IBM DB2 Information Integrator

アプリケーション開発者向けガイド

バージョン 8.2

SC88-9609-01
(英文原典：SC18-7359-01)

IBM DB2 Information Integrator

アプリケーション開発者向けガイド

バージョン 8.2

SC88-9609-01
(英文原典：SC18-7359-01)

ご注意！

本書および本書で紹介する製品をご使用になる前に、 299 ページの『特記事項』 に記載されている情報をお読みください。

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

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

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

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

(URL は、変更になる場合があります)

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

原 典： SC18-7359-01
IBM DB2 Information Integrator
Application Developer's Guide
Version 8.2

発 行： 日本アイ・ビー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2004.8

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

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

目次

まえがき

本書では、統一ビューとデータ配置を介してデータを統合するのに IBM® DB2 Information Integrator が有用なソリューションである理由を述べています。また、情報を最も有効に使用する方法も示しています。

本書の対象読者

データ管理者、情報分析者、システム統合担当者、Web 統合担当者、データ・ライブラリアン、データ設計者、およびアプリケーション開発者は、IBM DB2 Information Integrator ソリューションを使って戦略的かつオープンなインフォメーション・インテグレーション・プラットフォームを作成することができます。

用語

IBM DB2 Information Integrator では、データベース、接続、構造化照会言語 (SQL)、およびローカル・エリア・ネットワーク (LAN) の概念を表す標準用語が使われています。本書で用いられている DB2 Information Integrator の概念はすべて、用語集に定義されています。他に指定がないかぎり、次のような意味が想定されます。

データ 生の事実。これは、構造化、非構造化、または半構造化することができます。通常、データは分析できるよう編成されます。データはまた、決定をくだす際にも有用です。

情報 使用可能フォームのデータ。通常はいずれかの方法で処理または解釈されます。

DB2 Information Integrator の特徴

IBM DB2 Information Integrator はいくつかの製品に分かれています。ライセンス使用条件を熟読して、インストールするエディションの使用条件を確かめてください。Information Integrator のインストールおよび構成の詳細は、「*DB2 Information Integrator インストール・ガイド*」を参照してください。インフォメーション・インテグレーションの進化に関する詳細は、インフォメーション・インテグレーションのサポート・サイト http://www.ibm.com/software/data/integration/db2ii/support.html を参照してください。

第 1 章 インフォメーション・インテグレーションの開発の概要

第 1 部では、社内の情報を統合するためのアプリケーションの開発に関する概念と手順について説明します。

インフォメーション・インテグレーション・ソリューションの概要

業務の中で、インフォメーション・インテグレーションに取り組む必要に迫られることはよくあります。経済情勢の急激な変化によって、情報へのアクセス手段の改善、分析機能の柔軟化、および情報品目の正式決定の必要性が生じています。

インフォメーション・インテグレーションの概要

IBM® は過去 30 年以上の間、データ管理テクノロジーを世界規模で提供しつづけてきました。 IBM は大中小の規模の企業向けのインフォメーション・インテグレーション機能の開発を通して、企業への貢献をさらに推進しつつあります。インフォメーション・インテグレーションは、既存のデータ管理ソリューションを確固たる基盤としてその上に築かれています。インフォメーション・インテグレーションは、現在の市場に存在するデータのボリュームと多様性のいずれをも容易に管理するためのエンドツーエンドのソリューションを実現します。

業務の運営コストには、多様性があってしかも互いにつながりのないインフラストラクチャーの統合が含まれます。企業には、次のような目標が必要です。

- 新規業務どうしをシームレスに統合し、ビジネス・アプリケーションをレガシー・システムにリンクする。
- それぞれ異なるシステムの管理や、自動化の各種手段にまたがる統合にかかるコストの高騰を制御する。
- 新しい市場に速やかに進出しつつ、人員およびスキルの不足を補う。

情報へのアクセスや情報管理を効率よく行うためのソリューションの必要性は、製品や業界の境界線を越えて広がります。

関連概念:

- 6 ページの『DB2 Information Integrator — 統合のソリューション』
- 1 ページの『インフォメーション・インテグレーションとは何か』
- 2 ページの『企業にとってインフォメーション・インテグレーションが重要な理由』

インフォメーション・インテグレーションとは何か

インフォメーション・インテグレーションとは、データベース管理システム、Web サービス、レプリケーション、フェデレーテッド・システム、およびウェアハウジング機能を 1 つの共通プラットフォームにまとめたテクノロジーの集まりのことです。また、さまざまなプログラミング・インターフェースとデータ・モデルも組み込まれます。インフォメーション・インテグレーション・テクノロジーを使えば、さまざまなタイプのデータ (構造化、非構造化、および半構造化) にアクセスできる

ようになります。それらのデータを、社内のどこの情報にでも簡単にアクセスするのに使えるフォーマットにトランスフォームすることができます。

インフォメーション・インテグレーションは、データとコンテンツのソースを次のような機能に統合するための手段になります。

- リアルタイムの読み取りおよび書き込みアクセスを実現する。
- 業務分析およびデータ交換用にデータをトランスフォームする。
- パフォーマンス、通用性、および可用性をふまえてデータ配置を管理する。

関連概念:

- 6 ページの『DB2 Information Integrator — 統合のソリューション』

企業にとってインフォメーション・インテグレーションが重要な理由

IBM® インフォメーション・インテグレーション・ストラテジーは、以下を行います。

- 既存のデータを操作する機能をユーザーに提供する。
- 慣れ親しんだソフトウェアの利点をいかして既知の資産とリソースを使用する機能をユーザーに提供する。
- 新しいデータを獲得してより簡単に保守する機能をユーザーに提供する。
- データがどこにあっても、既存のデータ管理ツールを使ってそのデータにアクセスする機能をユーザーに提供する。

IBM では、オープン・サービス・インフラストラクチャーに基づいた 5 通りの統合が確認されています。それらの統合をまとめて使ったり別々に使ったりして、業務上の問題点を解決することができます。ここに一覧で示してある 5 通りの統合は、今日の商業界が取り組んでいるさまざまな統合への対策を表しています。インフォメーション・インテグレーションは、その種の統合の核をなしています。

ユーザー対話

ユーザーはトランザクションの完全サポートを使って、実質的にどのような装置からでも利用できる単一の調整済みユーザー・インターフェースで作業することができます。ユーザーの対話の結果は、複数のビジネス・システムに統合されます。

プロセスの統合

企業は、社内および社外の両方の人員および異種システムを対象としたプロセスのモデル化、自動化、およびモニターを介して業務の運営方法を変更することができます。

アプリケーションの接続

アプリケーションを互いに接続して、情報の共用と利用を企業レベルで改善することができます。

統合のための構築

ユーザーは、Web サービスと既存の資産を用いて、統合に対応したアプリケーションを作成してデプロイすることができます。新規のソリューションを既存の社有資産に統合することができます。

インフォメーション・インテグレーション

さまざまなフォームの業務情報を企業全体を通して統合することができます。統合することによって、情報資産の統一ビューを介した一貫性のある検索、アクセス、レプリケーション、トランスフォーメーション、および分析を行って、企業ニーズを満たすことができます。

IBM DB2® Information Integrator は、インフォメーション・インテグレーションに対する顧客要件に対処するための包括的ソリューションを実現するテクノロジーです。 DB2 Information Integrator の機能には、情報アクセス、インフォメーション・インテグレーション、および情報分析などがあります。

統合に対する最近のニーズは、アプリケーションとインフォメーション・インテグレーションの両方の混合です。ソリューション提供企業によって、情報の統合、共用、および配布を行うための多数の手段が用意されています。多種多様な統合の境界をあえて定めるよりも、さまざまな統合の要件ごとにマッチするそれぞれ異なった利点をもつアプローチをとるほうがより重要になってきます。

関連概念:

- 1 ページの『インフォメーション・インテグレーションとは何か』
- 3 ページの『DB2 Information Integrator が解決する問題』

インフォメーション・インテグレーションによってアプリケーション開発が容易になる理由

今日存在する業務システムは、顧客、供給業者、パートナー、および電子マーケットプレイスなどで成り立っています。このようなシステムが対話する相手としては、データベース、アプリケーション・サーバー、コンテンツ・マネージメント・システム、データウェアハウス、ワークフロー・システム、検索エンジン、メッセージ・キュー、Web クローラー、マイニングおよび分析パッケージ、およびその他の社内アプリケーションなどがあります。業務システムには、Open Database Connectivity、Java Database Connectivity、Web サービス、Java オブジェクト、および Java 2 Platform, Enterprise Edition などのさまざまなプログラミング・インターフェースが必要です。また業務システムでは、構造化照会言語、Extensible Markup Language、Web サービス記述言語、および Simple Object Access Protocol などのさまざまなデータ・モデルおよび言語を取り扱う必要があります。

インフォメーション・インテグレーションは、データ管理システム、データウェアハウス、Web サービス、およびその他の社内アプリケーションの中核を成すエレメントを 1 つの共通プラットフォームにまとめるテクノロジー・アプローチです。

DB2 Information Integrator が解決する問題

IBM® DB2® Information Integrator は、企業の問題の多くに対処するソリューションとして機能します。 DB2 Information Integrator を使って、以下を行うことができます。

- すべてのフォームの情報を管理する。管理には、DB2 Information Integrator に備えられているストレージ、統合、アーカイブ、トランスフォーメーション、ロード、モニター・アクセス、レプリケーション・セキュリティー、およびクレンジングの各種技法を活用します。

- XML、SQL、Web サービス、およびメッセージング・アプリケーションを利用するローカル・データとフェデレーテッド・データの照会モデルを統一する。
- マイニング、カテゴリー化、要約、およびリアルタイムの意思決定を使って効率よく情報を分析する。

以下の表は、顧客問題の典型例をいくつか概略しています。どの例においても、DB2 Information Integrator とそれに関連したテクノロジーが、ソリューションのソフトウェアおよびフレームワークとして活躍します。

表 1. 典型的な顧客問題

典型的な顧客問題	解決策	結果または付加価値	技術上の要件
電話販売の効率の向上	営業でのやり取りを記録して、テキスト・フォーマットで保管する。データおよびテキストのマイニング技法を、販売とトランザクションのデータおよび営業活動データと結合する。	構造化データと非構造化データが未分離のままであったなら確かめられなかったであろうパターンを調べることで、営業活動とストラテジーが改善される。	構造化データと非構造化データの統合。統合後のデータに対して必要なマイニング技法。
別々の情報グループを統合して、業務の効率を改善する。これには、社内作成のアプリケーション、パッケージ化アプリケーション、または合併あるいは買収によって獲得したアプリケーションも関与することがあります。	社内全体で情報を統合および交換するために Extensible Markup Language (XML) を使用する。新しいアプリケーションで使用するために XML で企業モデルを作成する。	既存の情報から競合値が抽出される。業務の効率化によって収益が増大する。業務サイクルと意思決定のプロセスが短縮される。	アプリケーション・データなどの構造化および非構造化データの統合。アプリケーションおよびプロセスの統合 (メッセージング、ワークフロー)。テキスト・データのマイニング (カテゴリー化および検索)。
現在の顧客ベースでの収益を改善する。	顧客のアカウントと動作を分析する。レプリケーションを介してウェアハウス内の業務データを統合する。データに対して Intelligent Miner™ を使って情報と MQ メッセージングをマイニングして、業務ユーザーに情報を流す。	目的の提供商品を顧客プロファイルに合わせることができる。顧客あたりの収益率が向上する。	レプリケーション、関連情報のマイニング、信頼性の高い分散機構。

表 1. 典型的な顧客問題 (続き)

典型的な顧客問題	解決策	結果または付加価値	技術上の要件
Oracle の分散ソースと外部の非リレーショナル・ソースに保管されている化学および生物学的情報の統合ビューが学術ユーザーにとって必要。	複数の異種データ・ソースにまたがる DiscoveryLink 特殊データ・ソースのサポート。学術的マイニング・アルゴリズム。	情報へのリアルタイム・アクセス。	複雑な照会の最適化。関連情報のマイニング。
顧客の多様なニーズに対処するためにオーダー・プロセスが改善される。既存の操作可能システムとの統合。	新たにパーツのオーダー・アーキテクチャーを作成する。オーダーをデータベースに入れる。レプリケーションとフェデレーテッド・システムを使って、オーダーを旧システムに転送する。	オーダーの応答時間が改善される (請求書作成発行サイクルの改善と現金の流れの改善)。	レプリケーション、フェデレーテッド・システム、WebSphere®。
社内の複数の部門に対してより良い顧客サービス統合情報を提供して、シームレスな顧客関係情報を提供する。	さまざまな業務単位にまたがってクライアント情報を統合して、個別またはグループ別の統合顧客ビューを提供する。	顧客の電子メール (E メール) に対する応答において、顧客に対する勧告が個別設定される。社内全体を通して固定客のポイント追跡が提供される。	ウェアハウジング、フェデレーテッド・システム、レプリケーション、WebSphere。

関連概念:

- 16 ページの『DB2 Universal Database ファミリー — インフォメーション・インテグレーションの基礎』
- 6 ページの『DB2 Information Integrator — 統合のソリューション』
- 1 ページの『インフォメーション・インテグレーションの概要』
- 6 ページの『インフォメーション・インテグレーション・アーキテクチャーの計画』

インフォメーション・インテグレーションの基礎

リレーショナル、Extensible Markup Language (XML)、またはフラット・ファイルなどのさまざまなデータ・ソースをモデル化する機能がインフォメーション・インテグレーションの基礎になります。インフォメーション・インテグレーションのテクノロジーを使えば、そのようなモデル化データ・ソースで使えるトランスフォーメーション機能を手に入れることができます。その後、社内で有用と認められた、たとえばアプリケーションに組み込んだ SQL ステートメントを使用する方法を使うか、または Web サービス・アプリケーションを使って、そのようにマージしたビューを表すことができます。インフォメーション・インテグレーションのテクノロジーにおいては、次のような理由で、十分に配慮の行き届いたデータベース・アーキテクチャーが便利でしかも必要です。

- これまで企業アプリケーションで起きていた情報の急激な増加は、データベース管理システム (DBMS) によって管理できることが実証されています。データベース管理者は、ストレージ、検索、トランスフォーメーション、スケーラビリティー、信頼性、および可用性の諸問題を実感しています。
- データベース業界は、 e-business アプリケーションにおけるデータの多様性とアクセス・パターンへの適応を習得しつつあります。データベース業界内ではアプリケーションと機能において、組み込みのオブジェクト・リレーショナル・サポートと、Extensible Markup Language (XML) 機能を使用しています。そのような機能は、外部データ・ソースへのフェデレーテッド・アクセスをサポートします。
- データベース・テクノロジーへの投資は今後も増えつづけますが、それにはデータベース、それをサポートするツール、アプリケーション開発環境、およびデータベース技法に熟達した人員などが含まれます。

関連概念:

- 16 ページの『DB2 Universal Database ファミリー — インフォメーション・インテグレーションの基礎』
- 6 ページの『DB2 Information Integrator — 統合のソリューション』
- 3 ページの『DB2 Information Integrator が解決する問題』

DB2 Information Integrator — 統合のソリューション

企業向けのインフォメーション・インテグレーション・ソリューションは、IBM® DB2® Information Integrator です。 DB2 Information Integrator は、データとそのデータをサポートするプロセスの完全統合のためのフレームワークです。 DB2 Information Integrator テクノロジーが対応できる問題は多岐にわたります。企業にとって最適のソリューションは、 Information Integrator の個々のテクノロジーとその関連製品を組み合わせることと考えられます。複数の組み合わせのソリューションを手元に用意するために、 1 つのテクノロジーから別のテクノロジーに簡単に移動できることがさらに重要になります。 DB2 Information Integrator は、まとまってシームレスに稼働する各種テクノロジーとデータの完全な統合を実現します。

関連概念:

- 1 ページの『インフォメーション・インテグレーションの概要』
- 3 ページの『DB2 Information Integrator が解決する問題』

インフォメーション・インテグレーション・アーキテクチャーの計画

統合環境のアーキテクチャーを計画する際には、 Web サービス・ツール、データベース管理ツール、およびウェアハウス・ツールを使用することができます。統合の言語である XML と、データベース・テクノロジーの言語である SQL を使用できます。

Extensible Markup Language (XML) は Web アプリケーションで使用できるテクノロジーですが、これはメタデータと一緒にデータをパッケージ化します。 XML は情報を統合するストラテジーの軸となるエレメントです。 XML は企業間取引の通信を可能にする共通言語として機能します。これは、トランザクション中に何も消失しないように Web を通して簡単に半構造化データを送信する手段になります。その場合、エンドポイントのインフラストラクチャーにおける相違点がマスクされ

ます。 XML では、多様な装置ごとに情報が適宜変換されます。それは、文書の内容と文書の表示が分離されているからです。 XML はその内容または意思決定支援に応じて要領よく機能します。それは、データの記述と、それが他のデータとどのように関連付けられているかが記入されているからです。 XML を使えば、検索引き数のコンテキストが装備されているので、検索結果を簡潔化するのに役立ちます。

SQL は、リレーショナル・データベース管理システムへのアクセス用の ANSI (American National Standards Institute) 標準です。 SQL は、リレーショナル、フェデレーテッド、および非リレーショナルのデータへアクセスする手段になります。 Web サービスで SQL ベースの照会を使用すれば、 SQL ステートメントとストアード・プロシージャー呼び出しを DB2® Universal Database に送ることができます。その後 Web サービスから、いずれかのデフォルトのタグ付けを使用した結果が戻されます。 Web 上で SQL ベースの照会が便利なのは、エレメントとして列名を使った SQL データ・タイプの単純なマッピングのみが戻りのデータで使用される場合です。 XML のエレメントと属性に対するユーザー定義の SQL データ・マッピングが必要な場合は、 DB2 XML エクステンダーおよび SQL ベースの照会を使用します。 DB2 XML エクステンダーを使って XML 文書を表内の 1 つの列に保管すると、 SQL ベースの照会を使ってその文書を現状のまま文字ラージ・オブジェクト (CLOB) として検索することができます。また、SQL ベースの照会と一緒に DB2 XML エクステンダーを使って、その文書の一部を抽出するユーザー定義関数を呼び出すこともできます。

SQL ベースの照会を使って、DB2 UDB ストアード・プロシージャーを呼び出すことができます。ストアード・プロシージャーは、当然のこととして Web サービスに変換されますが、それは、ストアード・プロシージャーそのものが、プログラミング論理とデータベース・アクセスをカプセル化したものだからです。 Web サービスでストアード・プロシージャーを呼び出せば、入力パラメーターの動的な指定や結果の検索を行うことができます。

インフォメーション・インテグレーションのアーキテクチャーは 3 つの層からなります。

- データ層

  データ層は統合の基盤になります。この層は、さまざまなフォーマットのベース・ソースから、データの保管、検索、およびトランスフォーメーションを行う手段になります。この層はフェデレーテッド DBMS アーキテクチャーをベースにしています。データは、構造化リレーショナル表または半構造化 XML 文書として、あるいは非構造化フォーマットで保管されます。ハイブリッド XML およびリレーショナルの保管と検索のインフラストラクチャーによって、高いパフォーマンスとデータ耐久性が確保されます。このデータ層では、柔軟性に富んだラッパー・アーキテクチャーを使用するフェデレーテッド・データベース・テクノロジーが使用されて、外部のデータ・ソースとの統合が図られます。なおそのデータ・ソースは、従来のデータ・サーバー、社内アプリケーション、またはワークフローのいずれでもかまいません。

- サービス層

  インフォメーション・インテグレーションには、データへのアクセスとそのデータの使用方法の決定が関連します。この層は、データ・アクセス・サービスを社

内アプリケーションや業務プロセスに透過的に組み込むインフラストラクチャーを成します。それには、照会の処理、テキストの検索とマイニング、トランスフォーメーション、およびレプリケーションも関与します。

- アプリケーション層

  インフォメーション・インテグレーションには、データを使用できるプログラミング・インターフェースも関連します。アプリケーション層は、標準ベースのプログラミング・モデルと、他の層に対する照会言語を備えています。 Web サービスまたは従来のアプリケーション・プログラミング・インターフェースをこのインターフェースのベースとすることができます。アプリケーション層では標準の照会言語が拡張されます。

各層は別々のエンティティーですが、統合ストラテジーをより完全なものにするには、他の層に従属するとともにそれらと連携して稼働する必要があります。 8 ページの図 1 は、データ層、サービス層、およびアプリケーション層を示しています。

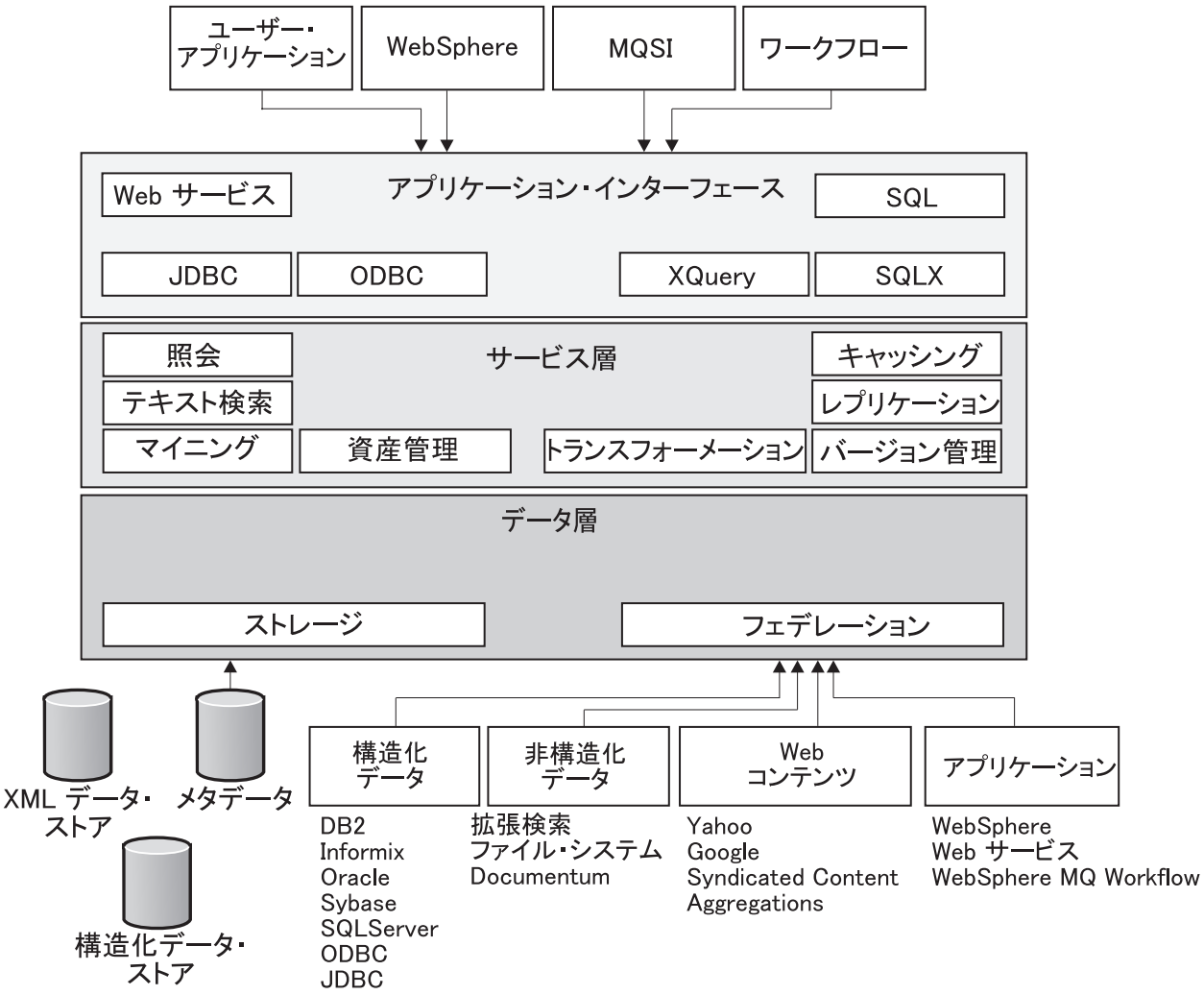


図 1. 統合層

関連概念:

- 6 ページの『DB2 Information Integrator — 統合のソリューション』

- 3 ページの『DB2 Information Integrator が解決する問題』
- 2 ページの『企業にとってインフォメーション・インテグレーションが重要な理由』

DB2 Information Integrator — リレーショナル・フェデレーテッド・テクノロジー

フェデレーテッド・データベース管理システムは、異種データへのアクセスやその取り扱いにおいて活躍します。フェデレーテッド・システムでは、さまざまなフォーマットで保管または生成された分散データの単一サイトのイメージを利用することができます。フェデレーテッド・システムには、そのデータにアクセスするための共通したインターフェースが用意されています。 IBM® DB2® Information Integrator のフェデレーテッド・データベース管理システムは、拡張カタログを特徴とします。このカタログは、リモート・データに関する統計を保守し、その統計を使ってデータ・アクセスをグローバルに最適化することができます。フェデレーテッド・システムを使えば、リモート・データ・ソースに関する統計を収集して、DB2 Universal Database™ に備えられた照会オプティマイザー・テクノロジーにおいてその情報を使用することができます。それによって、データへのアクセスを高速化することができます。フェデレーテッド・システムは、DB2 Universal Database の照会再書き込み機能を使って、パフォーマンスの低い照会をより高速なフォームに書き直します。

フェデレーテッド・データベース・エンジンでは、ラッパーという名前のソフトウェア・コンポーネントを通してデータ・ソースにアクセスします。どのラッパーにも、データ・ソースのデータ・タイプと DB2 Universal Database データ・タイプの間のデフォルト・マッピングなどの、データ・ソースに関する情報が入っています。ラッパーをインプリメントするためにサーバーは、ライブラリーに保管されているラッパー・モジュールという名前のルーチンを使用します。このルーチンを使うと、データ・ソースへの接続やそこでのデータの反復検索といった操作をサーバーから実行することができます。データの照会、検索、および操作のためにラッパーを使用するには、ラッパーをインストールしておく必要があります。インストール後に、各ラッパーを登録してフェデレーテッド・システムに追加する必要があります。フェデレーテッド・システム・オブジェクトの定義などの、フェデレーテッド・システムのテクノロジーに関する詳細は、「フェデレーテッド・システム・ガイド」を参照してください。

DB2 Information Integrator のフェデレーテッド・テクノロジーには、複数のデータ・ソースに対する仮想データベースが備えられています。このデータ・ソースは、以下を実現します。

- それぞれ異なるハードウェアとそれぞれ異なるオペレーティング・システム・プラットフォーム上で稼働する。
- さまざまなベンダーから提供される。
- それぞれ異なるアプリケーション・プログラミング・インターフェースとそれぞれ異なる SQL ダイアレクトを使用する。

プログラマーはフェデレーテッド・システム・データベースを使ってデータベース管理システムをカスタマイズして、任意のリレーショナルまたは非リレーショナルのどちらのデータ・ソースにでもアクセスすることができます。

リレーショナル・フェデレーテッド・ラッパーは、 10 ページの 一覧で示してあるような他のリレーショナル・データベース管理システム (RDBMS) のデータを照会することができます。リレーショナルおよび非リレーショナルのラッパーは、ソースを DB2 UDB にマッピングすることで他のデータベース・システムに透過的にアクセスする手段になります。そのようなデータ・ソースに単一の照会でアクセスして、フェデレーテッド・システムおよび DB2 UDB に用意されているパフォーマンス技法や照会の書き直し機能を利用することができます。

フェデレーテッド・リレーショナル・テクノロジーに備えられたリレーショナル・ラッパーの中には、 IBM 以外のリレーショナル・データベースと Informix® データベース用のものがあります。次のようなデータ・ソースのいずれかに保管されたデータにアクセスしたい場合は、リレーショナル・ラッパーが必要です。

- Oracle
- Sybase
- Microsoft® SQL Server
- IBM DB2 製品ファミリー
- Teradata
- Open Database Connectivity (ODBC) ソース

関連概念:

- 「*フェデレーテッド・システム・ガイド*」の『照会処理のチューニング』

関連タスク:

- 「*フェデレーテッド・システム・ガイド*」の『グローバルな最適化』

DB2 Information Integrator — 非リレーショナル・フェデレーテッド・テクノロジー

非リレーショナル・フェデレーテッド・テクノロジーは、非リレーショナル・データへのアクセス手段となるラッパーで構成されます。このようなラッパーを使えば、表構造化ファイル、Excel ファイル、Extensible Markup Language (XML) 文書、 BLAST 検索アルゴリズム、Documentum データ、Entrez ソース、HMMER ソース、BioRS、Extended Search ソース、ビジネス・アプリケーション、および Web サービス記述言語ファイルで記述される Web サービス・プロバイダーにアクセスすることができます。 Extended Search ソースを通して、広範囲にわたる非構造化データ・ソースへのアクセスが可能になります。そのようなソースには、Domino™、Microsoft® Exchange、Microsoft Index Server、および Lightweight Directory Access Protocol (LDAP) ディレクトリーなどがまず挙げられます。フェデレーテッド・システムの詳細は、「*フェデレーテッド・システム・ガイド*」を参照してください。ラッパーについてと、その作成方法についての詳細は、「*DB2 Information Integrator ラッパー開発者向けガイド*」を参照してください。

一連の非リレーショナル・ラッパーには、次のような複数のインストール可能コンポーネントが収まっています。

学術的データ・ソース

これには、非構造化データが入っています。そのようなデータには、生命科学の分野で開発されたジェノミックス、プロテオミックス、バイオインフォーマティックス、およびケムインフォーマティックスに関する情報などが含

まれます。このようなラッパーによって、フェデレーテッド・システムが、分散ソースにある遺伝学、化学、生物学、およびその他の研究データを統合できるようになります。たとえば、1 つの SQL ステートメントを使って、スイスのデータベースにあるたんぱく質のシーケンス・データ、日本のデータベースにある化学構造データ、およびご自分のローカル・エリア・ネットワーク上の表構造のフラット・ファイルに保管してあるスペクトル解析データを統合することができます。そのデータは、1 つの仮想データベース上にあるものとして表示されます。

構造化ファイル・データ・ソース
これには、定義済みの反復可能な構造をもったファイルに保管されているデータが入っています。たとえば、Excel スプレッドシートやフラット・ファイル (このファイルのどのレコードも、それぞれが区切り文字で区切られた同一数のフィールドを保有します) などがあります。

アプリケーション・データ・ソース
アプリケーション・ラッパーは、アプリケーションを使って基盤のデータにアクセスします。ロー・データは、いくつかの標準および非標準のフォーマットになっていることがあります。

Web サービス・プロバイダー・ソース
Web サービス・ラッパーの目的は、DB2® UDB および SQL ユーザーが、Web サービス記述言語 (WSDL) ファイルで記述された Web サービス・プロバイダーへアクセスできるようにすることです。 Web サービスは、Web サービス・プロバイダーとコンシューマーとの間で SOAP メッセージをやり取りすることで呼び出されます。 Web サービス・ラッパーは、SOAP ユーザー定義関数が Web サービスを使用する場合と同様のやり方で、 Web サービスを使用します。フェデレーテッド・ニックネームおよびビューを指定した SQL ステートメントを発行して Web サービスにアクセスするか、一連のユーザー定義関数を発行して Web サービスにアクセスできます。 Web サービスは、WSDL を指定することで発見でき、その後、 WSDL の情報に基づいて必要なニックネームを作成できます。 Web サービスにアクセスするニックネームを照会して、情報を変更したりアクセスしたりします。

非リレーショナル・ラッパーを必要とするデータ・ソースの例には、次のようなデータ・ソースがあります。このリストは、例に過ぎません。完全なラッパー情報については、「*DB2 Information Integrator データ・ソース構成ガイド*」を参照してください。

- BLAST
- Excel
- 表構造ファイル
- Documentum
- HMMER
- Entrez
- Extended Search
- BioRS
- Web サービス

- WebSphere® Business Integration を使用してアクセスできるビジネス・アプリケーション (SAP、PeopleSoft、および Siebel を含む)

関連概念:

- 「フェデレーテッド・システム・ガイド」の『非リレーショナル・データ・ソースのデータ・タイプ・マッピング』
- 164 ページの『Web サービスのコンシューマー機能』
- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『Web サービス・ラッパーおよび Web サービス記述言語文書』

WebSphere Portal の例と DB2 Information Integrator

WebSphere® Portal 環境では、WebSphere Application Server とともに、ランタイム環境として、DB2® Information Integrator が使用できます。 DB2 Information Integrator を使用すれば、Web 対応アプリケーションの開発スキル向上のために大きな投資をすることなく、ユーザーの要望に応えることができます。 DB2 Information Integrator は、WebSphere Portal、WebSphere Application Server とともに機能するもので、これにより、Web アプリケーション、あるいはコンポーネントが、単一の API (SQL) によって多種のデータ・ソースにアクセスする、ということが可能になります。つまり、多種データへのアクセス、多種データのマージのために通常必要になる複雑なアプリケーション・ロジックの多くが不要になるということです。 DB2 Information Integrator は、ソースの全データ内容を表示したり、(若干の構成を加えて) データ・ソースへの書き込みを行ったりするためにも使用できます。 DB2 の最適化機能により、リモート・データ・ソースのデータも効率的に抽出することが可能です。

実際のビジネスの場で DB2 Information Integrator を WebSphere Portal とともに使用して開発の効率化を図る一例として、ポータルによって様々な情報やサービスを 1 つのビューで社員に提供するというものが挙げられます。ここでは、架空の保険会社、Cotton-wood Insurance Corporation を例にとって説明を進めていきましょう。Cotton-wood Insurance Corporation では、請求関連の多岐にわたる情報を 1 つのビューでカスタマー・サポート担当者に提供したいと考えていました。カスタマー・サポート担当者が必要とするデータには、以下のような様々な種類にまたがり、リレーショナル・データもあれば、非リレーショナル・データもある、という状態でした。

- DB2 UDB for z/OS™ データベースに収められた顧客情報
- Lotus® Domino™ に収められた査定人評価フォーム、業務日報
- XML ファイルのかたちで保管された警察の調書
- WebSphere MQ キューに書き込まれた新規の保険金請求 (これによって新規請求の処理が開始される)

Cotton-wood の開発者は、請求関連の情報をすべて単一の Web ベース・ビューで提供できる簡単なポータルを作成し、カスタマー・サポート担当者の顧客サービスを支援することにしました。このポータルでは、ポータルに情報やサービスを提供するポートレットがいくつか使用されます。社内のあらゆるデータ・ソースのデータに単一のビューでアクセスするために、Cotton-wood の開発者が利用したのが、DB2 Information Integrator です。開発者は、標準の WebSphere ポートレットと DB2 Information Integrator を使用して、多種データ・ソースへのアクセスのために必要に

なるコードの量を最小限に抑えました。開発者は、多種のデータ・ソースのフォーマットそれぞれに対応するネイティブ API を使用したコードを書くのではなく、DB2 Information Integrator にアクセスする SQL コードを作成して、データへの照会を発行したのです。複雑なアクセスや結合などのためのロジックの取り扱いは、DB2 Information Integrator に任せることができました。

Cotton-wood の開発者が作成したポートレットのうちの 1 つは、カスタマー・サポート担当者に、公開されている全請求を単一のビューで提供するものでした。このポートレットでは、DB2 UDB に格納された顧客情報、XML ファイルのかたちで保管された警察調書、Lotus Domino サーバーに格納された査定人評価フォームなどへのアクセス、またこうした情報のマージが必要になります。 DB2 Information Integrator がなければ、個々のデータ・ソースに個別にアクセスしなければならず、すべてのデータをマージして 1 つにした上でユーザーに提示する、ということができるポートレットのコードを開発者が自ら書かなければならなくなります。しかし、DB2 Information Integrator を使ったため、開発チームはポートレットを 1 つ開発するだけで済みました。1 つの SQL ステートメントだけで、DB2 UDB のデータにも、XML のデータにも、Lotus Domino のデータにも対応できたのです。

ポートレットが動作すると、SQL 照会がポータル・アプリケーションから DB2 Information Integrator フェデレーテッド・サーバーに送られます。必要なデータへのアクセスは、すべてフェデレーテッド・サーバーが行います。これによって、多種のデータ・ソースが 1 つのリソースであるかのように見えるのです。通常であれば書かなければならないはずの、複数の接続、照会、結合ロジックなどを管理するためのコードは書かなくて済みます。フェデレーテッド・サーバーは、様々なソースからデータを収集した後、クライアントに単一の結果セットを戻します。これにより、カスタマー・サポート担当者に、必要な請求情報 (DB2 UDB、XML、Lotus Domino などのデータ・ソースから統合されたもの) がすべて一度に提供されることになります。

Cotton-wood の例を見れば、DB2 Information Integrator を利用することで、(リレーショナル・データ、非リレーショナル・データが混在する) 多種のデータ・ソースへのアクセス、あるいは多種のデータのマージのために開発者が自ら書かねばならないコードの量、必要になるスキルのレベル、アプリケーション・ロジックの開発に要する時間などを大きく低減できることがわかるはずです。 DB2 Information Integrator を利用すれば、開発チームは単一の API だけで開発ができ、DB2 UDB のデータ・マージや最適化の機能も利用できるため、開発者は、データへのアクセスやデータのマージにあまり神経を使う必要がなく、アプリケーションの使いやすさの部分に注力できます。

WebSphere Portal、DB2 Information Integrator に関しては、 WebSphere Portal および DB2 Information Integrator のためのサンプル・コードも参照してみてください。

関連概念:

- 183 ページの『フェデレーテッド・システムの利点』
- 10 ページの『DB2 Information Integrator — 非リレーショナル・フェデレーテッド・テクノロジー』

関連タスク:

- 203 ページの『フェデレーテッド・アプリケーションのデプロイ』

本書全体を通して使用されるシナリオの紹介

この項では、本書全体を通して示されているシナリオを紹介します。これらは顧客における実例ですが、複数の企業を組み合わせています。企業名は架空のものです。

Cottonwood Distributors, Inc. — ウェアハウスの例

Cottonwood Distributors, Incorporated (CDI) は現在活動中の堅実な流通業者です。CDI は商品パーツのブローカーとして活躍しています。同社は長年にわたってこの業務に携わっていますが、従来からの DB2® Universal Database の固定顧客です。CDI はリレーショナル・データベース (DB2) を使って情報を保管しています。この情報は、数十万種のパーツ、それらのパーツの数千社の供給業者、およびそれらのパーツをオーダーする見込みの顧客で構成されます。同社には、クライアント・アプリケーションを通してオーダーを入力する営業担当員がいます。営業担当員は電話でオーダーを受け、別のパーツを求めている顧客には指定価格を提示します。顧客販売のフォローアップは、社外のベンダー (この会社は人的資源の処理も扱っています) に外注されています。 CDI は Extensible Markup Language (XML) を使って通信とレポートを行っています。そのシステムは、きわめて高いスキルをもったデータベース管理者の管理下にあり、その管理者によってシステムは綿密に調整されて長年に渡って安定した状態に保たれています。

CDI は、データのサイズとロケーションに関する問題をかかえています。業界内の競合企業である MyDogwood, Incorporated と MyOak, Incorporated の 2 社の買収に際して、それはさらに大きな問題になります。この新しい 2 社は似通った業務運営を行っており、親会社と同じ量のデータを保有していますが、そのデータは別々のデータベースに置かれています。 MyDogwood, Incorporated は Oracle データベースに、 MyOak, Incorporated は Informix® データベースにそれぞれデータを置いています。

合併の結果、CDI はさまざまなフォーマットの異種データ・ソースをもつことになります。それらのデータ・ソースは、すでにロードおよび構成済みで順調に稼働しているシステム上に存在し、しかもそのシステムは多数の非トラステッド・ユーザーをかかえています。そのプロセスでは、数百万種のパーツと数千社の供給業者を取り扱う必要があります。 CDI が競争力を保ちつづけるには、電話活用の営業担当員を、価格提示を求めてからオンラインでオーダーを入れる数千社の Web ユーザーに置き換える必要があります。しかも、供給業者は、パーツの見積価格をインターネットで送信したいと考えています。そのため CDI は、Web ベースの仲介取引を導入します。顧客は、インターネットを介してパーツ情報にアクセスして、オーダーを出します。供給業者はインターネットにアクセスして、パーツの見積価格を提示します。販売のフォローアップと人的資源の処理は、引き続き外注されます。今後 CDI は、複数のデータベースにまたがって、データへのアクセスのしやすさ、データの新鮮さ、およびリアルタイムの更新といったさらに別の管理対策に取り組む必要があります。

関連資料:

- 243 ページの『付録 A. Cottonwood Distributors, Inc. および YBar, Inc. シナリオのスクリプト例』

データの発見 — 従業員のスキルのシナリオ

YBar, Incorporated という会社は、人的資源を専門に取り扱っていて、スキルと企業のニーズを特定し、履歴書を調べて新規採用者を配属します。同社には、人員とそれぞれの職能スキルに関する情報を収めた社員データベースがあります。同社は、データベースの外側に存在する関連付けのないアプリケーションによって管理されるフラット・ファイルを使って作業しなければなりません。このアプリケーションは、ジョブ・プロファイルを管理します。さらに同社は、社外のソリューション提供会社が保有しているアプリケーションを利用する必要もあります。

同社のかかえる問題は 2 つに大別されます。

1. 欠員のある職種に配属するのに最適の候補者を特定する必要がある。

   これはたいていの場合、社内の欠員職種であり、個人の履歴書に示されているスキルを基に検索を行います。同社は、候補者の Extensible Markup Language (XML) フォーマットの履歴書を収集する Web アプリケーションを持っています。人的資源システム (HR) によって、全社員の現在の職種プロファイルの追跡記録がとられています。
2. 同社は、カスタマイズされたクラスに適切な人員が参加できるようにするために、社員リスト (現在の職種記述も含む) を外部の社員教育機関に送る必要がある。

   同社は WebSphere® MQ を使って、アプリケーションのメッセージを送信します。送信後にフェデレーテッド・システムを使って、社員データベースと職種データベースを結合してから、 WebSphere MQ を通してそのリストをパブリッシュします。

YBar, Incorporated は、社員に関する情報の入った *Employee* という表を持っています。また、職種を記述した *Job* という表も持っています。

表 2. YBar, Inc. の社員および職種の表と列

社員	JOB (職種)
Emp_ID	Job_ID
Lastname (姓)	Job_Description (職種の記述)
Firstname (名)	Title (肩書き)
Dept_ID	Responsibilities (職責)
Current_Job_ID	

関連概念:

- 192 ページの『従業員のデータベースのシナリオ - ソリューションの設計』

関連資料:

- 243 ページの『付録 A. Cottonwood Distributors, Inc. および YBar, Inc. シナリオのスクリプト例』

インフォメーション・インテグレーションのコンポーネント

インフォメーション・インテグレーションでは、実証済みのデータ管理テクノロジー (DB2 Universal Database ファミリー) を IBM DB2 Information Integrator の基盤

として使用します。 DB2 Universal Database は、単一システム内とクラスター・システム上での並行処理を使用して、数千件の並行ユーザーとテラバイト単位のデータをサポートします。また、オブジェクト・リレーショナルを土台として、独自に調整された機能を使って新規のデータ・タイプを追加するための拡張可能なフレームワークを提供します。インフォメーション・インテグレーションのインフラストラクチャーは、 IBM WebSphere に備えられているようなメッセージングおよびワークフロー機能を使用するクライアント・アプリケーション・プログラミング・インターフェース (API) として機能します。新規情報の着信などのデータベース・イベントによって、新規メッセージをキューに入れるといった、通知を透過的に作成することができます。 WebSphere Studio は、 Java ベースのデータベースおよびフェデレーテッド・データベース・アプリケーションのオープンな統合開発環境として機能します。また、以下のタスクのような各種機能を実行するグラフィカル機能も備えています。

- SQL 要求の作成
- 結合パスの理解
- ストアード・プロシージャーの開発
- ユーザー定義関数を使ったデータベースの拡張
- Web サービスへのデータベース要求の変更
- Extensible Markup Language (XML) スキーマおよび文書タイプ定義 (DTD) の開発

DB2 Universal Database ファミリー — インフォメーション・インテグレーションの基礎

DB2® Universal Database は、IBM® DB2 Information Integrator テクノロジーの基礎です。 DB2 Universal Database™ は、DB2 Universal Database™、Oracle、Sybase、その他のデータベースのいずれに情報が保管されていても、多種多様な情報を管理できます。

DB2 エクステンダー (DB2 Extenders™) は、イメージ、ビデオ、オーディオまたは音声録音、 Extensible Markup Language (XML) 文書、複雑なテキスト文書、空間オブジェクトなどを管理することができます。 DB2 Universal Database Data Links Manager は、外部ファイル・システム内のデータを管理することができます。 DB2 Universal Database は、参照保全、アクセス・コントロール、整合性、およびリカバリーを処理することができます。 DB2 XML エクステンダー、Net Search Extender、および Spatial Extender は、さまざまなデータ・オブジェクトの照会、アクセス、更新、および管理を行うためのデータ・タイプ別の拡張機能を備えています。たとえば、DB2 エクステンダーを使って、 XML 文書のアクセス、イメージ形状か色による照会、または特定のロケーション別の照会を行うことができます。

DB2 Universal Database とそれに関連したコンポーネントは、以下のタイプのデータを保管および検索する機能を備えています。

- 構造化リレーショナル表
- 半構造化 XML 文書
- バイト・ストリームやスキャン・イメージなどの非構造化コンテンツ

DB2 Universal Database の詳細は、http://www.ibm.com/software/data/db2/ を参照してください。

DB2 XML エクステンダー

DB2 XML エクステンダーは、XML 文書とその文書タイプ定義 (DTD) のリポジトリーとして機能します。また、データ保全性、セキュリティー、リカバリーの容易性、および管理の容易性などのデータ管理機能も備えています。文書全体を XML のユーザー定義列として保管できますが、文書を複数の表と列に分解することもできます。 XML のエレメントと属性では、確実に高速検索を行うために索引を使用することができます。 SQL 照会内で、文書全体を検索するか、または XML のエレメントと属性を動的に抽出することができます。さらに XML エクステンダーには、既存のデータから XML 文書を作成するためのストアード・プロシージャーも備えられています。

DB2 XML エクステンダーでは、 DB2 XML アプリケーションから DB2 Universal Database メッセージ・キューイング関数に直接アクセスするための関数とストアード・プロシージャーを使用できます。 WebSphere® MQ ツールを使って、メッセージ照会関数およびストアード・プロシージャーを開発することができます。 DB2 XML エクステンダーには、いくつかのメッセージ照会関数が付属しています。そのような関数を使って、アプリケーションで以下を行うことができます。

- SQL ステートメントを使って、Application Messaging Interface に定義されているとおりのサービス・ポイント (キュー) 内の XML メッセージとして文書の送信、読み取り、または受信を行います。
- 表から XML メッセージを作成してメッセージ・キューに直接送信するか、またはキュー内の XML メッセージをリレーショナル表に分解します。

Net Search Extender

DB2 Information Integrator のフェデレーテッド・サポートを組み込まれた IBM Net Search Extender を使って、 DB2 Universal Database および Informix® IDS データベースに保管されているテキスト・データの索引付けや検索を行うことができます。また、Oracle、Sybase、および Microsoft® SQL Server などのフェデレーテッド・ソース内で IBM Net Search Extender を使用することもできます。データベース・マネージャー・オプティマイザーのインテリジェント・ストラテジーとの統合によって、ハイパフォーマンスが得られ、SQL の全選択内でシームレスに稼働する全テキスト検索が可能となります。

Spatial Extender

DB2 Spatial Extender を使って、地理情報に定義可能なオブジェクト (地球上のロケーションに対して定義可能なオブジェクトや、地球上のある一地域内にあるオブジェクトなど) のファクトと図を得ることができます。そのようなファクトと図が空間情報であり、そのようなオブジェクトが地理フィーチャーです。たとえば、DB2 Spatial Extender を使って、いずれかの居住地域がごみ処理の候補地と重複しているかどうかを確かめることができます。つまり居住地域と候補地はフィーチャーです。重複が存在するかどうかに関する確認内容は、空間情報の一例です。重複が存在する場合、その範囲も空間情報の一例になります。

DB2 Spatial Extender が空間情報を作成するには、フィーチャーのロケーションを定義しているデータを処理する必要があります。空間情報データと呼ばれるそのデータは、マップまたはそれに類似した射影上のロケーションを参照する座標で構成されます。たとえば 2 つのフィーチャーどうしが重なり合うかどうかを判別するに

は、一方のフィーチャーの座標がもう一方のものの座標と重なるかどうかを DB2 Spatial Extender で判別する必要があります。

DB2 Spatial Extender を使えば、社内のデータベースと社外のデータ・ソースとで空間情報データをやりとりすることができます。詳しく言うと、データ交換ファイルと呼ばれるファイルで空間情報データを社内のデータベースに転送することで社外のデータ・ソースから空間情報データをインポートすることができます。また、社内のデータベースから空間情報データをデータ交換ファイルにエクスポートすれば、外部ソースがそれを取得することができます。

DB2 Warehouse Manager

データウェアハウスは、さまざまなデータ・ソースから得たデータの集まりです。エンド・ユーザーは、業務の観点から理解および使用可能なやり方でデータを取得します。そのデータは、業務または組織構造に関してグループ分けすることができますが、それには、異種環境から取り出したデータを構築、管理、および分析するウェアハウス・ツールを使用します。

DB2 Warehouse Manager には、データを移動およびトランスフォームするための、SQL ベースの抽出、トランスフォーム、およびロード機能が配備されています。さらに、DB2 Warehouse Manager には、メタデータ管理ソリューションであるインフォメーション・カタログ・センターも組み込まれています。またインフォメーション・カタログ・マネージャーには、サード・パーティーの独立ソフトウェア販売会社を対象とした統合点が設けられていて、それによってメタデータとジョブ・スケジューリングの双方向交換を行います。 DB2 Warehouse Manager には、ジョブ・スケジューリングの抽出、トランスフォーム、およびロードの分散システムも組み込まれています。 DB2 Warehouse Manager エージェントは、中央サーバーの追加コストをかけずにソース・システムとターゲット・システムでデータを直接やりとりすることをサポートします。 DB2 Warehouse Manager は、IBM の統合データ・レプリケーション機能を含め、フル・リフレッシュ、インクリメンタル更新、およびデータ移動オプションをサポートします。

レプリケーション

レプリケーション・テクノロジーを利用すれば、中央データベースと各地のトランザクション・データベースとでデータを互いに複製しあうことができます。レプリケーションによって各地のデータベースで業務データを利用できるようになるので、トランザクション処理が高速化されます。レプリケーション環境では、ソース・データベースからデータの変更内容をキャプチャーして、任意のターゲット・データベースにその内容を伝搬することができ、アプリケーションを変更する必要もありません。レプリケーションは、複数表の結合やストアード・プロシージャーなどの、標準 SQL を使用するデータ・トランスフォーメーションの手段になります。レプリケーションでは、データウェアハウスおよびデータマートにデータを追加するための組み込みトランスフォーメーションを使用する、特定時点のレプリケーションとほぼリアルタイムのレプリケーションがサポートされます。

レプリケーションは、 DB2 Universal Database トランザクションを再構成して、それらのトランザクションを XML のメッセージとして発行する機能もサポートしています。レプリケーションはトランザクションを WebSphere MQ メッセージ・キューへのメッセージとして送信することもできます。次いで、このようなメッセージ

はメッセージ・サブスクライブ・アプリケーションまたは Q アプライ・プログラムにより使用できます。イベント発行は Q キャプチャー・プログラム (Q レプリケーションの別のコンポーネント) によりデータをキャプチャーします。イベント発行により、DB2 Universal Database 表からのコミット済みのトランザクション・データ、または行レベル・データを WebSphere MQ メッセージ・キューのメッセージとして発行することができます。メッセージはユーザー・アプリケーションで直接に読み取って解釈することもできますし、最初に WebSphere Business Integration Message Broker または DB2 MQ リスナー・デーモンなどのメッセージ・ブローカーで解釈することもできます。

関連概念:

- 「*データウェアハウス・センター 管理ガイド*」の『データウェアハウジングが提供するソリューション』
- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『XML Extender の概要』
- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『DB2 における XML データの処理方法』
- 「*データウェアハウス・センター 管理ガイド*」の『データウェアハウス・センターでのレプリケーション』
- 「*IBM DB2 Information Integrator レプリケーションとイベント・パブリッシング ガイドおよびリファレンス*」の『Q レプリケーションの紹介 — 概要』
- 「*IBM DB2 Information Integrator レプリケーションとイベント・パブリッシング ガイドおよびリファレンス*」の『イベント発行の紹介 — 概要』
- 「*IBM DB2 Spatial Extender and Geodetic Extender ユーザーズ・ガイドおよびリファレンス*」の『DB2 Spatial Extender の目的』
- 「*IBM DB2 Spatial Extender and Geodetic Extender ユーザーズ・ガイドおよびリファレンス*」の『DB2 Spatial Extender の使用法』
- 「*IBM DB2 Spatial Extender and Geodetic Extender ユーザーズ・ガイドおよびリファレンス*」の『地形、地理情報、地理情報データおよび図形の組み合わせ方』

関連タスク:

- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『XML 文書を検索するための方法』
- 「*IBM DB2 Information Integrator SQL レプリケーション・ガイドおよびリファレンス*」の『SQL レプリケーションの計画』

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『全選択』

インフォメーション・インテグレーションでの Web サービスの諸機能

次のような点で、Web サービスは統合のための鍵となるイノベーションです。

- Web サービスはインターオペラビリティーを促進します。つまり、Web サービスでは、サービス・プロバイダーとサービス要求元がやりとりする対話は、プラットフォームや言語にまったくとらわれないように設計されています。

- Web サービスはタイミングの良い統合を可能にします。つまり、サービス要求元がサービス・ブローカーを使ってサービス・プロバイダーを探すときは、ディスカバリーが動的に行われます。
- Web サービスは、カプセル化を通して複雑さを軽減します。つまりサービス要求元とサービス・プロバイダーは、互いに対話するのに必要なインターフェースにのみ専心することができます。その結果、サービス・プロバイダーがサービスをどのようにインプリメントしているかをサービス要求元が知ることはなく、またサービス要求元がサービスをどのように使用するかをサービス・プロバイダーが知ることもありません。 Web サービスはそのような詳細をリクエスターおよびプロバイダー内部にカプセル化します。
- Web サービスでは、これまでのアプリケーションを Web サービスとしてキャストすることができます。つまり、社内に従来からあったアプリケーションとパッケージを斬新な方法で使用できるということです。さらに、これまでのアプリケーション (セキュリティー、ディレクトリー・サービス、およびトランザクションなど) に関連したインフラストラクチャーを、やはり一連のサービスとしてラップ することができます。

Web サービスは、Web サービス記述言語 (WSDL) を利用した、プロバイダーとアプリケーション・リソースのコンシューマー間の簡便なインターフェースです。

Web サービス・プロバイダー

Web サービス・クライアント・アプリケーションは、Web サービス記述言語 (WSDL) インターフェースを利用することで、DB2® Universal Database へのアクセスができます。 DB2 UDB データへの WSDL インターフェースは、Web Services Object Runtime Framework (WORF) (Document Access Definition Extension (DADX) ファイルとも言う) を使用することによって作成できます。 DADX ファイルによって DB2 UDB データへのアクセスのための操作を定義すれば、その DADX ファイルとそのランタイム環境 (Apache SOAP バージョン 2.3 あるいは Apache Axis バージョン 1.2) を、サポートされている Java™ Web アプリケーション・サーバー環境 (Apache Jakarta Tomcat あるいは IBM® WebSphere® アプリケーション・サーバー) にデプロイできます。 DB2 Web サービスのテストとデプロイが済んだら、あらゆる Web サービス・クライアントがそのサービスを利用できます。

Web サービス・コンシューマー - ユーザー定義関数

DB2 Universal Database™ がコンシューマーになった場合は、データベース内での最適化を Web サービスに活かすことができます。 SQL ステートメントを使用することにより、Web サービス・データを利用および統合できます。 SQL を使用して Web サービス・データにアクセスすることによって、クライアント・アプリケーションに戻る前のデータを SQL ステートメントのコンテキスト内で操作できるので、アプリケーションのプログラミングにかかる労力を軽減できます。 WebSphere Studio バージョン 5 以降に用意されているツールを使用すれば、既存の WSDL インターフェースを DB2 UDB の表やスカラー関数に変換するということもできます。 SQL ステートメントの実行中には、Web サービス・プロバイダーとの接続を確立して、応答文書をリレーショナル表、あるいはスカラー値のかたちで受け取ることができます。

Web サービス・コンシューマー - Web サービス・ラッパー

フェデレーテッド・システムでは Web サービス・ラッパーが利用でき、これによってユーザーは、ニックネームやビューに対する SQL ステートメントによって Web サービスを呼び出して、これにアクセスできます。 Web サービスへの入力を指定し、SELECT ステートメントで Web サービスからの出力にアクセスする、 Web サービス・ラッパーとニックネームを作成できます。

図 2 は、Web サービス環境での DB2 Universal Database についてまとめたものです。

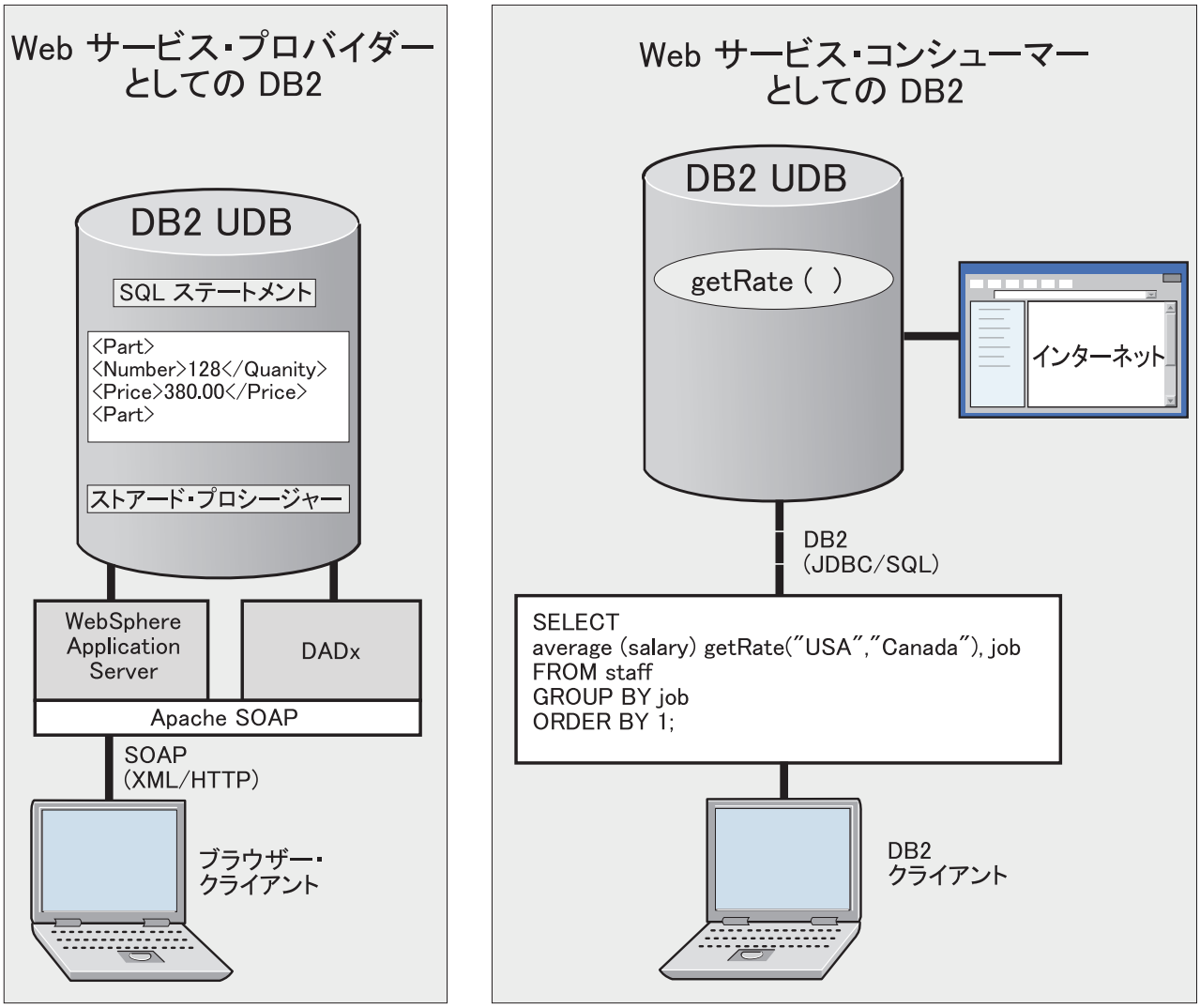


図 2. Web サービス・プロバイダーと SOAP ユーザー定義関数

DB2 Information Integrator では、次のような機能をはじめとして多数の Web サービス機能を使用します。

- ストアード・プロシージャー機能を Web サービスとして公開する機能。

  Web Services Object Runtime framework (WORF) と Document Access Definition Extension (DADX) を使えば、アプリケーション・サーバーはそのような Web サ

ービスをクライアントに提供することができます。アプリケーション・サーバーは WebSphere Application Server または Apache Tomcat となります。

- SOAP クライアントとなったり、 SOAP サーバーに対して Web サービスを要求したりするためのストアード・プロシージャーも含め、 DB2 UDB が実行するすべての SQL ステートメントに関する機能。それによって、SQL 値として、または他の SQL データと結合できる表として、データが Web サービスによって表示されます。

WebSphere Application Server は、ダイナミックな e-business のインフラストラクチャー・ソフトウェアです。 WebSphere Studio アプリケーション開発環境には、e-business を構築、デプロイ、および統合するのに必要なツールが用意されています。

WebSphere Application Server は J2EE 対応のアプリケーション・サーバーであり、オープンな分散コンピューティングのための環境を実現します。 WebSphere Application Server は、クライアントとリソース管理システム (データベースなど) を結び付ける仲介基盤として機能します。これを使ってクライアント (アプレットまたは C++ クライアントなど) は、データ・リソース (リレーショナル・データベースや WebSphere MQ など) および既存のアプリケーションと対話することができます。

関連概念:

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』
- 37 ページの『Web サービス・プロバイダーのフィーチャー』
- 164 ページの『Web サービスのコンシューマー機能』
- 40 ページの『Web サービス・プロセスの概要』
- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『Web サービス・ラッパーおよび Web サービス記述言語文書』

関連タスク:

- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『Web サービス・データ・ソースのニックネームの登録』

WebSphere MQ

IBM® WebSphere® MQ (旧称は IBM MQSeries®) は、動的統合に使用されます。これは、一貫性のある単純なプログラミング・インターフェースを使うか、またはさまざまなプラットフォーム上で侵害性のないアダプターを使って、すべての主なネットワーキング・システム上でアプリケーションを接続します。 WebSphere MQ は、システムが独自に稼働しながら、しかも確実に情報を配信する手段になります。これには、セキュリティーの強化とパフォーマンスの拡張のために Secure Sockets Layer (SSL) を使ったメッセージ暗号化機能が組み込まれています。WebSphere MQ は信頼性と堅実性に優れているため、今日のどの業界でも業務に不可欠でしかも貴重なソリューションにおいて使用することができます。

WebSphere MQ は、多数のアプリケーション・プログラミング・インターフェースのサポートをアプリケーションに提供します。

Message Queuing Interface

メッセージ・キューイングは、プログラムどうしが通信するための方法の 1 つです。メッセージ・キューイングを使うと、直接接続がなくてもプログラムはアプリケーション独自のデータの送受信を行うことができます。プログラムは、名前付きのキューへのメッセージの送信またはそのキューからのメッセージの検索によって通信を行います。プログラムは、名前付きのキューのロケーションを知っている必要はありません。可用性またはパフォーマンスを高めるために、プログラムを複製することができます。プログラムまたはキューを再配置することもできます。

Application Messaging Interface

MQSeries Application Messaging Interface は、 2 地点間のメッセージングのサポートを提供するとともにメッセージングをパブリッシュおよびサブスクライブする単純なアプリケーション・プログラミング・インターフェースです。 Application Messaging Interface は、アプリケーション・プログラムからデータ・リポジトリーへ機能を移動することでアプリケーション開発を単純化します。 Application Messaging Interface は、サービス、ポリシー、およびメッセージの 3 つの根幹パーツに分かれます。サービスは、どこにメッセージを送るかを定義し、ポリシーは、どのようにメッセージを送るかを定義し、メッセージは送られる実体です。サービスはローカルまたはリモートのキューをカプセル化します。ポリシーは、*優先順位* や*再試行* などのメッセージ・オプションをカプセル化します。メッセージ・パーツには、アプリケーション・メッセージ・データと、フォーマットまたは相関 ID などの属性を入れることができます。

Java™ Messaging Service

Java Messaging Services のアプリケーション・プログラミング・インターフェースは、アプリケーションがメッセージの作成、送信、受信、および読み取りを行うための手段になります。これによって、信頼性の高い非同期の通信が可能になります。

WebSphere のメッセージング機能を使えば、情報の受信、処理、および保管を簡単に行うことができます。その後 DB2® Warehouse Manager を使って、収集を調整してメッセージ・データを処理します。 IBM DB2 Information Integrator は、MQSeries Integrator によって処理される Extensible Markup Language (XML) 情報のソースと宛先を兼任します。 WebSphere のメッセージング機能は、アプリケーション開発とテストの単純化によって、また、どのプラットフォームでもアプリケーション・プログラミング・インターフェースを整合化することによって、分散アプリケーションのインプリメンテーションを高速化します。

関連概念:

- 6 ページの『DB2 Information Integrator — 統合のソリューション』
- 224 ページの『DB2 内で WebSphere MQ 機能を使用する方法』

関連タスク:

- 213 ページの『DB2 WebSphere MQ 機能のインストール』

アプリケーションの計画とテスト

使用予定のデータ・オブジェクトの種類 (データベース、表、ニックネームなど) と、そのオブジェクトを最も有効に使用する方法に関する計画を立てる必要があり

ます。次に、オブジェクトをデプロイする方法を決定する必要があります。たとえば、抽象モデルのビューを定義することができます。次に、ビューをネスティングして、より複雑なモデルをサポートします。ビューを使用して、さまざまなデータ・ソースを使用する複数の表をマスクすることができます。次に、Web アプリケーション内でビューに手を加えて、インターネットで使用できるようにすることができます。

インストールの計画

インフォメーション・インテグレーションのインフラストラクチャーのインストール計画では、現在の環境の理解と、どのコンポーネントが業務ソリューションを最も効果的に解決するかの知識も必要になってきます。必要となるツールと環境ではたいてい、 Windows® プラットフォーム上に少なくとも 256 MB のランダム・アクセス・メモリー (RAM) を必要とします。また、ディスク容量、通信セットアップ、前提条件、および保守の観点から、各コンポーネントごとのソフトウェアとハードウェア構成の要件を調べる必要もあります。フェデレーテッド・システムのインストールに関する考慮事項の詳細は、「*DB2 Information Integrator インストール・ガイド*」を参照してください。

関連概念:

- 6 ページの『DB2 Information Integrator — 統合のソリューション』
- 6 ページの『インフォメーション・インテグレーション・アーキテクチャーの計画』

関連資料:

- 「*IBM DB2 Information Integrator インストール・ガイド*」の『DB2 Information Integrator インストール・ワークシート』

アプリケーションおよび環境の構成

データベースのクライアントとサーバーを、互いに接続しあうように構成します。通信ネットワークとして Transmission Control Protocol/Internet Protocol (TCP/IP) を使用することができます。 Windows® 環境では、各システムの *services* ファイルに項目を追加して、サービス名とポート番号を指定することができます。

通常の構成では WebSphere® Message Queue サーバーと DB2® サーバーは同じマシン上に置かれています。 DB2 クライアントは、WebSphere サーブレット、Enterprise Java™ bean、または Web アプリケーションを使用するローカルまたはリモート・クライアントとすることができます。

フェデレーテッド・システム・テクノロジーでは、データ・ソースをホスティングするマシン上にソフトウェアをインストールする必要はありません。フェデレーテッド・データベースは、データ・ソースの通常クライアントを使って、クライアント・サーバー・アーキテクチャーを通してデータ・ソースと通信します。そのため、フェデレーテッド・データ・ソースは、ソースに対する他のアプリケーションとまったく同じように見えます。

フェデレーテッド・システムを作成するには、DB2 Universal Database™ エンジンをインストールしてから、フェデレーテッド機能を使用可能にします。次に、データ・ソースと対話するようにフェデレーテッド・システムを構成します。フェデレ

ーテッド・システムに新規のデータ・ソースを追加するためのステップはいくつかあります。まず、ソース用のラッパーをインストールしなければなりません。次に、そのラッパーがどこで見つかるかをフェデレーテッド・データベースに知らせる必要があります。それには、CREATE WRAPPER ステートメントを発行します。同一タイプのソースであれば、ソースが複数あってもラッパーは 1 つしか必要ありません。たとえば、フェデレーテッド・システムに 5 つの Oracle データベース・インスタンスが (おそらくは別々のマシン上に) あったとしても、 1 つの Oracle ラッパーしか必要ありません。 1 つの CREATE WRAPPER ステートメントを発行します。ただし、CREATE SERVER ステートメントを使って、各ソースを 1 つずつシステムに対して識別する必要があります。 5 つの Oracle データベース・インスタンスがある場合、5 つの CREATE SERVER ステートメントを発行します。

サーバーとフェデレーテッド・システムのプロパティー (SVCENAME、FEDERATED) をデータベース上で確実に設定します。サーバーからデータベースに接続します。アクセスする予定の各データ・ソースごとに、必要なラッパー・オブジェクト、サーバー・オブジェクト、およびユーザー・マッピングを作成します。

```
db2 update dbm cfg using svcename myID authentication server
db2 update dbm cfg using federated yes
db2 connect reset
db2stop
db2start
db2 connect to rdjdb user user1 using pass1word
db2 create wrapper net8 options (DB2_FENCED 'N')
db2 create server oracle8 type oracle version 8.1.5
    wrapper net8 authorization oracleuser1
    password oraclepwd
    options (node 'orafcle8.world', password 'Y', pushdown 'Y')
db2 create user mapping for user1
    server oracle8
    options (REMOTE_AUTHOID 'oracleuser1',
       REMOTE_PASSWORD 'oraclepwd')
...
```

図 3. データベース接続とラッパーのセットアップの例

関連概念:

- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・システム』
- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・システムと対話する方法』

関連タスク:

- 232 ページの『MQListener の構成と実行』

関連資料:

- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『フェデレーテッド・システム構成の計画のためのチェックリスト』

パフォーマンスと調整の計画 — フェデレーテッド・システム中のマテリアライズ照会表

マテリアライズ照会表は、照会の結果に基づいて定義される表です。マテリアライズ照会表のメカニズムを使って管理者は、基礎を成す一連の表またはニックネーム内でデータのマテリアライズ・ビューを定義することができます。フェデレーテッ

ド・システム・オブジェクトとその定義に関する詳細は、「フェデレーテッド・システム・ガイド」を参照してください。データベースのクラスによっては、データベースは基本表にアクセスしなくてもマテリアライズ照会表が照会に自動的に応答できるかどうかを確かめることができます。

マテリアライズ照会表を使うと、更新をデータベースに透過的にルーティングしながら、読み取り専用の照会をデータ・キャッシュに透過的にルーティングすることができます。変更後のデータは、レプリケーションによって、ユーザー指定のポリシーに従ってキャッシュに非同期で伝搬されます。フェデレーテッド・キャッシングとレプリケーションを使用すれば、ユーザーはさらに効率よく複雑な照会を要求できるようになります。

集約表を使用するマテリアライズ・ビューを使用すれば、 e-business 環境でのパフォーマンスを向上することができます。たとえば e-commerce において、マテリアライズ・ビューを使用して商品カタログを中間層のサーバーのキャッシュに入れることで、サマリー・データが介在しなくてもカタログのブラウズのパフォーマンスを向上することができます。 DB2® Universal Database では、リモート表の定義に使用されるニックネームでマテリアライズ・ビューを定義できるので、キャッシングがサポートされることになります。マテリアライズ・ビューにデータを追加するには、リモート表からデータをプルしてからローカルで保管しますが、これによって大幅なパフォーマンス上の利点が生じます。マテリアライズ・ビューに REFRESH DEFERRED パラメーターを組み込むと、パフォーマンスは最高になります。マテリアライズ照会表、つまりマテリアライズ・ビューは、多くの場合参照したデータをアプリケーション・サーバー近辺に保持します。マテリアライズ照会表つまりマテリアライズ・ビューは、 Web アプリケーションで生成されたトラフィックから綿密に調整したシステムをバッファーに入れる手段にもなります。

マテリアライズ照会表をニックネームで定義することができます。ニックネームとは、アクセスしたいデータ・ソース側にあるオブジェクトを参照するために使用される ID です。ニックネームは、いわゆるデータ・ソース・オブジェクトであるオブジェクトを識別します。ニックネームを使ってマテリアライズ照会表を参照すると、ローカル DB2 UDB インスタンスはリモート・データをキャッシュに入れることができます。フェデレーテッド照会の場合はキャッシング機能によってパフォーマンスが改善されます。それは、照会はリモート・データにローカルでアクセスするからです。リモート表を利用できない場合でも、リモート表で定義されているマテリアライズ照会表がルーティング基準を満たせば、 DB2 UDB はそのマテリアライズ照会表を使用することができます。この手法を使用すると、可用性とパフォーマンスが向上します。 REFRESH IMMEDIATE オプションは、ニックネームを参照するマテリアライズ照会表には適用できません。

マテリアライズ照会表を使用すれば、照会のたびに SUM 表に表されるような計算を反復しなくて済みます。数年にわたって顧客オーダーを保管している CUSTOMER_ORDER という名前の表があったとします。この表には、百万件を超えるレコードが収まっていて、その平均の行幅は 400 バイトあります。次に、2001 年度のオーダーに関する複数の照会を実行する必要が生じたけれども、その表の 3 つの列しか必要ないと想定します。以下は、3 つの列の情報を得るための典型的な SQL ステートメントです。

```
select SUM(AMOUNT), trans_dt
  from db2inst2.CUSTOMER_ORDER
  where trans_dt between '1/1/2001' and '12/31/2001'
  group by trans_dt
```

CUSTOMER_ORDER 表上に妥当な索引があれば、アクセス・パスはこのステートメントの索引スキャンを示します。

ただし、次のようにして、総計計算を含めて、必要な列と行の入ったマテリアライズ照会表を作成することができます。

```
CREATE TABLE DB2INST2.SUMMARY_CUSTOMER_ORDER_2001
AS
(SELECT SUM(AMOUNT) AS
  TOTAL_SUM, TRANS_DT, STATUS
  FROM DB2INST2.CUSTOMER_ORDER
  WHERE TRANS_DT
  BETWEEN '1/1/2001' AND '12/31/2001'
  GROUP BY TRANS_DT, STATUS)
DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

DATA INITIALLY DEFERRED 文節を使用すると、データは CREATE TABLE ステートメントの一部として表に挿入されません。表にデータを追加するには、REFRESH TABLE ステートメントを発行してください。表のデータは、 REFRESH TABLE ステートメントの発行時のスナップショットとしての照会結果を反映します。作成したマテリアライズ照会表にデータを追加するには、次のようなステートメントを発行します。

