
"      <ORDERDATE>2000-05-03 00:00:00.0</ORDERDATE>" +
"      <CONTACTNAME>JDevBC4J</CONTACTNAME>" +
"      <TRACKINGNO>AX23321</TRACKINGNO>" +
"      <STATUS>Pending</STATUS>" +
"      <ITEMS>" +
"         <ITEM_ROW NUM=¥"1¥">" +
"            <ID>1242</ID>" +
"            <QUANTITY>2</QUANTITY>" +
"            <ITEM_ID>403</ITEM_ID>" +
"            <ORD_ID>23321</ORD_ID>" +
"            <DISCOUNT>0</DISCOUNT>" +
"         </ITEM_ROW>" +
"      </ITEMS>" +
"   </ROW>" +
"</ROWSET>";
  /**
   * Constructor
   */
  public ParserTest()
  {
  }

  public static void main(String[] args)
  {
    parser.setValidationMode(false);
    try
    {
      parser.parse(new InputSource(new ByteArrayInputStream(XMLDoc.getBytes())));
      XMLDocument xml = parser.getDocument();
      XMLElement elmt = (XMLElement)xml.getDocumentElement();
      NodeList nl = elmt.getElementsByTagName("ID"); // ORD ID
      for (int i=0; i<nl.getLength(); i++)
      {
        XMLElement ordId = (XMLElement)nl.item(i);
```

```
        XMLNode theText = (XMLNode)ordId.getFirstChild();
        String ordIdValue = theText.getNodeValue();
        System.out.println(ordIdValue);
        break;
      }
      nl = elmt.getElementsByTagName("CUSTOMER_ID"); // CUSTOMER ID
      for (int i=0; i<nl.getLength(); i++)
      {
        XMLElement ordId = (XMLElement)nl.item(i);
        XMLNode theText = (XMLNode)ordId.getFirstChild();
        String custIdValue = theText.getNodeValue();
        System.out.println(custIdValue);
      }
    }
    catch (SAXParseException e)
    {
      System.out.println(e.getMessage());
    }
    catch (SAXException e)
    {
      System.out.println(e.getMessage());
    }
    catch (Exception e)
    {
      System.out.println(e.getMessage());
    }
  }
}
```

Java の例 8: TableInDocument.java

```
package B2BDemo.XMLUtil;
/**
 *   This class is used by the XMLtoDMLv2.java class
 *   It describes the matching between an XML document and a SQL table.
 *   Created to managed multi-level XML documents (Master-Details)
 *
 * @see XMLtoDMLv2
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class TableInDocument extends Object
{
  public String rowSet = "ROWSET";
  public String row    = "ROW";
  public String table  = "";
```

```
  public TableInDocument (String rset,
                          String r,
                          String t)
  {
    this.rowSet = rset;
    this.row    = r;
    this.table  = t;
  }
}
```

Java の例 9: XMLFrame.java

```
// Copyright (c) 2000 Oracle Corporation
package B2BDemo.XMLUtil;

import javax.swing.*;
import java.awt.*;
import oracle.xml.srcviewer.*;

import org.w3c.dom.*;
import oracle.xml.parser.v2.*;
import org.xml.sax.*;

/**
 * A Swing-based top level window class.
 * Implements the Code View of the Transviewer Bean.
 * Used in the demo to enhance the XML code propagated from one
 * component to another.
 *
 * @author Olivier LE DIOURIS
 */
public class XMLFrame extends JFrame
{
  BorderLayout borderLayout1 = new BorderLayout();
  JPanel jPanel1 = new JPanel();
  BorderLayout borderLayout2 = new BorderLayout();
  XMLSourceView xmlSourceViewPanel = new XMLSourceView();

  private String frameTitle = "";
  private XSLTWrapper xsltw = new XSLTWrapper();
  /**
   * Constructs a new instance.
   */
  public XMLFrame(String fTitle)
  {
    super();
```

```
    this.frameTitle = fTitle;
    try
    {
      jbInit();
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
  }

  /**
   * Initializes the state of this instance.
   */
  private void jbInit() throws Exception
  {
    this.getContentPane().setLayout(borderLayout1);
    this.setSize(new Dimension(400, 300));
    jPanel1.setLayout(borderLayout2);
    this.setTitle(this.frameTitle);
    this.getContentPane().add(jPanel1, BorderLayout.CENTER);
    jPanel1.add(xmlSourceViewPanel, BorderLayout.CENTER);
  }

  public void setXMLDocument(String xmlContent) throws Exception
  {
    xmlSourceViewPanel.setXMLDocument(xsltw.parseDocument(xmlContent));
  }
}
```

Java の例 10: XMLProducer.java

```
package B2BDemo.XMLUtil;
/**
 * A Wrapper around the XML SQL Utility
 * Could be called from any java object
 * to produce an XML document after a SQL query,
 * not only from a servlet.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
/**
 */
import java.sql.*;
import oracle.xml.sql.query.*;
```

```
public class XMLProducer
{
  Connection conn = null;
  String rowset = null;
  String row = null;

  public XMLProducer(Connection conn)
  {
    this.conn = conn;
  }

  public String getXMLString(ResultSet rSet)
  {
    return getXMLString(rSet, "N");
  }

  public String getXMLString(ResultSet rSet,
                             String DTD)
  {
    String finalDoc = "";

    try
    {
      boolean dtdRequired = false;
      if (DTD != null && DTD.length() > 0 && DTD.toUpperCase().equals("Y"))
        dtdRequired = true;
      // The Skill ! ////////////////////////////////////////////
      OracleXMLQuery oXmlq = new OracleXMLQuery(conn, rSet);      //
  //  oXmlq.useUpperCaseTagNames();                               //
      if (this.rowset != null)
        oXmlq.setRowsetTag(this.rowset);
      if (this.row != null)
        oXmlq.setRowTag(this.row);
      finalDoc = oXmlq.getXMLString(dtdRequired);                 //
      // That's it ! ////////////////////////////////////////////
    }
    catch (Exception e)
    {
      System.err.println(e);
    }
    return finalDoc;
  }

  public void setRowset(String rSet)
  {
    this.rowset = rSet;
  }
```

```
  public void setRow(String row)
  {
    this.row = row;
  }
}
```

Java の例 11: XMLtoDMLv2.java

```
package B2BDemo.XMLUtil;
/**
 *    This class takes an XML document as input to execute
 *    an insert into the database.
 *    Multi level XML documents are supported, but not if
 *    one element has several sons as
 *      <elem1>
 *        <elem11/>
 *        <elem12/>
 *      </elem1>
 *
 * @see TableInDocument
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import org.xml.sax.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.sql.*;

import oracle.xml.sql.query.*;
import oracle.xml.sql.dml.*;

import org.w3c.dom.*;
import oracle.xml.parser.v2.*;
import org.xml.sax.*;

public class XMLtoDMLv2 extends Object
{
  static DOMParser parser = new DOMParser();
  Connection conn = null;
  String username = "";
  String password = "";
  String connURL = "";

  public XMLtoDMLv2(String username,
                    String password,
                    String connURL)
```

```
  {
    this.username = username;
    this.password = password;
    this.connURL  = connURL;
  }

  public void insertFromXML(TableInDocument tInDoc[],
                            String document) throws Exception
  {
    if (conn == null)
      getConnected();

    String xmlString = "";
    try
    { xmlString = readURL(createURL(document)); }
    catch (Exception e)
    { xmlString = document; }

//  System.out.println("xml2Insert = ¥n" + xmlString);

    // The returned String is turned into an XML Document
    XMLDocument xmlDoc = parseDocument(xmlString);
    // And we take a reference on the root of this document
    XMLElement e = (XMLElement) xmlDoc.getDocumentElement();

    // Let's walk thru the ROW nodes
    NodeList nl  = e.getChildrenByTagName(tInDoc[0].row); // "ROW"
//  System.out.println("This document has " + nl.getLength() + " ROW(s)");

    Vector sqlStmt = new Vector();
    scanLevel(0, tInDoc, nl, sqlStmt);

    // Now execute all the statements in the Vector, in reverse order (FK...)
    int i = sqlStmt.size();
    Enumeration enum = sqlStmt.elements();
    while (i > 0)
    {
      i--;
      String s = (String)sqlStmt.elementAt(i);
   // System.out.println("Executing " + s);
      executeStatement(s);
    }
  }

  // This one is recursive
  private int scanLevel(int level,
                        TableInDocument tInDoc[],
```

```
                        NodeList nl,
                        Vector sqlStmt) throws Exception
  {
    int nbRowProcessed = 0;
    Vector columnNames = new Vector();
    Vector columnValues = null;
    String[] colTypes = null;

    String columns = "", values = "";
    // Loop in tree...
    boolean firstLoop = true;
    for (int i=0; i<nl.getLength(); i++)  // Loop on all rows of XML doc
    {
      columnValues = new Vector();
      XMLElement aRow = (XMLElement) nl.item(i);
  //  String numVal   = aRow.getAttribute("num");
  //  System.out.println("NUM = " + numVal);
      NodeList nlRow = aRow.getChildNodes();
  //  System.out.println("a Row has " + nlRow.getLength() + " children");
      for (int j=0; j<nlRow.getLength(); j++)
      {
        XMLElement anXMLElement = (XMLElement)nlRow.item(j);
        if (anXMLElement.getChildNodes().getLength() == 1 &&
            (level == (tInDoc.length - 1) || (level < (tInDoc.length - 1) &&
!(anXMLElement.getNodeName().equals(tInDoc[level+1].rowSet)))) )
        {
      //  System.out.println("Element " + (j+1) + "=" + anXMLElement.getNodeName());
      //  System.out.print(anXMLElement.getNodeName());
          if (firstLoop)
            columnNames.addElement(anXMLElement.getNodeName());
          // Value
          XMLNode nodeValue = (XMLNode) anXMLElement.getFirstChild();
      //  System.out.println("¥t" + nodeValue.getNodeValue());
          columnValues.addElement(nodeValue.getNodeValue());
        }
        else
        {
   //     System.out.println(anXMLElement.getNodeName() + " has " +
anXMLElement.getChildNodes().getLength() + " children");
   //     System.out.println("Comparing " + anXMLElement.getNodeName() + " and " +
tInDoc[level+1].rowSet);
          if (level < (tInDoc.length - 1) &&
anXMLElement.getNodeName().equals(tInDoc[level+1].rowSet))
          {
   //       System.out.println("Searching for " + tInDoc[level+1].row);
            NodeList nl2  = anXMLElement.getChildrenByTagName(tInDoc[level+1].row);
// "ROW"
```

```
            if (nl2 == null)
              System.out.println("Nl2 is null for " + tInDoc[level+1].row);
            scanLevel(level + 1, tInDoc, nl2, sqlStmt);
          }
        }
      }
 //   System.out.println("INSERT INTO " + tableName + " (" + columns + ") VALUES ("
+ values + ")");
      try
      {
        if (firstLoop)
        {
          firstLoop = false;
          String selectStmt = "SELECT ";
          boolean comma = false;
          Enumeration cNames = columnNames.elements();
          while (cNames.hasMoreElements())
          {
            columns += ((comma?", ":"") + (String)cNames.nextElement());
            if (!comma)
              comma = true;
          }
          selectStmt += columns;
          selectStmt += (" FROM " + tInDoc[level].table + " WHERE 1 = 2"); // No row
retrieved
          Statement stmt = conn.createStatement();
   //     System.out.println("Executing: " + selectStmt);
          ResultSet rSet = stmt.executeQuery(selectStmt);
          ResultSetMetaData rsmd = rSet.getMetaData();
          colTypes = new String[rsmd.getColumnCount()];
          for (int ci=0; ci<(rsmd.getColumnCount()); ci++)
          {
        //  System.out.println("Col " + (ci+1) + ":" + rsmd.getColumnName(ci+1) + ",
" + rsmd.getColumnTypeName(ci+1));
            colTypes[ci] = rsmd.getColumnTypeName(ci+1);
          }
          rSet.close();
          stmt.close();
        }
        // Build Value Part
        int vi = 0;
        Enumeration cVal = columnValues.elements();
        boolean comma = false;
        while (cVal.hasMoreElements())
        {
          if (comma)
            values += ", ";
```

```
          comma = true;
          if (colTypes[vi].equals("DATE"))
            values += ("TO_DATE(SUBSTR(");
          values += ("'" + cVal.nextElement() + "'");
          if (colTypes[vi].equals("DATE"))
            values += (", 1, 19), 'YYYY-MM-DD HH24:MI:SS')");
          vi++;
        }
        // Go !
    //  System.out.println("Stmt:" + "INSERT INTO " + tInDoc[level].table + " (" +
columns + ") VALUES (" + values + ")");
        sqlStmt.addElement("INSERT INTO " + tInDoc[level].table + " (" + columns +
") VALUES (" + values + ")");
        nbRowProcessed++;
      }
      catch (Exception execE)
      {
 //     System.err.println("Executing " + execE);
        throw execE;
      }
      values = "";
    }
//  System.out.println("End of Loop");
    return nbRowProcessed;
  }

  public static XMLDocument parseDocument(String documentStream) throws Exception
  {
    XMLDocument returnXML = null;
    try
    {
      parser.parse(new InputSource(new
ByteArrayInputStream(documentStream.getBytes())));
      returnXML = parser.getDocument();
    }
    catch (SAXException saxE)
    {
 //   System.err.println("Parsing XML¥n" + "SAX Exception:¥n" + saxE.toString());
 //   System.err.println("For:¥n" + documentStream + "¥nParse failed SAX : " +
saxE);
      throw saxE;
    }
    catch (IOException e)
    {
 //   System.err.println("Parsing XML¥n" + "Exception:¥n" + e.toString());
 //   System.err.println("Parse failed : " + e);
      throw e;
```

```
  }
  return returnXML;
}
// Create a URL from a file name
private static URL createURL(String fileName) throws Exception
{
  URL url = null;
  try
  {
    url = new URL(fileName);
  }
  catch (MalformedURLException ex) // It is not a valid URL, maybe a file...
  {
    File f = new File(fileName);
    try
    {
      String path = f.getAbsolutePath();
      // This is a bunch of weird code that is required to
      // make a valid URL on the Windows platform, due
      // to inconsistencies in what getAbsolutePath returns.
      String fs = System.getProperty("file.separator");
      if (fs.length() == 1)
      {
        char sep = fs.charAt(0);
        if (sep != '/')
          path = path.replace(sep, '/');
        if (path.charAt(0) != '/')
          path = '/' + path;
      }
      path = "file://" + path;
      url = new URL(path);
    }
    catch (MalformedURLException e)
    {
  //  System.err.println("Cannot create url for: " + fileName);
      throw e;   // It's not a file either...
    }
  }
  return url;
}

private static String readURL(URL url) throws Exception
{
  URLConnection newURLConn;
  BufferedInputStream newBuff;
  int nBytes;
  byte aByte[];
```

```
  String resultBuff = "";

  aByte = new byte[2];
  try
  {
 // System.out.println("Calling " + url.toString());
    try
    {
      newURLConn = url.openConnection();
      newBuff = new BufferedInputStream(newURLConn.getInputStream());
      resultBuff = "";
      while (( nBytes = newBuff.read(aByte, 0, 1)) != -1)
        resultBuff = resultBuff + (char)aByte[0];
    }
    catch (IOException e)
    {
 //   System.err.println("Opening locator¥n" + e.toString());
 //   System.err.println(url.toString() + "¥n : newURLConn failed :¥n" + e);
      throw e;
    }
  }
  catch (Exception e)
  {
 // System.err.println("Read URL¥n" + e.toString());
    throw e;
  }
  return resultBuff;
}

private void executeStatement(String strStmt) throws SQLException, Exception
{
  if (conn == null)
    throw new Exception("Connection is null");
  try
  {
    Statement stmt = conn.createStatement();
    stmt.execute(strStmt);
    stmt.close();
  }
  catch (SQLException e)
  {
    System.err.println("Failed to execute statement¥n" + strStmt);
    throw e;
  }
}

private void getConnected() throws Exception
```

```
  {
    try
    {
      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      conn = DriverManager.getConnection(connURL, username, password);
    }
    catch (Exception e)
    {
 //   System.err.println(e);
      throw e;
    }
  }
  public Connection getConnection()
  {
    return conn;
  }
}
```

Java の例 12: XMLGen.java

```
package B2BDemo.XMLUtil;

import java.sql.*;
/**
 * This class is used by the Action Handler called by the XSQL Servlet
 * in placeOrder.xsql. It generates the original XML Document to be
 * sent to the broker
 *
 * @see B2BMessage
 * @see XMLProducer
 * @see RetailActionHandler
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class XMLGen extends Object
{
  static Connection conn = null;
  // Default connection parameters
  static String appURL      = "jdbc:oracle:thin:@localhost:1521:ORCL";
  static String appUser     = "retailer";
  static String appPassword = "retailer";

  static String XMLStmt =
  "SELECT O.ID as ¥"Id¥","                +
  "       O.ORDERDATE as ¥"Orderdate¥", " +
  "       O.CONTACTNAME as ¥"Contactname¥"," +
  "       O.TRACKINGNO as ¥"Trackingno¥"," +
```

```
"       O.STATUS as ¥"Status¥"," +
"       O.CUSTOMER_ID as ¥"CustomerId¥"," +
"       CURSOR (SELECT L.ID as ¥"Id¥"," +
"                      L.QUANTITY as ¥"Quantity¥", " +
"                      L.ITEM_ID as ¥"ItemId¥"," +
"                      L.ORD_ID as ¥"OrdId¥"," +
"                      L.DISCOUNT as ¥"Discount¥" " +
"               FROM LINE_ITEM L " +
"               WHERE L.ORD_ID = O.ID) as ¥"LineItemView¥" " +
"FROM ORD O " +
"WHERE O.ID = ?";

public static String returnDocument (Connection c, String ordId)
{
  String XMLDoc = "";
  try
  {
    if (c != null)
      conn = c;
    if (conn == null)
      _getConnected(appURL, appUser, appPassword);
    XMLProducer xmlp = null;
    xmlp = new XMLProducer(conn);  // The XML Producer
    xmlp.setRowset("Results");
    xmlp.setRow("OrdView");
    PreparedStatement stmt = conn.prepareStatement(XMLStmt);
    stmt.setString(1, ordId);
    ResultSet rSet = stmt.executeQuery();

    XMLDoc = xmlp.getXMLString(rSet, "Y");
    rSet.close();
    stmt.close();
    if (c == null)
    {
      conn.close();
      conn = null;
    }
  }
  catch (SQLException e)
  {}
  return XMLDoc;
}

private static void _getConnected(String connURL,
                                  String userName,
                                  String password)
{
```

```
    try
    {
      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      conn = DriverManager.getConnection(connURL, userName, password);
    }
    catch (Exception e)
    {
      System.err.println(e);
      System.exit(1);
    }
  }

  public static void main (String[] args) // Just for test !!
  {
    System.out.println(returnDocument(null, "28004"));
  }
}
```

Java の例 13: XMLUtil.java

```
package B2BDemo.XMLUtil;
/**
 * Matches a record of the Stylesheet table in the AQ Schema.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class XslRecord
{
  public String FROM;
  public String TO;
  public String TASK;
  public String XSL;

  public XslRecord(String FROM,
                   String TO,
                   String TASK,
                   String XSL)
  {
    this.FROM = FROM;
    this.TO = TO;
    this.TASK = TASK;
    this.XSL = XSL;
  }

  public boolean equals(XslRecord x)
  {
```

```
    if (this.FROM.equals(x.FROM) &&
        this.XSL.equals(x.XSL) &&
        this.TASK.equals(x.TASK) &&
        this.TO.equals(x.TO))
      return true;
    else
      return false;
  }
}
```

Java の例 14: XSLTWrapper.java

```
package B2BDemo.XMLUtil;
/**
 * Wraps some parser features.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.net.*;
import java.io.*;

import org.w3c.dom.*;
import oracle.xml.parser.v2.*;
import org.xml.sax.*;

/**
 * This class is a wrapper for the XSL Transformer provided with the
 * Oracle XML Parser for Java V2.
 *
 * It processes XSL Transformations from Strings, files or URL as well.
 *
 * @author Olivier Le Diouris. Partner Services. Oracle Corp.
 * @version 1.0
 *
 */
public class XSLTWrapper
{
  DOMParser parser;

  String xml    = "";
  String xsl    = "";
  String result = "";

  private static boolean _debug = false;

  public XSLTWrapper()
  {
```

```
    parser = new DOMParser();
  }

  public void process() throws Exception
  {
    if (xml.length() == 0)
      throw new Exception("XML Document is empty");
    if (xsl.length() == 0)
      throw new Exception("XSL Document is empty");
    result = processTransformation(xml, xsl);
  }

  public void putXml(String xml) throws Exception
  {
    if (_debug) System.out.println("Recieved XML : ¥n" + xml);
    this.xml = xml;
  }
  public void putXsl(String xsl) throws Exception
  {
    this.xsl = xsl;
    if (_debug) System.out.println("Recieved XSL: ¥n" + xsl);
  }
  public String getXml() throws Exception
  {
    return xml;
  }
  public String getXsl() throws Exception
  {
    return xsl;
  }
  public String getProcessResult() throws Exception
  {
    return result;
  }

  // Turns a String into an XMLDocument
  public XMLDocument parseDocument(String documentStream) throws Exception
  {
    XMLDocument returnXML = null;
    try
    {
      parser.parse(new InputSource(new
ByteArrayInputStream(documentStream.getBytes())));
      returnXML = parser.getDocument();
    }
    catch (SAXException saxE)
    {
```

```
      if (_debug) System.err.println("For:¥n" + documentStream + "¥nParse failed SAX
: " + saxE);
      throw new Exception("Parsing XML¥n" + "SAX Exception:¥n" + saxE.toString());
    }
    catch (IOException e)
    {
      if (_debug) System.err.println("Parse failed : " + e);
      throw new Exception("Parsing XML¥n" + "IOException:¥n" + e.toString());
    }
    return returnXML;
  }

  private XMLDocument processXML(XMLDocument xml,
                                 XMLDocument xslDoc) throws Exception
  {
    XMLDocument out = null;
    URL xslURL = null;

    try
    {
      parser.setPreserveWhitespace(true);
      parser.setValidationMode(false);       // Validation. Should be an option.
      // instantiate a stylesheet
      XSLStylesheet xsl = new XSLStylesheet(xslDoc, xslURL);
      XSLProcessor processor = new XSLProcessor();

      // display any warnings that may occur
      processor.showWarnings(true);
      processor.setErrorStream(System.err);

      // Process XSL
      DocumentFragment result = processor.processXSL(xsl, xml);

      // create an output document to hold the result
      out = new XMLDocument();
      /*
      // create a dummy document element for the output document
      Element root = out.createElement("root");
      out.appendChild(root);
      // append the transformed tree to the dummy document element
      root.appendChild(result);
       */
      out.appendChild(result);
      // print the transformed document
      // out.print(System.out);
    }
    catch (Exception e)
```

```
  {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    PrintWriter pw = new PrintWriter(baos);
    e.printStackTrace(pw);
    e.printStackTrace();
    throw new Exception("ProcessXML¥n" +  baos.toString());
  }
  return(out);
}

/**
 *   XML String and XSL String as input
 *     Input Strings may content :
 *            the name of a URL
 *            the name of a file (on the local file system)
 *            the document itself
 *   XML String as output.
 */
public String processTransformation(String xmlStream,
                                    String xslStream) throws Exception
{
  String xmlContent = "";
  String xslContent = "";

  try
  { xmlContent = readURL(createURL(xmlStream)); }
  catch (Exception e)
  { xmlContent = xmlStream; }

  try
  { xslContent = readURL(createURL(xslStream)); }
  catch (Exception e)
  { xslContent = xslStream; }

  if (_debug) System.out.println("xmlStream = " + xmlContent);
  if (_debug) System.out.println("xslStream = " + xslContent);