```
REFRESH TABLE DB2INST2.SUMMARY_CUSTOMER_ORDER_2001;
```

マテリアライズ照会表に対して実行する照会の方が高速です。マテリアライズ照会表のほうがサイズが小さく、行も短い (基本表の 400 バイトに比べて 45 バイトのみ) からです。

マテリアライズ照会表を使うと、 LAN 上の複数のサーバーにまたがって複数の表を結合する照会を最適化できます。マテリアライズ照会表を使用すると、1 つのサーバー上で応答セットをキャッシュに入れてから、いずれかのサーバーでのデータ変更の際にそのキャッシュを更新できます。どのタイプのフェデレーテッド・リレーショナル・データでも、この種の結合の候補にすることができます。それには、Oracle 表、SQL サーバー、Sybase、メッセージ・キュー、Web サービス、およびその他のリレーショナル・データ・ソースが含まれます。

運用データまたは常用データへの直接アクセスをなくすようにデータウェアハウスを使用する計画を立てます。随時照会でウェアハウス・データにアクセスするので、パフォーマンスは改良されます。データをデータウェアハウスに入れることで、運用データから抽出して意思決定のためにトランスフォームできる通知データのストアを作成します。たとえば、顧客表とアカウント表を結合して次のような表を作成して、不良アカウントに関する顧客情報とアカウント情報を保管します。

```
CREATE TABLE bad_account AS
     (SELECT customer_name, customer_id, a.balance
       FROM account a, customers c
       WHERE status IN ('delinquent', 'problematic', 'hot')
       AND a.customer_id = c.customer_id)
       DATA INITIALLY DEFERRED REFRESH DEFERRED
```

あるアカウントが義務不履行かどうかをユーザーがたずねると、 DB2 Universal Database™ オプティマイザーは、要求された情報をそのマテリアライズ照会表がキ

ャッシュに入れたことを認識します。 DB2 Universal Database は表 *account* にアクセスする代わりに、表 *bad_account* にアクセスします。これによって応答時間が改善されて、顧客情報が戻されます。

IBM® フェデレーテッド・システムには、コスト・ベースのオプティマイザーが組み込まれています。オプティマイザーは、カーディナリティーや索引などの標準データベース統計だけでなく、ネットワークおよびサーバー・リソースと、データ・ソース・エンジンで利用できる照会機能も取り扱います。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『プッシュダウンの可否に影響を与えるニックネーム特性』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『マテリアライズ照会表の作成』

セキュリティーおよび許可

分散コンピューティング・システムでは、ユーザー、アプリケーション・サーバー、およびリソース・マネージャーが世界中に散在していることが多いので、コンピューター・システム・リソースの安全を確保するのは複雑な作業になります。優れたセキュリティー・サービスは、認証と許可という 2 つの主な機能を備えています。

認証が行われるのは、プリンシパル (ユーザーまたはコンピューター・プロセス) がコンピューティング・リソースへのアクセスを初めて試みたときです。その時点で、有効なプリンシパルであることを証明するようセキュリティー・サービスから指示されます。ユーザーが人間であれば、通常はユーザー ID とパスワードを入力して自分の ID を証明します。このプロセスでは通常、暗号化鍵が提示されます。パスワードまたはキーが有効であれば、セキュリティー・サービスはそのユーザーにトークンまたはチケットを与えます。このトークンは、プリンシパルを識別して認証するためのものです。認証後にプリンシパルは、セキュリティー・サービスによって保護されたコンピューター・システムの境界内のリソースの使用を試みます。ただし、コンピューティング・リソースによっては、プリンシパルは正しい許可を受けている場合のみそれを使用できます。

認証済みのプリンシパルがリソースの使用を要求すると、許可が行われます。セキュリティー・サービスは、ユーザーがそのリソースを使用できるかどうかを判別します。通常は、アクセス・コントロール・リスト (ACL) をリソースに関連付けて許可を処理します。リソースでは、どのユーザーまたはプロセス (あるいは、ユーザーまたはプロセスのグループ) がそのリソースを使用する許可をもつかが定義されています。プリンシパルは、セキュリティー・サービスから許可を受けると、リソースにアクセスできるようになります。分散コンピューティング環境ではプリンシパルとリソースは、それぞれが相手に対して自分の ID を証明しないかぎり、互いの ID に疑いをもつ必要があります。その必要があるのは、リソースにアクセスするためにプリンシパルが ID を偽る可能性があるからです。またリソースが、プリンシパルから貴重な情報を得ることもあります。この問題を解決するためにセキュリティー・サービスには、信頼のおける第三者機関として稼働するセキュリティー・サーバーが配備されています。このサーバーはプリンシパルとリソースを認証して、この両者が互いに相手に対して自分の ID を証明できるようにします。

認証済みのユーザーは、適切な特権 (SELECT、INSERT、UPDATE、または DELETE など) をもっていなければなりません。その特権は、フェデレーテッド・データベースでのニックネームと、リモート・データ・ソース上の基礎を成す表またはその他のオブジェクトの両方を対象とするものでなければなりません。このやり方で、フェデレーテッド・データベースは要求を受け入れることができます。ローカルの DB2® Universal Database に対する特権をもっているだけでは十分ではありません。許可と特権の詳細は、「フェデレーテッド・システム・ガイド」を参照してください。

関連概念:

- 184 ページの『IBM DB2 Information Integrator で照会を設計する利点』
- 33 ページの『DADX Web サービスのセキュリティー』

第 2 章 Web サービスの開発

第 2 部では、Web サービスの開発と使用について説明します。 IBM DB2 Information Integrator には、Web サービス・プロバイダー (WORF)、および 2 種類の Web サービス・コンシューマー (SOAP ユーザー定義関数および Web サービス・ラッパー) が含まれています。

Web サービス・プロバイダーの概要

Web サービスとは、Web サービス・クライアント・インターフェースを使用することにより、インターネットを介してアプリケーションまたはその他の Web サービスがプログラマチックに呼び出せる一連のビジネス機能のことです。 IBM DB2 Information Integrator では、標準の SQL ステートメントおよび DB2 XML エクステンダーのストアード・プロシージャーを使って、基本的な Web サービスを定義することができます。 XML とリレーショナル・データ間の拡張トランスフォーメーションに携わる Web サービスの場合、DB2 XML エクステンダーを使用します。

Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF

DB2® Universal Database 情報へのリモート・アクセスを使用可能にするために、Web サービスを使用することができます。 Web サービスには、通知機能やトランザクション機能などの有益なサービスをコンシューマーまたはリクエスターのために実行する一連のアプリケーション機能が組み込まれています。 Web サービスは、単純な要求から複雑なビジネス・プロセスにいたるまでを網羅した機能を実行します。コンシューマーは一般的に、Web サービスへの Web サービス記述言語のインターフェースを知っている必要があるだけです。さらに、インターフェースに変更を加えない限り、通常はコンシューマーに影響を与えないで Web サービスを変更することができます。

Web サービスはインターオペラビリティーを促進します。現実のインターオペラビリティーは、情報技術業界が Web サービスの開発と統合へのガイダンスとなる一連の標準を使用することを想定しています。 Web Services Interoperability Organization は、プラットフォーム、アプリケーション、およびプログラム言語を越えた Web サービスのインターオペラビリティーを促進および確立するための、業界のオープンな取り組みです。 Web Services Interoperability Organization は、World Wide Web Consortium (W3C) と UDDI.org が開発した仕様を使用します。Web サービスのインターオペラビリティーとは、さまざまな SOAP または Web サービスのプラットフォーム (Apache SOAP、Apache Axis、または Microsoft® Visual Studio.Net を含む) で Web サービスを作成できることを意味します。

Web サービス・アプリケーション・プログラマーの設計では、サービス・プロバイダー、コンシューマー、またはサービス要求元の対話が、プラットフォームや言語による拘束をまったく受けないようになっています。サービス要求元はサービス・プロバイダーを動的に見つけられるので、タイミングよく統合を使用することがで

きます。 Web サービスは、カプセル化を通して複雑さを軽減します。サービス要求元とサービス・プロバイダーは、基礎を成すインプリメンテーションを度外視して、互いに対話するのに必要なインターフェースにのみ専心することができます。既存のアプリケーションを Web サービスとしてキャストできるので、レガシー・アプリケーションは Web サービスによって生まれ変わります。 Web サービスの基本エレメントには、Simple Object Access Protocol (SOAP)、Universal Description, Discovery, and Integration (UDDI)、および Web サービス記述言語 (WSDL) などがあります。

Web サービスを使えば、多様なデータベースおよびインターネットのロケーションからデータにアクセスすることができます。データにアクセスした後、Web サービス・コンシューマーはこれを検索、マイニング、およびトランスフォームすることができ、こうしてデータをデータウェアハウスと共に使用してさらに分析できるようにします。

簡単な Document Access Definition Extension (DADX) ファイルを使用することにより、データベースのデータにアクセスするよう Web サービスを定義できます。単純なテキスト・エディター、または WebSphere® Studio Application Developer と、WebSphere Studio から利用可能なウィザードを使用することにより、この DADX ファイルを作成することができます。これは XML ファイルです。

DADX ファイルは Web サービス・ランタイム環境を動かします。この環境には、さまざまなデータベース管理ツールと Web Object Runtime Framework (WORF) が含まれます。 WORF ランタイム環境は、SQL データ・タイプに対する XML スキーマの単純なマッピングの手段になります。 DADX ファイルには、 SELECT、 INSERT、 UPDATE、 DELETE、および CALL ステートメントなどの標準 SQL ステートメントを含むことができ、こうしてデータベースの照会と更新、およびストアード・プロシージャーの呼び出しを行うことができます。実行時に SQL ステートメントを処理するには、WORF に備わっている動的照会サービスを使用可能にする必要があります。そのためには DADX ファイルを使用します。このファイルには <DQS/> タグしか入っていません。

WORF ランタイム環境を使用しない場合は、独自の WSDL を開発するなどの Web サービスの作成の詳細を扱うために、独自のプログラムを作成する必要があります。 WORF が提供する機能には、次のようなものがあります。

- Web サービス要求の分析
- データベースへの接続
- SQL 要求の実行
- SQL 結果からの出力メッセージのエンコード
- クライアントへのメッセージの返送

DADX ファイルには DB2 XML エクステンダー・エレメントも含めることができます。たとえば、Document Access Definition (DAD) ファイル参照、XML コレクション操作 (XML 文書を生成および格納する)、またはユーザー定義タイプ (UDT) およびユーザー定義関数 (UDF) などです。 DAD ファイルは XML とリレーショナル・データ間のマッピングを定義します。 DB2 XML エクステンダーは、XML 文書をそのまま保管したり、必要に応じてサイド表に索引を付けたりするのに使用することができます。 DB2 XML エクステンダーはそれを行うのに、XML 列アクセ

ス方式を使用しますが、リレーショナル表のコレクションとしての場合は XML コレクション・アクセス方式を使用します。

WORF は、IBM® DB2 Information Integrator だけでなく、DB2 Universal Database™ バージョン 8 および WebSphere Studio バージョン 5 と一緒に使用可能です。 WORF は Informix® と一緒に使用することもできます。

関連概念:

- 37 ページの『Web サービス・プロバイダーのフィーチャー』
- 37 ページの『DADX ファイルの定義』
- 40 ページの『Web サービス・プロセスの概要』

DADX Web サービスのセキュリティー

アプリケーション・サーバーのセキュリティー・メカニズムを使用して、 Web サービスを保護できます。認証、暗号化、およびデータベース・ユーザー ID のメカニズムについて以下に説明します。

認証

認証を使用可能にして、DADX Web サービスのエンドポイントを保護できます。認証を使用可能にすると、認証された人物しか Web サービスを呼び出せないことが保証されます。 Java™ 2 Enterprise Edition (J2EE) では、アクセス・コントロールは、URL で指定されます。 DADX Web サービスごとに、Web サービスの別の部分 (エンドポイント、実際の WSDL の生成中、テスト・ページ中など) の URL を使用します。 DADX が使用する URL ごとに、それぞれの URL へのアクセスが許可されている役割を (および定義の一部としてユーザーも) 指定できます。 Web サービスの認証済みユーザーへの許可設定のプロセスは、 DB2® Universal Database 表への SELECT 特権の付与に似ています。

以下のリストは、Web サービス、特定の操作、または WORF により提供されるサービスを保護するために使用できる URL の例および URL パターンを示しています。

- DADX、テスト・ページ、および WSDL 生成のすべての URL に関するアクセス・コントロールを使用可能にするには、以下の URL を使用します。

  ```
  http://hostname:port/myContext/myGroup/myDadx.dadx/*
  ```

- テスト・ページのみのアクセス・コントロールを使用可能にするには、以下の URL を使用します。

  ```
  http://hostname:port/myContext/myGroup/myDadx.dadx/TEST
  ```

- グループ中のすべての DADX Web サービスに関するアクセス・コントロールを使用可能にするには、以下の URL を使用します。

  ```
  http://hostname:port/myContext/myGroup/*
  ```

WebSphere® Application Server バージョン 5 の Application Server Toolkit または Application Assembly Tool で、セキュリティーの制約を定義することもできます。 WebSphere Studio Application Developer バージョン 5 の

Web パースペクティブでもセキュリティーの制約を定義できます。制約の定義中に役割名を組み込んで、特定の役割名の人物が Web の領域にアクセスできるようにすることができます。

暗号化

HTTPS を使用してメッセージを暗号化することにより、DADX Web サービスを保護できます。暗号化を使用すると、 Web サービス・クライアントと Web サービス・プロバイダーの間で交換されるメッセージを誰も読み取れないことが保証されます。 HTTPS を使用可能にする方法を調べるには、ご使用のアプリケーション・サーバーに付属の資料を参照してください。

データベース・セキュリティー

WebSphere Application Server バージョン 5 では、アプリケーション・サーバー上で JNDI データ・ソースのデータベース・ユーザー ID を指定できます。 group.properties ファイル中でこの JNDI データ・ソースを参照して、 WebSphere で指定したユーザー ID を DB2 Web サービス・プロバイダーが使用できるようにすることができます。データベース・ユーザー・パスワードは、アプリケーション・サーバー上で暗号化されます。

認証、暗号化、およびデータ・ソース認証について詳しくは、ご使用の特定のアプリケーション・サーバーに関連した情報を参照してください。セキュリティーに関する特定の情報については、以下の WebSphere の資料も参照してください。

- 「*IBM WebSphere Application Server, Version 5: Security*」
- 「*IBM WebSphere V5.0 Security WebSphere Handbook Series*」

関連概念:

- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『WebSphere Studio』

関連タスク:

- 155 ページの『Web アーカイブ・ファイルの準備と作成』
- 76 ページの『group.properties ファイルのカスタマイズ』

関連資料:

- 289 ページの『付録 E. Web サービスのコマンド・リファレンス』
- 287 ページの『付録 D. Web サービスのエンコード・アルゴリズム』

iSeries での Web サービス・プロバイダーの使用

Web サービスとは、インターネットを介してアプリケーションまたはその他のサービスがプログラマチックに呼び出せる一連のビジネス機能のことです。 Web サービスの使用方法の 1 つに、DB2® Universal Database 情報へのリモート・アクセスの可能化があります。 Web サービスには、通知機能やトランザクション機能などの有益なサービスをリクエスターのために実行する一連のアプリケーション機能が組み込まれています。 Web サービスは、単純な要求から複雑なビジネス・プロセスにいたるまでを網羅した機能を実行します。リクエスターは一般的に、 Web サービスへのアプリケーション・プログラミング・インターフェース (API) を知っている必要があるだけです。さらに、通常は、リクエスターに影響を与えないで Web サービスを変更することができます。

Web サービスはインターオペラビリティーを促進します。 Web サービスの設計では、サービス・プロバイダーとサービス要求元の対話が、プラットフォームや言語による拘束をまったく受けないようになっています。サービス要求元はサービス・プロバイダーを動的に見つけられるので、タイミングよく統合を使用することができます。 Web サービスは、カプセル化を通して複雑さを軽減します。サービス要求元とサービス・プロバイダーは、基礎を成すインプリメンテーションを度外視して、互いに対話するのに必要なインターフェースにのみ専心することができます。既存のアプリケーションを Web サービスとしてキャストできるので、レガシー・アプリケーションは Web サービスによって生まれ変わります。 Web サービスの基本エレメントには、Simple Object Access Protocol (SOAP)、Universal Description, Discovery, and Integration (UDDI)、および Web サービス記述言語 (WSDL) などがあります。

Web サービスを使えば、多様なデータベースおよびインターネットのロケーションからデータを収集することができます。データを収集した後で、Web サービス・コンシューマーは、データを検索してマイニングしてから、さらに分析を行うためにデータウェアハウスに入っているデータのサブセットを使ってそのデータをトランスフォームすることができます。

SELECT、INSERT、UPDATE、DELETE、および CALL ステートメントなどの標準 SQL ステートメントをインプリメントする Web サービスを定義することができます。また、DB2 XML エクステンダーのストアード・プロシージャーを使って、そのような Web サービスを定義することもできます。

DB2 XML エクステンダーは、Document Access Definition (DAD) という名前の XML 文書フォーマットを使って、 XML とリレーショナル・データの間のマッピングを定義します。 Document Access Definition Extension (DADX) ファイルは Web サービスを指定します。その指定のためには、 SQL ステートメント、パラメーター・リスト、および DAD ファイル参照によって定義された一連の操作を用います。操作は、起動可能なプログラミング方式に似通っています。 XML コレクション操作を使って、XML 文書を生成および保管することができます。 SQL 操作を使えば、データベースの照会と更新や、ストアード・プロシージャーの呼び出しを行うことができます。

使用可能なデータベース管理ツールと、Web Object Runtime Framework (WORF) を使えば、 Web サービスから DB2 Universal Database™ ストアード・プロシージャーおよびデータにアクセスすることができます。これらのツールを使うと、XML データに対する保管および検索の操作の指定に加えて、ストアード・プロシージャーと SQL ステートメントを Web サービス操作として呼び出すことができます。 WORF は、SQL データ・タイプに対する XML スキーマの単純なマッピングの手段になります。

ストアード・プロシージャーを使用するときは、各 CREATE PROCEDURE ステートメントごとに作成される *PGM オブジェクトを許可する必要があります。 Java™ ストアード・プロシージャーを使用するときは、 Java クラス・ファイルに対してユーザー (または *PUBLIC) を許可しなければなりません。 Java ストアード・プロシージャーの使用時には、クラス・ファイルをすべて以下のディレクトリーに保管します。

```
/QIBM/UserData/OS400/SQLLib/Function
```

Spserver.class がこのディレクトリー内にあることを確認してください。ストアード・プロシージャー SAMPLE.TESTRS は、 SQL ストアード・プロシージャーである唯一のストアード・プロシージャーです。これは Java クラスに依存しません。

DB2 Universal Database でサンプル・ストアード・プロシージャーを定義してカタログするには、次のステップを行います。

1. >qsh
2.
```
>cd  /QIBM/UserData/WebASAEs4/worf/
     installedApps/servicesApp.ear/services.war/WEB-INF/
      classes/groups/dxx_sample
```

   ここで、**WebASAEs4** は WebSphere® のバージョン、また **worf** は WebSphere インスタンスの名前です。

3. >db2 -f Spcreate.db2

ストアード・プロシージャーの定義を除去するには、次のコマンドを実行します。

```
>db2 -f Spdrop.db2
```

WORF は IBM® DB2 Information Integrator の一部として使用可能です。またこれは、 DB2 Universal Database バージョン 8 および WebSphere Studio バージョン 4 とバージョン 5 にも添付されています。このツールが WebSphere Studio の付属品である場合、 DADX Web サービスの構築を自動化するのに使用することができます。このようなツールには、 SQL ステートメントまたは DAD ファイルをベースにした DADX ファイルを作成するウィザードが組み込まれています。また、DAD ファイルを作成するためのツールも組み込まれています。 WORF は、Informix® とも一緒に稼働します。

DB2 で WORF ランタイム環境を使用すれば、 DB2 XML エクステンダーを使って Web サービスをインプリメントすることができます。 DB2 XML エクステンダーは、ストアード・プロシージャー、ユーザー定義タイプ (UDT)、およびユーザー定義関数 (UDF) で構成されます。 DB2 を使用すれば、これらのフィーチャーを使って XML データの保管と検索を行うことができます。 DB2 XML エクステンダーは、XML 文書をそのまま保管したり、必要に応じてサイド表に索引を付けたりするのに使用することができます。このエクステンダーはそれを行うのに、XML 列アクセス方式を使用しますが、リレーショナル表のコレクションとしての場合は XML コレクション・アクセス方式を使用します。 DB2 XML エクステンダーは、Document Access Definition (DAD) という名前の XML 文書フォーマットを使って、 XML とリレーショナル・データの間のマッピングを定義します。

関連概念:

- 40 ページの『Web サービス・プロセスの概要』

関連タスク:

- 73 ページの『iSeries プラットフォームでの web.xml ファイルおよび group.properties ファイルの定義』
- 65 ページの『iSeries での WORF の例のインストールおよびデプロイ』
- 46 ページの『iSeries での Web サービス・プロバイダー・ソフトウェア要件のインストール』

DADX ファイルの定義

Document Access Definition Extension (DADX) ファイルは、どのように Web サービスを作成するかを指定します。 Web サービスは、Web を経由して呼び出す機能です。 Web サービスを作成するには、 SQL ステートメント、ストアード・プロシージャー呼び出し、または DAD ファイルによって定義された一連の操作を用います。 Web サービスは、Extensible Markup Language (XML) 文書の保管や、 DB2® XML エクステンダーによって管理されるものを含む XML 文書の検索を行います。 DADX ファイルに指定されている Web サービスは *DADX Web* サービス または IBM® DB2 Information Integrator Web サービスと呼ばれます。

WORF は、DADX 文書を Web サービスとして呼び出すためのランタイム・サポートを備えています。このような Web サービスは Apache Simple Object Access Protocol (SOAP) (バージョン 2.3 以降) エンジン、または Apache Axis エンジン (バージョン 1.2) を使用します。これらの SOAP エンジンは両方とも WebSphere® Application Server および Apache Jakarta Tomcat によりサポートされます。

DADX ファイルの内容により、Web サービスが、事前に決定された SQL 操作のセット、または動的 SQL 操作を使用するかどうかが決まります。 DADX ファイルに動的照会サービス・タグ (<DQS/>) が入っている場合、ブラウザーからSQL 操作を指定することができます。あるいは、WORF テスト Web アプリケーションがインストールしてある場合、操作をアプリケーションに組み込むことができます。

XML または SQL に関する最低限の知識があれば、単純なテキスト・エディター、または WebSphere Studio に付属しているツールを使用して、 DADX 文書を作成することができます。

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』

関連資料:

- 88 ページの『単純な DADX ファイル』

Web サービス・プロバイダーのフィーチャー

WORF には次のようなフィーチャーが備わっています。

- 38 ページのリソース・ベースのデプロイメント
- 開発時のリソース変更を定義するときのサービスの自動再デプロイメント
- SOAP に加えて、Hypertext Transfer Protocol (HTTP) の GET および POST バインディング
- WSDL および XSD ファイルの生成。UDDI の最良実例のサポートも含みます。これは業界標準となっています。
- 文書およびテスト・ページの生成
- HTML および XML 形式の Web Services Inspection Language (WSIL) ページの生成

WORF の主要フィーチャーは、Web サービスのリソース・ベースのデプロイメントのサポートです。 DADX ファイルなどのリソース・ファイルは WORF に対する Web サービスを記述しているので、このファイルから WORF は適切な Web サービスを生成することができます。そのリソース・ファイルを要求すると、 WORF はそのファイルをロードして、Web サービスとして利用できるようにします。リソース・ファイルを編集してからこのファイルを再度要求すると、 WORF は変更を検出して、新バージョンを自動的にロードします。 Web サービスの開発は、このようなリソース・ファイルの自動再ロードのプロセスによってさらに生産性が高まります。

自分専用のリソース・ファイルを作成することができます。ただしリソース・ファイルは、所定の構文と意味規則に従っていなければなりません。リソース・ファイルは互いに参照しあうことができます (たとえば、 DAD ファイルへの参照を DADX ファイルに組み込むことができます)。 Web サービスを正しくデプロイするためには、そのような参照が正しくなければなりません。

WORF では、Extensible Markup Language (XML) データの保管および検索の操作の指定以外に、呼び出し可能な Web サービス操作としてストアード・プロシージャーと SQL ステートメントを公開することができます。任意のデータベース・ストアード・プロシージャーを公開することができます。 WORF では、ストアード・プロシージャーの結果セット内には固定のメタデータがあることが前提になっています。固定のメタデータとは、固定数および固定形状のデータのことを言いますが、それは、特定の列名とデータ・タイプをもった特定数の列を意味します。操作シグニチャーには、入力および出力のパラメーターを併記します。 WORF が提供する動的照会サービス (DQS) を使用するときには、メタデータの修正セットまたは必要な結果セットなしで、ストアード・プロシージャーを実行できます。また、データの選択、挿入、更新、および削除のための SQL ステートメントを指定することもできます。さらに WORF には、 SQL データ・タイプに対する Extensible Markup Language (XML) スキーマの単純マッピングも用意されています。これらのフィーチャーは XML エクステンダーを必要としません。

関連概念:

- 101 ページの『DADX ファイルからの WSDL』
- 128 ページの『GET、POST、および SOAP バインディングによる Web サービスへのアクセス』
- 145 ページの『Web サービスの自動再ロード』
- 144 ページの『Web サービス文書』
- 138 ページの『Web サービス・プロバイダーに存在する Web サービス』

関連資料:

- 79 ページの『DADX ファイルの構文』

DADX ファイルを使った Web サービス・プロバイダー操作

DADX ファイルは、非動的 SQL 操作、動的 SQL 操作、XML コレクション操作の 3 つの種類の Web サービス操作をサポートします。 SQL ベースの照会を使用すると、ストアード・プロシージャー呼び出しをはじめとする SQL ステートメントを DB2® に送信して、デフォルトのタグの付いた結果を戻すことができます。エ

レメントとして列名を使った SQL データ・タイプの単純なマッピングのみの使用で、アプリケーションはデータを戻します。

SQL 操作: 非動的

SQL 操作は非動的なものとなることがあります。非動的操作は DADX ファイル内で事前定義された操作です。事前定義 SQL 操作のタイプは次の 3 つのエレメントで構成されます。

<query>
データベースを照会する。

<update>
データベースへの挿入、データベースからの削除、またはデータベースの更新を行う。

<call> 0 個以上の結果セットを戻す可能性のあるストアード・プロシージャーを呼び出す。

SQL 操作: 動的

SQL 操作は DADX ファイルの内容によっては、動的操作になる可能性があります。動的操作とは、SQL 操作が事前定義されていない、SOAP メッセージで生成される操作です。以下のエレメントが動的操作になります。

<getTables>
使用可能な表の記述を検索する。

<getColumns>
列の記述を検索する。

<executeQuery>
単一の SQL ステートメントを発行する。

<executeUpdate>
単一の INSERT、UPDATE、DELETE を発行する。

<executeCall>
単一のストアード・プロシージャーを呼び出す。

<execute>
単一の SQL ステートメントを発行する。

Extensible Markup Language (XML) コレクション操作 (DB2 XML エクステンダーが必要)

このような保管および検索の操作は、DB2 Universal Database™ 表に対する XML 文書構造のマッピングに役立ちます。既存の DB2 データからの XML 文書の作成、または DB2 データへの XML 文書の分解 (タグなしのエレメントまたは属性コンテンツの保管によって) のどちらでも行うことができます。この方式は、データ交換アプリケーションにとって便利ですが、特に XML 文書のコンテンツをひんぱんに更新するアプリケーションにとって便利です。

XML コレクション操作タイプを構成するエレメントには次の 2 種類があります。

<retrieveXML>
XML 文書を生成する。

<storeXML>
XML 文書を保管する。

DAD ファイルは、 DB2 データベースに対する XML 文書のマッピングを、保管および検索の両面においてきめ細かく制御する手段になります。

関連概念:

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』
- 126 ページの『Web サービス・アプリケーションのテスト - シナリオ』
- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』

関連資料:

- 94 ページの『DADX 操作の例』
- 114 ページの『Web サービス・プロバイダーにおける動的照会サービス操作』

Web サービス・プロセスの概要

DB2® Universal Database を Web サービス・プロバイダーにするのに必要なステップの概要を以下に示します。このステップにより、Web アプリケーション開発者は以下の一般的な階層を作成します。

```
Web アプリケーション -> グループ ->
  DADX ファイル (Web サービス) -> SQL 操作
```

エンタープライズ・アーカイブ・ファイルを使用する場合には、一般的な階層は以下のようになります。

```
エンタープライズ・アプリケーション -> Web アプリケーション ->
  グループ -> DADX ファイル (Web サービス) -> SQL 操作
```

1. データベース管理者は、データベースをセットアップします。
2. データベース管理者は、判断によっては DB2 XML エクステンダー用のデータベースを使用可能にします。 retrieveXML 操作と storeXML 操作には DB2 XML エクステンダーが必要です。 XML 列を使用するのにも DB2 XML エクステンダーが必要です。 XML エクステンダー用にデータベースを使用可能にする方法の詳細は、「*DB2 XML Extender 管理およびプログラミングの手引き*」の中の管理の章を参照してください。
3. Web アプリケーション開発者は、エンタープライズ・アーカイブまたは Web アーカイブ (EAR または WAR) ファイルを作成します。 EAR または WAR ファイルがデプロイされると、それは Web アプリケーションを入れるフォルダーとなり、そこでさらに変更を加えることができます。 EAR ファイルは、Java™ 2 Enterprise Edition 仕様 (J2EE) で記述されるエンタープライズ・アプリケーションです。 WAR ファイルも J2EE 仕様で記述されています。 Web アプリケーション・フォルダーは、サーバー上の、関連しあったファイルとツールのコレクションです。インターネットを通して業務を遂行するインフラストラクチャーを作成するための、インターフェース、プログラムの流れ、プログラム・ロジック、およびデータ・アクセス情報などが、Web アプリケーションに組み込まれています。詳細は、以下の資料を参照してください。
   - WebSphere® Application Server Advanced Edition バージョン 5 の資料

WebSphere Application Server 上に WAR または EAR ファイルを作成できます。

- Apache Jakarta Tomcat の資料

  Tomcat 上には WAR ファイルのみを作成できます。

4. Web アプリケーション開発者はその後、グループおよびそのグループの group.properties ファイルを作成します。 group.properties ファイルには、データベース接続に関する情報と、WORF で使用されるその他の関連情報が入っています。グループとは、データベースにアクセスする複数の Web サービス操作のことです。データベースごとに 1 つのグループを設けることができます。あるいは同じデータベースに複数のグループを設けることさえできます。さらにそのグループ内に 1 つ以上の DADX ファイルを定義できます。 Web サービスを具体的に定義する DADX ファイルには、Web サービスを実行する操作が入っています。グループ・プロパティーの詳細は、『web.xml ファイルおよび group.properties ファイルの定義』の項を参照してください。
5. データベース開発者は、判断しだいで XML およびリレーショナル・データ変換のマッピングのための DAD を作成します (XML エクステンダーのストアード・プロシージャーの使用時に必要です)。
6. Web サービス開発者は、DADX 文書を作成します。 DADX ファイルの内容によって、Web アプリケーション内で動的照会を使用できるかが決定されます。DADX ファイルには、動的照会サービスのタグ (</DQS>) があります。このファイルには、動的照会を使用可能にするスイッチの役割を果たすそのタグのみが入っています。非動的 DADX ファイルは一連の操作を定義します。このファイルには、Web サービスを作成するために使用される情報が入っています。DADX ファイルの作成規則に関する詳細は、 『DADX ファイルの構文』の項を参照してください。
7. オプション: Windows® または UNIX® 用の WebSphere Application Server 構成マネージャーを使用する場合、 Web サービス開発者はその Web サービスのデプロイメント記述子を作成してデプロイします。デプロイメント記述子とは、**isd** ファイル (Apache SOAP エンジンと共に使用される)、または **deploy.wsdd** ファイル (Apache Axis エンジンと共に使用される) のことで、これらは構成とデプロイメントの情報を識別します。 Apache Axis エンジンでは、デプロイメント記述子は WORF により自動的に作成されます。 Apache SOAP を使用するそれぞれの Web サービスには、 1 つの **.isd* ファイルを入れることができます。 Web アプリケーションには複数の Web サービスを入れることができます。すべての isd または wsdd ファイルを dds.xml ファイルにコピーします。(Apache Jakarta Tomcat の使用時にはこのステップは自動化されます。) デプロイメント記述子の詳細は、『デプロイメント記述子の生成』の項を参照してください。
8. DADX テスト・ページを使用することにより、 Web サービスを検査することができます。IBM® DB2 Information Integrator に付属する WORF の例をデプロイした場合に、このページを使用可能です。 Java Server Page を WORF ディレクトリーから apache-services.war ファイルまたは axis-services.war ファイルにコピーして、アプリケーションのいくつかの WORF 機能をテストできます。

環境によっては、これらのタスクは 1 人の担当者が実行できることに注意してください。

関連概念:

- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』
- 69 ページの『Web サービスのグループの定義』

関連タスク:

- 203 ページの『フェデレーテッド・アプリケーションのデプロイ』
- 152 ページの『デプロイメント記述子の生成』
- 70 ページの『web.xml ファイルおよび group.properties ファイルの定義』
- 53 ページの『WebSphere Application Server Version 4.0.4 (z/OS または OS/390 用) での WORF の例のデプロイ』
- 50 ページの『WebSphere Application Server Version 5.1 以降 (Windows および UNIX 用) での WORF の例のデプロイ』

関連資料:

- 79 ページの『DADX ファイルの構文』

Web サービス・プロバイダーのインストールと構成

システムの容量を判別してから、ご使用のアプリケーション・プログラミング・インターフェースと Web サービス・アプリケーション用にインストールする必要のあるソフトウェアの計画を立てます。

Web サービス・プロバイダー・ソフトウェアの要件 (UNIX および Windows)

以下のいずれかのオペレーティング・システムで、 Web Services Object Runtime Framework (WORF) つまり Web サービス・プロバイダーをセットアップします。

- Windows NT®
- Windows 2000®
- Linux
- AIX®
- Solaris オペレーティング環境

WORF は、IBM DB2 Information Integrator と共にパッケージされているだけでなく、 DB2 Universal Database Extended Server Edition バージョン 8、および WebSphere Studio バージョン 5 の一部ともなっています。 DB2 Universal Database バージョン 8 では WORF は以下のパス内にあります。 *<DB2 UDB がインストールされているロケーション>¥samples¥java¥Websphere¥dxxworf.zip*。

以下のデータベース環境を使用することができます。

- IBM DB2 Universal Database™ バージョン 8 以降

  http://www.ibm.com/software/data/db2。これには DB2 XML エクステンダーも組み込まれています。
- Informix Dynamic Server (IDS) バージョン 9.3

さらに、以下のソフトウェアを使用します (使用するサーバーによっては、このソフトウェアのほとんどはすでにご利用の環境の一部となっています)。

- Java™ Java Development Kit (JDK) バージョン 1.2 または 1.3 (http://java.sun.com、または http://www.ibm.com/java)
- 以下の Web サーバーのうちのいずれか
  - WebSphere Application Server Advanced Edition バージョン 5 (http://www.ibm.com/software/webservers/appserv/)
  - Apache Web サーバー
    - Apache Jakarta Tomcat バージョン 3.3.1 から 4.0.3 あるいはそれ以降 (http://www.apache.org/)
      - Apache Jakarta Tomcat バージョン 4 標準は、該当する Xerces に付属しています
      - バージョン 4 より前のバージョンの Apache Jakarta Tomcat の場合、XML パーサーとして使用するには CLASSPATH に Xerces パーサーを追加する必要があります。
  - SOAP 2.3 以降のバイナリー、または Apache Axis 1.2 (http://xml.apache.org/ (Document Object Model レベル 2 (DOM 2) が必要です。これは Xerces Java 1.4.4 以降によってサポートされます。))
  - Xerces Java パーサー、バージョン 1.4.4 (http://xml.apache.org/)
  - JavaMail バージョン 1.2 (http://java.sun.com/)
  - JavaBeans™ JavaBeans Activation Framework (http://java.sun.com/)。 Apache SOAP には JavaBeans Activation Framework が必要です。
  - j2ee.jar バージョン 1.3 以降 (http://java.sun.com/)
  - qname.jar (http://java.sun.com/)
  - *wsdl4j.jar*。このファイルは http://oss.software.ibm.com/developerworks/projects/wsdl4j からダウンロードできます。

関連概念:

- 37 ページの『DADX ファイルの定義』

関連タスク:

- 63 ページの『Apache Jakarta Tomcat での WORF の例のインストールおよびデプロイ』
- 44 ページの『Web サービス・プロバイダー・ソフトウェア要件のインストール』

OS/390 および z/OS 用の Web サービス・プロバイダー・ソフトウェア要件

以下のどのオペレーティング・システムでも、 Web サービス・プロバイダー (Web Object Runtime Framework) をセットアップすることができます。

- OS/390 バージョン 2.8 以降
- z/OS バージョン 1.1 以降

以下のデータベース環境を使用することができます。

- IBM DB2 Universal Database for OS/390 バージョン 7 または DB2 Universal Database for z/OS (http://www.ibm.com/software/data/db2/os390)
- IBM DB2 XML Extender for OS/390 バージョン 7 以降 (http://www.ibm.com/software/data/db2/extenders/xmlext/index.html)。保管および検索の操作で必要です。

それ以外に、以下のソフトウェアも使います。

- WebSphere Application Server バージョン 4.01 サービス・レベル W401505 以降
- JavaMail バージョン 1.2 (http://java.sun.com/)
- JavaBeans Activation Framework バージョン 1.0.1(http://java.sun.com/)
- j2ee.jar バージョン 1.3 以降 (http://java.sun.com/)
- qname.jar (http://java.sun.com/)
- IBM XML Toolkit for z/OS and OS/390 バージョン 1.4 (プログラム一時修正 (PTF) UW95866 付き) (http://www.ibm.com/servers/eserver/zseries/software/xml/)
- *wsdl4j.jar*。このファイルは http://oss.software.ibm.com/developerworks/projects/wsdl4j からダウンロードできます。

関連タスク:

- 127 ページの『Web サービスのテスト』
- 53 ページの『WebSphere Application Server Version 4.0.4 (z/OS または OS/390 用) での WORF の例のデプロイ』

UNIX、Windows、z/OS、および OS/390 での WebSphere Application Server 向けの Web サービス・プロバイダーの構成

Web サービスを WebSphere® Application Server Advanced Edition 上で実行することができます。 Apache SOAP 2.3 (またはそれ以降) または Apache Axis 1.2 (またはそれ以降) を使い Hypertext Transfer Protocol (HTTP) を介して Web サービスとして Document Access Definition Extension (DADX) 文書を呼び出すためのランタイム・サポートが、 Web Services Object Runtime Framework (WORF) に備えられています。 WebSphere Application Server 5 以降と、その他のサーブレット・エンジンもこれをサポートしています。 WebSphere では、SOAP Web サービスを保護することができます。詳細は、SOAP サービスの安全化に関する WebSphere の資料を参照してください。以下の項に、Web サービスについての解説が述べられています。

Web サービス・プロバイダー・ソフトウェア要件のインストール

前提条件:

必要なソフトウェアのインストールが完了したことを確認してください。 Windows および UNIX でのインストールの検査の詳細は、 『Web サービス・プロバイダー・ソフトウェアの要件 (UNIX および Windows)』の項を参照してください。

XML とリレーショナル・データの間の拡張マッピングの制御には、 DB2® XML エクステンダーが必要です。 DB2 UDB SAMPLE データベースがまだ作成されていない場合には、それを作成して DB2 UDB のインストールを検査します。 Web サービスには、 DB2 Universal Database バージョン 8 におけるデフォルトである Java Database Connectivity (JDBC) が必要です。

z/OS のインストールの場合は、 『OS/390 および z/OS 用の Web サービス・プロバイダー・ソフトウェア要件』の項を参照してください。

手順:

UNIX および Windows で Web サービス環境を準備する手順は次のとおりです。

1. DB2 Universal Database を使用するすべてのサービス (WebSphere Application Server など) を停止します。
2. DB2 を停止します。
3. バージョン 8 より前の DB2 Universal Database の場合、 Java Database Connectivity (JDBC) 2.0 を選択します。 *C:¥SQLLIB¥java12¥usejdbc2.bat* ファイルを実行します。ただし、Windows 環境を使用して、 DB2 を C:¥SQLLIB¥ にインストールしていると想定しています。
4. DB2 を再始動します。
5. WebSphere Application Server Advanced Edition 5.1 をインストール・ディレクトリーから始動します。これらの指示では、WebSphere Application Server を C:¥WebSphere¥Appserver の Windows 環境にインストール済みであると想定しています。

OS/390 または z/OS で Web サービス環境を準備する手順は次のとおりです。

1. アプリケーション拡張を保管するディレクトリーがまだなければ、新規に作成します。
2. 指定の J2EE サーバー・インスタンス内の APP_EXT_DIR 環境変数を、そのアプリケーション拡張ディレクトリーに設定します。
3. 以下の JAR ファイルをそのアプリケーション拡張ディレクトリーに追加します。

   xerces.jar
   このファイルは、 http://www.ibm.com/servers/eserver/zseries/software/xml/ からダウンロードできる IBM XML Toolkit for z/OS にあります。

   mail.jar
   このファイルは、JavaMail 内にあります。

   activation.jar
   このファイルは、Java Beans Activation Framework 内にあります。

   j2ee.jar
   このファイルは、http://java.sun.com/products からダウンロードできます。

   qname.jar
   このファイルは、http://java.sun.com/products からダウンロードできます。

   wsdl4j.jar
   このファイルは http://oss.software.ibm.com/developerworks/projects/wsdl4j からダウンロードできます。
4. 以下のステップを行って、J2EE サーバー・インスタンスの構成を検査します。
   - WebSphere Application Server に組み込まれている soap.jar が CLASSPATH の中にあることを確認します。

- J2EE サーバー・インスタンスの jvm.properties ファイルに以下の設定を追加します。

```
com.ibm.ws390.server.classloadermode=2
com.ibm.ws.classloader.ejbDelegationMode=false
```

5. J2EE サーバーを再始動します。

関連タスク:

- 43 ページの『OS/390 および z/OS 用の Web サービス・プロバイダー・ソフトウェア要件』
- 61 ページの『Apache Jakarta Tomcat での WORF のインストールまたは移行』
- 48 ページの『WebSphere Application Server Version 5 以降 (Windows および UNIX 用) での WORF のインストールおよび移行』

関連資料:

- 42 ページの『Web サービス・プロバイダー・ソフトウェアの要件 (UNIX および Windows)』

iSeries での Web サービス・プロバイダー・ソフトウェア要件のインストール

前提条件:

必要なソフトウェアのインストールが完了したことを確認してください。以下のコマンドを使って、対話式 SQL から SAMPLE データベースを作成します。

```
CALL QSYS/CREATE_SQL_SAMPLE('SAMPLE')
```

XML とリレーショナル・データの間の拡張マッピングの制御には、 DB2 XML エクステンダーが必要です。 DB2 Universal Database の XML エクステンダーを使用するには、同製品がインストールされていることを確認します。システムに DB2 Universal Database の XML エクステンダーがインストールされていることを確認するには、 CL コマンド **GO LICPGM** を発行します。 DB2 for iSeries, V5R2 の場合に DB2 Universal Database の XML エクステンダーがあれば、 GO LICPGM コマンドの実行によって次のような項目が表示されます。

- 5722DE1 *COMPATIBLE DB2 UDB Extenders
- 5722DE1 *COMPATIBLE DB2 UDB Text Extender
- 5722DE1 *COMPATIBLE DB2 UDB XML Extender
- 5722DE1 *COMPATIBLE Text Search Engine

CL コマンド **CALL PGM(QDBXM/QZXMADM) PARM(enable_db LOCALRDB)** を使って、DB2 Universal Database の XML エクステンダーを使用可能にします。LOCALRDB は、リレーショナル・データベース・ディレクトリー内の *LOCAL データベース名です。リレーショナル・データベースの項目を処理するには、 CL コマンド **WRKRDBDIRE** を発行します。サンプル・ファイル内の文書タイプ定義 (DTD) を使用する場合、スクリプト *setup-dxx.cmd* を実行します。 WORF には、DB2 Universal Database バージョン 8 におけるデフォルトである Java Database Connectivity (JDBC) 2.0 が必要です。

手順:

WORF 環境を準備する手順は次のとおりです。

1. DB2 を使用するすべてのサービス (WebSphere Application Server など) を停止します。
2. バージョン 8 より前の DB2 Universal Database の場合、 Java Database Connectivity (JDBC) 2.0 を選択します。 *C:¥SQLLIB¥java12¥usejdbc2.bat* ファイルを実行します。ただし、Windows 環境を使用して、 DB2 を C:¥SQLLIB¥ にインストールしていると想定しています。
3. WebSphere Application Server Advanced Edition を C:¥WebSphere¥Appserver にインストールしたと想定して、 WebSphere Application Server Advanced Edition 4.01 または 5.0 を始動します。

関連概念:

- 34 ページの『iSeries での Web サービス・プロバイダーの使用』

関連タスク:

- 64 ページの『iSeries での Apache Jakarta Tomcat 用の Web サービス・プロバイダー・ソフトウェア要件のインストール』
- 73 ページの『iSeries プラットフォームでの web.xml ファイルおよび group.properties ファイルの定義』
- 65 ページの『iSeries での WORF の例のインストールおよびデプロイ』

XML エクステンダーのための DTD 定義

データベース管理者はアプリケーションに必要なすべてのデータベースまたはサブシステムのセットアップを完了し、 DB2 XML エクステンダーで使える (XML エクステンダーを使用している場合) ようになったことを確認してください。以下の表は、XML エクステンダー・サンプルを参照するデフォルト・ロケーションを一覧で示しています。

表 3. XML エクステンダー・サンプルが参照する文書タイプ定義 (DTD)

プラットフォーム	DTD のデフォルト・ロケーション
DB2 UDB バージョン 7.2 フィックスパック 7 以降の Windows	c:¥dxx¥samples¥dtd¥ getstart.dtd c:¥dxx¥dtd¥dad.dtd
DB2 UDB バージョン 8 Windows	c:¥<DB2 UDB installed location>¥samples¥ db2xml¥dtd¥getstart.dtd c:¥<DB2 UDB installed location>¥samples¥ db2xml¥dtd¥dad.dtd
Solaris オペレーティング環境上の DB2 UDB バージョン 8	/opt/IBMdb2/V8.1/samples/ db2xml/dtd/dad.dtd
AIX 上の DB2 UDB バージョン 8	/usr/opt/db2_08_01/ samples/db2xml/dtd/dad.dtd
Linux 上の DB2 UDB バージョン 8	/usr/IBMdb2/V8.1/samples/ db2xml/dtd/dad.dtd
OS/390 および z/OS 上の DB2 UDB バージョン 7 あるいは 8	/u/USER/dxx/dtd/dad.dtd

たとえば、dad.dtd を参照するファイルには以下のようなものがあります (これですべてではありません)。

- department.dad
- department2.dad
- departmentStd.dad
- order.dad
- order-public.dad
- getstart.xml
- order-10.xml
- sales_db.nst

関連タスク:

- 100 ページの『文書タイプ定義を XML スキーマに変換する』

関連資料:

- 94 ページの『DADX 操作の例』

WebSphere Application Server Version 5 以降 (Windows および UNIX 用) での WORF のインストールおよび移行

前提条件:

WebSphere Application Server をワークステーションの C:¥WebSphere¥Appserver (Windows 環境の場合) といったパスにインストールします。

手順:

WORF を旧バージョンからバージョン 8.2 に移行するには、『移行のサマリー』のセクションを参照してください。 WORF バージョン 8.2 をインストールするには、以下のステップを行います。

1. *dxxworf.zip* を *C:¥worf* などのディレクトリーに unzip して、ディレクトリーに以下の内容が含まれるようにします。
   - *readme.html*
   - *lib¥apache-services.war* および *lib¥axis-services.war* - WORF を使用する Web サービスが含まれるサンプル Web アプリケーション。
   - *lib¥worf.jar* - WORF ライブラリー。これは、サーブレット・エンジンのクラス・パスにインストールします。
   - *lib¥worf-servlets.jar*
   - *schemas¥* - DADX およびネーム・スペース表 (NST) XML ファイルの Extensible Markup Language (XML) スキーマ。`wsdl.xsd`、`db2WebRowSet.xsd`、および `dadx.xsd` が含まれます。
   - *tools¥* - tools ディレクトリーには、DAD および DADX チェッカー・ツールが入っています。
2. 使用するサーバー (WebSphere Application Serverなど) のディレクトリーに、該当する Web サービス・エンジンの JAR ファイルが含まれているかどうかを確かめてください。ファイルがディレクトリーにない場合、Apache Axis または

Apache SOAP ファイルが含まれるディレクトリーから JAR ファイルをコピーして、該当する Web サービス・エンジンを使用可能にします。

- Apache Axis フレームワークを使用する場合、axis.jar を c:¥WebSphere¥AppServer¥lib にコピーします。次に、他の Apache Axis JAR ファイルにアクセスするため、axis/lib の内容を c:¥WebSphere¥AppServer¥lib にコピーします。
- Apache SOAP フレームワークを使用する場合、soap.jar を WebSphere¥AppServer¥lib にコピーします。

3. *worf.jar* を *C:¥WebSphere¥AppServer¥lib* にコピーします。
4. WebSphere 5.0.2 よりも前のリリースの WebSphere サーバーの場合、http://java.sun.com/xml/downloads/saaj.html から `saaj.jar` という名前のファイルをダウンロードする必要があります。 `saaj.jar` を *C:¥WebSphere¥AppServer¥lib* にコピーします。
5. WebSphere Application Server を開始します。
6. 「スタート」—> 「プログラム」—>「IBM WebSphere」—>「管理コンソール」と選択して、管理コンソールを開きます。
7. 次のように WebSphere を DB2 UDB 環境で実行するように構成します。
   a. 左側のナビゲーション・ペインで、「サーバー (Servers)」—>「Application Servers」とクリックします。
   b. 右側のコンテンツ・ペインで、ご使用のサーバーの名前を検索し、サーバー名をクリックします。
   c. 「処理定義 (Process Definition)」—>「Java 仮想マシン (Java Virtual Machine)」とクリックします。
   d. 「構成 (Configuration)」ページで、クラスパスを Java データベース情報のパスに指定します。 DB2 Universal Database がディレクトリー sqllib¥ にインストール済みであり、WORF サンプルに付属する group.properties ファイルを使用する場合、以下が有効なパスの例になります。

      ```
      C:¥SQLLIB¥java¥db2java.zip
      ```

   e. **「適用」**または**「OK」**をクリックします。
   f. 構成を保管します。
8. WebSphere Application Server を停止します。

移行のサマリー

WORF を旧バージョンからバージョン 8.2 に移行するには、以下を行います。

1. バージョン 8.2 の dxxworf.zip の *lib¥worf.jar* を *C:¥WebSphere¥AppServer¥lib* にコピーして、worf.jar ファイルを置き換えます。 dxxworf.zip は次のパスに置かれています。 *<DB2 UDB をインストールした場所>¥samples¥java¥Websphere¥dxxworf.zip*。
2. デプロイした各アプリケーションの worf ディレクトリーに入っている JSP ファイルを、 apache-services.war または axis-services.war の worf ディレクトリーのファイルで置き換えます。次に、アプリケーションを再デプロイします。

関連概念:

- 69 ページの『Web サービスのグループの定義』

関連タスク:

- 44 ページの『Web サービス・プロバイダー・ソフトウェア要件のインストール』

WebSphere Application Server Version 5.1 以降 (Windows および UNIX 用) での WORF の例のデプロイ

前提条件:

- WebSphere Application Server をワークステーションの C:¥WebSphere¥Appserver (Windows 環境の場合) といったパスにインストールします。
- WORF をインストールします。

手順:

WORF の例をインストールしてデプロイするには、以下のステップを行います。

1. WebSphere Administration Server を始動します。
2. 「スタート」—> 「プログラム」—>「IBM WebSphere」—>「管理コンソール」と選択して、管理コンソールを開きます。
3. 「アプリケーション (Applications)」—>「エンタープライズ・アプリケーション (Enterprise Applications)」と選択します。コンテンツ・ウィンドウには、現行サーバーにインストール済みのすべてのエンタープライズ・アプリケーションが表示されます。
4. **「インストール」** ボタンをクリックして、*apache-services.war* または *axis-services.war* をエンタープライズ・アプリケーションとしてインストールします。
   a. **「ローカル・パス (Local path)」**フィールドから、**「ブラウズ」**ボタンをクリックして、c:¥WORF¥lib ディレクトリーに含まれる正しいサービス・ファイルへのパスを見付けます。 WORF には 2 つのサービス・ファイルが付属しており、サンプルの実行時にどちらかを選択できるようになっています。これらのファイルは、apache-services.war と axis-services.war です。実行中の SOAP エンジンに応じて正しいサービス・ファイルを選択します。
   b. **「コンテキスト・ルート (Context Root)」**フィールドで Web アプリケーションのコンテキスト名を指定します。本書で説明されている例を実行するには、コンテキスト・ルート名を `services` に指定する必要があります。

   51 ページの図 4 は、アプリケーションのインストール中の WebSphere Application Server 管理者コンソールを示しています。

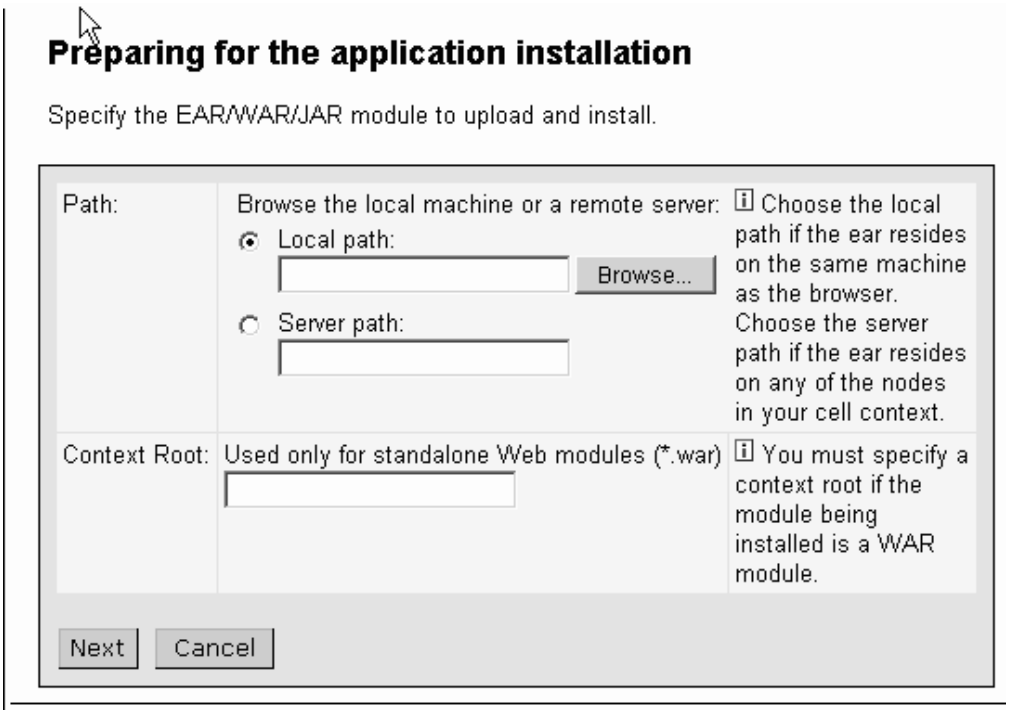


図 4. アプリケーションまたはモジュールの指定

c. **「次へ」**をクリックします。

d. 他のデフォルトすべてを受け入れ、ウィザードの残りの部分では**「次へ」**をクリックします。**「Web モジュール用の仮想ホストのマップ (Map virtual hosts for web modules)」**ウィンドウで、.WAR ファイルを選択し、**「次へ」**をクリックします。**「アプリケーション・サーバーに対するモジュールのマップ (Map modules to application servers)」**ウィンドウで、.WAR ファイルを選択し、**「次へ」**をクリックします。構成オプションでは、仮想ホスト (たとえば default_host) と、アプリケーション・サーバー (たとえば Default Server) が指定されます。ウィザードの最後に、**「完了」**をクリックします。

e. 最後のウィンドウには、「構成への保管 (Save to Configuration)」が表示されます。**「保管」**をクリックします。

5. group.properties ファイル内のデータベース設定 (特にユーザー ID とパスワード) が正しいことを検証します。

6. 各データベース・ディレクトリーごとに Windows 環境の DB2 Universal Database コマンド・ウィンドウ (DB2 UDB コマンド行プロセッサー・ウィンドウ) で *setup.cmd* を発行するか、または、UNIX 環境のコマンド・ウィンドウで *setup.sh* を発行して、データベースを作成します。たとえば、DB2 XML エクステンダーを使用する SALES_DB データベースをセットアップするには、 *dxx_sales_db* ディレクトリーで setup.cmd を実行します。

   重要: セットアップ・コマンドを dxx_sample ディレクトリーで発行すると、コマンドはドロップし、SAMPLE データベースが再作成されます。 DB2 Universal Database 製品に付属する SAMPLE データベースを使用する場合、データベースに加えた変更が失われないよう注意してください。

7. ユーザー独自のアプリケーションをデプロイする場合、worf-servlets.jar ファイルを WORF ディレクトリーから WebSphere/AppServer/installedApps/<host>/<application WAR directory>/WEB-INF/lib にコピーしてください。
8. 現行サーバーを停止します。
9. サーバーを再始動します。
10. 「アプリケーション (Applications)」—>「エンタープライズ・アプリケーション (Enterprise Applications)」を選択して、services.war がすでに実行しているかどうか確かめます。
11. ブラウザー・ウィンドウを開いて、Web アプリケーションのウェルカム・ページにアクセスしてインストールをテストします。

    一部のポート番号は、WebSphere Application Server 構成によって異なります。デフォルトを使用した場合、サービスの Web アプリケーションのウェルカム・ページは *http://localhost:9080/services* である場合もあります。 `services` は前のステップで作成されたアプリケーションの名前であることを忘れないようにしてください。このページは、図 5 および 53 ページの図 6 に示されているようなものです。

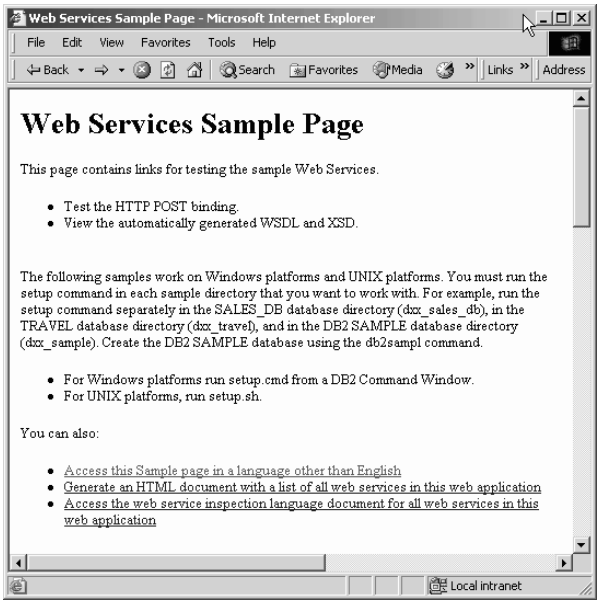


図 5. WORF サンプル・ページ

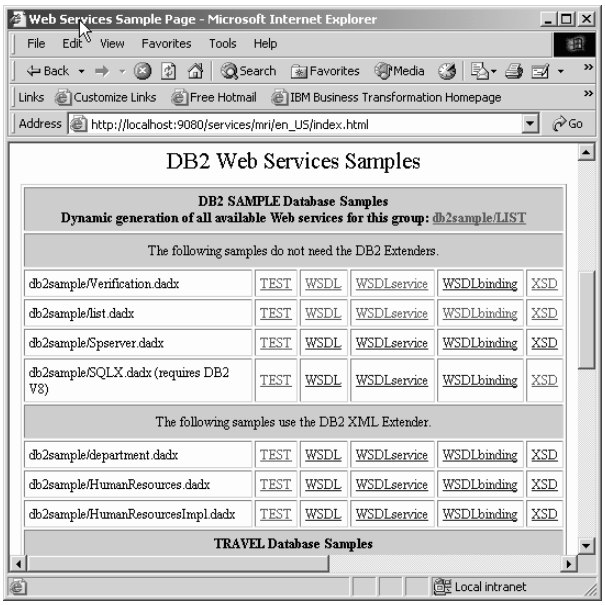


図 6. Web サービスのサンプルのページ - テスト・リンク

12. いずれかのリンクをクリックして、サンプル・サービスが稼働することを確かめてください。テスト・ページは、操作のツリー・ビュー、入力ビューおよび結果ビューで構成されます。サンプルのウェルカム・ページ内の TEST リンクからテスト・ページにアクセスするか、あるいはご使用のブラウザーで以下を入力します。<your-Web-server>:9080/<context_root_name>/<group_name>/<dadx file>/TEST

関連タスク:

- 48 ページの『WebSphere Application Server Version 5 以降 (Windows および UNIX 用) での WORF のインストールおよび移行』
- 70 ページの『web.xml ファイルおよび group.properties ファイルの定義』

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』
- 266 ページの『出力テキスト・ファイルでのエラーおよび警告の表示』
- 264 ページの『DADX 環境チェッカーの実行』

WebSphere Application Server Version 4.0.4 (z/OS または OS/390 用) での WORF の例のデプロイ

以下のステップは、z/OS または OS/390 プラットフォームで使用するために Web アプリケーションをデプロイします。また、WORF とその前提条件を正しくインストールおよび構成したことを検証することもできます。

手順:

WORF をインストールするには、以下のステップを行います。

1. *dxxworf.pax* をダウンロードして、 */u/USER/worf/* などの空のディレクトリーに解凍します。以下のコマンドを使用してこのファイルを解凍することができます。

```
pax -rvf dxxworf.pax
```

ファイルを解凍した後のこのディレクトリーの内容は次のとおりです。

- *readme.txt*
- *lib/apache-services.war* および *lib/axis-services.war* - WORF を使用する Web サービスが含まれるサンプル Web アプリケーション。

  注: 本書に含まれるコマンドおよび指示では、services.war または services.ear について言及しています。自分が実行することにした SOAP エンジン用の正しい SOAP ファイルを使用してください。たとえば、services.war ファイルについて説明している例の場合でも、Apache SOAP エンジンをインストールしているなら、使用するファイルは apache-services.war になります。
- *lib/worf.jar* - WORF ライブラリー。
- lib/worf-servlets.jar
- *schemas/* - DADX および NST XML ファイルの Extensible Markup Language (XML) スキーマ
- *tools/* - tools ディレクトリーには、DAD および DADX チェッカー・ツールが入っています。詳細は、『DADX 環境チェッカーのインストール』の項を参照してください。

2. J2EE サーバー・インスタンスのアプリケーション拡張ディレクトリーに *worf.jar* をコピーします。
3. J2EE サーバーを始動 (または再始動) します。

WORF の例をインストールしてデプロイするには、以下のステップを行います。

1. システム管理スクリプト記述のアプリケーション・プログラミング・インターフェース (API) を構成します。 OS/390 または z/OS システムでのスクリプト記述 API の構成に関する詳細は、「*IBM WebSphere Application Server V4 for z/OS and OS/390: インストールおよびカスタマイズ*」および「*IBM WebSphere Application Server V4.0.1 for z/OS and OS/390: システム管理スクリプト API*」を参照してください。
2. エンタープライズ・アーカイブ・ファイル (EAR) を準備します。 Java 2 Enterprise Edition (J2EE) アプリケーションを配信するのに、この EAR ファイルを使用します。これは、Web アーカイブ・ファイル (WAR) と Java アーカイブ・ファイル (JAR) で構成されます。
   a. UNIX システム・サービス (USS) では、書き込み可能な一時ディレクトリーに apache-services.war あるいは axis-services.war をコピーしてから、現在のディレクトリーをそのロケーションに変更します。
   b. USS コマンド行から次のようなコマンドを入力します (すべてを 1 行に収めます)。

      ```
      390fy -op "" -context_root "/services"
            -display_name "ServicesApp" services.war
      ```

      このコマンドにより、 *services.ear* という名前の初期 EAR ファイルが現在のディレクトリーに作成されます。この EAR ファイルは、『/services』 というコンテキスト・ルートと 『ServicesApp』 という表示名をもっています。コンテキスト・ルートは、各自のアプリケーションに WebSphere

Application Server をつなぐ URL の一部をなします。表示名は、 Systems Management End User Interface (SM/EUI) および USS において各自のアプリケーションを識別するストリングです。コンテキスト・ルートと表示名は、各自が選んだ任意の名前に編集することができます。

c. services.ear の Java Naming and Directory Interface (JNDI) 名マッピングを解決します。その解決には、 USS コマンド行から次のようなコマンドを発行します (すべてを 1 行に収めます)。

```
390fy -JNDIejbp "/<Sysplex>/<J2EE Server>"
         -op "_resolved" services.ear
```

上記の例で使われている用語は、次のように定義されています。

<J2EE Server>
アプリケーションのデプロイ先の J2EE サーバーの名前。

<Sysplex>
J2EE サーバーが置かれている Sysplex の名前。

このコマンドは、*services_resolved.ear* という名前の付いた新しいファイルを現在のディレクトリーに作成します。

3. アプリケーションをデプロイします。

注: このステップでコマンドを実行するには、 WebSphere のシステム管理の管理者として登録されているユーザー ID で USS にログオンしている必要があります。

a. WebSphere Application Server サンプルのディレクトリー (*<WAS_Home>/samples/smapi/*) から、作成したばかりの EAR ファイルを収めた一時ディレクトリーに以下のサンプル・ファイルをコピーします。

- *inputcreateconversation.xml*
- *inputprocessearfile.xml*
- *inputcommitconversation.xml*

b. 環境変数 DEFAULT_CLIENT_XML_PATH を、EAR ファイルの入っている一時ディレクトリーに設定します。

c. ファイル *inputcreateconversation.xml* を編集して、会話名と、必要があればその名前の説明を指定します。会話名と説明は、各自が選んだ任意のテキストでかまいません。ただし、その会話名は、次のステップで入力ファイルを処理するときも同じ名前にしなければなりません。以下に、会話名と説明の例を示してあります。

```
<inputcreateconversation conversationname="WORFSamples"
            conversationdescription="WORF Sample Test" />
```

ファイル *inputcreateconversation.xml* を保管します。

d. USS コマンド行から次のコマンドを入力します (すべてを 1 行に収めます)。

```
CB390CFG -action createconversation
   -xmlinput inputcreateconversation.xml
   -output createconv.out
```

このコマンドは、ファイル *createconv.out* をシステムの一時ディレクトリーに作成しますが、これには操作結果が入っています。このファイルが必要なのは、アプリケーションが正常にデプロイメントされたかどうかを確かめる場合だけです。

e. ファイル *inputprocessearfile.xml* を編集します。ターゲットの J2EE サーバーと、デプロイする EAR ファイルを指定します。以下は、J2EE サーバーと EAR ファイルの指定の例です。

```
<inputprocessearfile conversationname="WORFSamples"
                     j2eeservername="BBOASR2"
earfilename="/tmp/worfsamp/services_resolved.ear"
                     processingmode="standard" />
```

ファイル *inputprocessearfile.xml* を保管します。

f. USS コマンド行から次のコマンドを入力します (すべてを 1 行に収めます)。

```
CB390CFG -action processearfile
    -xmlinput inputprocessearfile.xml
    -output processear.out
```

このコマンドは、ファイル *processear.out* をシステムの一時ディレクトリーに作成しますが、これには操作結果が入っています。このファイルが必要なのは、アプリケーションが正常にデプロイメントされたかどうかを確かめる場合だけです。

g. ファイル *inputcommitconversation.xml* を編集します。前のステップで使用した会話名を指定します。以下に例を示します。

```
<inputcommitconversation conversationname="WORFSamples" />
```

ファイル *inputcommitconversation.xml* を保管します。

h. USS コマンド行から次のコマンドを入力します (すべてを 1 行に収めます)。

```
CB390CFG -action commitconversation
  -xmlinput inputcommitconversation.xml
  -output commitconv.out
```

このコマンドは、ファイル *commitconv.out* をシステムの一時ディレクトリーに作成しますが、これには操作結果が入っています。このファイルが必要なのは、アプリケーションが正常にデプロイメントされたかどうかを確かめる場合だけです。

4. Web サーバーをセットアップします。

a. ステップ 2 (54 ページ) で命名したコンテキスト・ルートを J2EE サーバーで使用できることを確認します。サーバーのファイル *webcontainer.conf* で、少なくとも 1 つのホストが次の例中の指定をもっていることを確認します。

```
host.<host_alias>.contextroots=/somewebapp,/services
```

任意のコンテキスト・ルートを受け入れるには、次のようなテキスト行を使用します。

```
host.<host_alias>.contextroots=/*
```

注: このやり方でコンテキスト・ルートを指定すると、今後アプリケーションをデプロイするときにこのステップをスキップすることができます。

b. Web サーバーのプラグインを使って J2EE サーバーにアクセスする場合、Web サーバーのファイル *httpd.conf* に Service ステートメントを追加します。このステートメントは、各自がデプロイしたアプリケーションのコンテキスト・ルートを指定するステートメントです。たとえば、ステップ 2 (54 ページ) でコンテキスト・ルートとして「/services」を指定した場合、新しい Service ステートメントは次の例のようになります。

```
Service  /services/*
<WAS_Home>/WebServerPlugIn/bin/was400plugin.so:service_exit
```

<WAS_Home> は、WebSphere Application Server がインストールされているディレクトリーです。

注: Service ステートメントは全体が 1 行に収まっていなければなりません。

c. Web サーバーを再始動します。

5. WORF 構成を検査します。

a. サンプルの WAR ファイルに組み込まれている WORF Web サービスのサンプル・ページにアクセスします。ステップ 2 (54 ページ) のコンテキスト・ルートを「/services」に設定する場合、以下の URL を入力します。

```
http://<hostname>/services/
```

図 7. WORF サンプル・ページ

b. 図 7 の**「インストール検査 (Installation Verification)」** というタイトルのサンプルの最初のセクションには、 1 つの DADX ファイル *ivt.dadx* が示されています。**TEST** リンクをクリックします。 WORF の組み込みテスト機能がオープンします。

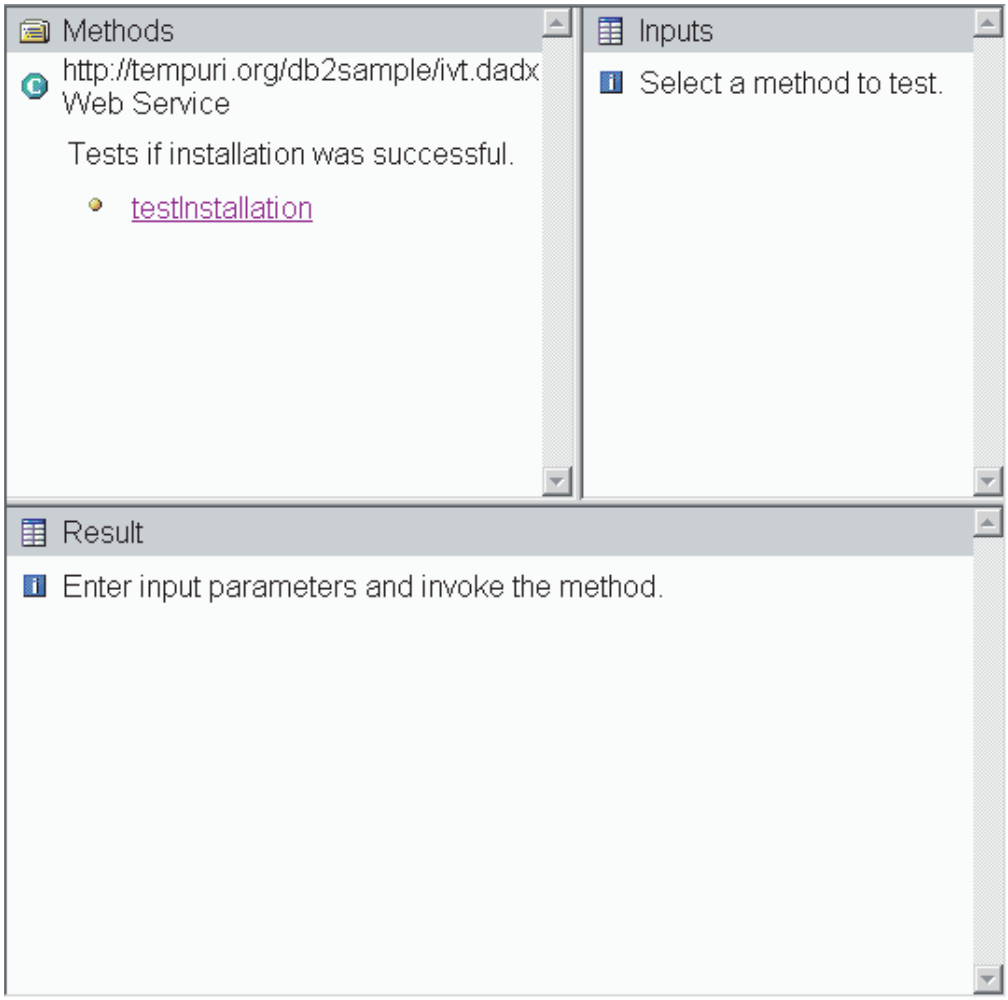


図 8. WORF テスト機能

c. 図 8 の WORF テスト機能から、「testInstallation」操作を選択します。**「呼び出し (Invoke)」**をクリックします。

d. XML 文書がウィンドウの下のフレームに表示されます。次の例のように、文書内に当日の現在の時刻が示されているのを検証します。