  XMLDocument xml = parseDocument(xmlContent);
  XMLDocument xsl = parseDocument(xslContent);

  XMLDocument out = processXML(xml, xsl);
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  try
  { out.print(baos); }
  catch (IOException ioE)
  {
    if (_debug) System.err.println("Exception:" + ioE);
```

```
    throw new Exception("XML Processing throws IOException¥n" +  ioE.toString());
  }
  return (baos.toString());
}

// Create a URL from a file name
private static URL createURL(String fileName) throws Exception
{
  URL url = null;
  try
  {
    url = new URL(fileName);
  }
  catch (MalformedURLException ex) // It is not a valid URL, maybe a file...
  {
    File f = new File(fileName);
    try
    {
      String path = f.getAbsolutePath();
      // This is a bunch of weird code that is required to
      // make a valid URL on the Windows platform, due
      // to inconsistencies in what getAbsolutePath returns.
      String fs = System.getProperty("file.separator");
      if (fs.length() == 1)
      {
        char sep = fs.charAt(0);
        if (sep != '/')
          path = path.replace(sep, '/');
        if (path.charAt(0) != '/')
          path = '/' + path;
      }
      path = "file://" + path;
      url = new URL(path);
    }
    catch (MalformedURLException e)
    {
      if (_debug) System.err.println("Cannot create url for: " + fileName);
      throw e;   // It's not a file either...
    }
  }
  return url;
}

private static String readURL(URL url) throws Exception
{
  URLConnection newURLConn;
  BufferedInputStream newBuff;
```

```
    int nBytes;
    byte aByte[];
    String resultBuff = "";

    aByte = new byte[2];
    try
    {
   // System.out.println("Calling " + url.toString());
      try
      {
        newURLConn = url.openConnection();
        newBuff = new BufferedInputStream(newURLConn.getInputStream());
        resultBuff = "";
        while (( nBytes = newBuff.read(aByte, 0, 1)) != -1)
          resultBuff = resultBuff + (char)aByte[0];
      }
      catch (IOException e)
      {
   //   System.err.println("Opening locator¥n" + e.toString());
   //   System.err.println(url.toString() + "¥n : newURLConn failed :¥n" + e);
        throw e;
      }
    }
    catch (Exception e)
    {
   // System.err.println("Read URL¥n" + e.toString());
      throw e;
    }
    return resultBuff;
  }
}
```

B2B XML アプリケーションで使用されるその他のスクリプト

XML の例 1: XSQL の構成 ー XSQLConfig.xml

```
<?xml version="1.0" ?>
<!--
| $Author: smuench $
| $Date: 2000/03/14 10:36:42 $
| $Source: C:¥¥cvsroot/xsql/src/XSQLConfig.xml,v $
| $Revision: 1.11 $
+-->
<XSQLConfig>

  <!--
  |
  | This section defines configuration settings
  | specific to the XSQL Servlet
  |
  +-->
  <servlet>

   <!--
    |
    | Sets the size (in bytes) of the buffered output stream.
    | If your servlet engine already buffers I/O to the
    | Servlet Output Stream, then you can set to 0
    | to avoid additional buffering.
    |
    |    <output-buffer-size>10000</output-buffer-size>
    |
    +-->
    <output-buffer-size>0</output-buffer-size>

   <!--
    |
    | Add <media-type> elements as shown below to cause
    | the XSQL Servlet to *suppress* sending the "charset=XXX"
    | portion of the Media/Content-type.
    |
    | For example, sending a character set for "image/svg"
    | documents seems to confuse current SVG plugins.
    |
    | <suppress-mime-charset>
    |   <media-type>image/svg</media-type>
    | </suppress-mime-charset>
    |
```

```
   +-->

   <suppress-mime-charset>
      <media-type>image/svg</media-type>
   </suppress-mime-charset>

 </servlet>

<!--
 |
 | This section defines XSQL Page Processor configuration settings.
 |
 +-->
 <processor>

   <!--
   |
   | Connection definitions (see <connectiondefs> below)
   | are cached when the XSQL Page Processor is initialized.
   |
   | Set to "yes" to cause the processor to
   | reread the XSQLConfig.xml file to reload
   | connection definitions if an attempt is made
   | to request a connection name that's not in the
   | cached connection list. The "yes" setting is useful
   | during development when you might be adding new
   | <connection> definitions to the file while the
   | servlet is running. Set to "no" to avoid reloading
   | the connection definition file when a connection name
   | is not found in the in-memory cache.
   |
   +-->

   <reload-connections-on-error>yes</reload-connections-on-error>

   <!--
   |
   | Set the default value of the Row Fetch Size
   | for retrieving information from SQL queries
   | from the database. Only takes effect if you
   | are using the Oracle JDBC Driver, otherwise
   | the setting is ignored. Useful for reducing
   | network roundtrips to the database from
   | the servlet engine running in a different tier.
   |
   |     <default-fetch-size>50</default-fetch-size>
   |
```

```
+-->

<default-fetch-size>50</default-fetch-size>

<!--
|
| Set the value of the XSQL LRU Cache for XSQL Pages
| This determines the maximum number of stylesheets
| that will be cached. Least recently used sheets get
| "bumped" out of the cache if you go beyond this number.
|
| <page-cache-size>25</page-cache-size>
|
+-->

<page-cache-size>25</page-cache-size>

<!--
|
| Set the value of the XSQL LRU Cache for XSL Stylesheets.
| This determines the maximum number of stylesheets
| that will be cached. Least recently used sheets get
| "bumped" out of the cache if you go beyond this number.
|
|     <stylesheet-cache-size>25</stylesheet-cache-size>
|
+-->

<stylesheet-cache-size>25</stylesheet-cache-size>

<!--
|
| Set the parameters controlling stylesheet pools.
|
| Each cached stylesheet is actually a cached pool
| of stylesheet instances. These values control
| The initial number of stylesheet instances in the
| pool, the number that will be added/incremented
| when under-load the pool must be grown, and
| the number of seconds that must transpire without
| activity before a stylesheet instance will be
| dropped out of the pool to shrink it back towards
| its initial number.
|
| <stylesheet-pool>
|   <initial>1</initial>
|   <increment>1</increment>
```

```
|   <timeout-seconds>60</timeout-seconds>
| </stylesheet-pool>
|
+-->

<stylesheet-pool>
  <initial>1</initial>
  <increment>1</increment>
  <timeout-seconds>60</timeout-seconds>
</stylesheet-pool>

<!--
|
| Set the parameters controlling database connection pools.
|
| When used, each named connection defined can have a pool of
| connection instances to share among requests.  These values
| control The initial number of stylesheet instances in the pool,
| the number that will be added/incremented when under-load the
| pool must be grown, and the number of seconds that must
| transpire without activity before a stylesheet instance will be
| dropped out of the pool to shrink it back towards its initial
| number.
|
| If the "dump-allowed" element has the value "yes"
| then a browser-based status report that dumps the
| current state of the connection pools is enabled.
|
| <connection-pool>
|   <initial>2</initial>
|   <increment>1</increment>
|   <timeout-seconds>60</timeout-seconds>
|   <dump-allowed>no</dump-allowed>
| </connection-pool>
|
+-->

<connection-pool>
  <initial>2</initial>
  <increment>1</increment>
  <timeout-seconds>60</timeout-seconds>
  <dump-allowed>no</dump-allowed>
</connection-pool>

<!--
|
| Include timing information (in Milliseconds)
```

```
    |
    | <timing-info>
    |   <page>yes</page>
    |  <action>yes</action>
    | </timing-info>
    |
    +-->

    <timing-info>
      <page>no</page>
      <action>no</action>
    </timing-info>

  </processor>

  <!--
  |
  | This section defines HTTP Proxy Server name
  | and port for use by the <xsql:include-xml>
  | action. If you intend to use <xsql:include-xml>
  | to include XML from URL's outside a firewall,
  | uncomment the:
  |
  |  <http>
  |    <proxyhost>your-proxy-server.yourcompany.com</proxyhost>
  |    <proxyport>80</proxyport>
  |  </http>
  |
  | section below and change the proxyhost and proxyport
  | as appropriate. If left commented out, then the XSQL
  | Page processor does not use a proxy server.
  |
  +-->

<!--

  <http>
    <proxyhost>your-proxy-server.yourcompany.com</proxyhost>
    <proxyport>80</proxyport>
  </http>

-->

  <!--
  |
  | This section defines convenient "nicknames" for
  | one or more database connections. You can include
```

```
| any number of <connection> elements inside of
| the <connectiondefs> element. XSQL Pages refer to
| these connections by their name in the "connection"
| attribute on the document element of the page.
|
+-->

<connectiondefs>

  <connection name="demo">
    <username>scott</username>
    <password>tiger</password>
    <dburl>jdbc:oracle:thin:@localhost:1521:ORCL</dburl>
    <driver>oracle.jdbc.driver.OracleDriver</driver>
  </connection>
  <connection name="xmlbook">
    <username>xmlbook</username>
    <password>xmlbook</password>
    <dburl>jdbc:oracle:thin:@localhost:1521:ORCL</dburl>
    <driver>oracle.jdbc.driver.OracleDriver</driver>
  </connection>
  <connection name="lite">
    <username>system</username>
    <password>manager</password>
    <dburl>jdbc:Polite:POlite</dburl>
    <driver>oracle.lite.poljdbc.POLJDBCDriver</driver>
  </connection>
  <connection name="retail">
    <username>retailer</username>
    <password>retailer</password>
    <dburl>jdbc:oracle:thin:@atp-1.us.oracle.com:1521:ORCL</dburl>
    <driver>oracle.jdbc.driver.OracleDriver</driver>
  </connection>

</connectiondefs>

<!--
|
| This section registers pre-defined element names and
| handler classes for user-defined XSQL page actions
|
| The section looks like:
|
| <actiondefs>
|  <action>
|    <elementname>myAction</elementname>
|    <handlerclass>mypackage.MyActionHandler</handlerclass>
```

```
  |  </action>
  |      :
  | <actiondefs>
  |
  | Action Handler classes must implement the interface
  | oracle.xml.xsql.XSQLActionHandler.
  |
  | Once registered here, user-defined actions can be
  | used in the same way as built-in XSQL actions, for example
  | including the <xsql:myAction> element in your page.
  |
  +-->
  <actiondefs>
    <action>
      <elementname>param</elementname>

<handlerclass>oracle.xml.xsql.actions.ExampleGetParameterHandler</handlerclass>
    </action>
    <action>
      <elementname>current-date</elementname>

<handlerclass>oracle.xml.xsql.actions.ExampleCurrentDBDateHandler</handlerclass>
    </action>
  </actiondefs>

</XSQLConfig>
```

Java の例 15: メッセージ・ヘッダー・スクリプト — MessageHeaders.java

メッセージ・ヘッダー・スクリプトを次に示します。

```
package B2BDemo;
/**
 * Describes the headers used in the messages
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class MessageHeaders extends Object
{
  public static String APP_A          = "RETAIL";
  public static String APP_B          = "SUPPLY";
  public static String BROKER         = "BROKER";
  public static String EXIT           = "EXIT";
  public static String NEW_ORDER      = "NEW ORDER";
  public static String UPDATE_ORDER   = "UPDATE ORDER";
}
```

Java の例 16: メッセージ・ブローカで使用される定数の保持 ― AppCste.java

```
package B2BDemo;
/**
 * Holds the constants to be used by the Message Broker
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class AppCste extends Object
{
 public final static String AQDBUrl =
  "jdbc:oracle:thin:@atp-1.us.oracle.com:1521:ORCL";
  public final static String AQuser  = "aqMessBrok";
  public final static String AQpswd  = "aqMessBrok";
}
```

リテーラ・スクリプト

リテーラは、次のスクリプトを使用します。

- Java の例 17: リテーラ側でのサプライヤから送信されるステータス更新の待機 ― UpdateMaster.java

Java の例 17: リテーラ側でのサプライヤから送信されるステータス更新の待機 ― UpdateMaster.java

```
package B2BDemo.Retailer;
/**
 *
 * This class implements the component waiting on the retailer side for
 * the status update made by the Supplier after shipping.
 * The recieved document is parsed and its content is used to make
 * the convenient update in the database.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 *
 */
import java.io.*;
import java.util.*;
import java.net.*;
import java.sql.*;

import oracle.xml.sql.query.*;
```

```
import oracle.xml.sql.dml.*;

import org.w3c.dom.*;
import oracle.xml.parser.v2.*;
import org.xml.sax.*;

import B2BDemo.AQUtil.*;
import B2BDemo.*;
import B2BDemo.XMLUtil.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//import oracle.bali.ewt.border.UIBorderFactory;
//import oracle.bali.ewt.olaf.OracleLookAndFeel;

public class UpdateMaster extends Object
{
  private BufferedReader stdin = new BufferedReader(new
InputStreamReader(System.in));

  private static boolean stepByStep = false;
  private static boolean verbose    = false;
  private static Integer pauseTime  = null;

  AQReader aqr;

  private static final String userName = "retailer";
  private static final String password = "retailer";
  private static String url      = "jdbc:oracle:thin:@localhost:1521:ORCL"; // This
is the default value !
  private static Connection conn = null;

  String currOrdId = "";

  DOMParser parser = new DOMParser();
  /**
   * Constructor
   */
  public UpdateMaster()
  {
    XMLFrame frame = new XMLFrame("Retailer");
    /**
    try
    {
      OracleLookAndFeel.setColorScheme(Color.cyan);
//    OracleLookAndFeel.setColorScheme("Titanium");
```

```
      UIManager.setLookAndFeel(new OracleLookAndFeel());
      SwingUtilities.updateComponentTreeUI(frame);
      frame.setBackground(UIManager.getColor("darkIntensity"));
    }
    catch (Exception e)
    {
      System.err.println("Exception for Oracle Look and Feel:" + e );
    }
    */
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
    {
      frameSize.height = screenSize.height;
    }
    /**
    if (frameSize.width > screenSize.width)
    {
      frameSize.width = screenSize.width;
    }
    */
    frameSize.width = screenSize.width / 3;

//  frame.setLocation((screenSize.width - frameSize.width)/2, (screenSize.height -
frameSize.height)/2);
    frame.setLocation(0, (screenSize.height - frameSize.height)/2);
//  frame.addWindowListener(new WindowAdapter() { public void
windowClosing(WindowEvent e) { System.exit(0); } });
    frame.setVisible(true);

      // Initialize AQ reader
    aqr = new AQReader(AppCste.AQuser,
                       AppCste.AQpswd,
                       AppCste.AQDBUrl,
                       "AppFour_QTab",
                       "AppFourMsgQueue");
    boolean go = true;
    while (go)
    {
      String ordIdValue = "";
      B2BMessage sm = aqr.readQ();
      if (verbose)
        System.out.println("Recieved¥nFrom > " + sm.getFrom() +
                                  "¥nTo   > " + sm.getTo() +
                                  "¥nType > " + sm.getType() +
                                  "¥nContent >¥n" + sm.getContent());
```

```
      else
        System.out.println("Recieved¥nFrom > " + sm.getFrom() +
                                   "¥nTo   > " + sm.getTo() +
                                   "¥nType > " + sm.getType());
      String xmlDoc = sm.getContent();
      if (xmlDoc != null && xmlDoc.length() > 0)
      {
        try { frame.setXMLDocument(sm.getContent()); }
        catch (Exception e)
        {
          e.printStackTrace();
        }
      }
      if (stepByStep)
      {
        if (pauseTime != null)
        {
          System.out.println("Waiting for " + pauseTime.longValue() + "
milliseconds");
          try { Thread.sleep(pauseTime.longValue()); } catch (InterruptedException
e) {}
        }
        else
          try { String s = _userInput("[Hit return to continue]"); } catch
(Exception e) {}
      }

      if (sm.getType().equals(MessageHeaders.EXIT))
        go = false;
      else
      {
        System.out.println("Updating");
        try
        {
          parser.parse(new InputSource(new
ByteArrayInputStream(sm.getContent().getBytes())));
          XMLDocument xml = parser.getDocument();
          XMLElement elmt = (XMLElement)xml.getDocumentElement();
          NodeList nl = elmt.getElementsByTagName("SHIP"); // ORD ID
          for (int i=0; i<nl.getLength(); i++)
          {
            XMLElement ordId = (XMLElement)nl.item(i);
            XMLNode theText = (XMLNode)ordId.getFirstChild();
            currOrdId = theText.getNodeValue();
            System.out.println("Gonna update " + currOrdId);
            try
            {
```

```
            if (conn == null)
              getConnected(url, userName, password);
            String strStmt = "update ORD set STATUS = 'Shipped' where ID = ?";
            PreparedStatement pStmt = conn.prepareStatement(strStmt);
            pStmt.setString(1, currOrdId);
            pStmt.execute();
            conn.commit();
            pStmt.close();
            System.out.println("Done !");
          }
          catch (SQLException e)
          {
            System.out.println("Pb updating the ORD¥n" + e.toString());
          }

        }
      }
      catch (SAXParseException e)
      {
        System.out.println(e.getMessage());
      }
      catch (SAXException e)
      {
        System.out.println(e.getMessage());
      }
      catch (Exception e)
      {
        System.out.println(e.getMessage());
      }
    }
  }
  frame.setVisible(false);
  System.exit(0);
}

private static void getConnected(String connURL,
                                 String userName,
                                 String password)
{
  try
  {
    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
    conn = DriverManager.getConnection(connURL, userName, password);
  }
  catch (Exception e)
  {
    System.err.println(e);
```

```
        System.exit(1);
      }
    }

    private String _userInput(String prompt) throws Exception
    {
      String retString;
      System.out.print(prompt);
      try { retString = stdin.readLine(); }
      catch (Exception e)
      {
        System.out.println(e);
        throw(e);
      }
      return retString;
    }

    private static void setRunPrm(String[] prm)
    {
      for (int i=0; i<prm.length; i++)
      {
        if (prm[i].toLowerCase().startsWith("-verbose"))
          verbose = isolatePrmValue(prm[i], "-verbose");
        else if (prm[i].toLowerCase().startsWith("-help"))
        {
          System.out.println("Usage is:");
          System.out.println("¥tjava B2BDemo.Retailer.MessageBroker");
          System.out.println("¥tparameters can be -dbURL -verbose, -step, -help");
          System.out.println("¥tdbURL contains a string like
jdbc:oracle:thin:@localhost:1521:ORCL");
          System.out.println("¥tparameters values can be (except for -help):");
          System.out.println("¥t¥tnone - equivalent to 'y'");
          System.out.println("¥t¥ty");
          System.out.println("¥t¥ttrue - equivalent to 'y'");
          System.out.println("¥t¥tn");
          System.out.println("¥t¥tfalse - equivalent to 'n'");
          System.out.println("¥t¥t-step can take a value in milliseconds");
          System.exit(0);
        }
        else if (prm[i].toLowerCase().startsWith("-step"))
        {
          String s = getPrmValue(prm[i], "-step");
          try
          {
            pauseTime = new Integer(s);
            System.out.println("Timeout " + pauseTime);
            stepByStep = true;
```

```
      }
      catch (NumberFormatException nfe)
      {
        pauseTime = null;
        if (s.toUpperCase().equals("Y") || s.toUpperCase().equals("TRUE"))
          stepByStep = true;
        else
          stepByStep = false;
      }
    }
    else if (prm[i].toLowerCase().startsWith("-dburl"))
    {
      url = getPrmValue(prm[i], "-dbURL");
    }
    else
      System.err.println("Unknown parameter [" + prm[i] + "], ignored.");
  }
}

private static boolean isolatePrmValue(String s, String p)
{
  boolean ret = true;
  if (s.length() > (p.length() + 1)) // +1 : "="
  {
    if (s.indexOf("=") > -1)
    {
      String val = s.substring(s.indexOf("=") + 1);
      if (val.toUpperCase().equals("Y") || val.toUpperCase().equals("TRUE"))
        ret = true;
      else if (val.toUpperCase().equals("N") || val.toUpperCase().equals("FALSE"))
        ret = false;
      else
      {
        System.err.println("Unrecognized value for " + p + ", set to y");
        ret = true;
      }
    }
  }
  return ret;
}

private static String getPrmValue(String s, String p)
{
  String ret = "";
  if (s.length() > (p.length() + 1)) // +1 : "="
  {
    if (s.indexOf("=") > -1)
```

```
        {
          ret = s.substring(s.indexOf("=") + 1);
        }
      }
      return ret;
    }

    /**
     * main
     * @param args
     */
    public static void main(String[] args)
    {
      if (args.length > 0)
        setRunPrm(args);
      UpdateMaster updateMaster = new UpdateMaster();
    }
}
```

AQ Broker - トランスフォーマおよびアドバンスト・キューイングのスクリプト

AQ Broker - トランスフォーマは、次のスクリプトを使用します。

- Java の例 18: AQ Broker による単一の AQ スレッドのリスニング ー BrokerThread.java
- Java の例 19: MessageBroker.java
- Java の例 20: AQReader.java
- Java の例 21: AQWriter.java
- Java の例 22: B2BMessage.java
- Java の例 23: ReadStructAQ.java
- Java の例 24: StopAllQueues.java
- Java の例 25: WriteStructAQ.java

Java の例 18: AQ Broker による単一の AQ スレッドのリスニング ー BrokerThread.java

```
package B2BDemo.Broker;

import java.sql.*;
import oracle.AQ.*;
import java.io.*;
import oracle.sql.*;
import oracle.jdbc.driver.*;

import B2BDemo.AQUtil.*;
import B2BDemo.XMLUtil.*;
import B2BDemo.*;

/**
 * This class implements a thread listening on one AQ.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class BrokerThread extends Thread
{
  private BufferedReader stdin = new BufferedReader(new
InputStreamReader(System.in));

  private static boolean stepByStep = false;
  private static boolean verbose    = false;

  AQReader aqReader;
  AQWriter aqWriter;
  String threadName;
  XSLTWrapper wrapper;
  Connection conn;
  Integer pause;
  XMLFrame frame;
  /**
   * Constructor
   */
  public BrokerThread(String name,
                      AQReader aqr,
                      AQWriter aqw,
                      XSLTWrapper wrap,
                      Connection c,
                      boolean v,
                      boolean s,
                      Integer p,
                      XMLFrame f)
```

```
  {
    this.aqReader = aqr;
    this.aqWriter = aqw;
    this.threadName = name;
    this.wrapper = wrap;
    this.conn = c;

    this.verbose = v;
    this.stepByStep = s;
    this.pause = p;
    this.frame = f;
  }

  public void run()
  {
    boolean go = true;
    while (go)
    {
      B2BMessage sm = this.aqReader.readQ();
      if (verbose)
        System.out.println(this.threadName + " Recieved¥nFrom > " + sm.getFrom() +
                                   "¥nTo   > " + sm.getTo() +
                                   "¥nType > " + sm.getType() +
                                   "¥nContent >¥n" + sm.getContent());
      else
        System.out.println(this.threadName + " Recieved¥nFrom > " + sm.getFrom()
+
                                   "¥nTo   > " + sm.getTo() +
                                   "¥nType > " + sm.getType());
      String xmlDoc = sm.getContent();
      if (xmlDoc != null && xmlDoc.length() > 0)
      {
        try { this.frame.setXMLDocument(sm.getContent()); }
        catch (Exception e)
        {
          e.printStackTrace();
        }
      }
      if (stepByStep)
      {
        if (pause != null)
        {
          System.out.println("Waiting for " + pause.longValue() + "
                                                                  milliseconds");
          try { Thread.sleep(pause.longValue()); } catch (InterruptedException
                                                                       e) {}
        }
```

```
  else
    try { String s = _userInput("[Hit return to continue]"); } catch
                                                        (Exception e) {}
}
if (sm.getType().length() >= MessageHeaders.EXIT.length() &&
                              sm.getType().equals(MessageHeaders.EXIT))
  go = false;
else
{
  // Transform !
  String processedXMLDoc = "";
  String xslDoc = getXSL(sm.getFrom(),
                         sm.getTo(),
                         sm.getType());
  if (verbose)
    System.out.println("Read:¥n" + xslDoc);
  try
  {
    processedXMLDoc = wrapper.processTransformation(sm.getContent(),
                                       xslDoc /*defaultStyleSheet*/);
    if (verbose)
      System.out.println("¥nResult :¥n" + processedXMLDoc);
    System.out.println("Transformation done.");
  }
  catch (Exception e)
  {
    System.err.println("Ooops...¥n");
    e.printStackTrace();
  }
  if (stepByStep)
  {
    if (pause != null)
    {
      System.out.println("Waiting for " + pause.longValue() + "
                                                        milliseconds");
      try { Thread.sleep(pause.longValue()); } catch (InterruptedException
                                                        e) {}
    }
    else
      try { String s = _userInput("[Hit return to continue]"); } catch
                                                        (Exception e) {}
  }

  // Send new document to destination
  this.aqWriter.writeQ(new B2BMessage(sm.getFrom(),
                                      sm.getTo(),
                                      sm.getType(),
```

```
                                                processedXMLDoc));
          this.aqWriter.flushQ();
        }
      }
      if (frame.isVisible())
        frame.setVisible(false);
      System.exit(0);
    }

    private String getXSL(String from,
                          String to,
                          String task)
    {
      if (verbose)
        System.out.println("Processing From " + from + " to " + to + " for " + task);
      String xsl = "";
      String stmt = "SELECT XSL FROM STYLESHEETS WHERE APPFROM = ? AND APPTO = ? AND
OP = ?";

      try
      {
        PreparedStatement pStmt = conn.prepareStatement(stmt);
        pStmt.setString(1, from);
        pStmt.setString(2, to);
        pStmt.setString(3, task);
        ResultSet rSet = pStmt.executeQuery();
        while (rSet.next())
          xsl = _dumpClob(conn, ((OracleResultSet)rSet).getCLOB(1));
        rSet.close();
        pStmt.close();
      }
      catch (SQLException e)
      { }
      catch (Exception e)
      { }
      return xsl;
    }

    static String _dumpClob (Connection conn, CLOB clob)
      throws Exception
    {
      String returnStr = "";

      OracleCallableStatement cStmt1 =
        (OracleCallableStatement)
          conn.prepareCall ("begin ? := dbms_lob.getLength (?); end;");
      OracleCallableStatement cStmt2 =
        (OracleCallableStatement)
```

```
        conn.prepareCall ("begin dbms_lob.read (?, ?, ?, ?); end;");

    cStmt1.registerOutParameter (1, Types.NUMERIC);
    cStmt1.setClob (2, clob);
    cStmt1.execute ();

    long length = cStmt1.getLong (1);
    long i = 0;
    int chunk = 100;
    if (verbose)
      System.out.println("Length to read from DB : " + length);

    while (i < length)
    {
      cStmt2.setClob (1, clob);
      cStmt2.setLong (2, chunk);
      cStmt2.registerOutParameter (2, Types.NUMERIC);
      cStmt2.setLong (3, i + 1);
      cStmt2.registerOutParameter (4, Types.VARCHAR);
      cStmt2.execute ();

      long readThisTime = cStmt2.getLong (2);
      String stringThisTime = cStmt2.getString (4);

//    System.out.print ("Read " + read_this_time + " chars: ");
      returnStr += stringThisTime;
      i += readThisTime;
    }

    cStmt1.close ();
    cStmt2.close ();

    return returnStr;
  }

  private String _userInput(String prompt) throws Exception
  {
    String retString;
    System.out.print(prompt);
    try { retString = stdin.readLine(); }
    catch (Exception e)
    {
      System.out.println(e);
      throw(e);
    }
    return retString;
  }
}
```

Java の例 19: MessageBroker.java

```
package B2BDemo.Broker;
/**
 * Implements the AQ Broker-Transformer.
 * This "Message Broker" uses 4 message queues, provided by
 * Oracle8i Advanced Queuing.
 * AQ Broker uses the threads described in BrokerThread
 * Each thread is waiting on one queue, and writing on another.
 * The message broker uses - for this demo - two threads :
 *   One from retailer to supplier
 *   One from supplier to retailer
 * 2 Threads := 4 queues
 *
 * When a message is recieved, the broker knows :
 *   where it comes from (origin)
 *   where it goes to (destination)
 *   what for (operation)
 * Those three elements are used as the primary key for the
 * stylesheet table (belonging to the AQ schema) to fetch the
 * right sXSL Stylesheet from the database, in order to turn
 * the incoming document into an outgoing one fitting the requirements
 * of the destination application.
 *
 * @see BrokerThread
 * @see B2BMessage
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.sql.*;
import oracle.AQ.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import oracle.sql.*;
import oracle.jdbc.driver.*;
//import oracle.bali.ewt.border.UIBorderFactory;
//import oracle.bali.ewt.olaf.OracleLookAndFeel;

import B2BDemo.AQUtil.*;
import B2BDemo.*;
import B2BDemo.XMLUtil.*;

public class MessageBroker extends Object
{
  private static boolean stepByStep = false;
  private static boolean verbose    = false;
```

```
  private static Integer pauseTime  = null;

  XSLTWrapper wrapper = null;

  // To get the style sheet from its CLOB
  Connection conn = null;
  String userName = AppCste.AQuser;
  String password = AppCste.AQpswd;
  String dbUrl    = AppCste.AQDBUrl;

  public MessageBroker()
  {
    XMLFrame frame = new XMLFrame("Message Broker");
    /**
    try
    {
      OracleLookAndFeel.setColorScheme(Color.cyan);
//    OracleLookAndFeel.setColorScheme("Titanium");
      UIManager.setLookAndFeel(new OracleLookAndFeel());
      SwingUtilities.updateComponentTreeUI(frame);
      frame.setBackground(UIManager.getColor("darkIntensity"));
    }
    catch (Exception e)
    {
      System.err.println("Exception for Oracle Look and Feel:" + e );
    }
    */
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
    {
      frameSize.height = screenSize.height;
    }
    /**
    if (frameSize.width > screenSize.width)
    {
      frameSize.width = screenSize.width;
    }
    */
    frameSize.width = screenSize.width / 3;
//  frame.setLocation((screenSize.width - frameSize.width)/2, (screenSize.height -
frameSize.height)/2);
    frame.setLocation(frameSize.width, (screenSize.height - frameSize.height)/2);
//  frame.addWindowListener(new WindowAdapter() { public void
windowClosing(WindowEvent e) { System.exit(0); } });
    frame.setVisible(true);
```

```
AQReader aqr = null;
AQWriter aqw = null;
// Initialize AQ reader and writer
aqw = new AQWriter(AppCste.AQuser,
                   AppCste.AQpswd,
                   AppCste.AQDBUrl,
                   "AppTwo_QTab",
                   "AppTwoMsgQueue");
aqr = new AQReader(AppCste.AQuser,
                   AppCste.AQpswd,
                   AppCste.AQDBUrl,
                   "AppOne_QTab",
                   "AppOneMsgQueue");
wrapper = new XSLTWrapper();
if (conn == null)
  _getConnected();

BrokerThread retail2supply = new BrokerThread("Retail to Supply",
                                              aqr,
                                              aqw,
                                              wrapper,
                                              conn,
                                              verbose,
                                              stepByStep,
                                              pauseTime,
                                              frame);
aqw = new AQWriter(AppCste.AQuser,
                   AppCste.AQpswd,
                   AppCste.AQDBUrl,
                   "AppFour_QTab",
                   "AppFourMsgQueue");
aqr = new AQReader(AppCste.AQuser,
                   AppCste.AQpswd,
                   AppCste.AQDBUrl,
                   "AppThree_QTab",
                   "AppThreeMsgQueue");
BrokerThread supply2retail = new BrokerThread("Supply to Retail",
                                              aqr,
                                              aqw,
                                              wrapper,
                                              conn,
                                              verbose,
                                              stepByStep,
                                              pauseTime,
                                              frame);
```

```
  retail2supply.start();
  supply2retail.start();

  System.out.println("<ThreadsOnTheirWay/>");
}

private void _getConnected()
{
  try
  {
    Class.forName ("oracle.jdbc.driver.OracleDriver");
    conn = DriverManager.getConnection (dbUrl, userName, password);
  }
  catch (Exception e)
  {
    System.out.println("Get connected failed : " + e);
    System.exit(1);
  }
}

private static void setRunPrm(String[] prm)
{
  for (int i=0; i<prm.length; i++)
  {
    if (prm[i].toLowerCase().startsWith("-verbose"))
      verbose = isolatePrmValue(prm[i], "-verbose");
    else if (prm[i].toLowerCase().startsWith("-help"))
    {
      System.out.println("Usage is:");
      System.out.println("¥tjava Intel.iDevelop.MessageBroker");
      System.out.println("¥tparameters can be -verbose, -step, -help");
      System.out.println("¥tparameters values can be (except for -help):");
      System.out.println("¥t¥tnone - equivalent to 'y'");
      System.out.println("¥t¥ty");
      System.out.println("¥t¥ttrue - equivalent to 'y'");
      System.out.println("¥t¥tn");
      System.out.println("¥t¥tfalse - equivalent to 'n'");
      System.out.println("¥t¥t-step can take a value in milliseconds");
      System.exit(0);
    }
    else if (prm[i].toLowerCase().startsWith("-step"))
    {
      String s = getPrmValue(prm[i], "-step");
      try
      {
        pauseTime = new Integer(s);
        System.out.println("Timeout " + pauseTime);
```

```
        stepByStep = true;
      }
      catch (NumberFormatException nfe)
      {
        pauseTime = null;
        if (s.toUpperCase().equals("Y") || s.toUpperCase().equals("TRUE"))
          stepByStep = true;
        else
          stepByStep = false;
      }
    }
    else
      System.err.println("Unknown parameter [" + prm[i] + "], ignored.");
  }
}

private static boolean isolatePrmValue(String s, String p)
{
  boolean ret = true;
  if (s.length() > (p.length() + 1)) // +1 : "="
  {
    if (s.indexOf("=") > -1)
    {
      String val = s.substring(s.indexOf("=") + 1);
      if (val.toUpperCase().equals("Y") || val.toUpperCase().equals("TRUE"))
        ret = true;
      else if (val.toUpperCase().equals("N") || val.toUpperCase().equals("FALSE"))
        ret = false;
      else
      {
        System.err.println("Unrecognized value for " + p + ", set to y");
        ret = true;
      }
    }
  }
  return ret;
}

private static String getPrmValue(String s, String p)
{
  String ret = "";
  if (s.length() > (p.length() + 1)) // +1 : "="
  {
    if (s.indexOf("=") > -1)
    {
      ret = s.substring(s.indexOf("=") + 1);
    }
```

```
    }
    return ret;
  }

  public static void main(String args[])
  {
    // java B2BDemo.OrderEntry.MessageBroker -verbose[=[y|true|n|false]]
-step[=[y|true|n|false]] -help
    if (args.length > 0)
      setRunPrm(args);

    new MessageBroker();
  }
}
```

Java の例 20: AQReader.java

```
package B2BDemo.AQUtil;
/**
 * This class is a wrapper around the Advanced Queuing facility of Oracle 8i.
 * Used to dequeue a message.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.sql.*;
import oracle.AQ.*;
import java.io.*;

public class AQReader extends Object
{
  Connection conn  = null;
  AQSession aqSess = null;

  String userName   = "";
  String qTableName = "";
  String qName      = "";

  AQQueueTable aqTable = null;
  AQQueue      aq      = null;

  public AQReader(String userName,
                  String password,
                  String url,
                  String qTable,
                  String qName)
  {
```

```
  this.userName = userName;
  this.qTableName = qTable;
  this.qName = qName;
  aqSess = createSession(userName, password, url);

  aqTable = aqSess.getQueueTable(userName, qTableName);
  System.out.println("Successful getQueueTable");
  // Handle to q
  aq = aqSess.getQueue(userName, qName);
  System.out.println("Successful getQueue");
}

public AQSession createSession(String userName,
                               String pswd,
                               String url)
{
  try
  {
    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
    conn = DriverManager.getConnection(url, userName, pswd);
    System.out.println("JDBC Connection opened");
    conn.setAutoCommit(false);
    // Load Oracle 8i AQ Driver
    Class.forName("oracle.AQ.AQOracleDriver");
    // Create the AQ Session
    aqSess = AQDriverManager.createAQSession(conn);
    System.out.println("AQ Session successfully created.");
  }
  catch (Exception e)
  {
    System.out.println("Exception : " + e);
    e.printStackTrace();
  }
  return aqSess;
}

public B2BMessage readQ() throws AQException
{
  AQMessage message;
  AQRawPayload rawPayload;

  // Read with REMOVE option
  AQDequeueOption dqOption = new AQDequeueOption();
  dqOption.setDequeueMode(AQDequeueOption.DEQUEUE_REMOVE);
  message = aq.dequeue(dqOption);

  System.out.println("Successfull dQueue");
```

```
    rawPayload = message.getRawPayload();

    try
    {
      conn.commit(); // Commit the REMOVE
    }
    catch (Exception sqle)
    {
      System.err.println(sqle.toString());
    }

    return (B2BMessage)deserializeFromByteArray(rawPayload.getBytes());
  }

  private static Object deserializeFromByteArray (byte[] b)
  {
    ByteArrayInputStream inputStream = new ByteArrayInputStream(b);
    try
    {
      ObjectInputStream ois = new ObjectInputStream(inputStream);
    return ois.readObject();
    }
    catch (Exception e)
    {
      System.err.println("deserializeFromByteArray failed :" + e);
      return null;
    }
  }
}
```

Java の例 21: AQWriter.java

```
package B2BDemo.AQUtil;
/**
 * This class is a wrapper around the Advanced Queuing facility of Oracle 8i.
 * Used to enqueue a message.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.sql.*;
import oracle.AQ.*;
import java.io.*;

public class AQWriter extends Object
{
  Connection conn = null;
```

```
AQSession aqSess = null;

String userName   = "";
String qTableName = "";
String qName      = "";

public AQWriter(String userName,
                String password,
                String url,
                String qTable,
                String qName)
{
  this.userName = userName;
  this.qTableName = qTable;
  this.qName = qName;
  aqSess = createSession(userName, password, url);
}

public void flushQ()
{
  if (conn != null)
  {
    try { conn.commit(); } catch (SQLException e) {}
  }
}

public void closeConnection()
{
  if (conn != null)
  {
    try { conn.close(); }
    catch (SQLException e)
    { }
  }
}

public AQSession createSession(String userName,
                               String pswd,
                               String url)
{
  try
  {
    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
    conn = DriverManager.getConnection(url, userName, pswd);
    System.out.println("JDBC Connection opened");
    conn.setAutoCommit(false);
    // Load Oracle 8i AQ Driver
```

```
    Class.forName("oracle.AQ.AQOracleDriver");
    // Create the AQ Session
    aqSess = AQDriverManager.createAQSession(conn);
    System.out.println("AQ Session successfully created.");
  }
  catch (Exception e)
  {
    System.out.println("Exception : " + e);
    e.printStackTrace();
  }
  return aqSess;
}

public void writeQ(B2BMessage sm) throws AQException
{
  AQQueueTable         qTable;
  AQQueue              q;

  qTable = aqSess.getQueueTable(userName, qTableName);
  System.out.println("Successful getQueueTable");
  // Handle to q
  q = aqSess.getQueue(userName, qName);
  System.out.println("Successful getQueue");

  // Q is identified, let's write
  AQMessage message;
  AQRawPayload rawPayload;

  message = q.createMessage();
  byte[] bArray = serializeToByteArray(sm);
  rawPayload = message.getRawPayload();
  rawPayload.setStream(bArray, bArray.length);
  AQEnqueueOption eqOption = new AQEnqueueOption();
  q.enqueue(eqOption, message);
}

private static byte[] serializeToByteArray (Object o)
{
  ByteArrayOutputStream outStream = new ByteArrayOutputStream();
  try
  {
    ObjectOutputStream oos = new ObjectOutputStream(outStream);
    oos.writeObject(o);
    return outStream.toByteArray();
  }
  catch (Exception e)
  {
```

```
      System.err.println("serialize2ByteArray failed : " + e);
      return null;
    }
  }
}
```

Java の例 22: B2BMessage.java

```
package B2BDemo.AQUtil;
/**
 * This class decsribes the structure of the messages used in this demo
 * Subject to changes in 817
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.io.Serializable;

public class B2BMessage extends Object implements Serializable
{
  String from;
  String to;
  String type;
  String content;

  public B2BMessage(String f,
                    String t,
                    String typ,
                    String c)
  {
    this.from    = f;
    this.to      = t;
    this.type    = typ;
    this.content = c;
  }

  public String getFrom()
  { return this.from; }
  public String getTo()
  { return this.to; }
  public String getType()
  { return this.type; }
  public String getContent()
  { return this.content; }

}
```

Java の例 23: ReadStructAQ.java

```
package B2BDemo.AQUtil;
/**
 * A main for tests - Not used in the demo itself.
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
import java.sql.*;
import oracle.AQ.*;
import java.io.*;

import B2BDemo.*;

public class ReadStructAQ extends Object
{
  public static void main(String[] args)
  {
    AQReader aqr = new AQReader(AppCste.AQuser,
                                AppCste.AQpswd,
                                AppCste.AQDBUrl,
                                "objMsgsStruct_QTab",
                                "structMsgQueue");
    // Loop while EXIT is not recieved
    boolean goLoop = true;
    while (goLoop)
    {
      B2BMessage sm = aqr.readQ();
      System.out.println("Recieved¥nFrom > " + sm.getFrom() +
                                "¥nTo   > " + sm.getTo() +
                                "¥nType > " + sm.getType() +
                                "¥nContent >¥n" + sm.getContent());
      if (sm.getType().equals("EXIT"))
        goLoop = false;
    }
    System.out.println("<bye/>");
  }
}
```

Java の例 24: StopAllQueues.java

```
package B2BDemo.AQUtil;

import B2BDemo.*;

/**
 * Used in the demo to stop the queues and the applications waiting on them.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class StopAllQueues extends Object
{
  /**
   * Constructor
   */
  public StopAllQueues()
  {
     AQWriter aqw1 = new AQWriter(AppCste.AQuser,
                                  AppCste.AQpswd,
                                  AppCste.AQDBUrl,
                                  "AppOne_QTab",
                                  "AppOneMsgQueue");
     aqw1.writeQ(new B2BMessage(MessageHeaders.APP_B,
                                MessageHeaders.APP_A,
                                MessageHeaders.EXIT,
                                ""));
     aqw1.flushQ();
     AQWriter aqw2 = new AQWriter(AppCste.AQuser,
                                  AppCste.AQpswd,
                                  AppCste.AQDBUrl,
                                  "AppTwo_QTab",
                                  "AppTwoMsgQueue");
     aqw2.writeQ(new B2BMessage(MessageHeaders.APP_B,
                                MessageHeaders.APP_A,
                                MessageHeaders.EXIT,
                                ""));
     aqw2.flushQ();
     AQWriter aqw3 = new AQWriter(AppCste.AQuser,
                                  AppCste.AQpswd,
                                  AppCste.AQDBUrl,
                                  "AppThree_QTab",
                                  "AppThreeMsgQueue");
     aqw3.writeQ(new B2BMessage(MessageHeaders.APP_B,
                                MessageHeaders.APP_A,
                                MessageHeaders.EXIT,
                                ""));
```

```
    aqw3.flushQ();
    AQWriter aqw4 = new AQWriter(AppCste.AQuser,
                                 AppCste.AQpswd,
                                 AppCste.AQDBUrl,
                                 "AppFour_QTab",
                                 "AppFourMsgQueue");
    aqw4.writeQ(new B2BMessage(MessageHeaders.APP_B,
                               MessageHeaders.APP_A,
                               MessageHeaders.EXIT,
                               ""));
    aqw4.flushQ();
  }

  /**
   * main
   * @param args
   */
  public static void main(String[] args)
  {
    StopAllQueues stopAllQueues = new StopAllQueues();
  }
}
```

Java の例 25: WriteStructAQ.java

```
package B2BDemo.AQUtil;
/**
 * A Main for tests - Not used in the demo itself.
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
/**
 * A main for tests
 */
import java.sql.*;
import oracle.AQ.*;
import java.io.*;

import B2BDemo.*;

public class WriteStructAQ extends Object
{
  private static BufferedReader stdin = new BufferedReader(new
InputStreamReader(System.in));
```

```
static AQWriter aqw = null;

public static void main(String[] args)
{
  try
  {
    aqw = new AQWriter(AppCste.AQuser,
                       AppCste.AQpswd,
                       AppCste.AQDBUrl,
                       "objMsgsStruct_QTab",
                       "structMsgQueue");
    String messSubject     = "";
    String messTxt         = "";
    String messOrigin      = "";
    String messDestination = "";
    try
    {
      messOrigin      = userInput("Message Origin      > ");
      messDestination = userInput("Message Destination > ");
      messSubject     = userInput("Message Subject     > ");
      messTxt         = userInput("Message Text        > ");
    } catch (Exception e){}

    // Write the queue
    B2BMessage sm = new B2BMessage(messOrigin,
                                   messDestination,
                                   messSubject,
                                   messTxt);
    aqw.writeQ(sm);
    try { String s = userInput("Written"); }
    catch (Exception ne) {}
    aqw.closeConnection();
    try { String s = userInput("Closed !"); }
    catch (Exception ne) {}
  }
  catch (Exception e)
  {
    System.err.println("Arghh : " + e);
    e.printStackTrace();
    try { String s = userInput("..."); }
    catch (Exception ne) {}
  }
}

private static String userInput(String prompt) throws Exception
{
  String retString;
```

```
    System.out.print(prompt);
    try { retString = stdin.readLine(); }
    catch (Exception e)
    {
      System.out.println(e);
      throw(e);
    }
    return retString;
  }
}
```

サプライヤ・スクリプト

サプライヤは、次のスクリプトを使用します。

- Java スクリプトの例 26: SupplierFrame.java
- Java の例 27: リテーラから受信した注文によるエージェントの起動ー SupplierWatcher.java

Java スクリプトの例 26: SupplierFrame.java

```
package B2BDemo.Supplier;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

import java.io.*;
import java.util.*;
import java.net.*;
import java.sql.*;

import oracle.xml.sql.query.*;
import oracle.xml.sql.dml.*;

import org.w3c.dom.*;
import oracle.xml.parser.v2.*;
import org.xml.sax.*;

import B2BDemo.AQUtil.*;
import B2BDemo.*;
import B2BDemo.XMLUtil.*;

/**
```

```
 * This class implements the Frame suggesting to ship the order.'
 *
 * @see SupplierWatcher in the same package.
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class SupplierFrame extends JFrame
{
  private BufferedReader stdin = new BufferedReader(new
InputStreamReader(System.in));
  private static boolean stepByStep = false;
  private static boolean verbose    = false;
  private static Integer pause      = null;
  private XMLFrame frame            = null;

  AQReader aqr;
  XMLtoDMLv2 x2d = null;

  String userName = "supplier";
  String password = "supplier";
  String url      = null;

  String currOrdId = "";

  DOMParser parser = new DOMParser();

  AQWriter aqw = null;

  BorderLayout borderLayout1 = new BorderLayout();
  JPanel jPanel1 = new JPanel();
  BorderLayout borderLayout2 = new BorderLayout();
  JPanel southPanel = new JPanel();
  JButton shipButton = new JButton();
  JPanel centerPanel = new JPanel();
  JLabel ordMessage = new JLabel();