```
<CURTIME>14:38:26.000Z</CURTIME>
```

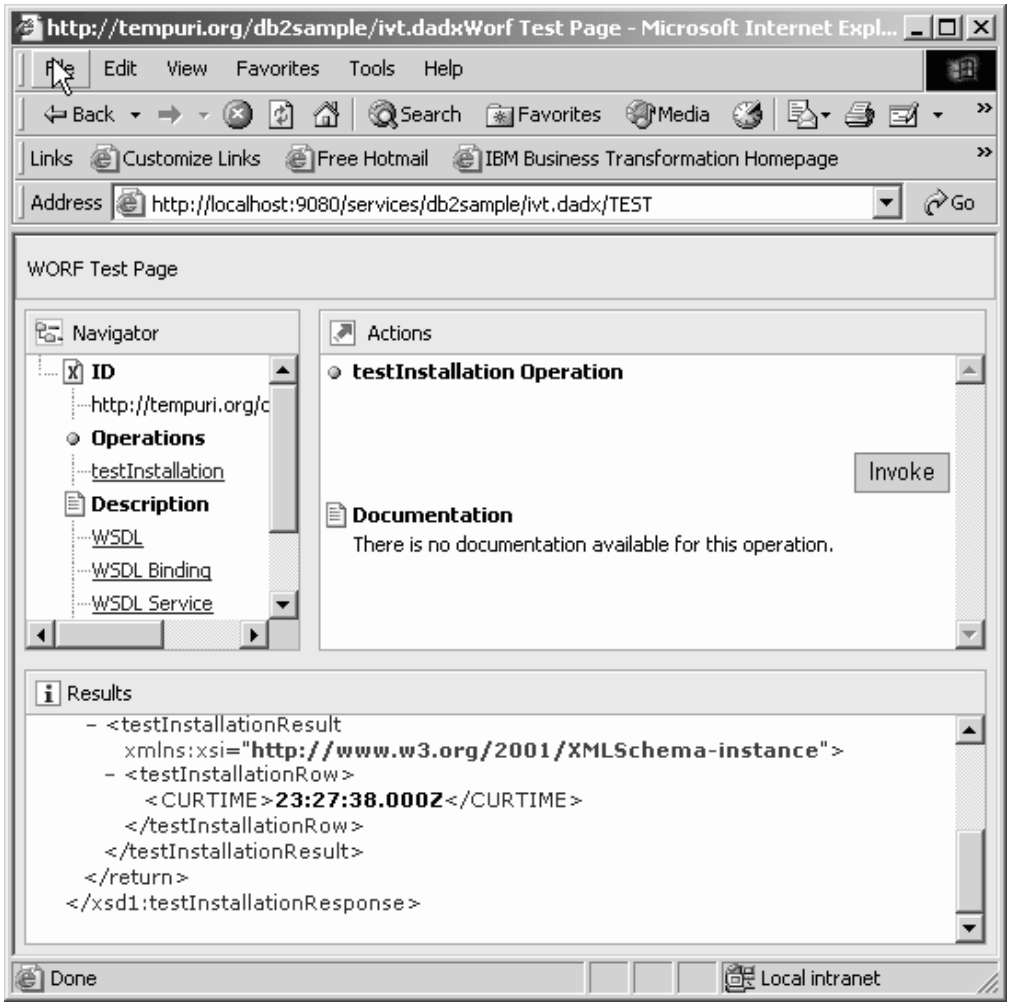


図 9. WORF テスト結果の予想出力

e. テストに失敗した場合、構成においてなんらかの不具合があります。ソフトウェア要件を正しくインストールしたことを確認してください。また、WebSphere Application Server の構成を完了したことと、 DB2 Universal Database へのアクセスの許可を受けていることも確認してください。

6. 例を準備して実行します。

   a. サンプル・ページ (ステップ 5a (57 ページ) を参照) には、 *services.war* アプリケーションに付属している DADX サンプルがあります。サンプルの実行の前に、各カテゴリー別に記載されているセットアップ手順を行ってください。

   b. 各サンプルごとに **TEST** リンクを選択して、個々の Web サービスを実行します。 **WSDL**、**WSDLservice**、**WSDLbinding**、および **XSD** リンクを選択して、それぞれの例を実行します。

アプリケーションのデプロイを完了したら、アプリケーションを変更することができます。また、デプロイメント用にさらに別の WAR ファイルを作成することもできます。 WAR ファイルを作成し終わったら、ステップ 2 (54 ページ)、ステップ 3 (55 ページ)、およびステップ 4 (56 ページ) に従って Web アプリケーションをデプロイします。

関連概念:

- 40 ページの『Web サービス・プロセスの概要』

関連タスク:

- 65 ページの『iSeries での WORF の例のインストールおよびデプロイ』
- 63 ページの『Apache Jakarta Tomcat での WORF の例のインストールおよびデプロイ』

関連資料:

- 263 ページの『DADX 環境チェッカーのインストール』

UNIX および Windows での Apache Jakarta Tomcat 用の Web サービス・プロバイダーの構成

Apache Jakarta Tomcat 上で Web サービスを実行できます。この後の項では、そのような Web サービスについて説明します。

UNIX および Windows での Apache Jakarta Tomcat 用の Web サービス・プロバイダー・ソフトウェアのインストール

必要なソフトウェアのインストールが完了したことを確認してください。該当する資料を参考に、各自のプラットフォームでのインストールを検証してください。

前提条件:

XML とリレーショナル・データの間の拡張マッピングの制御には、 DB2 XML エクステンダーが必要です。 DB2 SAMPLE データベースを作成して、インストールを検査してください。 WORF には、 DB2 Universal Database バージョン 8 におけるデフォルトである Java Database Connectivity (JDBC) 2.0 が必要です。

手順:

WORF 環境を準備する手順は次のとおりです。

1. DB2 を停止します。
2. DB2 Universal Database バージョン 8 を実行していない場合は、JDBC 2.0 を選択します。 Windows 環境で DB2 を C:¥SQLLIB¥ にインストールしていると想定した場合、 *C:¥SQLLIB¥java12¥usejdbc2.bat* を実行します。
3. DB2 を再始動します。
4. 以下のインターネット・ソフトウェアをインストールします。
   - Apache のソフトウェア:
     - http://jakarta.apache.org/site/binindex.html にある Apache Jakarta Tomcat バージョン 4.0.6 またはそれ以降のバイナリー。 (Apache Jakarta Tomcat バージョン 4 標準は、該当する Xerces パーサーに付属しています。旧バージョンの場合、 XML パーサーとして使用するには CLASSPATH に Xerces パーサーを追加する必要があります。)
     - http://xml.apache.org/soap にある Apache SOAP 2.3 またはそれ以降のバイナリー
     - http://www.apache.org/ にある Apache Axis 1.2

- http://xml.apache.org/ にある Xerces 1.4.4
- Sun (http://java.sun.com/products) のソフトウェア:
  - JavaMail 1.2
  - JavaBeans Activation Framework (JAF) 1.0 1
  - j2ee.jar バージョン 1.3 以降。
  - qname.jar
- *wsdl4j.jar*。このファイルは http://oss.software.ibm.com/developerworks/projects/wsdl4jからダウンロードできます。

関連タスク:

- 43 ページの『OS/390 および z/OS 用の Web サービス・プロバイダー・ソフトウェア要件』

関連資料:

- 42 ページの『Web サービス・プロバイダー・ソフトウェアの要件 (UNIX および Windows)』

Apache Jakarta Tomcat での WORF のインストールまたは移行

手順:

WORF を旧バージョンからバージョン 8.2 に移行するには、『移行のサマリー』のセクションを参照してください。 Apache Jakarta Tomcat で WORF バージョン 8.2 をインストールするには、以下のステップを行います。

- Apache SOAP または Apache Axis を使って WORF を実行するには、以下の JAR ファイルをアプリケーション・サーバーのクラス・パスに追加します。
  - *soap.jar (Apache SOAP エンジンの場合) または axis.jar (Apache axis エンジンの場合)*
  - *xerces.jar* (または Java XML パーサーの jar ファイル)
  - *mail.jar*
  - *activation.jar*
  - *worf.jar*
  - j2ee.jar バージョン 1.3 以降
  - qname.jar
  - *wsdl4j.jar*。このファイルは http://oss.software.ibm.com/developerworks/projects/wsdl4jからダウンロードできます。
  - *jaxrpc.jar*
  - *log4j-1.2.8.jar*
  - *commons-logging.jar*
  - *commons-logging-api.jar*
  - *commons-discovery.jar*
  - *db2java.zip*、または *jcc.jar* (またはデータベース・サーバーの JDBC インプリメンテーションの jar ファイル)。ドライバー・クラスの名前は使用するドライ

バー・パッケージによって異なります。使用するドライバー・パッケージは、group.properties ファイルで修正できます。

- 62 ページの表 4 に一覧で示されているファイルを修正します。行う修正は、Apache Jakarta Tomcat のバージョンとプラットフォームによって異なります。上記の各 jar ファイルごとに行を 1 つずつ追加します。 <jarfile> を jar ファイルの実際のロケーションに置き換えます。統合開発環境で Apache Jakarta Tomcat を実行する場合、 Tomcat の始動に使用する CLASSPATH 内にそれらの jar ファイルがすべてあることを確認します。修正する必要があるファイルは、すべて Apache Jakarta Tomcat を起動するディレクトリーにあります。 *app_server/bin* ディレクトリーにある、startup.bat または shutdown.bat、または startup.sh または shutdown.sh を使用してサーバーを起動または停止します。

表 4. クラス・パスの指定

プラットフォーム	サーバー・ソフトウェア	変更するファイル	追加するコマンド
UNIX	Apache Jakarta Tomcat 3.2.x	¥bin¥tomcat.sh (「export CLASSPATH」の前に)	CLASSPATH = $CLASSPATH: <jarfile>
	Apache Jakarta Tomcat 3.3.x	¥bin¥tomcat.sh (「export CLASSPATH」の前に)	CLASSPATH = $CLASSPATH: <jarfile>
	Apache Jakarta Tomcat 4.x	¥bin¥setclasspath.sh (「export CLASSPATH」の前に)	CLASSPATH = $CLASSPATH: <jarfile>
Windows	Apache Jakarta Tomcat 3.2.x	¥bin¥tomcat.bat (:setClasspath セクション)	set CP = %CP%; <jarfile>
	Apache Jakarta Tomcat 3.3.x	¥bin¥tomcat.bat	set CLASSPATH = %CLASSPATH%; <jarfile>
	Apache Jakarta Tomcat 4.x	¥bin¥setclasspath.bat	set CLASSPATH = %CLASSPATH%; <jarfile>

- WebSphere 5.0.2 よりも前のリリースの WebSphere サーバーの場合、http://java.sun.com/xml/downloads/saaj.html から saaj.jar という名前のファイルをダウンロードする必要があります。 saaj.jar を *C:¥WebSphere¥AppServer¥lib* にコピーします。

移行のサマリー

WORF を旧バージョンからバージョン 8.2 に移行するには、以下を行います。

1. バージョン 8.2 の dxxworf.zip の *lib¥worf.jar* を *C:¥WebSphere¥AppServer¥lib* にコピーして、worf.jar ファイルを置き換えます。 dxxworf.zip は次のパスに置かれています。 *<DB2 UDB をインストールした場所 >¥samples¥java¥Websphere¥dxxworf.zip*。
2. デプロイした各アプリケーションの worf ディレクトリーに入っている JSP ファイルを、 apache-services.war または axis-services.war の worf ディレクトリーのファイルで置き換えます。次に、アプリケーションを再デプロイします。

関連タスク:

- 65 ページの『iSeries での WORF の例のインストールおよびデプロイ』
- 63 ページの『Apache Jakarta Tomcat での WORF の例のインストールおよびデプロイ』

- 48 ページの『WebSphere Application Server Version 5 以降 (Windows および UNIX 用) での WORF のインストールおよび移行』

Apache Jakarta Tomcat での WORF の例のインストールおよびデプロイ

手順:

WORF の例をインストールするには、次のようなタスクを行います。

1. ファイル *apache-services.war* または *axis-services.war* を *tomcat¥webapps* ディレクトリー (インストールされている SOAP エンジンによって異なる) に unjar します。

   apache-services.war または *axis-services.war* ファイルがインストール済みであれば、以下のタスクを実行します。

   a. Apache Jakarta Tomcat を停止します。

   b. *webapps* の下の *services* サブディレクトリーとそのすべての内容を削除します。

      注: このアクションをとると、「services」Web アプリケーションに以前デプロイした Web サービスはすべて失われるので、それでもよいかどうかを確認します。

   c. Apache Jakarta Tomcat を再始動します。

2. Apache Jakarta Tomcat を停止してから始動します (前のステップで *services* ディレクトリーを削除したのでないかぎり)。

   次のようにサービス・コンテキストを始動します。

   ```
   ContextManager: Adding context Ctx(¥services)
   ```

3. 以下の URL を入力して、インストール結果を検査します。以下で *8080* と指定してあるポート番号は、各自の現在のマシンによって異なります。

   ```
   http://localhost:8080/services
   ```

   個々のポート・アドレスは、環境に応じて変わることがあります。 52 ページの図 5 のようなページが表示されるはずです。 Web サービスのサンプル・ページの詳細は、『Web サービスのテスト』の項を参照してください。

4. ユーザー ID とパスワードをはじめとするデータベースの設定が正しいかどうかを、 group.properties ファイルで検証します。各自のシステムで verification.dadx をテストします (動的テスト・ページおよび WSDL)。

5. Extensible Markup Language (XML) 文書を表示するには、 Internet Explorer バージョン 5 またはテキスト・エディターを使用します。

6. システムにおけるサービス・コンテキストでデプロイ済みの SOAP サービスを一覧で表示します。実行されたテストごとに WORF は自動的にサービスをデプロイします。 Web サービスのサンプル・ページの SOAP 管理リンクをクリックします。

関連概念:

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』

関連タスク:

- 127 ページの『Web サービスのテスト』
- 60 ページの『UNIX および Windows での Apache Jakarta Tomcat 用の Web サービス・プロバイダー・ソフトウェアのインストール』
- 44 ページの『Web サービス・プロバイダー・ソフトウェア要件のインストール』

iSeries での Apache Jakarta Tomcat 用の Web サービス・プロバイダー・ソフトウェア要件のインストール

前提条件:

必要なソフトウェアのインストールが完了したことを確認してください。 XML とリレーショナル・データの間の拡張マッピングの制御には、 DB2 XML エクステンダーが必要です。 DB2 Universal Database の XML エクステンダーを使用するには、同製品がインストールされていることを確認します。システムに DB2 Universal Database の XML エクステンダーがインストールされていることを確認するには、CL コマンド **GO LICPGM** を発行します。 DB2 Universal Database for iSeries V5R2 の場合に DB2 Universal Database の XML エクステンダーがあれば、 **GO LICPGM** コマンドの実行によって次のような項目が表示されます。

- 5722DE1 *COMPATIBLE DB2 UDB Extenders
- 5722DE1 *COMPATIBLE DB2 UDB Text Extender
- 5722DE1 *COMPATIBLE DB2 UDB XML Extender
- 5722DE1 *COMPATIBLE Text Search Engine

CL コマンド **CALL PGM(QDBXM/QZXMADM) PARM(enable_db LOCALRDB)** を使って、DB2 Universal Database の XML エクステンダーを使用可能にします。 LOCALRDB は、リレーショナル・データベース・ディレクトリー内の *LOCAL データベース名です。リレーショナル・データベースの項目を処理するには、 CL コマンド **WRKRDBDIRE** を発行します。サンプル・ファイル内の文書タイプ定義文書 (DTD) を使用する場合、スクリプト *setup-dxx.cmd* を実行します。

WORF には、 DB2 Universal Database バージョン 8 におけるデフォルトである Java Database Connectivity (JDBC) 2.0 が必要です。

手順:

WORF 環境を準備する手順は次のとおりです。

1. DB2 Universal Database バージョン 8 を実行していない場合は、JDBC 2.0 を選択します。 Windows 環境で DB2 を C:¥SQLLIB¥ にインストールしていると想定した場合、 *C:¥SQLLIB¥java12¥usejdbc2.bat* を実行します。
2. 以下のインターネット・ソフトウェアをインストールします。
   - Apache のソフトウェア:
     - http://jakarta.apache.org/site/binindex.html にある Apache Jakarta Tomcat バージョン 4.0.3 またはそれ以降のバイナリー。 (Apache Jakarta Tomcat バージョン 4 標準は、該当する Xerces パーサーに付属しています。旧バージ

ョンの場合、 XML パーサーとして使用するには CLASSPATH に Xerces パーサーを追加する必要があります。)

- http://xml.apache.org/ にある Xerces 1.4.4

- Sun (http://java.sun.com/products) のソフトウェア:
  - JavaMail 1.2
  - JavaBeans Activation Framework (JAF) 1.0 1
  - j2ee.jar バージョン 1.3 以降。
  - qname.jar
- *wsdl4j.jar*。このファイルは http://oss.software.ibm.com/developerworks/projects/wsdl4j からダウンロードできます。

関連概念:

- 34 ページの『iSeries での Web サービス・プロバイダーの使用』

関連タスク:

- 73 ページの『iSeries プラットフォームでの web.xml ファイルおよび group.properties ファイルの定義』
- 65 ページの『iSeries での WORF の例のインストールおよびデプロイ』
- 46 ページの『iSeries での Web サービス・プロバイダー・ソフトウェア要件のインストール』

iSeries での WORF の例のインストールおよびデプロイ

手順:

WORF の例をインストールするには、次のようなタスクを行います。

1. worf.jar ファイルが以下のロケーションにあることを確かめます。

   ```
   /QIBM/UserData/WebASAEs4/worf/lib/app
   ```

   ただし、**WebASAEs4** は WebSphere のバージョン、また **worf** は WebSphere インスタンスの名前です。

2. ファイル *runtime.zip* がディレクトリー /QIBM/UserData/java400/ext 内にない場合、以下のコマンドを実行します。

   a. >qsh

   b.
   ```
   >ln -s  /QIBM/ProdData/OS400/Java400/ext/runtime.zip
            /QIBM/UserData/Java400/ext/runtime.zip
   ```

3. *services.war* を *tomcat¥webapps* ディレクトリーにコピーします。

   services.war ファイルがインストール済みであれば、以下のタスクを実行します。

   a. Apache Jakarta Tomcat を停止します。

   b. *webapps* の下の *services* サブディレクトリーとそのすべての内容を削除します。

注意:
このアクションをとると、「services」Web アプリケーションに以前デプロイした Web サービスはすべて失われるので、それでもよいかどうかを確認します。

c. Apache Jakarta Tomcat を再始動します。

4. Apache Jakarta Tomcat を停止してから始動します (前のステップで *services* ディレクトリーを削除したのでないかぎり)。

   次のようにサービス・コンテキストを始動します。

   ```
   ContextManager: Adding context Ctx(¥services)
   ```

5. http://<system>:<port>/services のテスト・ページにアクセスして、サンプル・アプリケーション内の例を呼び出します。
   - パラメーターを付けないで、
     http://<system>:<port>/services/travel/ZipCodes.dadx/findAll のサンプルを呼び出します。
   - パラメーターを使用して、以下のサンプルを呼び出します。

     ```
     http://<system>:<port>/services/
         travel/ZipCodes.dadx/findCityByZipCode?zipcode=55901
     ```

6. group.properties 内のデータベース設定と、特にユーザー ID とパスワードが正しいことを検証します。ユーザー ID の値を使用しない場合、Web サービスのコードは QEJBSVR のもとで稼働します。したがって、アクセス予定のすべてのデータベース・オブジェクトに対してこのプロファイルに許可を与えてください。各自のシステムで verification.dadx をテストします (動的テスト・ページおよび WSDL)。

関連概念:

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』

関連タスク:

- 64 ページの『iSeries での Apache Jakarta Tomcat 用の Web サービス・プロバイダー・ソフトウェア要件のインストール』
- 44 ページの『Web サービス・プロバイダー・ソフトウェア要件のインストール』

Web サービス・プロバイダーの管理とトラブルシューティング

この章では、Web サービス・プロバイダーのパフォーマンス上のいくつかの提案と、トラブルシューティング・ガイドについて説明しています。

接続プールを使用したパフォーマンスの改善

リソースがデータベースにアクセスを試みるたびに、そのリソースはデータベースに接続しなければなりません。データベースに接続するために、リソースは接続を作成し、これを維持し、不要になったらこれを解放しなければなりません。 Web ベースのアプリケーションの場合、Web ユーザーがより高い頻度で接続と切断を繰り返すため、必要なデータベース・リソースは大量に及ぶことがあります。データベース接続のプールを作成および保守する助けとして、IBM WebSphere Application Server を使用できます。これらのデータベース接続は、アプリケーション・サーバ

一上の複数のアプリケーションで共有できます。これはリソースの問題を解消する上で役立ちます。接続プールを設けることで、接続のオーバーヘッドが複数のユーザー要求に分散されます。結果として、将来の要求に備えてリソースを保持しておいて、パフォーマンスを改善できます。ユニークなデータ・ソースごとに 1 つずつプールを構成できます。接続プールに関しては「WebSphere ハンドブック」の第 10 章に詳しく述べられています。

前提条件:

1. 使用したい JDBC プロバイダーが存在しない場合は、これを作成します。
2. データ・ソースを作成します。

手順:

インストールされたアプリケーションは、JDBC プロバイダーを使用してデータベースのデータにアクセスします。 WebSphere Application Server バージョン 5 から、JDBC プロバイダー内の特定のデータ・ソースの接続プールのパラメーターを調整するには、以下のステップを実行します。

1. データ・ソースのパラメーターを構成します。
2. 図 10 のように接続プール情報を更新します。 WebSphere Application Server は、DB2 Universal Database への接続が容易になるよう、Java Naming サービス (JNDI) を提供しています。プールは、同じデータ・ソースに接続するすべてのアプリケーションにより共有されます。

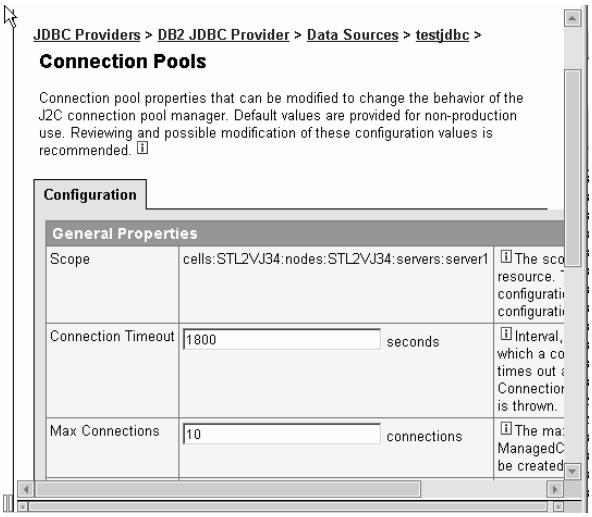


図 10. 接続プール・パラメーターの調整

3. グループ・サブディレクトリー内の *group.properties* ファイルを編集して、次のようなテキスト行を追加します。

   ```
   initialContextFactory=<your context factory>
   datasourceJNDI=<your DataSource>
   ```

   以下に例を示します。

   ```
   initialContextFactory=com.ibm.websphere.naming.WsnInitialContextFactory
   datasourceJNDI=jdbc/sampleDataSource
   ```

4. group.properties ファイルになんらかの変更を加えた場合、それを有効にするために Web アプリケーションを再始動します。

関連タスク:

- 70 ページの『web.xml ファイルおよび group.properties ファイルの定義』

Web サービスのトラブルシューティング

表 5 は、WebSphere Application Server 5.1 上で WORF を使用するときに起きる可能性のある問題を示しています。表には、推奨されている解決方法が提供されています。

表 5. エラーおよび解決方法

問題	解決策
Error 500: Server caught unhandled exception from servlet [isd_demos]: org.apache.soap.rpc.SOAPContext: method setClassLoader (java¥lang¥ClassLoader;) not found	SOAP 2.2 またはそれ以降 (soap.jar) が欠落しています。
Internet Explorer で Web サービス・テスト・ページから**「呼び出し (Invoke)」**ボタンをクリックすると、「ページが見つかりません」エラーになる。	より役立つメッセージを表示するには、Netscape を使用して問題をデバッグします。あるいは、以下の手順を行って Internet Explorer 環境を編集します。 1. Internet Explorer メニュー・バーから、**「ツール」**メニューを開きます。 2. メニューから**「インターネット オプション」**を選択して、「インターネット オプション」ウィンドウを開きます。 3. 「**詳細設定**」タブをクリックします。 4. **「HTTP エラー メッセージを簡易表示する」**の隣のチェック・ボックスを解除します。
Error 400: service 'http://tempuri.org /***/***.dadx' unknown	DADX ファイルからデプロイメント記述子を生成し、サービスを呼び出す前に Web アプリケーションを再始動しなければなりません。
Error 400: Unable to get DAD;unable to get Input Stream for: xxxxxx	指定した XML エクステンダー DAD ファイル (たとえば、 *.nst ファイルで指定された DAD) へアクセスできません。
Error 400: database connection error	• データベースが開始していません。 • DADX ファイルに示されるデータベース・オブジェクトが存在しません。 • JDBC ドライバーが見付かりません。
Error 400: unable to get input stream for /groups/xxx/yyy.dadx	DADX ファイルが、グループ・フォルダーに存在しないか、アクセスできません。
Error 404: File not found: aaaa/abc.dadx	サーブレット・マッピング 'aaaa' が、web.xml ファイルに存在しません。
ブランク・ページの結果	バージョン 5.0.2 より前のバージョンの WebSphere Application Server を使用している場合、 `jaas.jar` がない可能性があります。サーバー・ディレクトリーの SystemErr.log をオープンし、他の JAR ファイルがあるかどうか確認してください。

Web サービス・プロバイダーから実行時イベントおよび診断についての情報を入手して、デプロイ後の Web サービスをトラブルシューティングするために、アプリケーションを実行している Web アプリケーション・サーバーのトレース機能を使用できます。 Web アプリケーション・サーバーから受け取るトレース情報には、メッセージおよびイベント・アクティビティーが含まれます。トレースが使用可能でない場合でも、アプリケーション・サーバーのエラー・ログにはエラーが記入されています。 Web サービス・プロバイダー・イベントのトレース方法の詳細は、『Web サービス・プロバイダーのトレース』を参照してください。

関連概念:

- 157 ページの『Web サービス・プロバイダーのトレース』

関連タスク:

- 152 ページの『デプロイメント記述子の生成』

関連資料:

- 42 ページの『Web サービス・プロバイダー・ソフトウェアの要件 (UNIX および Windows)』

Web サービス・プロバイダーを使用するアプリケーションの開発

以下のセクションでは、Web サービス・プロバイダーの使用に関する概要と詳細について説明しています。

Web サービスのグループの定義

グループは、共通の構成オプションを共用する Web サービスのコンテナーです。構成オプションは、以下のものである場合があります。

- データベース構成
- ネーム・スペース・セットアップ
- メッセージ・エンコード・セットアップ

すべての DADX Web サービス・グループのリソースは、*groups* ディレクトリーに入ります。 WORF Web アプリケーションは、アプリケーション構成中にこのディレクトリーを作成します。このディレクトリーは、Web アプリケーションの基本ディレクトリーの *WEB-INF¥classes¥groups¥* サブディレクトリーにあります。

DADX ファイルは、Web サービスの記述を含んでいます。 WORF には Web サービスのインプリメンテーションが含まれるので、Java™ クラスと類似しています。*classes* ディレクトリーは Web アプリケーションの Java CLASSPATH の一部です。これは、Java クラス・ローダーで自分の DADX ファイルをロードできることを意味します。

groups ディレクトリー内で、 WORF は DADX Web サービスのグループを、そのサーブレット・インスタンスと同じ名前のディレクトリーに保管します。アプリケーション・サーバーは、特定の URL に対して呼び出す正しいサーブレットを検索します。これは web.xml ファイルに基づいて行われます。

WORF により使用される別のリソースは、group.imports ファイルです。これはオプションのリソースであり、生成された WSDL を使用するさまざまな Web サービス・コンシューマーまたはツールが、使用するスキーマを検索する上で助けとなります。 group.imports ファイルが存在する場合には、 WSDL は group.imports ファイルの内容およびエレメントの有効範囲を基にしてインポート・エレメントを生成します。 group.imports ファイルが存在しない場合、WSDL には、非動的照会サービスのためのインポート・エレメントは生成されません。動的照会サービスのため、WSDL には、db2WebRowSet.xsd に入っているいくつかのデータ・タイプが含まれています。 db2WebRowSet.xsd の場所を定義するための group.imports がない場合、WORF は、このスキーマ・ファイルはデフォルトの位置にあるものと想定します。たとえば、次の例のようになります。

```
http://<server>:<port>/<contextRoot>/db2WebRowSet.xsd
```

DB2® Web サービスに関連する例では、 WORF アプリケーションは DADX ファイルを *WEB-INF¥classes¥groups¥dxx_sample¥* ディレクトリー、*WEB-INF¥classes¥groups¥dxx_sales_db¥* ディレクトリー、および *WEB-INF¥classes¥groups¥dxx_travel¥* ディレクトリーに保管します。

関連タスク:

- 155 ページの『Web アーカイブ・ファイルの準備と作成』
- 76 ページの『group.properties ファイルのカスタマイズ』
- 152 ページの『デプロイメント記述子の生成』

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』

web.xml ファイルおよび group.properties ファイルの定義

手順:

DADX Web サービスの新規グループを定義するには、以下のステップを完了します。 (ファイルの作成には、WebSphere Studio バージョン 5 を使用することもできます。詳細については、WebSphere Studio インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/wsphelp/index.jsp) を参照してください。新しいバージョンの WORF に移行する際には、web.xml ファイルと group.properties ファイルの設定値を、確実に自分の環境に合ったものにする必要があります。

1. DADX グループに、自分のアプリケーションを反映するグループ名を選択します。この説明では `myapp_group` という名前を使用します。
2. *WEB-INF* ディレクトリーの *web.xml* ファイルを編集して、グループ名を定義します。 Java 2 Enterprise Edition バージョン 1.3 に対応した WebSphere バージョン 5 のデータ・ソース (WebSphere Studio あるいは WebSphere Application Server) を使用する場合には、必ず web.xml ファイル中で web-app_2.2.dtd を web-app_2.3.dtd に変更します。

   同じ *web.xml* ファイルに複数のグループ名を持つことができます。次の図は、*web.xml* ファイルの例を示しています。 servlet-mapping エレメントは**太字**で、下に定義されている値を取ります。

```
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
  <web-app>
  <servlet>
        <servlet-name>myapp_group</servlet-name>
        <servlet-class>com.ibm.etools.
          webservice.rt.dxx.servlet.
           DxxInvoker
        </servlet-class>
        <init-param id=InitParam_1076524994485>
          <param-name>faultListener</param-name>
          <param-value>
             org.apache.soap.server.DOMFaultListener
          </param-value>
        </init-param>
        <init-param id=InitParam_1076524994488>
          <param-name>soap-engine</param-name>
          <param-value>apache-soap</param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
  </servlet>
<servlet-mapping>
        <servlet-name>myyapp_group</servlet-name>
        <url-pattern>/myapp/*</url-pattern>
</servlet-mapping>
 <welcome-file-list>
       <welcome-file>index.html</welcome-file>
 </welcome-file-list>
</web-app>
```

図 11. web.xml

<servlet> セクションでは、グループに対応する新しいサーブレット・インスタンスを定義します。グループごとに少なくとも 1 つの <servlet> エレメントが存在していなければなりませんが、 1 つのグループが複数の <servlet-mapping> エレメントを持つことは可能です。 Java サーブレットの仕様については、http://java.sun.com/products/servlet/ を参照してください。この例では、<servlet-name> エレメントは myapp_group という名前のグループを定義します。

このファイルを更新する際、以下のエレメントの情報を提供します。

<servlet-name>

これは、<servlet> セクション、そして <servlet-mapping> セクションの子タグで、グループの名前が定義されます。サーブレット名は *groups* ディレクトリーの下にある有効なディレクトリー名でなければなりません。この名前を使用して、このグループの Web サービスの DADX リソースを保管します。たとえば、myapp_group は、<servlet> エレメントと <servlet-mapping> エレメントの両方で定義されています。

<servlet-mapping>

URL とグループのマッピングのために、<servlet-mapping> セクションが最低 1 つ必要になります。子タグ <servlet-name> ではグループ名を定義しますが、これはグループのディレクトリー名と同じでなくてはなりません。また、これは、<servlet-name> が <servlet> グループのものと同じでなければならないということを意味します。 <servlet-name> タグは、<servlet> タグと <servlet-mapping> タグの間のリンクです。

<url-pattern>

グループに関連した URL。 <servlet-mapping> エレメントは dxx_sales_db サーブレットを書式 */url_pattern/** の URL に関連付けます。 URL パターンは、WORF が正常に作動するために、この書式でなければなりません。たとえば、**/myapp/*** となります。この例では、サーブレット名は、myapp_group となっています。

<init-param>

このタグでは、使用する SOAP エンジンの名前を更新できます。SOAP エンジンを指定するには、<soap-engine> というパラメーターを使用します。使用するのが Apache SOAP であれば、パラメーターの値は *apache-soap* とします。また、使用するのが Apache Axis であれば、パラメーターの値は *apache-axis* とします。<soap-engine> パラメーターを指定しない場合、デフォルトの SOAP エンジン *apache-soap* を指定したことになります。

注: *web.xml* ファイルのデフォルト・エンコードは UTF-8 です。OS/390 または z/OS プラットフォーム上では、 *web.xml* ファイルを UNIX または Windows システムに送信することにより、このファイルを UTF-8 で更新します。ファイル転送プロトコル (FTP) バイナリー転送を使用することによって、このファイルを送信できます。次いでファイルを更新し、ファイルを元のシステムに戻します。

3. groups ディレクトリーから、前のステップで追加された <servlet-name> エレメントに指定されているグループの名前で、サブディレクトリーを作成します。サブディレクトリーには最終的にこのグループのリソースが含まれます。
4. group ディレクトリーで、*group.properties* ファイルを作成します。このファイルは DADX Web サービスの各グループのデータベース接続情報およびその他の共通属性を定義します。次に示すのは、新規グループの group.properties がどのようなものかを示す例です。

```
# myapp_group group properties
dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
dbURL=jdbc:db2:sample
userID=
password=
namespaceTable=myapp.nst
autoReload=true
reloadIntervalSeconds=5

for Informix, use the following database driver and URL:
dbDriver=com.informix.jdbc.IfxDriver
dbURL=jdbc¥:informix-sqli://::informixserver=
For OS/390 and z/OS, use the following database driver and URL:
dbDriver=COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver
dbURL=jdbc:db2os390:
```

図 12. group.properties の例

関連概念:

- 132 ページの『Web サービス記述言語』

- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』

関連タスク:

- 76 ページの『group.properties ファイルのカスタマイズ』

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』
- 268 ページの『web.xml ファイルのエラーのチェック』

iSeries プラットフォームでの web.xml ファイルおよび group.properties ファイルの定義

手順:

DADX Web サービスの新規グループを定義するには、以下のステップを完了します。

1. DADX グループに、自分のアプリケーションを反映するグループ名を選択します。この説明では `myapp_group` という名前を使用します。
2.
   - iSeries の場合、*WEB-INF* ディレクトリーで、 *web.xml* ファイルを編集して、グループ名を定義します。同じ *web.xml* ファイルに複数のグループ名を持つことができます。 Java 2 Enterprise Edition バージョン 1.3 に対応した WebSphere バージョン 5 のデータ・ソース (WebSphere Studio あるいは WebSphere Application Server) を使用する場合には、必ず web.xml ファイル中で web-app_2.2.dtd を web-app_2.3.dtd に変更します。

     サンプル・アプリケーションには、生成された Extensible Markup Language (XML) にスタイルシートを適用するサーブレットが組み込まれています。サーブレットのクラスおよびソースは次のディレクトリーにあります。

     ```
     /QIBM/UserData/WebASAEs4/worf/
     installedApps/servicesApp.ear/
     services.war/WEB-INF/classes
     ```

     サンプル・ファイルには、設定 serveServletsByClassnameEnabled=″true″ が含まれます。次のファイルを実行して、サーブレットを呼び出します。

     ```
     /QIBM/UserData/WebASAEs4/worf/
     installedApps/servicesApp.ear/
     services.war/WEB-INF/ibm-web-ext.xmi
     ```

     次のコンパイル・ステートメントを実行することにより、サーブレット内部 qshell を再コンパイルできます。

     ```
     javac -J-Djava.ext.dirs=/qibm/proddata/webasaes4/lib
             -d . SampleXSLTServlet.java
     ```

     次のファイルを実行して、サーブレットをインターネットから呼び出します。

     ```
     http://<system>:<port>/services/servlet/
     SampleXSLTServlet?XML=
     http://<system>:<port>/services/travel/ZipCodes.dadx/
        findAll&XSL=
     file:///home/zipcodes.xsl
     ```

上記の例は、*zipcodes.xsl* が /home ディレクトリーにあることを前提としています。このファイルはどこにでも置くことができます。このサーブレットの例を、XML Web サービスおよび Extensible Stylesheet Language (XSL) スタイル・シートの任意の組み合わせに使用できます。

次の図は、*web.xml* ファイルの例を示しています。 servlet-mapping エレメントは**太字**で、下に定義されている値を取ります。

```
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
<servlet>
        <servlet-name>myapp_group</servlet-name>
        <servlet-class>com.ibm.etools.
webservice.rt.dxx.servlet.DxxInvoker</servlet-class>
        <init-param>
          <param-name>faultListener</param-name>
          <param-value>
             org.apache.soap.server.DOMFaultListener
          </param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>myyapp_group</servlet-name>
        <url-pattern>/myapp/*</url-pattern>
    </servlet-mapping>
 <welcome-file-list>
       <welcome-file>index.html</welcome-file>
 </welcome-file-list>
</web-app>
```

図 13. web.xml

<servlet> セクションでは、グループに対応する新しいサーブレット・インスタンスを定義します。グループごとに少なくとも 1 つの <servlet> エレメントが存在していなければなりませんが、 1 つのグループが複数の <servlet-mapping> エレメントを持つことは可能です。 Java サーブレットの仕様については、http://java.sun.com/products/servlet/ を参照してください。この例では、<servlet-name> エレメントは `myapp_group` という名前のグループを定義します。

このファイルを更新する際、以下のエレメントの情報を提供します。

<servlet-name>
これは、<servlet> セクション、そして <servlet-mapping> セクションの子タグで、グループの名前が定義されます。サーブレット名は *groups* ディレクトリーの下にある有効なディレクトリー名でなければなりません。この名前を使用して、このグループの Web サービスの DADX リソースを保管します。たとえば、`myapp_group` は、<servlet> エレメントと <servlet-mapping> エレメントの両方で定義されています。

<servlet-mapping>
URL とグループのマッピングのために、<servlet-mapping> セクショ

ンが最低 1 つ必要になります。子タグ <servlet-name> ではグループ名を定義しますが、これはグループのディレクトリー名と同じでなくてはなりません。また、これは、<servlet-name> が <servlet> グループのものと同じでなければならないということを意味します。
<servlet-name> タグは、<servlet> タグと <servlet-mapping> タグの間のリンクです。

<url-pattern>
グループに関連した URL。 <servlet-mapping> エレメントは dxx_sales_db サーブレットを書式 */url_pattern/** の URL に関連付けます。 URL パターンは、WORF が正常に作動するために、この書式でなければなりません。たとえば、/myapp/* となります。この例では、サーブレット名は、myapp_group となっています。

<init-param>
このタグでは、使用する SOAP エンジンの名前を更新できます。
SOAP エンジンを指定するには、<soap-engine> というパラメーターを使用します。使用するのが Apache SOAP であれば、パラメーターの値は *apache-soap* とします。また、使用するのが Apache Axis であれば、パラメーターの値は *apache-axis* とします。 <soap-engine> パラメーターを指定しない場合、デフォルトの SOAP エンジン *apache-axis* を指定したことになります。

3. groups ディレクトリーから、前のステップで追加された <servlet-name> エレメントに指定されているグループの名前で、サブディレクトリーを作成します。サブディレクトリーにはこのグループのリソースが含まれます。
4. group ディレクトリーで、*group.properties* ファイルを作成します。このファイルは DADX Web サービスの各グループのデータベース接続情報およびその他の共通属性を定義します。次に示すのは、新規グループの group.properties がどのようなものかを示す例です。

```
# myapp_group group properties
dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
dbURL=jdbc:db2:*local/SAMPLE
userID=
password=
namespaceTable=myapp.nst
autoReload=true
reloadIntervalSeconds=5
```

75 ページの図 14 の例では、dbURL パラメーターで使用されている **SAMPLE** は使用したいデータベースまたはコレクションです。

図 14. group.properties の例

関連タスク:

- 76 ページの『group.properties ファイルのカスタマイズ』

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』

group.properties ファイルのカスタマイズ

group.properties は標準 Java プロパティー・ファイルです。 group.properties は、以下のパラメーターの少なくとも 1 つを指定して定義しなければなりません。

接続プーリングに group.properties を使用する場合
datasourceJNDI を指定した initialContextFactory パラメーターを使用して group.properties を定義する

通常の JDBC データベース接続に group.properties を使用する場合
dbDriver を指定した dbURL パラメーターを使用して group.properties を定義する

両方のタイプの接続を定義する場合、アプリケーションは DataSource を先に試行します。 WORF が DataSource を入手できない場合は、JDBC を試行します。プロパティーの完全セットは下にリストします。

手順:

group.properties ファイルを変更するには、環境に合わせて以下の定義を使用します。

データベース構成パラメーター

initialContextFactory
このパラメーターは `datasourceJNDI` と共に使用され、WebSphere 接続プールのために必要です。このパラメーターは、データベースの DataSource を探し出すために使用する JNDI 初期コンテキスト・ファクトリーの Java クラス名を指定します。このプロパティーと datasourceJNDI プロパティーが一緒になって、接続プールを使用可能にします。

datasourceJNDI
このパラメーターは `initialContextFactory` と共に使用され、WebSphere 接続プールのために必要です。このパラメーターは、データベースの DataSource の JNDI 名を指定します。これを `initialContextFactory` と一緒に使用すると、データベース接続の DataSource を定義します。このプロパティーは接続プーリングを使用可能にします。 DataSource または Java Database Connectivity (JDBC) 接続のいずれかを定義しなければなりません。

dbDriver
このパラメーターは、データベースに接続するための Java Database Connectivity (JDBC) ドライバーの Java クラス名を指定します。

dbURL
このパラメーターは dbDriver と共に使用され、データベースの JDBC URL を指定します。

userID
オプション。デフォルトは、WORF が実行されるときのユーザー ID で、データベースに接続するときに使用するのと同じユーザー ID である場合もあります。これは、データベースのユーザー ID を指定します。

password

オプションですが、ユーザー ID と一緒に使用します。これは、データベースのパスワードを指定します。パスワードをエンコードおよびデコードするのに役立つアルゴリズムが使用可能です。

enableXmlClob

オプション。これは、retrieveXML 操作が CLOB ベースの XML エクステンダー・ストアード・プロシージャーを使用するかどうかを指定します。デフォルト値は *true* です。このパラメーターは、後方互換性にのみ使用可能です。 OS/390 および z/OS プラットフォームの場合は、このプロパティーを定義しないか、または常にこの値を *true* に設定してください。

Web サービス構成パラメーター

groupNamespaceUri

オプション。このパラメーターは、生成された Web service description language (WSDL) および Extensible Markup Language (XML) スキーマ・ファイル (XSD) で使用されるターゲット・ネーム・スペースを定義します。ターゲット・ネーム・スペースは、このグループの Web サービス用です。

useDocumentStyle

オプション。デフォルト値は *false* で、Web サービスが実行時に RPC エンコードを使用することを意味します。この値を true に設定すると、 Web サービスは実行時に文書体裁およびリテラル・エンコードを使用します。IBM DB2 Information Integrator Web サービス・プロバイダーには、RPC スタイルを使用するよう設定されているサンプルが含まれています。新規のアプリケーションの場合は、インターオペラビリティーが最大になるよう、文書スタイルを使用する必要があります。

namespaceTable

オプション。これは、ネーム・スペース表のリソース名を指定します。 DB2 XML エクステンダー DTDID から XML Schema (XSD) ネーム・スペースおよびロケーションへのマッピングを定義するネーム・スペース表 (NST) リソースを参照します。 NST ファイルの例については、97 ページの図 24 をご覧ください。

ランタイム構成パラメーター:

autoReload

オプションですが、`reloadIntervalSeconds` と一緒に使用します。これはリソースを再ロードするかどうかを指定します。値は true または false にできます。デフォルトは false です。

reloadIntervalSeconds

オプションですが、`autoReload` と一緒に使用します。これは、リソースのロードとキャッシングを制御します。整数自動再ロード時間間隔を秒数で指定します。デフォルトは 0 で、WORF が要求ごとにより新しいリソースをチェックすることを意味します。

オプション *autoReload* と *reloadIntervalSeconds* はリソースのロードとキャッシングを制御します。 autoReload が指定されていないまたは false である場合、リソースは再ロードされず、アプリケーションは reloadIntervalSeconds を無視します。 autoReload が true の場合、WORF がリソース (例えば以下のファイルのいずれか: DAD、DADX、文書タイプ定義 (DTD)、NST) にアクセスすると、 WORF は現在時刻とリソースが以前にロードされた時刻を比較します。 `reloadIntervalSeconds` の値以上が経過すると、 WORF はより新しいバージョンがあるかファイル・システムをチェックし、変更されたリソースを再ロードします。開発時には自動再ロードが便利で、その場合は `reloadIntervalSeconds` をゼロに設定してください。 Web サービスが実動中の場合は、 `autoReload` を false に設定するか、または `reloadIntervalSeconds` を大きい値に設定して、サーバー・パフォーマンスへの影響を回避してください。

関連概念:

- 132 ページの『Web サービス記述言語』
- 69 ページの『Web サービスのグループの定義』

関連タスク:

- 73 ページの『iSeries プラットフォームでの web.xml ファイルおよび group.properties ファイルの定義』
- 70 ページの『web.xml ファイルおよび group.properties ファイルの定義』

DADX ファイル

このセクションでは、DADX ファイルのプロパティー、構文、および操作について説明します。

文書アクセス定義拡張ファイルを使用した Web サービスの定義

Document Access Definition Extension (DADX) ファイルは Web サービスを指定します。 SQL ステートメント、パラメーターのリスト、およびオプションで操作内容のセットを定義する Document Access Definition (DAD) ファイル参照を使用して、サービスの指定を行います。動的な Web サービス操作のセットを、DADX ファイルを使って定義できます。このファイルには動的照会サービス・タグ (<DQS/>) しか入っていません。操作内容は、呼び出し可能なメソッドに類似しています。以下の操作タイプによって、DADX Web サービス内の操作を定義できます。

- SQL 操作 (非動的)

  <query>
  選択操作を使用することによりデータベースを照会する

  <update>
  データベース上で更新、挿入または削除操作を実行する

  <call> ストアード・プロシージャーを呼び出す
- SQL 操作 (動的照会サービス)

  <getTables>
  使用可能な表の記述を検索する。

  <getColumns>
  列の記述を検索する。

<executeQuery>
単一の SQL ステートメントを発行する。

<executeUpdate>
単一の INSERT、UPDATE、DELETE を発行する。

<executeCall>
単一のストアード・プロシージャーを呼び出す。

<execute>
単一の SQL ステートメントを発行する。

- Extensible Markup Language (XML) コレクション操作 (DB2® XML エクステンダーが必要)

<retrieveXML>
XML 文書を生成する

<storeXML>
XML 文書を保管する

関連概念:

- 38 ページの『DADX ファイルを使った Web サービス・プロバイダー操作』

関連資料:

- 79 ページの『DADX ファイルの構文』

DADX ファイルの構文

DADX ファイルは Extensible Markup Language (XML) 文書です。 80 ページの『DADX 構文定義』 および 80 ページの図 15 で、 DADX のエレメントについて説明します。 DADX スキーマは DADX ファイルの XML スキーマにあります。 80 ページの『DADX 構文定義』 でノードやエレメントの隣の番号は子グループを識別します。番号付けの構造は、XML 文書階層を表現したものです。たとえば、識別番号が 1.3 (result_set_metadata) から 1.3.1 (column) に変わるとき、 column が result_set_metadata の子であることを意味します。 1.1 (documentation) から 1.2 (implements) に変わったときは、これらのエレメントが兄弟関係にあることを意味します。

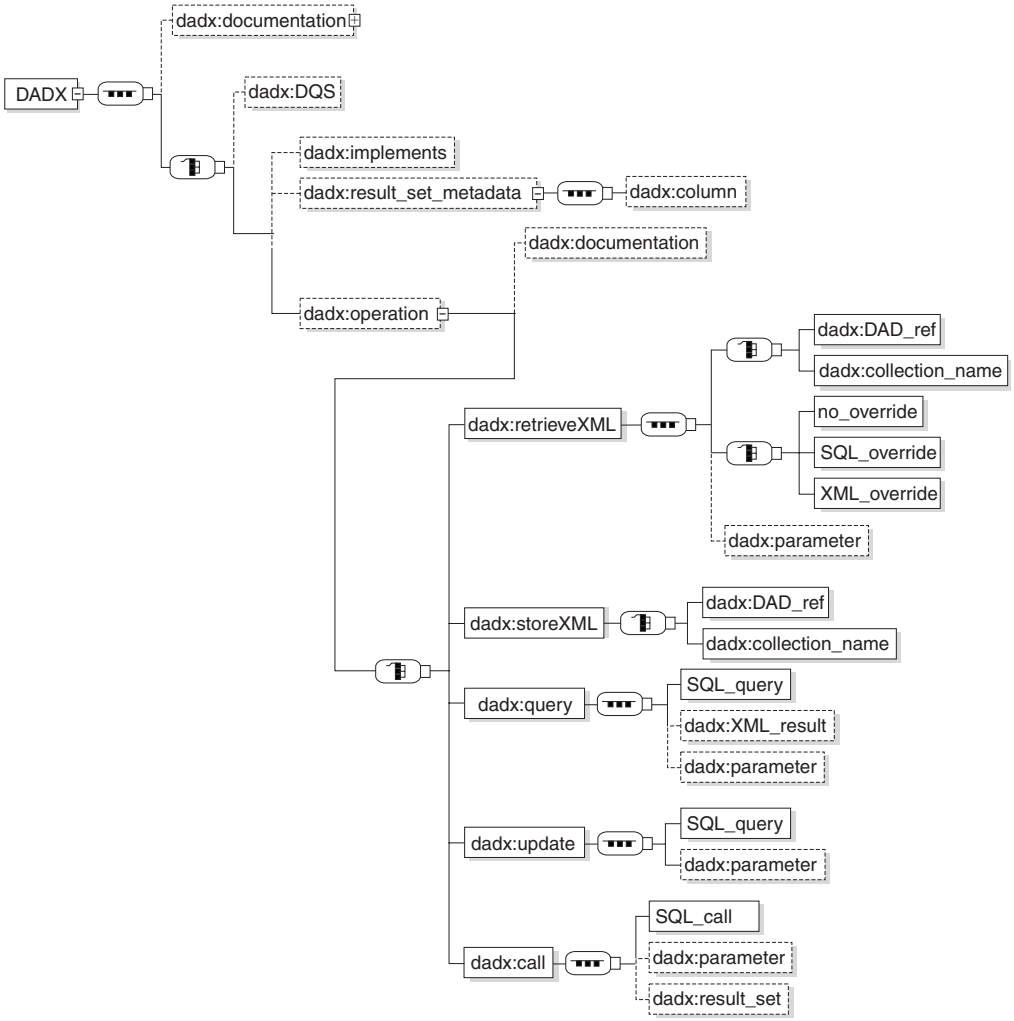


図 15. DADX 構文

0. ルート・エレメント: <DADX>

属性:

xmlns:dadx

DADX のネーム・スペース。

xmlns:xsd

Extensible Markup Language (XML) スキーマ指定のネーム・スペース。

子:

0.1 <documentation>

Web サービスの目的および内容についてのコメントまたは記述を指定します。 XHTML タグを使用できます。

1. 非動的操作を指定する DADX 関数

1.2 <implements>

Web サービス記述ファイルのネーム・スペースとロケーションを指定します。別の場所 (たとえば、UDDI レジストリー) に定義された再使用可能 WSDL 文書によって記述され

た標準 Web サービスを DADX Web サービスがインプリメントすることを、サービス・インプリメンターが宣言することを許可します。

1.3 <result_set_metadata>

ストアード・プロシージャーは 1 つ以上の結果セットを戻すことができます。それらの結果セットは出力メッセージに組み込むことができます。ストアード・プロシージャー結果セットのメタデータは、<result_set_metadata> エレメントを使用して、非動的 DADX 内で明示的に定義されなければなりません。ランタイムに、結果セットのメタデータを入手します。メタデータは DADX ファイルに含まれている定義と一致していなければなりません。

注: メタデータが修正された結果セットを持つストアード・プロシージャーのみを呼び出すことができます。

Web サービス用 WSDL ファイルを洗練されたものにするには、この制限が必要です。 <result_set> エレメントを使用することにより、単一の結果セット・メタデータを複数の <call> 操作で参照することができます。結果セット・メタデータ定義は DADX に対してグローバルであり、すべての操作定義エレメントより前になければなりません。

属性:

name 結果セットのルート・エレメントを識別します。

rowname
結果セットの各行のエレメント名として使用されます。

子:

1.3.1 <column>

列を定義します。列の順序は、ストアード・プロシージャーによって戻される結果セットの順序と一致していなければなりません。各列の名前、タイプ、および NULL 可能性は、結果セットと一致していなければなりません。

属性:

name 必須。これは、列の名前を指定します。

type **element** を指定しない場合に必須。列のタイプを指定します。

element
type を指定しない場合に必須。列のエレメントを指定します。

as オプション。これは、列の名前を提供します。

nullable

オプション。 nullable は true または false です。列値が NULL になりうるかどうかを示します。

1.4 <operation>

Web サービス操作を指定します。操作エレメントおよびその子は、操作の名前、および Web サービスが実行する操作のタイプを指定します。 Web サービスは XML 文書の構成、データベースの照会、またはストアード・プロシージャーの呼び出しを行えます。 1 つの DADX ファイルには、単一のデータベースまたはロケーション上での複数の操作を含めることができます。次のリストは、これらのエレメントを記述したものです。

- 属性:

  name 操作を識別するユニークなストリング。このストリングは DADX ファイル内でユニークでなければなりません。たとえば、「`findByColorAndMinPrice`」。

- 子:

次のエレメントを使用して、操作を文書化します。

1.4.1 <dadx:documentation>

操作の目的および内容についてのコメントまたは記述を指定します。 XHTML タグを使用できます。

1.4.2 <retrieveXML>

このエレメントは、XML コレクション・アクセス・メソッドを使用する際に、リレーショナル表のセットからゼロまたは 1 つの XML 文書を生成することを指定します。 DAD ファイルまたは XML コレクション名を指定するかどうかに応じて、操作は適切な XML エクステンダー構成ストアード・プロシージャーを呼び出します。

子:

– これらのストアード・プロシージャーのどれを使用したいかを指定します。以下のエレメントの 1 つを使用し、 DAD ファイルの名前またはコレクションの名前のどちらかを渡すことによって指定します。

1.4.2.1 <DAD_ref>

このエレメントの内容は、DAD ファイルの名前およびパスです。 DAD ファイルの相対パスを指定する場合、ア

プリケーションはグループ・ディレクトリーが現行作業ディレクトリーであると想定します。

1.4.2.2 <collection_name>

このエレメントの内容は XML コレクションの名前です。「*DB2 XML Extender 管理およびプログラミングのガイド*」で説明されているように、コレクションは XML エクステンダー管理インターフェースを使用して定義します。

– 以下のエレメントの 1 つを使って、オーバーライド値を指定します。

1.4.2.3 <no_override/>

DAD ファイル内の値をオーバーライドしないことを指定します。<SQL_override> または <XML_override> のいずれかを指定しない場合に必須。

1.4.2.4 <SQL_override>

SQL マッピングを使用する DAD ファイル内の SQL ステートメントをオーバーライドすることを指定します。

1.4.2.5 <XML_override>

RDB マッピングを使用する DAD ファイル内の XML ステートメントをオーバーライドすることを指定します。

– 次のエレメントを使用して、パラメーターを定義します。

1.4.2.6 <parameter>

<SQL_override> または <XML_override> エレメント内のパラメーターを参照する場合に必須。このエレメントは操作のパラメーターを指定します。操作で参照される各パラメーターごとに別個のパラメーター・エレメントを使用します。各パラメーター名は、操作内でユニークでなければなりません。パラメーターの内容は、XML スキーマ・エレメント (複合タイプ) または単純タイプのいずれかによって定義されたものでなければなりません。

属性:

name パラメーターのユニークな名前。

element
「element」属性を使用して XML スキーマ・エレメントを指定します。

type 「type」属性を使用して、単純タイプを指定します。

kind パラメーターが入力データを受け渡すか、出力データを戻すか、またはその両方を実行するかを指定します。この属性の有効な値は、以下のとおりです。

- in

1.4.3 <storeXML>

このエレメントは、XML コレクション・アクセス・メソッドを使用して、リレーショナル表のセットに XML 文書を保管 (分解) することを指定します。 DAD ファイルまたは XML コレクション名を指定するかどうかに応じて、操作は適切な XML エクステンダー分解ストアード・プロシージャーを呼び出します。
子:

– これらのストアード・プロシージャーのどれを使用したいかを指定します。以下のエレメントの 1 つを使用し、 DAD ファイルの名前またはコレクションの名前のどちらかを渡すことによって指定します。

1.4.3.1 <DAD_ref>
このエレメントの内容は、DAD ファイルの名前およびパスです。 DAD ファイルの相対パスを指定する場合、アプリケーションはグループ・ディレクトリーが現行作業ディレクトリーであると想定します。

1.4.3.2 <collection_name>
このエレメントの内容は XML コレクションの名前です。「*DB2 XML Extender 管理およびプログラミングのガイド* 」で説明されているように、コレクションは XML エクステンダー管理インターフェースを使用して定義します。

1.4.4 <query>

照会操作を指定します。 <SQL_select> エレメントに SQL SELECT ステートメントを使用することにより、操作を定義します。ステートメントは、ゼロまたは 1 つ以上の名前付き入力パラメーターを取ることができます。ステートメントが入力パラメーターを取る場合は、各パラメーターが <parameter> エレメントによって記述されます。

この操作は各データベース列を結果セットから対応する XML エレメントにマップします。<query> 操作に XML エクステンダー・ユーザー定義タイプ (UDT) を指定できます。しかし、その場合 <XML_result> エレメント、およびそれをサポートする照会対象の XML 列のタイプを定義する文書タイプ定義 (DTD) も必要です。

子:

1.4.4.1 <SQL_query>

SQL SELECT ステートメントを指定します。

1.4.4.2 <XML_result>

オプション。これは、XML 文書を含む名前付き列を定義します。このルートの XML スキーマ・エレメントは文書タイプを定義します。

属性:

name 列に保管される XML 文書のルート・エレメントを指定します。

element
列内の特定のエレメントを指定します。

1.4.4.3 <parameter>

<SQL_query> エレメント内のパラメーターを参照する場合に必須。操作のパラメーターを指定します。操作で参照される各パラメーターごとに別個のパラメーター・エレメントを使用します。各パラメーター名は、操作内でユニークでなければなりません。パラメーターの内容は、XML スキーマ・エレメント (複合タイプ) または単純タイプの 1 つによって定義されたものでなければなりません。

属性:

name パラメーターのユニークな名前。

element
「element」属性を使用して XML スキーマ・エレメントを指定します。

type 「type」属性を使用して、単純タイプを指定します。

kind パラメーターが入力データを受け渡すか、出力データを戻すか、またはその両方を実行するかを指定します。この属性の有効な値は、以下のとおりです。

– in

1.4.5 <update>

操作は、<SQL_update> エレメント内の SQL INSERT、 DELETE、または UPDATE ステートメントによって定義されます。ステートメントは、ゼロまたは 1 つ以上の名前付き入力パラメーターを取ることができます。ステートメントが入力パラメーターを取る場合は、各パラメーターが <parameter> エレメントによって記述されます。

子:

1.4.5.1 <SQL_update>

これは SQL INSERT、UPDATE、または DELETE ステートメントを指定します。

1.4.5.2 <parameter>

<SQL_update> エレメント内のパラメーターを参照する場合に必須。操作のパラメーターを指定します。操作で参照される各パラメーターごとに別個のパラメーター・エレメントを使用します。各パラメーター名は、操作に対してユニークでなければなりません。パラメーターの内容は、XML スキーマ・エレメント (複合タイプ) または単純タイプの 1 つによって定義されたものでなければなりません。

属性:

name パラメーターのユニークな名前。

element
「element」属性を使用して XML スキーマ・エレメントを指定します。

type 「type」属性を使用して、単純タイプを指定します。

kind パラメーターが入力データを受け渡すか、出力データを戻すか、またはその両方を実行するかを指定します。この属性の有効な値は、以下のとおりです。

– in

1.4.6 <call>

ストアード・プロシージャーへの呼び出しを指定します。この処理は更新操作に似ていますが、呼び出し操作のパラメーターは「in」、「out」、または「in/out」として定義できます。デフォルト・パラメーターの種類は「in」です。「out」および「in/out」パラメーターは出力メッセージに表示されます。

1.4.6.1 <SQL_call>

ストアード・プロシージャー・コールを指定します。

1.4.6.2 <parameter>

<SQL_call> エレメント内のパラメーターを参照する場合に必須。これは操作のパラメーターを指定します。操作で参照される各パラメーターごとに別個のパラメーター・エレメントを使用します。各パラメーター名は、操作内でユニークでなければなりません。パラメーターの内容は、XML スキーマ・エレメント (複合タイプ) または単純タイプの 1 つによって定義されたものでなければなりません。

属性:

name パラメーターのユニークな名前。

element
「element」属性を使用して XML スキーマ・エレメントを指定します。

type 「type」属性を使用して、単純タイプを指定します。

kind パラメーターが入力データを受け渡すか、出力データを戻すか、またはその両方を実行するかを指定します。この属性の有効な値は、以下のとおりです。

– in

– out

– in/out

1.4.6.3 <result_set>

これは結果セットを定義しますが、<parameter> エレメントの後に来なければなりません。結果セット・エレメントは、操作のパラメーターと結果セットすべての中で必ずユニークな名前を持っています。それは、<result_set_metadata> エレメントを参照しなければなりません。1 つの <result_set> エレメントが、ストアード・プロシージャーから戻される各結果セットごとに定義されていなければなりません。

属性:

name SOAP 応答の結果セットのユニーク ID。

metadata
DADX ファイル内の結果セット・メタデータ定義。 ID はエレメントの名前を参照していなければなりません。

2. **<DQS>**
   動的照会サービス。

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 37 ページの『DADX ファイルの定義』

関連資料:

- 88 ページの『単純な DADX ファイル』
- 273 ページの『付録 C. DADX ファイルの XML スキーマ』

単純な DADX ファイル

以下は、SQL 照会を使用した 1 つの操作が含まれる簡単な DADX ファイルの例です。この DADX ファイルは非動的照会用です。動的照会サービスを使用可能にする DADX ファイルの例に関しては、「Web サービス・プロバイダーの一部としての動的データベース照会の構成および実行」を参照してください。

89 ページの図 16 に、簡単な Web サービスを定義する DADX ファイルを示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<DADX
  xmlns="http://schemas.ibm.com/db2/dxx/dadx"
  >
  <documentation>
    Simple DADX example that accesses the SAMPLE database.
  </documentation>
  <operation name="listDepartments">
    <documentation>
      Lists the departments.
    </documentation>
    <query>
      <SQL_query>SELECT * FROM DEPARTMENT</SQL_query>
    </query>
  </operation>
</DADX>
```

図 16. 簡単な DADX ファイル

このサンプル DADX ファイルは、listDepartments という名前の 1 つの操作で Web サービスを定義しており、その操作は DEPARTMENT 表の内容をリストします。操作名は Web サービス・アクティビティーを識別し、プログラム言語でいうメソッド名に似ています。

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 37 ページの『DADX ファイルの定義』
- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』

関連タスク:

- 107 ページの『Web サービス・プロバイダーの一部としての動的データベース照会の構成および実行』

XML 集合の操作

<retrieveXML> または <storeXML> 操作を使用して、 XML 文書を生成または保管できます。これらの操作は、XML エクステンダー・ストアード・プロシージャーを呼び出し、 DAD ファイルまたは XML コレクション参照を必要とします。これらのストアード・プロシージャーは、 DAD ファイルでマッピングを使用するか、または使用可能 XML コレクションを参照することによって、 XML 文書を生成または保管します。DAD ファイルを作成する方法については、「*DB2 XML Extender 管理およびプログラミングのガイド*」を参照してください。

次の例は、DAD ファイルから XML 文書を生成するより複雑な DADX ファイルを示しています。これは、<RetrieveXML> エレメントを使用することによって、ストアード・プロシージャーを参照します。 <DAD_ref> エレメントは DAD ファイルの名前を指定します。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <documentation>
   Provides queries for part order information at myco.com.
   See
   <xhtml:a href="../documentation/PartOrders.html"
      target="_top">PartOrders.html
  </xhtml:a>
   for more information.
  </documentation>
  <operation name="findAll">
    <documentation>
      Returns all the orders with their complete details.
    </documentation>
    <retrieveXML>
       <DAD_ref>getstart_xcollection.dad</DAD_ref>
       <no_override/>
    </retrieveXML>  </operation>
</DADX>
```

図 17. XML 文書を生成する DADX ファイル

この DADX ファイルから生成される Web サービスは、 dxxGenXML ストアード・プロシージャーを呼び出し、XML 文書を生成します。ストアード・プロシージャーは、XML 文書を生成するときにどの表を使用するかを決定するために getstart_xcollection.dad ファイルを、さらに XML 文書構造を参照します。

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 126 ページの『Web サービス・アプリケーションのテスト - シナリオ』

関連資料:

- 94 ページの『DADX 操作の例』

DADX ファイルでのオーバーライドの使用

DADX は、<XML_override> および <SQL_override> エレメントを使用することにより、 DAD ファイル内の Extensible Markup Language (XML) 値および SQL ステートメントをオーバーライドできます。オーバーライドのタイプは、DAD ファイルが SQL マッピングまたは RDB マッピングを使用するかどうかによって決定されます。 DAD 値をオーバーライドする必要がない場合は、図 17 に示されているように、 <no_override/> エレメントを使用します。

次の例では、SQL オーバーライド・ステートメントが使用されています。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <documentation >
    Provides queries for part order information at myco.com.
    See <xhtml:a href="../documentation/PartOrders.html" target="_top">
         PartOrders.html</xhtml:a> for more information.
  </documentation>
  <operation name="findAll">
    <documentation >
      Returns all the orders with their complete details.
    </documentation>
    <retrieveXML>
      <DAD_ref>getstart_xcollection.dad</DAD_ref>
        <SQL_override>
           select o.order_key, customer_name, customer_email,
            p.part_key, color, quantity, price, tax, ship_id, date, mode
           from order_tab o, part_tab p,
             table(select substr(char(timestamp(generate_unique())),16)
               as ship_id, date, mode, part_key from ship_tab) s
           where p.order_key = o.order_key and s.part_key = p.part_key
              order by order_key, part_key, ship_id
       </SQL_override>
   </retrieveXML>
  </operation>
</DADX>
```

図 18. SQL オーバーライドを使って XML 文書を生成する DADX ファイルの例

SQL ステートメントをオーバーライドすることは可能ですが、新規 SQL ステートメントは、DAD ファイルに定義された SQL マッピングと互換性のある結果セットを生成しなければなりません。たとえば、DAD ファイルに現れる列名は SQL オーバーライドにも現れていなければなりません。

DAD ファイルが RDB ノード・マッピングを使用する場合、 <XML_override> エレメントを使用することによって、 RDB ノードをオーバーライドすることが必要です。 RDB ノード・エレメントは、XML データを含むことになっている DB2 Universal Database 表、列、および条件を定義します。 92 ページの図 19 の例は、RDB ノード DAD ファイルを参照する DADX ファイルを示しています。 <XML_override> エレメント内容は、 DAD ファイルに指定された条件をオーバーライドします。オーバーライド・ストリングには、ホスト変数構文を使用して入力パラメーターを含めることができます。この操作内で固有の名前が付けられているパラメーター・エレメントのリストに、すべてのパラメーターの名前とタイプを定義しなければなりません。この例では、価格を $50.00 以上に制限し、日付を 1998-12-01 以降に制限することにより、オーバーライド・パラメーターは照会をオーバーライドします。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <documentation >
    Provides queries for part order information at myco.com.
    See <xhtml:a href="../documentation/PartOrders.html" target="_top">
         PartOrders.html</xhtml:a> for more information.
  </documentation>
  <operation name="findByExtendedPriceAndShipDate">
    <documentation >
      Returns all the orders with an extended price greater than $50.00
      and a ship date later than 1998-12-01.
    </documentation>
<retrieveXML>
      <DAD_ref>order_rdb.dad</DAD_ref>
        <XML_override>
           /Order/Part/ExtendedPrice > 50.00 AND
           Order/Part/Shipment/ShipDate > '1998-12-01'
       </XML_override>
    </retrieveXML>
</operation>
</DADX>
```

図 19. XML オーバーライドを使って XML 文書を生成する DADX ファイルの例

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』

関連資料:

- 94 ページの『DADX 操作の例』
- 88 ページの『単純な DADX ファイル』

DADX ファイルでのパラメーターの宣言および参照

各操作にパラメーターを使用できます。 SQL 操作用の <SQL_query>、<SQL_update>、 <SQL_call> ステートメントはパラメーターを参照できます。<retrieveXML> および <storeXML> 操作で使用する Extensible Markup Language (XML) と SQL のオーバーライドもパラメーターを参照できます。パラメーターを宣言するには、<parameter> エレメントを使用します。パラメーターには、組み込み SQL データ・タイプに対応するシンプルな XML スキーマ・タイプがあります。表 6 には、サポートされているタイプが説明されています。

表 6. サポートされている XML スキーマおよび SQL タイプ

XML スキーマ単純タイプ	SQL タイプ
string	CHAR, VARCHAR, CLOB, LONGVARCHAR
decimal	DECIMAL, NUMERIC
int	INTEGER
short	SMALLINT
float	FLOAT
double	REAL, DOUBLE PRECISION
date	DATE
time	TIME

表 6. サポートされている XML スキーマおよび SQL タイプ (続き)

XML スキーマ単純タイプ	SQL タイプ
timestamp	TIMESTAMP
long	BIGINT
byte	TINYINT

パラメーターを参照するには、コロン接頭部を使用します。以下に例を示します。

```
<SQL_query>
 select * from order_tab where customer_name =:customer_name
</SQL_query>
```

パラメーターを定義するには、次の例のように、<parameter> エレメントを使用します。

```
<parameter name="customer_name" type="xsd:string"/>
```

<parameter> エレメントで参照する各パラメーターは、定義しなければなりません。このエレメントの名前属性はパラメーターを識別し、操作内でユニークでなければなりません。

93 ページの図 20 の例は、 SQL SELECT ステートメントを使用することによってリレーショナル・データのセットを検索する照会操作を示しています。ステートメントには、パラメーター構文を使用することによって、 1 つの入力パラメーターが含まれています。

```
<operation name="findCustomerOrders">
    <documentation>Returns all the orders for a given customer.
    </documentation>
    <query>
      <SQL_query>select * from order_tab where customer_name =
        :customer_name</SQL_query>
      <parameter name="customer_name" type="xsd:string"/>
    </query>
  </operation>
```

図 20. パラメーターを持つ照会操作

94 ページの図 21 の例は、 SQL オーバーライドで retrieveXML 操作が使用するパラメーターを示しています。

```
<operation name="findByColorAndMinPrice">
    <documentation>Returns all the orders that have the specified color and
       at least the specified minimum price.
    </documentation>
    <retrieveXML>
      <DAD_ref>getstart_xcollection.dad
</DAD_ref>
      <SQL_override>
       select o.order_key, customer_name, customer_email,
         p.part_key, color, quantity, price, tax, ship_id, date, mode
       from order_tab o, part_tab p,
         table(select substr(char(timestamp(generate_unique())),16)
           as ship_id, date, mode, part_key from ship_tab) s
       where p.order_key = o.order_key and s.part_key = p.part_key
         and color = :color and price >= :minprice
       order by order_key, part_key, ship_id
      </SQL_override>
      <parameter name="color" type="xsd:string">
      <parameter name="minprice" type="xsd:decimal">
    </retrieveXML>
  </operation>
```

図 21. retrieveXML 操作によって使用される SQL オーバーライド

SQL ステートメントの WHERE 文節を変更して、検索条件を組み込むことができます。 SQL オーバーライドは、1 つ以上のパラメーターを組み込むことができ、それらはコロンを使用して識別されます。この例では、`findByColorAndMinPrice` は `:color` と `:minprice` を参照します。 <parameter> エレメントを使用して、パラメーターを宣言します。パラメーターには、組み込み SQL データ・タイプに対応するシンプルな XML スキーマ・ファイル (XSD) タイプがあります。

関連概念:

- 137 ページの『XML スキーマ定義』

関連資料:

- 88 ページの『単純な DADX ファイル』
- 79 ページの『DADX ファイルの構文』
- 273 ページの『付録 C. DADX ファイルの XML スキーマ』

DADX 操作の例

以下のサンプルは、さまざまな操作を含む DADX ファイルを示しています。

例 1: Query 操作

この例は、デフォルト・タグを使用した Query 操作を示しています。この例の場合、XML エクステンダーは必要ありません。この操作は、特定の顧客の注文すべてを選択します。このサンプルを実行するには、sales_db XML エクステンダー・サンプル・データベースが必要です。

```
<?xml version="1.0"?>
 <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <documentation>
      mycompany part orders service.
    </documentation>
  <implements namespace="http://www.poia.org/part_orders.wsdl"
      location="http://www.poia.org/part_orders.wsdl"/>
<operation name="findCustomerOrders">
    <documentation>Returns all the orders for a given customer.
    </documentation>
    <query>
      <SQL_query>select * from order_tab
                  where customer_name = :customer_name
      </SQL_query>
      <parameter name="customer_name" type="xsd:string"/>
    </query>
  </operation>
```

図 22. Query 操作を含む DADX

この操作内で固有に名前が付けられているパラメーター・エレメントのリストによって、入力パラメーターが定義されていなければなりません。これ以上のマッピングの制御が必要である場合は、DAD ファイルを使用することができます。

Query 操作を使用して、 XML エクステンダー・ユーザー定義タイプ (UDT) およびユーザー定義関数 (UDF) を使用することができます。この操作によって、XML 文書を含む XML 列からデータを照会、抽出、および更新できます。これらの XML 文書を使用するには、<XML_result> エレメントのタイプを定義する文書タイプ定義 (DTD) を作成することが必要です。このエレメントは、列名と、それに含まれる XML 文書のルート・エレメントを指定します。

96 ページの図 23 の例は、 <XML_result> によって宣言された VARCHAR UDT を使用する Query 操作を示しています。 `retrieveOrders` 操作は、UDF db2xml.varchar を使用して、 SALES_TAB 表からすべての XML 注文文書を検索します。 XML エクステンダー UDT XMLVARCHAR を使用して、文書を保管します。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:dtd1="http://schemas.myco.com/sales/order.dtd">
  <documentation>
       Queries part orders at myco.com.
  </documentation>

<operation name="retrieveOrders">
    <documentation>
        Retrieves all the Order documents.
    </documentation>
    <query>
      <SQL_query>
        select db2xml.varchar(order) from sales_tab
      </SQL_query>
      <XML_result name="ORDER" element="dtd1:Order"/>
    </query>
  </operation>
</DADX>
```

図 23. UDF および UDT を使用する Query 操作

列に XML 文書があり、この文書のタイプを WSDL が参照するようにしたい場合は、 XML_result タグを使用することができます。 96 ページの図 23 の例では、 ORDER 列にエレメント *dtd1:Order* のフラグメントが含まれるように指定しています。エレメント <XML_result name = "ORDER" element = "dtd1:Order"/> はネームスペース宣言を参照します。 XML エクステンダーはネーム・スペースのない XML 文書および DTD によって定義される XML 文書を保管します。 Web サービスは DTD の代わりに XML スキーマ (XSD) を使用し、ネーム・スペースを活用します。ネーム・スペース表に項目を作成して、ネーム・スペースを DTD と関連付けます。 WORF は XML 文書を検索するときにネーム・スペースを追加し、文書を保管するときにネーム・スペースを削除します。 WORF は自動的に DTD を XSD に変換することもします。行 <XML_result name = "ORDER" element = "dtd1:Order"/> は、*order.dtd* ファイルの列情報を定義します。参照先の特定の宣言は、次の例にあるとおりです。

```
<?xml encoding="US-ASCII"?>
<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
...
```

DTD を指し示すには、ネーム・スペース表ファイル (NST ファイル) を使用します。例として、97 ページの図 24 を参照してください。

```
<?xml version="1.0"?>
  <namespaceTable xmlns="http://schemas.ibm.com/db2/dxx/nst">
    <mapping dtdid="c:¥dxx¥samples¥dtd¥getstart.dtd"
      namespace="http://schemas.ibm.com/db2/dxx/samples/dtd/getstart.dtd"
      location="/dxx/samples/dtd/getstart.dtd/XSD"/>
    <mapping dtdid="getstart.dtd"
      namespace="http://schemas.myco.com/sales/getstart.dtd"
      location="/getstart.dtd/XSD"/>
    <mapping dtdid="order.dtd"
      namespace="http://schemas.myco.com/sales/order.dtd"
      location="/order.dtd/XSD"/>
  </namespaceTable>
```

図 24. NST ファイル

group.properties ファイル内で、このファイルを参照しなければなりません。このファイルについてさらに学習するには、 72 ページの図 12 にある例をご覧ください。

例 2: Update 操作

97 ページの図 25 にある例は、特定の注文に関して顧客の電子メール・アドレスを更新する操作を示しています。更新操作では、 SQL INSERT、DELETE、または UPDATE ステートメントを <SQL_update> エレメントに含めることができます。

```
  <operation name="updateOrderEmail">
    <documentation>Updates the email address for an order.
    </documentation>
    <update>
      <SQL_update>update order_tab set customer_email = :email
                  where order_key = :key</SQL_update>
      <parameter name="key" type="xsd:int"/>
      <parameter name="email" type="xsd:string"/>
    </update>
  </operation>
 </DADX>
```

図 25. Update 操作

例 3: Call 操作

以下の例は、ストアード・プロシージャーを呼び出す Call 操作を示しています。

ストアード・プロシージャーが結果セットを戻す場合、 DADX ファイルの result_set_metadata タグにこれらの結果セットを定義しなければなりません。これは、 WORF がこの Web サービス操作用の WSDL および XML スキーマ・ファイル (XSD) を生成できるようにするためです。 98 ページの図 26 は、2 回参照される結果セット・メタデータの定義を示しています。

```
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<result_set_metadata name="employeeSalaryReport" rowName="employee">
  <column name="NAME" type="VARCHAR" nullable="true" />
  <column name="JOB" type="CHAR" nullable="true" />
  <column name="3" as="SALARY" type="DOUBLE" nullable="true" />
</result_set_metadata>
<operation name="twoResultSets">
<call>
<SQL_call>CALL TWO_RESULT_SETS (:salary, :sqlCode)
</SQL_call>
  <parameter name="salary" type="xsd:double" kind="in" />
  <parameter name="sqlCode" type="xsd:int" kind="out" />
  <result_set name="employees1" metadata="employeeSalaryReport" />
  <result_set name="employees2" metadata="employeeSalaryReport" />
  </call>
 </operation>
</DADX>
```

図 26. 2 回参照される結果セット・メタデータの定義

98 ページの図 27 に示されているフォーマットを使用することによって、ストアード・プロシージャーを呼び出すこともできます。

```
<operation name="callProc1">
    <documentation>Call the Proc1 stored procedure.
    </documentation>
    <call>
      <SQL_call>
        CALL Proc1 (:x, :y, :z)
      </SQL_call>
      <parameter name="x" type="xsd:string" kind="in"/>
      <parameter name="y" type="xsd:int" kind="in/out"/>
      <parameter name="z" element="dtd1:Order" kind="out"/>
    </call>
</operation>
```

図 27. 代替 Call 操作を含む DADX

例 4: RetrieveXML 操作

99 ページの図 28 の DADX ファイルは、ストアード・プロシージャー dxxGenXMLCLOB を使用することによって、 1 つの retrieveXML 操作をインプリメントします。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <documentation>
      mycompany part orders service.
    </documentation>
  <operation name="findByColorAndMinPrice">
    <documentation>Returns all the orders that have the specified color
        and at least the specified minimum price.</documentation>
    <retrieveXML>
      <DAD_ref>getstart_xcollection.dad</DAD_ref>
      <SQL_override>
       select o.order_key, customer_name, customer_email,
         p.part_key, color, quantity, price, tax, ship_id, date, mode
       from order_tab o, part_tab p,
         table(select substr(char(timestamp(generate_unique())),16)
           as ship_id, date, mode, part_key from ship_tab) s
       where p.order_key = o.order_key and s.part_key = p.part_key
         and color = :color and price >= :minprice
       order by order_key, part_key, ship_id
      </SQL_override>
      <parameter name="color" type="xsd:string"/>
      <parameter name="minprice" type="xsd:decimal"/>
    </retrieveXML>
  </operation>
</DADX>
```

図 28. retrieveXML 操作を含む DADX

99 ページの図 28 の操作は、 *getstart_xcollection.dad* ファイル内のマッピングに基づく XML 文書を生成します。この操作は SQL オーバーライドを指定します。また、DAD ファイルに定義されている SQL ステートメントを置き換え、オーバーライド・ステートメントの 2 つのパラメーター `:color` と `:minprice` を参照します。

この例の DAD ファイルは、「*DB2 XML Extender 管理およびプログラミングのガイド*」の付録にあります。

例 5: StoreXML 操作

100 ページの図 29 の例は、 RDB_node マッピング *getstart_xcollection_rdb.dad* を使用して、 DAD を参照する DADX ファイルを示しています。

```
<?xml version="1.0"?>
 <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <documentation>
      mycompany part orders service.
    </documentation>

  <implements namespace="http://www.poia.org/part_orders.wsdl"
      location="http://www.poia.org/part_orders.wsdl"/>

   <operation name="storeOrder">
     <documentation>Stores an automotive part order.
     </documentation>
     <storeXML>
       <DAD_ref>getstart_xcollection_rdb.dad</DAD_ref>
     </storeXML>
   </operation>
 </DADX>
```

図 29. StoreXML 操作を含む DADX

<DAD_ref> の代わりに <collection_name> エレメントを使用する場合、storeXML 操作は dxxInsertXML ストアード・プロシージャーによってインプリメントされます。この操作は dxxShredXML プロシージャーと同じ操作を実行しますが、 DAD ファイルの代わりに XML コレクションの名前を使用します。

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』

関連資料:

- 88 ページの『単純な DADX ファイル』

文書タイプ定義を XML スキーマに変換する

現在 XML エクステンダーは文書構造を定義するために文書タイプ定義ファイル (DTD) を使用していますが、 Web サービス記述言語 (WSDL) は XML スキーマ (XSD ファイル) を使用します。 Web サービス・オブジェクト・ランタイム・フレームワーク (WORF) は自動的に XML スキーマ (XSD) ファイルを作成します。DTD に関連付けられているネーム・スペースを定義するには、ネーム・スペース表 (NST ファイル) に項目を追加しなければなりません。このようにしても、DTD から XSD への変換が可能になります。

手順:

次の URL 構文を使用して、XSD ファイルを要求します。

```
http://host/path/dtd_file.dtd/XSD
```

以下に例を示します。

```
http://host_name:port/services/sample/order.dtd/XSD
```

この場合、order.dtd ファイルは WEB-INF¥groups¥dxx_sample になければなりません。

WORF と XML エクステンダーは文書タイプ定義 ID (DTDID) から DTD を探し出します。 DTDID は、ファイル名、またはデータベースの DTD_REF 表のキー値のいずれかです。データベースを使用可能にすると、 XML エクステンダーは DTD_REF 表を作成します。最良実例は DTD_REF 表に DTD を保管することです。なぜなら、Web アプリケーションを別のマシンに移動したときに、ファイル・ロケーションが変更する場合があるからです。 SALES_DB の例の Windows 2000 setup-xcollection.cmd ファイルからの次の抜き出しは、 DTD を DTD_REF 表に挿入する方法を示しています。

```
db2 "connect to SALES_DB"
rem Insert DTDs
db2 "insert into db2xml.dtd_ref values('getstart.dtd',
 db2xml.XMLClobFromFile('%CD%¥getstart.dtd'),
 0, 'user1', 'user1', 'user1')"
db2 "insert into db2xml.dtd_ref
  values('order.dtd', db2xml.XMLClobFromFile('%CD%¥order.dtd'),
         0, 'user1', 'user1', 'user1')"
```

関連概念:

- 137 ページの『XML スキーマ定義』

関連資料:

- 273 ページの『付録 C. DADX ファイルの XML スキーマ』

DADX ファイルからの WSDL

DADX 文書には、Web サービスをインプリメントするために必要な情報が含まれています。また、Web サービスを記述する WSDL 文書を生成するために必要な情報も含まれています。

Web サービス記述言語 (WSDL) 文書は、ビジネス・サービスのインターフェースを記述するために使用される Extensible Markup Language (XML) のボキャブラリーです。これを使用して、UDDI レジストリーにサービスを発行することができます。 WSDL を使用することにより、Web サービスをバインドする際に使用するリクエスター・コードとプロバイダー・コードを開発ツールを使ってプログラマチックに作成することが可能になります。また、前提条件のあるアプリケーションが Web サービスに動的にバインドすることも可能になります。 WSDL を使用して、要求と応答に必要なデータを指定することができます。 WSDL は精密なデータ定義のために XML スキーマを使用します。

WSDL バインディングは、どのようにサービスがメッセージング・プロトコル (特に SOAP メッセージング・プロトコル) にバインドするかを記述します。 WSDL SOAP バインディングは RPC スタイルのバインディングまたは文書スタイルのバインディングのいずれかになります。また、SOAP バインディングは、エンコードして使用することも、リテラルで使用することもできます。

WSDL を生成するには、次の URL をサブミットします。その中で <*yourWebAppServer*> によって指定される *localhost* ポート番号は、現在使用しているマシンごとに異なります。

```
http://yourWebAppServer:port/webapp_name/group_name/dadx_file.dadx/WSDL
```

Web サービス・オブジェクト・ランタイム・フレームワーク (WORF) は WSDL 文書を動的に生成します。これを UDDI または他の Web サービス・ディレクトリー内で発行することができます。

インストール時に WORF に組み込まれているサンプルを使用する場合は、次の URL をサブミットできます。

```
http://yourWebAppServer/services/sales/PartOrders.dadx/WSDL
```

関連概念:

- 102 ページの『UDDI 登録のための WSDL』
- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 132 ページの『Web サービス記述言語』

関連タスク:

- 127 ページの『Web サービスのテスト』

UDDI 登録のための WSDL

Universal Description, Discovery, and Integration (UDDI) の最良実例文書は、 UDDI レジストリーでの Web サービス記述言語 (WSDL) の使用について説明します。この文書は、WSDL 文書をデプロイメントと再使用可能の 2 つのパートに分割することを推奨します。

デプロイメント・パートには、サービスがデプロイされる URL を含む <service> エレメントが組み込まれます。デプロイメント・パートは、その他のトップレベル WSDL エレメントを含む再使用可能パートをインポートします。

再使用可能パートは UDDI <tModel> エレメントに対応します。デプロイメント・パートは UDDI <businessService> に対応します。 <businessService> エレメント内では、各 WSDL <port> エレメントが UDDI <bindingTemplate> エレメントに対応します。

UDDI および Web サービス登録の詳細については、 Universal Description, Discovery, and Integration of Business for the Web サイトを参照してください。

WSDL のパートを生成するには、URL に WSDL パス情報を指定してサブミットします。

- デプロイメント・パートを生成するには、 URL に `WSDLservice` キーワードを指定してサブミットします。

  ```
  http://yourWebAppServer:port/webapp_name/
     group_name/DADX_file.dadx/WSDLservice
  ```

- 再使用可能パートを生成するには、 URL に `WSDLbinding` キーワードを指定してサブミットします。

  ```
  http://yourWebbAppServer:port/webapp_name/
      group_name/DADX_file.dadx/WSDLbinding
  ```

次の例は、WSDL 文書のデプロイメント・パートおよび再使用可能パートを生成する方法を例示しています。デプロイメント・パートを生成するには、次の例が示すように、URL に `WSDLservice` コマンドを指定してサブミットします。

```
http://yourWebAppServer/sales_db/part_orders.dadx/WSDLservice
```

再使用可能パートを生成するには、次の例が示すように、URL に WSDLbinding コマンドを指定してサブミットします。

```
http://yourWebAppServer/sales_db/part_orders.dadx/WSDLbinding
```

上記の例は、サービス・インプリメンターが会社にユニークな Web サービスを作成する例を扱っています。 UDDI が処理するように設計されている使用のシナリオの 1 つは、規格機関またはベンダーが Web サービス・インターフェース tModel を定義するものです。次いで、サービス・インプリメンターがその Web サービスを使用します。たとえば、航空業界はフライト・スケジュールを提供する Web サービスを定義し、航空会社がそれをインプリメントすることが可能です。 UDDI を使用することにより、ユーザーは上述の tModel をインプリメントするすべての登録済みサービスを検索できます。その後、旅行計画アプリケーションはすべての航空フライト・スケジュール・サービスを探索できます。

他の場所に定義されている再使用可能 WSDL 文書によって記述されている Web サービスを前述のサービスがインプリメントすることを宣言するには、 DADX <implements> エレメントを使用します。 <implements> エレメントの例は 104 ページの図 30 に示されています。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://schemas.ibm.com/db2/dxx/dadx"
  ...
  elementFormDefault="qualified">
  <import namespace="http://schemas.xmlsoap.org/wsdl/"
   schemaLocation="wsdl.xsd"/>
        ....
   <element name="DADX">
    <annotation>
      <documentation>
        Defines a Web Service.
        The Web Service is described by an optional
         WSDL documentation element.
        ....
      </documentation>
    </annotation>
    <complexType>
      <sequence>
        <element ref="wsdl:documentation" minOccurs="0"/>
        <element ref="dadx:implements" minOccurs="0"/>
        <element ref="dadx:result_set_metadata" minOccurs="0"
                maxOccurs="unbounded"/>
        <element ref="dadx:operation" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
 ...
  <element name="implements">
    <annotation>
      <documentation>
        Defines the namespace and location of a set of WSDL bindings
        defined elsewhere. This information is imported into the
        WSDL document generated for this Web Service.
      </documentation>
    </annotation>
    <complexType>
      <attribute name="namespace" type="anyURI" use="required"/>
      <attribute name="location" type="anyURI" use="required"/>
    </complexType>
  </element>
...
</schema>
```

図 30. エレメント <implements>

以下の例は DADX ファイルで <implements> タグを使用する方法を示します。

```
<?xml version="1.0"?>

<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"

  xmlns:xsd="http://www.w3.org/2001/XMLSchema"

  xmlns:xhtml="http://www.w3.org/1999/xhtml">

<documentation>

  Provides queries for part order information at myco.com.

  This Web Service is compliant with the Part Ordering Industry
   Association standard.

  </documentation>


<implements namespace="http://www.poia.org/PartOrders.wsdl"

  location="http://www.poia.org/PartOrders.wsdl"/>


<operation name="findAll">

<documentation%gt;

Returns an order with its complete details.

</documentation>

    ...


  </operation>
...
```

図 31. implements タグを使用する DADX ファイルの例

関連概念:

- 101 ページの『DADX ファイルからの WSDL』
- 132 ページの『Web サービス記述言語』

Web サービス・プロバイダーを使用する動的データベースの照会

動的照会サービスを使用することによって、アプリケーション・データを選択、挿入、および更新する照会をランタイムに動的に作成および実行することや、デプロイメント時に事前定義される照会を実行する代わりにストアード・プロシージャーを呼び出すことができます。

Web アプリケーションは、Web サービス・インターフェースを使用してデータベースにアクセスし、使用可能な表および列に関する情報を取り出すことができます。さらに、アプリケーションは表を照会したり、Web サービスを使用してデータベース内のデータを変更することができます。 Web アプリケーションは、データベースに対してデータ定義言語アクション (表の作成など) を実行することもできます。

Web サービス・プロバイダーの動的照会サービスを使用することによって、Web アプリケーションの柔軟性を増すことができます。

WORF は、動的照会サービス・タグ (<DQS/>) が入った DADX ファイルから、次の 2 つのスタイルの Web サービス記述言語ファイル (WSDL) を生成できます。

- 文書指向の情報スタイルを使用する WSDL ファイル
- プロシージャー指向の情報スタイル (RPC) を使用する WSDL ファイル

生成されるスタイルはグループ・レベルで定義されます。これは、group.properties ファイルに `useDocumentStyle=true` が存在するかどうかによります。 Web サービス記述言語の情報スタイルについて詳しくは、ブラウザーで Web サービス記述言語の仕様を参照してください。 WSDL ファイルには、サービス、ポート、および定義情報が含まれます。 DADX ファイル内の動的照会タグは、静的 DADX 関数に影響を与えません。

アプリケーションを実行するまで照会の検索条件がわからない場合には、動的照会サービスの使用を考慮してください。

Web サービス・プロバイダーの動的照会コンポーネントは、一般的に以下のカテゴリー別に定義される Web サービス操作をサポートします。

メタデータの入手
データベースに存在する表、およびそれらの表の列情報を検索することができます。

DDL の実行
CREATE TABLE ステートメントを発行することができます。

DML の実行
SELECT、INSERT、UPDATE、および DELETE ステートメント、さらにストアード・プロシージャーを実行する CALL ステートメントを発行することができます。

サーバー管理者は、group.properties ファイルを使ってグループに特定のユーザー ID とパスワード設定を定義することにより、特定のデータベースへのアクセスを制御します。さらに、管理者は個別の WORF インスタンスを作成して、データベースへのアクセスを処理することもできます。

関連概念:

- 101 ページの『DADX ファイルからの WSDL』
- 37 ページの『Web サービス・プロバイダーのフィーチャー』
- 38 ページの『DADX ファイルを使った Web サービス・プロバイダー操作』
- 132 ページの『Web サービス記述言語』
- 108 ページの『動的照会サービス - 照会例』

関連タスク:

- 76 ページの『group.properties ファイルのカスタマイズ』

関連資料:

- 114 ページの『Web サービス・プロバイダーにおける動的照会サービス操作』

Web サービス・プロバイダーの一部としての動的データベース照会の構成および実行

動的照会サービスを使用すると、ストアード・プロシージャーを作成して実行し、ランタイムに以前にデプロイした Web サービスにアクセスするデータベース照会をサブミットできます。 Document Access Definition Extension (DADX) ファイルで、すべてのデータベース照会を定義する必要はなくなりました。動的照会は、group.properties ファイルの情報に基づき、グループ・レベルで、またはグループ・ディレクトリーの範囲内で実行できます。

前提条件:

- 動的 Web 照会を実行するグループの *group.properties* ファイルが存在することを確認してください。
- Web アプリケーションは、Web サービス記述言語 (WSDL) 文書で定義される Web サービス操作ごとに、ターゲット・データベースへの接続を確立する必要があります。
- WSDL でインポート定義を定義するのでない限り、XML スキーマ記述ファイル *db2WebRowSet.xsd* が、 Web アプリケーションのコンテキスト・ルートに含まれるようにする必要があります。ファイル *db2WebRowSet.xsd* は、 *dxxworf.zip* ファイルに含まれています。

制約事項:

- アプリケーションで、Web サービス・プロバイダーの動的照会サービスを使用する場合、そのアプリケーションでは、カーソルを使用したり、サーバーの状態を想定した操作を実行したりすることはできません。 1 度の照会で結果を獲得する必要があります。
- 動的照会サービス操作を示す XML タグ (<DQS/>) は、同じファイル内で DADX 固有の Web サービス定義と共存させることはできません。

手順:

Web サービス・プロバイダーを使用して、DB2 Universal Database で動的照会を実行するよう Web サービス環境を準備するには、次のようにします。

1. XML タグ <DQS/> を含む DADX ファイルを作成します。

   このタグにより、グループは動的照会を実行できます。 DADX ファイルでは、他のタグは必要ありません。
2. ファイルを、動的照会を実行するグループのディレクトリーに保管します。
3. WSDL を使用し、アプリケーションのクライアントを開発します。このクライアントには、少なくとも以下の情報が必要です。
   - グループ名
   - DADX ファイルの名前 (*mydqs.dadx* など)
   - Web サービス操作 (`getTables` など)
4. 受け入れられているいずれかの DQS 操作 (getTables 操作など) を発行するよう、クライアントを変更します。
5. `getTables` 操作を発行するクライアントを実行します。

照会の結果は、表の行と列を記述するメタデータと、表に含まれるデータです。SQL ステートメントは、自動コミット・モードで実行されます。クライアントは、他のグループで動的照会サービスを呼び出すこともできます。変更する必要のある情報は、エンドポイント URL だけです。しかし、クライアントは、RPC スタイル WSDL か文書スタイル WSDL のいずれかの場合にのみ互換性を持ちます。 RPC スタイル WSDL で定義された動的照会サービス・クライアントは、文書スタイル WSDL を使用するグループのために使用することはできません。

関連概念:

- 132 ページの『Web サービス記述言語』
- 108 ページの『動的照会サービス - 照会例』
- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』
- 69 ページの『Web サービスのグループの定義』

関連資料:

- 88 ページの『単純な DADX ファイル』

動的照会サービス - 照会例

例: DADX ファイル

次の例では、DADX ファイルは *mydqs.dadx* という名前です。ファイル mydqs.dadx は、動的照会を実行するグループのディレクトリーの中にあります。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx">
    <documentation>
        This is optional documentation about DQS
    </documentation>
      <DQS/>
</DADX>
```

例: ブラウザーからの動的照会の実行

以下の例は、ブラウザーから実行できる単純な動的照会です。このステートメントをアプリケーションに組み込むこともできます。 Web サービス・プロバイダーでの動的照会に関する必要な情報は、**太字**で印刷されています。この例での Web サービス操作は `executeQuery` です。この操作に関連したパラメーターは `queryInput` です。ステートメントは、次のように `employee` 表の `lastname` 列のすべての行を取り出します。

```
http://localhost:9080/services/<group_name>
  /somefile.dadx/executeQuery?queryInputParameter
     =select%20lastname%20from%20employee
```

この例では、完全な SOAP エンベロープではなく GET バインディング要求が発行されます。 GET、POST、および SOAP バインディングについての詳細は、『GET、POST、および SOAP バインディングによる Web サービスへのアクセス』の項を参照してください。以下の出力は executeQuery 操作によるものであり、db2WebRowSet スキーマ定義によって定義されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:executeQueryResponse
    xmlns:ns1="http://schemas.ibm.com/db2/dqs">
  <queryOutputParameter>
   <db2WebRowSet
       xmlns="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<metadata>
 <column-count>14</column-count>
 <column-definition>
  <column-index>1</column-index>
  <nullable>0</nullable>
  <column-name>EMPNO</column-name>
  <column-precision>6</column-precision>
  <column-scale>0</column-scale>
  <column-type>CHAR</column-type>
  <column-type-name>CHAR</column-type-name>
  <xml-type>string</xml-type>
 </column-definition>
 <column-definition>
  <column-index>2</column-index>
  <nullable>0</nullable>
  <column-name>FIRSTNME</column-name>
  <column-precision>12</column-precision>
  <column-scale>0</column-scale>
  <column-type>VARCHAR</column-type>
  <column-type-name>VARCHAR</column-type-name>
  <xml-type>string</xml-type>
 </column-definition>
...
 <column-definition>
  <column-index>14</column-index>
  <nullable>1</nullable>
  <column-name>COMM</column-name>
  <column-precision>9</column-precision>
  <column-scale>2</column-scale>
  <column-type>DECIMAL</column-type>
  <column-type-name>DECIMAL</column-type-name>
  <xml-type>decimal</xml-type>
 </column-definition>
...
 <column-definition>
  <column-index>14</column-index>
  <nullable>1</nullable>
  <column-name>COMM</column-name>
  <column-precision>9</column-precision>
  <column-scale>2</column-scale>
  <column-type>DECIMAL</column-type>
  <column-type-name>DECIMAL</column-type-name>
  <xml-type>decimal</xml-type>
 </column-definition>
</metadata>
</data>
</db2WebRowSet>
</queryOutputParameter>
  </ns1:executeQueryResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

例: db2WebRowSet.xsd のインポート

グループに動的照会サービス DADX が含まれる場合、db2WebRowSet.xsd ファイルは、Web サービス・コンシューマーに対してアクセス可能でなければなりません。db2WebRowSet.xsd ファイルの場所が確実に分かるよう、group.imports ファイルは必要なスキーマ・ロケーションを定義します。以下に、db2WebRowSet.xsd をインポートする group.imports ファイルの例を示します。以下の例では、ローカル・グループ・ディレクトリーに db2WebRowSet.xsd ファイルがないことを想定しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<imports
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:import namespace="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
      schemaLocation="http://myServer.myCo.com/schemas/misc/ibm/db2WRS.xsd"/>
</imports>
```

group.imports ファイルが存在しない場合、WORF は、動的照会サービス用のデフォルトのインポート・エレメントだけを WSDL に生成します。この場合、WORF は db2WebRowSets.xsd ファイルが以下の場所にあると想定します。

```
http://<server>:<port>/<contextRoot>/db2WebRowSet.xsd
```

例: getTables

以下に、getTables 操作の例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
 <SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns0:getTables
      xmlns:ns0="http://schemas.ibm.com/db2/dqs">
     <tablesInputParameter>
       <tablesInputData>
        <schemaPattern>MSCHENK</schemaPattern>
        <tableNamePattern>EMPLOYEE</tableNamePattern>
       </tablesInputData>
      </tablesInputParameter>
    </ns0:getTables>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

例: getColumns

以下に、getColumns 操作の例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
 <SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SOAP-ENV:Body>
    <ns0:getColumns
      xmlns:ns0="http://schemas.ibm.com/db2/dqs">
      <columnsInputParameter>
        <columnsInputData>
          <schemaPattern>MSCHENK</schemaPattern>
          <tableNamePattern>EMPLOYEE</tableNamePattern>
          <columnNamePattern>EMPNO</columnNamePattern>
        </columnsInputData>
      </columnsInputParameter>
    </ns0:getColumns>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

例: executeQuery

以下の照会の例では、表 employee からすべての行を取り出し、いくつかのパラメーターを指定します。

```
<?xml version="1.0" encoding="UTF-8"?>
 <SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
     <ns0:executeQuery
       xmlns:ns0="http://schemas.ibm.com/db2/dqs">
      <queryInputParameter>
       select * from employee
      </queryInputParameter>
      <extendedInputParameter>
      <properties>
         <loginInfo>
           <userid>userid</userid>
           <password>some_password</password>
         </loginInfo>
         <readOnly>true</readOnly>
         <isolationLevel>READ_UNCOMMITTED</isolationLevel>
         <escapeProcessing>true</escapeProcessing>
         <startAtRow>4</startAtRow>
         <fetchSize>80</fetchSize>
         <maxFieldSize>20</maxFieldSize>
         <maxRows>100</maxRows>
         <queryTimeout>2000</queryTimeout>
      </properties>
     </extendedInputParameter>
   </ns0:executeQuery>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

例: executeUpdate

以下の例は、動的照会サービスの更新ステートメントを示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
   <ns0:executeUpdate
     xmlns:ns0="http://schemas.ibm.com/db2/dqs">
     <queryInputParameter>
        update bo_events set OBJECTEVENTID=&'testestest&'
     </queryInputParameter>
     <extendedInputParameter>
       <properties/>
     </extendedInputParameter>
   </ns0:executeUpdate>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

次の例は、戻される応答文書を示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
```

```
  <ns1:executeUpdateResponse
   xmlns:ns1="http://schemas.ibm.com/db2/dqs">
    <updateOutputParameter xsi:type="xsd:int">
        1
    </updateOutputParameter>
   </ns1:executeUpdateResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

例: executeCall

この要求の例では、ストアード・プロシージャー multipleResultSets が呼び出されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns0:executeCall
      xmlns:ns0="http://schemas.ibm.com/db2/dqs">
    <callInputParameter>
      <callInputData>
          <spName>
             multipleResultSets
          </spName>
          <parameters>
            <parameter>
              <inParam>
                 <kind>IN</kind>
                 <type>string</type>
                 <value>000130</value>
              </inParam>
            </parameter>
            <parameter>
              <inParam>
                <kind>INOUT</kind>
                <type>string</type>
                <value>000130</value>
              </inParam>
             </parameter>
             <parameter>
               <outParam>
                 <kind>OUT</kind>
                 <type>string</type>
               </outParam>
             </parameter>
          </parameters>
       </callInputData>
    </callInputParameter>
   <extendedInputParameter>
    <properties/>
   </extendedInputParameter>
  </ns0:executeCall>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

以下に、出力例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
```

```
<ns1:executeCallResponse
   xmlns:ns1="http://schemas.ibm.com/db2/dqs">
<callOutputParameter>
<dqs:callOutputData
   xmlns:dqs="http://schemas.ibm.com/db2/dqs/types/soap">
 <dqs:outputResultSequences>
 <db2WebRowSet
   xmlns="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <metadata>
    <column-count>5</column-count>
    <column-definition>
      <column-index>1</column-index>
      <nullable>0</nullable>
      <column-name>DEPTNO</column-name>
      <column-precision>3</column-precision>
      <column-scale>0</column-scale>
      <column-type>CHAR</column-type>
      <column-type-name>CHAR</column-type-name>
      <xml-type>string</xml-type>
    </column-definition>
      ...
    <column-definition>
      <column-index>5</column-index>
      <nullable>1</nullable>
      <column-name>LOCATION</column-name>
      <column-precision>16</column-precision>
      <column-scale>0</column-scale>
      <column-type>CHAR</column-type>
      <column-type-name>CHAR</column-type-name>
      <xml-type>string</xml-type>
 </column-definition>
</metadata>
<data>
 <row>
  <column>A00</column>
  <column>
     SPIFFY COMPUTER SERVICE DIV.
  </column>
  <column>000010</column>
  <column>A00</column>
  <column xsi:nil="true"/>
 </row>
...
 <row>
   ...
 </row>
</data>
</db2WebRowSet>
</dqs:outputResultSequences>
   <dqs:outputParameterSequences>
    <dqs:callOutputParam>
     <position>2</position>
     <type>string</type>
     <value>xxxxxx</value>
    </dqs:callOutputParam>
    <dqs:callOutputParam>
     <position>3</position>
     <type>string</type>
     <value>This is the value of name3</value>
    </dqs:callOutputParam>
    </dqs:outputParameterSequences>
    </dqs:callOutputData>
   </callOutputParameter>
  </ns1:executeCallResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

例: execute

以下の例では、1 つの列を持つ表を作成します。

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <ns0:execute
    xmlns:ns0="http://schemas.ibm.com/db2/dqs">
     <queryInputParameter>
       create table temptable(in varchar(500))
     </queryInputParameter>
     <extendedInputParameter>
       <properties/>
     </extendedInputParameter>
  </ns0:execute>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

以下は、execute 操作からの出力です。

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
   <ns1:executeResponse
     xmlns:ns1="http://schemas.ibm.com/db2/dqs">
    <executeOutputParameter>
    <dqs:executeOutputData
       xmlns:dqs="http://schemas.ibm.com/db2/dqs/types/soap">
      <resultsPresent>false</resultsPresent>
     <dqs:outputResultSequences>
     </dqs:outputResultSequences>
    </dqs:executeOutputData>
    </executeOutputParameter>
  </ns1:executeResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

関連概念:

- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』

関連タスク:

- 107 ページの『Web サービス・プロバイダーの一部としての動的データベース照会の構成および実行』

関連資料:

- 114 ページの『Web サービス・プロバイダーにおける動的照会サービス操作』

Web サービス・プロバイダーにおける動的照会サービス操作

以下の表では、DB2 Web サービス・プロバイダーでサポートされる動的照会操作について説明します。

表 7. メタデータ検索のための操作

Web サービス操作	説明
getTables **tablesInputParameter** 入力; タイプ = tablesInputData (118 ページの表 11 を参照) **tablesOutputParameter** 出力; タイプ = db2WebRowSet	指定したカタログおよびスキーマ内の表の説明を検索します。カタログの名前、スキーマの名前、および表の名前などです。スキーマを入力パラメーターとして使用する場合、Java Database Connectivity ではスキーマの大文字小文字を区別する必要がある場合があります。
getColumns **columnsInputParameter** 入力; タイプ = columnsInputData (119 ページの表 12 を参照) **columnsOutputParameter** 出力; タイプ = db2WebRowSet	指定したカタログ、スキーマ、および表内の列の説明を検索します。スキーマを入力パラメーターとして使用する場合、Java Database Connectivity ではスキーマの大文字小文字を区別する必要がある場合があります。

表 8. 照会およびストアード・プロシージャーを実行するための操作

操作	説明
executeQuery **queryInputParameter** 必要な入力; タイプ = string **extendedInputParameter** 必要な入力; タイプ = properties (116 ページの表 9 を参照) **queryOutputParameter** 出力; タイプ = db2WebRowSet	データベース・サーバー上で単一の SQL SELECT ステートメントを発行し、単一の結果セットを戻します。
executeUpdate **queryInputParameter** 必要な入力; タイプ = string **extendedInputParameter** 必要な入力; タイプ = properties (116 ページの表 9 を参照) **updateOutputParameter** 出力; タイプ = int	データベース・サーバー上で単一の INSERT、UPDATE、 DELETE ステートメントを発行し、完了コードを戻します。
executeCall **callInputParameter** 入力; タイプ = callInputData **extendedInputParameter** 入力; タイプ = properties (116 ページの表 9 を参照) **callOutputParameter** 出力; タイプ = callOutputData	データベース・サーバー上で単一のストアード・プロシージャーを呼び出し、出力パラメーターのセットおよび結果セットのシーケンスを戻します。

表 8. 照会およびストアード・プロシージャーを実行するための操作 (続き)

操作	説明
execute **queryInputParameter** 必要な入力; タイプ = string **extendedInputParameter** 必要な入力; タイプ = properties (表 9 を参照) **executeOutputParameter** 出力; タイプ = executeOutputData	データベース・サーバー上で単一の SQL ステートメントを発行し、完了コードおよび結果セットのシーケンスを戻します。

表 9 にリストされているオプショナル・パラメーターは、 115 ページの表 8 にリストされている操作と一緒に使用できます。

表 9. 拡張パラメーター用の入力データ・タイプ

プロパティー・タイプ	説明
loginInfo • userid • password	loginInfo には、アクセス制御を目的としてデータベースに渡されるユーザー ID が含まれます。さらに、アクセス制御を目的としてデータベースに渡されるユーザー ID と関連したパスワードも含まれます。これらのプロパティーのタイプはストリングです。ユーザー ID を指定する場合には、パスワードも指定する必要があります。
readOnly	これにより、Web アプリケーションはデータベースを読み取り専用で使用することを指定できます。これはバイナリー形式であり、true または false のいずれかになります。
escapeProcessing	これにより、Web アプリケーションは照会ストリングのエスケープ処理を制御することができます。エスケープ・スキャンが使用可能 (true) にされている場合、ドライバーは SQL をデータベースに送信する前にエスケープ置換を実行します。これはバイナリー形式であり、true または false のいずれかになります。デフォルト値は true です
fetchSize	あらゆる取り出し操作に関して、取り出されて Web アプリケーションに戻される行数を指定します。これは整数タイプです。デフォルト値は 0 です。
maxFieldSize	列内の最大バイト数の限界を、指定されたバイト数に設定します。この値は、列値に関して戻すことができる最大バイト数です。これは整数タイプです。
maxRows	取り出して Web アプリケーションに戻す最大行数を指定します。これは整数タイプです。 maxRows パラメーターを指定しない場合、最大 1000 行を戻すことができます。
startAtRow	これにより、Web アプリケーションは結果セットの中で、指定した行数をスキップできます。これは整数タイプです。
queryTimeout	これにより、Web アプリケーションは照会用のタイムアウト値を指定できます。ドライバーがステートメント・オブジェクトの実行を待機する秒数を設定します。限度を超えると、例外が発生します。値 0 (秒) は、ドライバーが待機する秒数が無制限であることを示しています。

表 9. 拡張パラメーター用の入力データ・タイプ (続き)

プロパティー・タイプ	説明
isolationLevel	これにより、Web アプリケーションは照会の分離レベルを制御することができます。 • READ_UNCOMMITTED • READ_COMMITTED • REPEATABLE_READ • SERIALIZABLE • NONE

表 10. *callInputParameter* の入力データ・タイプ

callInputData type	説明
spName タイプ: string	呼び出すストアード・プロシージャーの名前。このパラメーターは必須です。
schema タイプ: string	ストアード・プロシージャーのスキーマ。このパラメーターはオプションです。パラメーターを指定しない場合、値は現在のスキーマになります。

表 10. callInputParameter の入力データ・タイプ (続き)

callInputData type	説明
parameters タイプ: パラメーターのシーケンス。それぞれのシーケンスは inParam または outParam のいずれかで構成されます。 **inParam** タイプは次のように定義されます。 • **kind**:「IN」または「INOUT」のいずれか • **type**: パラメーターのタイプ (int、またはストリングなど) • **value**: パラメーターの値 **outParam** タイプは次のように定義されます。 • **kind**:「IN」または「INOUT」のいずれか • **type**: パラメーターのタイプ	ストアード・プロシージャーは、3 種類のパラメーター IN、OUT、および INOUT を持つことができます。このパラメーター・タイプは拡張可能です。任意の数の inParam および outParam タイプを自由に組み合わせることができます。 Web アプリケーションは、自分が呼び出そうとしているストアード・プロシージャーにパラメーターが必要かどうかを認識しておかなければなりません。パラメーターが必要な場合には、パラメーター数およびパラメーター・タイプを認識していなければなりません。 非サポートのデータ・タイプの 1 つをストアード・プロシージャーがストアード・プロシージャー・パラメーターとして取る場合、このストアード・プロシージャーを WORF を通して実行することはできません。 WORF は、ストアード・プロシージャー・パラメーター用にいくつかの XML タイプを受け入れます。パラメーターは、組み込み SQL データ・タイプに対応しています。 92 ページの表 6 には、サポートされているタイプが説明されています。 以下の値のいずれかを使用して、入力パラメーターを NULL に設定できます。 **absent** 入力パラメーターの <value/> タグが提供されません。 **nil = true** タグは属性 nil とマークされ、この属性は <value xsi:nil="true"/> のように true に設定されます。 入力パラメーターの順序は、ストアード・プロシージャーの予期する順序と同じでなければなりません。

表 11. tablesInputData タイプの入力データ・タイプ

tablesInputData タイプ	説明
catalogPattern タイプ =「string」 **schemaPattern** タイプ =「string」 **tableNamePattern** タイプ =「string」	それぞれのパターン値はオプションです。値を指定しない場合、それはデフォルトでブランクの値になります。それぞれの値の説明および性質は JDBC で指定されています。指定する catalogPattern、schemaPattern、および tableNamePattern を満たす表のリストを戻すには、getTables Web サービス操作を使用します。

表 11. *tablesInputData タイプの入力データ・タイプ (続き)*

tablesInputData タイプ	説明

例 (説明を簡単にするために、ネーム・スペース定義などをここに示さないことに注意してください):
`<tablesInputData>`
` <catalogPattern></catalogPattern>`
` <schemaPattern>`**`userSchema`**
` </schemaPattern>`
` <tableNamePattern>`**`EMPLOYEE`**
` </tableNamePattern>`
`</tablesInputData>`	

表 12. *columnsInputData タイプの入力データ・タイプ*

columnsInputData タイプ	説明
catalogPattern タイプ =「string」 **schemaPattern** タイプ =「string」 **tableNamePattern** タイプ =「string」 **columnNamePattern** タイプ =「string」	それぞれのパターン値はオプションです。値を指定しない場合、それはデフォルトでブランクの値になります。それぞれの値の説明および性質は JDBC で指定されています。指定するカタログ・ストリング・パターン、schemaPattern、表名、および columnNamePattern を満たす列のリストを受け取るには、getColumns Web サービス操作を使用します。

例 (説明を簡単にするために、ネーム・スペース定義などをここに示さないことに注意してください):
`<columnsInputData>`
` <catalogPattern></catalogPattern>`
` <schemaPattern>`**`userSchema`**
` </schemaPattern>`
` <tableNamePattern>`**`EMPLOYEE`**
` </tableNamePattern >`
` <columnNamePattern>`**`LASTNAME`**
` </columnNamePattern>`
`</columnsInputData>`	

表 13. *callOutputData タイプの出力データ・タイプ*

callOutputData タイプ
outputResultSequences ストアード・プロシージャーによってタイプ db2WebRowSet として戻されるすべての結果セットのシーケンスが含まれます。
outputParameterSequences: callOutputParam (ストアード・プロシージャーから戻された、kind=INOUT または kind=OUT のいずれかになるパラメーター) のシーケンスが含まれます。

表 13. *callOutputData* タイプの出力データ・タイプ (続き)

callOutputData タイプ	
callOutputParam	戻されるパラメーター。以下のものが含まれます。 • <position> タイプ: int - ストアード・プロシージャーのパラメーター・リストにあるパラメーターの位置。 • <type> タイプ: string - XML データ・タイプ (タイプ情報について詳しくは、callInputData を参照。) • <value> タイプ: any - パラメーターの値。 出力パラメーターが NULL の場合、absent メソッドが使用されます。 結果には次のものが含まれます。 `<value xsi:nil="true"/>`
例 (説明を簡単にするために、ネーム・スペース定義などをここに示さないことに注意してください):	

```
<callOutputParameter>
   <dqs:callOutputData
      xmlns:dqs="http://schemas.ibm.com/db2/dqs/types/soap">
      <dqs:outputResultSequences>
         <db2WebRowSet
           xmlns="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <metadata>
             ....
         </db2WebRowSet>
      </dqs:outputResultSequences>
      <dqs:outputParameterSequences>
            <dqs:callOutputParam>
                <position>1</position>
                <type>short</type>
                <value>123</value>
            </dqs:callOutputParam>
            <dqs:callOutputParam>
                <position>2</position>
                <type>int</type>
                <value xsi:nil="true" />
            </dqs:callOutputParam>
         </dqs:outputParameterSequences>
      </dqs:callOutputData>
</callOutputParameter>
```

表 14. executeOutputData タイプの出力データ・タイプ

executeOutputData タイプ	説明
resultsPresent タイプ =「ブール」 **outputResultSequences** db2WebRowSet の 0 個以上のオカレンス	**execute** Web サービス操作が、結果セットを戻す照会ストリングを指定して呼び出された場合、ブールは execute と outputResultSequences のそれぞれに結果セットの 1 つが含まれることを示します。

例 (説明を簡単にするために、ネーム・スペース定義などをここに示さないことに注意してください):

```
<executeOutputParameter>
   <dqs:executeOutputData
      xmlns:dqs="http://schemas.ibm.com/db2/dqs/types/soap">
      <resultsPresent>true</resultsPresent>
      <dqs:outputResultSequences>
         <db2WebRowSet
           xmlns="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <metadata>
             ....
         </db2WebRowSet>
      </dqs:outputResultSequences>
   </dqs:executeOutputData>
</executeOutputParameter>
```

関連概念:

- 108 ページの『動的照会サービス - 照会例』
- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』

関連タスク:

- 107 ページの『Web サービス・プロバイダーの一部としての動的データベース照会の構成および実行』

関連資料:

- 121 ページの『db2WebRowSet』

db2WebRowSet

動的照会サービス・タイプ db2WebRowSet は、SQL 結果セットから XML 文書を生成する汎用的な方法を規定するものです。スキーマ文書 db2WebRowSet.xsd には、特定の結果セットに関するメタデータ情報は収められていません。このスキーマ文書に収められているのは、結果セット・メタデータに関する汎用的なメタデータ情報だけです。実際の結果セットに関するメタデータや結果セットのデータは XML インスタンス文書に収められます。インスタンス文書には、メタデータ・セクションとデータ・セクションがあります。

メタデータ・セクション

メタデータ・セクションには、結果に含まれるすべての列に関するメタデータ情報が収められます。

最初のフラグは、列カウント・タグです。このタグには、結果セット中に列がいくつあるか、という情報が収められます。その後には、各列につき列定

義タグが 1 つずつあります。列定義には、以下のメタデータ情報が含まれます。

表 15. 列定義メタデータ

タグ名	説明
<column-index>	結果セット中での列の位置を表す数値 (1 から始まる)。
<nullable>	列に NULL を入れることができるかを示す数値。NULL を入れることができれば 1、できなければ 0。
<column-name>	列の名前。
<column-precision>	このタグの記述は、SQL データ・タイプに応じて異なります。 SQL データ・タイプが文字であれば、column-precision は長さになります。 SQL データ・タイプが 10 進数であれば、column-precision は精度になります。
<column-scale>	10 進数データのタイプ。
<column-type>	Java Database Connectivity のタイプに対応して変わる。BINARY、VARBINARY、CHAR、VARCHAR など。
<column-type-name>	DB2 Universal Database のデータ・タイプ名。 CHAR FOR BIT DATA、 VARCHAR FOR BIT DATA、 CHAR、 VARCHAR など。
<xml-type>	XML データ・タイプ。 base64binary、 int、 string、 dateTime など。

データ・セクション

データ・セクションには、結果セットの実データが収められます。各行は、1 つの行タグにマッピングされます。行タグには、結果セット中の列と同じ数の列タグが収められ、列インデックスによって順序が決められます。行タグの中には、XML データ・タイプの形で実データが収められます。

表 16. データ・タイプ・マッピングに関する規則

DB2 UDB データ・タイプ <column-type-name>	JDBC データ・タイプ <column-type>	XML データ・タイプ <xml-type>
BLOB	BLOB	base64Binary
CLOB	CLOB	string
LONGVARCHAR	LONGVARCHAR	string
VARCHAR	VARCHAR	string
CHAR	CHAR	string
CHAR FOR BIT DATA	BINARY	base64Binary
VARCHAR FOR BIT DATA	VARBINARY	base64Binary
LONGVARCHAR FOR BIT DATA	LONGVARBINARY	base64Binary
DATE	DATE	date
TIME	TIME	time
TIMESTAMP	TIMESTAMP	dateTime
-	BOOLEAN	boolean
-	BIT	boolean

表 16. データ・タイプ・マッピングに関する規則 (続き)

DB2 UDB データ・タイプ <column-type-name>	JDBC データ・タイプ <column-type>	XML データ・タイプ <xml-type>
TINYINT	TINYINT	int
SMALLINT	SMALLINT	int
INTEGER	INTEGER	int
BIGINT	BIGINT	int
DOUBLE	DOUBLE	double
FLOAT	FLOAT	double
REAL	REAL	float
DECIMAL	DECIMAL	decimal
NUMERIC	NUMERIC	decimal
DATALINK	DATALINK	string
-	ARRAY	anyType
DISTINCT	DISTINCT	string
-	JAVA_OBJECT	string
-	NULL	string
-	OTHER	string
-	STRUCT	string
-	REF	string
	その他	string

db2WebRowSet.xsd ファイルの内容は以下に示すようなものになります。このファイルのデフォルト・ロケーションは、<contextRoot> ディレクトリーです。

```
<?xml version="1.0" encoding="UTF-8"?>
 <xs:schema targetNamespace="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:db2wrs="http://schemas.ibm.com/db2/dqs/db2WebRowSet"
  elementFormDefault="qualified">
   <xs:element name="db2WebRowSet">
  <xs:complexType>
   <xs:sequence>
      <xs:element ref="db2wrs:metadata"/>
    <xs:element name="data">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="row"
                             minOccurs="0"
                             maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element ref="db2wrs:column"
                                    minOccurs="0"
                                    maxOccurs="unbounded"/>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
```

図 32. db2WebRowSet.xsd (1/2)

```
<xs:element name="column"
               type="xs:anyType"
               nillable="true"/>
 <xs:element name="metadata">
  <xs:complexType>
   <xs:sequence>
    <xs:element
                name="column-count"
                type="xs:string" />
    <xs:choice>
     <xs:element name="column-definition"
                      minOccurs="0"
                      maxOccurs="unbounded">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="column-index"
                             type="xs:string" />
        <xs:element name="nullable"
                             type="xs:string" />
        <xs:element name="column-name"
                             type="xs:string" />
        <xs:element name="column-precision"
                             type="xs:string" />
        <xs:element name="column-scale"
                             type="xs:string" />
        <xs:element name="column-type"
                             type="xs:string" />
        <xs:element name="column-type-name"
                             type="xs:string" />
        <xs:element name="xml-type"
                             type="xs:string"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:choice>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

図 32. db2WebRowSet.xsd (2/2)

関連概念:

- 108 ページの『動的照会サービス - 照会例』
- 105 ページの『Web サービス・プロバイダーを使用する動的データベースの照会』

関連タスク:

- 107 ページの『Web サービス・プロバイダーの一部としての動的データベース照会の構成および実行』

関連資料:

- 114 ページの『Web サービス・プロバイダーにおける動的照会サービス操作』

Web サービス・プロバイダーの検査とテスト (WORF)

このセクションでは、 DB2 に添付されている SAMPLE データベースを使用して、WORF の概要を提供します。始めに、自分のアプリケーション・サーバーの実行準備ができていることを確認してください (詳細については、 44 ページの『UNIX、Windows、z/OS、および OS/390 での WebSphere Application Server 向け

の Web サービス・プロバイダーの構成』および 60 ページの『UNIX および Windows での Apache Jakarta Tomcat 用の Web サービス・プロバイダーの構成』を参照)。 SAMPLE データベースにアクセスする Web サービスを作成する準備ができました。このシナリオは、WORF サンプルが *services* という名前の Web アプリケーションとしてインストールされていることを前提としています。また、ご使用のアプリケーション・サーバー上に *services* が構成されていることも前提としています。

Web サービス・アプリケーションのテスト - シナリオ

WORF では、Document Access Definition Extension (DADX) ファイルを使用して、Web サービスを作成することができます。 DADX ファイルには、Web サービスを作成するために必要な情報が含まれており、 DAD ファイルを参照できます。このシナリオは *HelloSample.dadx* という名前の簡単な DADX ファイルを使用します。

```
<?xml version="1.0" encoding="UTF-8"?>
 <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx">

    <operation name="listDepartments">
      <query>
        <SQL_query>SELECT * FROM DEPARTMENT</SQL_query>
      </query>
    </operation>
 </DADX>
```

図 33. 簡単な DADX ファイル: HelloSample.dadx

OS/390® および z/OS™ プラットフォームの場合、すでにインストールされているサンプル DEPARTMENT 表に対応するように表の名前を変更する必要があるかもしれません。この表のデフォルト名は DSN8710.DEPT です。

DADX ファイルに定義されている Web サービスをデプロイするには、ディレクトリー dxx_sample のグループ *db2sample* によって定義されているディレクトリーにあるアプリケーション・サーバーにそれをコピーします。

HelloSample.dadx は `listDepartment` という名前の単一操作を含む Web サービスを定義します。この操作は、DEPARTMENT 表の内容をリストします。子タグ <query> は操作のタイプを指定します。

関連概念:

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』
- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 40 ページの『Web サービス・プロセスの概要』

関連タスク:

- 127 ページの『Web サービスのテスト』

Web サービスのテスト

次の作業を完成させて、Web サービスをテストできます。

手順:

1. WORF サンプルが、ディレクトリー ¥WEB-INF¥classes¥groups¥dxx_sample にインストールされていることを確認します。
2. サンプル・アプリケーションを、WebSphere Application Server (WAS) などのサーバーにデプロイしたことを確認します。
3. ブラウザー・ウィンドウをオープンし、次の URL を入力してテストを開始します。

   ```
   http://<your WebAppServer>/services/db2sample/ivt.dadx/TEST
   ```

   your WebAppServer ID は自分の Web サーバー構成によって決まることを忘れないでください。アドレスをタイプすると、次の自動で生成される文書とテスト・ページが表示されます。

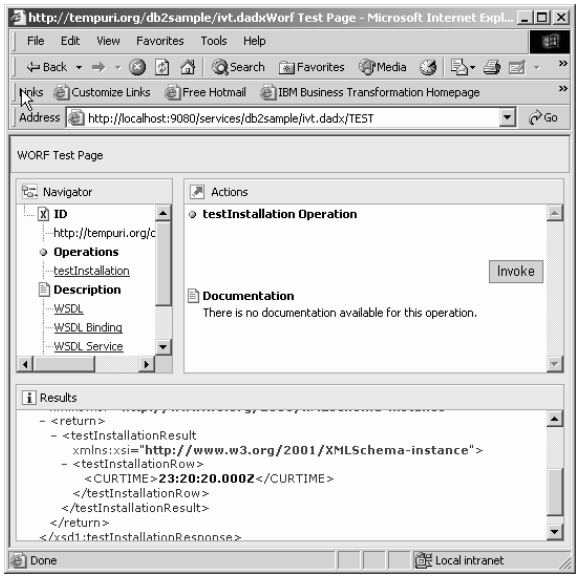


図 34. WORF テスト・ページ

4. `listDepartments` 操作をテストします。
   a. **「メソッド (Methods)」**ペインの `listDepartments` リンクをクリックします。
   b. **「入力 (Inputs)」** ペインの**「呼び出し (Invoke)」**プッシュボタンをクリックします。

      「結果 (Result)」ペインに操作の Extensible Markup Language (XML) 結果が表示されます。

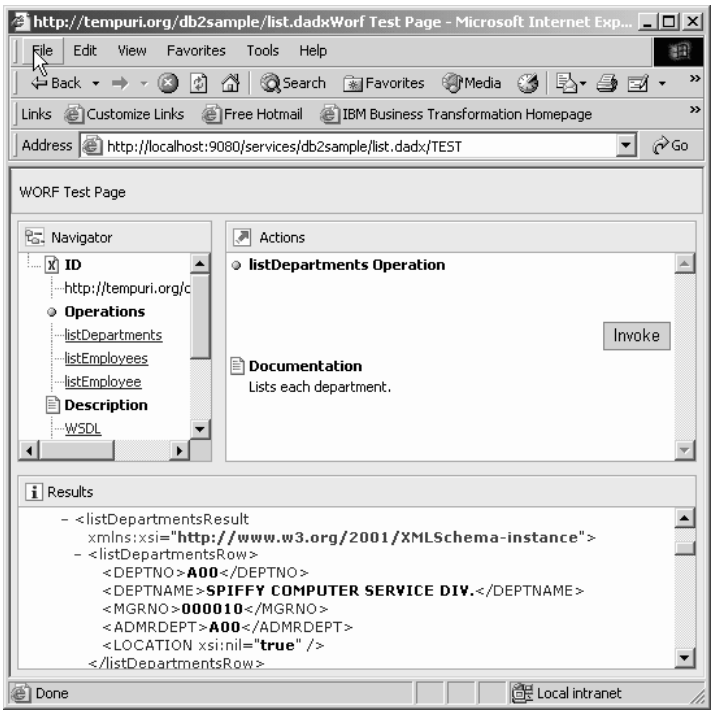


図 35. 照会の結果

関連概念:

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』
- 126 ページの『Web サービス・アプリケーションのテスト - シナリオ』
- 149 ページの『Web アプリケーションのインストール』

関連タスク:

- 48 ページの『WebSphere Application Server Version 5 以降 (Windows および UNIX 用) での WORF のインストールおよび移行』

GET、POST、および SOAP バインディングによる Web サービスへのアクセス

Web オブジェクト・ランタイム・フレームワーク (WORF) テスト・ページは、Web サービスの単純な Hypertext Markup Language (HTML) クライアントの役割を果たし、 Hypertext Transfer Protocol (HTTP) POST バインディングを使用します。HTTP GET バインディングと SOAP バインディングを使用することによって、Web サービスにアクセスできます。 HTTP GET および POST バインディングを含む `listDepartments` 操作を呼び出すことができます。以下の例は GET または POST バインディングの基本的な構文を示しています。

```
http://server:port/contextRoot/group/dadx_file/operationName
```

WORF サンプルがインストールされていれば、以下の URL を入力して GET 要求を発行できます。

```
http://<yourWebAppServer:9080>/services/db2sample/HelloSample.dadx/listDepartments
```

その中で *<yourWebAppServer>* によって指定される *localhost* ポート番号は、現在使用しているマシンごとに異なります。 WORF `listDepartments` 操作は Extensible Markup Language (XML) 応答を戻しますが、これはファイルに保管できます。 HTTP 応答は、GET の場合も POST の場合も同じです。

```
<?xml version="1.0" ?>
<xsd1:listDepartmentsResponse
  xmlns:xsd1="http://schemas.ibm.com/sample/department.dadx/XSD"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<return>
 <xsd1:listDepartmentsResult
  xmlns:xsd1="http://schemas.ibm.com/sample/department.dadx/XSD"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<listDepartmentsRow>
  <DEPTNO>A00</DEPTNO>
  <DEPTNAME>SPIFFY COMPUTER SERVICE DIV.</DEPTNAME>
</listDepartmentsRow>
<listDepartmentsRow>
  <DEPTNO>B01</DEPTNO>
  <DEPTNAME>PLANNING</DEPTNAME>
</listDepartmentsRow>
<listDepartmentsRow>
  <DEPTNO>C01</DEPTNO>
  <DEPTNAME>INFORMATION CENTER</DEPTNAME>
</listDepartmentsRow>
<listDepartmentsRow>
  <DEPTNO>D01</DEPTNO>
  <DEPTNAME>DEVELOPMENT CENTER</DEPTNAME>
</listDepartmentsRow>
          ...
<listDepartmentsRow>
  <DEPTNO>E21</DEPTNO>
  <DEPTNAME>SOFTWARE SUPPORT</DEPTNAME>
</listDepartmentsRow>
</xsd1:listDepartmentsResult>
</return>
</xsd1:listDepartmentsResponse>
```

図 36. XML 応答文書

GET バインディング要求は要求文書を送信しません。代わりに、照会ストリング内の必要なパラメーターすべてを URL に追加するようにします。照会ストリングを URL に追加するには、疑問符 (?) を使用します。パラメーター=値の対同士の区切り文字は & 記号です。特殊文字は URL エンコードする必要があります。以下に挙げるのは、疑問符 (?) と区切り文字によって照会ストリングを使用する GET バインディング要求の例です。

```
http://server:port/contextRoot/group/dadx/
   operationName?param1=abc&param2=1234&param3=thi&20is&20a&20parameter
```

以下に挙げるのは、動的照会サービスの例です。これは GET バインディング要求です。

```
http://localhost:9080/services/db2sample/dqs.dadx/
   executeQuery?queryInputParameter=select+*+from+employee&extendedInputParameter=
      %3Cproperties%3E%0D%0A%3C%2Fproperties%3E%0D%0A
```

POST バインディングは HTTP POST 要求を発行します。 POST バインド要求は要求文書を送信します。文書には要求パラメーターが入っていますが、パラメーターは XML フォーマットではありません。要求文書を作成するのは Web ブラウザーなどの HTTP クライアント・アプリケーションです。通常、Web ブラウザーは入力フォームから要求文書を作成します。これがサーバーに送られます。以下に典型的な POST バインド要求の構文を挙げます。

```
http://server:port/contextRoot/group/dadx/operationName
```

以下に挙げるのは、動的照会サービスの例です。これは POST バインディング要求です。

```
http://localhost:9080/services/db2sample/dqs.dadx/executeQuery
```

照会は GET バインディング要求の場合と同じです。ただし、疑問符 (?) の後の情報が URL の一部にではなく、要求文書の中にある点が異なります。以下の例では、コンテンツ・タイプは www-urlencoded です。

```
queryInputParameter=
   select+*+from+employee&extendedInputParameter=
%3Cproperties%3E%0D%0A%3C%2Fproperties%3E%0D%0A
```

動的照会サービス要求に対する GET および POST 応答を以下に示します。

```
<?xml version="1.0"?>
<xsd1:executeQueryResponse
  xmlns:xsd1="http://schemas.ibm.com/db2/dqs/types/soap"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <queryOutputParameter>
      ...
   </queryOutputParameter>
</xsd1:executeQueryResponse>
```

HTTP GET および POST バインディングは一般に、その他の HTTP GET および POST 要求と同じです。 HTTP GET バインディングは、操作への入力パラメーターをすべて URL に追加します。しかし、HTTP POST バインディングは要求本体のパラメーターを送信します。

関連概念:

- 37 ページの『Web サービス・プロバイダーのフィーチャー』
- 154 ページの『Apache SOAP の構成』
- 130 ページの『SOAP バインディング』

関連資料:

- 114 ページの『Web サービス・プロバイダーにおける動的照会サービス操作』
- 145 ページの『Web サービスのサンプル - PartOrders.dadx』

SOAP バインディング

Simple Object Access Protocol (SOAP) バインディングでは、Hypertext Transfer Protocol (HTTP) POST も使用しますが、 SOAP は操作名、入力パラメーター、およびその他の情報を Extensible Markup Language (XML) 要求本体として送信します。

SOAP 要求バインディングは、HTTP 経由で SOAP 要求を発行します。 SOAP は、要求メッセージと応答メッセージのためのメッセージ・プロトコルとして使用されます。 SOAP 要求は、要求メッセージと応答メッセージの表示方法を指定します。 SOAP バインディング要求は、特定のスキーマに従った XML 文書です。

SOAP は、HTTP POST 要求の最上位で運用されます。 HTTP はトランスポート・プロトコルですが、SOAP はメッセージ・プロトコルです。クライアント・アプリケーション側は、SOAP 要求文書の作成方法を知っている必要があります。パラメーターの定義およびフォーマット、および他の情報は、個々の Web サービス記述言語 (WSDL) 文書で定義されます。

次の URL を使用して、 SOAP バインディングにアクセスしてください (*your WebAppServer* ID はご使用の Web サーバー構成によって決まることを忘れないでください)。

```
http://<your WebAppServer>/services/db2sample/HelloSample.dadx/SOAP
```

要求の中には操作名はありません。情報は SOAP 要求文書に収められています。

次の RPC スタイル用の例は、SOAP バインディングを使用した動的照会サービスです。

```
 <SOAP-ENV:Envelope
   1 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <SOAP-ENV:Body>
     2 <ns0:executeQuery
      3 xmlns:ns0="http://schemas.ibm.com/db2/dqs">
      4 <queryInputParameter>
            select * from employee
     </queryInputParameter>
     <extendedInputParameter>
          <properties/>
     </extendedInputParameter>
   </ns0:executeQuery>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP 要求には、次の重要な情報が含まれています。

1. SOAP エンベロープ。
2. 操作名。
3. 実際の要求文書。 WORF 環境では、これは実際の Web サービス SOAP 要求となっている XML 文書です。
4. パラメーター。

SOAP の応答は次のとおりです。

```
<?xml version='1.0' encoding='UTF-8'?><SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
   <ns1:executeQueryResponse
      xmlns:ns1="http://schemas.ibm.com/db2/dqs"
          SOAP-ENV:encodingStyle="http://xml.apache.org/xml-soap/literalxml">
    <queryOutputParameter>
       ...
```

```
    </queryOutputParameter>
   </ns1:executeQueryResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP-ENV:Body タグ内に含まれる情報は、実際の応答文書です。 WORF 環境では、応答文書は XML 文書です。これが実際の Web サービス SOAP 応答になります。

注: Java™ および JavaScript™ クライアント用に SOAP バインディングを使用することを考慮してください。 WebSphere® Studio には Java Web サービス・クライアントを生成する機能があります。

関連概念:

- 128 ページの『GET、POST、および SOAP バインディングによる Web サービスへのアクセス』
- 164 ページの『Web サービスのコンシューマー機能』
- 40 ページの『Web サービス・プロセスの概要』

関連タスク:

- 127 ページの『Web サービスのテスト』

関連資料:

- 114 ページの『Web サービス・プロバイダーにおける動的照会サービス操作』

Web サービス記述言語

Web サービス・プロバイダーは、Web サービス記述言語 (WSDL) 文書によって記述されます。 Web サービスのかぎとなるのは、Web サービス記述言語文書です。

WSDL とは、Web サービスをエンドポイントまたはポートの集合として記述する XML 文書のことです。エンドポイントとは、指定したインターフェースの関連バインディングに応じて Web サービスにアクセスできる、アドレス可能なロケーションのことです。 1 つの Web サービスは複数のエンドポイントを持つことができます。 WSDL で記述されたエンドポイントはメッセージを使って機能します。 WSDL バインディングは、どのようにサービスがメッセージング・プロトコル (特に SOAP メッセージング・プロトコル) にバインドするかを記述します。 WSDL SOAP バインディングは、文書指向スタイルのバインディングまたはプロシージャー指向 (RPC) スタイルのバインディングのいずれかになります。また、SOAP バインディングは、エンコードして使用することも、リテラルで使用することもできます。

133 ページの図 37 が示すとおり、Web サービス・プロバイダーはサービスをインプリメントし、UDDI などのサービス・ブローカーにインターフェースを公開します。サービス・リクエスターは、サービス・ブローカーを使用して Web サービスを検索します。リクエスターは、サービスを検出すると、サービス・プロバイダーにバインドして Web サービスを使用できるようにします。リクエスターは、プロバイダーとの間で SOAP (Simple Object Access Protocol) メッセージを交換してサービスを呼び出します。

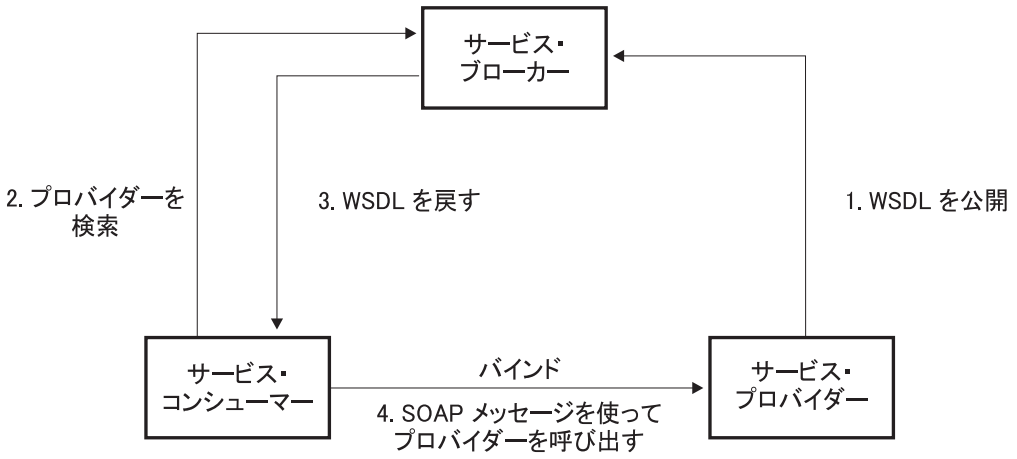


図 37. サービス指向アーキテクチャーとしての Web サービス

SOAP 仕様は XML ベースのメッセージのレイアウトを定義します。 SOAP メッセージは SOAP エンベロープに入れられます。エンベロープは、オプションの SOAP ヘッダーと、必須の SOAP 本体で構成されます。 SOAP ヘッダーには、暗号化情報または認証情報などの、実メッセージに関する情報を含めることができます。 SOAP 本体には実メッセージが含まれます。さらに SOAP 仕様には、プログラム言語バインディング用のデフォルトのエンコードが含まれています。これを SOAP エンコードと言います。

WSDL 文書には、1 つ以上の Web サービスを含めることができます。サービスは、バインディングを持つ 1 つ以上のポートで構成されます。 WSDL 文書は 1 つ以上のポート・タイプを持つことができます。 1 つのポート・タイプには、抽象的な入出力メッセージを持つ 1 つ以上の操作があります。バインディングとは、プロトコルまたはデータ・フォーマット情報を、メッセージ、操作、または portType といった抽象エンティティーと関連付ける処理のことを指します。バインディングは、特定のポート・タイプの具体的なプロトコルおよびデータ・フォーマット仕様を作成します。ポートとは、バインディングであるエンドポイントおよび Web アドレスです。

135 ページの図 38 にある例は、株価情報を提供する簡単なサービスの WSDL 定義について説明しています。 Web サービスは、`GetLastTradePrice` と呼ばれる単一の操作をサポートします。サービスは、HTTP 上で SOAP 1.1 プロトコルを使用してデプロイされます。要求は、ストリング・データ・タイプであるチッカー・シンボルを入力として読み取り、浮動データ・タイプである価格を戻します。この例で示されているタイプは XML スキーマ定義です。 XSD ファイルを使用して、DB2® Universal Database 表内の表および列を、ご使用の Web サービスと関連付けることができます。

<soap:binding> エレメントで指定された WSDL スタイルは文書スタイルです。 <soap:operation> エレメントは、操作全体に関する情報を提供します。 <soap:operation> エレメントでのスタイル属性は、操作が RPC 指向 (パラメーターおよび戻り値を含むメッセージ) であるか、それとも文書指向 (文書を含むメッセージ) であるかを示します。この属性の値は、SOAP メッセージ本文を構成する方法にも影響します。属性を指定しない場合と、属性値はデフォルトで、<soap:binding> エレメントで指定された値になります。 IBM® DB2 Information Integrator Web サ

ービス・プロバイダーには、RPC スタイルを使用するよう設定されているサンプルが含まれています。新規のアプリケーションの場合は、インターオペラビリティーが最大になるよう、文書スタイルを使用する必要があります。バージョン 8.2 の場合、Web サービス・プロバイダーは、 RPC スタイルをリテラル使用およびエレメント・ノードとともに使用するのではなく、 RPC スタイルをリテラル使用およびタイプ・ノードとともに使用します。ただし、SOAP メッセージについては、以前のリリースの Web サービス・プロバイダーから変更がありません。

完全な例および WSDL 仕様は、W3C サイト (http://www.w3.org/TR/2001/NOTE-wsdl-20010315) にあります。

```
<?xml version='1.0'?>
<definitions name='StockQuote'
...


<types>
       <schema targetNamespace='http://example.com/stockquote.xsd'
              xmlns='http://www.w3.org/2000/10/XMLSchema'>
           <element name='TradePriceRequest'>
              <complexType>
                  <all>
                      <element name='tickerSymbol' type='string'/>
                  </all>
              </complexType>
           </element>
           <element name='TradePrice'>
              <complexType>
                  <all>
                      <element name='price' type='float'/>
                  </all>
              </complexType>
           </element>
       </schema>
    </types>

<message name='GetLastTradePriceInput'>
...
</message>

    <portType name='StockQuotePortType'>
        <operation name='GetLastTradePrice'>
           <input message='tns:GetLastTradePriceInput'/>
           <output message='tns:GetLastTradePriceOutput'/>
        </operation>
    </portType>


    <binding
       name='StockQuoteSoapBinding'
       type='tns:StockQuotePortType'>
        <soap:binding
          style='document'
            transport='http://schemas.xmlsoap.org/soap/http'/>
        <operation name='GetLastTradePrice'>
           <soap:operation
             soapAction='http://example.com/GetLastTradePrice'/>
           <input>
               <soap:body use='literal'/>
           </input>
           <output>
               <soap:body use='literal'/>
           </output>
        </operation>
    </binding>

   <service name='StockQuoteService'>
        <documentation>My first service</documentation>
        <port name='StockQuotePort'
           binding='tns:StockQuoteBinding'>
           <soap:address
             location='http://example.com/stockquote'/>
        </port>
    </service>

</definitions>
```

図 38. WSDL の例

WSDL 文書には一定の構造があるため、 Web サービス開発者は、WSDL 文書の定義レベルか WSDL 文書のタイプ・レベルのいずれかにおいて、外部スキーマからのタイプを使用しなければならない場合があります。外部スキーマを使用する場合、インポートされたスキーマ定義を WSDL で使用することができます。 WORF は、WSDL 生成時に次の 2 つのタイプのインポートをサポートします。

WSDL の /definitions 範囲でのインポート

http://schemas.xmlsoap.org/wsdl/:import

WSDL の /definitions/type/schema 範囲でのインポート

http://www.w3.org/2001/XMLSchema:import

group.imports ファイルを使用してインポート定義を追加できます。 group.imports ファイルが Web サービス・グループ・ディレクトリーのリソース内に存在する場合、WORF は生成する WSDL に group.imports 情報を含めます。以下の例は、group.imports ファイルを示しています。

```
<?xml version='1.0' encoding='UTF-8'?>
<imports xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
         xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
   <wsdl:import namespace='http://some/namespace/1'
           location='schema1.xsd'/>
   <wsdl:import namespace='http://some/namespace/2'
           location='schema2.xsd'/>
   <xsd:import namespace='http://some/namespace/3'
           schemaLocation='schema3.xsd'/>
   <xsd:import namespace='http://some/namespace/4'
           schemaLocation='schema4.xsd'/>
</imports>
```

この例では、/definitions 範囲の 2 つのインポート (schema1.xsd および schema2.xsd) を定義しており、これが WSDL に追加されることになります。また、/definitions/types/schema 範囲の 2 つのインポート (schema3.xsd および schema4.xsd) を定義しており、これも WSDL に追加されることになります。上のファイルのインポート定義を含む、WORF が生成する WSDL の構造は次のとおりです。

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions>
  <wsdl:documentation xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
       xmlns='http://schemas.xmlsoap.org/wsdl/'>
     Documentation Text Node
  </wsdl:documentation>
  <import location='schema2.xsd'
            namespace='http://some/namespace/2'/>
  <import location='schema1.xsd'
            namespace='http://some/namespace/1'/>
  <types>
    <schema>
      <import namespace='http://some/namespace/4'
             schemaLocation='schema4.xsd'/>
      <import namespace='http://some/namespace/3'
             schemaLocation='schema3.xsd'/>
      <element name='executeQueryResponse'> .... </element>
      <element name='executeQuery'> .... </element>
    </schema>
  </types>
    ...
</definitions>
```

WORF サンプルがインストール済みで、services というアプリケーションがある場合、サービス用の Web サービス記述言語 (WSDL) 文書 *HelloSample.dadx* を要求することができます。 WSDL を要求するには、次の URL を使用します。その中で <*yourWebAppServer*> によって指定される *localhost* ポート番号は、現在使用しているマシンごとに異なります。