  /**
   * Constructs a new instance.
   */
  public SupplierFrame(boolean v, boolean s, Integer p, XMLFrame f, String url)
  {
    super();
    this.verbose = v;
    this.stepByStep = s;
    this.pause = p;
    this.frame = f;
    this.url = url;
    try
```

```
  {
    jbInit();
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
}

/**
 * Initializes the state of this instance.
 */
private void jbInit() throws Exception
{
  this.getContentPane().setLayout(borderLayout1);
  this.setSize(new Dimension(400, 300));
  shipButton.setText("Ship Order");
  shipButton.setEnabled(false);
  shipButton.addActionListener(new java.awt.event.ActionListener()
  {

    public void actionPerformed(ActionEvent e)
    {
      shipButton_actionPerformed(e);
    }
  });
  ordMessage.setText("Waiting for Orders");
  ordMessage.setFont(new Font("Dialog", 1, 20));
  jPanel1.setLayout(borderLayout2);
  this.setTitle("Supplier Watcher");
  this.getContentPane().add(jPanel1, BorderLayout.CENTER);
  jPanel1.add(southPanel, BorderLayout.SOUTH);
  southPanel.add(shipButton, null);
  jPanel1.add(centerPanel, BorderLayout.CENTER);
  centerPanel.add(ordMessage, null);
}

public void enterTheLoop()
{
    // Initialize AQ reader
  aqr = new AQReader(AppCste.AQuser,
                     AppCste.AQpswd,
                     AppCste.AQDBUrl,
                     "AppTwo_QTab",
                     "AppTwoMsgQueue");
  // Initialize XSL Transformer
  x2d = new XMLtoDMLv2(userName,
```

```
                        password,
                        url);
    // Initialize the AQ Writer
    aqw = new AQWriter(AppCste.AQuser,
                       AppCste.AQpswd,
                       AppCste.AQDBUrl,
                       "AppThree_QTab",
                       "AppThreeMsgQueue");

    boolean go = true;
    while (go)
    {
      String ordIdValue = "";
      String custIdValue = "";
      B2BMessage sm = aqr.readQ();
      if (verbose)
        System.out.println("Recieved¥nFrom > " + sm.getFrom() +
                                  "¥nTo   > " + sm.getTo() +
                                  "¥nType > " + sm.getType() +
                                  "¥nContent >¥n" + sm.getContent());
      else
        System.out.println("Recieved¥nFrom > " + sm.getFrom() +
                                  "¥nTo   > " + sm.getTo() +
                                  "¥nType > " + sm.getType());
      String xmlDoc = sm.getContent();
      if (xmlDoc != null && xmlDoc.length() > 0)
      {
        try { this.frame.setXMLDocument(sm.getContent()); }
        catch (Exception e)
        {
          e.printStackTrace();
        }
      }
      if (stepByStep)
      {
        if (pause != null)
        {
          System.out.println("Waiting for " + pause.longValue() + " milliseconds");
          try { Thread.sleep(pause.longValue()); } catch (InterruptedException e) {}
        }
        else
          try { String s = _userInput("[Hit return to continue]"); } catch
(Exception e) {}
      }
      if (sm.getType().equals(MessageHeaders.EXIT))
        go = false;
      else
```

```
      {
        System.out.println("Inserting");
        TableInDocument d[] = new TableInDocument[2];
        d[0] = new TableInDocument("ROWSET", "ROW", "ORD");
        d[1] = new TableInDocument("ITEMS", "ITEM_ROW", "LINE_ITEM");
        try
        {
          String XMLDoc = sm.getContent();
          x2d.insertFromXML(d, XMLDoc);
          System.out.println("Document processed.");
          // We want to read elements
          parser.setValidationMode(false);
          try
          {
            parser.parse(new InputSource(new
ByteArrayInputStream(XMLDoc.getBytes())));
            XMLDocument xml = parser.getDocument();
            XMLElement elmt = (XMLElement)xml.getDocumentElement();
            NodeList nl = elmt.getElementsByTagName("ID"); // ORD ID
            for (int i=0; i<nl.getLength(); i++)
            {
              XMLElement ordId = (XMLElement)nl.item(i);
              XMLNode theText = (XMLNode)ordId.getFirstChild();
              ordIdValue = theText.getNodeValue();
              currOrdId = ordIdValue;
              break; // Just the first one !!!
            }
            nl = elmt.getElementsByTagName("CUSTOMER_ID"); // CUSTOMER ID
            for (int i=0; i<nl.getLength(); i++)
            {
              XMLElement ordId = (XMLElement)nl.item(i);
              XMLNode theText = (XMLNode)ordId.getFirstChild();
              custIdValue = theText.getNodeValue();
            }
          }
          catch (SAXParseException e)
          {
            System.out.println(e.getMessage());
          }
          catch (SAXException e)
          {
            System.out.println(e.getMessage());
          }
          catch (Exception e)
          {
            System.out.println(e.getMessage());
          }
```

```
        }
        catch (Exception e)
        {
          System.err.println("Ooops:¥n" + e);
        }
        this.shipButton.setEnabled(true);
        String custName = "";
        try
        {
          PreparedStatement pStmt = x2d.getConnection().prepareStatement("Select
C.NAME from CUSTOMER C where C.ID = ?");
          pStmt.setString(1, custIdValue);
          ResultSet rSet = pStmt.executeQuery();
          while (rSet.next())
            custName = rSet.getString(1);
          rSet.close();
          pStmt.close();
        }
        catch (SQLException e)
        {}

        this.ordMessage.setText("Order [" + ordIdValue + "] to process for [" +
custName + "]");
        JOptionPane.showMessageDialog(this, "New Order Pending !", "Wake Up !",
                                      JOptionPane.INFORMATION_MESSAGE);
      }
    }
    frame.setVisible(false);
  }

  void shipButton_actionPerformed(ActionEvent e)
  {
    // Send message
    String doc2send = "<SHIP>" + currOrdId + "</SHIP>";
    // sending XMLDoc in the Queue
    aqw.writeQ(new B2BMessage(MessageHeaders.APP_B,
                              MessageHeaders.APP_A,
                              MessageHeaders.UPDATE_ORDER,
                              doc2send));
    aqw.flushQ();  // Commit !

    // Disable Button
    this.shipButton.setEnabled(false);
    // display wait message
    this.ordMessage.setText("Waiting for orders...");
  }
```

```
  private String _userInput(String prompt) throws Exception
  {
    String retString;
    System.out.print(prompt);
    try { retString = stdin.readLine(); }
    catch (Exception e)
    {
      System.out.println(e);
      throw(e);
    }
    return retString;
  }
}
```

Java の例 27: リテーラから受信した注文によるエージェントの起動ー SupplierWatcher.java

```
package B2BDemo.Supplier;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//import oracle.bali.ewt.border.UIBorderFactory;
//import oracle.bali.ewt.olaf.OracleLookAndFeel;

import B2BDemo.XMLUtil.*;

/**
 * This class implements the agent waiting on the queue where the orders are
delivered.
 * When a message is read, the agent "wakes up" and suggests to ship the order.
 * Shipping the order will then fire a new B2B process to update the status of
 * the order in the Retailer database.
 *
 * @author Olivier LE DIOURIS - Partner Technical Services - Oracle Copr.
 */
public class SupplierWatcher
{
  private static boolean stepByStep = false;
  private static boolean verbose    = false;
  private static Integer pauseTime  = null;
  private static String url      = "jdbc:oracle:thin:@localhost:1521:ORCL";  //
Default value
```

```
  /**
   * Constructor
   */
  public SupplierWatcher()
  {
    XMLFrame xmlFrame = new XMLFrame("Supplier");
    /*
    try
    {
      OracleLookAndFeel.setColorScheme(Color.cyan);
//    OracleLookAndFeel.setColorScheme("Titanium");
      UIManager.setLookAndFeel(new OracleLookAndFeel());
      SwingUtilities.updateComponentTreeUI(xmlFrame);
      xmlFrame.setBackground(UIManager.getColor("darkIntensity"));
    }
    catch (Exception e)
    {
      System.err.println("Exception for Oracle Look and Feel:" + e );
    }
    */
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = xmlFrame.getSize();
    if (frameSize.height > screenSize.height)
    {
      frameSize.height = screenSize.height;
    }
    /**
    if (frameSize.width > screenSize.width)
    {
      frameSize.width = screenSize.width;
    }
    */
    frameSize.width = screenSize.width / 3;
//  xmlFrame.setLocation((screenSize.width - frameSize.width)/2, (screenSize.height
- frameSize.height)/2);
    xmlFrame.setLocation((2 * frameSize.width), (screenSize.height -
frameSize.height)/2);
//  xmlFrame.addWindowListener(new WindowAdapter() { public void
windowClosing(WindowEvent e) { System.exit(0); } });
    xmlFrame.setVisible(true);

    SupplierFrame frame = new SupplierFrame(verbose, stepByStep, pauseTime,
xmlFrame, url);
    /*
    try
    {
```

```
      OracleLookAndFeel.setColorScheme(Color.cyan);
//    OracleLookAndFeel.setColorScheme("Titanium");
      UIManager.setLookAndFeel(new OracleLookAndFeel());
      SwingUtilities.updateComponentTreeUI(frame);
      frame.setBackground(UIManager.getColor("darkIntensity"));
    }
    catch (Exception e)
    {
      System.err.println("Exception for Oracle Look and Feel:" + e );
    }
    */
    //Center the window
    screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
    {
      frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width)
    {
      frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width)/2, (screenSize.height -
frameSize.height)/2);
//  frame.addWindowListener(new WindowAdapter() { public void
windowClosing(WindowEvent e) { System.exit(0); } });
    frame.setVisible(true);

    frame.enterTheLoop();
    frame.setVisible(false);
    xmlFrame.setVisible(false);
    System.exit(1);
  }

  private static void setRunPrm(String[] prm)
  {
    for (int i=0; i<prm.length; i++)
    {
      if (prm[i].toLowerCase().startsWith("-verbose"))
        verbose = isolatePrmValue(prm[i], "-verbose");
      else if (prm[i].toLowerCase().startsWith("-help"))
      {
        System.out.println("Usage iB2BDemo.Supplier.MessageBroker");
        System.out.println("¥tparameters can be -dbURL -verbose, -step, -help");
        System.out.println("¥tdbURL contains a string like
jdbc:oracle:thin:@localhost:1521:ORCL");
        System.out.println("¥tparameters values can be (except for -help):");
```

```
      System.out.println("¥t¥tnone - equivalent to 'y'");
      System.out.println("¥t¥ty");
      System.out.println("¥t¥ttrue - equivalent to 'y'");
      System.out.println("¥t¥tn");
      System.out.println("¥t¥tfalse - equivalent to 'n'");
      System.out.println("¥t¥t-step can take a value in milliseconds");
      System.exit(0);
    }
    else if (prm[i].toLowerCase().startsWith("-step"))
    {
      String s = getPrmValue(prm[i], "-step");
      try
      {
        pauseTime = new Integer(s);
        System.out.println("Timeout " + pauseTime);
        stepByStep = true;
      }
      catch (NumberFormatException nfe)
      {
        pauseTime = null;
        if (s.toUpperCase().equals("Y") || s.toUpperCase().equals("TRUE"))
          stepByStep = true;
        else
          stepByStep = false;
      }
    }
    else if (prm[i].toLowerCase().startsWith("-dburl"))
    {
      url = getPrmValue(prm[i], "-dbURL");
    }
    else
      System.err.println("Unknown parameter [" + prm[i] + "], ignored.");
  }
}

private static boolean isolatePrmValue(String s, String p)
{
  boolean ret = true;
  if (s.length() > (p.length() + 1)) // +1 : "="
  {
    if (s.indexOf("=") > -1)
    {
      String val = s.substring(s.indexOf("=") + 1);
      if (val.toUpperCase().equals("Y") || val.toUpperCase().equals("TRUE"))
        ret = true;
      else if (val.toUpperCase().equals("N") || val.toUpperCase().equals("FALSE"))
        ret = false;
```

```
        else
        {
          System.err.println("Unrecognized value for " + p + ", set to y");
          ret = true;
        }
      }
    }
    return ret;
  }

  private static String getPrmValue(String s, String p)
  {
    String ret = "";
    if (s.length() > (p.length() + 1)) // +1 : "="
    {
      if (s.indexOf("=") > -1)
      {
        ret = s.substring(s.indexOf("=") + 1);
      }
    }
    return ret;
  }

  /**
   * main
   * @param args
   */
  public static void main(String[] args)
  {
    if (args.length > 0)
      setRunPrm(args);

    try
    {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
    new SupplierWatcher();
  }
}
```

9

Service Delivery Platform（SDP）と XML

この章の内容は、次のとおりです。

Oracle Service Delivery Platform

この章では、Oracle Service Delivery Platform（SDP）の概要について説明します。

SDP ビジネス・ソリューション

Business-to-Business（B2B）XML メッセージ・ペイロードを使用する多数のソリューションを、Service Delivery Platform（SDP）上で構築できます。これには、次の要素が含まれます。

- Phone Number Portability
- Wireless Number Portability およびデータ・サービス自動化
- NPAC
- サービス・ゲートウェイ
- ADSL
- Voice over IP

その他にも、帯域幅交換やキャリア事前選択などのソリューションがあります。

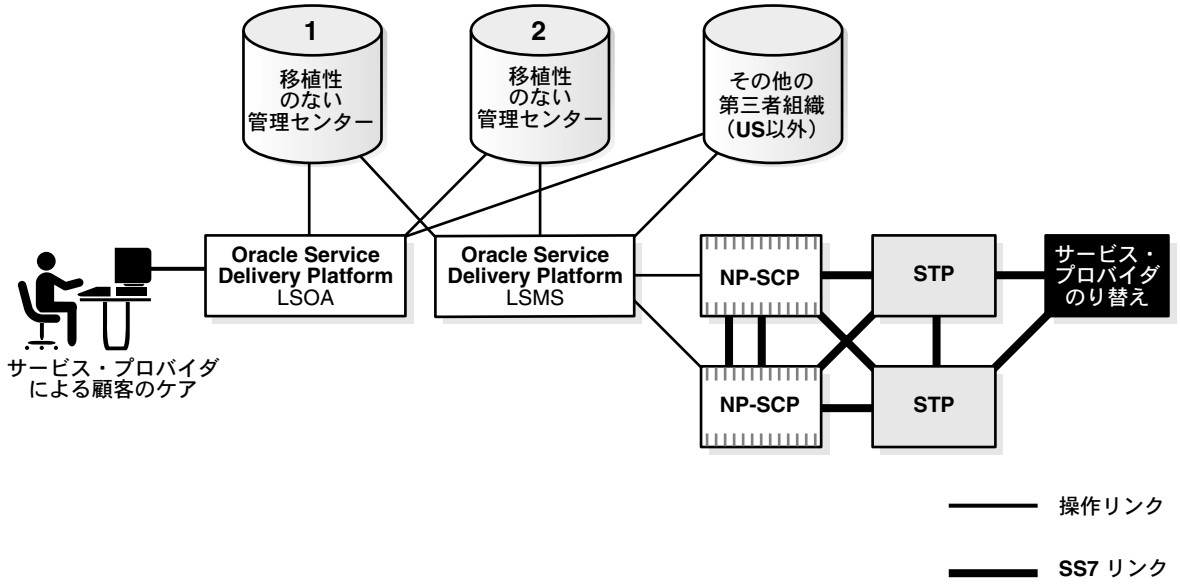
Phone Number Portability

Phone Number Portability（Number Portability）は、消費者が通信サービス・プロバイダを切り替えたり、物理的に移動したり、サービスを変更する場合に、その電話番号を追跡できるメカニズムです。その概念は、競争を促す規制当局によって提唱されたものであり、消費者が各自の電話番号を追跡できれば、サービス・プロバイダ間の移動に従来よりも関心を持つとしています。Number Portability は、アメリカで競争の激しい長距離電話市場で急成長が続く主要因として広く認知されています。

Number Portability メッセージ・ベース・アプリケーションは、*i*Message Studio、Event Manager およびアダプタを使用します。このアプリケーションは、メッセージ・ペイロードとして XML を使用し、通信業界では一般的な B2B プロトコルを使用して 2 つのサービス・プロバイダ間で通信しています。

Number Portability アプリケーションは、Oracle CRM Applications 11*i* のうち Oracle Service Delivery Platform（OSDP または SDP）のメッセージ機能およびイベント管理機能を具体的に示すものです。これは CRM 機能です。

図 9-1 Number Portability および関連アーキテクチャ



Number Portability プロセス

Number Portability は、次の作業を実行します。

1. XML または DTD 要素を SQL の表または PL/SQL のストアド・ファンクションにリンクします。
2. バックグラウンドでストアド・プロシージャを動的に作成および構築します。XML メッセージをさらに処理できるようにエンキューします。表から値を抽出または問合せするか、要素に対応付けられたストアド・ファンクションを動的に実行し、XML を構築します。
3. 実行時に、ユーザーまたはプログラムは、このように動的に実行されるプロシージャを実行します。このプロシージャはインテリジェンスを持ち、XML メッセージを作成して、さらに処理できるようにエンキューします。

Number Portability 製品は、ワークフロー・マネージャの役割を果たし、顧客から要求されたサービスを提供するために使用されます。

新しい電話サービスを申し込んだ場合の処理

たとえば、新しい電話サービスを申し込むと、電話会社は Oracle Order Management アプリケーションを使用して申込みを受け取り、情報を獲得します。

発生するイベントの流れを次に示します。Number Portability アプリケーションは、次のすべての処理で使用され、この過程でインスタンスとして機能します。

1. 顧客が、新しい電話の設置などのサービスを申し込みます。
2. 設置アプリケーションが受注を獲得し、指定された検証および認可プロセスを開始します。
3. 設置アプリケーションが外部システムと通信します。たとえば、顧客の信用照会を行ったり、その他の機関と対話して他のアクションを実行します。

   この通信には、2 つのシステム間で XML 形式で定義されたプロトコルを使用できます。Oracle Workflow が使用されるため、コンサルタントは実行時にグラフィカルなフォーマットでもビジネス・プロセスを構成して表示できます。

市内通話サービス・プロバイダを変更した場合の処理

Number Portability は、市内通話サービス・プロバイダを切り替える場合にも使用されます。この場合のプロセスは、次のとおりです。

1. 顧客が市内通話サービス・プロバイダに連絡します。
2. 市内通話サービス・プロバイダは、要求を古いサービス・プロバイダで検証します。これは、中立的立場の第三者仲介機関を通じて行われます。アメリカの場合、この第三者仲介機関は、NPAC（Number Portability Administration Center）です。
3. 長距離電話会社を変更する場合、仲介機関は音声を介してオンラインになります。市内通話サービス・プロバイダを切り替える場合、これは電子メッセージ機能を通じて行われます。
4. 新しいサービス・プロバイダは、NPAC にメッセージを送信します。NPAC は、要求を検証し、新しいサービス・プロバイダがこの新規顧客を獲得できるように、承認または認可します。
5. NPAC は古いプロバイダ（移植元）にメッセージを送信します。移植元は、注文を検討して承認し、XML を使用してメッセージを NPAC に送信します。
6. NPAC は新しいサービス・プロバイダ（移植先）に認可を送信します。
7. これで、注文が両者に承認されたことになります。

注意： ここで発生するすべてのメッセージ機能では、SDP の Number Portability 製品内のメイン・フォーマットとして XML が使用されます。他のプロトコルが必要な場合は、カスタム・アダプタをプラグインし、XSL またはカスタム・コードを使用して変換を実行できます。

8. 実際に顧客が変更を希望している日付に、NPAC は全国のすべての電話会社にブロードキャスト・メッセージを送信します。この時点で、すべての電話会社は、システム内でプロセス中のネットワーク要素（ネットワーク・データベース）を更新する必要があります。

データ・フォーマットとしての XML とアドバンスト・キューイングの使用

XML は、すべてのメッセージ機能に使用されるデータ・フォーマットです。アドバンスト・キューイング（AQ）は、図 9-2 のようにプロセス（およびシステム）内の各ポイントで使用されます。

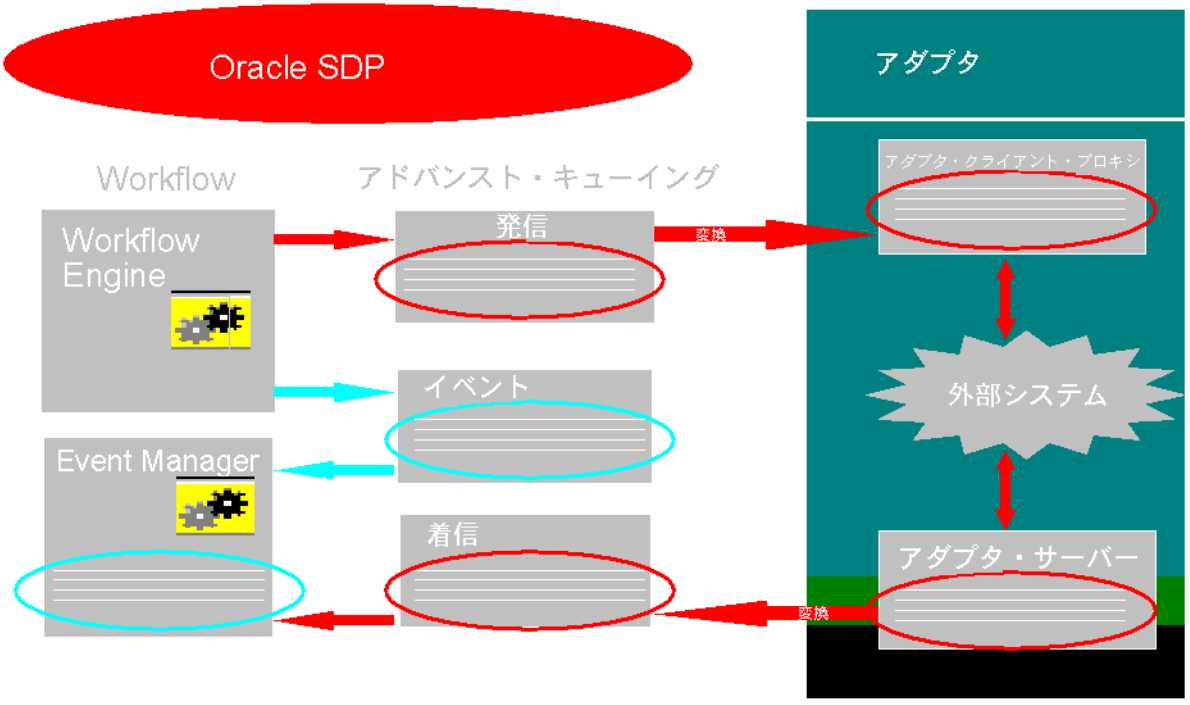
Message Builder モジュールでは、XML メッセージが作成され、エンキューされます。通信プロトコル・アダプタ（アダプタ）は、メッセージのデキューを開始して外部システムに送信します。

AQ は、実際には単にメッセージをキューに格納するために使用され、先入れ先出し（FIFO）方式のキューイング・システムとして機能します。メッセージの送信に使用されるプロトコルはシステムごとに異なり、フラット・ファイルや CORBA のようにエンド・ユーザーが指定します。

要約すると、次のようになります。

- 市内または長距離プロバイダの変更申込みを受信すると、要求は XML または変換済みメッセージとして NPAC に送信されます。
- 変更の承認および認可時に、NPAC は全国のすべての電話会社に XML メッセージを送信します。メッセージを受信した会社は、自社ネットワークへの措置を施します。

図 9-2 アドバンスト・キューイングを使用したメッセージ機能



メッセージ機能に XML が使用される理由

XML が使用されるのは、変更を加えたり、必要な他のフォーマットに変換できる柔軟なフォーマットであるためです。

たとえば、ある国では、メッセージの配布やネットワーク要素（データベース）の設置（更新）にフラット・ファイル・メッセージ・フォーマットを必要としている場合があります。XSL やカスタム・コードを使用すると、生成される XML を必要なフラット・ファイル形式に簡単に変換できます。

この Number Portability アプリケーションは、この方法でベルギーに順調に配置され、この方法で使用されています。ベルギーでは、フラット・ファイル・メッセージ・フォーマットが必要です。

迅速な構成を可能にする Number Portability

Number Portability 製品により、コンサルタントは次の作業を実行できます。

- アプリケーション内で XML メッセージの DTD を構成し、製品をすばやく実装できます。
- XML メッセージの様々なノードを 1 つの Oracle データ・ソース、SQL 問合せまたはストアド・プロシージャに割り当てることができます。
- SQL 問合せをネストできます。

たとえば、次の手順を実行すると、XML メッセージ内の dept と各 dept のすべての emp のリストを取得できます。

1. Number Portability アプリケーションで 2 つの問合せを記述します。
2. 提供された GUI でメッセージを構成します。コーディングは不要です。

Number Portability のアドバンスト・キューイングでは、データベースと外部システムのアダプタ間で通信の標準形式として XML メッセージが使用されます。

外部アダプタの概要

外部アダプタは、次の作業を実行する Java プログラムです。

- 実行するコマンドをデータベース・パイプでリスニングします。
- 処理するメッセージ（複数スレッド内）をアドバンスト・キューイング（AQ）でリスニングします。

コマンドは XML 形式でメソッド `performControlMessageProcessing` に送信されます。これにより、動的な数のパラメータをアダプタに渡すことができます。

たとえば、パフォーマンスのためにデフォルトで 3 つのスレッドを使用するアダプタを起動するには、次の STARTUP コマンドを使用します。