```
http://<yourWebAppServer>/services/db2sample/HelloSample.dadx/WSDL
```

WORF は自動的に DADX から WSDL 文書を生成します。

関連概念:

- 101 ページの『DADX ファイルからの WSDL』

関連タスク:

- 76 ページの『group.properties ファイルのカスタマイズ』
- 127 ページの『Web サービスのテスト』

UDDI ビジネス・レジストリー

Web サービスを Universal Discovery, Description, and Integration (UDDI) ビジネス・レジストリーに登録します。推奨されている方法は、WSDL 文書をサービス・インスタンス文書とバインディング文書に分割することです。 UDDI および最良実例については、UDDI 最良実例を参照してください。

サービス・インスタンス文書はサービスのデプロイ元のアドレスを含んでおり、バインディング文書をインポートします。多数のサービス・インスタンスが 1 つの共通バインディング文書を参照する場合があります。そのバインディング文書を UDDI に再使用可能 tModel として登録します。 tModel は、Web サービスの使用に関する情報です。

URL を指定して WSDL サービス・インスタンス文書を要求します。その中で <*yourWebAppServer*> によって指定される *localhost* ポート番号は、現在使用しているマシンごとに異なります。

```
http://<yourWebAppServer>/services/db2sample/HelloSample.dadx/WSDLservice
```

次の URL を指定して WSDL バインディング文書を要求します:

```
http://<yourWebAppServer>/services/db2sample/HelloSample.dadx/WSDLbinding
```

関連概念:

- 40 ページの『Web サービス・プロセスの概要』

関連タスク:

- 127 ページの『Web サービスのテスト』

XML スキーマ定義

XML スキーマは、Web サービス・インターフェースに使用されるデータ・タイプを定義します。サービスの XML スキーマ定義を URL によって要求します。その中で <*yourWebAppServer*> によって指定される *localhost* ポート番号は、現在使用しているマシンごとに異なります。

```
http://<yourWebAppServer>/services/db2sample/HelloSample.dadx/XSD
```

WORF は、図 39 の例に似た XML スキーマ・ファイルを生成します。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="http://localhost:8080/services/sample/HelloSample.dadx/XSD"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://localhost:8080/services/sample/HelloSample.dadx/XSD">
  <element name="listDepartmentsResult">
    <complexType>
      <sequence>
        <element maxOccurs="unbounded" minOccurs="0" name="listDepartmentsRow">
          <complexType>
            <sequence>
              <element name="DEPTNO" type="string"/>
              <element name="DEPTNAME" type="string"/>
              <element name="MGRNO" nillable="true" type="string"/>
              <element name="ADMRDEPT" type="string"/>
              <element name="LOCATION" nillable="true" type="string"/>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```

図 39. XML スキーマ定義ファイル

DB2® XML エクステンダーは文書タイプ定義 (DTD) を使用して XML 文書のスキーマを定義できるので、 WORF ランタイムは自動的に DTD を XML スキーマに変換します。たとえば、DTD *order.dtd* が XML 文書を定義する場合、次の URL を使用して XML スキーマへの変換を要求します。

```
http://<yourWebAppServer>/services/db2sample/order.dtd/XSD
```

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 126 ページの『Web サービス・アプリケーションのテスト - シナリオ』
- 138 ページの『Web サービス・プロバイダーに存在する Web サービス』

関連資料:

- 273 ページの『付録 C. DADX ファイルの XML スキーマ』

Web サービス・プロバイダーに存在する Web サービス

Web アプリケーションのディレクトリー内、または Web サービスのグループ内には、ネットワーク上で使用できる Web サービスが多数含まれている可能性があります。これらの Web サービスを使用する前に、それらを検索して情報を取得する必要があります。 Web Services Inspection Language (WSIL) を使用すると、この検索プロセスを容易に行えます。

Web Services Inspection Language 文書

DB2® Web サービスは、必要な Web サービスの操作を検索する方法を提供します。 WORF を使用することにより、アプリケーション内、またはグループ内で使用可能なすべての Web サービスを検査します。検査生成プログラムは、使用可能な Web サービスのリストとなる XML 文書を生成します。そのリストは、生成プログラムをグループ・ディレクトリーから実行した場合には、そのグループ・ディレクトリー・レベルでの Web サービスのレポートとなります。そのリストは、生成プログラムをアプリケーション・ディレクトリーから実行した場合には、そのアプリケーション・ディレクトリー・レベルでの Web サービスのレポートとなります。ブラウザーから検査生成プログラムを実行するには、ブラウザーに *<your-Web-server>:9080/<context_root_name>/inspection.wsil* と入力します。

このトピックの例は、WORF サンプルと *services* という名前の WORF サンプル・アプリケーションを基にしています。

以下の例では、*services* という名前の Web アプリケーションから使用可能な Web サービスのリストを作成します。

```
http://localhost:9080/services/inspection.wsil
```

アプリケーション・レベルにおける inspection.wsil は、以下のようになります。

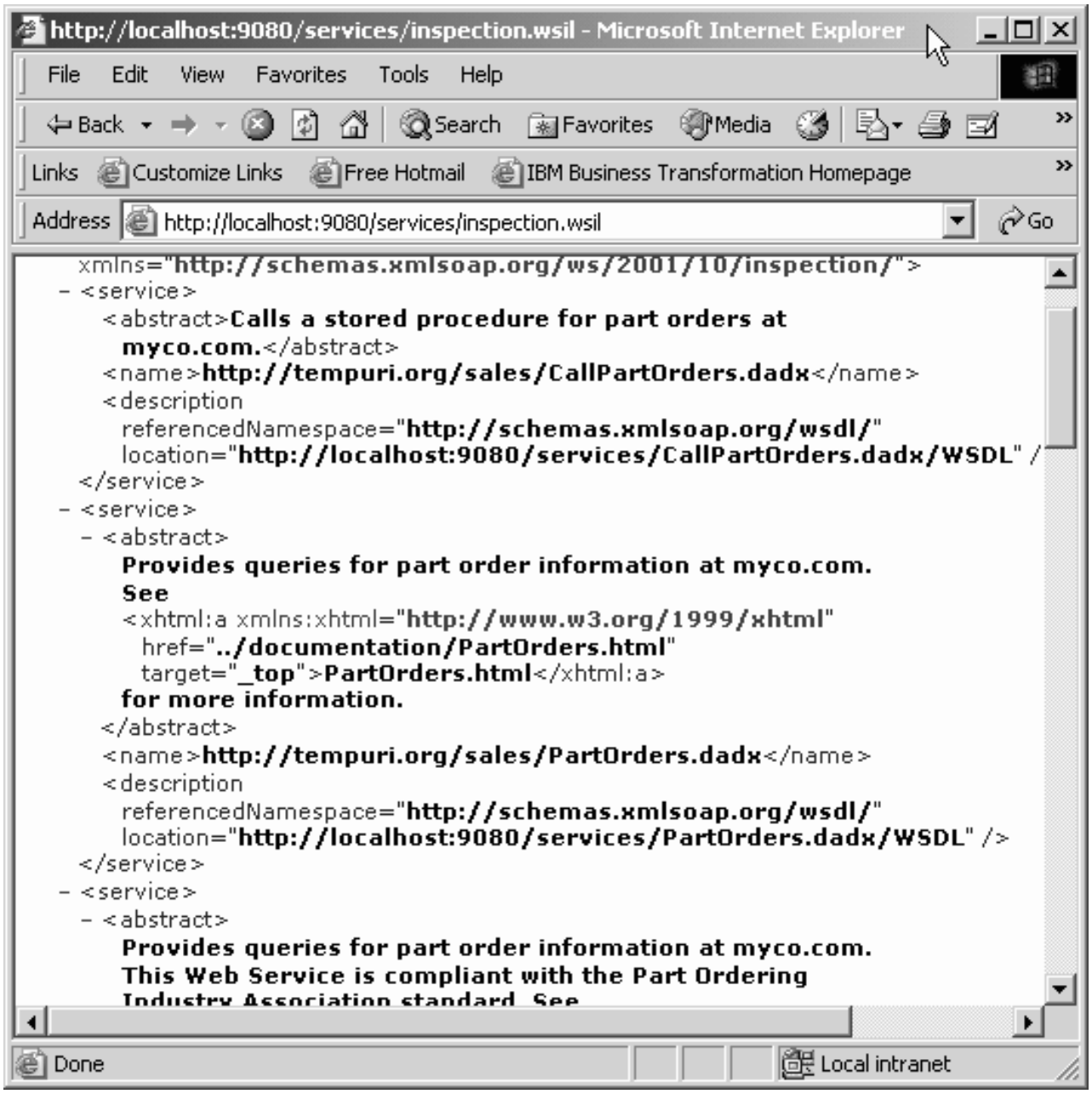


図 40. アプリケーション・レベルにおける WSIL

グループ・レベルにおける inspection.wsil は、以下のようになります。

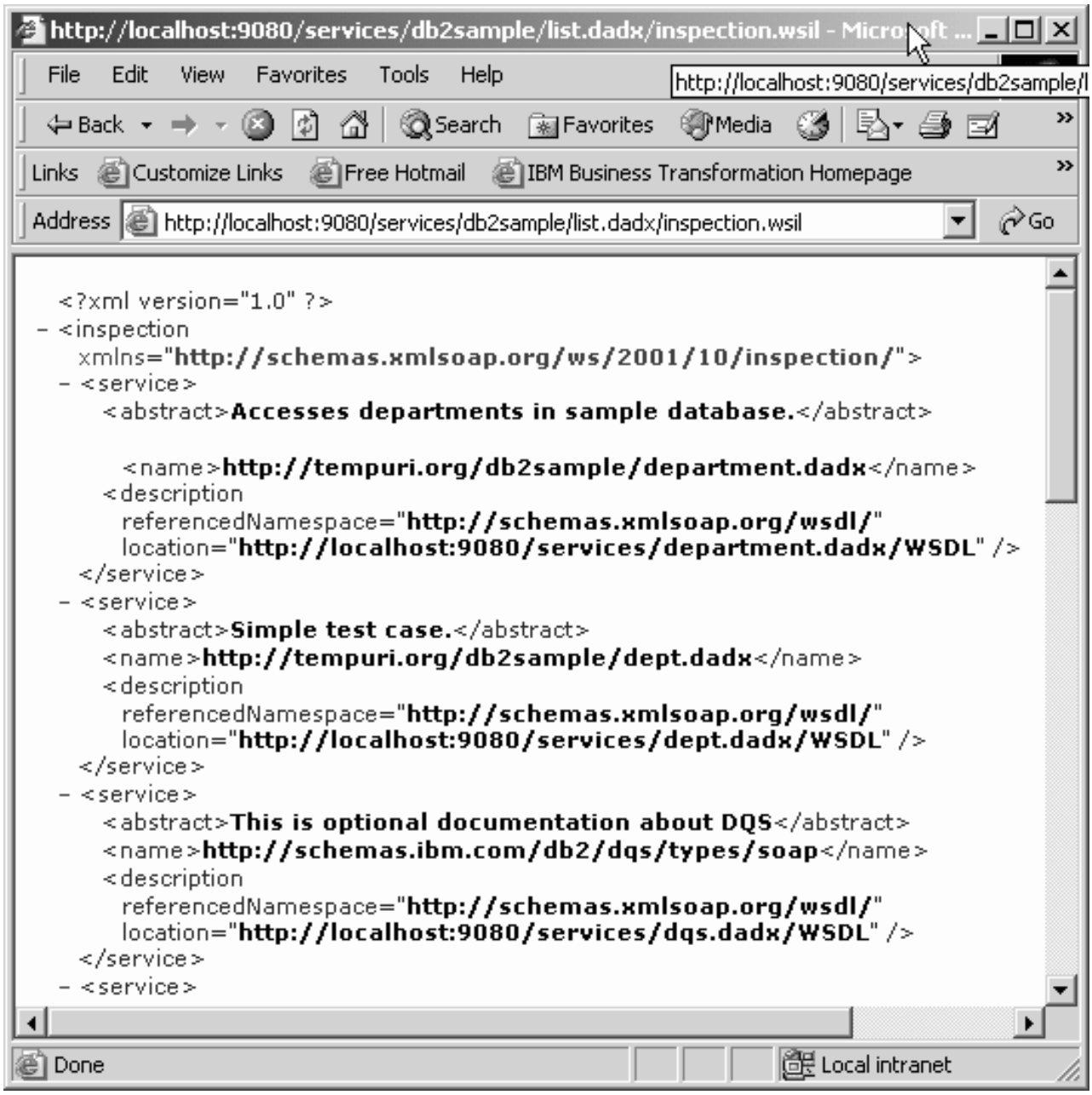


図 41. グループ・レベルにおける WSIL

生成する WSIL に Web サービスが確実に含まれるようにするには、Web サービスが以下の基準に準拠していなければなりません。

- Web サービスを記述する DADX ファイルが有効でなければならない。
- Web サービスに属する group.properties ファイルを定義しなければならない。
- web.xml ファイルには、グループを識別する適切なサーブレット・マッピングが入っていなければならない。

WSIL 用の web.xml ファイル内のサーブレット・マッピングは、以下の例のようになります。

```
    <servlet>
        <servlet-name>wsil</servlet-name>
        <display-name>wsil</display-name>
        <servlet-class>
           com.ibm.etools.webservice.rt.wsil.servlet.WSILInvoker
        </servlet-class>
        <init-param>
          <param-name>soap-engine</param-name>
          <param-value>apache-axis</param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>wsil</servlet-name>
        <url-pattern>/inspection.wsil</url-pattern>
    </servlet-mapping>
```

Web サービスが呼び出されると、WORF は web.xml ファイルを読み取り、サーブレット情報やグループ情報など、サービスを実行する方法に関する情報を取得します。 web.xml ファイルにはサーブレット定義が含まれており、これによって WSIL の指定は正しい Web 操作を WORF サンプルに関連付けます。 WORF は、 Web アプリケーション・サーバーのグループ・ディレクトリー内の使用可能な Web サービスがすべて入っている WSIL 文書を動的に生成します。 Web アプリケーション・サーバーを開始した後で group.properties ファイルに Web サービスを追加する場合、 WSIL 生成プログラムを開始する前にサーバーを再始動する必要はありません。

Web サービスのリスト・ページ

生成できる別の種類の検査文書は、Web サービスのリスト・ページです。 Web サービスのリスト・ページは、 Web アプリケーション・サーバーのアプリケーション・ディレクトリーまたはグループ・ディレクトリー内の使用可能なすべての Web サービスのリストを含む HTML 文書です。このリストには、WORF サンプルおよびサービスの WSDL へのリンクも含まれています。ブラウザーに以下の URL を入力すると、このページにアクセスできます。

アプリケーション・レベル

```
<your-Web-server>:9080/<context_root_name>/LIST
```

グループ・レベル

```
<your-Web-server>:9080/<context_root_name>/<group name>/LIST
```

リスト・ページ用の web.xml ファイル内のサーブレット・マッピングおよび URL パターンは、以下の例のようになります。

```
<servlet>
        <servlet-name>list</servlet-name>
        <display-name>list</display-name>
        <servlet-class>
          com.ibm.etools.webservice.rt.list.servlet.ListInvoker
        </servlet-class>
        <init-param>
          <param-name>soap-engine</param-name>
          <param-value>apache-axis</param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
    </servlet>
```

```
<servlet-mapping>
     <servlet-name>list</servlet-name>
     <url-pattern>/LIST</url-pattern>
 </servlet-mapping>
```

グループ・レベルにおけるリスト・ページは、以下のようになります。

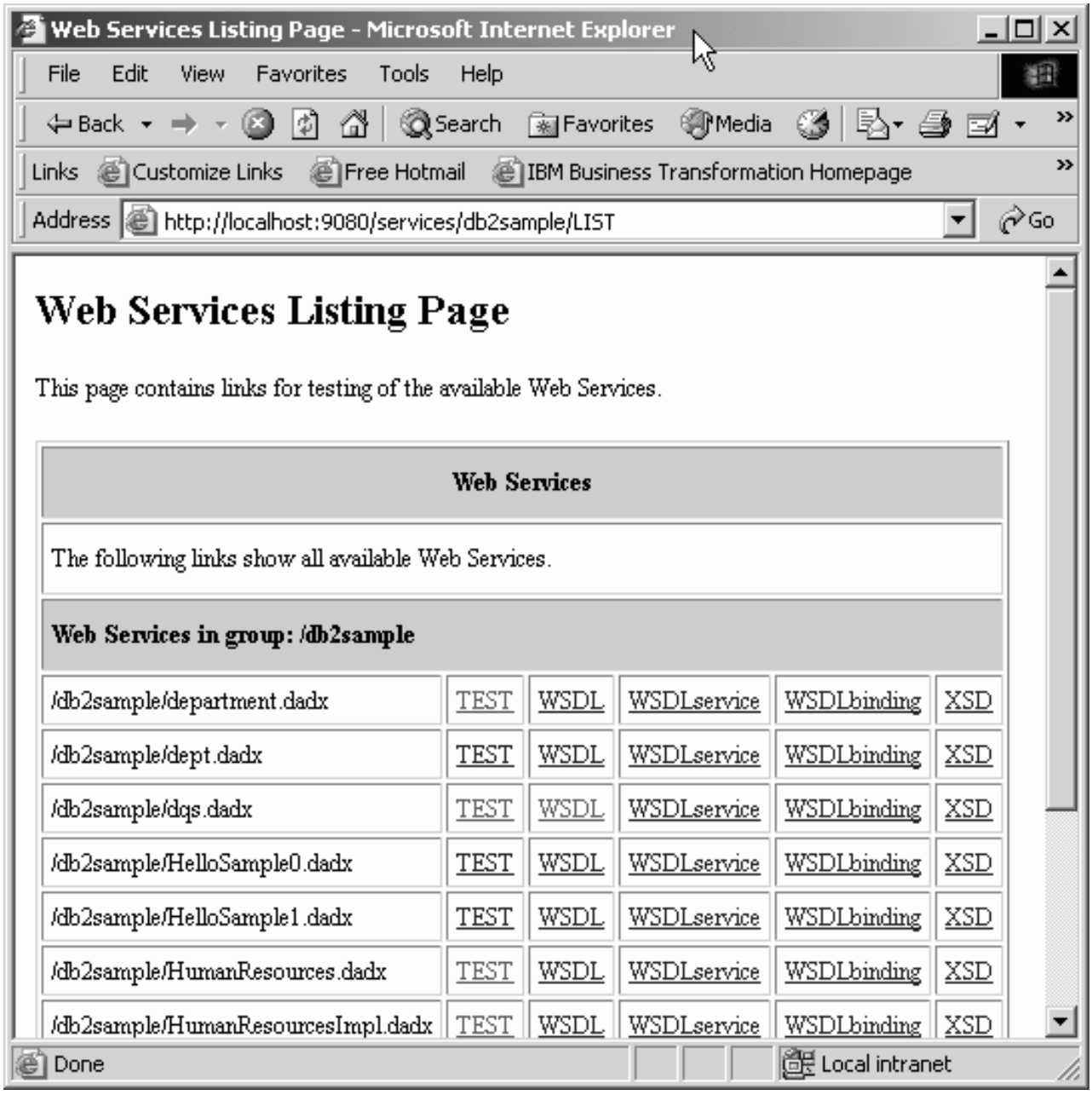


図 42. Web サービスのリスト・ページ

関連概念:

- 137 ページの『XML スキーマ定義』

関連タスク:

- 127 ページの『Web サービスのテスト』

- 70 ページの『web.xml ファイルおよび group.properties ファイルの定義』
- 48 ページの『WebSphere Application Server Version 5 以降 (Windows および UNIX 用) での WORF のインストールおよび移行』
- 50 ページの『WebSphere Application Server Version 5.1 以降 (Windows および UNIX 用) での WORF の例のデプロイ』

Web サービス文書

サービス全体または各操作のための文書を DADX に組み込むことができます。 図 43 は文書を追加する方法を例示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<DADX
  xmlns="http://schemas.ibm.com/db2/dxx/dadx">
  <documentation>
    Simple DADX example that accesses the SAMPLE database.
  </documentation>
  <operation name="listDepartments">
    <documentation>
      Lists the departments.
    </documentation>
    <query>
      <SQL_query>SELECT * FROM DEPARTMENT</SQL_query>
    </query>
  </operation>
</DADX>
```

図 43. HelloSample.dadx

文書には任意の有効な Extensible Markup Language (XML) を含めることができます。ブラウザーで適切に表示されるようにするため、XHTML を使用するようにします。 XHTML を使用する場合は、文書の XHTML ネーム・スペースを定義します。テスト・ページを要求すると、テスト・ページには文書も組み込まれます。

図 44. 文書を含む WORF テスト・ページ

関連概念:

- 126 ページの『Web サービス・アプリケーションのテスト - シナリオ』

関連資料:

- 88 ページの『単純な DADX ファイル』
- 79 ページの『DADX ファイルの構文』

Web サービスの自動再ロード

開発中には、DADX ファイルを頻繁に変更することもありえます。 WORF を使用すると、アプリケーション・サーバーを実行しながら DADX ファイルを変更することができ、WORF は新たに更新された DADX ファイルを自動的に再ロードします。自動再ロードのおかげで、DADX Web サービスの開発は、 Java™ Server Pages の開発と同じくらいシンプルになっています。 DADX Web サービスを実動サーバーにデプロイするときに、自動再ロードをオフにすることができます。

関連概念:

- 37 ページの『Web サービス・プロバイダーのフィーチャー』
- 40 ページの『Web サービス・プロセスの概要』

関連タスク:

- 76 ページの『group.properties ファイルのカスタマイズ』
- 127 ページの『Web サービスのテスト』

Web サービスのサンプル - PartOrders.dadx

このトピックの例では、dxx_sales_db と呼ばれるデータベース・サンプルを使用します。これは、資料の中で使用されるサンプル・データベースで、 DB2 XML エクステンダーおよび WORF に付属するサンプルです。 dxx_sales_db データベースは、部品注文についての情報を保管しています。

以下の条件に基づく注文を検索する Web サービスを提供しなければならないとします。

- すべての注文を検索する
- 指定した色の部品の注文をすべて検索する
- 価格が最低価格以上である注文すべてを検索する

以下の操作を含む *PartOrders.dadx* という DADX ファイルを作成します。

- findAll
- findByColor
- findByMinPrice

PartOrders.dadx ファイルを `services` Web アプリケーションにデプロイして、Web サービスを作成します。これは、WORF の dxx_sales_db インスタンスで構成されます。このファイルのデプロイメント・ロケーションは、*WEB-INF/classes/groups/dxx_sales_db/PartOrders.dadx* です。

Web サービスは、以下のプロトコルによるアクセスをサポートします。

- Hypertext Transfer Protocol (HTTP) GET

- HTTP POST
- HTTP SOAP

HTTP GET と POST は、Web ブラウザーからの簡単なアクセスに便利です。この場合、要求では application/x-www-form-urlencoded のコンテンツ・タイプを使用します。

たとえば、Web サービスをホスト www.mycompany.com にデプロイするとします。以下の URL は、HTTP GET を使用した Web サービスを呼び出します。

- http://www.mycompany.com /services/sales/PartOrders.dadx /findAll
- http://www.mycompany.com /services/sales/PartOrders.dadx /findByColor?color=red
- http://www.mycompany.com /services/sales/PartOrders.dadx /findByMinPrice?minprice=20000

この構文は、追加のパス情報として URL に、また照会ストリングとしてパラメーターにメソッドをエンコードします。これらの要求に対する応答には、text/xml のコンテンツ・タイプが含まれます。 HTTP POST の場合、照会ストリングを URL ではなく要求の本体に送信しますが、コンテンツ・タイプは application/x-www-form-urlencoded のままです。ここで、Transmission Control Protocol (TCP) トレース・ユーティリティーで取り込んだときの、 HTTP POST 要求の一例を紹介します。この例は、HTTP ヘッダーと本体の両方を示しています。

```
POST /services/sales/PartOrders.dadx/findByColor
HTTP/1.1
User-Agent: Java1.3.0
Host: localhost:9081
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-length: 12
color=red+++
```

DADX ファイルで定義された Web サービスは、自己記述式です。文書とテスト・ページ、WSDL 文書、および XML スキーマを動的に生成します。次の HTTP GET URL は、文書とテスト・ページを要求するものです。

```
http://www.mycompany.com/services
/sales/PartOrders.dadx/TEST
```

次の HTTP GET URL は、サービスの WSDL 記述を要求するものです。

```
http://www.mycompany.com/services
/sales/PartOrders.dadx/WSDL
```

HTTP SOAP の場合、POST を使用して SOAP エンベロープを次の URL へ送信することにより、サービスを呼び出します。

```
http://www.mycompany.com/services
/sales/PartOrders.dadx/SOAP
```

ただし、要求のコンテンツ・タイプは、application/x-www-form-urlencoded ではなく、 text/xml を使用します。次の例は、TCP モニターでトレースされる SOAP 要求です。これは、WebSphere Studio に組み込まれた要求、または Apache SOAP の一部である要求に似ています。この例には、HTTP ヘッダー情報と HTTP 本体が含まれます。

```
POST /services/sales/PartOrders.dadx/SOAP
HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 547
SOAPAction: "http://tempuri.org/sales/PartOrders.dadx"

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<ns1:findByColor xmlns:ns1="http://tempuri.org/sales/PartOrders.dadx" SOAP-
ENV:encodingStyle="http://xml.apache.org/xml-soap/literalxml">
<color xsi:type="xsd:string" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">red </color>
</ns1:findByColor>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

.

PartOrder DADX ファイル

PartOrders.dadx は、XML collection アクセス・メソッドを使用する <retrieveXML> 演算子を使用して、 3 つの操作すべてをインプリメントします。一般に、それぞれの操作では、異なる演算子およびアクセス・メソッドを使用できます。

```
<?xml version="1.0"?>
  <DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xhtml="http://www.w3.org/1999/xhtml">
    <documentation>
      Provides queries for part order information at myco.com.
      See <xhtml:a href="../documentation/PartOrders.html" target="_top">
       PartOrders.html</xhtml:a> for more information.
    </documentation>

    <operation name="findAll">
      <documentation>

        Returns all the orders with their complete details.
      </documentation>
      <retrieveXML>
         <DAD_ref>getstart_xcollection.dad</DAD_ref>
         <SQL_override>
         select o.order_key, customer_name, customer_email,
           p.part_key, color, quantity, price, tax, ship_id, date, mode
         from order_tab o, part_tab p,
           table(select substr(char(timestamp(generate_unique())),16)
             as ship_id, date, mode, part_key from ship_tab) s
         where p.order_key = o.order_key and s.part_key = p.part_key
         order by order_key, part_key, ship_id
         </SQL_override>
      </retrieveXML>
    </operation>
```

図 45. PortOrder.DADX ファイル (1/3)

```
<operation name="findByColor">
      <documentation>
Returns all  the orders that include one or
      more parts that have the specified
      color, and only shows the details for those parts.
</documentation>

      <retrieveXML>
        <DAD_ref>getstart_xcollection.dad</DAD_ref>
        <SQL_override>
         select o.order_key, customer_name, customer_email,
           p.part_key, color, quantity, price, tax, ship_id, date, mode
         from order_tab o, part_tab p,
           table(select substr(char(timestamp(generate_unique())),16)
             as ship_id, date, mode, part_key from ship_tab) s
         where p.order_key = o.order_key and s.part_key = p.part_key
           and color = :color
         order by order_key, part_key, ship_id
        </SQL_override>
        <parameter name="color" type="xsd:string"/>
      </retrieveXML>
    </operation>
```

図 45. PortOrder.DADX ファイル (2/3)

```
 <operation name="findByMinPrice">
      <:documentation>
Returns all the orders that include one or more
   parts that have a price greater than
  or equal to the specified minimum price,
and only shows the details for
  those parts.
</documentation >
      <retrieveXML>
        <DAD_ref>
           getstart_xcollection.dad
       </DAD_ref>
        <SQL_override>
         select o.order_key, customer_name, customer_email,
           p.part_key, color, quantity, price, tax, ship_id, date, mode
         from order_tab o, part_tab p,
           table(select substr(char(timestamp(generate_unique())),16)
             as ship_id, date, mode, part_key from ship_tab) s
         where p.order_key = o.order_key and s.part_key = p.part_key
           and p.price >= :minprice
         order by order_key, part_key, ship_id
        </SQL_override>
        <parameter name="minprice" type="xsd:decimal"/>
      </retrieveXML>
    </operation>

  </DADX>
```

図 45. PortOrder.DADX ファイル (3/3)

関連概念:

- 128 ページの『GET、POST、および SOAP バインディングによる Web サービスへのアクセス』

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』
- 126 ページの『Web サービス・アプリケーションのテスト - シナリオ』
- 130 ページの『SOAP バインディング』

関連タスク:

- 127 ページの『Web サービスのテスト』

Web アプリケーションのデプロイおよびテスト

これで、ビジネス・ゴールを達成するために、自分の設計から作業し、照会を構成し、アプリケーションを作成できます。次いで、プログラムを Web にデプロイします。同じセットのファイルを多くの方法でデプロイできます。

Web アプリケーションのインストール

Web アーカイブ・ファイル (WAR) を使用して、Web アプリケーションのパッケージ、配布、およびインストールを行うことができます。一般には、Web アプリケーションを構成するファイルを、デプロイメント用に 1 つの WAR ファイルにパッケージします。 WAR ファイルには、*web.xml* サーバー構成ファイル、group.properties 構成ファイル、および DAD ファイルと DADX ファイルを含めることができます。 『WebSphere Application Server Version 5.1 以降 (Windows および UNIX 用) での WORF の例のデプロイ』でデプロイした WORF のサンプルには、 *apache-services.war* と *axis-services.war* という 2 つの WAR ファイルのサンプルが含まれます。 WebSphere® Studio のような一部の開発環境では、自動デプロイメント機能が提供されます。しかし、Web サービスに独自のデプロイメント記述子ファイルを提供したいのであれば、この自動デプロイメントを使用不可にすることができます。

関連タスク:

- 155 ページの『Web アーカイブ・ファイルの準備と作成』
- 50 ページの『WebSphere Application Server Version 5.1 以降 (Windows および UNIX 用) での WORF の例のデプロイ』

Java 2 Enterprise Edition アプリケーション

Java™ 2 Enterprise Edition (J2EE) アプリケーションをデプロイする際には、e-business アプリケーション用に以下のコンポーネントを構築しなければなりません。

Web アーカイブ (WAR)
Web 関連のコンポーネント (HTML、JavaScript™、JavaServer Page)

Java アーカイブ (JAR)
ビジネス・ロジック・コンポーネントを作成する Java クラス

エンタープライズ・アーカイブ (EAR)
エンタープライズ・ソリューションを作成する Java アーカイブ・ファイルと Web アーカイブ・ファイル

WebSphere® Application Server 5.0 でデプロイ可能な最小単位は Web アーカイブ・ファイルです。このアプリケーションが Enterprise JavaBeans™ を開発する場合は、Java アーカイブ・ファイルとエンタープライズ・アーカイブ・ファイルが必要です。

関連概念:

- 184 ページの『IBM DB2 Information Integrator で照会を設計する利点』
- 69 ページの『Web サービスのグループの定義』

DB2 Information Integrator に DB2 用のアプリケーション・サーバーをインストールする

アプリケーション・サーバーは、企業が次世代の e-business アプリケーションを開発、デプロイ、および統合できるようにします。アプリケーション・サーバーを、Web アプリケーションを管理するツールとして使用できます。

DB2® Universal Database のバージョン 8.1.2 以降には、組み込みアプリケーション・サーバーが備えられており、これは DB2 用のアプリケーション・サーバーとして示されています。この DB2 UDB 用アプリケーション・サーバーを使用する場合には、ご使用の DB2 UDB Web アプリケーションを Windows®、 Linux、AIX、および Solaris 上で実行する際にその他のアプリケーション・サーバーをインストールする必要はありません。

前提条件:

- DB2 Universal Database ESE バージョン 8.1.2 以上。
- 少なくとも 1 つの DB2 UDB インスタンスが存在すること。
- ご使用の環境に応じて以下のコマンドを発行します。

```
<db2instance path>/sqllib/db2profile (for Windows)
. <db2instance path>/sqllib/db2profile (for UNIX systems)
```

制約事項:

1 つのシステムには、 1 つまたは複数の DB2 UDB インスタンスのある 1 つの DB2 アプリケーション・サーバーのみインストールできます。

手順:

Java application development and Web administration tools supplement for DB2 CD から DB2 用のアプリケーション・サーバーをインストールします。 DB2 Universal Database™ には、 DB2 Universal Database™ インストール・パッケージとこの CD が共に備えられています。 DB2 アプリケーション・サーバーをインストールするには、以下のようにします。

```
db2appserverinstall
     -asroot path
     -hostname name
```

-asroot
アプリケーション・サーバー・インストールのための絶対パス。

-hostname
ホスト・システムの名前。

関連タスク:

- 「インストールおよび構成 補足」の『DB2 用のアプリケーション・サーバーをインストールする』
- 「インストールおよび構成 補足」の『DB2 用のアプリケーション・サーバーをアンインストールする』

Information Integrator 中で DB2 用のアプリケーション・サーバーの開始と停止を実行する

DB2 UDB 用のアプリケーション・サーバーの開始および停止は、 DB2 用アプリケーション・サーバー・ディレクトリーの *bin¥* サブディレクトリーから行えます。さらに、DB2EAS.SERVER という名前のストアード・プロシージャーを使用しても、アプリケーション・サーバーを開始および停止できます。

手順:

DB2 UDB アプリケーション・サーバーの開始と停止を行うには、以下のコマンドを使用します。

```
startServer <serverName>
stopServer <serverName>
```

開始および停止コマンドには、以下のパラメーターが必要です。

serverName
開始したいアプリケーション・サーバーの名前。

DB2 用のアプリケーション・サーバーで WORF サンプルをデプロイできるようにするためのアプリケーションのデプロイおよび管理に関する情報は、「*WebSphere Application Server System Administration*」を参照してください。 DB2 用のアプリケーション・サーバーで WORF サンプルをデプロイした後、ブラウザーから WORF テスト・ページにアクセスすることができます。

関連タスク:

- 「インストールおよび構成 補足」の『DB2 用のアプリケーション・サーバーをローカルに開始する』
- 「インストールおよび構成 補足」の『DB2 用のアプリケーション・サーバーをローカルに停止する』
- 「インストールおよび構成 補足」の『DB2 用のアプリケーション・サーバーをリモート側で開始する』
- 「インストールおよび構成 補足」の『DB2 用のアプリケーション・サーバーをリモート側で停止する』
- 50 ページの『WebSphere Application Server Version 5.1 以降 (Windows および UNIX 用) での WORF の例のデプロイ』
- 150 ページの『DB2 Information Integrator に DB2 用のアプリケーション・サーバーをインストールする』

デプロイメント記述子の生成

WebSphere Application Server 5.1 は、Apache SOAP を使用する場合、ランタイムにデプロイメントとアンデプロイメントを使用不可にするカスタム ConfigManager (com.ibm.soap.server.XMLDrivenConfigManager) を使用します。その代わりに、WebSphere は、アプリケーションが開始したときに、デプロイされたサービスをリストするファイルを読み取ります。 OS/390 および z/OS プラットフォームの場合、自動デプロイメントは常に使用可能であるため、独自にデプロイメント記述子を生成する必要はありません。

手順:

このファイルを作成するには、 Web サービスごとにデプロイメント記述子ファイル、すなわち構成およびデプロイメント情報を識別する DADX ファイルを作成しなければなりません。このファイルを dds.xml という名前の Extensible Markup Language (XML) ファイルに挿入しておくと、 WebSphere ConfigManager が Web アプリケーション起動時にそれを読み取ります。 Apache SOAP 構成マネージャーはランタイムにオンデマンドで Web サービスをデプロイするので、デプロイメント記述子ファイルにそれらを追加する必要はありません。

1. *worf.jar* と *soap.jar* のファイルが、クラス・パスにあることを確認します。
2. DADX2DD コマンドを使用して、DADX ファイルからデプロイメント記述子を生成します。デプロイメント記述子ファイルは、Apache Axis では必要ありません。以下のパラメーターを指定して、クラス `com.ibm.etools.webservice.rt.dadx.Dadx2Dd` を使用します。

   -r グループに関連する Web サービスのリソース名。デプロイメント記述子にはこのパラメーターが必要です。

   -p グループ・パス。デプロイメント記述子にはこのパラメーターが必要です。

   -n グループ名。デプロイメント記述子にはこのパラメーターが必要です。

   -i DADX ファイルまたは 1 つ以上の DADX ファイルを含むディレクトリーの名前。このディレクトリーには、1 つ以上の DADX ファイルを含む、 1 つ以上のサブディレクトリーを含めることができます。ディレクトリー名を使用する場合、ファイル dds.xml および WEB-INF を含むディレクトリーのような、 Web アプリケーションのルート・ディレクトリーを使用します。 *-i* パラメーターはオプションです。このパラメーターを指定する場合は、DADX ファイルが存在しており、かつ読み取り可能でなければなりません。指定しない場合は、DADX ファイルの名前を標準入力から受け取ります。標準入力を示すには、ダッシュ (-) を使用します。

   -o デプロイメント記述子ファイル名。この引き数はオプションです。このパラメーターを指定する場合、ファイルが書き込み可能でなければなりません。すでにそのファイルが存在している場合は、上書きされます。指定しない場合は、デプロイメント記述子は標準出力に書き込みます。標準出力を示すには、ダッシュ (-) を使用します。

   -s ターゲット SOAP エンジン。有効値には、*apache-soap* と *apache-axis* が含まれます。

Web サービスが呼び出されると、WORF は web.xml ファイルを読み取り、ロードする SOAP エンジン・クラスを決定します。 SOAP エンジンをロードできない場合、デフォルトの SOAP エンジンが使用されます。 web.xml ファイルで soap-engine パラメーターが指定されない場合、デフォルトの SOAP エンジンは Apache Axis です。 Apache SOAP では、Dadx2Dd コマンドの出力を、 ddx.xml ファイルに挿入することができます。 Apache Axis では、デプロイメント記述子は必ず実行時に動的に生成されます。出力は通知用だけです。

Web サービスの実行時に、SOAP エンジンを切り替えることができます。 SOAP エンジンを切り替える前に、デプロイメント記述子を dds.xml ファイルに含める必要があります。 dds.xml ファイルは、Apache SOAP 専用です。 Apache Axis では、デプロイメント記述子ファイルは必要ありません。 Apache Axis に切り替える前に、アプリケーションで deploy.wsdd ファイルを使用できるようにする必要があります。

SOAP エンジンへ接続しなくても、デプロイメント記述子生成プログラム (Dadx2Dd) は実行されます。

たとえば、現行ディレクトリーが *WEB-INF* の場合、次のコマンドは dxx_travel グループから *ZipCity.dadx* ファイルを読み取ります。次いで、デプロイメント記述子を *dds* サブディレクトリーに書き込みます。

```
java com.ibm.etools.webservice.rt.dadx.Dadx2Dd -r ZipCity.dadx -p ¥travel
   -n ¥dxx_travel -i classes¥groups¥dxx_travel¥ZipCity.dadx
    -o classes¥dds¥dxx_travel¥ZipCity.isd
```

3. 新たに作成された Apache SOAP ISD ファイルの内容をコピーして、それを Web アプリケーション・ディレクトリーの *dds.xml* ファイルに追加します。

```
<?xml version='1.0'?>

<dds>

<isd:service xmlns:isd='http://xml.apache.org/xml-soap/deployment'
  id='http://tempuri.org/travel/ZipCity.dadx'>
   <isd:provider
    type='com.ibm.etools.webservice.rt.framework.ServiceProvider'
    scope='Request'
    methods='findCityByZipCode insertZipCodeAndCity
       updateCityForZipCode deleteZipCode'>
    <isd:java class='com.ibm.etools.webservice.rt.dxx.DxxService'/>
    <isd:option key='group.name' value='/dxx_travel'/>
    <isd:option key='group.path' value='/travel'/>
    <isd:option key='group.class.name'
           value='com.ibm.etools.webservice.rt.dxx.DxxGroup'/>
   </isd:provider>
 <isd:faultListener>org.apache.soap.server.DOMFaultListener
 </id:faultListener>
 <isd:mappings
    defaultRegistryClass=
    'com.ibm.etools.webservice.rt.dxx.DxxMappingRegistry'/>
</isd:service>
...
...
</dds>
```

図 46. isd ファイルを含む dds.xml の一部の例

4. Web アプリケーションを再始動します。

関連概念:

- 154 ページの『Apache SOAP の構成』
- 149 ページの『Web アプリケーションのインストール』
- 69 ページの『Web サービスのグループの定義』

関連タスク:

- 127 ページの『Web サービスのテスト』

Apache SOAP の構成

Apache SOAP エンジンを使用する場合、Apache 構成マネージャー (デフォルト) か、 IBM® 構成マネージャー (XMLDrivenConfigManager) を使用するよう、 Web アプリケーションを構成することができます。 Apache 構成マネージャーは、デプロイ後のサービスを、 DeployedServices.ds というシリアライズ化された Java™ ファイルに保管します。 XMLDrivenConfigManager は、新しい DADX ファイルの追加の際に、Web サービスの自動デプロイメントを使用不可にし、代わりに XML フォーマットを使用するので、手動でデプロイしなければなりません (『デプロイメント記述子の生成』を参照)。

サンプル・アプリケーションをデプロイできるように、WORF では、 WORF エンジンに付属する *services.war* ファイルの中に、構成ファイル (*dds-example.xml*) が用意されています。 *dds-example.xml* は、例中のすべての DADX ファイルで使用できる、デプロイメント記述子ファイルです。サンプル *dds.xml* ファイルは、いくつか

のサービスをデプロイします。デプロイされていないサービスを呼び出すと、不明のサービスであることがサーバーから報告されます。

IBM 構成マネージャーを使用したい場合、*soap-ibm.xml* を *soap.xml* に名前変更し、*dds-example.xml* を *dds.xml* に改名します。アプリケーション・サーバーを再始動して、installedApps¥servicesApp.ear¥services.war サブディレクトリー内で XMLDrivenConfigManager を使用します。

関連概念:

- 40 ページの『Web サービス・プロセスの概要』

関連タスク:

- 152 ページの『デプロイメント記述子の生成』

Web アーカイブ・ファイルの準備と作成

Web アーカイブ (WAR) ファイルを作成するには、次のようにします。

手順:

1. 以下の例にあるように、 WAR ファイルの基本ディレクトリー構造を作成します。

   ```
   WEB-INF¥lib¥worf-servlets.jar
   WEB-INF¥web.xml
   ```

   これらのファイルは、ダウンロードした *worf¥* のファイルです。

   この WORF ディレクトリー階層は apache-services.war ファイルまたは、axis-services.war ファイルにあります。これらのファイルは TEST ページを実行するときに使用します。 WORF の組み込みテスト機能を使用するつもりがない場合、 *worf¥* サブディレクトリーのファイルは必要ありません。worf-servlets.jar ファイルは、WORF がインストールされた *lib¥* サブディレクトリーにあります。 web.xml は標準 J2EE web.xml です。空の web.xml の例を 図 47 に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
</web-app>
```

図 47. 空の web.xml ファイル

2. グループごとに、
   a. グループ・サブディレクトリー (たとえば WEB-INF¥classes¥groups¥myGroup) を作成し、そこに group.properties および自分の DADX ファイルを組み込みます。
   b. WEB-INF¥web.xml ファイルを編集して、サーブレットおよびサーブレット・マッピングを追加します (たとえば、サーブレット名 *myGroup* と URL マッピング *myURLPath*を追加します)。
   c. オプション: デプロイメント記述子を生成します (OS/390 および z/OS プラットフォームの場合は、このステップを飛ばします)。このステップは IBM

構成マネージャーの場合にだけ必要です。次の例では、最初の指示で現行ディレクトリーを Windows プラットフォーム上の WEB-INF サブディレクトリーに変更します。その後、アプリケーションを実行します。アプリケーションが完了した後、ルート・ディレクトリーに戻ることができます。

```
cd WEB-INF
java com.ibm.etools.webservice.rt.dadx.Dadx2Dd
  -r ivt.dadx
  -p ¥myURLPath
  -n ¥myGroup
 -i classes¥groups¥myGroup¥ivt.dadx
 -o classes¥dds¥myGroup¥ivt.dadx
cd ..
```

注: ivt.dadx は製品添付の特定サンプルであるため、新規に作成した WAR ファイルにはこのファイルが含まれていない場合があります。

d. オプション: *dds.xml* を作成または変更して、生成された記述子の内容を追加します (OS/390 および z/OS プラットフォームの場合は、このステップを飛ばします)。このステップは IBM 構成マネージャーの場合にだけ必要です。空の dds.xml ファイルの例を以下に示します。

```
<?xml version='1.0'?>
<dds>
</dds>
```

3. 次の方式のどれかを使って WAR ファイルを作成します。
   - コマンド行から次のコマンドを発行します。

     ```
     jar -cvf minWORFwar.war WEB-INF worf
     ```

   - WebSphere Studio で、メニューから**「ファイル (File)」**を選択し、次いで**「エクスポート (Export)」**、**「WAR ファイル (WAR file)」**の順に選択します。Web アプリケーションが含まれるプロジェクト名を選択して、ファイル名を指定します。
4. サンプル・インストールに記述されているように、 WAR ファイルを WebSphere Application Server または Apache Jakarta Tomcat に対してデプロイします (たとえば、Web アプリケーション・コンテキストとして myContext を指定する)。
5. TEST ページを実行して、WAR ファイルが正しく作成されていることを確認します。たとえば、TEST の URL は次のようであるかもしれません。

   ```
   http://<your WebAppServer>/myContext/myURLPath/ivt.dadx/TEST
   ```

   注: ivt.dadx は製品添付の特定サンプルであるため、新規に作成した WAR ファイルにはこのファイルが含まれていない場合があります。

関連概念:

- 69 ページの『Web サービスのグループの定義』

関連タスク:

- 76 ページの『group.properties ファイルのカスタマイズ』
- 73 ページの『iSeries プラットフォームでの web.xml ファイルおよび group.properties ファイルの定義』

Web サービス・プロバイダーのトレース

Web サービスをデプロイした後、ランタイム・イベントに関する情報および Web サービス・プロバイダーからの診断情報を取得する必要があるかもしれません。 Web サービス・アプリケーションのデバッグおよびトラブルシューティングをするために、 DB2® Web サービスは、アプリケーションが実行される Web アプリケーション・サーバーのトレース機能を使用します。 Web アプリケーション・サーバーから受け取るトレース情報には、メッセージおよびイベント・アクティビティーが含まれます。

DB2 Web サービス・プロバイダーは、以下の 2 つのトレース・システムをサポートします。

log4j Jakarta-log4j-1.2.8 および commons-logging-1.0.3 (Apache Jakarta Tomcat 4.0.6 以降)

JRas WebSphere® Application Server バージョン 5 以降により使用されるトレースおよびロギング・システム

これらのトレース・システムを使用すると、メッセージ・ロギングとトレース機能を Java™ アプリケーションに組み込むことができます。

アプリケーション内で使用可能にしたトレースから生成される出力は、使用する Web アプリケーション・サーバーのルート・ディレクトリーに出力されます。 表 17 には、出力ログ・ファイルのロケーションが示されています。

表 17. トレース出力のロケーション

サーバー	出力ログ・ファイルのロケーション
WebSphere Application Server	${SERVER_LOG_ROOT}/trace.log
Apache Jakarta Tomcat	<tomcat>/logs/worf_log4j.log

すべてのトレース・メッセージとイベントは、Web サービスの操作名、サーブレット名、または DADX ファイル名で識別されます。

DB2 Web サービス・プロバイダーは、以下のタイプのイベントをトレースできます。

通知メッセージ
Web サービス要求イベントまたは Web サービス応答イベントが正常に完了したこと (DADX ファイルが正常に構文解析された場合など) を示すメッセージ。

警告メッセージ
Web サービス要求または Web サービス応答の処理中に警告状態が検出されたことを示すメッセージ (DADX ファイルに対する XML パーサーからの警告メッセージなど)。

エラー・メッセージ
Web サービス要求または Web サービス応答の処理中にエラーが検出されたこと (アプリケーションが例外を出した場合など) を示すメッセージ。

トレース・イベント
アプリケーションがメソッド、例外、呼び出しスタック、変数の値に入ったり、これを終了したりしたことを示すイベント。

関連概念:

- 37 ページの『Web サービス・プロバイダーのフィーチャー』

関連タスク:

- 158 ページの『DB2 Web サービス・プロバイダー Apache Tomcat バージョン 4.0 以降の Web アプリケーション・サーバーでのトレースの使用可能化』
- 159 ページの『DB2 Web サービス・プロバイダー WebSphere Application Server でのトレースの使用可能化』
- 161 ページの『DB2 Web サービス・プロバイダー WebSphere Studio Application Developer でのトレースの使用可能化』

関連資料:

- 68 ページの『Web サービスのトラブルシューティング』

DB2 Web サービス・プロバイダー Apache Tomcat バージョン 4.0 以降の Web アプリケーション・サーバーでのトレースの使用可能化

Apache Tomcat サーバーを構成して DB2 Web サービスをトレースできます。

前提条件:

使用するサーバーの構成を変更するための許可が必要です。

手順:

DB2 Web サービス・プロバイダーにおけるデフォルトの `log4j` トレースを変更するには、以下のようにします。

1. 以下の例のような項目を持つ構成ファイルを `log4j.configuration` という名前で作成します。

```
log4j.rootCategory=DEBUG, console, rollingFile
log4j.logger.com.ibm.etools.rt.webservice.*=INFO
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%5p [%t] (%F:%L) - %m%n
log4j.appender.rollingFile=org.apache.log4j.RollingFileAppender");
log4j.appender.rollingFile.File=<servletContext>¥..¥..¥logs¥worf_log4j.log
log4j.appender.rollingFile.MaxFileSize=100KB
log4j.appender.rollingFile.layout=org.apache.log4j.TTCCLayout
log4j.appender.rollingFile.layout.layout.ConversionPattern=%p %t %c - %m%n
```

2. 構成ファイルの設定を以下のように変更して、特定のタイプのメッセージのみを表示するようにします。

表 18. log4j 構成ファイルのメッセージ設定

メッセージ・タイプ	構成設定
log4j 警告メッセージ以上	log4j.logger.com.ibm.etools.webservice.*

表 18. log4j 構成ファイルのメッセージ設定 (続き)

メッセージ・タイプ	構成設定
log4j 通知メッセージ	log4j.logger.com.ibm.etools.webservice.*=INFO
log4j エラー・メッセージ	log4j.logger.com.ibm.etools.webservice.*=ERROR

3. 構成ファイル `log4j.configuration` を、Web アプリケーションの WEB-INF/classes ディレクトリーに置きます。

トレース・イベントのログを、 <インストールされた Web サーバーのロケーション >¥AppServer¥logs¥<ローカル・サーバー名> で参照することができます。

関連概念:

- 37 ページの『Web サービス・プロバイダーのフィーチャー』
- 157 ページの『Web サービス・プロバイダーのトレース』

関連タスク:

- 63 ページの『Apache Jakarta Tomcat での WORF の例のインストールおよびデプロイ』

DB2 Web サービス・プロバイダー WebSphere Application Server でのトレースの使用可能化

WebSphere Application Server を構成して、管理コンソールから DB2 Web サービスをトレースできます。

前提条件:

使用するサーバーの構成を変更するための許可が必要です。

手順:

DB2 Web サービス・プロバイダーにおけるデフォルトの jRAS トレースを変更するには、以下のようにします。

1. WebSphere Application Server 管理コンソールを始動します。
2. ナビゲーション・ツリーで、**「トラブルシューティング (Troubleshooting)」** —> **「ログとトレース (Log and Trace)」**とクリックします。「ロギングとトレース (Logging and Tracing)」ウィンドウが開きます。
3. 「ロギングとトレース (Logging and Tracing)」ウィンドウで、サーバー名をクリックした後、**「診断トレース (Diagnostic Trace)」**を選択します。
4. **「Trace Specification」**フィールドで、以下のようにトレース・ストリングを入力します。

   ```
   com.ibm.etools.webservice.*=all=enabled
   ```

5. サーバーが停止している場合には、「構成 (Configuration)」ページに進みます。サーバーが実行されている場合には、「ランタイム (Runtime)」ページに進みます。
   - オプション: 「ランタイム (Runtime)」ページから、**「トレースの保管 (Save trace)」**チェック・ボックスを選択し、変更をサーバー構成に書き込みます。

「**トレースの保管 (Save trace)**」チェック・ボックスがチェックされていない場合には、行った変更は、現在実行中のサーバー・プロセスの存続期間中のみ適用されます。

- オプション: 「構成 (Configuration)」ページから、「**トレースを使用可能にする (Enable Trace)**」チェック・ボックスを選択します。

サーバーが実行されていない場合のトレースの使用可能化の例は、図 48 にあります。

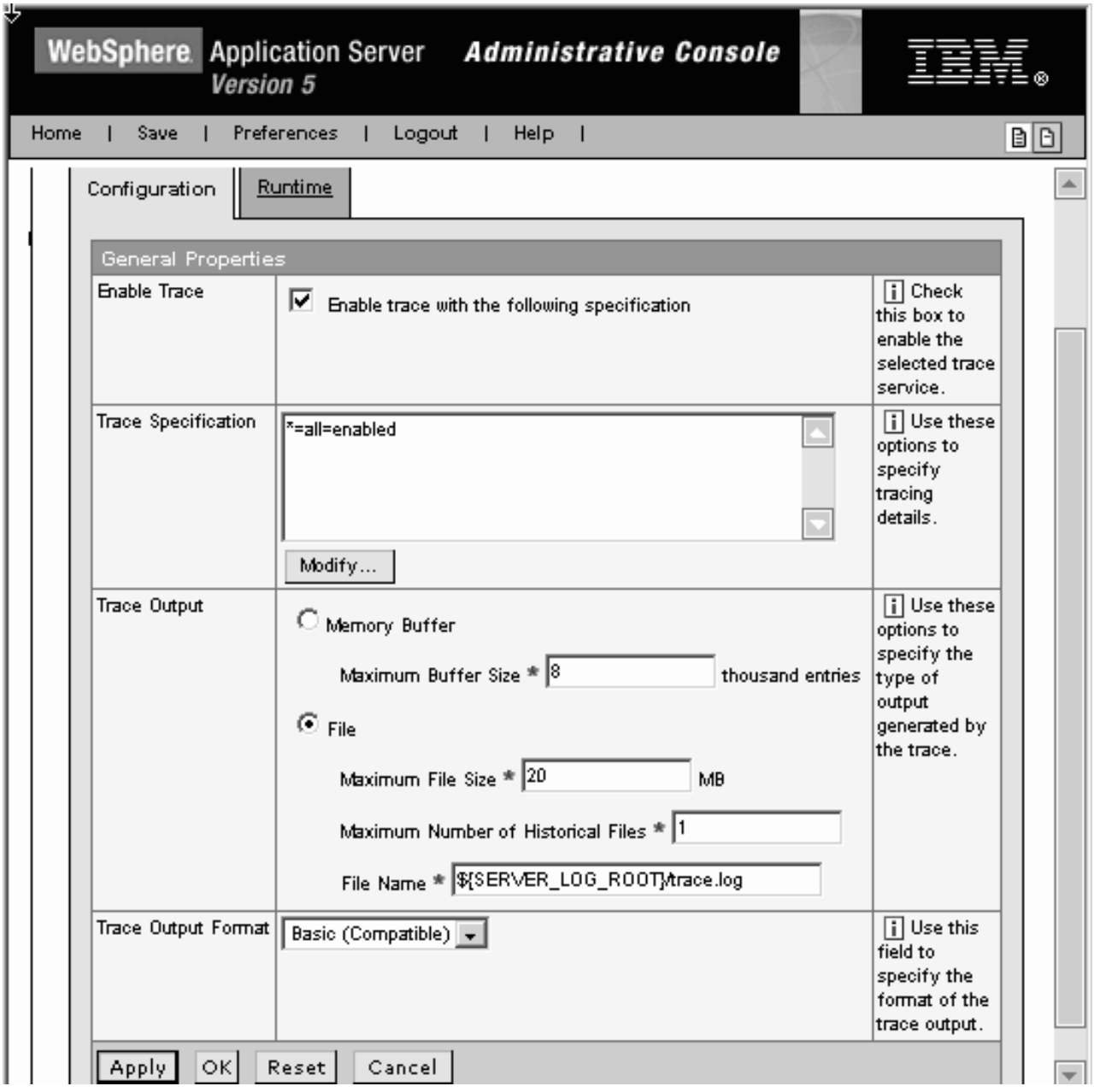


図 48. Web サービス・プロバイダーのトレースの使用可能化

6. 変更を保管し、サーバーを再始動します。

トレース・イベントのログを、 <インストールされた Web サーバーのロケーション >¥AppServer¥logs¥<ローカル・サーバー名> で参照することができます。

関連概念:

- 37 ページの『Web サービス・プロバイダーのフィーチャー』
- 157 ページの『Web サービス・プロバイダーのトレース』

関連タスク:

- 161 ページの『DB2 Web サービス・プロバイダー WebSphere Studio Application Developer でのトレースの使用可能化』

DB2 Web サービス・プロバイダー WebSphere Studio Application Developer でのトレースの使用可能化

WebSphere Studio を構成して、管理コンソールから DB2 Web サービスをトレースできます。

前提条件:

使用するサーバーの構成を変更するための許可が必要です。

手順:

DB2 Web サービス・プロバイダーにおけるデフォルトの jRAS トレースを変更するには、以下のようにします。

1. WebSphere Studio 管理コンソールを始動します。
2. メインメニューから、**「ウィンドウ (Window)」** —> **「ビューの表示 (Show View)」** —> **「サーバー構成 (Server Configuration)」**とクリックし、「サーバー構成 (Server Configuration)」ビューを開きます。
3. **「サーバー (Servers)」**メニューで、**「WebSphere v5.0 Test Environment」**をダブルクリックしてサーバー・エディターを開きます。
4. 「トレース (Trace)」ページに進みます。
5. **「Trace Specification」**フィールドで、以下のトレース・ストリングを入力します。

   ```
   com.ibm.etools.webservice.*=all=enabled
   ```

6. **「トレースを使用可能にする (Enable Trace)」**チェック・ボックスを選択します。
7. 変更を保管し、サーバーを再始動します。

トレース・イベントのログを、 <インストールされた Web サーバーのロケーション >¥AppServer¥logs¥<ローカル・サーバー名> で参照することができます。

関連概念:

- 37 ページの『Web サービス・プロバイダーのフィーチャー』
- 157 ページの『Web サービス・プロバイダーのトレース』

関連タスク:

- 159 ページの『DB2 Web サービス・プロバイダー WebSphere Application Server でのトレースの使用可能化』

Web サービスの発行

Web サービス・プロバイダーは Web サービスを公開し、クライアントは Hypertext Transfer Protocol (HTTP) 上で Simple Object Access Protocol (SOAP) を使用してサービスにアクセスできます。これは、Enterprise Java™ Bean (EJB) クライアントが Internet Inter-Orb Protocol (IIOP) 上でリモート・メソッド呼び出し (RMI) を使用して Bean にアクセスするのと対照的です。 Web サービスは、適切なビジネス関数を呼び出し、通常どおりに応答を戻すことによって、 Web クライアントからの要求を処理します。 Web サービス記述言語 (WSDL) 文書は Web サービスを記述します。 WSDL はリポジトリー (UDDI レジストリーなど) または Web サービス・プロバイダーのサーバー上に保管します。適切なリポジトリーに Web サービス記述を保管することによって、関心を持つ顧客が Web サービスの存在を発見できる可能性を提供し、 Web サービス・プロバイダーが新規ビジネスを創出できるようにしています。

関連概念:

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』

関連タスク:

- 127 ページの『Web サービスのテスト』

Web サービス・コンシューマーのインストールと使用

DB2 Universal Database は Web サービス・プロバイダーへのアクセスをコンシューマーとして最適化できます。 SQL ステートメントを使用することにより、Web サービス・データを利用および統合できます。 SQL を使用して Web サービス・データにアクセスすることによって、 SQL ステートメントのコンテキスト内でデータを操作できるので、労力を軽減できます。また、ステートメントをクライアント・アプリケーションに戻すこともできます。 Web サービス・コンシューマー・ツール・セットは、 SQL から Web サービス・データにアクセスするのに役立ちます。 Web サービス・コンシューマーは既存の WSDL インターフェースを DB2 表またはスカラー関数に変換します。このセクションでは、IBM が WSDL から DB2 SQL 関数に変換するために提供している Web サービス・コンシューマー・スタンドアロン・ツールおよび WebSphere Studio プラグインについて説明します。

Web サービスのユーザー定義のコンシューマー機能のインストール

前提条件:

Web サービス・コンシューマーのユーザー定義関数 (UDF) は次のプラットフォーム (すべて 32 ビット) で使用可能です。

- Windows 2000
- Linux
- AIX
- Solaris オペレーティング環境 (DB2 for Universal Database バージョン 8、Fix Pack 2 適用)

SOAP UDF を実行する前に、次のソフトウェアをインストールする必要があります。

- DB2 Universal Database
  - バージョン 8 (Xerces パーサーおよび XML エクステンダーが含まれている)
- オプション: WebSphere Studio Application Developer (WSAD) バージョン 5.1.1

  プラグインは、WebSphere Studio が WSDL から Web サービス UDF を生成することを要求します。 Web サービス・コンシューマーを直接呼び出して、独自の SQL 関数を開発することもできます。

dxxadm enable_db sample コマンドを使用して、 DB2 XML エクステンダーを使用可能にすることも必要です。 DB2 XML エクステンダー・コマンドのそれ以外のオプションについては、「*DB2 XML Extender 管理およびプログラミングのガイド*」を参照してください。

手順:

Web サービス・コンシューマーを使用可能、またはインストール、および使用不可にするには:

1. 次のユーティリティーを実行して、5 つのユーザー定義関数を登録します。

   ```
   db2enable_soap_udf  -n dbName [-u uID] [-p password] [-force]
   ```

   パラメーターは次のように定義されています。

   dbName
   データベース名

   uID オプション: ユーザー ID

   password
   オプション: ユーザー ID と関連したパスワード

   -force 既存の関数すべてのドロップを試行します。

   enable コマンドを使用すると、データベースで SOAP リクエスター機能を使用できるようになります。
2. Web サービス・コンシューマーを使用不可にする場合は、関数をドロップします。次のユーティリティーを実行します。

   ```
   db2disable_soap_udf  -n dbName [-u uID] [-p password]
   ```

   このパラメーターの意味は、前述の使用可能化ユーティリティーの場合と同じです。
3. DB2 CLP (コマンド行プロセッサー) を使用することにより、ユーザー定義関数の作成および削除も行えます。

   ```
   CREATE SCHEMA db2xml;

      CREATE FUNCTION db2xml.soaphttpv (
                  endpoint_url VARCHAR(256),
                  soap_action VARCHAR(256),
                  soap_body VARCHAR(3072))
            RETURNS VARCHAR(3072)
         LANGUAGE C PARAMETER STYLE DB2SQL
         SPECIFIC soaphttpvivo EXTERNAL NAME 'db2soapudf!soaphttpvivo'
         SCRATCHPAD FINAL CALL FENCED
         NOT DETERMINISTIC CALLED ON NULL INPUT
   ```

```
   NO SQL EXTERNAL ACTION DBINFO;

CREATE FUNCTION db2xml.soaphttpv (
            endpoint_url VARCHAR(256),
            soapaction VARCHAR(256),
            input_message CLOB(1M))
      RETURNS VARCHAR(3072)
   LANGUAGE C PARAMETER STYLE DB2SQL
   SPECIFIC soaphttpcivo EXTERNAL NAME 'db2soapudf!soaphttpcivo'
   SCRATCHPAD FINAL CALL FENCED
   NOT DETERMINISTIC CALLED ON NULL INPUT
   NO SQL EXTERNAL ACTION DBINFO;

CREATE FUNCTION db2xml.soaphttpc (
            endpoint_url VARCHAR(256),
            soapaction VARCHAR(256),
            input_message CLOB(1M))
      RETURNS clob(1M)
   LANGUAGE C PARAMETER STYLE DB2SQL
   SPECIFIC soaphttpcico EXTERNAL NAME 'db2soapudf!soaphttpcico'
   SCRATCHPAD FINAL CALL FENCED
   NOT DETERMINISTIC CALLED ON NULL INPUT
   NO SQL EXTERNAL ACTION DBINFO;

CREATE FUNCTION db2xml.soaphttpc (
            endpoint_url VARCHAR(256),
            soapaction VARCHAR(256),
            soap_body varchar(3072))
      RETURNS clob(1M)
   LANGUAGE C PARAMETER STYLE DB2SQL
   SPECIFIC soaphttpvico EXTERNAL NAME 'db2soapudf!soaphttpvico'
   SCRATCHPAD FINAL CALL FENCED
   NOT DETERMINISTIC CALLED ON NULL INPUT
   NO SQL EXTERNAL ACTION DBINFO;

CREATE FUNCTION db2xml.soaphttpcl (
            endpoint_url VARCHAR(256),
            soapaction VARCHAR(256),
            soap_body varchar(3072))
      RETURNS CLOB(1M) as locator
   LANGUAGE C PARAMETER STYLE DB2SQL
   SPECIFIC soaphttpviclo EXTERNAL NAME 'db2soapudf!soaphttpviclo'
   SCRATCHPAD FINAL CALL NOT FENCED
   NOT DETERMINISTIC CALLED ON NULL INPUT
   NO SQL EXTERNAL ACTION DBINFO;
```

4. Web サービス・コンシューマー WebSphere Studio プラグインは、 WebSphere Studio Application Developer (WSAD) バージョン 5.1.1 のコンポーネントの 1 つです。

関連概念:

• 166 ページの『Web サービスのユーザー定義のコンシューマー機能』

関連資料:

• 180 ページの『Web サービス・コンシューマー UDF の使用』

Web サービスのコンシューマー機能

IBM® DB2® Information Integrator および DB2 Universal Database™ は、構造化照会言語 (SQL) ステートメント内から Web サービスを呼び出せるようにして、DB2 Universal Database および Web サービスの機能を拡張します。このことは、

アクセス可能な Web サービスへの Hypertext Transfer Protocol (HTTP) インターフェース上に、高速クライアント Simple Object Access Protocol (SOAP) を提供するユーザー定義関数 (UDF) のセットを呼び出すことによって行います。これらの関数を SQL ステートメントから直接呼び出すこともできます。

Web サービスの Web サービス記述言語 (WSDL) に応じて、SOAP 本体を構成することができます。 WebSphere® Studio Application Developer (WSAD) の Web サービス・ユーザー定義関数 (UDF) ツールを使用して、特定の UDF を自動的に生成することもできます。これらの UDF は、ユーザー指定の Web サービス記述言語ファイルによって定義された操作を呼び出すことができます。生成された UDF は DB2 UDB 関数であり、以下の処理を行います。

- Web サービス要求にパラメーターを提供する
- SOAP クライアント関数を呼び出す
- Web サービス呼び出しの結果を、ユーザー指定の戻りタイプにマップする

ネットワークの一部では、インターネットへのアクセスでファイアウォールを通過しなければならないことがあります。トラフィックは、ネットワーク・トラフィックを送信できる特定のマシンと特定のポートに限定されている可能性があります。一部のシステムでは、アプリケーションがファイアウォールをトンネルできる場合があります。 SOAP UDF では、SOCKS クライアントと HTTP プロキシーを使用したトンネリングをサポートしています。ファイアウォールをトンネルする SOCKS サーバーを使用するには、システムに SOCKS クライアント・ソフトウェアをインストールする必要があります。 HTTP プロキシーを使用するには、DB2 Universal Database に対して構成する 2 つの環境変数を設定する必要があります。 `DB2SOAP_PROXY` は、HTTP プロキシーを備えたマシンのホスト名を組み込むよう設定します。 `DB2SOAP_PORT` は、HTTP プロキシーのポートに設定します (たとえば、8080)。どちらのケースでも、SOAP トラフィックは、ファイアウォールをトンネルするシステムを通過します。

以下のステップを行って、DB2 Information Integrator に添付されているサンプル・アプリケーションをテストすることができます。

1. データベース・マネージャーを開始する (db2start コマンドを使用)。
2. 「sample」データベースを作成する (db2sampl コマンドを使用)。
3. 「sample」データベースとの接続を確立する。
4. 次のコマンドでサンプル・ファイル (Windows® 環境では、これらのサンプル・ファイルは、 <DB2 UDB インストール・パス>¥samples¥soap にある) を呼び出す。 *db2 -vf filename -t*。

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『レジストリーおよび環境変数の宣言』
- 162 ページの『Web サービスのユーザー定義のコンシューマー機能のインストール』

関連資料:

- 180 ページの『Web サービス・コンシューマー UDF の使用』

Web サービスのユーザー定義のコンシューマー機能

Simple Object Access Protocol (SOAP) は、以下の特性を持つ Extensible Markup Language (XML) プロトコルです。

- メッセージの内容とメッセージの処理方法を記述する枠組みを定義するエンベロープ
- アプリケーション定義のデータ・タイプのインスタンスを表現するエンコード規則のセット
- SOAP 要求および応答を表現する規則

DB2® Universal Database は、SOAP 要求を作成し、SOAP 応答を受け取るために、以下の情報を必要とします。

- サービス・エンドポイント。たとえば、*http://services.xmethods.net/soap/servlet/rpcrouter*
- SOAP 本体の Extensible Markup Language (XML) 内容。それには、要求先のネーム・スペース URI を含む操作の名前、エンコード・スタイル、および入力引き数が含まれます。
- オプション: SOAP アクション URI 参照。次の例に示されているように、*http://tempuri.org/* または " だけで、参照を空にすることもできます。

DB2 UDB 関数 db2xml.soaphttp() は次のアクションを実行します。

1. SOAP 要求を構成する
2. サービス・エンドポイントに要求を通知する
3. SOAP 応答を受け取る
4. SOAP 本体の内容を戻す

これは、SOAP 本体に応じて、VARCHAR() または CLOB() に使用される多重定義関数です。

```
db2xml.soaphttpv returns VARCHAR():
    db2xml.soaphttpv (endpoint_url VARCHAR(256),
                      soap_action VARCHAR(256),
                      soap_body VARCHAR(3072))
                  RETURNS VARCHAR(3072)
db2xml.soaphttpv returns VARCHAR():
    db2xml.soaphttpv (endpoint_url VARCHAR(256),
                      soap_action VARCHAR(256),
                      soap_body CLOB(1M))
                  RETURNS VARCHAR(3072)


db2xml.soaphttpc returns CLOB():
       db2xml.soaphttpc (endpoint_url VARCHAR(256),
                         soapaction VARCHAR(256),
                         soap_body VARCHAR(3072))
                        RETURNS CLOB(1M)
db2xml.soaphttpc returns CLOB():
       db2xml.soaphttpc (endpoint_url VARCHAR(256),
                         soapaction VARCHAR(256),
                         soap_body CLOB(1M))
                        RETURNS CLOB(1M)

db2xml.soaphttpcl returns CLOB() as locator:
       db2xml.soaphttpcl(endpoint_url VARCHAR(256),
```

```
                    soapaction VARCHAR(256),
                    soap_body varchar(3072))
                RETURNS CLOB(1M) as locator
```

DB2 UDB 構成済み SOAP 要求エンベロープ

167 ページの図 49 の例は、 SOAP 要求エンベロープをホストに通知する Hypertext Transfer Protocol (HTTP) 通知ヘッダーを示しています。太字の部分は Web サービス・エンドポイント (通知パスおよびホスト) および SOAP 本体の内容を示しています。 SOAP 本体は、郵便番号 95120 の温度要求を示しています。

```
POST /soap/servlet/rpcrouter HTTP/1.0
Host: services.xmethods.net
Connection: Keep-Alive User-Agent: DB2SOAP/1.0
Content-Type: text/xml; charset="UTF-8"
SOAPAction: ""
Content-Length: 410

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
                   xmlns:SOAP-ENC=http://schemas.xmlsoap.org/soap/encoding/
                   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                   xmlns:xsd=http://www.w3.org/2001/XMLSchema >
   <SOAP-ENV:Body>
       <ns:getTemp xmlns:ns="urn:xmethods-Temperature">
         <zipcode>95120</zipcode>

       </ns:getTemp>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図 49. DB2 UDB 構成済み SOAP 要求エンベロープ

DB2 UDB を使用して SOAP 応答エンベロープの内容を取り出す例

168 ページの図 50 の例は、 SOAP 応答エンベロープを含む HTTP 応答ヘッダーを示しています。 SOAP 本体の **太字**の内容は、温度要求の結果を示しています。 SOAP エンベロープからのネーム・スペース定義はここには示されていませんが、それらも含まれています。

```
HTTP/1.1 200 OK
Date: Wed, 31 Jul 2002 22:06:41 GMT
Server: Enhydra-MultiServer/3.5.2
Status: 200
Content-Type: text/xml; charset=utf-8
Servlet-Engine: Lutris Enhydra Application Server/3.5.2
  (JSP 1.1; Servlet 2.2; Java™ 1.3.1_04;
   Linux 2.4.7-10smp i386; java.vendor=Sun Microsystems Inc.)
Content-Length: 467
Set-Cookie:JSESSIONID=JLEcR34rBc2GTIkn-0F51ZDk;Path=/soap
X-Cache: MISS from www.xmethods.net
Keep-Alive: timeout=15, max=10
Connection: Keep-Alive
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
                   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                   xmlns:xsd=http://www.w3.org/2001/XMLSchema >
     <SOAP-ENV:Body>
           <ns1:getTempResponse xmlns:ns1="urn:xmethods-Temperature"
           SOAP-ENV:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/ >
              <return xsi:type="xsd:float">85<return>
           </ns1:getTempResponse>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図 50. DB2 UDB を使用して SOAP 応答エンベロープの内容を取り出す

関連タスク:

- 162 ページの『Web サービスのユーザー定義のコンシューマー機能のインストール』

関連資料:

- 180 ページの『Web サービス・コンシューマー UDF の使用』

Web サービス・コンシューマー・イベントのトレース

Web サービス・コンシューマー・ユーザー定義関数は、 DB2 Universal Database トレース・ユーティリティーを使用してトレースできます。さらに、Windows プラットフォームを使用している場合は、 Hypertext Transfer Protocol (HTTP) SOAP 要求および応答をファイルにトレースすることができます。

手順:

DB2 Universal Database で SOAP コンポーネントをトレースするには、次のトレース・マスクを使用します。

```
db2trc on -m *.*.147.*.*
```

関連概念:

- 164 ページの『Web サービスのコンシューマー機能』

関連資料:

- 180 ページの『Web サービス・コンシューマー UDF の使用』

Web サービス・コンシューマー — WebSphere Studio ユーザー定義関数ツールの使用

Web サービス・コンシューマー・ユーザー定義関数ウィザードは、 WebSphere® Studio バージョン 5 でユーザー定義関数を生成およびテストします。

WebSphere Studio で使用されるウィザードは、 Web Services Definition Language (WSDL) ファイルを読み取ります。次いで、データベース・アプリケーションから Web サービスへの容易なアクセスを提供するユーザー定義関数 (UDF) を生成します。生成された UDF を SQL ステートメントで使用して、リレーショナル・データを Web サービスから検索した動的データに結合することもできます。 SQL 内で Web サービス・コンシューマー関数を直接呼び出すこともできます (『Web サービスのコンシューマー機能』を参照)。ただし、この作業には上級のプログラミング・スキルが必要とされ、また時間がかかる場合もあります。 UDF を生成しデプロイした後で、この関数を SQL 内で使用して、リレーショナル・データを Web サービスから検索した動的データに結合することもできます。生成された UDF は次のような構造になっています。