```
<COMMAND>
   <MESSAGE_CODE>STARTUP</MESSAGE_CODE>
   <INITIAL_THREADS>5</INITIAL_THREADS>
</COMMAND>
```

これにより、アダプタのカスタマイズが必要な場合に、より厳密な制御と高度な柔軟性が得られます。また、独自のコマンドを定義することもできます。XML メッセージの解析時に適用される制限はありません。

関連項目：

- PL/SQL ユーザーズ・ガイドおよびリファレンス
- ユーザー・インタフェースと *i*Message Studio の詳細は、『Oracle Number Portability 11*i* User's Guide』を参照してください。
- 『Implementing Oracle SDP Number Portability』

この章で使用される用語

この章では、次の用語を使用しています。

- GSM － Global System for Mobile Communication。これは、国際的に容認されたデジタル携帯電話の標準で、ローミングのために 900M 帯域幅の周波数を必要とします。
- NPAC － Number Portability Administration Center。
- NRC － Number Registration Center。NPAC の別名です。
- SDP － Oracle Service Delivery Platform（SDP）。

Wireless Number Portability（WNP）

Wireless Number Portability（WNP）ソリューションのプロバイダ間通信プロセスは、次のプロバイダ間での通信を考慮する必要があります。

- ワイヤレス・プロバイダとワイヤレス・プロバイダ
- ワイヤレス・プロバイダと有線プロバイダ

ワイヤレス・プロバイダ間の通信は、次のプロバイダ・タイプのポートに従ってさらに分類できます。

- GSM プロバイダ間のポート
- 携帯電話プロバイダ間のポート
- GSM プロバイダと携帯電話プロバイダ間のポート

Mobile Directory Number（MDN）および Mobile Subscriber Identity（MSID）

ワイヤレス・システムでは、サブスクライバを Mobile Directory Number（MDN）および Mobile Subscriber Identity（MSID）で識別できます。MDN は移動可能でダイアル可能です。MSID は移動もダイアルもできず、ローミング・ユーザーのホーム・サービス・プロバイダの識別に使用されます。MSID は再利用可能でもあります。

GSM（北米）では、MDN は 10 桁の電話番号で、MSISDN（Mobile Subscriber ISDN）と呼ばれます。GSM プロバイダの MSID は 1 ～ 15 桁の番号で、IMSI（International Mobile Subscriber Identity）と呼ばれます。

携帯電話の場合、MDN と MSID（Mobile Identification Number: MIN）はどちらも 10 桁のディレクトリ番号です（従来は、両方の番号が同じでした）。

WNP ソリューションでは、この各タイプのポート要求をすべてサポートできる必要があります。

> **関連項目：**『Porting Process - Reference CTIA Numbering Advisory Working Group Report On Wireless Intercarrier Communications Version 2.0』

NPAC

番号の移植性が必須となっている多くの国では、Number Porting Administration Center（NPAC）などの組織がオペレータ間でのサービスの切替えを制御しています。また、移植された番号に関する情報を記録し、ポート情報の一元的な配布元ポイントを提供しています。

集中化された公正な団体にポート・プロセスを監督させることで、次のようなメリットがあります。

- 移植された番号に関する記録が全国で 1 つになります。データベースは、新規ネットワーク・オペレータ用に移植された番号に関するすべての情報にアクセスできます。また、データベースでは、あるオペレータの内部データベースが破損して同期しなくなった場合に、同期機能も実行できます（障害時リカバリ）。
- 移植先や移植元が、移植した番号を他のすべてのオペレータに通知することを容易にします（これを提供しているのは、最後の 2 つのモデルのみです）。

前述した以外に、集中化されたデータベースは次のような民間サービス組織でも参照できます。

- ディレクトリ・サービス・プロバイダ
- 緊急サービス

サービス・ゲートウェイ

サブスクライバへのサービス提供に 2 つの通信オペレータ（TO）またはサービス・プロバイダ（SP）が関係する場合は、両者が別々の OSS を相互接続できる必要があります。そのためには、次の多数のポリシーが必要になります。

- 一方のビジネス・プロセスが他方のビジネス・プロセスに影響する範囲。
- 一方が他方の保持している情報を自律的に変更できる範囲。

この一連の要件における重要な点は、データの整合性を制御するために組織の共同運用が必要になることです。

サービス・ゲートウェイは、次の機能を有効化して、企業間プロセスにフロースルー機能を提供します。

- 透過的サービス。複数のプロバイダの共同運用によりサービスが提供される場合は、エンド・ユーザーから透過的である必要があります。
- 自律性。サービス・プロバイダがサービスの提供については共同運用していても、その自律性は維持する必要があります。
- 簡潔さ。新規サービスの構成、サービスの追跡およびビジネス・プロセスの変更が、複雑にならないようにする必要があります。
- テクノロジの選択。共同運用エンティティは、独自の条件に基づいてテクノロジを選択できる必要があります。そのため、サービス・ゲートウェイはテクノロジの制約を伴わないようにする必要があります。
- 規格への準拠。ゲートウェイは、法的組織、業界フォーラムや標準化団体、市場の条件のいずれによって設定されたかに関係なく、一般的な慣行と規格に準拠する必要があります。

非対称デジタル加入者回線（ADSL）

ADSL は非対称デジタル加入者回線（Asymmetric Digital Subscriber Line）です。

ADSL のビジネス手順

代表的な ADSL のビジネス・プロセスは、次のとおりです。

1. 事前の適格性チェック
2. 事前の実現可能性チェック
3. 注文の作成
4. ループ適格性チェックの実施
5. DSL 回線の設置
6. DSL ループの検査
7. DSL CPE の配信
8. DSL CPE のスケジューリングとインストール
9. DSL および ATM 接続を使用した DSL CPE のテスト
10. DSL 回線を使用した IP サービスのテスト
11. 注文完了メッセージの発行

ADSL のメリット

ADSL のメリットは、次のとおりです。

- 市場提供までの期間。DSL ソリューションは事前定義 / 事前構成済みのビジネス・フロー、データ、構成およびインタフェースを提供するため、顧客は DSL サービスを迅速に実装または変更できます。
- DSL サービスは通信領域で大きな割合を占めているため、顧客にとってのメリットが大きくなります。

DSL の概要

DSL テクノロジは、顧客を将来へと導く新しいテクノロジ・プラットフォームです。DSL は、既存の銅線による電話回線を使用して、顧客の施設に直接接続されるブロードバンド・デジタル接続です。DSL テクノロジを使用すると、企業はインターネット使用を拡張してユーザーの生産性を高めるように設計された大規模なサービス・パッケージを活用できます。DSL ネットワークのコンポーネントは、コロラドにある複数サービスの DSLAM と、顧客の施設にある DSL リモート・トランシーバ・ユニット（DSL モデム）で構成されます。DSLAM は、DSL 回線を T1/E1、T3/E3 または ATM 出力に集中させることで、パケット、セル、回路ベースのアプリケーションにバックホール・サービスを提供します。DSL モデムは、顧客からネットワークへのデジタル・リンクを維持します。このモデムは、通常の電話回線を従来のダイアル・アップ・モデムよりはるかに高速（最大 7+Mbps）で使用できます。このモデムを既存の Local Area Network（LAN）に接続するか、特殊ケーブルを使用して PC に接続できます。

DSL テクノロジは次のように様々で、総称的に xDSL と呼ばれます。

- ADSL − 非対称デジタル加入者回線 − このテクノロジの下り速度はレポートされていますが、上り速度は下り速度の数分の 1 です。主として住宅用で、多くのプロバイダは帯域幅レベルを保証していません。ADSL は、同じ電話回線で 3 つの異なる周波数チャネルを提供します。この 3 つのチャネルのうち、1 つは通話用、1 つは Network Service Provider（NSP）からエンド・ユーザーに転送される下りのデータ用、1 つはエンド・ユーザーから NSP への上りデータ用です。
- SDSL − 対称デジタル加入者回線（Symmetric Digital Subscriber Line）− このテクノロジは、上りと下りの双方向に同じ帯域幅を提供します。つまり、情報をアップロードするかダウンロードするかに関係なく、同じ高品質のパフォーマンスが得られます。SDSL の送信速度は T1/E1 の範囲内で、中央のオフィスから最大 12,000 ～ 18,000 フィートの範囲内では、単一のペア銅線経由で最大 1.5Mbps です。このオプションは、インターネット経由でデータのダウンロードにもアップロードにも同じ速度が必要な中小企業に最適です。
- RADSL − Rate Adaptive Digital Subscriber Line − このテクノロジでは、回線の状態に基づいてアクセス速度が自動的に調整されます。
- IDSL − ISDN デジタル加入者回線（ISDN Digital Subscriber Line）− このテクノロジは SDSL と同様に対称的ですが、SDSL より低速で長距離に対応します。

- HDSL － High-Data-Rate Digital Subscriber Line － このテクノロジは対称的ですが、主として T-1 回線経由の PBX 用に配置されます。
- VDSL － Very-High-Rate Digital Subscriber Line － これは高速テクノロジですが、範囲がきわめて限定的です。

DSL テクノロジを使用すると、サービス・プロバイダは一連の end-to-end ビジネス管理ソリューションを提供し、顧客にコア・インターネット・サービス、Web ホスティング、拡張電子メール、管理された装置、仮想プライベート・ネットワークおよびサーバー・ベース・アプリケーションなど、最先端の機能を提供できます。これにより、顧客はそのリソースを従来より適切かつ高速に最大限まで操作できます。

提供できる DSL ベース・サービスの例を次に示します。

- Web ホスティング － 顧客が重要なビジネスに専念できるように、顧客 Web サイトの日常的な技術的領域を管理します。
- 拡張電子メール － いつでもどこでも電子メールにアクセスできます。共通の電子メール・クライアントを通じて送受信できるようにします。インターネット接続機能を持つコンピュータであれば、どこからでも Web 経由で電子メールにアクセスできます。
- インターネット・ベースの画像への高速アクセス。
- Local Area Network（LAN）へのリモート・アクセスや社内ネットワークへのアクセス。
- DNS 管理 － 顧客の Web 識別情報の適切な処理と、顧客システムの管理が保証されます。
- ホーム・ショッピング。

Voice Over IP （Clarent）

Clarent VoIP システムは、次のコンポーネントで構成されています。

- **Command Centers（CC）**。Call Managers（CM）Gateways（GW）とインタフェースします。
- **Customer/Provisioning Database**。CPE とインタフェースする CM です。
- ゲートウェイ。PSTN、IP ネットワークおよび他のゲートウェイとインタフェースします。

Customer/Provisioning Database には、Customer、Routing（Trunks）、Rating など、CC で顧客をアクティブ化するための表が含まれています。

たとえば、サービス・プロバイダ X が Clarent Network を使用して顧客に VoIP サービスを提供するとします。顧客から VoIP サービスを要求された場合、X は顧客のアクティブ化に必要なすべての情報を Clarent の Customer/Provisioning データベースに入力する必要があります。CC は、この情報を使用して、その顧客用の VoIP サービスをアクティブ化します。

現在、Customer、Routing、Rating の各データは、Clarent Assist 経由で 1 つずつ追加されます。

Clarent Assist は、Clarent Customer/Provisioning データベースの行の追加、変更および削除に使用できる HTML 画面です。

帯域幅交換（プロトタイプ）

Communications Services Exchange Prototype（以降では Exchange と呼びます）は、SDP エンジンを使用した Telco ソリューションとして意図されています。Exchange は、サービス・プロバイダ間や大企業間における大規模通信サービスを売買し、そのサービスを容易に実現するために使用されるインターネット・コミュニティです。Exchange により、買い手と売り手の取引が容易になります。買い手と売り手は、どちらも相互にサービスを売買する大企業であってもかまいませんし、自分の所属とは別のサービス・プロバイダからサービスを購入するのでもかまいません。いずれにしても企業は、他のサービス・プロバイダとサービスを交換した後も、現行のサービス・プロバイダに所属します。買い手と売り手が取引に同意して締結すると、オプションが使用可能になり、Exchange の Service Fulfillment 機能を通じて交換したサービスを設置し、アクティブ化するかどうかを選択できます。

Exchange の機能とプラットフォームは、Oracle E-Business Internet Procurement アプリケーション（Oracle Exchange）に準拠しています。その他に必要な機能は、SDP Telco Solution により開発されます。Exchange の主なコンポーネントと機能は、次のとおりです。

- Registration and Profiling Management － Oracle Exchange に基づく機能です。主な機能は、市場管理者による参加者の検討と承認、ビジネス機能およびデータ・アクセス・レベルのセキュリティ・ルール、ユーザーによる各自の個人プロファイルおよびユーザー設定項目の管理などです。
- Products & Services Catalog － Oracle Exchange および CRM Product Catalog に基づく機能です。主な機能は、セルフサービス・ツール、XML オープン・インタフェース、カタログの作成および保守などです。
- Service Level Agreement（SLA）－ Oracle Exchange に基づく機能です。主な機能は、Service Bulletins での SLA を使用した RFI および RFQ（ビッド）の作成と配布、SLA 用標準テンプレートのメンテナンス、ビッドの分析と複数のビッドの往復のうち 1 つへの対応、Service Bulletins のダウンロード / アップロード、イベント駆動通知などです。
- Security － Oracle Exchange に基づく機能です。主な機能は、エンド・ユーザー / 企業データのセキュリティ、接続性です。
- Buyer and Seller Self-Service － Oracle Exchange に基づく機能です。主な機能は、HTML ベースの照会およびトランザクション機能、新規注文の検討および受入れ、各買い手と売り手またはメンバーに対する市場参加の許可です。
- Service Configurator － Exchange の新機能です。主な機能は、売り手にサービス構成データを提供するための自動移入または入力機能（ファイルなど）、Service Configurator のメンテナンス（表示、追加、変更および削除）です。

- Service fulfillment ― Exchange の新機能です。主な機能は、Oracle Exchange の注文から SDP で受け入れられる設置可能なサービス注文への変換、SDP fulfillment エンジンを使用し、サービス構成に基づいたネットワーク要素の設置とアクティブ化または削除です。

SDP 内での Number Portability およびメッセージ機能アーキテクチャ

Oracle Service Delivery Platform（SDP）フレームワークの Number Portability およびメッセージ機能アーキテクチャは、次のコンポーネントで構成されています。主要なコンポーネントは Event Manager です。

- 通信プロトコル・アダプタ
- Order Processing Engine
- Workflow Engine
- Fulfillment Engine
- Event Manager
- SDP Repository

図 9-3 および図 9-4 に、Number Portability 全体のワークフローを示します。

図 9-3 Number Portability のワークフロー（1/2）

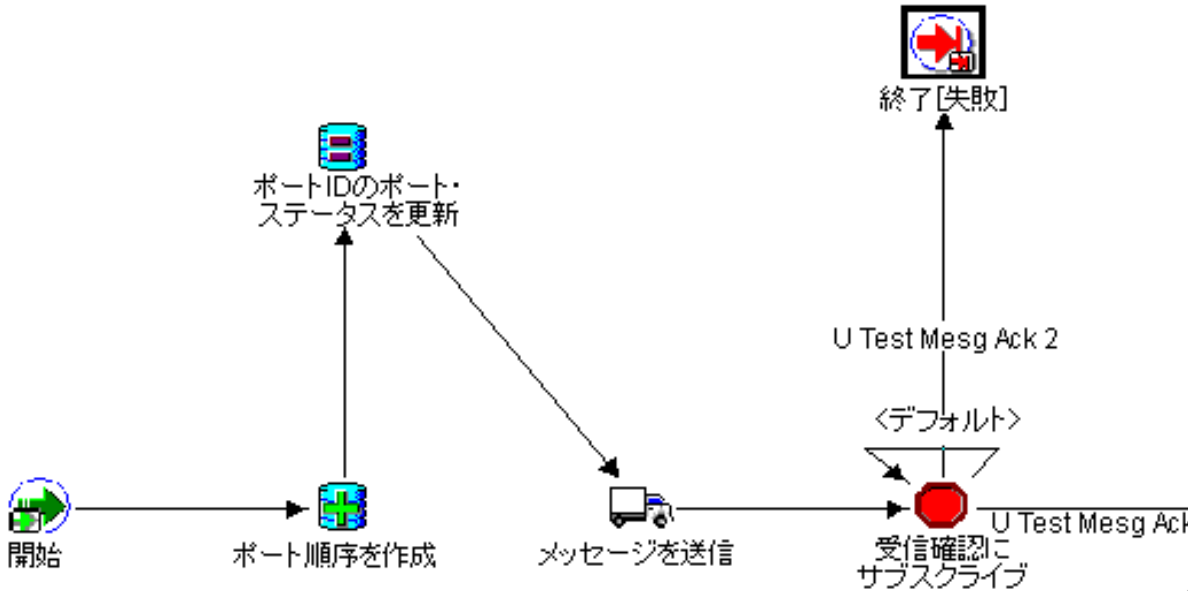
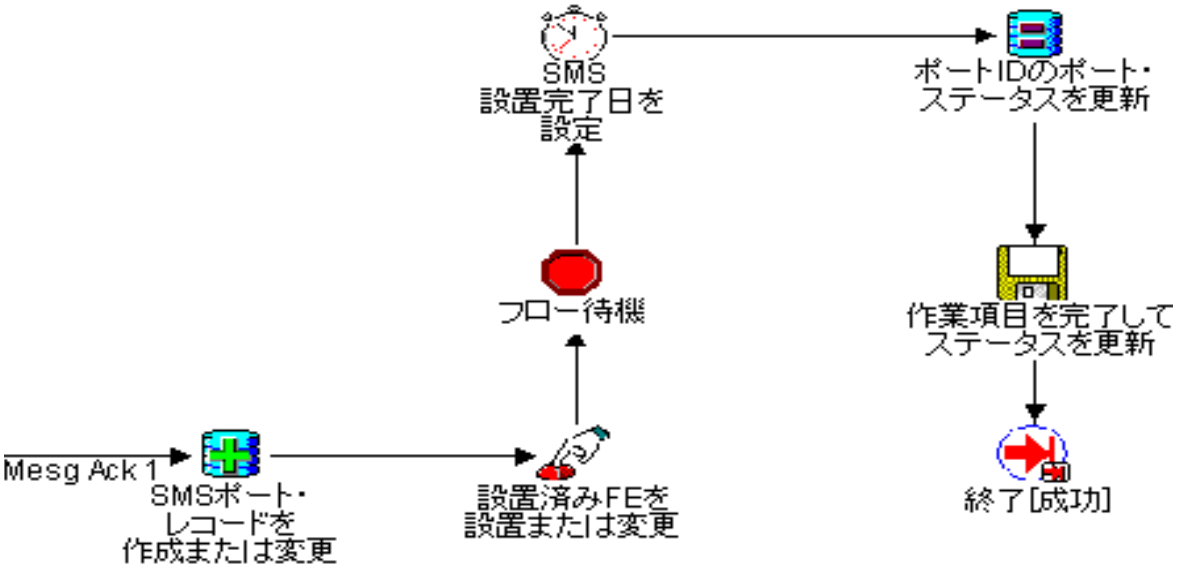


図 9-4 Number Portability のワークフロー（2/2）



通信プロトコル・アダプタ

通信プロトコル・アダプタは、SDP と外部システム間をインタフェースします。次のメッセージ・フローを処理します。

- 着信した注文は、適切な通信プロトコル・アダプタにより取り込まれ、SDP に渡されます。
- 発信メッセージは、SDP から外部システムに渡されます。

サポートされるアダプタは、次のとおりです。

- ファイル /FTP。このアダプタでは、バッチ・モード処理がサポートされます。
- HTTP。
- スクリプト。
- Telnet セッションなどの対話型アダプタ。

Order Processing Engine

受注管理システムからの注文は、SDP 処理アイテム、つまり論理明細項目に変換されます。注文は、メッセージの処理から内部的にも作成されます。

このようにして作成された明細項目は、Dependency Manager または Order Analyzer で分析されます。実行のために、正規化処理アイテムの最終セットが作成されます。

Workflow Engine

このモジュールでは、アプリケーションの機能を満たすために実行される実際のアクション・フロー（実行アクション）を指定します。このモジュールでは、実行アクションを再利用して、NP Service Provider Mediation などの新機能や NRC 自体をカスタマイズできます。

Workflow Engine では、処理アイテムごとに実行する実行アクションが決定され、対話が必要なネットワーク要素が決定されます。また、実行要素タイプ、そのソフトウェア・バージョンおよびアダプタ・タイプに基づいて、適切な実行プロシージャが選択され、実行されます。

実行プロシージャでは、Internet Message Studio で生成されたコードを使用してメッセージが送信され、処理されます。実行アクションの（Event Manager による）実行結果に関するイベント通知を受信すると、このエンジンは処理アイテムの完了に進み、特定の注文のキューにある次の処理アイテムを選択します。このコンポーネントでは、Oracle Workflow Engine が使用されます。

Fulfillment Engine

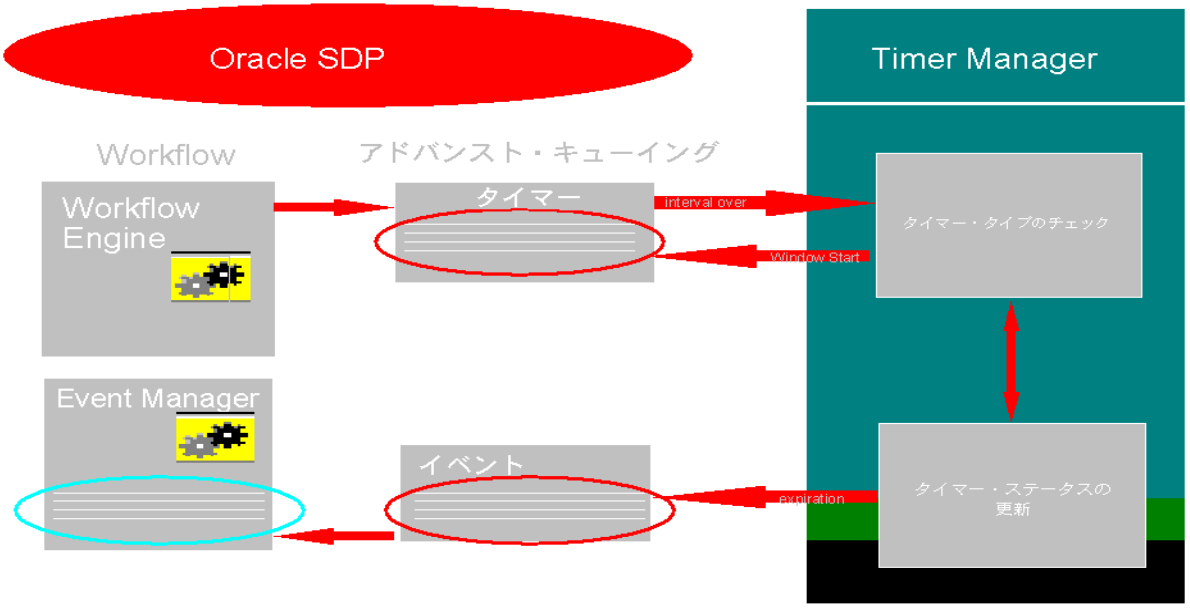
実行アクションと、それに適用する必要のあるネットワーク要素は、SDP の設置エンジンにより使用され、実行する実行プログラムが決定されます。実際には、現在データベースにある PL/SQL エンジンを使用して、ユーザー定義プロシージャが実行されます。

Fulfilment Engine の主要機能は、次のとおりです。