1. SOAP 本体の構成
2. SOAP コンシューマーの呼び出し (SOAP 要求エンベロープのサブミット)
3. SOAP 応答からの値の抜き出し

関連概念:

- 164 ページの『Web サービスのコンシューマー機能』
- 166 ページの『Web サービスのユーザー定義のコンシューマー機能』

関連タスク:

- 162 ページの『Web サービスのユーザー定義のコンシューマー機能のインストール』
- 168 ページの『Web サービス・コンシューマー・イベントのトレース』

関連資料:

- 180 ページの『Web サービス・コンシューマー UDF の使用』

WebSphere Studio からユーザー定義関数を生成する方法

前提条件:

1. DB2 XML エクステンダー・データベースを使用可能にする
2. このデータベースに Web サービス・コンシューマー UDF を使用可能にする
3. Web サービス UDF で使用したいプロジェクトを作成する
4. 使用可能にしたばかりのデータベースへの接続を作成する
5. WebSphere Studio バージョン 5 プロジェクトにデータベースをインポートする。詳細については、「*WebSphere Studio Application Developer Programming Guide*」をご覧ください。

手順:

WebSphere Studio 内では、ユーザー定義関数 (UDF) を生成するウィザードを 3 通りの異なる方法で起動できます。

- 「ファイル *(File)*」>「新規 *(New)*」>「その他 *(Other)*」>メニューから呼び出すことができます。次いで、**「データ (Data)」**を選択します。フォルダーが展開するので、メニューから**「Web サービス・ユーザー定義関数 (Web Service User-Defined Function)」**を選択します。**「次へ」**プッシュボタンをクリックして、次に進みます。
- Web サービス・クライアント・ウィザードで、Java プロキシーの生成に加えて、オプションとして表示されるので、そこで開始できます。
- テスト・クライアントを生成するときの Web サービス・ウィザードのオプションです。

以下のステップを行って、UDF を生成します。

1. ウィザードの先頭ページから WSDL ファイルを指定します (171 ページの図 51 を参照)。この WSDL ファイルを使用して、UDF を生成します。

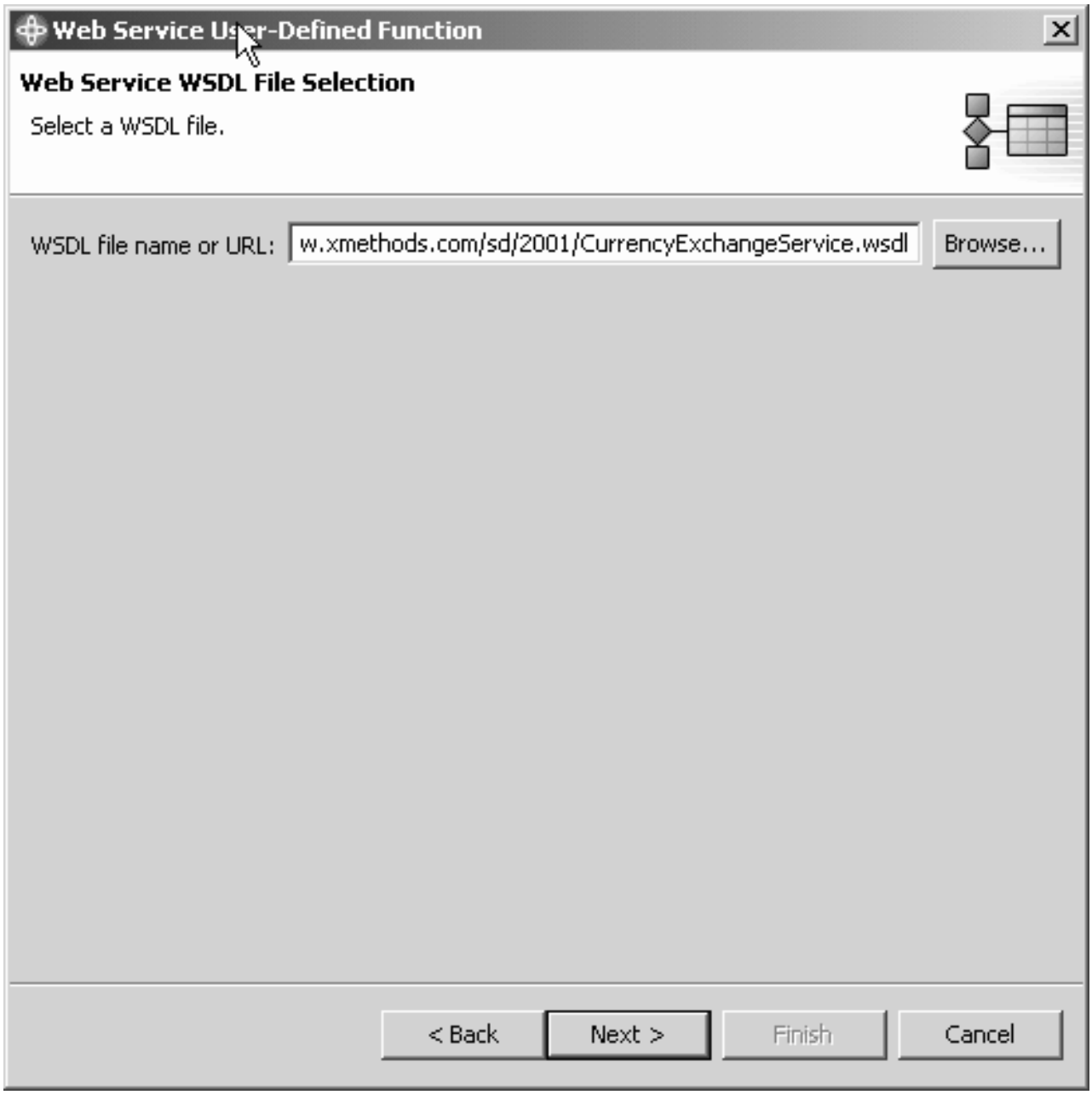


図 51. WSDL ファイルの選択

ワークスペースから WSDL ファイルを選択するか、または適切な URL を指定します。たとえば、通貨為替レート Web サービスは 2 つの国を入力パラメーターとして取り、その 2 国間の通貨為替レートを戻します。この WSDL ファイルは、*www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl* にあります。

2. データベースを選択します。 172 ページの図 52 には、データベース接続、および UDF が生成されているスキーマがあります。**「ブラウズ (Browse)」**プッシュボタンをクリックして、 WebSphere Studio ワークスペースからデータベース・スキーマを選択します。ウィザードは、データベースが Web サービス・コンシューマー UDF 用に使用可能になっていること、および DB2 XML エクステンダーを要求します。指定したデータベースへの接続が現在使用可能になっていない場合、追加のメッセージ・ウィンドウが表示されて接続情報を要求しま

す。生成された UDF を直ちにデータベースにデプロイするか、または UDF を WebSphere Studio ワークスペース内にのみ生成するかを選択できます。 UDF は後でデプロイすることも可能です。

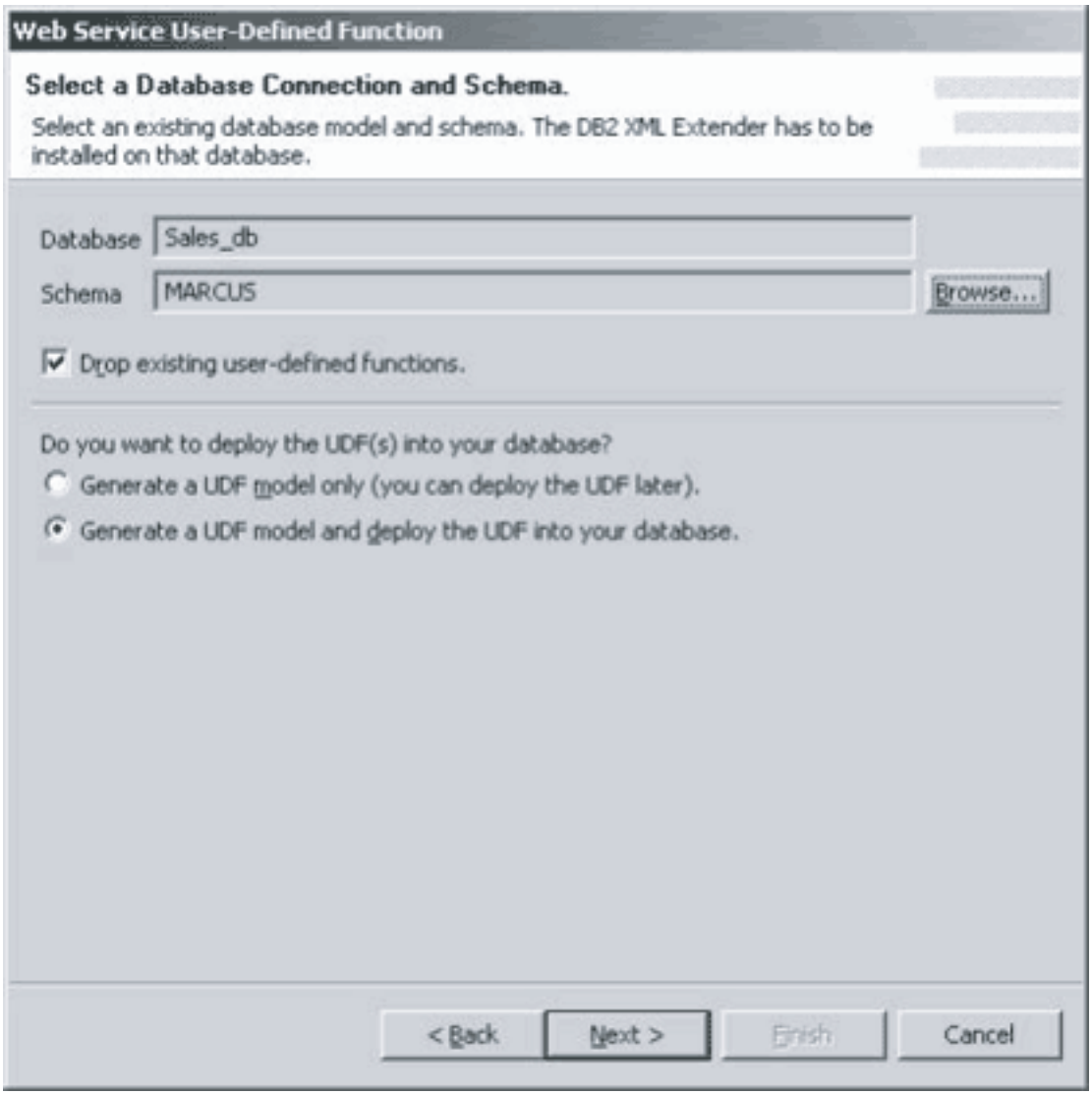


図 52. WSDL ページ 2

3. 作成したい UDF を選択します。 173 ページの図 53 に記述されている操作のリストから、作成したいものを選択します。ウィザードは、選択した操作ごとに 1 つの UDF を生成します。この例で使用されている Web サービス には 1 つの操作しか提供されていないので、ウィザードはそれを自動的に選択します。次のページに進みます。

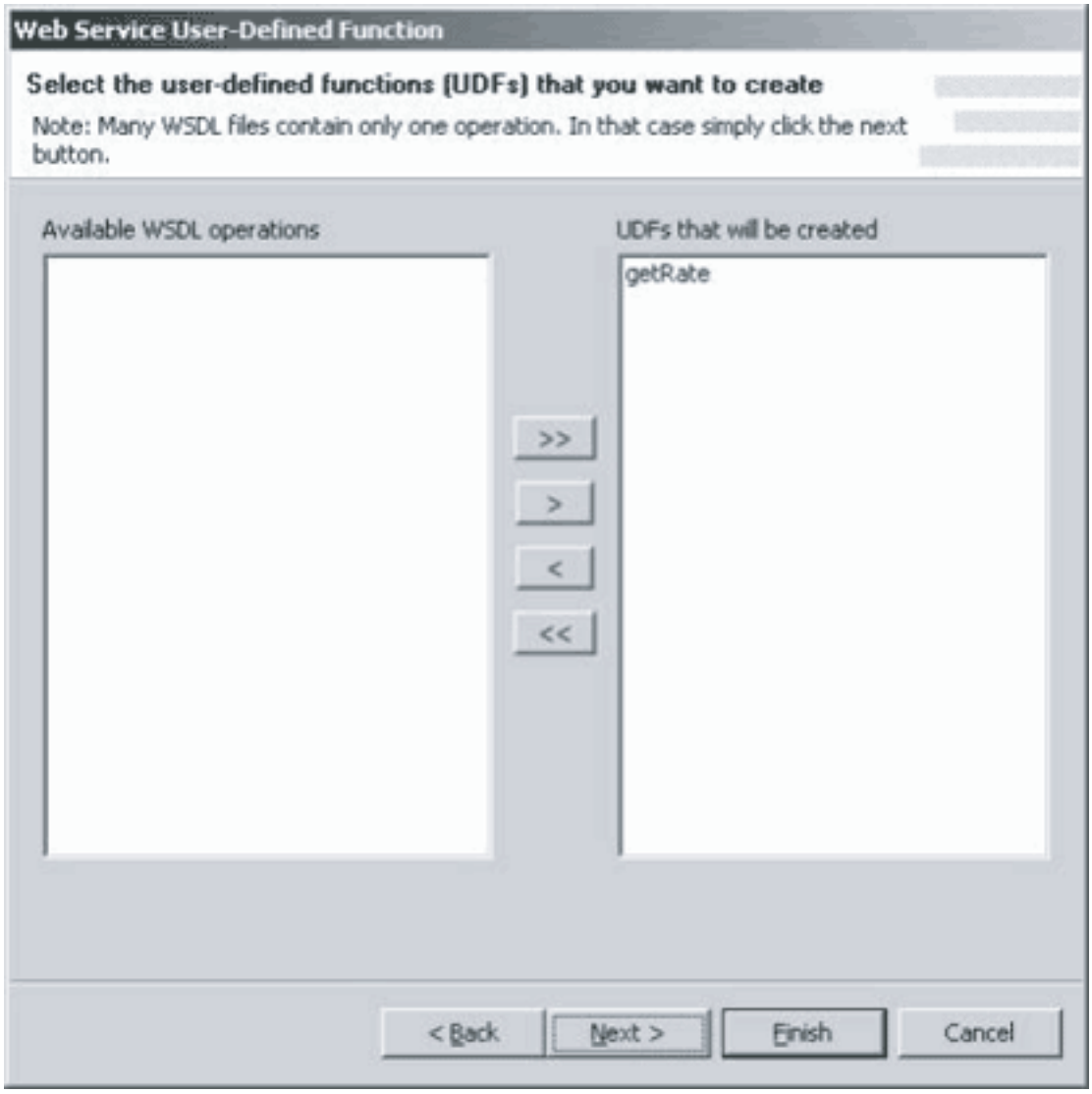


図 53. ウィザード ページ 3

4. UDF のオプションを選択します。前のステップで選択した各操作に対して、関数名を変更したり、関数にコメントを付けたりして UDF にオプションを定義することができます。 174 ページの図 54 および 175 ページの図 55 を参照してください。

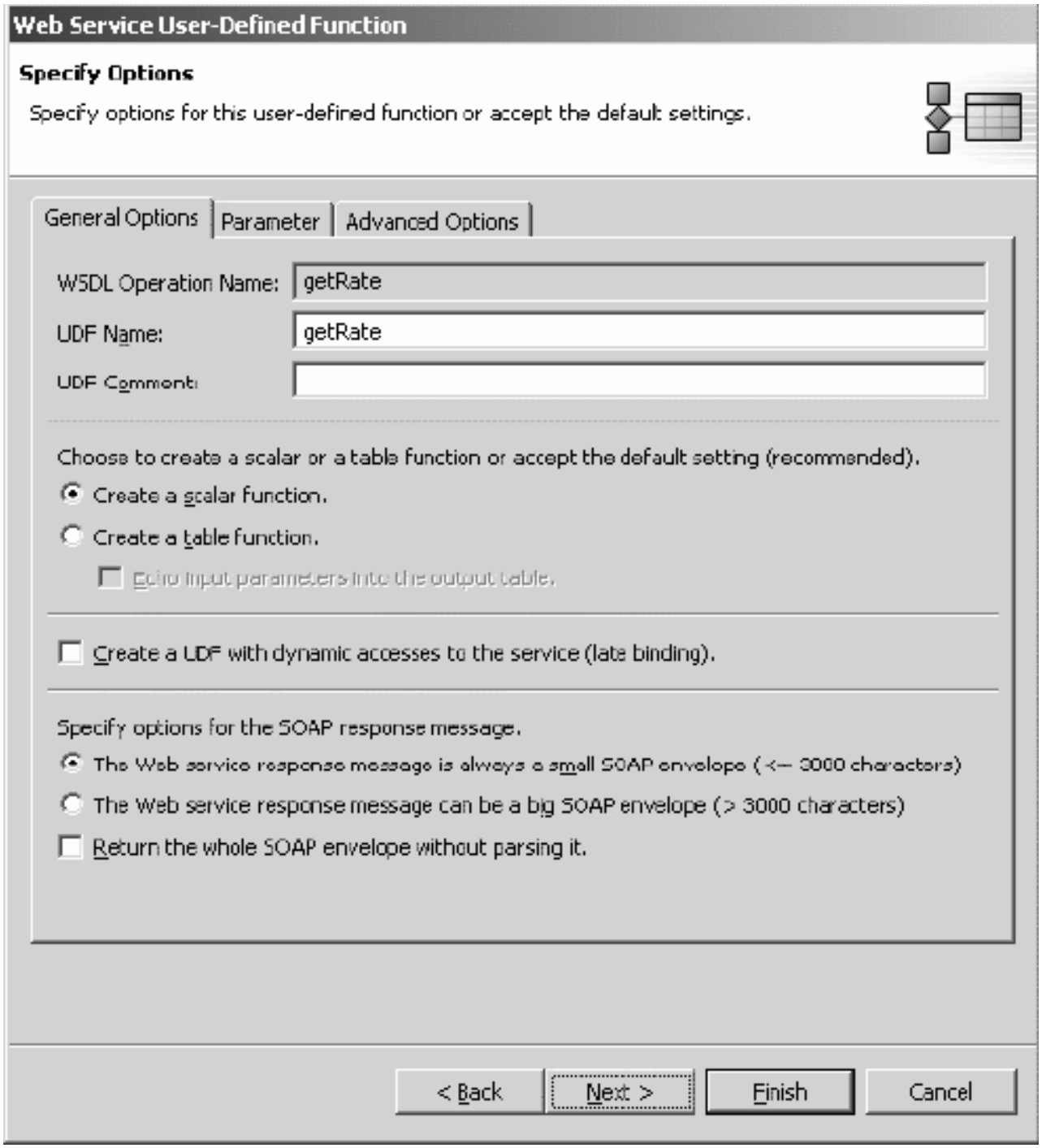


図 54. オプションの選択 ページ 1

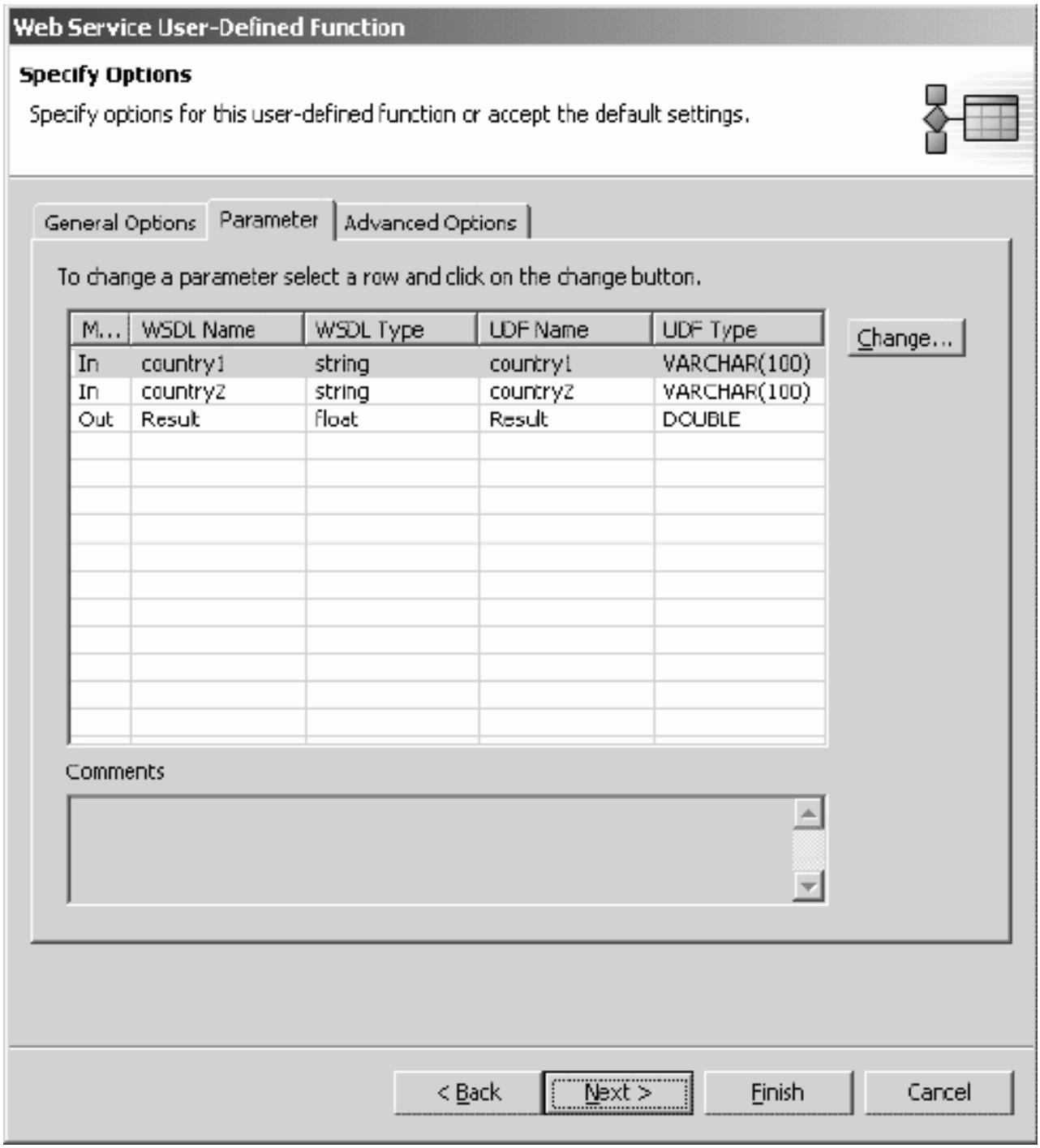


図 55. オプションの選択 ページ 2

- スカラー関数または表関数を作成することを選択できます。
  - Web サービスが単純 XML タイプを戻す場合は、ウィザードではスカラー関数が生成されます。
  - 複合 XML タイプを戻す場合は、ウィザードでは表関数が生成されます。表関数は自動的に複合 XML タイプを複数の列にマップします。

戻されたタイプをウィザードが自動的にマップしないようにするには、表関数からスカラー関数に切り替えたほうがよいでしょう。この場合、ウィザードは XML フラグメントとして戻します。スカラー関数から表関数に切り替えられるので、FROM 文節に UDF を使用できます、

- **「入力パラメーターを出力表にエコーする (Echo the input parameters into the output table)」**チェック・ボックスを選択して、入力パラメーターを列として出力表に組み込むことができます。
- Web サービスへの動的アクセスを含む UDF を生成することを選択できます。

  WSDL 文書にサービス・ロケーション (soap:address エレメントのサービス属性) を指定しない場合、動的関数が生成されます。サービス・ロケーションが遅延バインディングを利用するように指定されている場合でも、**「サービスへの動的アクセスを含む UDF を作成する (Create a UDF with dynamic accesses to the service)」**チェック・ボックスを選択できます。動的関数を生成した場合は、実行時に UDF のパラメーターとしてサービス・ロケーションを指定してください。
- 3000 文字以上の応答を戻すことが可能な Web サービスを使用している場合は、**「Web サービス応答メッセージが大きな SOAP エンベロープでもよい (The Web service response message can be a big SOAP envelope)」**ラジオ・ボタンを指定します。デフォルトでは、ほとんどの Web サービスにより良いパフォーマンスを提供するので、**「Web サービス応答メッセージは常に小さい SOAP エンベロープである (The Web service response message is always a small SOAP envelope)」**ラジオ・ボタンが指定されています。小さな SOAP 応答オプションを指定し、ウィザードが 3000 文字以上の SOAP エンベロープを戻すと、生成された Web サービス UDF は記述的なエラー・メッセージを戻します。
- Web サービス・コンシューマー UDF をデバッグする際に役立つ**「SOAP エンベロープ全体を構文解析せずに戻す (Return the whole SOAP envelope without parsing it)」** チェック・ボックスを選択します。

5. オプション・ウィンドウの「パラメーター (Parameter)」ページで、 WSDL タイプから SQL タイプへのパラメーター・マッピングを検討および変更できます (175 ページの図 55 を参照)。
6. 「拡張オプション (Advanced Options)」では、UDF の名前を指定できます。名前を指定しないと、UDF をデプロイするときに、ユニーク名がデータベースによって自動的に生成されます。
7. UDF の生成に関する設定を検討します。データベースとスキーマ、およびデータベース上で発行されることになる CREATE ステートメントを調べます (177 ページの図 56 を参照)。
8. **「次へ」**または**「完了」**プッシュボタンをクリックします。これで、生成とデプロイを実行することにしたこれまでの選択に基づいて、 UDF が生成され、データベースにデプロイされます。

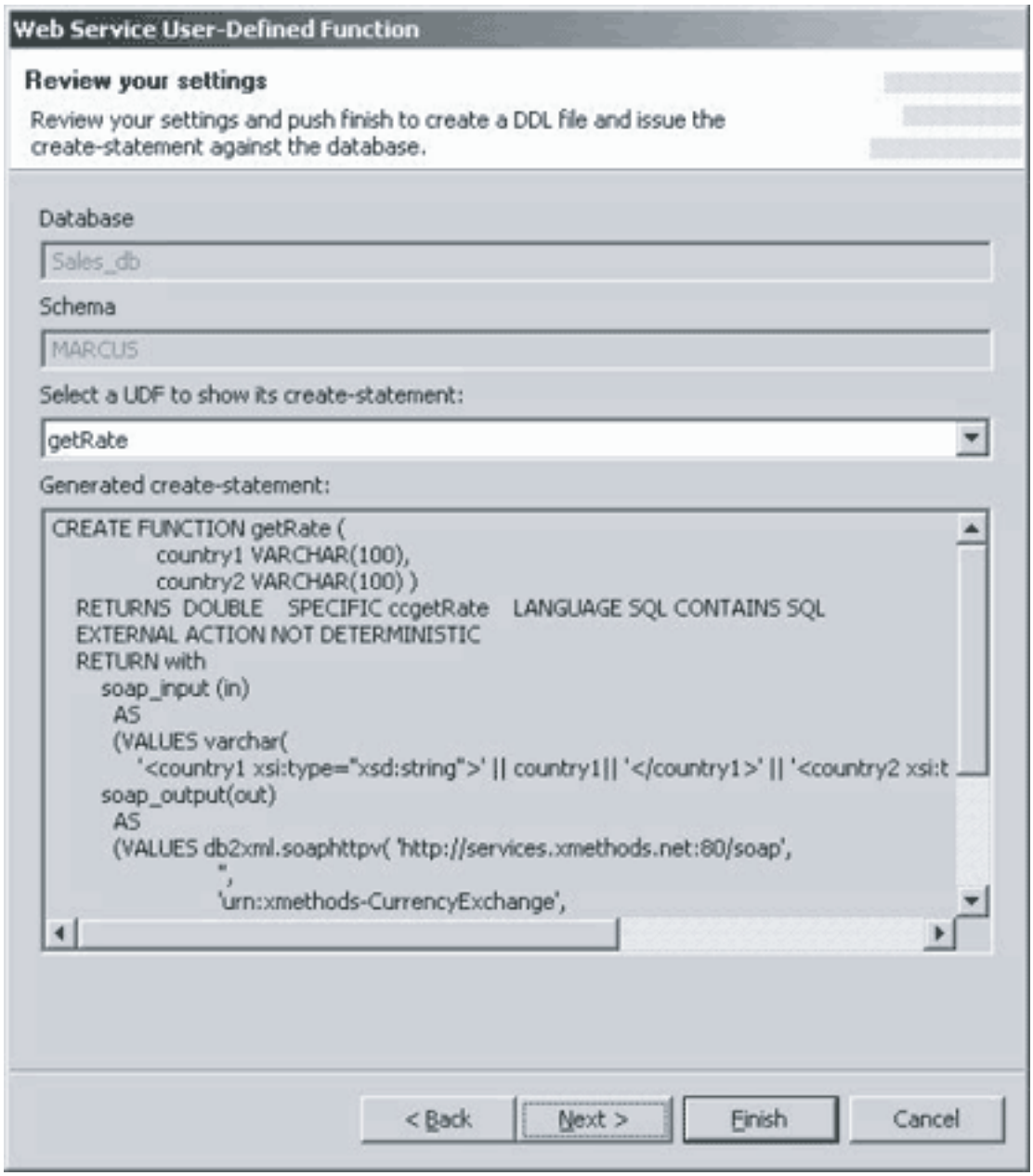


図 56. 検討

9. Web サービス・コンシューマー UDF をワークスペースから直接実行できます。デプロイされた UDF を実行するには、次のようにします。
   a. UDF を右クリックします。
   b. **「実行 (Run)」**を選択します (178 ページの図 57 を参照)。「設定の実行 (Run Settings)」ウィンドウが開きます (179 ページの図 58)。
   c. 「設定の実行 (Run Settings)」ウィンドウで、パラメーター値を指定できます。
   d. **「OK」**プッシュボタンをクリックして、テストの結果を表示します (179 ページの図 59)。

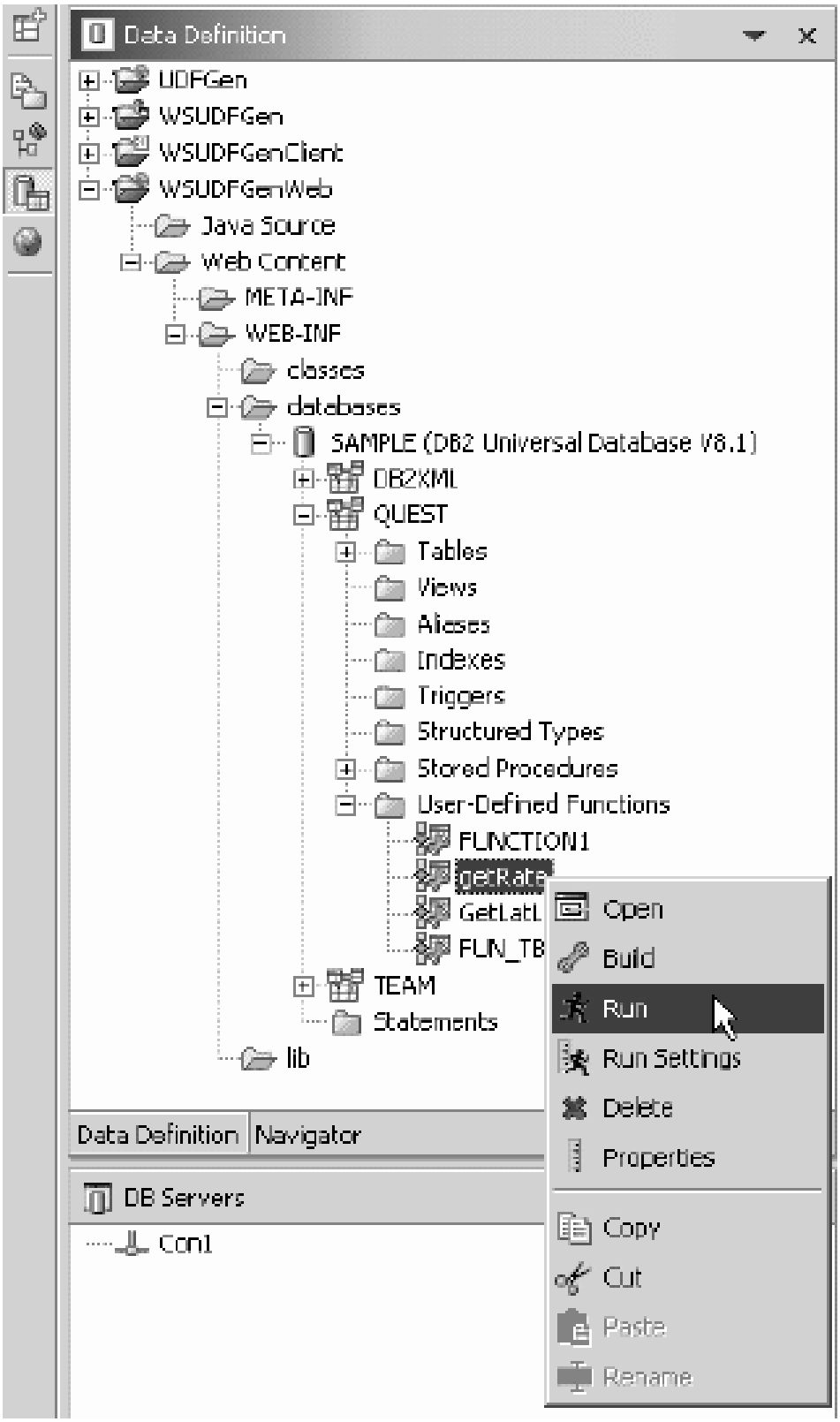

Data Definition
UDFGen
WSUDFGen
WSUDFGenClient
WSUDFGenWeb
Java Source
Web Content
META-INF
WEB-INF
classes
databases
SAMPLE (DB2 Universal Database V8.1)
DB2XML
QUEST
Tables
Views
Aliases
Indexes
Triggers
Structured Types
Stored Procedures
User-Defined Functions
FUNCTION1
getRate
GetLatL
FUN_TB
TEAM
Statements
lib
Data Definition
Navigator
DB Servers
Con1
Open
Build
Run
Run Settings
Delete
Properties
Copy
Cut
Paste
Rename

図 57. テスト

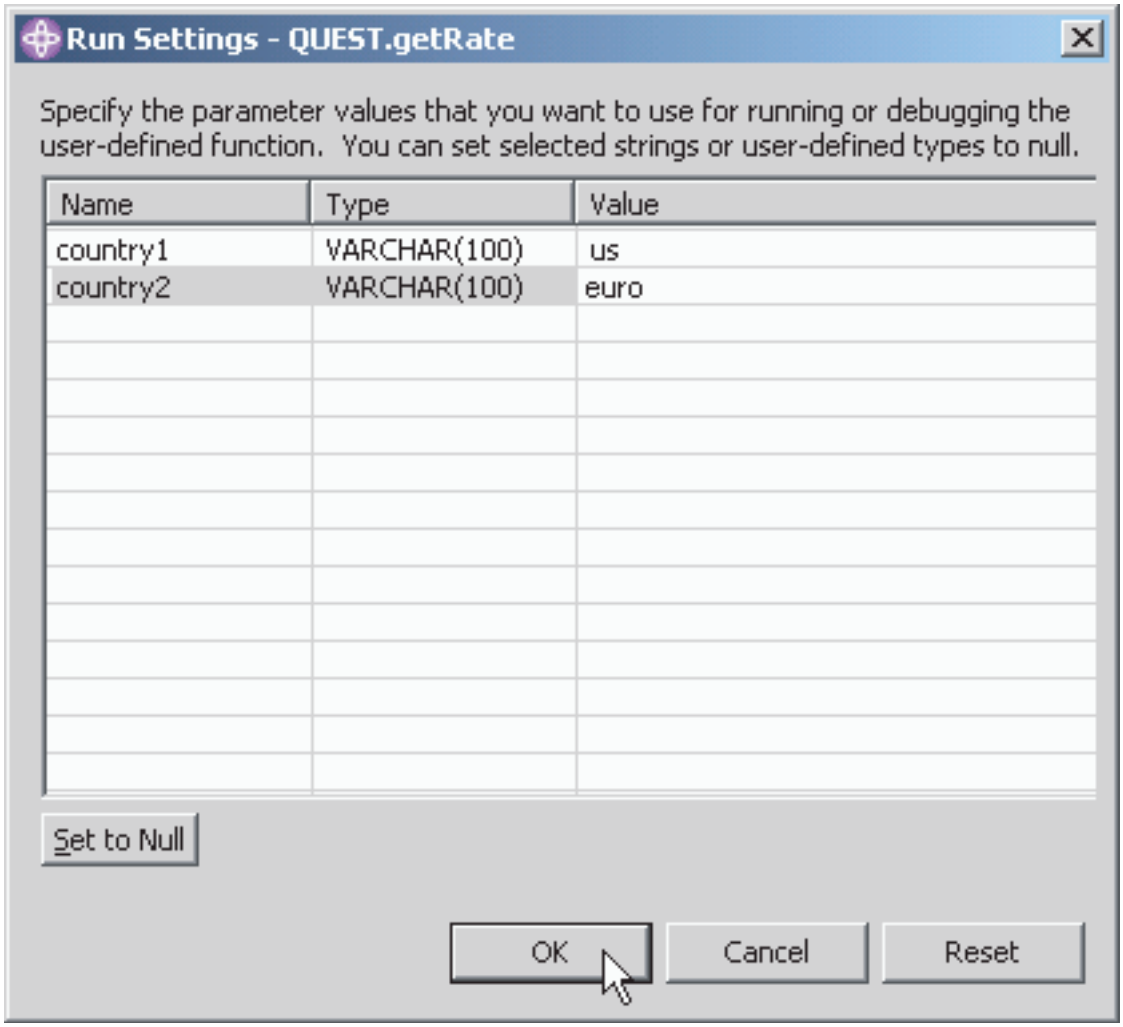


図 58. 設定の実行

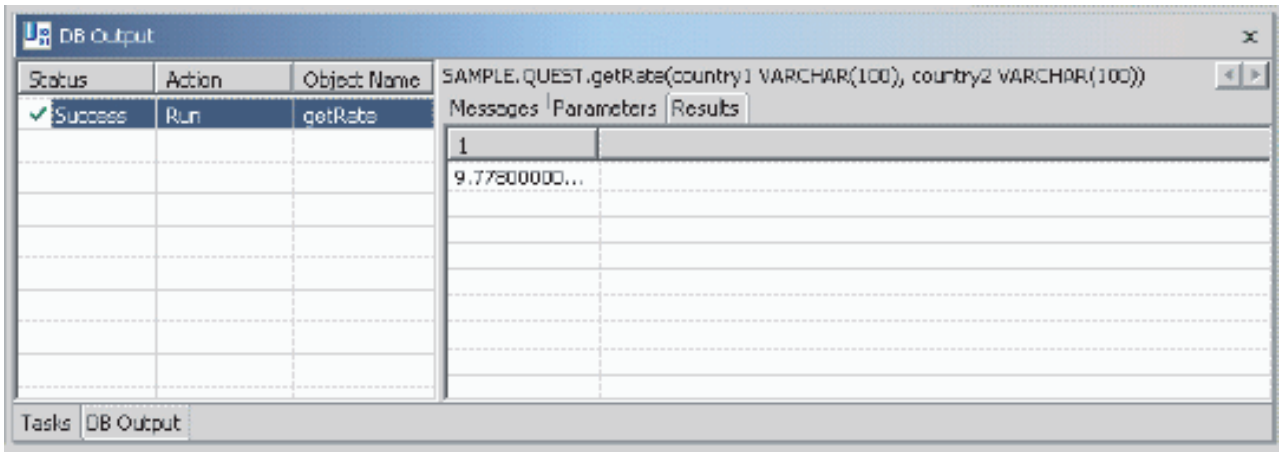


図 59. 結果

関連概念:

- 164 ページの『Web サービスのコンシューマー機能』

- 166 ページの『Web サービスのユーザー定義のコンシューマー機能』
- 169 ページの『Web サービス・コンシューマー — WebSphere Studio ユーザー定義関数ツールの使用』

関連タスク:

- 162 ページの『Web サービスのユーザー定義のコンシューマー機能のインストール』

関連資料:

- 180 ページの『Web サービス・コンシューマー UDF の使用』
- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『dxxEnableDB() ストアード・プロシージャー』

Web サービス・コンシューマー UDF の使用

リレーショナル表と Web サービスの間で情報を共用するために、この UDF を使用します。リレーショナル・データベースに、次のデータを含む表があるとします。

表 19. 製品表

製品	価格
Gear	950.00
Nut	25.00
Bolt	35.00

さらに、リモート表に通貨タイプに関する情報があるとします。

表 20. 通貨表

地域
US
EURO
UK

180 ページの SQL ステートメントを使用して、価格情報を米ドルの代わりにユーロで表示するために、通貨為替レート関数を使用する方法を決定します。これは、リアルタイム為替レートにアクセスします。ステートメントが組み込み 10 進数関数を使用して価格情報をキャストすることに注意してください。

```
SELECT product, decimal(getRate('us', 'euro') * price, 10, 2)
  as 'EUR_Price'
  FROM products
```

180 ページの ステートメントの結果は次のとおりです。

表 21. リアルタイム為替レートの使用

製品	EUR_Price
Gear	1019.82
Nut	26.84
Bolt	37.57

リレーショナル・データを Web サービスへの入力としてどのように使用できるかを示すには、次の SQL ステートメントを使用します。 181 ページの 例は、通貨為替レート関数が価格情報を異なる通貨でどのように表示するかを示しています。

```
SELECT p.product, c.area,
    decimal(getRate('us', c.area) * price, 10, 2)
  as Price
  FROM products, areas
```

結果は次のとおりです。

表 22. 別の通貨での価格情報の表示

製品	地域	価格
Gear	us	950.00
Nut	us	25.00
Bolt	us	35.00
Gear	euro	1019.82
Nut	euro	26.84
Bolt	euro	37.57
Gear	uk	650.84
Nut	uk	17.12
Bolt	uk	23.97

この照会を頻繁に使用する場合は、よりシンプルなインターフェースを提供するビューを定義することもできます。このビューの例を次に示します。

```
CREATE VIEW prices AS
  SELECT p.product, c.area,
      decimal(getRate('us', c.area) * price, 10, 2)
      as Price
  FROM p.products, c.areas
```

このビューを使用して、次の簡潔な照会をコード化できます。

```
SELECT * FROM prices
```

FROM 文節内で表関数として生成された UDF をどのように使用できるかを示すために、次の SQL ステートメントを使用します。この例は、getRate-UDF を表関数として再生成します。入力パラメーターは出力表にエコーされます。

```
SELECT t.*
FROM countries c,
table( getRate('us', c.countries) ) t
```

結果は次のとおりです。

表 23. 表関数として getRate を使用

AREA1	AREA2	RESULT
us	us	+1.00000000000000E+000
us	euro	+1.07280000000000E+000
us	uk	+6.84800000000000E-001

関連概念:

- 164 ページの『Web サービスのコンシューマー機能』

- 166 ページの『Web サービスのユーザー定義のコンシューマー機能』

関連タスク:

- 168 ページの『Web サービス・コンシューマー・イベントのトレース』

Web サービス・コンシューマーの例

ここで参照される例は、DB2 Universal Database バージョン 8 で使用するものです。ファイル *<DB2_installed path>/samples/soapsample.sql* には、サンプルの実行方法が説明されています。ファイル *soapsample.sql* には、以下の例のリストおよびサンプル照会が含まれています。

- getTemp - 華氏で温度を検索する
- getRate - 任意の 2 つの通貨間の為替レートを戻す

関連概念:

- 164 ページの『Web サービスのコンシューマー機能』
- 166 ページの『Web サービスのユーザー定義のコンシューマー機能』
- 169 ページの『Web サービス・コンシューマー — WebSphere Studio ユーザー定義関数ツールの使用』

関連タスク:

- 162 ページの『Web サービスのユーザー定義のコンシューマー機能のインストール』

関連資料:

- 180 ページの『Web サービス・コンシューマー UDF の使用』

第 3 章 フェデレーテッド・アプリケーション、ウェアハウス・アプリケーション、およびメッセージ・キュー・アプリケーションの開発

第 3 部では、フェデレーテッド・アプリケーションおよびウェアハウス・アプリケーションの開発方法についてと、メッセージ・キュー (MQ) 機能の使用方法について説明します。

フェデレーテッド・サーバーを使用するアプリケーションの開発

フェデレーテッド・システム内の DB2 Universal Database サーバーはフェデレーテッド・サーバーと呼ばれています。透過なデータ・アクセスを可能にするために、アクセスしたいリモート・データのニックネームを作成できます。異なるデータ・ソース間の差異を補正し、また本来サポートされていない機能をシミュレートする関数を作成する、またはすでに定義されているそのような関数を使用することができます。マルチサイト結合および合併は、複数のソースからのデータの統合を促進します。

フェデレーテッド・システムの利点

フェデレーテッド・システムは分散データベース管理システムの 1 つのタイプであり、1 つの SQL ステートメントで複数のデータ・ソースに分散要求を送信することができます。 IBM® DB2® Information Integrator のフェデレーテッド・システムは、Web アプリケーション・サーバーによって提供された組み込みデータベース・サポートを補完します。

フェデレーテッド・システムなしで異種混合のデータ・ソースにアクセスするには、以下の複数のステップが必要です。

1. 各データ・ソースに個別に接続しなければならない。
2. いくつかの異なる固有のアプリケーション・プログラミング・インターフェースを使用して、必要なデータを取り出さなければならない。
3. データを手動でフィルターし、ソートし、統合しなければならない。

フェデレーテッド・システムを使用すると、SELECT、INSERT、UPDATE、および DELETE ステートメントを使用して簡単にニックネームを照会できます。

フェデレーテッド・システムは、複数の異機種混合ソースにまたがるデータへの透過的アクセスも提供します。フェデレーテッド・システムにより、リモート・データ (物理的に保管されたデータ、あるいは動的に生成されたデータ) をサポートする上での Web アプリケーション・サーバーの効用と能力が高まります。フェデレーテッド・システムを、 Enterprise Bean や Web サービスなどのアプリケーション・サーバー・コンポーネントと共に使用できます。 Enterprise Bean とフェデレーテッド・システム・オブジェクトを使用することにより、プログラマーはデータベース

操作またはトランザクション処理、および複数のデータ・ソースへのアクセスを実行することや、異種混合データを統合するアプリケーションの作成を行うことができます。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・システム』
- 186 ページの『フェデレーテッド・システムでの Enterprise Bean』
- 10 ページの『DB2 Information Integrator — 非リレーショナル・フェデレーテッド・テクノロジー』
- 25 ページの『パフォーマンスと調整の計画 — フェデレーテッド・システム中のマテリアライズ照会表』
- 9 ページの『DB2 Information Integrator — リレーショナル・フェデレーテッド・テクノロジー』

IBM DB2 Information Integrator で照会を設計する利点

データベース管理者は、DB2® Universal Database を使用して、異なるデータ・ソースからの複数の表にまたがったデータ・ビューを作成できます。フェデレーテッド・システムの場合、ビューには複数のサーバーに異なるフォーマットで保存されているデータが含まれます。このようなビューを作成するには、リモート・データ・オブジェクトのニックネームを作成します。次に、SQL を使用してこれらのニックネームを結合または合併するビューを作成します。複数のデータ・ソースを結合または合併するビューは、読み取り専用ビューです。これらのビューにマップするコンテナー管理パーシスタンス Entity Bean を構築することができます。それらの Bean は読み取り専用 Bean です。

フェデレーテッド・ビューに加えて、フェデレーテッド・システムは、リレーショナル・データベース・マネージャーの SQL データ定義言語 (DDL) の透過性を組み込んでいます。 DDL の透過性により、OPTIONS 文節を使用して DB2 UDB 表を作成し、 1 つのステートメントで 2 つの別個のタスクが実行できるようになります。 図 60 の例に示すように、リモート・データ・ソースで表を作成して、この表の DB2 UDB ニックネームを作成します。

```
    CREATE TABLE orarest (
id int primary key not null,
name varchar(20),
cuisine varchar(20),
budget int)
OPTIONS (remote_server 'ORACLE8',
         remote_schema 'ORACLEUSER1')

          CREATE TABLE msrest (
id int primary key not null,
name varchar(20),
cuisine varchar(20),
budget int)
OPTIONS (remote_server 'MSSQL',
         remote_schema 'MSUSER1')
```

図 60. DDL 透過性の例

OPTIONS 文節付きの create ステートメントにより、以下のデータベース・オブジェクトが作成されます。

- ORAREST という表 (リモートの Oracle データベース内)
- ORAREST というニックネーム (DB2 UDB フェデレーテッド・データベース内)
- MSREST という表 (リモートの Microsoft® SQL Server データベース内)
- MSREST というニックネーム (DB2 UDB フェデレーテッド・データベース内)

次にビューに生成された表とニックネームを使用して、2 つの異種混合データ・ソースの情報を合併または統合できます。

```
CREATE VIEW multirest
   (id, name, cuisine, budget)
  AS
  SELECT id, name, cuisine, budget
  FROM orarest UNION
   SELECT id, name, cuisine, budget FROM msrest;
```

DDL 透過性を使用すれば、データベース・マネージャーはほとんどの場合に SQL 変換を実行して、リモート表を適切に作成することができます。有効な表を作成するために、データ・ソースに合う特定の SQL 構文を学習する必要はありません。

マテリアライズ照会表をアプリケーションに組み込んで、パフォーマンスを改善することも可能です。マテリアライズ照会表を使用すると、各照会の全体または一部を事前に計算しておき、今後の照会に答えるために計算結果を使用できます。マテリアライズ照会表により、事前に照会した結果を保存しておき、共通する紹介結果を後続の照会で再利用することが可能になります。マテリアライズ照会表は、スキャン、集約、および結合の重複を避けるのに役立ちます。

- マテリアライズ照会表により、リモート・データ・ソースが使用不可 (たとえばネットワークが使用不可) な場合でさえ、照会の処理が可能になります。リモート表のマテリアライズ照会表はその表のキャッシュと考えることができます。これにより、システム使用可能性が高まったように見えるかもしれません。
- マテリアライズ照会表はシステム全体のスケーラビリティーを増加します。マテリアライズ照会表は、1 次データベースの負荷を軽減します。ローカル・データベースを多数設け、それぞれ頻繁に使用する 1 次データのコピーのサブセットを入れておくことができます。 1 次データベースの企業に対する制約は少なくなります。
- マテリアライズ照会表を使用すれば、照会によってはリモート・システムに接続しなくても済みます。全体的なシステム・スループットの増加と、応答時間の合計の減少につながる可能性があります。
- マテリアライズ照会表は、リモート・データ・ソースにアクセスするときに DB2 UDB の全機能を使用します。リモート・データ・ソースがマテリアライズ照会表をサポートしなくても、マテリアライズ照会表は有益です。

データベース環境内で開発された Web 対応アプリケーションは、 Java™ 2 Enterprise Edition (J2EE) サーバー環境のいくつかのコンポーネントを使用できます。 Enterprise Beans、サーブレット、JavaServer Page コード、および Java Naming and Directory Interface 拡張機能のサポートが組み込まれている J2EE コンポーネントもあります。 J2EE はデータベース・マネージャーへの接続およびアク

セスのサポートも提供します。その中には、Java Database Connectivity コードと Java トランザクション・アプリケーション・プログラミング・インターフェースのサポートも含まれます。

関連概念:

- 「フェデレーテッド・システム・ガイド」の『透過 DDL とは ?』
- 「フェデレーテッド・システム・ガイド」の『マテリアライズ照会表とフェデレーテッド・システムの概説』
- 「フェデレーテッド・システム・ガイド」の『照会処理のチューニング』
- 186 ページの『フェデレーテッド・システムでの Enterprise Bean』
- 25 ページの『パフォーマンスと調整の計画 — フェデレーテッド・システム中のマテリアライズ照会表』

関連タスク:

- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・マテリアライズ照会表の作成』

フェデレーテッド・システムでの Enterprise Bean

Enterprise Beans は、 Web サーバー上で実行される Java™ コンポーネントです。IBM® DB2® Information Integrator フェデレーテッド・システムに作成したニックネームにマップするコンテナー管理のパーシスタンス Entity Bean を作成できます。コンテナー管理のパーシスタンス Entity Bean は、リレーショナル・データベースに置かれているデータにアクセスできます。読み取り専用のコンテナー管理のパーシスタンス Entity Bean は、非リレーショナル・データベースに置かれているデータにアクセスできます。 Entity Bean を使用し、Enterprise JavaBean アーキテクチャーを通じて異種データを統合できます。

Enterprise JavaBean のアーキテクチャー

Enterprise bean のコンポーネントは Enterprise JavaBean アーキテクチャーの一部であり、Enterprise JavaBean コンテナーで実行されます。コンテナーは Enterprise JavaBean サーバー内で実行されます。 Enterprise Bean はビジネス・ロジックを実装します。 Enterprise JavaBean コンテナーは、トランザクションおよびリソース管理、永続性、およびセキュリティーなどのサービスを Enterprise Bean コンポーネントへ提供します。データベース操作の詳細は Enterprise JavaBean コンテナーに任せることもできます。

Entity Bean と Session Bean

フェデレーテッド・システムにおいて重要な 2 種類の Enterprise Bean があります。Session Bean と Entity Bean です。

Session Bean

Session Bean は通常、単一のクライアントと関連し、通常は永続しません。Session Bean の目的は、既存のデータベース内容を表したり更新したりすることではありません。むしろ Session Bean の目的は、サーバーに何らかのアクションを行う単一のクライアントとして機能することです。

Entity Bean

Entity Bean はデータベースに継続的に保存される情報を表します。 Entity Bean は データベース・トランザクションと関連があります。 Entity Bean は複数のユーザーにデータ・アクセスを提供できます。 Entity Bean は基礎となるデータベース行、または SELECT ステートメントの結果 (単一行) を表す場合があります。

フェデレーテッド・システムは、単一のコンテナー管理パーシスタンス Entity Bean の自動開発およびデプロイメントをサポートし、この Bean の属性は複数のリソースからのデータにマップされます。コンテナー管理パーシスタンス Entity Bean がフェデレーテッド・データベース・オブジェクトにマップする際には、フェデレーテッド・サーバーはデータベース・アクセスを、データ・ソースに該当するデータベース・アクセス要求に透過的に変換します。 Enterprise Bean をデプロイすると、Bean はパーシスタンスのサポートなどのサービスを備えているコンテナーの中に入れられます。 Entity Bean は、Enterprise Bean をデプロイする際に永続性を実装するコードを自動的に生成します。それとは対照的に、永続データにアクセスする Session Enterprise Bean を作成する場合は、独自に Java Database Connectivity ステートメントを書いてデータベース接続を確立し、 SQL ステートメントを出さなければなりません。

コンテナー管理パーシスタンス Entity Bean はデータベースとのすべての対話を Enterprise JavaBean コンテナーにゆだねます。通常は、Enterprise Bean がデータベースからデータを読み取り、そのデータをコンテナー管理パーシスタンス Entity Bean のフィールドに配置します。 Entity Bean でデータを参照または更新できます。トランザクションが終了するときに、Enterprise JavaBean コンテナーは Entity Bean 内のデータにアクセスして、表の基本行を更新します。

関連概念:

- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『Enterprise Java Beans』

関連タスク:

- 195 ページの『コンテナー管理のパーシスタンス Bean の作成とデプロイ』

従業員のスキルのシナリオ - ソリューションの設計

YBar, Incorporated は、DB2® Universal Database 表および XML 文書に従業員の情報を保管しています。 YBar, Incorporated は DB2 UDB 表および XML 文書にあるデータを操作できます。この論理ソリューションは、従業員の履歴書を Extensible Markup Language (XML) 文書として従業員データベースに直接保管するアプリケーションを作成するためのものです。次に会社は XML 検索機能を使用して、特定のプロジェクトに必要とされるスキルと一致する履歴書を検索します。

YBar, Incorporated は、リレーショナル・データとして XML フォーマットで保存されている履歴書ファイルにアクセスするために、 IBM® DB2 Information Integrator のフェデレーテッド・システムを使用し、業務問題を解決します。

1. フラット・ファイル・データ・ソース内のデータ・アクセスのパフォーマンスを向上させるために、 YBar, Incorporated は、DAD ファイルで定義されたサイド

表に索引を作成します。 DB2 XML エクステンダーは DAD ファイルのスキーマ定義に基づいてサイド表を作成します。

```
CREATE INDEX KEY_Skill ON
      resume_skills_sidetable(skill)
```

2. YBar, Inc のデータベース管理者は、リレーショナル表の従業員データに 3 つの行を挿入します。 Resume という列は XML 列です。
   XMLVARCHARFROMFILE は、XML 文書をサーバー・ファイルから読み取り、文書を XMLVARCHAR タイプで戻す、DB2 XML エクステンダー関数です。 Resume 列の値は 'Some_Path'¥<履歴書ファイルの内容> にある履歴書ファイルから読み込まれます。

```
INSERT INTO Employee
    (Emp_ID, Lastname, Firstname,
    Dept_ID, Current_job_ID, Resume)
    VALUES(12,'Douglas','Laurie',
      123, 1,
      db2xml.XMLVarcharFromFile
      ('Some_Path'¥resume_ld.xml'))

INSERT INTO Employee
    (Emp_ID, Lastname, Firstname,
    Dept_ID, Current_job_ID, Resume)
    VALUES(13,'Smith','John',
      123, 2,
      db2xml.XMLVarcharFromFile
      ('Some_Path'¥resume_js.xml'))

INSERT INTO Employee
    (Emp_ID, Lastname, Firstname,
    Dept_ID, Current_job_ID, Resume)
    VALUES(14,'Jackson','George',
      123, 3,
      db2xml.XMLVarcharFromFile
      ('Some_Path'¥resume_gl.xml'))
```

3. 次に YBar, Incorporated の開発者は、XML 列とサイド表を使って基本表を照会します。この SQL ステートメントは Java™ のスキルを持った全従業員を検索します。索引アクセスを使用するため、サイド表を利用します。

```
SELECT e.firstname, e.lastname,e.resume
  FROM employee
   AS e, resume_skills_sidetable AS r
    WHERE e.emp_id=r.emp_id
      AND r.skill LIKE '%Java%'
```

YBar, Inc. の開発者は以下のいくつかの方法で XML 文書を照会できます。

- 以下の例では、スカラー関数を使用して 1 人の従業員の経歴を戻します。

```
SELECT db2xml.EXTRACTVARCHAR
   (resume,'/resume/experience')
   FROM employee
   WHERE emp_id=13
```

  ユーザー定義関数 extractVarchar は、列 resume を入力として、ロケーション・パス /resume/experience を選択 ID として使用します。スカラー関数は 1 人の従業員の経歴を戻します。 WHERE 文節によって、抽出関数は ID が「13」の履歴書だけを評価します。

- 以下の例では、表関数を使用していくつかの行を戻します。

```
SELECT e.firstname, e.lastname, e.resume
   FROM employee
   AS e,
    TABLE(db2xml.EXTRACTVARCHARS(e.resume,'/resume/skill'))
    AS t
   WHERE t.returnedvarchar LIKE '%Java%'
```

ユーザー定義関数 extractVarchars は、列 resume を入力として、ロケーション・パス /resume/skill を選択 ID として使用します。表関数は、履歴書ファイルのスキル・ノードに「Java」というストリングが含まれている従業員のスキルを戻します。以下にオリジナルの resume.xml ファイルの一部を例として示します。

```
<resume emp_id="12" email_address="some_person@email_address.com">
  <experience start_date="12/01/1999" end_date="06/30/2002">
    Develop partner and customer demos.
  </experience>
  <experience start_date="12/01/1999" end_date="06/30/2002">
    IBM DB2 Information Integrator development
  </experience>
  <skill>
    Databases
  </skill>
  <skill>
    Java
  </skill>
  <skill>
    C++
  </skill>
  <skill>
    FORTRAN
  </skill>
</resume>
```

外部教育の問題の解決

人事部は、従業員のリストおよび現行のジョブ記述を外部の教育機関に送る必要があります。これによって教育機関は YBar, Incorporated の従業員に合わせた授業を行えます。 YBar, Inc. の開発者は従業員データベースとジョブ・データベースを結合し、 WebSphere® MQ を使用してメッセージ・キューにリストを発行します。以下のステップはソリューションの例を示しています。

1. ジョブ表は、外部フラット・ファイルのフェデレーテッド・システム・ニックネームとして作成されます。ニックネームはフラット・ファイル・ラッパーの一部として作成されます。

   ```
   CREATE NICKNAME job
   (Job_ID INTEGER,
   Job_description VARCHAR(255),
   Title VARCHAR(50),
   responsibilities VARCHAR(100))
   FOR SERVER local_flat_files OPTIONS
   (FILE_PATH 'C:¥SPC¥job_table.txt',
   COLUMN_DELIMITER ',',
   KEY_COLUMN 'job_id',
   VALIDATE_DATA_FILE 'y')"
   ```

   FILE_PATH オプションは job_table.txt という外部ファイルの場所を定義します。
2. 開発者は従業員の情報とジョブの情報を選択する照会を実行します。

```
SELECT 1 as x, emp_id , firstname, lastname,
Job_description
FROM employee
as e, job as j
WHERE e.Current_Job_ID = j.Job_ID
and j.Job_ID != 1000
ORDER BY x, emp_id
```

3. YBar の開発者は新しい XML 文書を生成します。これには、DB2 UDB 表 employee と外部フラット・ファイル情報 job からの結合された情報が入ります。新しく生成するファイルの名前は All_employee.xml です。開発者は employee.dad ファイルを使用して、XML 情報を DB2 UDB リレーショナル表の情報にマップします。
4. 以下のステートメントを使用して、XML 文書を保持する DB2 UDB 一時表を作成します。

```
create table tmpTable
    (x_doc DB2XML.XMLCLOB not logged)
```

5. employee.dad にあって、tmpTable に読み込まれた XML 情報を分解します。
6. 以下の例を使用して、XML データを employee.xml ファイルに書き込んで保管します。

```
select DB2XML.Content
    (x_doc, 'c:¥YourName¥employee.XML')
    from tmpTable
```

7. 情報を外部の教育機関に対してメッセージとして送信する準備ができました。結合した情報を WebSphere MQ メッセージ・キューに入れます。
8. 以下の例では、メッセージを XML 文書としてキューに書き込み、メッセージ・キューのコンテンツを表示し、メッセージをそのままにしておきます。

   MQSENDXML 関数を使用して XML メッセージをキューに送信します。メッセージは従業員番号 12 の履歴書の情報です。

```
db2 "SELECT db2xml.MQSENDXML('DB2.DEFAULT.SERVICE',
    'DB2.DEFAULT.POLICY', resume)
    FROM employee
    WHERE emp_id=12"
```

   XML 文書全体を保持する 1 つの列を持った DB2 UDB 表を作成します。

```
db2 "CREATE TABLE temporaryXML
    (XML_doc DB2XML.XMLVARCHAR)"
```

   以下のようにして、DB2 UDB 従業員表と外部ジョブ・ファイルを結合した XML 情報を表に追加します。

```
db2 "INSERT INTO temporaryXML
    SELECT CONCAT('<FullName>'||firstname
    ||' '||lastname||'</FullName>          ','<Job_Desc>'||
    job_description||
    '</Job_Desk>')
    FROM employee
    AS e, job AS j
    WHERE e.current_job_id=j.job_id
    AND  j.Job_ID = 1"
```

   MQSENDXML 関数を使用して XML メッセージをキューに送信します。メッセージは XML 列にある、従業員とジョブの情報です。

```
db2 "SELECT db2xml.MQSENDXML('DB2.DEFAULT.SERVICE',
    'DB2.DEFAULT.POLICY', XML_doc)
    FROM temporaryXML"
```

YBar, Incorporated が使用するコンポーネントの完全セットは、図 61 にあります。

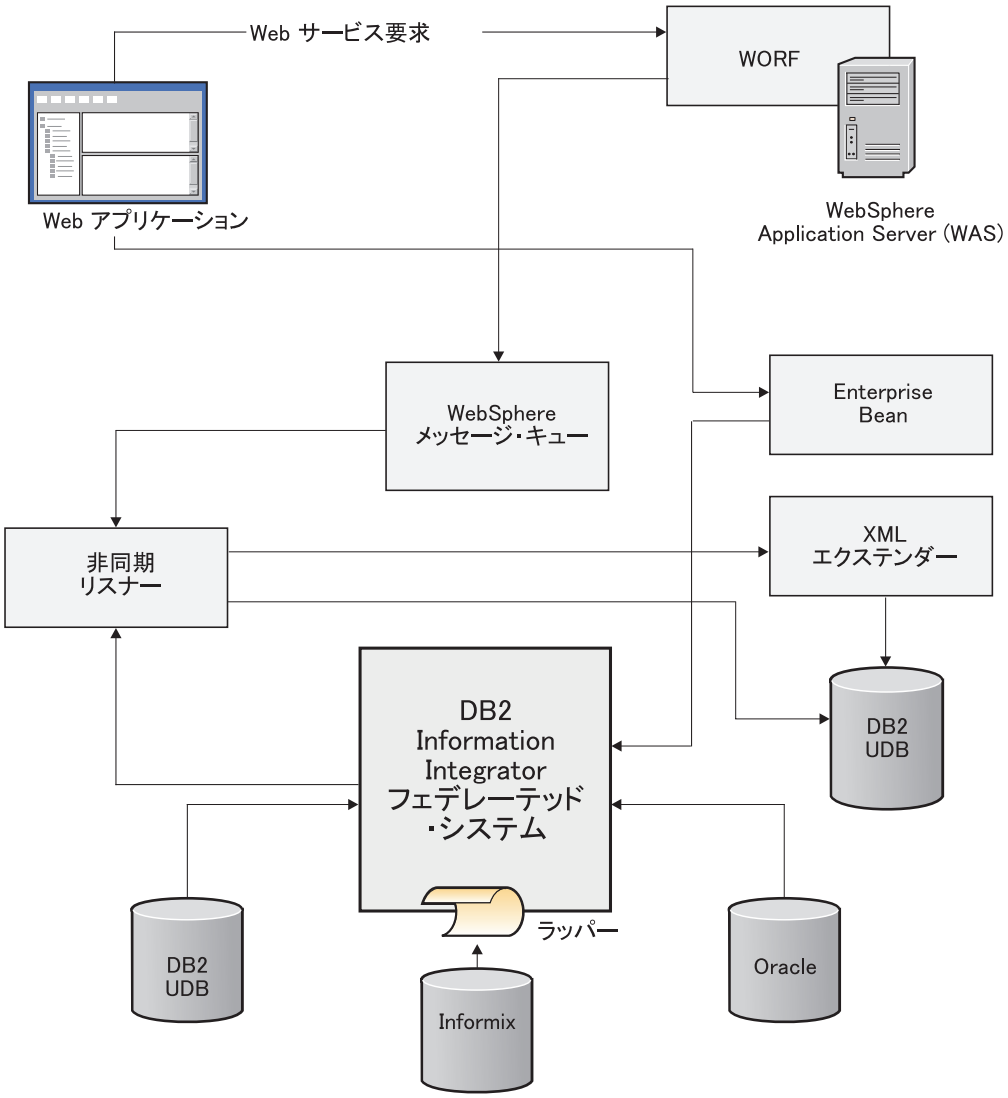


図 61. Web アプリケーションを使用するフェデレーテッド・システムのコンポーネント

関連概念:

- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『サイド表の計画』
- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『XML 列データの索引の使用』
- 192 ページの『従業員のデータベースのシナリオ - ソリューションの設計』
- 15 ページの『データの発見 — 従業員のスキルのシナリオ』

従業員のデータベースのシナリオ - ソリューションの設計

YBar, Incorporated, は人的資源を専門とする会社で、以下の 2 つの問題を解決しようとしています。

- 欠員のある社内の職種に配属するのに最適の候補者を特定する必要があります。XML 形式で保存されている、社内にいる人員の履歴書ファイルをもとに特定します。
- 従業員のリストを外部の教育機関に提供します。教育機関は従業員の必要に合わせた授業を行います。

YBar, Inc, は、IBM® DB2® Information Integrator のフェデレーテッド・システムを使用してデータの保管や検索を行うことにより、業務問題を解決します。 YBar, Inc. 設計ソリューションは 192 ページの図 62 にあります。この設計では DB2 XML エクステンダーのいくつかの機能を使用し、データ・タイプ定義ファイルのスキーマに一致するサイド表を活用します。サイド表は、頻繁に検索される XML 文書内容を抽出するために使用される DB2 UDB の表です。 XML 列は、XML 文書の内容を保持しているサイド表に関連付けられます。 XML 文書がアプリケーション表で更新されると、サイド表の値も自動的に更新されます。

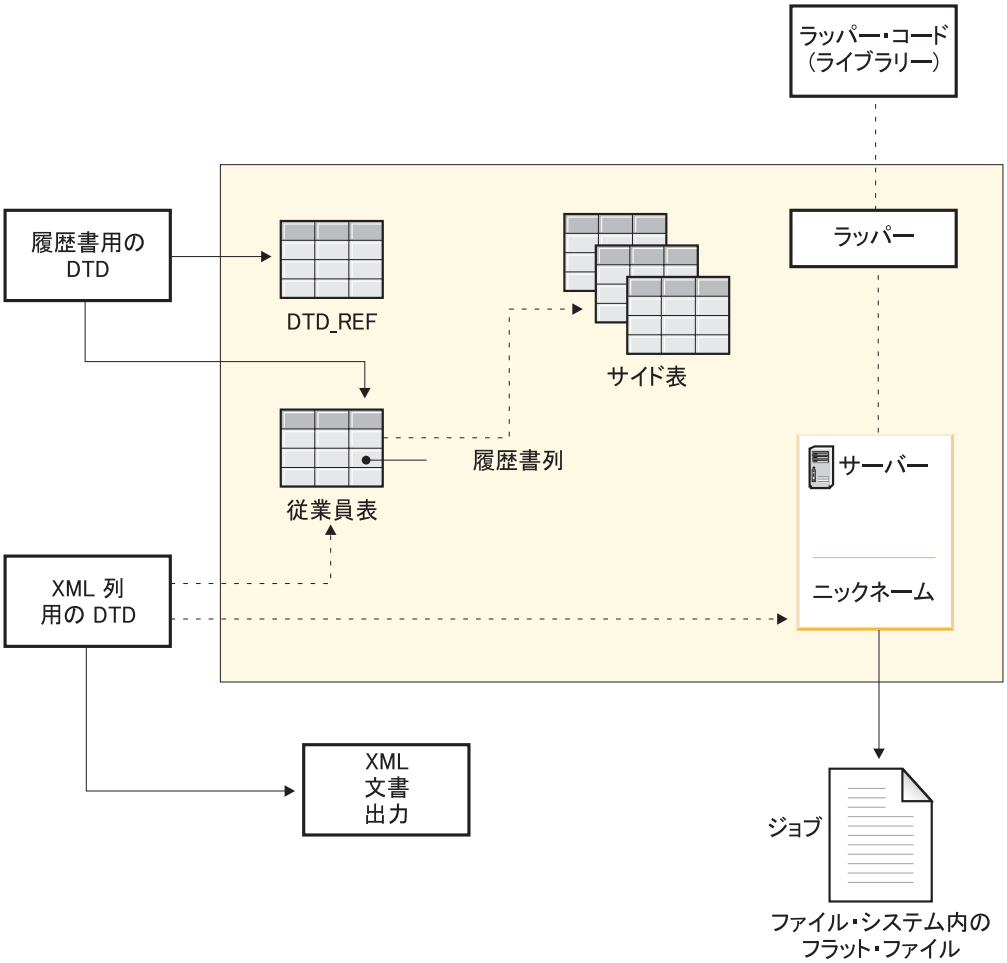


図 62. YBar, Inc 設計フロー

YBar, Inc, は DB2 UDB 表に格納されている XML データを、データの他の列と共に使用可能なリレーショナル・データとして扱います。この会社では、DB2 XML エクステンダーの機能を使用して、XML 形式の履歴書を完全な XML 文書として格納します。

1. 外部フラット・ファイルに格納されているこのデータにアクセスできるよう、データベースをフェデレーテッド・システムに対して使用可能にする必要があります。

   ```
   DB2 UPDATE DBM CFG USING FEDERATED YES
   ```

2. XML の機能を使用するには、データベースを DB2 XML エクステンダーに対して使用可能にする必要があります。

   ```
   DXXADM ENABLE_DB emp_db
   ```

3. データベース管理者は、Employee 表に「Resume」という列を追加します。XMLVARCHAR データ・タイプは DB2 XML エクステンダー・タイプの 1 つで、 XML 文書を DB2 UDB 表に含めることを可能にします。

   ```
   ALTER TABLE employee ADD COLUMN Resume XMLVARCHAR
   ```

   新しい列には、XML 文書として従業員データベースにある履歴書が含まれます。 Resume 列では、どの履歴書が特定のプロジェクトに必要とされるスキルと一致しているかを検索する XML 検索機能を使用することができます。

4. ラッパー、サーバー、およびニックネームなどのフェデレーテッド・データ・ソースを作成する必要があります。フェデレーテッド・データ・オブジェクトを作成するために、DB2 UDB コントロール・センターを使用できます。

   以下の例は AIX® に表構造のファイル・ラッパーを登録します。

   ```
   CREATE WRAPPER flat_files LIBRARY 'libdb2lsfile.a'
   ```

   使用するラッパーごとにサーバー定義を登録する必要があります。

   ```
   CREATE SERVER local_flat_files WRAPPER flat_files
   ```

   SQL ステートメントで非リレーショナル・データを使用するためにニックネームを登録します。 FILE_PATH オプションは他のデータ・ソースのフラット・ファイルに格納されている履歴書のデータへのパスを定義します。

   ```
   CREATE NICKNAME job
   (Job_ID INTEGER,
   Job_description VARCHAR(255),
   Title VARCHAR(50),
   responsibilities VARCHAR(100))
   FOR SERVER local_flat_files OPTIONS
   (FILE_PATH 'C:¥SPC¥job_table.txt',
   COLUMN_DELIMITER ',',
   KEY_COLUMN 'job_id',
   VALIDATE_DATA_FILE 'y')"
   ```

5. 以下のステートメントを使用して、ニックネームのオプションが有効か検証します。

   ```
   SELECT tabschema,
   tabname, option, setting
   FROM SYSCAT.TABOPTIONS
   ```

   SELECT ステートメントの結果は、194 ページの表 24 にあります。この表は、CREATE NICKNAME ステートメントで定義されるスキーマ名および表オプショ

ンを示します。

表 24. TABOPTIONS 表からの選択結果

TABSCHEMA	TABNAME	OPTION	SETTING
USERNAME	JOB	FILE_PATH	c:¥SPC¥job_table.txt
USERNAME	JOB	COLUMN_DELIMITER	,
USERNAME	JOB	KEY_COLUMN	JOB_ID
USERNAME	JOB	VALIDATE_DATA_FILE	Y
USERNAME	JOB	REMOTE_TABLE	JOB
USERNAME	JOB	REMOTE_SCHEMA	USERNAME
USERNAME	JOB	SERVER	LOCAL_FLAT_FILES