- ネットワーク要素の設置の許可
- Oracle Provisioning との緊密な統合
- 設置プロトコル・アダプタ
  - 対話型（Telnet など）
  - スクリプト（PERL などの実行可能ファイル）
  - HTTP
- NP SMS 機能の標準アクティビティ
  - 処理済み番号範囲設定のサポート

図 9-5 を参照してください。

図 9-5 Number Portability（NP）での Oracle Provisioning の統合



Event Manager

Event Manager は、様々なサブスクライバの関心を異なるイベント・タイプに登録する、汎用的なパブリッシュ・サブスクライブ・モジュールです。サブスクライバには、SDP Translator（この場合、イベントは新規注文として伝播）、Workflow Engine（この場合、イベントにより外部イベントを待機中のワークフローが再開）または API が考えられます。Event Manager により、非同期アプリケーション・メッセージ機能が構築されます。多様な API セットが提供されており、開発者はアプリケーションに基づく非同期メッセージの作成に使用できます。

SDP Repository

コア SDP リポジトリを使用すると、ユーザーは注文を作成し、ネットワーク要素を構成できます。たとえば、処理アイテム、実行アクション、実行プログラムおよびネットワーク要素の定義などです。NP データベースには、サブスクリプションのバージョン、サービス・プロバイダ、ルーティング番号など、NP 固有のデータを格納するためのエンティティが含まれています。

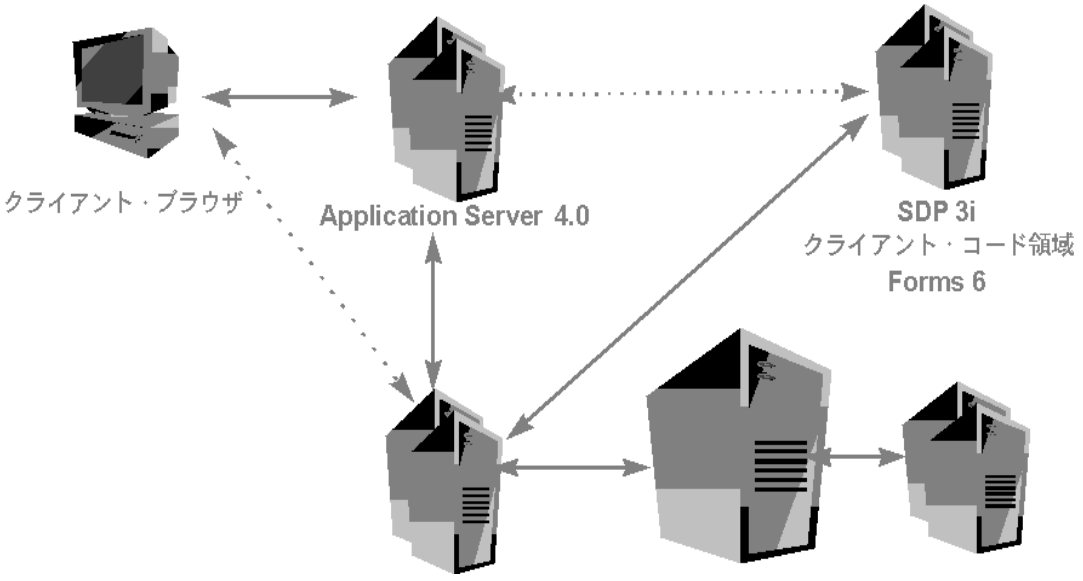
Phone Number Portability アプリケーション構築の要件

Number Portability アプリケーション構築の要件は、次のとおりです。

- Oracle Applications 11*i*
- Oracle SDP Number Portability 11*i* リリース 2
- Oracle8*i* リリース 8.1.5 Enterprise Edition 以上、Objects Option 付き
- Oracle8 リリース 8.0.5 / 8.1.5 以上の JDBC ドライバ
- Oracle Developer リリース 6.0.5 Runtime 以上
- Oracle Workflow 2.5.0 Cartridge（スタンドアロン・バージョン）
- Oracle9*i* Application Server 1.0.1 以上
- JDK / JRE 1.1.7 以上
- JDK 1.1 以上の対応ブラウザ
- （XML 用に Netscape 4.5+ / MS IE 5.0 を推奨）

図 9-6 に、Number Portability 環境を示します。この図の下部にあるサーバーは、左から右へと順に Forms Server、Oracle9*i* Application Server および SDP Adapter Server です。これは単なる概略図で、サーバーが 1 つのマシンに常駐することもあります。

図 9-6 Number Portability 環境



ネットワーク要素の設置

電話システムには、個々のエンド・ユーザーの電話番号の他にも、設置（更新）できるネットワーク要素があります。このようなネットワーク要素には、Service Control Points（SCP）、ルーター、LDAP サーバーなどがあります。

SDP Provisioning アプリケーションの使用例を次に示します。

1. 市内通話サービス・プロバイダが設置の切替えを要求します。
2. 仲介層がスイッチと対話します。
3. Service Delivery Platform（SDP）が仲介層にメッセージを送信します。SDP は XML 対応の従来型システムでもかまいません。
4. SDP は、設置（更新）が完了すると、仲介層から応答を受信します。

このメッセージ機能は、メッセージ・ペイロードおよびアドバンスト・キューイング（AQ）として XML も使用します。ここでは、AQ は主として XML キューの記憶メディアとして使用されます。将来のリリースでは、AQ 上の JMS インタフェースを使用して標準インタフェースを提供できます。

Internet Message Studio（*i*Message）を使用したアプリケーションのメッセージ・セットの作成

Internet Message Studio（*i*Message）ユーティリティは、Number Portability アプリケーションまたは企業のメッセージ・セットを定義するために使用します。このユーティリティを使用すると、メッセージ・ベース・アプリケーションを開発し、アプリケーション・メッセージの構成、発行、検証および処理に必要なコードをすべて生成できます。

また、アプリケーション間でメッセージを共有でき、同じメッセージを社内の各種アプリケーションで再定義する必要がなくなります。アプリケーションでは実行時に、すべてのメッセージ機能のニーズに合せて、生成されたプロシージャを実行できます。また、ビジネス固有のロジックを組み込むために必要なフックまたはカスタマイズ・ポイントも提供します。メッセージは標準 XML を使用して生成されます。

Message Builder（*i*Message）の機能

SDP Message Builder（*i*Message）の機能は、次のとおりです。

- *i*Message Studio を使用した GUI ベースのメッセージ定義
  - 将来のリリースにおける Oracle XML Parser/Utilities のサポート
  - サイト構成時に事前定義されるメッセージとイベント
- メッセージ、イベントおよびタイマーのサポート
- インターネット標準の XML メッセージ・フォーマットの使用
  - すべてのメッセージに対する XML Dekagrams の処理
- 自動的に最適化されるコード生成
  - 処理および検証ロジックの最小限のコーディング
  - SQL、PL/SQL ファンクションまたは SDP の注文 / 処理アイテム /FA パラメータの宣言によるメッセージ・データ・ソースの指定
  - メッセージ・タイプに基づく Send()、Publish()、Process()、Validate()、Fire() の生成
- Oracle XML Gateway および Oracle Integration Server（OIS）との計画済みの統合

コードの生成

*i*Message は、定義するメッセージごとに、パッケージの一部としてメッセージ名と次のプロシージャを使用してパッケージを作成します。

- CREATE_MSG()
- SEND()
- PUBLISH()
- VALIDATE()
- PROCESS()
- DEFAULT_PROCESS()

メッセージ・セットの定義

図 9-7「iMessage の「Data Source」ウィンドウを使用した XML メッセージ要素のデータ・ソースの定義（Oracle Developer Forms 内）」に、*i*Message を使用して XML メッセージを定義する方法を示します。このスクリーン・ショットは、XML メッセージ要素、構造および関連するソース SQL 問合せも示しています。

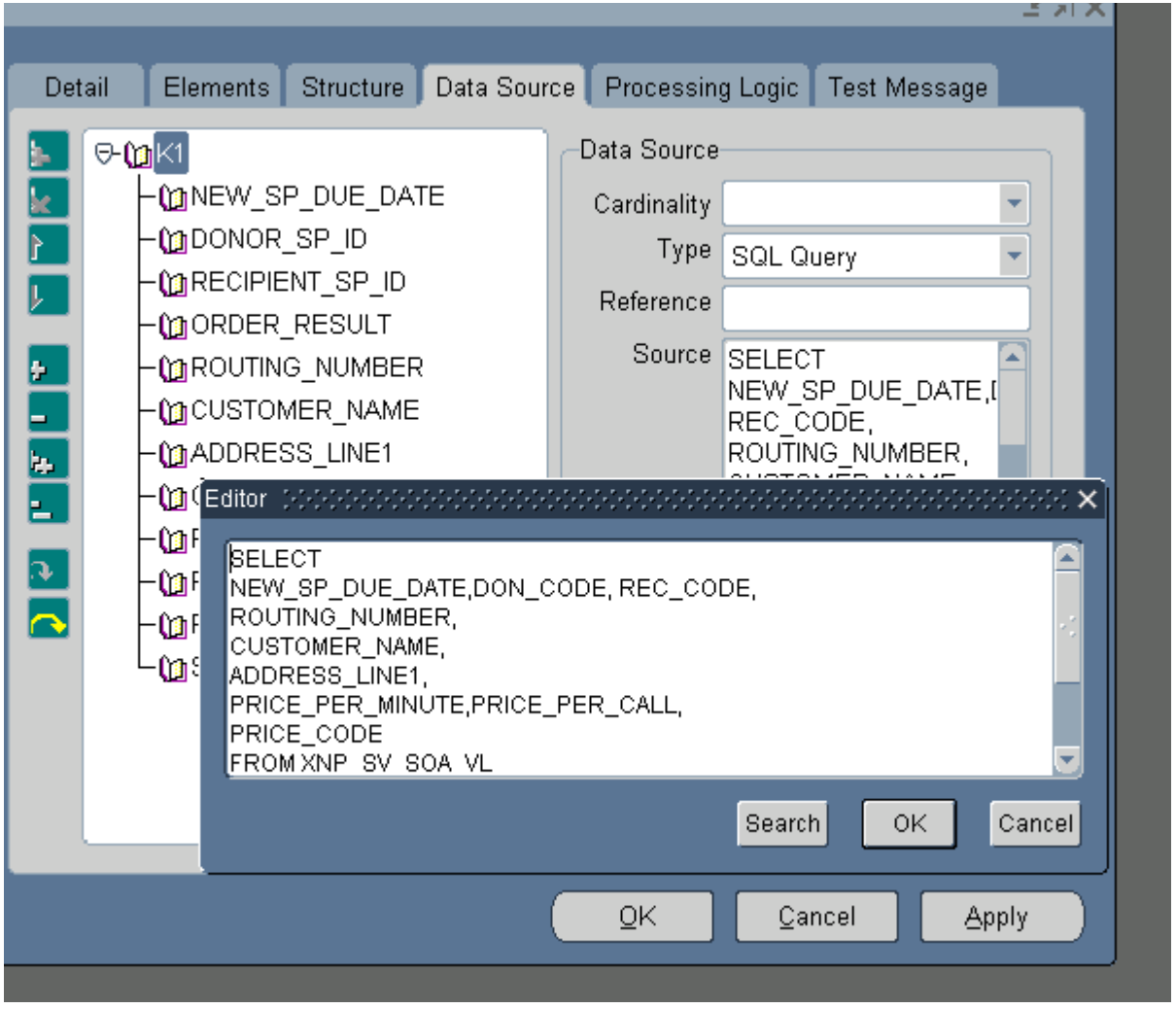
*i*Message を使用して XML メッセージ・セットを定義するには、多数のステップを実行する必要があります。これには、次のステップが含まれます。

- **メッセージの定義**。すべての要素（属性）とその構造の関係を指定して、メッセージを定義できます。必須かオプションか、最大データ長およびデフォルト値など、他の制約も指定できます。
- **メッセージ詳細の追加**。「Type」フィールド。Internet Message Studio を使用すると、アプリケーション・イベントも定義できます。メッセージとイベントの重要な違いは、メッセージはアプリケーション・システム間の通信に使用され、イベントはビジネス・オブジェクトのブロードキャストまたはマルチキャスト状態の変更に使用できることです。また、Internet Message Studio ではタイマー・メッセージも定義できます。

  Internet Message Studio を使用して定義したイベントは、外部および内部のアプリケーション・システムに対して公開されます。内部アプリケーションでは、「event Publisher」画面または前述の API を介して PL/SQL コールバック・プロシージャを登録でき、そのプロシージャはイベントの公開時に実行されます。定義では、外部アプリケーションはコールバック・プロシージャを登録しませんが、実行中のアダプタに公開されたイベントをリモート・システムへと中継させます。外部アプリケーションは、デフォルトのサブスクライバ画面を使用してイベントを登録できます。内部アプリケーションのよい例は、単一の Oracle インスタンスで実行される Oracle の SDP および Installed Base です。
- **記述**。記述は、メッセージが使用されるコンテキストを提供します。
- **表示名**。表示名は、メッセージの記述名です。

- **メッセージ要素の追加**。
- **メッセージ構造の構築**。メッセージの構造により、メッセージ要素の階層関係が定義されます。この階層に組み込めるのは、事前定義済みの要素のみです。メッセージ構造の構築方法の詳細は、ユーザーズ・ガイドを参照してください。メッセージ構造は、最上位要素をルートとする逆順のツリーとして表示できます。
- **ルート要素**。デフォルトでは、Internet Message Studio ではルート要素として 'MESSAGE' が組み込まれ、定義するメッセージはルート要素の子となります。ルート要素は表示されず、Internet Message Studio により暗黙的に定義されるため注意してください。ルート要素は削除できません。構造に含まれていない要素は、メッセージに表示されません。
- **データ・ソースの定義**。メッセージ定義の次のステップは、メッセージ要素のデータ・ソースを定義することです。メッセージ要素は、値を PL/SQL ファンクション・コールまたは SQL 問合せから取得できます。また、データは SDP 注文パラメータ、SDP 処理アイテム・パラメータまたは実行アクション・パラメータとしても取得できます。図 9-7 を参照してください。

図 9-7 *i*Message の「Data Source」ウィンドウを使用した XML メッセージ要素のデータ・ソースの定義（Oracle Developer Forms 内）



サポートされる SDP データ型は、次のとおりです。

- **PL/SQL ファンクション**。実行時に PL/SQL ファンクションを実行し、メッセージ要素の値を取得できます。指定した PL/SQL ファンクション・コールでは、定義済みのメッセージ要素のいずれかを参照して引数を渡すことができます。また、PL/SQL ファンクションでは、上位レベルのメッセージ要素でデータ・ソースとして定義されている列のうち、選択した列を参照できます。ファンクションは、ユーザー・インタフェースの「Source」フィールドで指定し、戻り値の型はメッセージ要素に指定されている型と同じにする必要があります。
- **SQL 問合せ**。SQL 問合せを使用すると、メッセージ要素のデータを導出できます。 他のメッセージ要素がツリー階層の下位レベルで定義されている場合は、そのメッセージ要素の参照として SQL 問合せの列を使用できます。

その他のデータ型は、次のとおりです。

- SDP 注文パラメータ
- SADP 処理アイテム・パラメータ
- SDP 実行アクション・パラメータ

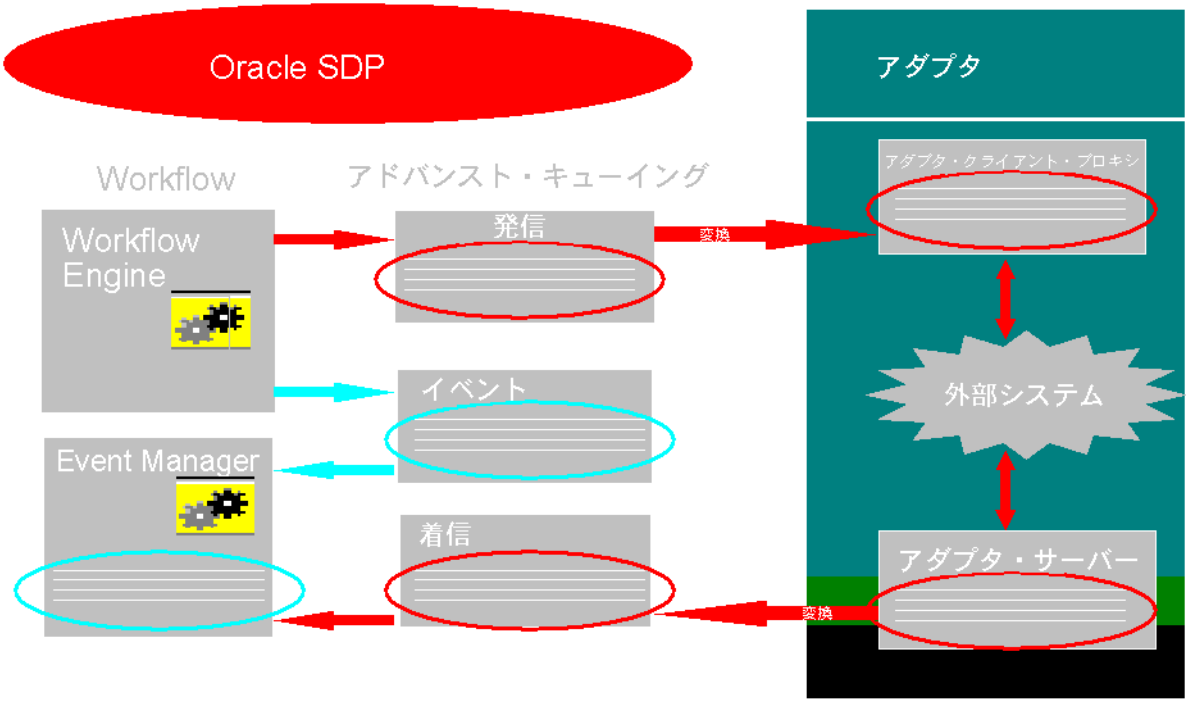
Timer Manager の使用

Timer Manager の主要機能は、次のとおりです。

- XML Message Builder を使用したタイマーの定義
- 複数タイプのタイマーのサポート
- タイマーの処理（明示的な起動と削除）
- タイマーの時間枠設定（遅延後の暗黙的な起動）
- メッセージ・タイマー（メッセージ関連のタイマーの起動）
- 外部 SLA システムとの簡単な統合
- API のフックによる遅延と間隔の決定
- タイマー API および標準ワークフロー・アクティビティ
- Fire()、Remove()、CheckTimerStatus()、Restart()、Recalculate()、StartRelatedTimers() など
- 危険管理に有効
- タイマーとしての危険イベントの定義
- タイマーが期限切れになった時点での通知の送信またはワークフローの開始

図 9-8 に、SDP のタイマーによるアドバンスト・キューイングの実装方法を示します。

図 9-8 アドバンスト・キューイングを使用したタイマー



A

XML の手引き

この付録の内容は、次のとおりです。

XML の概要

Extensible Markup Language（XML）は、Web 上でデータの識別と記述に使用されている標準的な手段です。広範囲に実装可能であり、配置も簡単です。

XML は、階層形式のデータを記述するための、判読可能でマシンで認識できる汎用構文であり、アプリケーション、データベース、E-Commerce、Java、Web の開発や検索などに応用できます。

カスタム・タグを使用すると、アプリケーション間や組織間でデータを定義、送信、検証および解析できます。

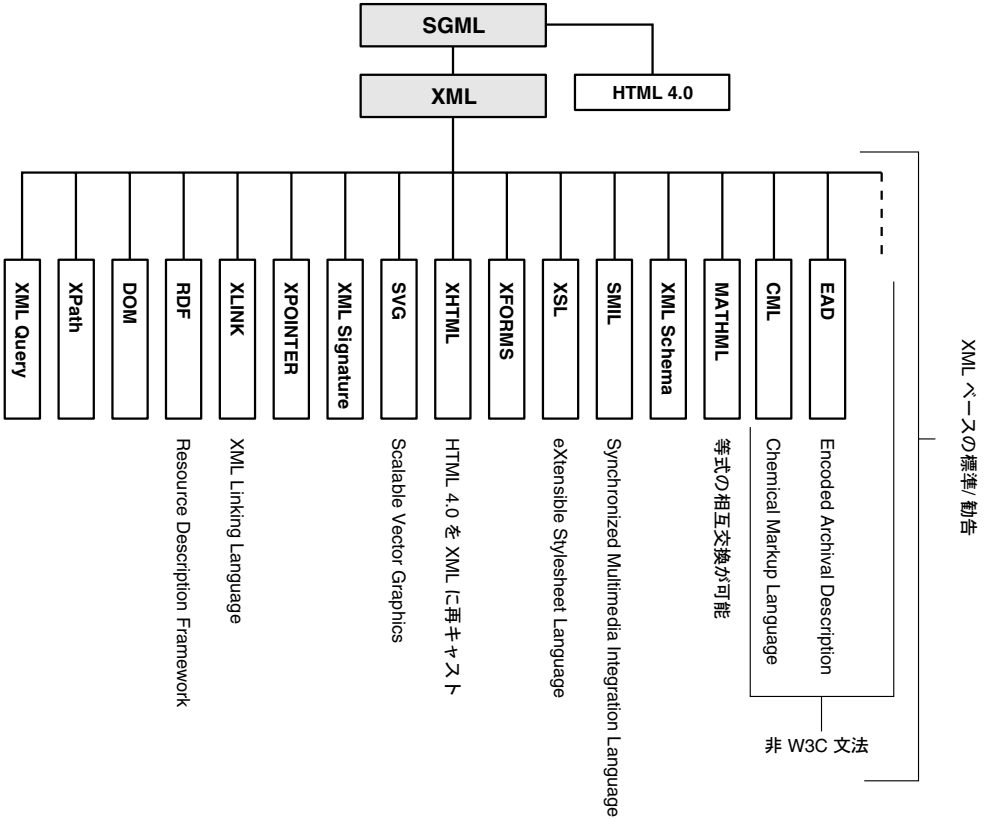
W3C による XML 勧告

World Wide Web Consortium（W3C）の XML 勧告は、絶えず成長を続けるインターロッキング仕様のセットです。

- **XML 1.0**。W3C による 1998 年 2 月の勧告です。これにより、多数の W3C ワーキング・グループ、Java プラットフォーム拡張エキスパート・グループおよび電子データ交換（Electronic Data Interchange: EDI）のような様々なデータ交換標準の XML 変換が登場しました。HTML の次のバージョンは、XHTML と呼ばれる XML アプリケーションになります。
- **XML 名前空間**。もう 1 つの W3C 勧告。複数の名前空間の整形式の XML アプリケーションにおける、要素のあいまいさの排除を目的としています。
- **XML Query**。XML 文書の問合せ言語を指定する W3C の標準。
- **XML Schema。**単純および複合データ型を XML 文書に追加し、DTD の機能を XML Schema 定義の XML 文書に置き換えるための W3C の標準。
- **XSL**。XSL は、次の 2 つの W3C 勧告で構成されています。
  - XML 文書間で変換するための XSL Transformations
  - XML 文書の表示を指定する XSL Formatting Objects
- **XPath**。XSL-T、XLink および XML Query に使用される XML 文書を操作するためのデータ・モデルおよび文法を指定する W3C 勧告。
- **XPointer**。XPath ナビゲーションを使用して、XML 文書内の個別のエンティティまたはフラグメントの識別を指定する W3C 勧告。この W3C 勧告案の定義は、http://www.w3.org/TR/WD-xptr を参照してください。
- **DOM**。プログラム・アクセス用の API など、XML 文書のドキュメント・オブジェクト・モデル（Document Object Model: DOM）を指定する W3C 勧告。

図 A-1 に、XML ファミリのアプリケーションを示します。

図 A-1 XML ファミリのアプリケーション（XML ベースの標準を含む）



XML 機能

XML の機能は、次のとおりです。

- **構造化データおよび非構造化データのデータ交換**。XML により、データ交換用の汎用的な標準構文が使用可能になります。XML では、データ固有の構造を表す堅固なテキスト・ベースの方法が指定されるため、明確に作成し、解析できます。単純なタグ・ベースのアプローチにより、開発者は HTML の知識を活用できるのみでなく、高度に構造化されたデータベース・レコードから構造化されていないドキュメントまで、あらゆるデジタル資産を処理できる柔軟で拡張可能なメカニズムが得られます。
- **ドキュメント専用に設計された SGML とは異なり、あらゆるデータ用に設計された XML**。SGML マークアップ言語は、ドキュメント専用に設計されました。Web 中心の XML は、他の言語の記述にも使用できるツールキットに似ています。ドキュメント専用に設計されているのではなく、ツリー形式で記述できるデータであれば、どんなものでも XML でプログラムできます。
- **データ・オブジェクトのクラス ― 限定的な形式の SGML**。www.oasis-open.org の XML に関する記述は、「...XML は、XML 文書と呼ばれるデータ・オブジェクトのクラスが記述され、それを処理するコンピュータ・プログラムの動作も潜在的に記述されます。XML は、アプリケーション・プロファイル、つまり限定的な形式の SGML（Standard Generalized Markup Language）です。これに対して、XML 文書は SGML ドキュメントに準拠しています。」となっています。
- **XML の多数の用途**。W3C.org のプレス・リリースでは、XML の記述は「...XML は、主として業界固有のマークアップ、ベンダーに依存しないデータ交換、メディアに依存しない公開、1 対 1 のマーケティング、コラボレーションによるオーサリング環境でのワークフロー管理、インテリジェント・クライアントによる Web ドキュメントの処理などに対する、大規模な Web コンテンツ・プロバイダの要求を満たすことを意図しています。」となっています。
- **メタデータ**。XML は、特定のメタデータ・アプリケーションで検索にも使用できます。
- **各国対応**。「XML は、UTF-8 および UTF-16 エンコードによる Unicode キャラクタ・セットのサポートに必要な準拠プロセッサをすべて備えており、ヨーロッパ言語とアジア言語の両面で完全に各国対応になっています。その主な用途は電子パブリッシングおよびデータ交換です。」
- **解析済みまたは未解析の記憶域エンティティ**。W3C.org XML の仕様案によれば、「...XML 文書はエンティティと呼ばれる記憶単位で構成され、この記憶単位には解析済みまたは未解析のデータが含まれています。解析済みのデータは文字で構成され、その一部がドキュメントのキャラクタ・データとなり、その他はマークアップを構成します。マークアップにより、ドキュメントの記憶レイアウトと論理構造の記述がエンコードされます。」となっています。

- **XML プロセッサによる XML 文書の読込み**。「...XML では、記憶レイアウトと論理構造に制約を課すメカニズムが提供されています。XML プロセッサと呼ばれるソフトウェア・モジュールを使用して XML 文書が読み込まれ、その内容と構造へのアクセスが提供されます。XML プロセッサは、アプリケーションと呼ばれる他のモジュールのかわりに処理を行うとみなされます。」
- **オープンなインターネット標準**。XML は、IBM 社、Sun 社、Microsoft 社、Netscape 社、SAP 社、CISCO 社など、業界の他のベンダーからも、プラットフォームやアプリケーションに依存しない情報交換形式として広く支持されています。

このマニュアルは、XML 構文の詳細を解説することではなく、一部の重要な XML トピックの概要を説明することを意図しています。XML 構文の詳細は、「その他の XML リソース」に示す多数の優れた資料を参照してください。

XML と HTML の違い

HTML と同様に、XML は Web での配信用に最適化された SGML（Structured Generalized Markup Language）のサブセットです。

HTML ではブラウザで表示できるように Web ページの要素が <bold>Oracle</bold> などのタグで囲まれますが、XML では <company>Oracle</company> のように要素がデータとしてタグで囲まれます。たとえば、XML を使用すると、Web ページ内のワードと値にコンテキストを与え、単純なテキスト要素や数値要素ではなくデータとして識別できます。

HTML コードの例と、それに対応する XML の例を次に示します。それぞれの例は、次の従業員データを示しています。

- 従業員番号
- 氏名
- ジョブ
- 給与

HTML の例 1