6. DB2 UDB 表に XML スキーマをマップするために文書型定義 (DTD) を登録し、次いで XML 列 resume を使用可能にします。

```
DXXADM enable_column Employee_db
  Employee resume resume.dad -v resume_view -r Emp_ID
```

   DB2 XML エクステンダーのコマンド DXXADM の enable_column オプションは、データベースに接続して XML 列を使用可能にし、そこに XML エクステンダー・ユーザー定義タイプを含めることができるようにします。 XML エクステンダーは、列を使用可能にするときに、以下のタスクを含むいくつかのタスクを行います。

   - DAD ファイルの中で指定されたサイド表を作成します。この表には、XML 表の行ごとにユニークな ID を含む列があります。この例では、サイド表のこの列は Emp_ID です。
   - XML 表とそのサイド表のデフォルト・ビューを作成します。この例の場合は、resume_view という名を使用します。

   以下の定義が enable_column ステートメントに関して使用されます。

   - Employee_db: データベースの名前。
   - Employee: 表の名前。
   - resume: XML 列の名前
   - resume.dad: スキーマ定義が入っている文書アクセス定義ファイルの名前。
   - resume_view: XML 列とサイド表を結合する、デフォルト・ビューの名前。
   - Emp_ID: サイド表の root_id として使用される、XML 列表内の主キーの名前。

7. DTD を使用して XML 列内または XML コレクション内の XML データを妥当性検査できます。 DTD は DTD リポジトリー表に保管できます。これは、DTD_REF という DB2 UDB 表です。 DTD_REF 表には DB2XML というスキーマ名があります。 DTD_REF 表内のそれぞれの DTD には、DTDID と命名された一意的な ID があります。 DTDID は ID、またはローカル・システム上の DTD のロケーションを指定するパスになります。 DTDID は、DAD ファイル内で <DTDID> エレメントに指定されている値と同じでなければなりません。あるデータベースを XML に対して使用可能にすると、XML エクステンダーは DTD_REF 表を作成します。コマンド行から以下のステートメントを発行して、DTD を挿入できます。

```
INSERT INTO db2xml.dtd_ref VALUES
  ('resume.dtd',db2xml.XMLClobFromFile
      ('%PATH_DEMO%¥resume.dtd'),0,
      'user1','user1','user1')
```

8. 以下のステートメントを発行することにより、DTD が正しく登録されていることを確認できます。

```
SELECT dtdid, content FROM db2xml.dtd_ref;
```

このステートメントでは、content が DTD の内容です。

9. サイド表および抽出用のユーザー定義関数を使用してデータを照会します。

```
SELECT e.firstname, e.lastname, e.resume
  FROM employee AS e,
 resume_skills_sidetable AS r
 TABLE(DB2XML.EXTRACTVARCHARS(e.resume,'/resume/skill')) AS t
  WHERE SUBSTR(t.returnedvarchar,1,8) LIKE '____Java'
```

XML 文書は resume 列に格納されています。ユーザー定義関数 extractVarchars() は、列 resume を入力として、ロケーション・パス /resume/skill を選択 ID として使用します。ユーザー定義関数は履歴書にリストされている従業員のスキルの表を戻します。

関連概念:

- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『サイド表の計画』
- 183 ページの『フェデレーテッド・システムの利点』
- 187 ページの『従業員のスキルのシナリオ - ソリューションの設計』

関連タスク:

- 「フェデレーテッド・システム・ガイド」の『フェデレーテッド・ビューを使用した異種データへのアクセス』
- 「*IBM DB2 Information Integrator データ・ソース構成ガイド*」の『フェデレーテッド・サーバーへの DB2 ファミリー・データ・ソースの追加』

コンテナー管理のパーシスタンス Bean の作成とデプロイ

コンテナー管理のパーシスタンス Entity Bean を定義する場合は、デプロイメント記述子ファイルを使用して、永続的なデータおよびすべてのアクセス制限が入ったデータ・ソースを制限します。 Entity Bean を開発した後、Bean の特性を管理するデプロイメント記述子を設定し、Bean をパッケージしてデプロイします。

フェデレーテッド・システムのニックネームにマップするコンテナー管理のパーシスタンス Entity Bean を作成できます。データベース・マネージャーへのリモート・データを表すフェデレーテッド・オブジェクトがニックネームになります。このセクションで使用されている例では、個々の Oracle 表、DB2 Universal Database 表、およびフラット・ファイルに関連付けられているニックネームにマップする、個々の Entity Bean を作成します。単一コンテナー管理のパーシスタンス Entity Bean は、複数のデータ・ソースにまたがる場合があります。 Entity Bean は、標準エンタープライズ JavaBean テクノロジーを使って、異質なデータを統合する方法を提供します。

前提条件:

WebSphere Studio バージョン 5 以降をインストールします。

Oracle の表またはフラット・ファイル用のフェデレーテッド・システム・オブジェクトを登録および作成します。

手順:

WebSphere Studio を使用してコンテナー管理のパーシスタンス Entity Bean を作成およびデプロイするには、以下のようにします。

1. WebSphere Studio の Java 2 Enterprise Edition (J2EE) パースペクティブから、作成する Entity Bean のための EJB プロジェクトを作成します。
2. コンテナー管理のパーシスタンス Entity Bean を作成します。
   - コンテナー管理のパーシスタンス Entity Bean に、データ・ソースのために定義したニックネームと同じ名前を付けます。
   - それぞれの列名に対応する属性をニックネームに追加します。ニックネームの列のデータ・タイプに合わせて、各列のデータ・タイプを指定します。 1 つの属性をキー・フィールドに指定します。これは、ニックネームの主キー列にマップする必要があります。
3. 以下のようにして、エンタープライズ JavaBean データ・モデル化ウィンドウを開きます。
   a. 作成した Entity Bean を選択し、右マウス・ボタンでクリックしてドロップダウン・メニューを開きます。**「生成」 —> 「EJB から RDB へのマッピング... (EJB to RDB mapping...)」**を選択し、**「トップダウン・モデリング (top-down modeling)」**を選択します。
   b. **「次へ」**をクリックします。データベース名およびスキーマ名が正しく設定されていることを確認してください。データベース名は、DB2 UDB クライアントに知られているフェデレーテッド・データベースにマップしなければなりません。スキーマ名は、許可されたフェデレーテッド・データベース・ユーザーにマップしなければなりません。
   c. **「DDL の生成」**をクリアします。
   d. **「終了 (Finish)」**をクリックします。
4. Entity Bean とデータベースの間のマッピングが正常に完了したか検査します。
   a. Entity Bean モジュールを選択し、右マウス・ボタンでクリックしてドロップダウン・メニューを開きます。
   b. **「開く (Open With)」 —> 「Mapping Editor」**を選択します。
   c. **「タスク」**ウィンドウのエラーをすべて訂正します。
5. Entity Bean を、フェデレーテッド・データベースのために作成したデータ・ソースにバインドします。
   a. Entity Bean を選択し、右マウス・ボタンでクリックしてドロップダウン・メニューを開きます。
   b. **「開く (Open with)」 —> 「Deployment Descriptor Editor」**を選択します。

c. 「概要 (Overview)」ページで、「WebSphere のバインディング (WebSphere Bindings)」にスクロールダウンします。「JNDI-CMP ファクトリー・バインディング (JNDI-CMP Factory Bindings)」で、有効な JNDI 名とコンテナーの許可タイプを指定します。

d. 変更を保管し、ウィンドウを閉じます。

6. Entity Bean をデプロイするためにコードを生成します。

a. Entity Bean を選択し、右マウス・ボタンでクリックしてドロップダウン・メニューを開きます。

b. **「デプロイメント・コードの生成 (Generate Deploy Code)」**を選択します。

関連概念:

- 149 ページの『Java 2 Enterprise Edition アプリケーション』
- 186 ページの『フェデレーテッド・システムでの Enterprise Bean』

フェデレーテッド・ソリューションのためのアプリケーション設計 — Cottonwood Distributors, Incorporated

Cottonwood Distributors, Incorporated (CDI) のデータベース・プログラマーは、新しく獲得したデータ・ソースすべてにアクセスする必要があるフェデレーテッド・オブジェクトを作成します。プログラマーはリモート・システムにアクセスする 3 つのデータ・ソースすべてにフェデレーテッド・オブジェクトを作成する必要があります。プログラマーは 図 63 に示されているステートメントを使用してアクセスを作成します。この例はデータ・ソースの 1 つである DB2® Universal Database に対する SQL データ定義言語 (DDL) ステートメントを示しています。

```
...
catalog tcpip node DB2_TPCH remote x.xx.xx.xx server 50000;
...
catalog database tpcd at node db2_tpch;
...
create wrapper drda;

create server db2_tpch type db2/udb version  8.1
  wrapper drda authorization "demo" password "xxxxx"
  options (dbname 'TPCD');
create user mapping
  for user SERVER db2_tpch
  OPTIONS ( REMOTE_AUTHID 'demo', REMOTE_PASSWORD 'xxxxx' );

create nickname db2_part for db2_tpch.tpcd.part;
create nickname db2_supplier for db2_tpch.tpcd.supplier;
create nickname db2_partsupp for db2_tpch.tpcd.partsupp;
create nickname db2_nation for db2_tpch.tpcd.nation;
create nickname db2_region for db2_tpch.tpcd.region;
create nickname db2_customer for db2_tpch.tpcd.customer;
create nickname db2_orders for db2_tpch.tpcd.orders;
```

図 63. DB2 データ・ソースに対するフェデレーテッド・オブジェクトの例

Cottonwood のプログラマーは、3 つの異なるデータ・ソースのメイン表 (part、partsupp、supplier) のビューを作成する必要があります。 3 つのデータ・ソ

ースに対して UNION ALL ステートメントを使用して、1 つのビューを作成します。 198 ページの図 64 の例は、ビューの 1 つを示しています。

```
CREATE  VIEW part_fed (
  p_partkey, p_mfgr, p_type, p_size, p_retailprice) AS
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM db2_part
UNION ALL
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM inf_part
UNION ALL
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM ora_part;
```

図 64. 3 つのデータ・ソースの結合

Cottonwood のデータベース・プログラマーが書くプログラムは、外注のベンダーが作成する XML ファイルにアクセスする必要があります。この XML ファイルには従業員データなどの人的資源の情報が含まれています。このファイルにアクセスするために、Cottonwood のプログラマーは、 198 ページの図 65 に示すように、XML ファイルのラッパー、サーバー、およびニックネームを作成します。

```
create wrapper XML_files library 'db2lsxml.dll';
create server LOCAL_XML_FILES wrapper XML_FILES;
create nickname Employees_From_XML (
  doc Varchar(100) OPTIONS(DOCUMENT 'FILE'),
  Employee_Number Varchar(5) OPTIONS(XPATH './@SerialNum'),
  First_Name Varchar(50) OPTIONS(XPATH './/Firstname'),
  Middle_Initial Varchar(50) OPTIONS(XPATH './/Initial'),
  Last_Name Varchar(50) OPTIONS(XPATH './/Lastname'),
  Department_Number Varchar(50) OPTIONS(XPATH './/Department'),
  Phone_Number Varchar(50) OPTIONS(XPATH './/PhoneNumber'),
  Job Varchar(50) OPTIONS(XPATH './/Job'),
  Education_Level Varchar(50) OPTIONS(XPATH './/EDLevel'),
  Gender Varchar(50) OPTIONS(XPATH './/Sex'),
  Hire_Date Varchar(50) OPTIONS(XPATH './/HireDate'),
  Birth_Date Varchar(50) OPTIONS(XPATH './/BirthDate'),
  Annual_Salary Varchar(50) OPTIONS(XPATH './/Salary'),
  Annual_Bonus Varchar(50) OPTIONS(XPATH './/Bonus'),
  Commission Varchar(50) OPTIONS(XPATH './/Comm'),
  cid Varchar(16) OPTIONS(PRIMARY_KEY 'YES'))
 FOR SERVER LOCAL_XML_FILES
 OPTIONS (XPATH '//Employee');
```

図 65. フェデレーテッド・システムのオブジェクトの作成

Web アプリケーションはカスタマー・オーダーと提供業者の価格更新をキューにルーティングするので、プログラマーは表を読む関数とメッセージ・キューのビュー (199 ページの図 66 を参照) を作成します。

```
CREATE FUNCTION NEW_SUPPLIERS_READ()
    RETURNS TABLE  ( SUPPLIER_NAME VARCHAR(80),
                           SUPPLIER_PHONE VARCHAR(12),
                           PART_KEY DOUBLE,
                           PART_PRICE DOUBLE,
                           MAN_DAYS DOUBLE,
                           MAX_QUANTITY DOUBLE,
                       CORRELID VARCHAR(80))
    LANGUAGE SQL
    NOT DETERMINISTIC
    EXTERNAL ACTION
    READS SQL DATA
    RETURN
 SELECT
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',1),80),
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',2),12),
         DOUBLE(DB2MQ.GETCOL(T.MSG,',',3)),
         DEC(DB2MQ.GETCOL(T.MSG,',',4),8,4),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',5)),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',6)),
  CORRELID
 FROM TABLE (DB2MQ.MQREADALL('DB2.DEFAULT.SERVICE',
       'DB2.DEFAULT.POLICY')) AS T;

CREATE VIEW READ_NEW_SUPPLIERS_FROM_QUEUE
  AS SELECT * FROM TABLE(NEW_SUPPLIERS_READ()) t
  WHERE CORRELID = 'CDI_NEW_SUPPLIER';
```

図 66. Cottonwood の表を読む関数

最後に、Cottonwood のプログラマーは処理に使用される一時表を作成する必要があります。

- XML の構成の実行用:

  ```
  create table Ucustomers (x_doc DB2XML.XMLCLOB not logged);
  ```

- Web 要求の格納用:

  ```
  create table request_bid
    (reqkey integer not null,
     partkey integer not null,
     bid double not null);

  create table request_status
    (reqkey integer not null,
     partkey integer not null,
     suppkey integer not null,
     newquote double not null,
     currentquote double not null,
     status varchar(15) not null);
  ```

関連タスク:

- 200 ページの『フェデレーテッド・ソリューションのためのアプリケーション開発 — Cottonwood Distributors, Inc.』

関連資料:

- 243 ページの『付録 A. Cottonwood Distributors, Inc. および YBar, Inc. シナリオのスクリプト例』

フェデレーテッド・ソリューションのためのアプリケーション開発 — Cottonwood Distributors, Inc.

Cottonwood Distributors, Incorporated (CDI) は、サーブレットによって Web サービスを呼び出すことで、あらゆる新規見積価格あるいは価格提示に関する XML メッセージを生成できます。ここでは、類似の環境を作るためのステップを示します。

手順:

Cottonwood Web サービスを作成するためのアプリケーションを開発するには、以下のようにします。

1. DB2 Universal Database サーバーをフェデレーテッド・データベースとして機能するように構成し、SVCENAME サーバーと FEDERATED オプションのプロパティーを以下のように設定します。

   ```
   db2 update dbm cfg using SVCENAME myID authentication server
   db2 update dbm cfg using FEDERATED yes
   db2 connect reset
   db2stop
   db2start
   ```

2. 以下のようにしてローカルの DB2 Universal Database に接続します。

   ```
   db2 connect to myLocalDB user user1 using myPW
   ```

3. それぞれのデータ・ソースごとにフェデレーテッド・オブジェクトを作成します。
4. DB2 UDB サーバーがあるリモート・ノードを識別するよう、 DB2 Universal Database クライアントを構成します。
5. アクセスするリモート・データのニックネームを作成します。

   ```
   create nickname ora_part for oraserver.CDI.part;
   ```

6. WebSphere Application Server を構成します。
   - Web サービス・プロジェクトに関連する Java ビルド・パスに、 *db2java.zip* または *jcc.jar* ファイルの場所を組み込みます。 `group.properties` ファイルの `dbDriver` パラメーターは、どのデータベースのドライバー・パッケージを使用するかを決定します。
   - WebSphere 環境に、フェデレーテッド・サポート対象として構成した DB2 Universal Database にマップするデータ・ソース・オブジェクトを作成します。データ・ソース・オブジェクトはプールされたデータベース接続を表します。このオブジェクトは、JNDI サービスを使用してデータ・ソースを検索し、そのデータ・ソースに関連するメソッドを呼び出すように設定できます。

これで、作成するすべての接続において、フェデレーテッド・オブジェクトのデータを選択、挿入、更新、または削除できるようになりました。 ora_part というニックネームを作成したので、作成する SQL ステートメントは ora_part ニックネームだけを参照します。 Cottonwood の例のステートメントには、複数のニックネームを参照する単一の SQL ステートメントを使って複数のソースから結合されたデータが含まれています。

201 ページの図 67 に示すコードは、Web サービスを使用してキューにメッセージを書き込みます。

```
public void doGet(HttpServletRequest req,
 HttpServletResponse res)
  throws javax.servlet.ServletException, java.io.IOException {
  try {
   PrintWriter pr = res.getWriter();
   pr.print(htmlHeader1);
   pr.print(message1);
   pr.print(message2);
   String method = "";

...
```

図 67. 新規の見積価格または価格提示によってメッセージがキューに書き込まれる (MessageFormatter.java) (1/5)

```
public class MessageFormatter
  extends javax.servlet.http.HttpServlet
  {

final static String htmlHeader1 = "<HTML><TITLE>MessageFormatter";
  final static String message1 =
 "<p align=¥"center¥"><font color=¥"navy¥" face=¥"verdana¥"
 size=¥"+2¥"><b>Thank You<p><font color=¥"black¥"
 face=¥"veranda¥" size=¥"+1¥">";
 final static String message2 = "";

 final static String htmlHeader2 = "";
 static String quote = "";

 static Connection con = null;
 final static String url = "jdbc:db2:demo";

...
```

図 67. 新規の見積価格または価格提示によってメッセージがキューに書き込まれる (MessageFormatter.java) (2/5)

```
WSProxy WSid = new WSProxy();
   boolean fCustomer = true;
if (req.getParameter("method")!=null) {
    //Get the common parms to the servlet
    //from the REQ object
    key = req.getParameter("name");
    part = req.getParameter("part");
    method = req.getParameter("method");

    // If customer order
    if (method.equals("orderNewParts")) {
     fCustomer = true;
     // Get the quantity the customer is ordering
     quantity = req.getParameter("quantity");

// set the table and column names for SELECT
     col_name1 = "c_name";
     col_name2 = "c_custkey";
     tab_name  = "db2_customer";

    ...
```

図 67. 新規の見積価格または価格提示によってメッセージがキューに書き込まれる (MessageFormatter.java) (3/5)

```
   // Get the real customer name from federated data source
    try {
     Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
     url = "jdbc:db2:demo";
     con = DriverManager.getConnection(url,"demo","xxxxx");
     stmt = con.createStatement();
     rs = stmt.executeQuery
        ("SELECT " + col_name1 + " from " + tab_name +
          " where " + col_name2 + " = " + key);
     while (rs.next()) {
      cust_name = rs.getString(1);
     }
...
```

図 67. 新規の見積価格または価格提示によってメッセージがキューに書き込まれる (MessageFormatter.java) (4/5)

```
   // Write request status back to user
   try{
    // If this is a supplier update request
    if (!fCustomer) {
     String message1id=  "2," + key + "," + part + ","
          + quantity + "," + price ;
     java.lang.String message1idTemp  = message1id;
     System.out.println
           ("message type 2 being written: " + message1idTemp);
     org.tempuri.worftestweb.
        demo.newdadx.dadx.xsd.Stmt1ResultElement mtemp =
     WSid.stmt1(message1idTemp);
```

図 67. 新規の見積価格または価格提示によってメッセージがキューに書き込まれる (MessageFormatter.java) (5/5)

CustomerRoutine はメッセージを解析します。ルーチンはメッセージ・タイプ (見積価格または価格提示) に基づいてアクションを起こします。ルーチンは解析した情報を REQUEST_BID または REQUEST_STATUS 表に書き込みます。メッセージのタイプがパーツの見積価格である場合、新しい見積価格を既存の見積価格と比較します。これが新しいパーツの見積価格である場合、状況は新規に変化します。ローカルな状況表に状況が NEW として追加されます。価格が下がった既存のパーツの見積価格の場合、プログラムはその価格を受け入れます。プログラムはローカル表に書き込みを行い、状況を ACCEPTED に更新します。価格が上がった既存のパーツの見積価格の場合、プログラムは状況表に REVIEW と書き込みを行います。

関連概念:

- 211 ページの『アプリケーションのデプロイ — Cottonwood Distributors, Inc. のソリューション』
- 207 ページの『データの発見 — Cottonwood Distributors, Inc.』

フェデレーテッド・アプリケーションのデプロイ

フェデレーテッド・アプリケーションをデプロイするために、Cottonwood Distributors, Incorporated のデータベース・プログラマーは、 Web サービス・アクション用の *newdadx.dad* を呼び出す必要があります。ここでは、アプリケーションをデプロイするのに役に立つ例を示します。

前提条件:

WebSphere Application Server 5.0 でデプロイ可能な最小単位は Web アーカイブ (WAR) ファイルです。このアプリケーションが Enterprise JavaBeans (EJB) を開発する場合は、 Java アーカイブ・ファイル (JAR) とエンタープライズ・アーカイブ・ファイル (EAR) が必要です。

アプリケーションをデプロイする前に、エンタープライズ・フェデレーテッド・アプリケーション用の以下のコンポーネントを構築する必要があります。

WAR ファイル
: Web 関連のコンポーネント (HTML、JavaScript、JSP)

JAR ファイル
: ビジネス・ロジック・コンポーネントを作成する Java クラス

EAR ファイル
: エンタープライズ・ソリューションを作成する JAR ファイルおよび WAR ファイル

手順:

フェデレーテッド・アプリケーションをデプロイするには、以下のようにします。

1. EAR ファイルを WebSphere Application Development 環境にインポートします。Cottonwood Distributors, Incorporated の場合、EAR ファイル (CDI.ear) には以下のファイルが入っています。
   - .project
   - META-INF/.modulemaps

- META-INF/application.xml
- META-INF/ibm-application-ext.xml
- META-INF/MANIFEST.MF
- WorfTestWeb.war

2. Web サービス (Document Access Definition Extension (DADX) ファイル) をデプロイします。そのためには、WebSphere Studio ウィンドウで各 DADX ファイルを選択して、**「Web サービスとしてデプロイする (Deploy as Web service)」**プッシュボタンをクリックします。

図 68. DADX ファイルの選択

3. WebSphere Application Development サーバーを再起動します。

デプロイしたアプリケーションは、顧客の価格提示要求と、 Cottonwood の新しい合併企業に必要な価格提示状況を示します。顧客の価格提示要求は Web サービスを介して送信されます。要求の実体は、WebSphere メッセージ・キューへ経路指定されるメッセージです。 Cottonwood では、このメッセージ・キュー上に、アクティビティーを listen するアプリケーションを設けています。こうして、データベース要求がキューに入ると同時に、プログラムはキューから要求を検索できます。このリスナー・アプリケーションは、要求を処理するためにアクションのセットを呼び出します。 Cottonwood が使用するリスナーは、多数の価格提示要求がキューに入ることで生じる負荷のバランスを取り、以下のステップに従って顧客の価格提示要求を処理します。

1. 顧客の価格提示要求によってメッセージがキューに書き込まれる。
2. リスナー・アプリケーションが DB2 Universal Database 機能を呼び出して、要求から注文情報を抽出する。
3. Cottonwood のデータベース・プログラマーが読み取り専用の照会を実行して、要求されたパーツの見積価格を入手する。
4. Cottonwood のデータベース・プログラマーが、レコードをローカル表に挿入して注文を記録する。

Cottonwood は、顧客の価格提示要求の処理と似た一連のステップで、提供業者の見積価格の更新を処理します。しかし、提供業者の見積価格の照会は、プログラムが見積価格を更新する必要があるので、読み取り専用ではありません。リスナー・アプリケーションのおかげで、Cottonwood は更新のコミットを許可する前に要求を検査できます。以下に、提供業者の見積価格要求のための具体的なステップを示します。

1. 提供業者の見積価格要求によってメッセージがキューに書き込まれる。
2. メッセージの形式をもとに、リスナー・アプリケーションが DB2 Universal Database 機能を呼び出して、パーツの見積価格の更新要求を抽出する。
3. Cottonwood のデータベース・プログラマーは、その提供業者がそのパーツに付けた過去最低の見積価格を検討する。新しい見積価格の方が安い場合、あるいはこれがその提供業者が以前に提供したことのないパーツの見積価格である場合、アプリケーションはデータベースを更新する。新しい見積価格の方が高い場合、Cottonwood は更新を加えない場合がある。代わりに Cottonwood アプリケーションはその見積価格に対し、後で Cottonwood のマーケティング・チームが検討すべき対象として、あるいは提供業者と交渉すべき対象としてマークを付ける。

顧客の注文と提供業者の価格更新の流れの詳細は、図 69 および 206 ページの図 70 で説明されています。

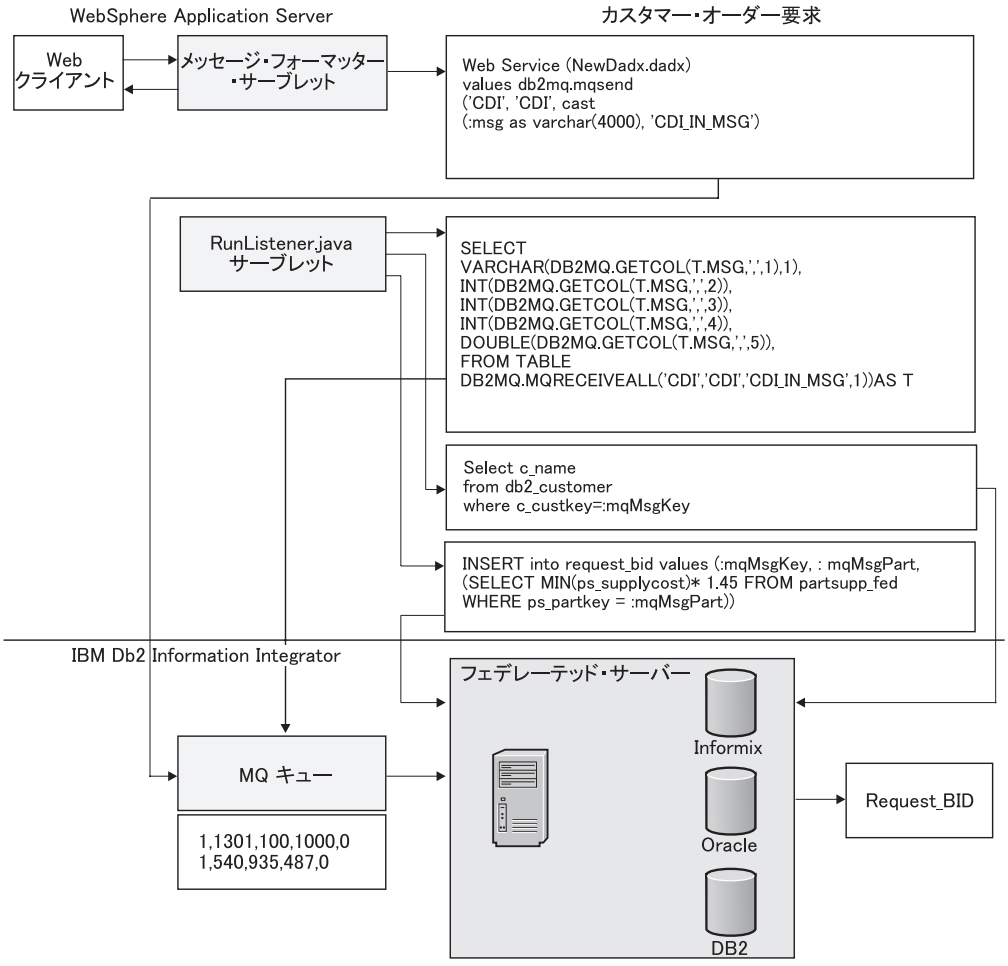


図 69. Cottonwood の顧客注文要求

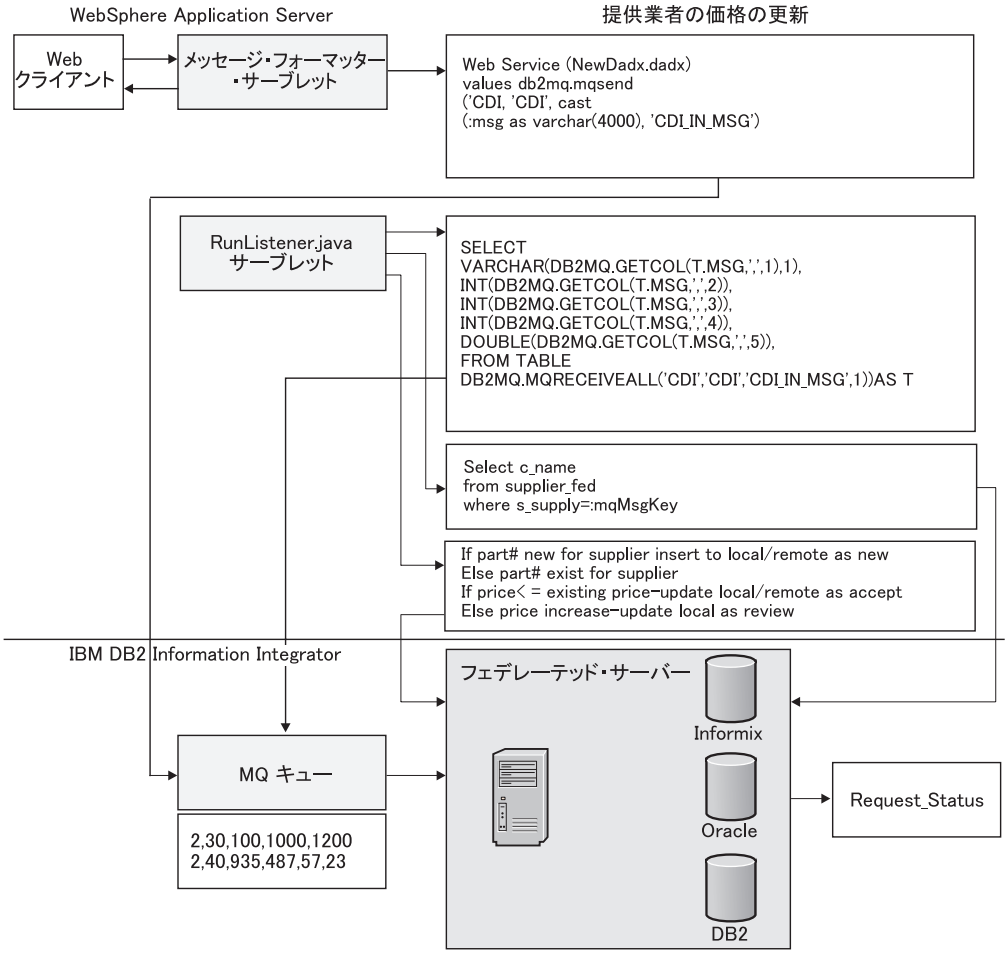


図 70. 提供業者の価格は更新可能

関連タスク:

- 155 ページの『Web アーカイブ・ファイルの準備と作成』

データウェアハウスの拡張

このセクションでは、インフォメーション・インテグレーション・ソリューションの一部として DB2 Warehouse Manager とデータウェアハウス・センターを使用する方法の具体例を検討します。

ビジネス・ソリューション: DB2 Warehouse Manager の拡張

DB2® Warehouse Manager では、DB2 と IBM® DB2 Information Integrator ユーティリティーの機能の統合が改善されています。 SQL UPDATE ステートメントを使用して、ソースのデータでターゲット・データを更新できます。加えて、複数のステップが完了するまで待機するプロセスを定義できます。ウェアハウス・トランスフォーマーにはストアード・プロシージャーとユーザー定義関数が組み込まれており、データウェアハウスを構築するために一般的に使用されるトランスフォーメーションを提供します。

データウェアハウス・センターは、メタデータ主導型のシステムです。メタデータ (すなわち、データに関連する情報) は、管理者およびビジネス・ユーザーに、データウェアハウスに格納されているデータの説明を提供します。ビジネスのメタデータをビジネス用語で記述するインフォメーション・カタログを作成できます。メタデータをサブジェクト・エリアに編成し、さらにワークグループや企業の要件に合わせてカスタマイズすることができます。

Clean Data トランスフォーマーなどのテクノロジーを使用することにより、ソース・データに対して基本的な修正、置換、およびマッピング操作を実行できます。アプリケーションをデータウェアハウス・センターに定義することにより、 1 つ以上のステップがプログラムを使用して処理を行えるようにもできます。ユーザー定義プログラムをデータウェアハウス・センターに定義すると、「プロセス・モデル」ウィンドウの 1 ステップとしてプログラム定義が使用可能になります。ウェアハウス・ソースは、データをウェアハウスに提供する表とファイルを識別します。データウェアハウス・センターは、ウェアハウス・ソースの指定を使用してデータにアクセスします。ほとんどすべてのリレーショナル・ソースまたは非リレーショナル・ソース (表、ビュー、ファイル、または定義済みのニックネーム) をソースにすることができます。

ウェアハウス・ステップのデータの関連とオブジェクト定義をグラフィカルに表現するには、インフォメーション・カタログ・マネージャーを使用します。インフォメーション・カタログ・マネージャーにより、ユーザーが企業のデータを見付け、理解し、アクセスするのに役立つ、強力なビジネス指向のソリューションが提供されます。インフォメーション・カタログ・マネージャーにより、ビジネス・ユーザーは、データの集約、履歴、データ派生、データ・ソース、および記述を表示することができます。

関連概念:

- 「*データウェアハウス・センター 管理ガイド*」の『データウェアハウス・センターの構成』
- 14 ページの『Cottonwood Distributors, Inc. — ウェアハウスの例』

データの発見 — Cottonwood Distributors, Inc.

Cottonwood Distributors, Incorporated は、吸収企業と自社の今後について判断を下し始め、気がかりな特定の領域を見定めます。

- Cottonwood のプログラマーが処理すべきデータの量は、合併前の処理量より増えました。データが置かれている物理的な位置はさまざまであり、データベースも形式も異なります。
- Cottonwood は競争力を保つ必要があります。会社は電話中心の営業担当員を Web ベースの仲介取引のシステムに置き換える決定をします。 Cottonwood の Web ベース仲介取引システムのユーザーは、パーツのオンライン価格提示要求をしたり、パーツの注文をオンラインで行うことになります。 Cottonwood には一連の提供業者がおり、それらの業者はパーツの新規あるいは更新された見積価格を発信する必要があります。

会社が新たに拡大したことを受けて、Cottonwood Distributors, Incorporated は必要なデータを見極める必要があります。また、データの使用目的も判断しなければなりません。それから、データ・ソースを置く場所を決める必要があります。

Cottonwood Distributors, Inc. が直面している問題を考慮するなら、 IBM® DB2® Information Integrator のフェデレーテッド・システムとデータウェアハウジングは、この企業にとって重要なテクノロジーとなります。フェデレーテッド・インフラストラクチャーにより、クライアント・アプリケーションは、多様な一連のソースにあるデータを単一のデータベースにあるかのように見ることができます。アプリケーション・プログラマーは複数のアプリケーション・プログラミング・インターフェース (API) を扱うコードを書く必要はなく、単一の SQL ステートメントで Cottonwood の 3 つのデータ・ソースすべてからのデータを結合できます。ウェアハウジングにより、Cottonwood のサーバーの近くにデータ・キャッシュが持てます。このキャッシュにより、頻繁に参照されるデータをユーザーの近くに置いておくことができます。

Cottonwood のデータ管理チームは、自社のデータにアクセスする方法と、自社が持つ情報を交換する方法を決定します。ある決定をまさに下そうというときに、決定者 (Cottonwood のような) がその決定に関連するデータに遭遇するときこそ、その会社がデータから知識を引き出せるときです。 Cottonwood の発見の手掛かりは、特定のデータが存在していることを知っているということです。 Cottonwood は、データを収集できること、およびデータが意思決定に関係することも知っています。

Cottonwood は、一連の提供業者が買収企業それぞれに提示するパーツの見積価格が異なるかどうかを知りたいと思っています。 Cottonwood のデータベース・プログラマーは、供給業者のデータが含まれているデータ・ソースがどこにあるかを知る必要があります。それらのデータ・ソースに、どこから、どのように接続するかも知る必要があります。フェデレーテッド・システムを利用しないでこの作業を行うには、3 つの SQL ステートメントを使って 3 つのデータベースに接続しなければなりません。さらに、顧客と提供業者、および顧客が Web ベースの仲介取引システムに対して行う要求の情報と関係を判別することが必要となります。注文をするに至ったことのない顧客に関する知識は、最適化手法またはインターフェースの改善について判断する上で有用な情報です。

関連概念:

- 206 ページの『ビジネス・ソリューション: DB2 Warehouse Manager の拡張』
- 197 ページの『フェデレーテッド・ソリューションのためのアプリケーション設計 — Cottonwood Distributors, Incorporated』
- 209 ページの『アプリケーションの設計 — Cottonwood Distributors, Inc のウェアハウス・シナリオ』
- 14 ページの『Cottonwood Distributors, Inc. — ウェアハウスの例』
- 211 ページの『アプリケーションのデプロイ — Cottonwood Distributors, Inc. のソリューション』

関連タスク:

- 200 ページの『フェデレーテッド・ソリューションのためのアプリケーション開発 — Cottonwood Distributors, Inc.』

関連資料:

- 243 ページの『付録 A. Cottonwood Distributors, Inc. および YBar, Inc. シナリオのスクリプト例』

アプリケーションの設計 — Cottonwood Distributors, Inc のウェアハウス・シナリオ

Cottonwood Distributors, Incorporated は、フェデレーテッド・システムとウェアハウジングのテクノロジーを結合して、データのスケーラビリティー、データのアクセス可能性、およびデータの現行性の問題を解決します。

Cottonwood には 1 つのデータベースがあり、これには PART という表、SUPPLIER という表、および PARTSUPP という表が入っています。

表 25. Cottonwood の SQL 1 次表と列

PART	SUPPLIER	PARTSUPP
p_partkey (ユニークな値)	s_suppkey	ps_partkey
p_name	s_name	ps_suppkey
p_mfgr	s_address	ps_availqty
p_brand	s_nationkey	ps_supplycost
p_type	s_phone	ps_comment
p_size	s_acctbal	
p_container	s_comment	
p_retailprice		
p_comment		

Cottonwood には、データ管理の問題の解決策を作る助けとして、1 次表とのリレーションシップを形成する他の表もあります。

表 26. CDI の 2 次表と列

NATION	REGION	CUSTOMER	ORDERS
n_nationkey	r_regionkey	c_custkey	o_orderkey
n_name	r_name	c_name	o_custkey
n_regionkey	r_comment	c_address	o_orderstatus
n_comment		c_nationkey	o_totalprice
		c_phone	o_orderdate
		c_acctbal	o_orderpriority
		c_mktsegment	o_clerk
		c_comment	o_shippriority
			o_comment

PART 表には、データベースに入っている各パーツの識別キー、製造メーカー、タイプ、サイズ、および小売価格に関する情報が入っています。パーツの ID は業界を通じてユニークなものです。パーツ・ナンバー 1234 は常に widgetA を示します。 SUPPLIER 表は提供業者の名前と住所についての情報を保管します。これには

識別キーが含まれています。 PARTSUPP 表はパーツと提供業者を関連付けます。これにはパーツ表と提供業者表の両方の ID、および提供業者がパーツに課す価格が入ります。

表には Cottonwood の在庫管理を容易にするため、データに索引が付けられています。 Cottonwood のデータベース・プログラマーは、3 つの基本表 (209 ページの表 25 参照) の共用体から構成されるビューを使用します。データが DB2®Universal Database、Informix®、および Oracle に保管されているにもかかわらず、Cottonwood のデータベース・プログラマーは、Cottonwood の提供業者と、それらの提供業者が合併会社の各々に提示するパーツの見積価格に関する最良の情報を得るために、単一の SQL ステートメントを使用できます。

プログラマーには、データベースに入っているデータに加えて、有用なレポート・システムを保守する必要も依然としてあります。これには XML コンテンツが含まれています。

Cottonwood の構成には、DB2 Universal Database™ サーバーと DB2 Universal Database クライアント、メッセージ・キュー・クライアント、および Web クライアントが含まれています。 Cottonwood は DB2 サーバー上に、データウェアハウス、フェデレーテッド・システム、ストアード・プロシージャー、XML、およびメッセージ・キュー統合を置いています。 Web アプリケーション・サーバーには、見積もりと価格提示を行うアプリケーション、いくつかの Web サービス、および Java™ Server Page (JSP) があります。 WebSphere® MQ もリスナーと一緒にサーバーに置かれています。 Cottonwood のプログラマーは、リモート DB2 サーバー、Oracle、および Informix サーバー上のデータにアクセスする必要があります。

Cottonwood は、スケーラビリティーとシステム負荷の問題に取り組むために、フェデレーテッド・システムとデータウェアハウスを組み合わせることができます。ウェアハウスには、すべてのデータ管理システムのデータが入っています。
Cottonwood がデータをさまざまなデータベースに保管しているということは、アプリケーションからは透過的です。ウェアハウスは、綿密に調整されたデータ管理システムを、エンド・ユーザー・アナリストが生成するトラフィックから保護するバッファーともなります。たとえば PART 表は、Informix、Oracle、および DB2 Universal Database PART 表の共用体です。さらに、表への参照は、基礎となるデータベースに表がある場合と同じです。以前は DB2 Universal Database でしか実行できなかったアプリケーションを、ほとんど変更することなくデータウェアハウスに移行できます。

ウェアハウスのシナリオ (211 ページの図 71 参照) の設計には、ウェアハウス環境 (ウェアハウス・コントロール・データベースを含む) の設定が関係しています。
Cottonwood のシナリオでは、データウェアハウス・センターにより作成される、デフォルトのデータウェアハウス・セキュリティー・グループを使用します。
Cottonwood のデータベース・プログラマーは、事前定義されたスケジュールに従ってデータ・ストアからデータを抽出します。次にプログラマーは、特定のビジネス規則を適用してデータを修正します。次に、データがデータウェアハウスにロードされます。 Cottonwood はデータウェアハウスを分割して、提供業者のデータ・ストア、注文のデータ・ストア、およびパーツのデータ・ストアにします。
Cottonwood のデータベース・プログラマーは、定義されたデータ集合をウェアハウスからデータマートに抽出します。データマートは、特に分析、報告、および測定

をサポートするために収集されたデータへのアクセスを可能にします。ユーザーは、これらのデータマートからデータ・マイニング機能を実行して、データの傾向、関係、およびパターンを見つけることができます。

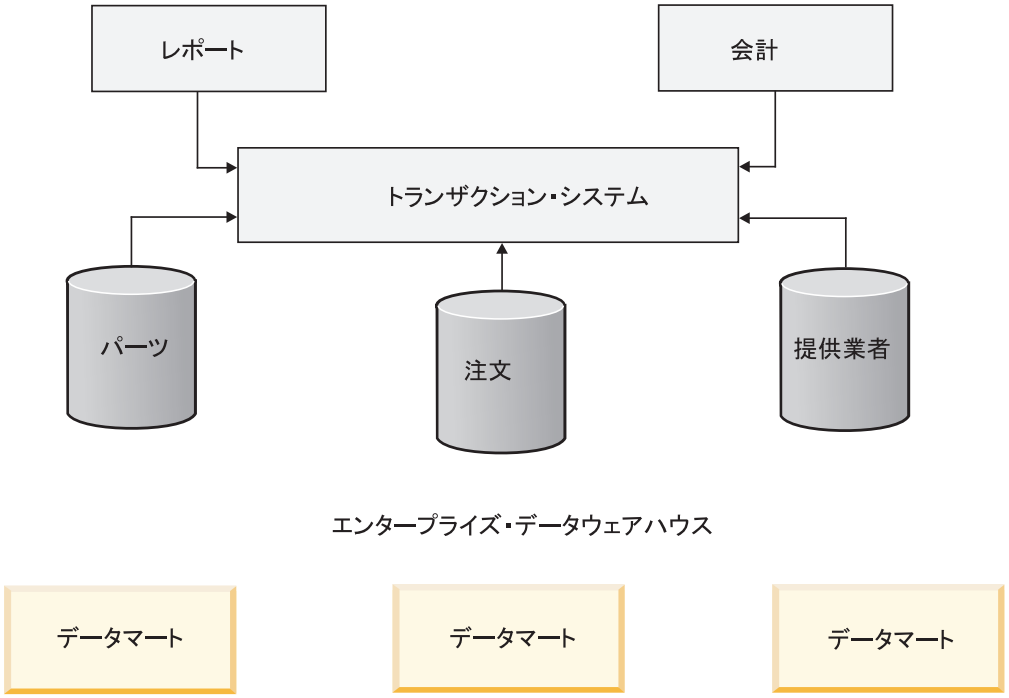


図 71. Cottonwood のウェアハウス構成

関連概念:

- 197 ページの『フェデレーテッド・ソリューションのためのアプリケーション設計 — Cottonwood Distributors, Incorporated』
- 211 ページの『アプリケーションのデプロイ — Cottonwood Distributors, Inc. のソリューション』

アプリケーションのデプロイ — Cottonwood Distributors, Inc. のソリューション

表を定義し、データを挿入したら、Cottonwood Distributors, Incorporated のデータベース・プログラマーは利用可能なパーツと現在の価格に関する情報を提供できるようになります。 Cottonwood のデータベース・プログラマーは統合システムをモニターおよび管理する助けとして DB2® Warehouse Manager を使用できます。 DB2 Warehouse Manager により、これらのプログラマーは、運用フェデレーテッド・システム・データ・ソースにアクセスして、ウェアハウス・データベースに書き込みを行うトランスフォーメーション・プロセスを定義し、実行することができます。 Cottonwood のデータベース・システムは、DB2 Warehouse Manager の外部トリガー機能を使用することにより、DB2 Warehouse Manager とインターフェースを取ることができます。

DB2 データウェアハウス・センターのグラフィカル・インターフェースは、Cottonwood のデータベース・プログラマーが、フェデレーテッド・システムの運用

データからウェアハウスへデータを移動するのを管理するのに役立ちます。 Cottonwood のエンド・ユーザーはこのウェアハウスを使用できます。ウェアハウス・センターのウィンドウを使用することにより、 Cottonwood のデータベース・プログラマーは複数のウェアハウス・プロセスを管理できます。 Cottonwood のプログラマーはこれらのプロセスをスケジュールに入れて、指定したスケジュールに従って実行することができます。

Cottonwood のプログラマーは、以下のローカル表を更新するためにウェアハウス・プロセスを実行できるようになりました。

- PART 表
- SUPPLIER 表
- PARTSUPP 表

Cottonwood Distributors のソリューションのために、Cottonwood のプログラマーは、フェデレーテッド索引、つまり索引の指定を作成することにより、アクセス・パフォーマンスを向上できます。フェデレーテッド・データベースに索引の指定を作成することについての詳細は、「フェデレーテッド・システム・ガイド」を参照してください。

Cottonwood データベース・プログラマーは、自分たちが作成するアプリケーションを使用して、フェデレーテッドおよびローカル・ウェアハウスの両方にアクセスできます。たとえば、Cottonwood のデータベース・プログラマーは、カスタマー・オーダーに対する現在の最良の価格を知ることができるレポートを作成したいと思います。現在、ウェアハウスには、すべてのカスタマー・オーダーが入った表が含まれています。 Cottonwood のプログラマーは、得意客に最良の価格を提供したいと思っています。また、提供業者による価格の変更も把握したいと思っています。 Cottonwood のデータベース・プログラマーは以下のアクションを実行します。

1. レポートを作成する。このレポートでは、ウェアハウスのカスタマー・オーダー表からデータを選択します。
2. フェデレーテッド・ソースから価格を選択し、最後にウェアハウスを更新してから価格が下がっていないか比較する。
3. 価格が下がっている場合、Cottonwood のプログラマーは、得意客を選んで価格が下がったことを伝える。

関連概念:

- 197 ページの『フェデレーテッド・ソリューションのためのアプリケーション設計 — Cottonwood Distributors, Incorporated』
- 209 ページの『アプリケーションの設計 — Cottonwood Distributors, Inc のウェアハウス・シナリオ』
- 「*DB2 Information Integrator ソリューション・ガイド*」の『フェデレーテッド開発のシナリオ —Cottonwood Distributors, Inc.』
- 207 ページの『データの発見 — Cottonwood Distributors, Inc.』
- 25 ページの『パフォーマンスと調整の計画 — フェデレーテッド・システム中のマテリアライズ照会表』

関連タスク:

- 200 ページの『フェデレーテッド・ソリューションのためのアプリケーション開発 — Cottonwood Distributors, Inc.』

WebSphere メッセージ・キュー関数を使用するデータベース・アプリケーションの開発

IBM は、Web サービスが単に統合アプリケーションだけではなく、統合データに対してもアーキテクチャーを提供することを認識しています。それで、DB2 にはデータの管理や、データへインテリジェントで最適化されたアクセスを提供する機能があります。

DB2 WebSphere MQ 機能のインストール

DB2 WebSphere MQ 機能は、DB2 Universal Database 内でユーザー定義関数として使用可能です。この機能は、ユーザーが DB2 UDB オブジェクトから WebSphere MQ キューへアクセスできるようにします。

前提条件:

1. DB2 UDB のインストール内容に以下のライブラリーが追加されていることを確認します。
   - UNIX プラットフォーム: sqllib/lib ディレクトリー内の *libdb2qgmq* および *libdb2mqsw*
   - Windows プラットフォーム: sqllib¥bin ディレクトリー内の *db2qgmq.dll* および *db2mqsw.dll*
2. DB2 Universal Database バージョン 8 以上に付属の Application Messaging Interface (AMI) インストール・イメージを使用して、 AMI バージョン 1.2.4 以上をインストールします。
3. DB2 UDB によって使用される AMT_DATA_PATH 環境変数をリストに追加して、メッセージ・キューイング・ユーザー定義関数 (MQ UDF) が正しく実行していることを確認します。ファイル *$INSTHOME/sqllib/profile.env* (UNIX) または *%DB2PATH%¥profile.env* (Windows) を編集して、 AMT_DATA_PATH を DB2ENVLIST に追加することができます。さらに、db2set コマンドを使用することもできます。

   ```
   db2set DB2ENVLIST="AMT_DATA_PATH"
   ```

   環境変数の変更を有効にするため、データベース・インスタンスを再始動します。
4. DB2 Universal Database バージョン 8 でオプションとして使用できる、 DB2 Universal Database XML エクステンダーをインストールします。
5. パブリッシュおよびサブスクライブ機能を使用する場合、`amtsamp.tst` ファイルを使用して、システム・デフォルト AMI オブジェクトを作成します。

DB2 WebSphere MQ 機能を構成および使用可能にする基本ステップは以下のとおりです。

1. WebSphere MQ バージョン 5.3、Corrective Service Diskette 05 (CSD05) 以降のリリースをインストールします。
2. http://www.ibm.com/software/ts/mqseries/ にある「Support」ページから、以下の SupportPacs をインストールします。
   - WebSphere MQ Application Messaging Interface

- パブリッシュまたはサブスクライブ機能が組み込まれている WebSphere MQ SupportPacs

3. トランザクション MQ UDF を使用する場合、データベースをフェデレーテッド操作用に構成していることを確認します。これは以下のコマンドを使用して行います。

   ```
   update dbm cfg using federated yes
   ```

4. DB2 UDB MQ 機能を使用可能にします (**『手順』**にあるステップ 1 (215 ページ) を参照してください)。

制約事項:

- スキーマ **db2mq1c** の下にある DB2 UDB MQ トランザクション機能は、CLOB タイプのメッセージをサポートしていません。
- トランザクション MQ ユーザー定義関数の使用可能化ユーティリティーによって、 40 個の AMI ポリシーのみが、-q オプションを指定したキュー・マネージャーの AMI リポジトリー・ファイルに存在できるようになります。
- トランザクション MQ ユーザー定義関数が単一トランザクション内でサポートできるのは、 1 つのキュー・マネージャーのみです。サービスおよびポリシー (amthost.xml 内の接続を通して) で指定するキュー・マネージャーは一致していなければなりません。サービス・ポイントでキュー・マネージャーをブランクにしておく場合、 WebSphere MQ のデフォルトはポリシーで指定されたマネージャーになります。 MQ インストール中および enable_MQFunctions プロセス中に通常作成される、 MQ キューのデフォルトのセットと、デフォルトのキュー・マネージャーがあります。
- SQL ステートメント内で使用する前に、キューおよび WebSphere MQ オブジェクトを作成する必要があります。
- パブリッシュおよびサブスクライブ機能を使用する場合、 WebSphere MQオブジェクトの中には SQL ステートメント内で使用する前に作成しなければならないものもあります。 WebSphere MQ および AMI (amtsamp.tst および amtsdfts.tst) により提供される *.tst ファイルを通して MQSC コマンドを発行することによってこれを行います。これを実行するには、以下のステップを使用してください。

  1. amtsamp.tst および amtsdfts.tst ファイルがあることを確認します。
  2. 必要に応じて、ご使用のキュー・マネージャーの *.tst ファイルを更新します。
  3. ご使用の AMI サービスが使用するキュー・マネージャーを開始します。
  4. 以下のコマンドに類似したコマンドを発行します。

     ```
     runmqsc QMName <amtsamp.tst
     ```

手順:

トランザクションおよび非トランザクションのユーザー定義関数には、**enable_MQFunctions** および **disable_MQFunctions** コマンドを使用します。MQ ユーザー定義関数はグループとして定義され、別のスキーマ名の下に設定されます。トランザクションをサポートしていないグループには、スキーマ **db2mq** があります。トランザクションをサポートしているグループには、スキーマ **db2mq1c** があります。トランザクションをサポートするオプションを持つ **enable_MQFunctions** コマンドを使用することによって、MQ ユーザー定義関数のセットを選択してトランザクション・サポートをインストールまたはアンインスト

ールすることができます。 **enable_MQFunctions** を使用するには、enable_MQFunctions および disable_MQFunctions の完全なコマンド構文を参照してください。

1. WebSphere MQ 機能のためにデータベースを構成および使用可能にします。enable_MQFunctions ユーティリティーはフレキシブルなコマンドです。これは、WebSphere MQ 環境が正しくセットアップされているかどうかを最初にチェックします。続いて、WebSphere MQ 機能のためにデフォルト構成をインストールおよび作成します。次に、これらの機能を持つ指定されたデータベースを使用可能にし、その構成が機能していることを確かめます。

   以下の例は、ユーザーがデータベース SAMPLE に接続していることを前提にしています。

   例 1: トランザクションおよび非トランザクションのユーザー定義関数を使用可能にする:

   ```
   enable_MQFunctions -n sample -u user1 -p password1
   ```

   例 2: スキーマ DB2MQ1C の下に DB2MQ1C 関数を作成する:

   ```
   enable_MQFunctions -n sample -u user1 -p password1 -v 1pc
   ```

2. コマンド行プロセッサー (Windows 環境で) を使用することによって MQ 機能をテストします。現在使用可能なデータベースに接続した後、以下のコマンドを発行します。

   ```
   values DB2MQ1C.MQSEND('a test')
   values DB2MQ1C.MQRECEIVE()
   ```

   最初のステートメントは、メッセージ `a test` を DB2MQ_DEFAULT_Q キューに送信します。 2 番目のステートメントはそれを受信します。このステートメントは、ユーザーがデフォルト構成のいくつかを使用していることを前提としています。作業単位の一部としてコミットまたはロールバックできる DB2 トランザクションでは、これらのステートメントのどちらも使用することができます。

関連概念:

- 229 ページの『DB2 Information Integrator の非同期メッセージング』
- 224 ページの『DB2 内で WebSphere MQ 機能を使用する方法』

関連タスク:

- 234 ページの『WebSphere MQ を MQListener 用に構成する』
- 236 ページの『MQListener の構成』
- 233 ページの『DB2 Universal Database 環境で実行するように MQListener を構成する』
- 232 ページの『MQListener の構成と実行』

関連資料:

- 「コマンド・リファレンス」の『enable_MQFunctions』
- 「コマンド・リファレンス」の『disable_MQFunctions』

WebSphere MQ および DB2 アプリケーションの統合の概要

DB2® Universal Database および WebSphere® MQ を使用して、SQL 要求の作成、ストアード・プロシージャーの開発、ユーザー定義関数を使用したデータベースの拡張、およびデータベース要求の Web サービスへの変換を行います。メッセージング、キューイング、パブリッシュおよびサブスクライブは、データベース・アプリケーション環境において共通のテクノロジーです。これらの技法は、異なるアプリケーションをリンクしたり、エンタープライズ内でリアルタイム情報を広めてデータおよび通信を統合したりするのに役立ちます。

WebSphere MQ は、異なるオペレーティング・システムおよびネットワーク間の分散環境でアプリケーションの通信を可能にするメッセージ処理システムです。WebSphere MQ は、アプリケーション・プログラミング・インターフェース (API) を使用して、あるプログラムから別のプログラムへの通信を処理します。

Application Messaging Interface (AMI) は、多くの高水準言語で使用可能な WebSphere MQ に通常使用される API です。 AMI に加えて、DB2 UDB は、DB2 WebSphere MQ 関数と呼ばれる外部のユーザー定義関数のセットを通して、それ自身のアプリケーション・プログラミング・インターフェースを WebSphere MQ メッセージング・システムに提供します。これらの関数を SQL ステートメント内で使用することによって、 DB2 Universal Database™ アクセスを WebSphere MQ メッセージ処理に結合することができます。

メッセージ処理および AMI の概要

WebSphere MQ メッセージ処理システムは、情報 (メッセージ) の一片を取得して、それを宛先に送信します。 WebSphere MQ は、起こり得るネットワークの中断にかかわりなくデリバリーを保証します。

アプリケーション・プログラマーは、AMI を使用してメッセージを送受信します。AMI にある 3 つのコンポーネントは次のとおりです。

- メッセージ。あるプログラムから別のプログラムに送信する**内容**を定義します。
- サービス。メッセージが行き来する**場所**を定義します。
- ポリシー。メッセージを処理する**方法**を定義します。

AMI を使用するメッセージを送信するには、アプリケーションはメッセージ・データ、サービス、およびポリシーを指定する必要があります。システム管理者は、デフォルト・サービスおよびデフォルト・ポリシーを含む、特定のインストールに必要な WebSphere MQ 構成を定義します。 DB2 UDB はデフォルト・サービスとデフォルト・ポリシー、 DB2.DEFAULT.SERVICE および DB2.DEFAULT.POLICY を提供しますが、アプリケーション・プログラマーはこれらを使用してプログラムを単純化することができます。

AMI についての詳細は、「*MQSeries Application Messaging Interface*」を参照してください。

WebSphere MQ メッセージ

WebSphere MQ はメッセージを使用してアプリケーション間で情報を受け渡します。メッセージは、以下のような部分から構成されています。

- メッセージ属性、これはメッセージおよびそのプロパティーを識別するものです。 AMI は属性およびポリシーを使用して、MQSeries® ヘッダーおよびメッセージ記述子を解釈および構成します。
- メッセージ・データ、これはメッセージ内で扱われているアプリケーション・データのことです。 AMI はこのデータ上では機能しません。

属性は AMI メッセージのプロパティーです。 AMI を使用することによって、メッセージに属性を含めたり、システム管理者はデフォルト・ポリシーで属性を定義することができます。アプリケーション・プログラマーは、メッセージ属性の詳細に配慮する必要はありません。

WebSphere MQ サービス

サービスは、アプリケーションがメッセージを送信する宛先か、アプリケーションがメッセージを受信する発信元を記述します。 WebSphere MQ は、宛先をメッセージ・キューとみなし、キューはキュー・マネージャー内に常駐します。

アプリケーションは、メッセージをキューに置くか、AMI を使用することによって、そこからメッセージを入手することができます。システム管理者は、サービスが定義する、キューを管理するためのパラメーターをセットアップします。それで、AMI はアプリケーション・プログラマーから複雑さを隠します。アプリケーション・プログラムは、あるサービスを DB2 MQSeries 関数呼び出しのパラメーターとして指定することによって、サービスを選択します。

WebSphere MQ ポリシー

ポリシーは、AMI 機能がメッセージを処理する方法を制御します。ポリシーは以下のようなアイテムを制御します。

- 例えば、優先順位といった、メッセージの属性。
- 例えば、ある操作が作業単位の一部かどうかといった、操作の送受信オプション。

AMI はデフォルト・ポリシーを提供します。またはその代わりに、システム管理者は、カスタマイズされたポリシーを定義しそれをリポジトリーに保管することができます。アプリケーション・プログラムは、あるポリシーを DB2 MQSeries 関数呼び出しのパラメーターとして指定することができます。

DB2 MQSeries 関数の性能

DB2 を使用した MQSeries 関数には 2 つの使用法があります。

- トランザクション・セマンティクス (スキーマ名は DB2MQ) を持たないユーザー定義関数
- 1 フェーズ・コミット・セマンティクス (スキーマ名は DB2MQ1C) を使用するユーザー定義関数

DB2 WebSphere MQ 関数は以下のタイプの操作をサポートしています。

- Send and forget、ここではメッセージは応答を必要としません。
- Read、ここでアプリケーションは、キューから除去することなく 1 つまたはすべてのメッセージを読み取ることができます。

- Receive、ここでアプリケーションは、1 つまたはすべてのメッセージをキューから受信および除去することができます。
- Request and response、ここでは送信中のアプリケーションは要求に対する応答を必要とします。

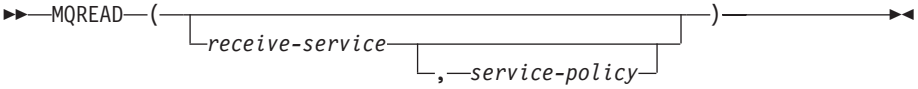
DB2 WebSphere MQ 関数を使用してメッセージをメッセージ・キューへ送信したり、メッセージ・キューからメッセージを受信したりすることができます。さらに、メッセージ・キューへ要求を送信して応答を受信することもできます。

DB2 データベース・サーバーと同じシステム上に、WebSphere MQ サーバーを配置しなければなりません。 DB2 WebSphere MQ 関数を DB2 データベース・サーバーに登録して、AMI を使用して WebSphere MQ サーバーへのアクセスを提供します。 DB2 MQSeries 関数のインストールについての詳細は、「DB2 WebSphere MQ 機能のインストール」を参照してください。

DB2 WebSphere MQ 関数には、スカラー関数および表関数の両方が含まれています。スキーマ名がユーザー定義関数のタイプを表していることを忘れないようにしてください。 MQ 関数についての詳細は、「DB2 WebSphere MQ 関数のセットアップ」を参照してください。以下の定義は、DB2 WebSphere MQ スカラー関数を説明しています。

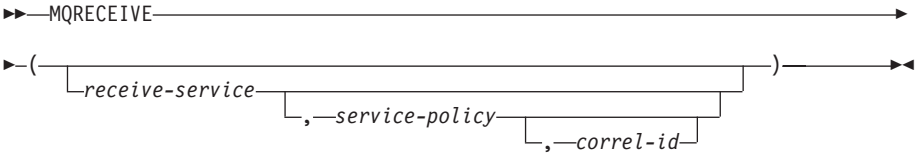
MQREAD

これは、*service-policy* で定義されたポリシーを使用して、 *receive-service* で指定された MQSeries ロケーションから VARCHAR 変数にメッセージを戻します。この操作はキューのヘッドからメッセージを除去しませんが、その代わりにメッセージを戻します。戻すことのできるメッセージが何もない場合は、NULL 値が戻されます。



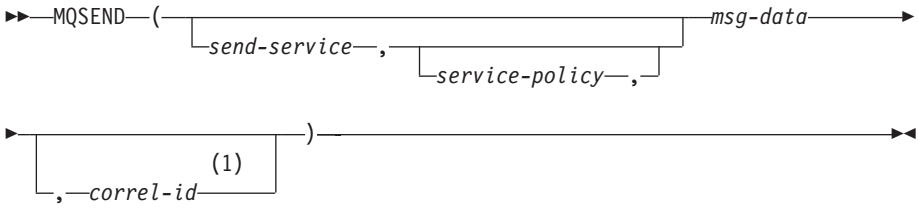
MQRECEIVE

これは、*service-policy* で定義されたポリシーを使用して、 *receive-service* で指定された MQSeries ロケーションから VARCHAR 変数にメッセージを戻します。この操作はメッセージをキューから除去します。 *correlation-id* を指定している場合、一致する相関 ID を持つ最初のメッセージが戻されます。 *correlation-id* を指定しない場合は、キューの先頭にあるメッセージが戻されます。戻すことのできるメッセージが何もない場合は、NULL 値が戻されます。



MQSEND

これは、*service-policy* で定義されたポリシーを使用して、 VARCHAR 変数 *msg-data* にあるメッセージを *send-service* で指定された MQSeries ロケーションへ送信します。 *correlation-id* を使用して、オプションのユーザー定義の相関 ID を指定することができます。正常に行われた場合、戻り値は 1、正常に行われなかった場合、戻り値は 0 です。



注:

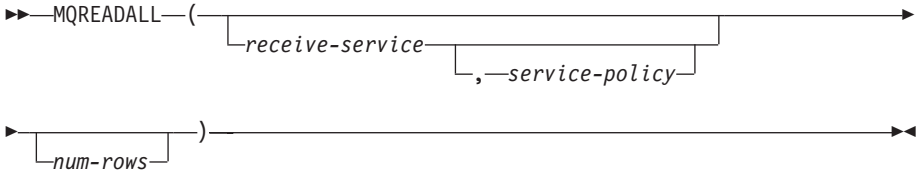
1 *service* および *policy* が指定されていなければ、*correl-id* を指定することはできません。

スキーマ DB2MQ および DB2MQ1C に対して VARCHAR 変数内のメッセージを送受信することができます。 VARCHAR 変数内の DB2MQ メッセージの最大長は、4,000 バイト長です。 DB2MQ1C は 32,000 バイト長までの VARCHAR をサポートしています。

以下の定義は、DB2 MQSeries 表関数を説明しています。

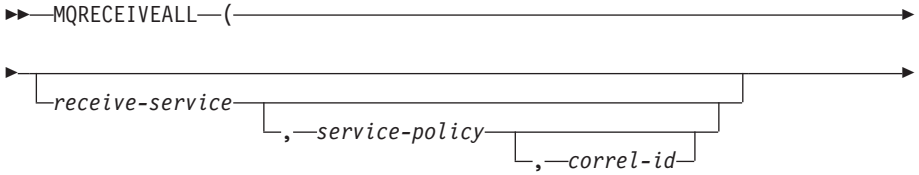
MQREADALL

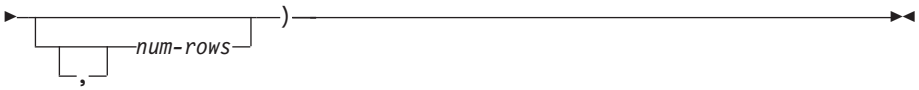
これは、*service-policy* で定義されたポリシーを使用して、 *receive-service* で指定された MQSeries ロケーションから VARCHAR 変数にメッセージおよびメッセージ・メタデータが入った表を戻します。この操作はメッセージをキューから除去しません。 *num-rows* が指定されている場合、最大で *num-rows* 個のメッセージが戻されます。 *num-rows* が指定されていない場合、使用可能なメッセージがすべて戻されます。



MQRECEIVEALL

これは、*service-policy* で定義されたポリシーを使用して、 *receive-service* で指定された MQSeries ロケーションから VARCHAR 変数にメッセージおよびメッセージ・メタデータが入った表を戻します。この操作はメッセージをキューから除去します。 *correlation-id* が指定されている場合、一致する相関 ID を持つメッセージのみが戻されます。 *correlation-id* が指定されていない場合、使用可能なメッセージがすべて戻されます。 *num-rows* が指定されている場合、最大で *num-rows* 個のメッセージが戻されます。 *num-rows* が指定されていない場合、使用可能なメッセージがすべて戻されます。





VARCHAR 変数内のメッセージを送受信することができます。 DB2MQ VARCHAR 変数内のメッセージの最大長は、4,000 バイトです。 DB2MQ1C 変数内のメッセージの最大長は、VARCHAR 32,000 バイトです。 DB2 MQSeries 表関数の結果表の最初の列にはメッセージが含まれます。

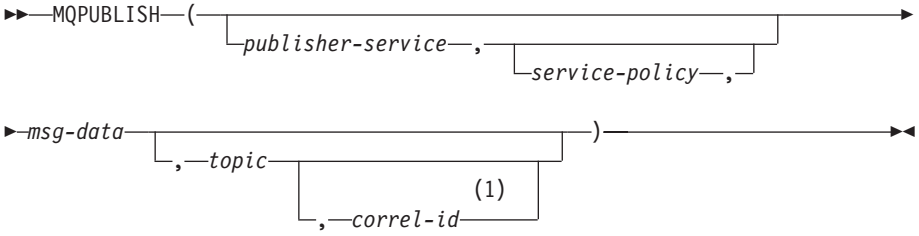
Information Integrator でビューとともに DB2 スカラーおよび表関数を使用することによって、すべての環境からメッセージ処理操作を SQL 照会に取り込むことができます。 WebSphere MQ クライアントまたはサーバーを持っている場合、 SQL ステートメント内でメッセージング操作を使用することができます。以下に例を示します。

```
SELECT DB2MQ1C.MQSend ('MyAddress'|| firstname ||' '|| lastname)
  FROM employee
```

メッセージをパブリッシュおよびサブスクライブすると、どのサービスがメッセージを受信するかをさらに制御できるようになります。システムをパブリッシュおよびサブスクライブすることによって、スケーラブルでセキュアな環境が提供され、多くのサブスクライバーが登録して、複数のパブリッシャーからメッセージを受信することができるようになります。 DB2 Universal Database 内でトリガー機能を使用して、トリガー起動の一部として自動的にメッセージを発行することができます。

MQPUBLISH

この関数はデータを MQSeries に発行します。この関数には、MQSeries Publish/Subscribe または MQSeries Integrator のいずれかのインストールが必要です。



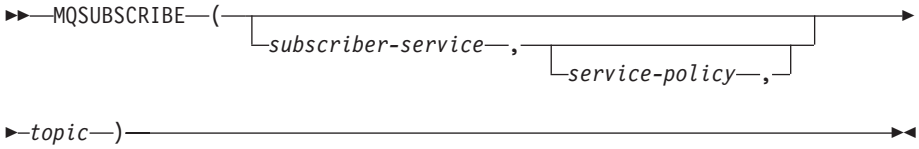
注:

1 *service* および *policy* が指定されていなければ、*correl-id* を指定することはできません。

MQSUBSCRIBE

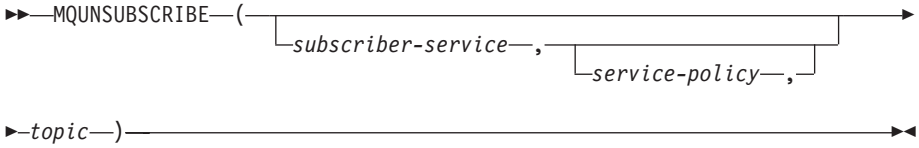
この関数は、特定のトピックについて発行された MQSeries メッセージにある興味のあるものを登録します。 subscriber-service は、指定されたトピックに一致するメッセージに対して論理宛先を指定します。トピックに一致するメッセージは、subscriber-service により定義されたキューに配置され、MQREAD、 MQRECEIVE、 MQREADALL、または MQRECEIVEALL への以降の呼び出しを通して読み取ったり受信したりすることができます。こ

の関数には、MQSeries Integrator または MQSeries Publish/Subscribe といった、パブリッシュおよびサブスクライブ・システムに基づく MQSeries のインストールおよび構成が必要です。



MQUNSUBSCRIBE

この関数を使用して、既存のメッセージ・サブスクリプションを登録抹消します。 subscriber-service、service-policy、および topic は、どのサブスクリプションをキャンセルするかを識別するのに使用します。この関数には、MQSeries Integrator または MQSeries Publish/Subscribe といった、パブリッシュおよびサブスクライブ・システムに基づく MQSeries のインストールおよび構成が必要です。



単純データ発行の例は、1 つのアプリケーションが興味のあるイベントに関して他のアプリケーションに通知する際に行われます。アプリケーションは、別のアプリケーションによってモニターされているメッセージをキューへ送信することによって、これを行います。メッセージの内容は、データベース列から合成されたユーザー定義のストリング、またはストリング値の関数呼び出し、他の正しいタイプのストリングを生み出す有効な式のいずれかです。

どのサービスが特定のメッセージを受信するかについてさらに制御する必要がある場合、パブリッシュおよびサブスクライブ機能を使用する必要があります。多くのサブスクライバーは複数のパブリッシャーからメッセージを受信するために登録する必要があります。ユーザーのメッセージと関連付けるトピックを指定することができます。例えば、DB2 アプリケーションはメッセージをサービス・ポイント *Weather* に発行することができます。メッセージは *Sleet* で、トピックは *Austin* です。

```
values DB2MQ1C.MQPublish ('Weather Bulletins','Sleet','Austin')
```

これは、興味のあるサブスクライバーにオースティンの天候がみぞれであることを通知します。サブスクライバーは、この種の情報を受信する際に以下のステートメントを使用して興味のあるものを登録します。

```
values DB2MQ1C.MQSUBSCRIBE('aSubscriber', 'Austin')
```

サブスクライバーが特定のトピックをサブスクライブすることに関心がなくなった場合、そのサブスクライバーは以下のステートメントを使用して明示的にアンサブスクライブする必要があります。

```
values DB2MQ1C.MQUNSUBSCRIBE('aSubscriber','Austin')
```

DB2 WebSphere MQ 関数のコミット環境

1 つのトランザクションは、通常、DB2 Universal Database では作業単位と呼ばれます。作業単位とは、1 つのアプリケーション・プロセス内の、リカバリー可能な一連の操作です。これは、データベースを整合性のある状態に保つために、データベース・マネージャーにより使用されます。データベースからの読み取りまたはデータベースへの書き込みは何であれ、 1 つの作業単位の中で行われます。最初の SQL ステートメントがデータベースで発行された時点で、作業単位は開始します。アプリケーションは、COMMIT または ROLLBACK ステートメントを実行して、作業単位を終了させる必要があります。 DB2 Universal Database は、DB2 WebSphere ユーザー定義関数を使用する場合、 2 つのバージョンのコミットを提供します。

- DB2MQC のスキーマ名を持つ非トランザクション UDF
- DB2MQ1C のスキーマ名を持つ単一フェーズ・コミット

MQ ユーザー定義関数のためのコミット環境は、ユーザーのアプリケーションに含まれる CONNECT のタイプにも依存しています。 CONNECT ステートメントは、アプリケーション・プロセスとそのサーバー間の接続を確立します。タイプ 1 CONNECT は、作業単位 (リモート作業単位) セマンティクスごとに単一データベースをサポートします。タイプ 2 CONNECT は、作業単位 (アプリケーション指定の分散作業単位) セマンティクスごとに複数のデータベースをサポートします。ONEPHASE の SYNCPOINT を指定することもできます。 SYNCPOINT は、COMMIT または ROLLBACK を複数のデータベース接続にまたがって調整する仕方を定義します。 ONEPHASE の SYNCPOINT を使用すると、作業単位の中で更新を行えるのは 1 つのデータベースに対してだけです。他のデータベースはすべて読み取り専用です。