```
<table>
   <tr><td>EMPNO</td><td>ENAME</td><td>JOB</td><td>SAL</td></tr>
   <tr><td>7654</td><td>MARTIN</td><td>SALESMAN</td><td>1250</td></tr>
   <tr><td>7788</td><td>SCOTT</td><td>ANALYST</td><td>3000</td></tr>
 </table>
```

XML の例 1

XML コードでは、XML データ・タグが追加され、要素の構造がネストされていることに注意してください。

```
<?xml version="1.0"?>
  <EMPLIST>
    <EMP>
    <EMPNO>7654</EMPNO>
    <ENAME>MARTIN</ENAME>
     <JOB>SALESMAN</JOB>
     <SAL>1250</SAL>
    </EMP>
    <EMP>
    <EMPNO>7788</EMPNO>
    <ENAME>SCOTT</ENAME>
    <JOB>ANALYST</JOB>
    <SAL>3000</SAL>
    </EMP>
  </EMPLIST>
```

HTML の例 2

タグを使用して表の 1 行のデータを表す次の HTML を考えてみます。「Java Programming」が書名なのか、大学の課程なのか、ジョブ・スキルなのかは、ページのデータとタグを見てもわかりません。これをコンピュータ・プログラムで示すことを考えてみます。

```
<HTML>
  <BODY>
    <TABLE>
     <TR>
     <TD>Java Programming</TD>
     <TD>EECS</TD>
     <TD>Paul Thompson</TD>
     <TD>Ron<BR>Uma<BR>Lindsay</TD>
     </TR>
    </TABLE>
  </BODY>
 </HTML>
```

同様の XML の例にも同じデータが含まれていますが、タグはデータの表示方法ではなくデータが表す情報を示します。つまり、「Java Programming」が大学の課程であることは明らかですが、その表示方法はわかりません。

XML の例 2

```
<?xml version="1.0"?>
  <Course>
      <Name>Java Programming</Name>
    <Department>EECS</Department>
    <Teacher>
      <Name>Paul Thompson</Name>
    </Teacher>
    <Student>
      <Name>Ron</Name>
    </Student>
    <Student>
      <Name>Uma</Name>
    </Student>
    <Student>
      <Name>Lindsay</Name>
    </Student>
  </Course>
```

XML と HTML は、どちらも情報を表します。

- XML は情報の内容を表します。
- HTML はその内容の表示方法を表します。

XML と HTML の違いのまとめ

表 A-1 に、XML と HTML の違いをまとめます。

表 A-1 XML と HTML の違い

XML	HTML
情報の内容を表します。	その内容の表示方法を表します。
ユーザー定義タグがあります。	固定のタグ・セットが標準により定義されています。
すべての開始タグには終了タグが必要です。	現行のブラウザでは、タグ <P>、<B> などについては、この要件が緩和されています。
属性は一重引用符または二重引用符で囲む必要があります。	現行のブラウザでは、この要件がタグについては緩和されています。
空の要素は明確に示されます。	現行のブラウザでは、この要件がタグについては緩和されています。
要素名と属性には大 / 小文字区別があります。	要素名と属性には大 / 小文字区別がありません。

スタイルシートを使用した XML の表示

XML をデータソースとして使用する場合の主なメリットは、表示（Web ページなど）を構造と内容から分離できることです。

- **表示**。表示は、適用されたスタイルシートにより定義されます。XML データは、単に異なるスタイルシートを適用するのみで、様々な形式と構成で表示できます。
- **構造と内容**。XML データでは構造と内容が定義されます。

スタイルシートの用途

スタイルシートについて次の使用方法を考えます。

- 表示スタイルごとに異なるスタイルシートを定義することで、ユーザー・プロファイル、ブラウザ・タイプまたは他の条件に基づいて、様々なユーザーに異なるインタフェースを表示できます。
- スタイルシートを使用すると、XML データを、それを受信して処理する特定のアプリケーションに合せて調整された形式に変換できます。

スタイルシートは、サーバー側またはクライアント側で適用できます。XSL-Transformation Processor（XSL-T Processor）により、ある XML 形式が別の XML 形式または HTML のような他のテキスト・ベース形式に変換されます。すべての Oracle XML Parsers には、XSL-T Processor が組み込まれています。

スタイルシートの適用方法と XSL-T Processor の使用方法は、『Oracle9*i* アプリケーション開発者ガイド - XML』の次の項を参照してください。

- 「XSL および XSLT の使用」
- 「XML Parser for Java の使用」

このマニュアルでは第 7 章「XSL を使用した Discoverer 4i Viewer のカスタマイズ」を参照してください。

eXtensible Stylesheet Language（XSL）

eXtensible Stylesheet Language（XSL）は XML のスタイルシート言語であり、もう 1 つの W3C 勧告です。XSL により、スタイルシートで次のことができます。

- XML から XML 形式または HTML のような他のテキスト・ベース形式への変換。
- データの再配置やフィルタリング。
- XML データから、別の Document Type Definition（DTD）に準拠する XML への変換。これは、様々なアプリケーションでデータの共有を可能にする重要な機能です。

カスケーディング・スタイルシート（CSS）

カスケーディング・スタイルシート（CSS1）は、当初は HTML ドキュメント用に作成された W3C 仕様です。CSS を使用すると、ドキュメントの表示のうち次の側面を制御できます。

- スペーシング。要素の可視性、位置およびサイズ。
- 色と背景。
- フォントとテキスト。

CSS2 は 1998 年に W3C により公開され、次の機能が追加されています。

- システム・フォントと色
- 自動採番
- ページ・メディアのサポート
- 表と音声スタイルシート

この場合のカスケーディングは、単一のドキュメントに複数のスタイルシートを適用できることを意味します。たとえば、CSS を使用する単一の Web ページに、次の 3 つのスタイルシートを適用、つまりカスケードできます。

1. ユーザー指定のスタイルシートが優先されます。
2. 次にカスケーディング・スタイルシートが適用されます。
3. 最後にブラウザのスタイルシートが適用されます。

> **関連項目：**『Oracle9*i* アプリケーション開発者ガイド - XML』の「XSL および XSLT の使用」

拡張性および Document Type Definition（DTD）

HTML を上回る XML のもう 1 つの重要なメリットは、タグの仕様とユーザーによる使用方法にあります。XML 文書を構成するには、データの意味と構造を表す独自のタグを作成します。

タグは、XML 文書内で使用することで定義する方法と、事前に Document Type Definition（DTD）で定義する方法があります。データやアプリケーションの要件が変化するたびに、タグを変更または追加して、新しいデータ・コンテキストを反映させたり、既存のデータ・コンテキストを拡張できます。

前述した XML の例に使用する単純な DTD を次に示します。

```
<!ELEMENT EMPLIST (EMP)*>
<!ELEMENT EMP (EMPNO, ENAME, JOB, SAL)>
<!ELEMENT EMPNO (#PCDATA)>
<!ELEMENT ENAME (#PCDATA)>
<!ELEMENT JOB (#PCDATA)>
<!ELEMENT SAL (#PCDATA)>
```

注意： DOCTYPE 宣言が使用されるのは、DTD が XML コードに埋め込まれている場合のみです。

整形式の有効な XML 文書

整形式の XML 文書

XML の構造および表記規則に準拠している XML 文書は、整形式とみなされます。整形式の XML 文書の場合、DTD を含めたり参照する必要はなく、データ要素とその関係を暗黙的に定義できます。整形式の XML 文書は、次の規則に従う必要があります。

- ドキュメントは、XML 宣言 <?xml version="1.0"> で始める必要があります。
- すべての要素は、単一のルート要素に含める必要があります。
- すべての要素は、重複なしにツリー構造でネストする必要があります。
- 空でない要素には、常に開始タグと終了タグが必要です。

妥当な XML 文書

整形式の XML 文書が DTD にも準拠している場合は、妥当であるとみなされます。DTD を含んでいるか参照している XML 文書が解析される場合、解析アプリケーションでは XML が DTD に準拠しているかどうか、つまり妥当性を検証できます。妥当な XML 文書は、解析アプリケーションで処理でき、すべてのデータ要素と内容が DTD に定義されている規則に従っていることが保証されます。

XML を使用する理由

XML は情報交換用のインターネット標準であり、次の理由で役立ちます。

- **データ交換に伴う問題の解決**。次のようなデータを効率的に通信できます。
  - 多数の異なる形式で様々なプラットフォームにある場合
  - 異なるプラットフォームに送信する必要がある場合
  - 異なる形式および表現で表示する必要がある場合
  - 多数の異なるエンド・デバイスで表示する必要がある場合

  つまり、XML により、アプリケーションのデータ交換の問題が解決されます。企業は、XML を使用して他の企業やワークフロー・コンポーネントと簡単に通信できるようになります。XML によるデータ交換問題の解決の詳細と例は、『Oracle9*i* アプリケーション開発者ガイド - XML』第 2 ～ 19 章を参照してください。

  XML を使用して Web ベースのアプリケーションを構築できます。これは、Web、データベース、ネットワークおよびミドルウェアのインターオペレーションに役立ちます。XML は、データ送信用に構造化された形式を提供します。

- **XML を使用して設計される業界固有のデータ・オブジェクト**。OAG や XML.org などの組織は、XML を使用して業界ごとにデータ・オブジェクトを標準化しています。これにより、B2B データ交換がさらに簡単になります。

- **XML を使用して簡単にアクセス、変換および格納されるデータベース常駐データ**。データベースには優れたデータ問合せ機能、拡張性および可用性があるため、大量のビジネス・データがリレーショナルおよびオブジェクト・リレーショナル表にあります。このデータを XML 形式から変換し、オブジェクト・リレーショナルおよび純粋なリレーショナル・データベース構造に格納したり、元の XML に生成してさらに処理できます。

XML を使用するその他のメリット

前述した以外にも、XML を使用すると次のようなメリットが得られます。

- 独自のタグを作成できます。
- XML は多数のツールでサポートされます。
- XML はオープンな標準です。
- オープンな標準に従って構築された XML パーサーは、相互運用可能であり、ベンダーによる制約が回避されます。XML 仕様は、業界で広範囲に承認されています。
- XML では、データ表現はデータの構造および内容から分離されます。データ表示を簡単にカスタマイズできます。「スタイルシートを使用した XML の表示」および「データ表示のカスタマイズ」を参照してください。
- 汎用性 － XML により、わかりやすい汎用的な方法でデータを表示できます。

- 永続性 ー データを XML 文書として実体化することで、プログラムでアクセスして操作できる状態のまま、このデータを永続的なものにできます。
- プラットフォームやアプリケーションに依存しません。
- 拡張性があります。

その他の XML リソース

XML に関するその他の情報源を次に示します。

- 『The Oracle XML Handbook』、Ben Chang、Mark Scardina 他著、Oracle Press 刊
- 『Building Oracle XML Applications』、Steve Muench 著、O'Reilly 刊
- 『XML Bible』、Elliotte Rusty Harold 著、IDG Books Worldwide 刊
- 『XML Unleashed』、Morrison 他著、SAMS 刊
- 『Building XML Applications』、St.Laurent および Cerami 著、McGraw-Hill 刊
- 『Building Web Sites with XML』、Michael Floyd 著、Prentice Hall PTR 刊
- 『Building Corporate Portals with XML』、Finkelstein および Aiken 著、McGraw-Hill 刊
- 『XML in a Nutshell』、O'Reilly 刊
- 『Learning XML - (Guide to) Creating Self-Describing Data』、Ray 著、O'Reilly 刊
- http://www.xml.com/pub/rg/46
- http://www.xml.org/xmlorg_resources/index.shtml
- http://www.xmlmag.com/
- http://www.webmethods.com/
- http://www.clarient.org/
- http://www.xmlwriter.com/
- http://webdevelopersjournal.com/articles/why_xml.html
- http://www.w3schools.com/xml/
- http://www.w3scripts.com/xml/default.asp
- http://www.xml101.com/examples/
- http://www.w3.org/TR/REC-xml
- http://msdn.microsoft.com/xml/default.asp
- http://www.w3.org/TR には、W3C のテクニカル・レポートのリストが表示されます。

- http://www.w3.org/xml は、W3C の XML アクティビティの概要ページです。
- http://www.xml.com には、XML に関する最新の業界ニュースが含まれています。
- http://www.xml-cml.org には、Chemical Markup Language（CML）に関する情報が表示されます。CML ドキュメントは、Jumbo ブラウザで表示および編集できます。
- http://www.loc.gov/ead/ には、米国議会図書館用に開発された Encoded Archival Description（EAD）の情報が表示されます。
- http://www.docuverse.com/xlf には、Extensible Log Format（XLF）に関する情報が表示されます。これは、ログ・ファイルを XML ログ・ファイルに変換し、ログ・ファイルの管理を簡素化するためのプロジェクトです。
- http://www.w3.org/Math には、アプリケーション間で等式を交換する方法を提供する MathML に関する情報が表示されます。
- http://www.naa.org には、分類された広告を簡単に交換できるようにする Newspaper Association of America（naa）の分類付き広告形式の説明が表示されます。
- http://www.w3.org/AudioVideo/ には、Synchronized Multimedia Integration Language（SMIL）に関する情報が表示されます。
- オラクル社は、OASIS の公式スポンサーです。OASIS、http://www.oasis-open.org は、XML アプリケーションの標準化を目的とする世界最大の独立系非営利団体です。あらゆる業界からの参加を募っており、競合する企業や重複する規格団体のとりまとめを行っています。

用語集

9*i*FS

「Oracle Internet File System」を参照。

API

アプリケーション・プログラム・インタフェース。「Application Program Interface（API）」を参照。

Application Program Interface（API）

一連のパブリック・プログラム・インタフェース。オペレーティング・システム、またはデータベース、Web サーバー、JVM などの他のプログラム環境と通信するための言語およびメッセージ形式で構成される。通常これらのメッセージは、アプリケーション開発に使用可能なファンクションおよびメソッドをコールする。

BC4J

Business Components for Java。

BFILES

オペレーティング・システム内に常駐するデータベース表領域の外に存在する外部バイナリ・ファイル。BFILE はデータベース・セマンティクスから参照され、外部 LOB ともいう。

BLOB

「バイナリ・ラージ・オブジェクト（Binary Large Object: BLOB）」を参照。

Business-to-Business（B2B）

商品販売およびサービス提供における企業間の相互のコミュニケーションを示す用語。これを実現するソフトウェア・インフラストラクチャが取引である。

Business-to-Consumer（B2C）

商品販売およびサービス提供における企業と顧客間のコミュニケーションを示す用語。

CDATA

「文字データ（Character Data: CDATA）」を参照。

CDF

チャネル定義形式。インターネット上のチャネルに関する情報を交換する方法を提供する。

CGI

「Common Gateway Interface（CGI）」を参照。

Class Generator

入力ファイルを受け入れ、対応する機能を持つ一連の出力クラスを作成するユーティリティ。XML Class Generator の場合、入力ファイルは DTD であり、出力はその DTD に準拠する XML 文書を作成するために使用できる一連のクラスである。

CLASSPATH

JVM がアプリケーションの実行に必要なクラスを検索するために使用する、オペレーティング・システムの環境変数。

CLOB

「キャラクタ・ラージ・オブジェクト（Character Large Object: CLOB）」を参照。

Common Gateway Interface（CGI）

Web サーバーが他のプログラムを実行し、ブラウザに送信された HTML ページ、図形、オーディオおよびビデオに出力を渡すことを可能にするプログラム・インタフェース。

Common Object Request Broker API（CORBA）

ネットワーク全体の分散オブジェクト間の通信用の、Object Management Group による標準。これらの自己完結型ソフトウェア・モジュールは、異なるプラットフォームまたはオペレーティング・システム上で実行しているアプリケーションで使用できる。CORBA オブジェクトとそのデータ形式、およびファンクションは、Java、C、C++、Smalltalk、COBOL などの様々な言語にコンパイルできるインタフェース定義言語（IDL）で定義される。

Common Oracle Runtime Environment（CORE）

C 言語で作成されたファンクションのライブラリ。これによって開発者は、事実上すべてのプラットフォームおよびオペレーティング・システムに簡単に移植できるコードを作成できる。

CORBA

「Common Object Request Broker API（CORBA）」を参照。

CSS

カスケーディング・スタイルシート。

DOCTYPE

XML 文書内に DTD またはその参照を指定するタグ名として使用される用語。たとえば、<!DOCTYPE person SYSTEM "person.dtd"> では、ルート要素名を person として、また外部 DTD を person.dtd としてファイル・システム内に宣言される。内部 DTD は、DOCTYPE 宣言内で宣言される。

Document Type Definition（DTD）

XML 文書の使用可能な構造を定義する一連の規則。DTD は、SGML から書式を導出し、DOCTYPE 要素を使用するか、あるいは DOCTYPE 参照を介して外部ファイルを使用して XML 文書内に含めることができるテキスト・ファイルである。

DOM

「ドキュメント・オブジェクト・モデル（Document Object Model: DOM）」を参照。

DTD

「Document Type Definition（DTD）」を参照。

EDI

電子データ交換。

Enterprise JavaBeans（EJB）

サーバー上の JVM 内で実行する独立プログラム・モジュール。CORBA によって EJB のインフラストラクチャが提供され、コンテナ・レイヤーによってサポートされたサーバーにセキュリティ、トランザクション・サポートおよびその他の共通機能が提供される。

Extensible Markup Language（XML）

データ記述のオープン標準。W3C の開発による SGML 構文のサブセットを使用した標準で、インターネットで使用できるよう設計されている。現行の標準はバージョン 1.0 で、1998 年 2 月に W3C 勧告として公開された。

eXtensible Stylesheet Language（XSL）

XML 文書を変換またはレンダリングするために、スタイルシート内で使用される言語。（W3C）の XSL は、XSL Transformations および XSL Formatting Objects という 2 つの W3C 勧告で構成されている。XSL Transformations は 1 つの XML 文書を別の XML 文書に変換し、XSL Formatting Objects は XML 文書の表示を指定する。XSL は、スタイルシートを表す言語である。XSL は、次の 2 つで構成される。

- XML 文書を変換するための言語（XSLT）
- 書式設定セマンティクスを指定するための XML ボキャブラリ（XSLFO）

XSL スタイルシートは、書式設定用ボキャブラリを使用する XML 文書へのクラスのインスタンスの変換方法を記述して、XML 文書のクラス表示を指定する。

eXtensible Stylesheet Language Formatting Object（XSLFO）

書式設定セマンティクスを指定するための XML 用語を定義する、W3C の標準仕様。

eXtensible Stylesheet Language Transformation（XSLT）

XSLT とも書く。XML 文書を別のドキュメントに変換する言語を定義する、W3C の XSL 標準仕様。

HTML

「Hypertext Markup Language（HTML）」を参照。

HTTP

「Hypertext Transfer Protocol（HTTP）」を参照。

Hypertext Markup Language（HTML）

Web ブラウザに送信するファイルを作成するのに使用し、Web の基礎として機能するマークアップ言語。HTML の次のバージョンは xHTML と呼ばれ、XML アプリケーションになる予定である。

Hypertext Transfer Protocol（HTTP）

インターネットを介して、Web サーバーとブラウザ間で HTML ファイルを転送するのに使用するプロトコル。

IDE

「統合開発環境（Integrated Development Environment: IDE）」を参照。

***inter*Media**

複合データ型のコレクションおよび Oracle9*i* 内でのコレクションのアクセスを示す用語。これには、テキスト、ビデオ、時系列および空間データ型が含まれる。

Internet Inter-ORB Protocol（IIOP）

インターネットなどの TCP/IP ネットワーク上で、メッセージを交換するために CORBA が使用するプロトコル。

Java

Sun Microsystems 社によって開発およびメンテナンスされた高水準のプログラミング言語。Java では、アプリケーションが JVM という仮想マシン内で実行する。JVM は、オペレーティング・システムに対するすべてのインタフェースの役割を担う。開発者は、このアーキテクチャによって、JVM を搭載するすべてのオペレーティング・システムまたはプラットフォームで実行する Java アプリケーションおよびアプレットを開発できる。

Java Database Connectivity（JDBC）

Java アプリケーションが、SQL 言語を介してデータベースにアクセスできるようにするプログラム API。JDBC ドライバはプラットフォームに依存しないように Java で作成されるが、各データベースに固有である。

Java Development Kit（JDK）

Java 開発環境を確立する Java バージョンの、Java クラス、ランタイム、コンパイラ、デバッガおよびソース・コードのコレクション。JDK はバージョンで指定され、Java 2 はバージョン 1.2 以上を指定するために使用される。

Java Runtime Environment（JRE）

プラットフォームごとの Java プログラムの実行環境で、コンパイラやデバッガを含まない。JRE はバージョンで指定され、Java 2 はバージョン 1.2 以上を指定するために使用される。

Java Virtual Machine（Java VM）

コンパイル済み Java バイトコードをプラットフォームのマシン言語に変換し、それを実行する Java インタプリタ。JVM は、クライアント側、ブラウザ内、中間層内、OAS などのアプリケーション・サーバー上、または Oracle9*i* などのデータベース・サーバー内で実行できる。

Java VM

「Java Virtual Machine（Java VM）」を参照。

JavaBeans

JVM 内で実行する独立プログラム・モジュール。通常、クライアント側のユーザー・インタフェースを作成するのに使用される。サーバー側の同等のモジュールは、Enterprise JavaBeans（EJB）という。「Enterprise JavaBeans（EJB）」を参照。

JavaServer Pages（JSP）

Web ページへの単純なプログラム・インタフェースを可能にする、サーブレットの拡張機能。JSP は、特殊タグ、および Web またはアプリケーション・サーバーで実行される埋込み Java コードを含む HTML ページであり、HTML ページに動的機能を提供する。JSP は、サーバーの JVM で最初に要求および実行されるときに、サーブレットにコンパイルされる。

JDBC

「Java Database Connectivity（JDBC）」を参照。

JDeveloper

アプリケーション、アプレットおよびサーブレットの開発を可能にする Oracle の Java IDE。エディタ、コンパイラ、デバッガ、構文チェッカ、ヘルプ・システムなどを含む。バージョン 3.1 の JDeveloper は、エディタで XML がサポートされるとともに、簡単に使用できるように統合された Oracle XDK for Java を搭載して、XML ベースの開発をサポートするように拡張されている。

JDK

「Java Development Kit（JDK）」を参照。

LAN

「Local Area Network（LAN）」を参照。

LOB

「ラージ・オブジェクト（Large Object: LOB）」を参照。

Local Area Network（LAN）

限定された地域内のユーザー用のコンピュータ通信ネットワーク。LAN は、サーバー、ワークステーション、通信ハードウェア（ルーター、ブリッジ、ネットワーク・カードなど）およびネットワーク・オペレーティング・システムで構成される。

NCLOB

「各国語キャラクタ・ラージ・オブジェクト（National character Large Object）」を参照。

N 層（N-tier）

クライアントおよびサーバーで構成される 1 つ以上の層からなる、コンピュータ通信ネットワーク・アーキテクチャの指定。通常、2 層システムは 1 つのクライアント・レベルおよび 1 つのサーバー・レベルで構成される。3 層システムは、通常、1 つのクライアント層とともに、データベース・サーバーと Web またはアプリケーション・サーバーの 2 つのサーバー層を使用する。

OAG

Open Applications Group。

OAI

Oracle Applications Integrator。Oracle iStudio 開発ツールを含むランタイム。CRM アプリケーションを Oracle ERP に加えて他の ERP システムと統合する。固有の API は、メッセージ対応である必要がある。標準の拡張フックを使用して、他のアプリケーション・システムと交換された XML ストリームを生成または解析する。開発中である。

OAS

「Oracle Application Server」を参照。

OASIS

「Organization for the Advancement of Structured Information（OASIS）」を参照。

Object Request Broker（ORB）

クライアント側の要求元プログラムとサーバー側のオブジェクト間のメッセージ通信を管理するソフトウェア。ORB は、アクション要求およびそのパラメータをオブジェクトに渡し、結果を戻す。共通の実装は、CORBA および EJB である。「Common Object Request Broker API（CORBA）」を参照。

OE

Oracle Exchange。

OIS

「Oracle Integration Server（OIS）」を参照。

Oracle Application Server（OAS）

Oracle アプリケーション・サーバー。オープン標準フレームワーク内で、高パフォーマンスで N 層のトランザクション指向 Web アプリケーションを構築、配置および管理するために必要なすべての主要なサービスおよび機能を統合する。

Oracle Integration Server（OIS）

アプリケーション統合のためのメッセージング・ハブとして機能する Oracle のサーバー製品。OIS には、AQ および Oracle Workflow を搭載した Oracle8*i* データベース、および Oracle Message Broker を使用して、アプリケーション間で XML 形式のメッセージを転送するアプリケーションへのインタフェースが含まれる。

Oracle Internet File System

Oracle9*i* データベース内または中間層上で実行する、Oracle のファイル・システムおよび Java ベースの開発環境。単一のデータベース・リポジトリに複数の型のドキュメントを作成、格納および管理する方法を提供する。

Oracle JVM

Oracle データベースのメモリー領域内で実行する Java VM。JVM は、Oracle 8*i* リリース 8.1.5 では Java1.1 準拠であり、リリース 8.1.6 以降では Java1.2 準拠である。

ORACLE_HOME

アプリケーションで Oracle データベースを使用する際の、Oracle データベースのインストール場所を識別するオペレーティング・システムの環境変数。

ORB

「Object Request Broker（ORB）」を参照。

Organization for the Advancement of Structured Information（OASIS）

会議、セミナー、展示会およびその他の教育イベントを通じて、パブリック情報標準の普及促進を目的として設立されたメンバーの組織。XML は、SGML と同様に、OASIS が活発に普及を促進している標準である。

Parser

XML で、XML 文書を入力として受け入れ、文書が整形式であり、また妥当（オプション）であるかどうかを判断するソフトウェア・プログラム。Oracle XML Parser は、SAX および DOM インタフェースの両方をサポートする。

PCDATA

「解析対象文字データ（Parsed Character Data: PCDATA）」を参照。

PDA

Palm Pilot などの携帯情報端末。

PL/SQL

データベース内で実行できるプログラムを作成するために SQL を拡張した、Oracle 手続き型言語。

PUBLIC

後に続く参照の、インターネット上の場所を指定する用語。

RDF

リソース記述フレームワーク。

SAX

「Simple API for XML（SAX）」を参照。

Secure Sockets Layer（SSL）

インターネット上のプライマリ・セキュリティ・プロトコル。ブラウザとサーバー間の暗号化形式に、公開鍵 / 秘密鍵を使用する。

SGML

「Standard Generalized Markup Language（SGML）」を参照。

Simple API for XML（SAX）

XML Parser によって提供され、イベント駆動型のアプリケーションによって使用される XML 標準インタフェース。

SQL

「Structured Query Language（SQL）」を参照。

SSI

「サーバー側インクルード（Server-side Include: SSI）」を参照。

SSL

「Secure Sockets Layer（SSL）」を参照。

Standard Generalized Markup Language（SGML）

マークアップおよびDTDを使用して実装された、テキスト・ドキュメントの書式を定義するISO標準。

Structured Query Language（SQL）

リレーショナル・データベース内のデータのアクセスおよび処理に使用する標準言語。

SYSTEM

後に続く参照の、ホスト・オペレーティング・システム上の場所を指定する用語。

TCP/IP

「Transmission Control Protocol/Internet Protocol（TCP/IP）」を参照。

Transmission Control Protocol/Internet Protocol（TCP/IP）

TCPおよびIPで構成される通信ネットワーク・プロトコル。TCPはトランスポート機能を制御し、IPはルーティング・メカニズムを提供する。TCP/IPは、インターネット通信の標準である。

Transviewer

XDK for Javaに含まれるOracle XML JavaBeans（XML JavaBeans）を示すOracle用語。これらのBeanには、XML Source View Bean、Tree View Bean、DOMParser Bean、Transformer BeanおよびTransViewer Beanが含まれる。

Uniform Resource Identifier（URI）

URLおよびXPathの作成に使用するアドレス構文。

Uniform Resource Locator（URL）

インターネット上のファイルの場所およびルートを定義するアドレス。URLは、ブラウザでWebをナビゲートする際に使用され、プロトコル接頭辞、ポート番号、ドメイン名、ディレクトリ名とサブディレクトリ名、およびファイル名で構成される。たとえば、http://technet.oracle.com:80/tech/xml/index.htmでは、Web上のOTNのXMLサイトを検索するためにブラウザが移動する、場所およびパスが指定されている。

URI

「Uniform Resource Identifier（URI）」を参照。

URL

「Uniform Resource Locator（URL）」を参照。

W3C

「World Wide Web Consortium（W3C）」を参照。

WAN

「Wide Area Network（WAN）」を参照。

Web Request Broker（WRB）

URL を処理し、適切なカートリッジに送信する OAS 内のカートリッジ。

Wide Area Network（WAN）

州や国などの広域内のユーザー用のコンピュータ通信ネットワーク。WAN は、サーバー、ワークステーション、通信ハードウェア（ルーター、ブリッジ、ネットワーク・カードなど）およびネットワーク・オペレーティング・システムで構成される。

World Wide Web Consortium（W3C）

1994 年に設立された、Web の標準を確立するための国際的な産業組合。W3C のサイトは、www.w3c.org である。

XLink

XML 文書内でのハイパーリンクの使用を制御する規則で構成された XML Linking 言語。これらの規則は、W3C の勧告プロセス下の XML Linking Group によって開発されている。XLink は、XML がドキュメントの表示およびハイパーリンクの管理をサポートする 3 つの言語（Xlink、Xpointer および XPath）の 1 つである。

XML

「Extensible Markup Language（XML）」を参照。

XML Developer's Kit（XDK）

ソフトウェア開発者に、アプリケーションを XML 対応にするための標準ベースの機能を提供する、一連のライブラリ、コンポーネントおよびユーティリティ。Oracle XDK for Java には、XML Parser、XSL Processor、XML Class Generator、Transviewer Beans および XSQL Servlet が含まれる。

XML Query

W3C が取り組む、XML 文書を問い合せるための言語および構文の標準。

XML Schema

W3C が取り組む、XML 文書内で単純なデータ型および複合構造を表すための標準。データ型の定義や妥当性など、現在 DTD で不足している領域に取り組んでいる。Oracle XML Schema Processor は、オンライン取引などの E-Business アプリケーションで使用される、XML 文書およびデータの妥当性を自動的に確認する。XML Schema は、XML 文書に単純および複雑なデータ型を追加し、DTD の機能を XML Schema 定義の XML 文書に置き換える。

XPath

XSL および XPointer で使用されるドキュメント内で要素を指定するための、オープン標準の構文。XPath は、現在 W3C 勧告である。XSLT、XLink および XML Query に使用される XML 文書を操作するためのデータ・モデルおよび文法を指定する。

XPointer

XML 文書フラグメントへの参照を記述するための W3C 勧告。XPointer は、XPath 形式の URI の終わりに使用できる。XPath ナビゲーションを使用して、XML 文書内の個別のエンティティまたはフラグメントの識別を指定する。

XSL

「Extensible Markup Language（XML）」を参照。

XSLFO

「eXtensible Stylesheet Language Formatting Object（XSLFO）」を参照。

XSLT

「eXtensible Stylesheet Language Transformation: XSLT）」を参照。

XSQL

Oracle XSQL Servlet で使用される指定。1 つ以上の SQL 問合せから動的 XML 文書を生成し、XML スタイルシートを使用して、サーバー内のドキュメントを変換する（オプション）機能を提供する。

アプリケーション・サーバー（Application Server）

アプリケーションおよびその環境をホストするために設計されたサーバーであり、サーバー・アプリケーションの実行を許可する。代表的な例は OAS で、リモート・クライアントがインタフェースを制御する場合に、Java、C、C++ および PL/SQL アプリケーションをホストできる。「Oracle Application Server（OAS）」を参照。

インスタンス化（instantiate）

Java や C++ などのオブジェクト・ベース言語で使用される用語で、特定のクラスのオブジェクト作成を示す。

エンティティ（entity）

別の文字列、またはドキュメントのキャラクタ・セットに属さない特殊文字を表すことができる文字列。エンティティおよびパーサーによってエンティティの代替となるテキストは、DTD に宣言される。

オブジェクト・ビュー（Object View）

1 つ以上のオブジェクト表または他のビューに含まれるデータの調整された外観。オブジェクト・ビュー問合せの出力は、表として処理される。オブジェクト・ビューは、表が使用されているほとんどの場所で使用できる。

オブジェクト・リレーショナル（object-relational）

テキスト・ドキュメント、オーディオ・ファイル、ビデオ・ファイル、ユーザー定義オブジェクトなどの高順序のデータ型を格納および操作できるリレーショナル・データベース・システムを示す用語。

親要素（parent element）

子要素という別の要素を囲む要素。たとえば、<Parent><Child></Child></Parent> は、Parent 要素がその Child 要素をラップしていることを示す。

カートリッジ（cartridge）

Java または PL/SQL のストアド・プログラム。データベースが新しいデータ型を理解および処理するために必要な機能を追加する。カートリッジは、Oracle 内の拡張フレームワークでインタフェースの役割を担う。Oracle Text はこの種類のカートリッジであり、データベース内に格納されたテキスト・ドキュメントの読込み、書込みおよび検索のサポートを追加する。

解析対象文字データ（Parsed Character Data: PCDATA）

解析する必要があるが、タグまたは解析対象外データの一部ではないテキストで構成される要素内容。

各国語キャラクタ・ラージ・オブジェクト（National Character Large Object: NCLOB）

データベースの各国語キャラクタ・セットに対応する文字データで構成された値を持つ LOB データ型。

空要素（empty element）

テキスト内容または子要素のない要素。属性およびその値のみを含む場合がある。空要素の書式は、<name/> または <name></name>（タグの間には空白なし）である。

キャラクタ・ラージ・オブジェクト（Character Large Object: CLOB）

データベース・キャラクタ・セットに対応する文字データで構成された値を持つ LOB データ型。CLOB は、Oracle Text の検索エンジンを使用して索引付けおよび検索できる。

クライアント / サーバー（client-server）

実際のアプリケーションはクライアント側で実行するが、ネットワークを介して、サーバー側のデータまたは他の外部プロセスにアクセスするアプリケーション・アーキテクチャを表す用語。

結果セット（result set）

1 行以上のデータで構成される SQL 問合せの出力。

コールバック（callback）

1 つのプロセスに他のプロセスを開始させ、それを継続させるプログラム方法。2 番目のプロセスは、アクションの結果、値または他のイベントとして 1 番目のプロセスをコールする。この方法は、継続的な対話を許可するユーザー・インタフェースを持つほとんどのプログラムに使用されている。

コマンドライン（command line）

ユーザーがコマンド・インタプリタのプロンプトにコマンドを入力するインタフェース・メソッド。

子要素（child element）

親要素という別の要素内に完全に含まれた要素。たとえば、
<Parent><Child></Child></Parent> は、Child 要素がその Parent 要素内にネストされていることを示す。

サーバー側インクルード（Server-side Include: SSI）

データまたは他の内容を、要求元のブラウザに送信する前に Web ページに置く HTML コマンド。

サーブレット（servlet）

サーバー（通常は Web またはアプリケーション・サーバー）内で実行される Java アプリケーション。このアプリケーションは、サーバー上で処理を実行する。サーブレットは、CGI スクリプトに相当する Java である。

スキーマ（schema）

データベース内の構造およびデータ型の定義。スキーマは、XML Schema の W3C 勧告をサポートする XML 文書も示す。

スタイルシート（Stylesheet）

XML において、XML 処理命令で構成される XML 文書を示す用語。XML 処理命令は、入力 XML 文書を出力 XML 文書に変換またはフォーマットするために、XSL Processor によって使用される。

スレッド（thread）

Windows、UNIX および Java などの複数のオペレーティング・システムがサポートする。1 つのプロセス内の、単一のプログラム実行単位。

整形式（well-formed）

XML 文書が、XML 宣言で宣言された XML バージョンの構文に準拠している状態を示す用語。これには、ルート要素が単一か、タグが適切にネストされているかなどが含まれる。

セッション（session）

2 つの層の間のアクティブ接続。

属性（attribute）

要素のプロパティ。等号で区切られた名前および値で構成され、開始タグ内の要素名の後に含まれる。たとえば、<Price units='USD'>5</Price> では、units（単位）が属性で USD がその値である。値は、一重または二重引用符で囲む必要がある。属性は、ドキュメントまたは DTD 内に格納できる。要素には多くの属性を指定できるが、その取得順序は定義されない。

タグ（tag）

XML マークアップの単一のピース。要素の開始または終了を指定する。タグは、< で始まり > で終わる。XML には、開始タグ（<name>）、終了タグ（</name>）および空タグ（<name/>）がある。

妥当（valid）

XML 文書の構造および要素内容が、参照 DTD または内部 DTD で宣言されたものと一貫している状態を示す用語。

データグラム（datagram）

XSQL Servlet が処理した SQL 問合せから、HTML ページに埋め込まれたリクエスタに戻されるテキストのフラグメント。XML 形式の場合もある。

データベース・アクセス記述子（Database Access Descriptor: DAD）

データベース・アクセスに使用される、名前付きの一連の構成値。DAD は、データベース名や SQL*Net V2 サービス名などの情報、ORACLE_HOME ディレクトリ、および言語、ソート型、日付言語などの NLS 構成情報を指定する。

統合開発環境（Integrated Development Environment: IDE）

ソフトウェアの開発を支援するために設計された、単一のユーザー・インタフェースから実行されるプログラム・セット。JDeveloper は、Java 開発用の IDE であり、エディタ、コンパイラ、デバッガ、構文チェッカ、ヘルプ・システムなどを含む。JDeveloper を使用すると、単一のユーザー・インタフェースを介して Java ソフトウェアを開発できる。

ドキュメント・オブジェクト・モデル（Document Object Model: DOM）

XML 文書のメモリー内ツリーベースのオブジェクト表現。要素および属性へのプログラム・アクセスを可能にする。DOM オブジェクトおよびそのインタフェースは、W3C 勧告である。プログラム・アクセス用の API など、XML 文書の DOM を指定する。DOM は、解析対象ドキュメントをオブジェクトのツリーとして表示する。

名前空間（namespace）

XML 文書内にある、関連する一連の要素名または属性を示す用語。名前空間の構文およびその使用方法は、W3C 勧告によって定義されている。たとえば <xsl:apply-templates/ > 要素は、XSL 名前空間の一部として識別される。名前空間は XML 文書または DTD 内で宣言されてから、属性の構文 xmlns:xsl="http://www.w3.org/TR/WD-xsl" で使用される。

バイナリ・ラージ・オブジェクト（Binary Large Object: BLOB）

内容がバイナリ・データで構成されたラージ・オブジェクト・データ型。このデータは、データ構造がデータベースに認識されないため、RAW 型とみなされる。

ハイパー・テキスト（hypertext）

ユーザーがハイパーリンクとして指定されたワードまたは句を選択して、他のドキュメントまたは図形間を操作できるテキスト・ドキュメントを作成および公開する方法。

表記法（NOTATION）

XML において、Parser が理解できない内容の型の定義。これらの型には、オーディオ、ビデオおよび他のマルチメディアが含まれる。

プロローグ（prolog）

XML 宣言および DTD、またはドキュメントの処理に必要な他の宣言を含む、XML 文書の最初の部分。

モード（mode）

XML において、DOM ツリー内のアドレス指定可能な各エンティティを示す用語。

文字データ（character Data: CDATA）

文書内の解析対象外のテキストは、CDATA セクションに格納される。これによって、&、<、> などの、他に特別な機能を持つ文字を含めることができる。CDATA セクションは、要素の内容または属性内で使用できる。

ユーザー・インタフェース（user interface: UI）

メニュー、スクリーン、キーボード・コマンド、マウス・クリック、およびユーザーによるソフトウェア・アプリケーションとの対話方式を定義するコマンド言語の組合せ。

要素（element）

XML 文書の基本論理単位。子、データ、属性とその値などの他の要素に対するコンテナとして機能する。要素は開始タグ <name> および終了タグ </name> で、空要素の場合は <name/> で識別される。

ラージ・オブジェクト（Large Object: LOB）

内部 LOB および外部 LOB に分割された SQL データ型のクラス。内部 LOB には BLOB、CLOB および NCLOB が含まれ、外部 LOB には BFILE が含まれる。「BFILE」、「バイナリ・ラージ・オブジェクト（Binary Large Object: BLOB）」および「キャラクタ・ラージ・オブジェクト（Character Large Object: CLOB）」を参照。

ラッパー（Wrapper）

通常、汎用またはオブジェクト・インタフェースを提供するために、他のデータまたはソフトウェアをラップするデータ構造またはソフトウェアを示す用語。

リスナー（listener）

入力プロセスを監視する個別のアプリケーション・プロセス。

ルート要素（root element）

XML 文書内にある他のすべての要素を囲む要素。オプションのプロローグとエピローグの間に存在する。XML 文書には、1 つのルート要素のみ置くことができる。

レンダラ（renderer）

ドキュメントを指定された形式に出力するソフトウェア・プロセッサ。

ワーキング・グループ（Working Group: WG）

特定のインターネット・テクノロジ分野における勧告を実行する業界のメンバーで構成された W3C の委員会。

索引

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

あ

い

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

と

な

に

ね

は

ひ

ふ

へ