非トランザクション関数—スキーマ DB2MQ: ご使用のアプリケーションが非トランザクションのユーザー定義関数を使用している場合、 DB2 COMMIT または ROLLBACK 操作はすべて WebSphere MQ 操作から独立しています。 MQ 関数は、トランザクションをロールバックする場合、ユーザーが現在の作業単位の中でキューに送信したメッセージを破棄しません。

この環境では、WebSphere MQ はそれ自身のキュー操作を制御します。 DB2 COMMIT または ROLLBACK は、ご使用のアプリケーションがメッセージを WebSphere MQ キューに追加またはそこから削除するかどうか、またそれをいつ行うかに影響しません。

単一フェーズ・コミット—スキーマ DB2MQ1C: ご使用のアプリケーションがデータ・ソースに対して 1 フェーズ・コミットを使用しており、トランザクションをロールバックしている場合、そのアプリケーションはメッセージを破棄するかエラーを発行する可能性があります。この結果、不整合状態になることがあります。SYNCPOINT=ONEPHASE を持つ 1 フェーズ・コミットの規則は以下の通りです。

- 更新を行えるのは 1 つのデータ・ソースに対してだけです。
- メッセージング機能を他の更新と結合することはできません。

表 27. DB2 MQ ユーザー定義関数セマンティクス

接続タイプ	1 フェーズ・コミット (スキーマ名=DB2MQ1c)

タイプ 1 (ONEPHASE)	`select db2mq1c.mqsend`
	`(e.LASTNAME || ' ' || d.DEPTNAME)`
	`from EMPLOYEE e,`
	`DEPT d`
	`where e.DEPARTMENT =`
	`d.DEPTNAME`
	
	アプリケーションは選択および送信することができます。更新を行えるのは 1 つのデータ・ソースだけです。

タイプ 2 (TWOPHASE)	メッセージ機能は許可されていません

DB2 MQ 関数が DB2 作業単位の一部である場合: DB2 MQ UDF を DB2 作業単位の一部として、または様々な種類の DB2 操作におけるトランザクションの一部として使用することができます。

複数の接続

これは 2 人のユーザーが同じデータベースに接続するシナリオを説明しています。 2 人とも DB2 MQ UDF を実行しています。一方の接続はメッセージを送信します。他方の接続は受信します。最初の接続がコミットするまで、2 番目の接続には最初の接続のメッセージが見えません。コミットした後、2 番目の接続には最初の接続のメッセージが見えます。最初の接続がロールバックを発行した場合、2 番目の接続にはそのメッセージが見えません。

表 28. 同じデータベースに接続している 2 人のユーザー

接続 1	接続 2
db2 +c // Turn auto commit off	
values db2mq1c.mqsend (「test message」)	

	`//The connection can not`
	`//see the`
	`//message yet:`
	`values db2mq1c.mqreceive();`

commit;	

	`//Now the connection`
	`//can see the message:`
	`values db2mq1c.mqreceive();`

トリガー

DB2 MQ UDF は、単一または複数のステートメント BEFORE または AFTER トリガーの一部になることができます。

```
create table EMPLOYEE
  (NAME VARCHAR(30), LASTNAME VARCHAR(30) NOT NULL PRIMARY KEY);

create trigger AFTER_TEST
    after insert on EMPLOYEE
    referencing NEW as NEWEMP
    for each row mode DB2SQL
    VALUES db2mq.mqsend(newemp.lastname);

insert into EMPLOYEE values ('MORGAN', 'TONG');
```

```
create trigger BEFORE_TEST
    no cascade before update of NAME on EMPLOYEE
    referencing NEW as NEWNAME OLD as OLDNAME
    for each row mode db2sql
    values db2mq.mqsend (oldname.lastname);

update EMPLOYEE set NAME = 'RAY';

.
```

制約事項:

ほとんどの DB2 SQL ステートメントで、メッセージング技法をデータベース操作に統合することができます。 DB2 MQ 関数を使用してエラーが発生した場合、DB2 Universal Database は自動的にトランザクションをロールバックします。ここには、DB2 MQ 関数を使用できないステートメントのいくつかの例があります。

- アプリケーションのセーブポイントを発行している場合。
- ユーザーが DB2 MQ UDF をアトミック・コンパウンド SQL ステートメント内から使用しようとしている場合。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CONNECT (タイプ 1) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CONNECT (タイプ 2) ステートメント』
- 「コマンド・リファレンス」の『SET CLIENT コマンド』
- 「*SQL Administrative Routines*」の『MQSEND スカラー関数』
- 「*SQL Administrative Routines*」の『MQRECEIVE スカラー関数』
- 「*SQL Administrative Routines*」の『MQREAD スカラー関数』
- 「*SQL Administrative Routines*」の『MQPUBLISH スカラー関数』
- 「*SQL Administrative Routines*」の『MQSUBSCRIBE スカラー関数』
- 「*SQL Administrative Routines*」の『MQUNSUBSCRIBE スカラー関数』
- 「*SQL Administrative Routines*」の『MQREADALL 表関数』
- 「*SQL Administrative Routines*」の『MQRECEIVEALL 表関数』

DB2 内で WebSphere MQ 機能を使用する方法

WebSphere® MQ および DB2® メッセージ操作は、データベース操作を単一の作業単位内でアトミック・トランザクションとして結合します。 DB2 MQ 機能を使用したメッセージングのもっとも基本的な形は、すべてのデータベース・アプリケーションが同じ DB2 UDB サーバーに接続している場合です。クライアントを、データベース・サーバーに対してローカルにするか、またはネットワーク環境に分配することができます。

簡単なシナリオでは、クライアント A は MQSEND 関数を呼び出して、ユーザー定義のストリングをデフォルトのサービスによって定義されたロケーションに送信します。 DB2 UDB は、データベース・サーバー上でこの操作を実行する MQSeries® 機能を実行します。その後で、クライアント B が MQRECEIVE 関数を呼び出します。これにより、デフォルトのサービスによって定義されたキューの先

頭にあるメッセージが除去されます。それから、関数はそれをクライアントに戻します。 DB2 UDB は、データベース・サーバー上でこの操作を実行する MQSeries 機能を実行します。

データベース・クライアントは、様々な方法で簡単なメッセージングを使用することができます。

- データ収集

  アプリケーションは、1 つかそれ以上のソースから、メッセージのフォーム内で情報を受け取ります。すべてのアプリケーションが情報源になることができます。アプリケーションは、キューからデータを受け取り、追加処理用としてデータベース表にそのデータを保管します。

- ワークロードの分散

  アプリケーションは、同じアプリケーションの複数インスタンスが共有するキューに作業要求を通知します。アプリケーション・インスタンスはある作業を実行する準備ができている場合、作業要求が含まれるメッセージをキューの先頭から受け取ります。アプリケーションの複数インスタンスは、プールされた要求の単一キューによって表されたワークロードを共有することができます。

- アプリケーション信号

  いくつかの処理が共同する状況では、メッセージを使用してそれぞれの働きを調整することができます。これらのメッセージには、作業を実行するためのコマンドまたは要求が含まれる場合があります。この技法について詳しくは、アプリケーション間の接続を参照してください。

以下のシナリオでは、基本メッセージングを拡張してリモート・メッセージングに組み込みます。マシン A はメッセージをマシン B に送信すると想定しています。

1. DB2 UDB クライアントは、MQSEND 関数呼び出しを実行して、マシン B でリモート・キューになるよう定義されたターゲット・サービスを指定します。
2. MQSeries 機能は作業を実行してメッセージを送信します。マシン A 上の MQSeries サーバーはそのメッセージを受け入れます。サーバーはメッセージを宛先に配信することを保証します。マシン A のサービスおよび現在の構成は宛先を定義します。サーバーは、宛先がマシン B 上のキューであると判断します。それから、サーバーはマシン B 上の MQSeries サーバーにメッセージの配信を試行し、必要に応じて再試行します。
3. マシン B 上の MQSeries サーバーは、マシン A 上のサーバーからメッセージを受け入れ、それをマシン B 上の宛先キューに配置します。
4. マシン B 上の MQSeries クライアントはキューの先頭にあるメッセージを要求します。

MQSEND を使用する場合、どのデータを、どこへ、いつ送信するかを選択します。このタイプのメッセージングは、*send and forget* と呼ばれます。送信側はメッセージの送信のみを行い、メッセージが宛先に到着するかどうかの確認は MQSeries に依存しています。

以下の例では、デフォルト・サービス DB2.DEFAULT.SERVICE およびデフォルト・ポリシー DB2.DEFAULT.POLICY を使用して、単一フェーズ・コミットに DB2MQ1C スキーマを使用します。単一フェーズ・コミットについて詳しくは、

『WebSphere MQ および DB2 アプリケーションの統合の概要』を参照してください。すべての例は、自動コミットがオフになっていることを前提としています。したがって、COMMIT が必要になります。 COMMIT がなければ、トランザクションの終了までロックを保持したままになる可能性があります。

例: 以下の CREATE TRIGGER ステートメントは、従業員表に挿入された姓名で構成されたメッセージを送信します。

```
CREATE TRIGGER T1
AFTER INSERT ON employee REFERENCING new AS newemp
   FOR EACH ROW MODE DB2SQL
  VALUES DB2MQ.MQSEND(newemp.name)
```

例: VARCHAR 列、LASTNAME、FIRSTNAME、および DEPARTMENT を持つ EMPLOYEE 表があると仮定します。また、自動コミットがオフになっていることを前提としています。 DEPARTMENT 5LGA 内のそれぞれの従業員ごとにこの情報が含まれるメッセージを送信するには、以下の SQL SELECT ステートメントを発行します。

```
SELECT DB2MQ1C.MQSEND (LASTNAME || ' ' || FIRSTNAME || ' ' || DEPARTMENT)
  FROM EMPLOYEE WHERE DEPARTMENT = '5lGA';
COMMIT;
```

メッセージ内容は、SQL ステートメント、式、関数、およびユーザー指定データの任意の組み合わせとなります。この MQSEND 関数は、単一フェーズ・コミット・セマンティクスを持つ DB2 MQ トランザクションのユーザー定義関数を使用するので、 COMMIT ステートメントはメッセージが MQSeries キューに追加されることを確認します。

DB2 MQSeries 機能を使用することによって、アプリケーションはメッセージを読み取りまたは受信できるようになります。読み取りと受信で違う点は、読み取りはメッセージをキューから除去せずにキューの先頭にあるメッセージを戻すという点です。受信すると、メッセージはキューから除去されます。受信操作を使用して取り出すメッセージは、一度だけ取り出すことができます。読み取り操作を使用して取り出すメッセージでは、同じメッセージを何度も取り出せます。

以下の例では、デフォルト・サービス DB2.DEFAULT.SERVICE およびデフォルト・ポリシー DB2.DEFAULT.POLICY を使用して、単一フェーズ・コミットに DB2MQ1C スキーマを使用します。単一フェーズ・コミットについて詳しくは、『WebSphere MQ および DB2 アプリケーションの統合の概要』を参照してください。

例: 以下の SQL SELECT ステートメントは、デフォルトのサービスおよびポリシーによって指定されたキューの先頭にあるメッセージを読み取ります。なお、自動コミットがオフになっていることを前提としています。

```
SELECT DB2MQ1C.MQREAD() FROM SYSIBM.SYSDUMMY1;
COMMIT;
```

SYSIBM.SYSDUMMY1 には 1 つの行しかないので、MQREAD 関数を一度呼び出します。 SELECT ステートメントは、VARCHAR(32000) ストリングを戻します。読み取ることのできるメッセージが何もない場合は、ステートメントの結果は NULL 値になります。

例: 以下の SQL SELECT ステートメントは、キューの内容を DB2 UDB 表としてマテリアライズします。

```
SELECT T.* FROM TABLE(DB2MQ1C.MQREADALL()) T;
```

表関数の結果表 T は、キューにあるすべてのメッセージとそれらのメッセージに関するメタデータで構成されます。キューはデフォルトのサービスによって定義されます。マテリアライズされた結果表の最初の列は、メッセージそのものであり、残りの列にはメタデータが含まれます。 SELECT ステートメントは、メッセージとメタデータの両方を戻します。

メッセージのみを戻すには、以下のステートメントを発行してください。

```
SELECT T.MSG FROM TABLE(DB2MQ1C.MQREADALL()) T;
```

表関数の結果表 T は、キューにあるすべてのメッセージとそれらのメッセージに関するメタデータで構成されます。キューはデフォルトのサービスによって定義されます。この SELECT ステートメントはメッセージのみを戻します。

例: 以下の SQL SELECT ステートメントは、キューからメッセージの送信を試行します。なお、自動コミットがオフになっていることを前提としています。

```
SELECT DB2MQ1C.MQSEND(name) FROM employees e;
ROLLBACK;
```

ROLLBACK ステートメントは、メッセージが DB2 UDB 操作と同じ作業単位の中にあるため、実際には送信されていないことを示しています。

例: 以下の SQL SELECT ステートメントは、デフォルトのサービス・キューからすべてのメッセージを取得します。

```
SELECT t.msg FROM table(DB2MQ1C.MQRECEIVEALL()) t;
COMMIT;
```

表関数の結果表 T は、デフォルトのサービス・キューにあるすべてのメッセージとそれらのメッセージに関するメタデータで構成されます。 SELECT ステートメントはメッセージのみを戻します。

関連概念:

- 216 ページの『WebSphere MQ および DB2 アプリケーションの統合の概要』

関連タスク:

- 213 ページの『DB2 WebSphere MQ 機能のインストール』

アプリケーション間の接続

通常、アプリケーション間の接続を使用して、アプリケーション・サブシステムの様々なセットを統合するという問題を解決できます。アプリケーションの統合を容易にするために、MQSeries® はアプリケーションを相互接続する手段を提供します。このセクションでは、*request-and-reply* 通信と呼ばれる 1 つの共通シナリオについて説明します。

request-and-reply メソッドは、あるアプリケーションが別のアプリケーションのサービスを要求できるようにします。これを行う 1 つの方法は、リクエスターがメッセ

ージをサービス・プロバイダーに送信して、ある作業を実行したい旨を要求することです。その作業を完了すると、プロバイダーは結果、または単に完了の確認をリクエスターに送信することを決定する場合があります。リクエスターが継続する前に応答を待っている場合を除いて、MQSeries は応答を要求に関連付ける方法を提供する必要があります。

MQSeries は、相関 ID を提供してリクエスターとプロバイダー間の取引におけるメッセージを相関します。リクエスターは、既知の相関 ID を使用してメッセージにマークを付けます。プロバイダーは、同じ相関 ID を使用して応答にマークを付けます。関連した応答を検索するために、リクエスターは、キューからメッセージを受信する際にその相関 ID を提供します。プロバイダーは、一致する相関 ID を使用して最初のメッセージをリクエスターに戻します。

以下の例では、単一フェーズ・コミットに DB2MQ1C スキーマを使用します。単一フェーズ・コミットについて詳しくは、222 ページの『DB2 WebSphere MQ 関数のコミット環境』を参照してください。

例: 以下の SQL SELECT ステートメントは、ストリング「Msg with corr id」で構成されているメッセージをサービス、MYSERVICE に送信します。これは、相関 ID CORRID1 を持つポリシー、MYPOLICY を使用します。

```
SELECT DB2MQ1C.MQSEND ('MYSERVICE', 'MYPOLICY', 'Msg with corr id', 'CORRID1')
  FROM SYSIBM.SYSDUMMY1;
COMMIT;
```

SYSIBM.SYSDUMMY1 には 1 つの行しかないので、MQSEND 関数を一度呼び出します。この MQSEND は、1 フェーズ・コミット UDF である DB2MQ1C スキーマを使用するため、メッセージは DB2® トランザクションの一部になります。

例: 以下の SQL SELECT ステートメントは、ID CORRID1 と一致する最初のメッセージを受け取ります。これは、サービス、MYSERVICE によって指定されたキューからメッセージを受け取ります。これには、ポリシー、MYPOLICY を使用します。

```
SELECT DB2MQ1C.MQRECEIVE ('MYSERVICE', 'MYPOLICY', 'CORRID1')
  FROM SYSIBM.SYSDUMMY1;
```

SELECT ステートメントは、VARCHAR(32000) ストリングを戻します。この相関 ID で使用可能なメッセージが何もない場合は、ステートメントの結果は NULL 値になり、キューは変更されません。

XML エクステンダー内で使用可能な WebSphere® MQSeries ユーザー定義関数を使用して、 DB2 と様々な WebSphere MQSeries インプリメンテーション間の XML メッセージのみを受け渡すことができます。まず、XML エクステンダー用にデータベースを使用可能にします。次に、MQSeries XML エクステンダー関数を以下の方法で使用可能にします。

```
enable_MQXML -n DATABASE -u USER -p PASSWORD
```

以下の表は、MQSeries XML 関数のいくつかについての要旨です。これらの関数は、DB2XML データベース・スキーマを持っています。これらは、MQ UDF トランザクションの制御下にありません。

表 29. MQSeries XML 関数

MQSeries XML 関数	説明
DB2XML.MQSendXML	XML メッセージをキューに送信します。
DB2XML.MQReadXML	一致する XML メッセージのキューからの非破壊読み取り。
DB2XML.MQReadAllXML	すべての XML メッセージのキューからの非破壊読み取り。
DB2XML.MQReadXMLCLOB	一致する XML CLOB メッセージのキューからの非破壊読み取り。
DB2XML.MQReadAllXMLCLOB	すべての XML CLOB メッセージのキューからの非破壊読み取り。

関連概念:

- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『MQSeries 使用可能性』

関連資料:

- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『MQReadAllXML 関数』
- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『MQReadXMLCLOB 関数』
- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『MQReadAllXMLCLOB 関数』
- 「*DB2 XML Extender 管理およびプログラミングのガイド*」の『MQSENDXML 関数』

DB2 Information Integrator の非同期メッセージング

プログラムは、同期リモート・プロシージャー呼び出しなどの構成を使用せずに、メッセージ内のデータを送信することによって互いに通信することができます。非同期メッセージングを使用することによって、メッセージを送信するプログラムは、メッセージの送信後に応答を待たずに処理を継続します。このプログラムが応答からの情報を必要とする場合は、処理を中断して、応答メッセージを待ちます。メッセージング・プログラムがメッセージを保持する中間キューを使用する場合は、要求側のプログラムと受信側のプログラムを同時に実行する必要はありません。要求側のプログラムはこのキュー上に要求メッセージを入れてから終了します。受信側のプログラムはキューから要求を取り出して、その要求を処理します。

非同期操作では、サービス・プロバイダーが通知なしでクライアントから要求を受諾できる必要があります。非同期リスナーは、WebSphere® MQ などのメッセージ・トランスポーターをモニターし、メッセージ・タイプに基づいてアクションを実行するプログラムです。非同期リスナーは、 WebSphere MQ を使用してエンドポイントに送信されたすべてのメッセージを受信することができます。非同期リスナーは、パブリッシュおよびサブスクライブ・インフラストラクチャーを使用してサブスクリプションを登録し、指定された制約を満たすメッセージのみ受信するように制限することもできます。

以下の例で、非同期メッセージングの一般的な使用法を示します。

メッセージ・アキュムレーター

非同期に送信されるメッセージを集計すると、リスナーはメッセージを検査して、データベース内に自動的にそれらのメッセージを保管できるようになります。このメッセージ・アキュムレーターとして行動するデータベースは、監査証跡などの特定のエンドポイントに対して、すべてのメッセージを保管することができます。非同期リスナーは、 *save only high value stock trades* (高値の株取引のみ保管する) というような、メッセージのサブセットをサブスクライブすることができます。メッセージ・アキュムレーターはメッセージ全体を保管し、メッセージ・コンテンツの選択、トランスフォーメーション、およびデータベース構造へのマッピングを提供しません。メッセージ・アキュムレーターはメッセージに応答しません。

メッセージ・イベント・ハンドラー

非同期イベント・ハンドラーはメッセージを listen して、該当するハンドラー (ストアード・プロシージャーなど) をメッセージ・エンドポイントに対して呼び出します。任意のストアード・プロシージャーを呼び出すことができます。非同期リスナーを使用すると、 1 つまたはそれ以上のデータベース構造に挿入するメッセージ・コンテンツを選択、マップ、または再フォーマットできるようになります。

非同期メッセージング・データベース対話を使用する場合の利点が以下にリストされています。

- クライアントおよびデータベースが同時に使用可能である必要はありません。クライアントが断続的に使用可能である場合、または要求が発行されてから応答が送信されるまでにクライアントで障害が起こった場合でも、依然としてクライアントは応答を受信することができます。または、クライアントがモバイル・コンピューター上にありデータベースから切断された場合でも、応答が送信されていれば、クライアントは依然として応答を受信することができます。
- データベース中のメッセージのコンテンツには、特定の要求を処理する時点に関する情報が含まれています。データベース中のメッセージは、優先順位および要求コンテンツを使用して、要求のスケジュール方法を決定します。
- 非同期メッセージ・リスナーは、異なるノードに対する要求を代行できます。 2 番目のコンピューターに要求を転送して処理を完了することができます。その要求が完了すると、 2 番目のコンピューターはメッセージ内の指定されたエンドポイントに対して応答を直接戻します。
- 非同期リスナーは、提供されたクライアントからの、またはユーザー定義のアプリケーションからのメッセージに応答することができます。これにより、データベース・クライアントとして行動できる環境の数は飛躍的に拡張します。ファクトリー・オートメーション (FA) 機能、パーベイシブ・デバイス、または組み込みコントローラーといったクライアントは、直接 WebSphere MQ を通して、または WebSphere MQ をサポートするいくつかのゲートウェイを通して、 DB2® Universal Database と通信することができます。

関連概念:

- 231 ページの『DB2 Information Integrator 中の MQListener』

関連タスク:

- 232 ページの『MQListener の構成と実行』

関連資料:

- 「コマンド・リファレンス」の『db2mqlsn - MQ Listener コマンド』
- 240 ページの『MQListener 構成中で使用するパラメーター』

DB2 Information Integrator 中の MQListener

IBM® DB2® Information Integrator は、 MQListener という名前の非同期リスナーを提供します。 MQListener は、WebSphere® MQ キューから読み取り、メッセージが到着するとそれとともに DB2 Universal Database™ ストアード・プロシージャーを呼び出す、というタスクのためのフレームワークです。

MQListener はメッセージングとデータベース操作を結合します。構成データベース中で指定した WebSphere MQ メッセージ・キューを listen するように、MQListener デーモンを構成できます。 MQListener は、キューから着信したメッセージを読み取ってから、メッセージを入力パラメーターとして使用して DB2 UDB ストアード・プロシージャーを呼び出します。メッセージに応答する必要がある場合は、 MQListener はストアード・プロシージャーで生成された出力から応答を生成します。メッセージの検索順序は、最も高い優先順位を最初に検索するよう固定されており、その優先順位のメッセージのうち先頭のメッセージが最初に検索されます。

MQListener は、単一のマルチスレッド処理として稼働します。各スレッドまたはタスクは、入力用に構成されたメッセージ・キューへの接続を確立します。各タスクは、ストアード・プロシージャーを実行する DB2 UDB データベースへの接続も確立します。キューやストアード・プロシージャーに関する情報は、構成データベース中の表に保管されます。キューとストアード・プロシージャーの組み合わせが 1 つのタスクになります。

MQListener タスクは、名前付きの構成にグループ化されます。デフォルトでは、構成名は空です。タスクの構成名を指定しない場合には、MQListener は空の名前の構成を使用します。

MQListener は、メッセージ・キューの読み書き操作とストアード・プロシージャーを、 1 つのトランザクションに統合できます。トランザクション・タスクの実行時に、キューからメッセージを読み取ってからストアード・プロシージャーがそのメッセージを受信するまでの間にコンピューターに障害が起きた場合でも、そのメッセージは失われません。デフォルトでは、ストアード・プロシージャーへの呼び出しのみでトランザクションになります。キューからメッセージを除去する操作とストアード・プロシージャーを呼び出す操作を同じトランザクションに結合したい場合は、 db2mqlsn コマンドに *-mqcoordinated* パラメーターを指定して使用し、WebSphere MQ 環境をコーディネーターとして構成してください。該当するキュー・マネージャーを構成し、 WebSphere MQ ガイドラインに従って適切なリソースに調整する必要があります。トランザクション・キュー操作を指定しない場合は、キュー・マネージャーをトランザクション・マネージャーとして構成しないでください。トランザクション・コーディネーターとして構成されたキュー・マネージャーを含む非トランザクション・タスクを実行しないでください。

MQListener configuration の一部として、構成ユーザー (-configUser) と実行ユーザー (-dbUser) を指定します。**構成ユーザー**と**実行ユーザー**は、違う種類のアクセス権を

持っている別のユーザーでもかまいません。実行ユーザーは、構成ユーザーの特権を継承しません。通常の MQListener のシナリオでは、1 人のユーザーが MQListener アプリケーションを実行します。 MQListener を実行するユーザーには、 WebSphere MQ 関数にアクセスする権限のみ必要です。通常は、Windows® および UNIX® オペレーティング・システムの *mqm* グループのメンバーになると、この権限を持つことができます。通常は、構成ユーザーが MQListener を実行するユーザーになります。

MQListener のストアード・プロシージャー・インターフェースは、着信メッセージを入力として使用し、応答 (NULL の場合もある) を出力として戻します。

```
schema.proc(in inMsg inMsgType, out outMsg outMsgType)
```

inMsgType のデータ・タイプと *outMsgType* のデータ・タイプは、任意の長さの VARCHAR、VARCHAR FOR BIT DATA、CLOB、または BLOB です。入力データ・タイプと出力データ・タイプは、違うデータ・タイプである場合もあります。

関連概念:

- 229 ページの『DB2 Information Integrator の非同期メッセージング』

関連タスク:

- 236 ページの『MQListener の構成』
- 232 ページの『MQListener の構成と実行』

関連資料:

- 「コマンド・リファレンス」の『db2mqlsn - MQ Listener コマンド』
- 240 ページの『MQListener 構成中で使用するパラメーター』

MQListener の構成と実行

以下の手順を使用して、MQListener 用に環境を構成します。さらにメッセージを受け取り、そのメッセージを表に挿入し、単純な応答メッセージを生成する単純なアプリケーションを開発します。

手順:

MQListener を構成して実行するには、以下のようにします。

1. DB2 Universal Database 環境で実行するために MQListener を構成します。
2. MQListener 用に WebSphere MQ を構成します。
3. MQListener を構成します。
4. MQListener で作業するためのストアード・プロシージャーを作成します。
5. 単純な MQListener アプリケーションを実行します。

関連概念:

- 231 ページの『DB2 Information Integrator 中の MQListener』

関連タスク:

- 233 ページの『DB2 Universal Database 環境で実行するように MQListener を構成する』
- 234 ページの『WebSphere MQ を MQListener 用に構成する』

- 236 ページの『MQListener の構成』
- 237 ページの『ストアード・プロシージャーを作成して MQListener と共に使用する』

関連資料:

- 238 ページの『MQListener の例』
- 240 ページの『MQListener 構成中で使用するパラメーター』
- 241 ページの『MQListener で使用する WebSphere MQ キュー』

DB2 Universal Database 環境で実行するように MQListener を構成する

ご使用のアプリケーションがメッセージングとデータベース操作を共に使用できるように、データベース環境を構成してください。

前提条件:

MQListener 構成用のデータベースと、メッセージ着信時に呼び出すストアード・プロシージャー用のデータベースを作成してください (有効なデータベースが使用可能になっていない場合)。構成用とストアード・プロシージャー用に同じデータベースを使用できます。

構成ユーザーに以下の特権と許可がなければなりません。

- DB2 UDB 表 SYSMQL.LISTENERS への読み取りおよび書き込みアクセス権。MQListener の run users は、 SYSMQL.LISTENERS へのアクセス権は必要はありません。
- 構成パッケージ MQLConfi を実行する権限。

実行ユーザーに、MQLRun パッケージを実行する権限がなければなりません。

手順:

DB2 Universal Database データベースと共に稼働するよう MQListener を構成するには、以下のようにします。

1. 以下のコマンドを発行して接続します。ご使用のデータベース環境に該当する値に置き換えます。

   ```
   db2 connect to ConfigDB user DBAdmin using DBAdminPwd
   ```

2. *MQLInstall.sql* スクリプトを実行します。このスクリプトは、MQListener の構成を保管する表を作成します。このスクリプトは以下のパス中にあります。
   - UNIX 環境の場合 .../sqllib/bin
   - Windows 環境の場合 ...¥sqllib¥bin

   ```
   db2 -td; -f MQLInstall.sql
   ```

3. 以下のコマンドを発行して、構成ユーザーにアクセス権を付与します。ご使用のデータベース環境に該当する値に置き換えます。

   ```
   db2 grant all privileges on table SYSMQL.LISTENERS to ConfigUser
   db2 connect reset
   ```

4. MQListener パッケージをバインドして、そのパッケージに対するアクセス権を付与します。構成データベース中で MQLConfig パッケージをバインドしなければなりません。以下のコマンドを発行します。ご使用のデータベース環境に該当する値に置き換えます。

```
db2 connect to ConfigDB user DBAdmin using DBAdminPwd
db2 bind MQLConfig.bnd
db2 grant execute on package MQLConfi to ConfigUser
db2 connect reset
```

名前 MQLConfi は、パッケージ名の長さに関する 8 文字の制限を満たしています。

5. 個々の実行データベース中で MQLRun パッケージをバインドします。実行データベースごとおよびそれぞれのデータベース中の実行ユーザーごとに、以下のコマンドを発行します。

```
db2 connect to RunDB user DBAdmin using DBAdminPwd
db2 bind MQLRun.bnd
db2 grant execute on package MQLRun to RunUser
db2 connect reset
```

関連概念:

- 231 ページの『DB2 Information Integrator 中の MQListener』

関連タスク:

- 234 ページの『WebSphere MQ を MQListener 用に構成する』
- 236 ページの『MQListener の構成』
- 237 ページの『ストアード・プロシージャーを作成して MQListener と共に使用する』

関連資料:

- 240 ページの『MQListener 構成中で使用するパラメーター』
- 241 ページの『MQListener で使用する WebSphere MQ キュー』
- 238 ページの『MQListener の例』

WebSphere MQ を MQListener 用に構成する

単純な WebSphere MQ の構成で、単純な MQListener アプリケーションを実行できます。複雑なアプリケーションほど、複雑な構成が必要になります。キュー・マネージャーとローカル・キューの 2 種類以上の WebSphere MQ エンティティーを構成してください。これらのエンティティーを、トランザクション管理、送達不能キュー、バックアウト再キューイング、およびバックアウト再試行しきい値などのインスタンスで使用するように構成してください。

前提条件:

mqm グループに入っている時点で、WebSphere MQ 制御コマンドを発行してください。 *mqm* グループは、WebSphere MQ 管理者が内部 MQ プログラム用に使用します。このグループのすべてのメンバーには、すべてのリソースへのアクセス権があります。

手順:

単純な MQListener アプリケーション用に WebSphere MQ を構成するには、以下のようにします。

1. キュー・マネージャーを作成します。

   ```
   crtmqm TransQM
   ```

2. キュー・マネージャーを開始します。

   ```
   strmqm TransQM
   ```

3. オプション: DB2 Universal Database とのトランザクションを調整するように、キュー・マネージャーを構成します。

   DB2 Universal Database とのトランザクションを調整するようにキュー・マネージャーを構成すると、 1 つのトランザクションで MQListener アプリケーションがメッセージを除去してストアード・プロシージャーを呼び出せます。以下のようにして、キュー・マネージャーを DB2 UDB 調整用に構成します。

   - WebSphere MQ が、 DB2 Universal Database の X/Open リソース・マネージャー関数と、 DB2 UDB に固有の拡張アーキテクチャー・オープン・ストリング (xa_open) を検索するのに使用する、共有ライブラリー (スイッチ・ロード・ファイルという) の名前を指定します。
   - DB2 UDB グローバル変数 db2xa_switch を戻す小さな C プログラムをコンパイルして、スイッチ・ロード・ファイル中に MQStart ルーチンを作成します。スイッチ・ロード・ファイルの作成方法に関する特定の情報については、「*WebSphere MQ: システム管理ガイド*」を参照してください。 MQStart は、DB2 UDB 中に X/Open リソース・マネージャー関数をインプリメントする関数を指すポインターの構造を戻します。以下の例は、拡張アーキテクチャー・オープン・ストリングの必須形式と、置き換えられる MQListener 構成パラメーターの該当値を示します。

     ```
     DB=RunDB, UID=RunUser, PWD=RunUserPwd, TPM=MQ, TOC=P
     ```

     この例のように、拡張アーキテクチャー・オープン・ストリング中で TPM=MQ を使用する場合は、 DB2 TP_MON_NAME インスタンス変数を設定する必要はありません。

   WebSphere MQ は、オペレーティング・システム環境に基づいて必要なパラメーターを取得します。 Windows オペレーティング・システムの場合は、このパラメーターは Windows レジストリー中にあります。 WebSphere MQ MQServices を使用して、このパラメーターを指定できます。 UNIX オペレーティング・システムの場合は、このパラメーターはキュー・マネージャー構成ファイル中にあります。ご使用の環境で有効なテキスト・エディターを使用して、このパラメーターを指定して使用してください。

4. WebSphere MQ スクリプト機能を使用して、ローカル・キューを作成します。

   a. 以下のコマンドを含むファイルを作成します (この例では、ファイルは mqconfig.mqs です)。

      ```
      define qlocal('DLQ')
      alter qmgr deadq('DLQ')
      define qlocal('Backout')
      define qlocal('Admin')
      define qlocal('In') boqname('Backout') bothresh(3)
      define qlocal('SYSTEM.SAMPLE.REPLY')
      ```

   b. 以下のコマンドを発行して、 mqconfig.mqs ファイルをスクリプト・インタープリター中にリダイレクトします。

```
runmqsc TransQM < mqconfig.mqs
```

関連タスク:

- 236 ページの『MQListener の構成』

関連資料:

- 240 ページの『MQListener 構成中で使用するパラメーター』
- 241 ページの『MQListener で使用する WebSphere MQ キュー』

MQListener の構成

MQListener を構成するには、 MQListener コマンド **db2mqlsn** を使用してください。任意のディレクトリー中のコマンド行から、コマンド **db2mqlsn** を発行してください。 Windows システムの場合は、DB2 UDB コマンド行プロセッサーでコマンドを発行して、確実に適切なメッセージが表示されるようにしてください。 db2mqlsn コマンドに add パラメーターを指定すると、 DB2 表 SYSMQL.LISTENERS 中の行が更新されます。

制約事項:

- 要求キューおよび応答キューには、同じキュー・マネージャーを使用します。
- Windows システムの場合、スレッドごとに 1 つのキュー・マネージャーに接続できます。
- UNIX システムの場合、プロセスごとに 1 つのキュー・マネージャーに接続できます。 UNIX システムの場合に、同一の MQListener 構成中でさまざまなキュー・マネージャーを指定すると、 WebSphere MQ からランタイム・エラーを受け取ります。
- MQListener は、複数の物理メッセージから成る論理メッセージをサポートしていません。 MQListener は物理メッセージを個別に処理します。

手順:

MQListener の構成を指定するには、以下のようにします。

- MQListener の構成を追加するには、以下のコマンドを発行します。

```
db2mqlsn add
  -configDB ConfigDB
  -config aConfiguration
  -configUser ConfigUser
  -configPwd ConfigUserPwd
  -queueManager TransQM
  -inputQueue In
  -procSchema RunUser
  -procName aProc
  -dbName RunDB
  -dbUser RunUser
  -dbPwd RunUserPwd
  -mqCoordinated
```

- 構成中のすべてのタスクを表示するには、以下のコマンドを発行します。

```
db2mqlsn show
  -configDB ConfigDB
  -config aConfiguration
  -configUser ConfigUser
  -configPwd ConfigUserPwd
```

- メッセージング・タスクを除去するには、以下のコマンドを発行します。

```
db2mqlsn remove
  -configDB ConfigDB
  -config aConfiguration
  -configUser ConfigUser
  -configPwd ConfigUserPwd
  -queueManager TransQM
  -inputQueue In
```

- コマンドと有効パラメーターに関するヘルプを表示するには、以下のコマンドを発行します。

  ```
  db2mqlsn help
  ```

- 特定のパラメーターに関するヘルプを表示するには、以下の例のように、特定のパラメーターを指定してコマンドを発行します。

  ```
  db2mqlsn help <command>
  ```

関連資料:

- 「コマンド・リファレンス」の『db2mqlsn - MQ Listener コマンド』
- 240 ページの『MQListener 構成中で使用するパラメーター』

ストアード・プロシージャーを作成して MQListener と共に使用する

実行データベースには、メッセージの着信時に実行されるストアード・プロシージャーが含まれています。実行ユーザーとは、 MQListener が実行データベースに接続してストアード・プロシージャーを実行する際に名前が使用されるユーザーのことです。実行データベースと実行ユーザーを定義するには、 db2mqlsn add コマンドに以下のパラメーターを指定して使用してください。

- -dbName
- -dbUser

実行ユーザーは、実行データベースに接続してストアード・プロシージャーを実行できなければなりません。ストアード・プロシージャーの所有者を実行ユーザーにする必要はありません。実行ユーザーは、MQListener 構成にアクセスする必要もありません。

MQListener はストアード・プロシージャー aProc を使用して、表にメッセージを保管します。メッセージが表に正常に挿入された場合は、このストアード・プロシージャーはストリング OK を戻します。

前提条件:

このストアード・プロシージャーには C コンパイラーが必要です。

手順:

以下のステップで、MQListener アプリケーションと共に使用できる DB2 Universal Database オブジェクトを作成します。

1. 実行ユーザーとして単純な表を作成します (DB2 UDB コマンド行プロセッサーを使用できます)。

   ```
   CREATE TABLE aTable (val VARCHAR(25) CHECK (val NOT LIKE 'fail%'))
   ```

この表にはチェック制約が含まれているので、先頭が文字 fail のメッセージはこの表に挿入できません。このチェック制約を使用して、ストアード・プロシージャーが失敗した場合の MQListener の動作が示されます。

2. 以下のストアード・プロシージャーを作成します。

```
CREATE PROCEDURE aProc (IN pin VARCHAR(25), OUT pout VARCHAR(2))
BEGIN
         INSERT INTO aTable VALUES(pin);
         SET pout = 'OK';
END
```

関連タスク:

- 233 ページの『DB2 Universal Database 環境で実行するように MQListener を構成する』

関連資料:

- 240 ページの『MQListener 構成中で使用するパラメーター』

MQListener の例

以下の例で、単純な MQListener アプリケーションを示します。このアプリケーションは、メッセージを受け取り、そのメッセージを表に挿入し、単純な応答メッセージを生成します。このアプリケーションには、処理の失敗をシミュレートするために、メッセージを含む表に関するチェック制約が組み込まれています。この制約により、先頭が文字 fail のストリングは表に挿入されません。チェック制約に違反するメッセージを挿入しようとすると、例のアプリケーションはエラー・メッセージを戻し、失敗したメッセージをバックアウト・キューに再キューイングします。

構成中に指定されているすべてのタスクを指定して MQListener を実行するには、以下のコマンドを発行します。

```
db2mqlsn run
  -configDB ConfigDB
  -config aConfiguration
  -configUser ConfigUser
  -configPwd ConfigUserPwd
  -adminQueue Admin
  -adminQMgr TransQM
```

以下の例は、MQListener を使用して、単純なメッセージを送信してから、WebSphere MQ キュー・マネージャーとデータベース中でメッセージの結果を検査する方法を示しています。この例には、入力キューにメッセージが含まれているかどうかや、ストアード・プロシージャーによってレコードが表に入れられているかどうかを判別する照会が組み込まれています。 DB2 Universal Database コマンド行プロセッサー、DB2 UDB Command Center、一部の WebSphere MQ コマンド行ユーティリティー、サンプル・プログラム、 MQ Explorer、MQ API エクササイザーを含む多数のツールがこれらの操作をサポートしています。さらに複雑なアプリケーションの場合は、 DB2 Universal Database と WebSphere MQ のツールを使用することを考慮してください。

MQListener の例 1: 単純なアプリケーションの実行:

1. 空のデータベース表を使用して作業を始めます。

```
db2 delete from aTable
```

2. データグラムを入力キューに送信します。
   a. sampleMsg1.txt という名前のファイル中にストリング `a sample message` を入れます。
   b. WebSphere MQ サンプル・プログラム **amqsput** を使用して、キューにメッセージを挿入します。

      ```
      amqsput In TransQM < sampleMsg1.txt
      ```
3. 表を照会して、サンプル・メッセージが挿入されているか検査します。

   ```
   db2 select * from aTable
   ```
4. 入力キュー中に残っているメッセージの数を表示し、メッセージが除去されているか検査します。
   a. ファイル checkIn.mqs 中に以下のコマンドを入れます。

      ```
      display queue('In') curdepth
      ```
   b. コマンドをスクリプト・インタープリター中にリダイレクトします。

      ```
      runmqsc TransQM < checkIn.mqs
      ```

MQListener の例 2: 入力キューへの要求の送信と応答の検査:

以下の例のステートメントは、入力キューに要求を送信し、応答を検査します。

1. 空のデータベース表を使用して作業を始めます。

   ```
   db2 delete from aTable
   ```
2. 要求を入力キューに送信します。
   a. sampleMsg2.txt という名前のファイル中にストリング `another sample message` を入れます。
   b. WebSphere MQ サンプル・プログラム **amqsreq** を使用して、入力キューに要求を送信します。

      ```
      amqsreq In TransQM < sampleMsg2.txt
      ```

      amqsreq プログラムは、要求中の応答先キューを SYSTEM.SAMPLE.REPLY に設定します。
3. 表を照会して、サンプル・メッセージが挿入されているか検査します。

   ```
   db2 select * from aTable
   ```
4. 入力キュー中に残っているメッセージの数を表示し、メッセージが除去されているか検査します。

   ```
   display queue('In') curdepth
   ```
5. WebSphere MQ サンプル・プログラム amqsget を使用し、SYSTEM.SAMPLE.REPLY キューを参照して応答を調べます。ストアード・プロシージャーによって OK ストリングが生成されているか検査します。

   ```
   amqsget SYSTEM.SAMPLE.REPLY TransQM
   ```

MQListener の例 3 失敗した挿入操作のテスト:

先頭がストリング `fail` のメッセージを送信すると、表定義中の制約に違反するので、ストアード・プロシージャーが失敗します。

1. 空のデータベース表を使用して作業を始めます。

   ```
   db2 delete from aTable
   ```
2. 要求を入力キューに送信します。

a. sampleMsg3.txt という名前のファイル中にストリング `failing sample message` を入れます。

b. WebSphere MQ サンプル・プログラム **amqsreq** を使用して、入力キューに要求を送信します。

```
amqsreq In TransQM < sampleMsg3.txt
```

amqsreq プログラムは、要求中の応答先キューを SYSTEM.SAMPLE.REPLY に設定します。

3. 表を照会して、サンプル・メッセージが挿入されていないか検査します。

```
db2 select * from aTable
```

4. 入力キュー中に残っているメッセージの数を表示し、メッセージが除去されているか検査します。

```
display queue('In') curdepth
```

5. SYSTEM.SAMPLE.REPLY キューを参照して、OK 応答ではなく例外報告を検索します。

```
amqsget SYSTEM.SAMPLE.REPLY  TransQM
```

6. バックアウト・キューを参照して、オリジナル・メッセージを検索します。

```
amqsget Backout TransQM
```

関連概念:

- 231 ページの『DB2 Information Integrator 中の MQListener』

関連タスク:

- 234 ページの『WebSphere MQ を MQListener 用に構成する』
- 236 ページの『MQListener の構成』
- 237 ページの『ストアード・プロシージャーを作成して MQListener と共に使用する』
- 233 ページの『DB2 Universal Database 環境で実行するように MQListener を構成する』
- 232 ページの『MQListener の構成と実行』

関連資料:

- 240 ページの『MQListener 構成中で使用するパラメーター』
- 241 ページの『MQListener で使用する WebSphere MQ キュー』

MQListener 構成中で使用するパラメーター

ConfigDB

MQListener 構成表を含む構成データベース。任意の有効な DB2 Universal Database でかまいません。構成表には、 MQListener の listen 先のキューや MQListener が呼び出すストアード・プロシージャーに関する情報が含まれます。

ConfigUser

構成データベースにアクセスする際に名前を使用するユーザーの ID。データベース管理者を構成ユーザーにする必要はありません。 MQListener コマンドで構成ユーザーとパスワードを指定できます。構成ユーザーとパスワードを指定しない場合に、データベースのインストール時の設定で暗黙接続が

サポートされていると、デフォルトでは MQListener を実行しているアカウントを持つユーザーが構成ユーザーになります。

ConfigUserPwd
構成ユーザー ID と共に使用するパスワード。

RunDB
実行データベースとは、メッセージの着信時に実行されるストアード・プロシージャーを含むデータベースのことです。構成データベース以外のデータベースにストアード・プロシージャーがある場合もあります。

RunUser
実行データベースにアクセスしてストアード・プロシージャーを実行する際に名前を使用するユーザーの ID。実行ユーザーには、実行データベースに接続してストアード・プロシージャーを実行する権限以外の特権は必要ありません。

RunUserPwd
実行ユーザーと関連したパスワード。

関連概念:

• 231 ページの『DB2 Information Integrator 中の MQListener』

関連資料:

• 「コマンド・リファレンス」の『db2mqlsn - MQ Listener コマンド』

MQListener で使用する WebSphere MQ キュー

単純な MQListener アプリケーションでは、通常は以下の WebSphere MQ キューが使用されます。

送達不能キュー
WebSphere MQ 中の送達不能キュー (DLQ) は、処理できないメッセージを保持します。 MQListener は、このキューを使用して、応答の送信先のキューがいっぱいになっているなどの理由で配送できない応答を保持します。送達不能キューは、すべての MQ インストール環境で、特に送信されないメッセージをリカバリーする場合に便利です。

バックアウト・キュー
WebSphere MQ がトランザクション・コーディネーターになっている MQListener タスクの場合、バックアウト・キューは送達不能キューと同じ目的を果たします。 MQListener は、指定された回数 (バックアウトしきい値という) だけ要求がロールバックされた後に、オリジナルの要求をバックアウト・キューに入れます。

管理キュー
管理キューは、 *shutdown* や *restart* などの制御メッセージを MQListener にルーティングするのに使用します。管理キューを使用しない場合には、MQListener をシャットダウンする方法は、 **kill** コマンドを発行することだけになります。

アプリケーション入出力キュー
アプリケーションは入力キューと出力キューを使用します。アプリケーションは入力キューからメッセージを受け取ります。アプリケーションは出力キ

ューに応答や例外を送信します。この出力キューは SYSTEM.SAMPLE.REPLY で、 WebSphere MQ サンプル・プログラム amqsreq での使用法に準拠します。

関連タスク:

- 234 ページの『WebSphere MQ を MQListener 用に構成する』
- 232 ページの『MQListener の構成と実行』

付録 A. Cottonwood Distributors, Inc. および YBar, Inc. シナリオのスクリプト例

```
-------------------------------------------------
-- Catalog remote DB2 UDB server machine and database
-------------------------------------------------
uncatalog node DB2_TPCH;
catalog tcpip node DB2_TPCH remote x.xx.xx.xx server 50000;

uncatalog database tpcd;
catalog database tpcd at node db2_tpch;

-------------------------------------------------
-- DB2 UDB wrapper, nicknames, MQTs, and indexes
-------------------------------------------------
drop wrapper drda;
create wrapper drda;

create server db2_tpch type db2/udb version  8.1
  wrapper drda authorization "demo" password "xxxxx"
  options (dbname 'TPCD');
create user mapping
  for user SERVER db2_tpch
  OPTIONS ( REMOTE_AUTHID 'demo', REMOTE_PASSWORD 'xxxxx' );

create nickname db2_part for db2_tpch.tpcd.part;
create nickname db2_supplier for db2_tpch.tpcd.supplier;
create nickname db2_partsupp for db2_tpch.tpcd.partsupp;
create nickname db2_nation for db2_tpch.tpcd.nation;
create nickname db2_region for db2_tpch.tpcd.region;
create nickname db2_customer for db2_tpch.tpcd.customer;
create nickname db2_orders for db2_tpch.tpcd.orders;
```

図 72. federated.sql (1/9)

```
-------------------------------------------------
-- Oracle wrapper, nicknames, MQTs and indexes
-------------------------------------------------
drop wrapper net8;
create wrapper net8;

create server oraserver type oracle version 8
  wrapper net8
  options (node 'iidemo2');
create user mapping
  for user SERVER oraserver
  OPTIONS ( REMOTE_AUTHID 'demo', REMOTE_PASSWORD 'xxxxx' );

create nickname ora_part for oraserver.demo.part;
create nickname ora_supplier for oraserver.demo.supplier;
create nickname ora_partsupp for oraserver.demo.partsupp;
create nickname ora_customer for oraserver.demo.customer;
create nickname ora_orders for oraserver.demo.orders;
create nickname ora_lineitem for oraserver.demo.lineitem;
```

図 72. federated.sql (2/9)

```
-------------------------------------------------
-- Informix wrapper, nicknames, MQTs, and indexes
-------------------------------------------------
drop wrapper informix;
create wrapper informix;

create server infserver type informix version 9
  wrapper informix
  options (node 'ol_informix',dbname 'tpcd2');
create user mapping
  for user SERVER infserver
  OPTIONS ( REMOTE_AUTHID 'informix', REMOTE_PASSWORD 'informix' );

create nickname inf_part for infserver."informix"."part";
create nickname inf_supplier for infserver."informix"."supplier";
create nickname inf_partsupp for infserver."informix"."partsupp";
create nickname inf_customer for infserver."informix"."customer";
create nickname inf_orders   for infserver."informix"."orders";
create nickname inf_lineitem for infserver."informix"."lineitem";
```

図 72. federated.sql (3/9)

```
-------------------------------------------------
-- Union views over federated nicknames
-------------------------------------------------
DROP VIEW part_fed;
CREATE  VIEW part_fed (
  p_partkey, p_mfgr, p_type, p_size, p_retailprice) AS
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM db2_part
UNION ALL
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM inf_part
UNION ALL
  SELECT p_partkey, p_mfgr, p_type, p_size,  p_retailprice
  FROM ora_part;

DROP VIEW partsupp_fed;
CREATE VIEW partsupp_fed (
   ps_partkey, ps_suppkey, ps_supplycost) AS
   SELECT ps_partkey, ps_suppkey, ps_supplycost
   FROM  db2_partsupp
UNION ALL
   SELECT ps_partkey, ps_suppkey, ps_supplycost
   FROM  inf_partsupp
UNION ALL
   SELECT ps_partkey, ps_suppkey, ps_supplycost
   FROM  ora_partsupp;

DROP VIEW supplier_fed;
CREATE VIEW supplier_fed (
   s_suppkey, s_name, s_address ) AS
   SELECT s_suppkey, s_name, s_address
   FROM  db2_supplier
UNION ALL
   SELECT s_suppkey, s_name, s_address
   FROM inf_supplier
UNION ALL
   SELECT s_suppkey, s_name, s_address
   FROM ora_supplier;
```

図 72. federated.sql (4/9)

```
-------------------------------------------------
-- Create Wrapper, Server and nickname for XML
-------------------------------------------------

drop wrapper xml_files;

create wrapper XML_files library 'db2lsxml.dll';
create server LOCAL_XML_FILES wrapper XML_FILES;
create nickname Employees_From_XML (
  doc Varchar(100) OPTIONS(DOCUMENT 'FILE'),
  Employee_Number Varchar(5) OPTIONS(XPATH './@SerialNum'),
  First_Name Varchar(50) OPTIONS(XPATH './/Firstname'),
  Middle_Initial Varchar(50) OPTIONS(XPATH './/Initial'),
  Last_Name Varchar(50) OPTIONS(XPATH './/Lastname'),
  Department_Number Varchar(50) OPTIONS(XPATH './/Department'),
  Phone_Number Varchar(50) OPTIONS(XPATH './/PhoneNumber'),
  Job Varchar(50) OPTIONS(XPATH './/Job'),
  Education_Level Varchar(50) OPTIONS(XPATH './/EDLevel'),
  Gender Varchar(50) OPTIONS(XPATH './/Sex'),
  Hire_Date Varchar(50) OPTIONS(XPATH './/HireDate'),
  Birth_Date Varchar(50) OPTIONS(XPATH './/BirthDate'),
  Annual_Salary Varchar(50) OPTIONS(XPATH './/Salary'),
  Annual_Bonus Varchar(50) OPTIONS(XPATH './/Bonus'),
  Commission Varchar(50) OPTIONS(XPATH './/Comm'),
  cid Varchar(16) OPTIONS(PRIMARY_KEY 'YES'))
 FOR SERVER LOCAL_XML_FILES
 OPTIONS (XPATH '/Employee');
```

図 72. federated.sql (5/9)

```
create nickname Employees_From_XML_Included (
  Employee_Number Varchar(5) OPTIONS(XPATH './@SerialNum'),
  First_Name Varchar(50) OPTIONS(XPATH './/Firstname'),
  Middle_Initial Varchar(50) OPTIONS(XPATH './/Initial'),
  Last_Name Varchar(50) OPTIONS(XPATH './/Lastname'),
  Department_Number Varchar(50) OPTIONS(XPATH './/Department'),
  Phone_Number Varchar(50) OPTIONS(XPATH './/PhoneNumber'),
  Job Varchar(50) OPTIONS(XPATH './/Job'),
  Education_Level Varchar(50) OPTIONS(XPATH './/EDLevel'),
  Gender Varchar(50) OPTIONS(XPATH './/Sex'),
  Hire_Date Varchar(50) OPTIONS(XPATH './/HireDate'),
  Birth_Date Varchar(50) OPTIONS(XPATH './/BirthDate'),
  Annual_Salary Varchar(50) OPTIONS(XPATH './/Salary'),
  Annual_Bonus Varchar(50) OPTIONS(XPATH './/Bonus'),
  Commission Varchar(50) OPTIONS(XPATH './/Comm'),
  cid Varchar(16) OPTIONS(PRIMARY_KEY 'YES'))
 FOR SERVER LOCAL_XML_FILES
 OPTIONS (FILE_PATH 'c:¥cdi_data_files¥CDI_Employees.xml', XPATH '//Employee');
```

図 72. federated.sql (6/9)

```
-------------------------------------------------
-- Create read functions and views over MQ queues for CR
-------------------------------------------------

CREATE FUNCTION NEW_SUPPLIERS_READ()
    RETURNS TABLE  ( SUPPLIER_NAME VARCHAR(80),
                          SUPPLIER_PHONE VARCHAR(12),
                          PART_KEY DOUBLE,
                          PART_PRICE DOUBLE,
                          MAN_DAYS DOUBLE,
                          MAX_QUANTITY DOUBLE,
     CORRELID VARCHAR(80))
    LANGUAGE SQL
    NOT DETERMINISTIC
    EXTERNAL ACTION
    READS SQL DATA
    RETURN
 SELECT
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',1),80),
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',2),12),
         DOUBLE(DB2MQ.GETCOL(T.MSG,',',3)),
         DEC(DB2MQ.GETCOL(T.MSG,',',4),8,4),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',5)),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',6)),
  CORRELID
 FROM TABLE (DB2MQ.MQREADALL('DB2.DEFAULT.SERVICE',
      'DB2.DEFAULT.POLICY')) AS T;

create view READ_NEW_SUPPLIERS_FROM_QUEUE
  as select * from table(new_suppliers_read()) t
  where CORRELID = 'CDI_NEW_SUPPLIER';
```

図 72. federated.sql (7/9)

```
-------------------------------------------------
-- Create destroy functions and views over MQ queues for CR
-------------------------------------------------
CREATE FUNCTION NEW_SUPPLIERS_REC()
    RETURNS TABLE  ( SUPPLIER_NAME VARCHAR(80),
                           SUPPLIER_PHONE VARCHAR(12),
                           PART_KEY DOUBLE,
                           PART_PRICE DOUBLE,
                           MAN_DAYS DOUBLE,
                           MAX_QUANTITY DOUBLE,
     CORRELID VARCHAR(80))
    LANGUAGE SQL
    NOT DETERMINISTIC
    EXTERNAL ACTION
    READS SQL DATA
    RETURN
 SELECT
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',1),80),
         VARCHAR(DB2MQ.GETCOL(T.MSG,',',2),12),
         DOUBLE(DB2MQ.GETCOL(T.MSG,',',3)),
         DEC(DB2MQ.GETCOL(T.MSG,',',4),8,4),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',5)),
         BIGINT(DB2MQ.GETCOL(T.MSG,',',6)),
  CORRELID
 FROM TABLE (DB2MQ.MQRECEIVEALL('DB2.DEFAULT.SERVICE',
  'DB2.DEFAULT.POLICY', 'CDI_NEW_SUPPLIER', 1)) AS T;

create view RECEIVE_NEW_SUPPLIERS_FROM_QUEUE
  as select * from table(new_suppliers_rec()) t;
```

図 72. federated.sql (8/9)

```
-------------------------------------------------
-- Create temporary tables used in processing
-------------------------------------------------

-- For running custom java programs
drop table aux_table;
create table aux_table (part_key integer,
  supplier_key int, supply_cost double);

-- For running XML composition
drop table UCustomers;
create table Ucustomers (x_doc DB2XML.XMLCLOB not logged);

-- For storing the web request
drop table request_bid;
drop table request_status;
create table request_bid (reqkey integer not null,
  partkey integer not null, bid double not null);
create table request_status (reqkey integer not null,
  partkey integer not null, suppkey integer not null,
  newquote double not null,currentquote double not null,
  status varchar(15) not null);

-- For shredding XML documents to DB2
IMPORT FROM c:¥CDI_Data_Files¥Setup¥CDI_Employees.ixf of
  IXF CREATE INTO EMPLOYEES_DB2;
CREATE TABLE EMPLOYEES_FROM_XML_FILE_SHRED LIKE EMPLOYEES_DB2;
```

図 72. federated.sql (9/9)

```
// Import all necessary classes
import java.lang.*;
import java.sql.*;
import java.util.*;

// USAGE: db, user, password, timeout

/**
 * Class Listener:
 * Class which will check a message queue for messages and
 *    call the "Director" stored procedure
 * to process the message.
 */
public class CDIListener{

 /**
  * Method checkQueue
  * Method to check the queue for messages.
    * If present calls the "Director" stored procedure
  *  otherwise it will wait for a user specified number of seconds
  */
```

図 73. CDIListener.java (1/12)

```
public void checkQueue(String db, String user,
          String pass, int timeout) {

  /* Local variables */
  String urlDB2 = null;
  Connection connDB2 = null;
  PreparedStatement stmtDB2 = null;
  PreparedStatement stmtDB2_2 = null;
  ResultSet rsDB2 = null;
  ResultSet rsDB2_2 = null;
  ResultSet rsTotal = null;
  String query = null;

  int mqMsgType  = 0;
  int mqMsgKey   = 0;
  int mqMsgPart  = 0;
  int mqMsgQuant = 0;
  double mqMsgPrice = 0;

  String comment = null;
  int intValue = 0;
  int numRecords = 0;

  System.out.println("¥nStarting listener execution");
  System.out.println("¥nConnecting to DB2");
```

図 73. CDIListener.java (2/12)

```
// Print out parms
System.out.println("¥nUsing startup parms of: ¥n Database: " +
     db + "¥n User: " + user + "¥n Password: **********¥n
     Sleep interval: " + timeout + " milliseconds");

try {
 //Load drivers' classes
 System.out.println("¥nLoading DB2 driver");
 Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();

 //URL for databases to be connected
 System.out.println("¥nSetting URL to database");
 urlDB2 = "jdbc:db2:" + db;

 //Get database connections
 System.out.println("¥nGetting DB2 connection");
 connDB2 = DriverManager.getConnection(urlDB2, user, pass);
} catch (Exception e) {
 e.printStackTrace();
}
```

図 73. CDIListener.java (3/12)

```
/*Issue a receive on the MQ queue*/
System.out.print("¥nStart MQ queue processing...");
System.out.flush();

double test = 0;

/* Loop forever to read the queue */
while (test == 0) {
 // System.out.println("¥nChecking queue...");
 try {
  stmtDB2 = connDB2.prepareStatement
            ("SELECT VARCHAR(DB2MQ.GETCOL(T.MSG,',',1),1),
              INT(DB2MQ.GETCOL(T.MSG,',',2)),
              INT(DB2MQ.GETCOL(T.MSG,',',3)),
              INT(DB2MQ.GETCOL(T.MSG,',',4)),
              DOUBLE(DB2MQ.GETCOL(T.MSG,',',5))
              FROM TABLE (DB2MQ.MQRECEIVEALL('DB2.DEFAULT.SERVICE',
              'DB2.DEFAULT.POLICY','CDI_IN_MSG',1)) AS T");
      rsDB2 = stmtDB2.executeQuery();
 }
 catch (SQLException e) {
  if (e.getErrorCode() == 100) {
   test = 0;
  }
  else {
   System.out.println(e);
   test = 1;
  }
 }
 catch (Exception e) {
  e.printStackTrace();
  test = 1;
 }
```

図 73. CDIListener.java (4/12)

```
try {
  // Get a message off the queue
 if (rsDB2 != null)
  while(rsDB2. next()) {
  // Get the type from the MQ message and
           //   call appropriate function
  mqMsgType  = rsDB2.getInt(1);

  // if END message, exit
  if (mqMsgType == 0) {
    System.out.println("¥nFound END message on queue");
    test = 1;
  }
  else {
   // Found a customer buy request message
   if (mqMsgType == 1) {
    // get other values from message
    mqMsgKey   = rsDB2.getInt(2);
    mqMsgPart  = rsDB2.getInt(3);
    System.out.println("¥nFound message type="+ mqMsgType + ";
                        customer="+ mqMsgKey + "; part=" + mqMsgPart);

    // Update the REQUEST_BID table
    System.out.print("Executing update to
                   local REQUEST_BID table...");
    // put customer order into REQUEST_BID table
                //    using max part price + 45% markup
    stmtDB2_2 = connDB2.prepareStatement
                    ("INSERT into request_bid
                      values (" + mqMsgKey + "," + mqMsgPart + "," + "
                      (SELECT MIN(ps_supplycost)*1.45
                          FROM partsupp_fed
                       WHERE ps_partkey = " + mqMsgPart + "))");
         stmtDB2_2.executeUpdate();
    System.out.println("¥nComplete");
    stmtDB2_2.close();
   }
```

図 73. CDIListener.java (5/12)

```
   else {
    // Else if it's a supplier price update
    if (mqMsgType == 2) {
     // Get the other values off the queue
     mqMsgKey   = rsDB2.getInt(2);
     mqMsgPart  = rsDB2.getInt(3);
     mqMsgQuant = rsDB2.getInt(4);
     mqMsgPrice = rsDB2.getDouble(5);
     System.out.println("¥nFound message
                         type="+ mqMsgType + ";
                         supplier="+ mqMsgKey + ";
                         part=" + mqMsgPart + ";
                         price=" + mqMsgPrice);
```

図 73. CDIListener.java (6/12)

```
// Check if this supplier has supplier this part before
System.out.print("¥nChecking if supplier
                       already supplies part...");
stmtDB2 = connDB2.prepareStatement
                      ("SELECT COUNT(*)
                      FROM partsupp_fed
                      WHERE ps_partkey = " + mqMsgPart + "
                     and ps_suppkey = " + mqMsgKey);
  rsDB2_2 = stmtDB2.executeQuery();
  rsDB2_2.next();
  intValue = rsDB2_2.getInt(1);
System.out.print(intValue + "...");
```

図 73. CDIListener.java (7/12)

```
  // If supplier has supplier before
  if (intValue > 0) {

   // Get current price by supplier
 System.out.print("Yes!¥nGetting current
                        minimum price...");
   stmtDB2 = connDB2.prepareStatement
                   ("SELECT MIN(ps_supplycost)
                     FROM partsupp_fed
                     WHERE ps_partkey = " + mqMsgPart + "
                    and ps_suppkey = " + mqMsgKey);
   rsDB2_2 = stmtDB2.executeQuery();
   rsDB2_2.next();
   intValue = rsDB2_2.getInt(1);

   System.out.println("Current price = " + intValue + "¥n");
```

図 73. CDIListener.java (8/12)

```
   //If new price less than or equal to
               //existing price, update database
               //and mark as accepted
   if (mqMsgPrice <= intValue) {
  comment = "'ACCEPT'";
   }
   // Else update the database but mark as review
   else {
  comment = "'REVIEW'";
   }
 System.out.print("¥nExecuting " + comment + "
                     update to federated db2 table db2_partsupp...");
   stmtDB2 = connDB2.prepareStatement
                  ("UPDATE db2_partsupp
                    set ps_availqty = " + mqMsgQuant + ",
                    ps_supplycost = " + mqMsgPrice + "
                    where ps_partkey = " +  mqMsgPart + "
                    and ps_suppkey = " + mqMsgKey);
   stmtDB2.executeUpdate();
 System.out.println("Complete¥n");
  }
```

図 73. CDIListener.java (9/12)

```
// Else this is a new supplier for this part
else {
 System.out.print("No!¥nExecuting 'NEW' insert to
                    federated db2 table db2_partsupp...");
 comment = "'NEW'";

 // Add new record to database
 stmtDB2 = connDB2.prepareStatement
                   ("INSERT into db2_partsupp
                       values (" + mqMsgPart + "," +
                       mqMsgKey + ","+ mqMsgQuant + "," +
                       mqMsgPrice + ",'New supplier added at: " +
                       new java.util.Date()+ " ')");
 stmtDB2.executeUpdate();
 System.out.println("Complete¥n");
}
// Update the local table
System.out.print("¥nUpdating local REQUEST_STATUS table...");
  stmtDB2_2 = connDB2.prepareStatement
                 ("INSERT into request_status
                     values (" + mqMsgKey + "," +
                     mqMsgPart + "," +mqMsgKey + ","+
                     mqMsgPrice + "," +
                     intValue + "," + comment + ")");
  stmtDB2_2.executeUpdate();
System.out.println("Complete¥n");
```

図 73. CDIListener.java (10/12)

```
     if (stmtDB2_2 != null)
      stmtDB2_2.close();
     if (rsDB2_2 != null)
      rsDB2_2.close();
    }
    else {
     System.out.println("¥nError - unknown message type");
    }
   }
  }
 System.out.flush();
 } // End secondary while
 System.out.flush();
} // End try

catch (Exception e) {
 e.printStackTrace();
}

// Sleep for the necessary time
try {
 if (rsDB2 != null)
  rsDB2.close();
 if (stmtDB2 != null)
  stmtDB2.close();
 Thread.sleep(timeout);
 System.out.print(".");
 System.out.flush();
}
catch (Exception e) {
 e.printStackTrace();
}
} // end Main while
```

図 73. CDIListener.java (11/12)

```
 System.out.println("Listener stopped¥n");
 System.out.println("¥nListener stopped¥n");

 /* Close all database connections */
 try {
  rsDB2.close();
  stmtDB2.close();
  connDB2.close();
 } catch (Exception e) {
  e.printStackTrace();
 }

 return;

}

/* Main program to invoke listener */
public static void main(String argv[]) {

 String DBName = argv[0];
 String DBUser = argv[1];
 String DBPass = argv[2];
 String TimeCk = argv[3];

 CDIListener dp = new CDIListener();
 int timeout = Integer.parseInt(TimeCk)*1000;
 dp.checkQueue(DBName, DBUser, DBPass, timeout);
}
}
```

図 73. CDIListener.java (12/12)

```
/*
 * @(#)MessageFormatter.java
 *
 * CopyrightVersion 1.0
 *
 */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.servlet.ServletException;
import proxy.soap.*;

public class MessageFormatter extends javax.servlet.http.HttpServlet {

 final static String htmlHeader1 = "<HTML><TITLE>MessageFormatter</TITLE>";
final static String message1 =
 "<p align=¥"center¥">
  <font color=¥"navy¥"
  face=¥"verdana¥" size=¥"+2¥">
<b>Thank You</b></font></p>
<p><font color=¥"black¥"
face=¥"veranda¥" size=¥"+1¥">";
```

図 74. MessageFormatter.java (1/7)

```
final static String message2 = "";

 final static String htmlHeader2 = "</HTML>";
 static String quote = "";

 static Connection con = null;
 final static String url = "jdbc:db2:demo";

  // method to generate a random REQUEST number
 public int randomint() {
  double first = java.lang.Math.random();
  String help = java.lang.Double.toString(first);
  help = help.substring(3,7);
  int number = Integer.parseInt(help);
  number = number / 5;
  return number;
 }
```

図 74. MessageFormatter.java (2/7)

```
// Overwriting doGet method to handle Http GET request

 public void doGet(HttpServletRequest req,
         HttpServletResponse res)
  throws javax.servlet.ServletException, java.io.IOException {
  try {


   PrintWriter pr = res.getWriter();
   pr.print(htmlHeader1);
   pr.print(message1);
   pr.print(message2);
   String method = "";
   String part = "";
   String price = "";
   String quantity = "";
   String key = "";
   String cust_name = "";
   String col_name1 = null;
   String col_name2 = null;
   String tab_name  = null;

   Connection con = null;
   String url = null;
   Statement stmt = null;
   ResultSet rs = null;

   WSProxy WSid = new WSProxy();
   boolean fCustomer = true;
```

図 74. MessageFormatter.java (3/7)

```
if (req.getParameter("method")!=null) {
 // Get the common parms to the servlet from the REQ object
 key = req.getParameter("name");
 part = req.getParameter("part");
 method = req.getParameter("method");

 // If customer order
 if (method.equals("orderNewParts")) {
  fCustomer = true;
  // Get the quantity the customer is ordering
  quantity = req.getParameter("quantity");

  // set the table and column names for SELECT
  col_name1 = "c_name";
  col_name2 = "c_custkey";
  tab_name  = "db2_customer";

 }
 else {
  // Else if supplier update
  if (method.equals("setSupplierQuotes")) {
   fCustomer = false;
   // Get the price the supplier is updating
   price = req.getParameter("price");

   // set the table and column names for SELECT
   col_name1 = "s_name";
   col_name2 = "s_suppkey";
   tab_name  = "supplier_fed";
  }
  else {
   pr.println
               ("method = "+req.getParameter("method")+
                    " Not supported! <br>");
   return;
  }
 }
```

図 74. MessageFormatter.java (4/7)

```
 // Get the real customer name from federated data source
 try {
  Class.forName
            ("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
  url = "jdbc:db2:demo";
  con = DriverManager.getConnection(url,"demo","xxxxx");
  stmt = con.createStatement();
  rs = stmt.executeQuery
            ("SELECT " + col_name1 + " from " +
              tab_name + " where " + col_name2 + " = " + key);
  while (rs.next()) {
   cust_name = rs.getString(1);
  }

  rs.close();
  stmt.close();
  con.close();
 }
 catch (Exception e) {
  pr.println(e);
  System.out.println(e);
  return;
 }


}
else {
 pr.println
          ("*** Severe error occurred-no method passed ***>");
 return;
}
```

図 74. MessageFormatter.java (5/7)

```
   // Write request status back to user
   try{
    // If this is a supplier update request
    if (!fCustomer) {
      String message1id=  "2,"
               + key + "," + part + ","
               + quantity + "," + price ;
            java.lang.String message1idTemp  = message1id;
     System.out.println
                ("message type 2 being written: "
                   + message1idTemp);
            org.tempuri.worftestweb
                .demo.newdadx.dadx.xsd
                     .Stmt1ResultElement
                     mtemp = WSid.stmt1(message1idTemp);
            if (mtemp == null)
      pr.println("*** Web Service error occurred ***>");
     else {
      pr.println
("<table border=¥"1¥"
bordercolor=¥"navy¥"
width=¥"100%¥">
<tr align+¥"left¥">
<td>Supplier</td>
<td>" + cust_name + "</td>
</tr>
<tr align+¥"left¥"><td>Part</td><td>" + part + "</td></tr>
<tr align+¥"left¥"><td>Price</td><td>" + price + "</td></tr>
</table>");
pr.println("<br><font color=¥"red¥" face=¥"verdana¥"
size=¥"-1¥">
Price update submitted for processing");
     }
    }
```

図 74. MessageFormatter.java (6/7)

```
    // else must be a customer buy request
    else {
     String message1id=  "1," + key +"," + part + ",0, 0";
           String message1idTemp  = message1id;
     System.out.println("message type 1 being written: " + message1idTemp);
           org.tempuri.worftestweb.demo
                    .newdadx.dadx.xsd
                    .Stmt1ResultElement
                    mtemp = WSid.stmt1(message1idTemp);
           if (mtemp == null)
      pr.println("*** Web Service error occurred ***>");
     else {
      pr.println
                  ("<table border=¥"1" bordercolor=¥"navy¥"
wide=¥"100%¥"><tr align+¥"left¥"><td>Customer</td>
<td>" + customer_name + "</td></tr>
<tr align+¥"left¥"><td>Part</td>
<td>" + part + "</td></tr>
<tr align+¥"left¥"><td>Price</td>
<td>" + quantity + "</td></tr></table>");
       pr.println("<br><font color=¥"red¥"
face=¥"verdana¥"
size=¥"-1¥">
Order submitted for processing");
     }
    }
   }
   catch (Exception e) {
    System.out.println(e);
    pr.println(e);
    return;
   }

  }
  catch (Exception e) {
  }
 }
}
```

図 74. MessageFormatter.java (7/7)

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:¥dxx¥dtd¥dad.dtd">
<DAD>
   <dtdid>resume.dtd</dtdid>
   <validation>NO</validation>
   <Xcolumn>
     <table name="resume_skills_sidetable">
       <column name="skill"
         type="varchar(20)"
         path="/resume/skill"
         multi_occurrence="YES">
       </column>
     </table>
   </Xcolumn>
</DAD>
```

図 75. resume.dad

```
<?xml version="1.0"?>
<!DOCTYPE Employees SYSTEM "employees.dtd">
<Employees>
  <Employee SerialNum="12">
    <Firstname>Laurie</Firstname>
    <Lastname>Douglas</Lastname>
    <Job_Description>
       DB2 engine Development
   </Job_Description>
  </Employee>
  <Employee SerialNum="13">
    <Firstname>John</Firstname>
    <Lastname>Smith</Lastname>
    <Job_Description>
       Information Integration Technology Solutions
   </Job_Description>
  </Employee>
  <Employee SerialNum="14">
    <Firstname>George</Firstname>
    <Lastname>Jackson</Lastname>
    <Job_Description>
        Customer contact
   </Job_Description>
  </Employee>
</Employees>
```

図 76. All_Employees.xml

関連概念:

- 14 ページの『Cottonwood Distributors, Inc. — ウェアハウスの例』
- 15 ページの『データの発見 — 従業員のスキルのシナリオ』

付録 B. DADX 環境チェッカー

DADX 環境チェッカーは、WORF で Web サービスを作成し実行するときに使用する NST、DAD、および DADX ファイルに対して、種々の構文チェックおよびセマンティック・チェックを実行します。 DADX 環境チェッカーを使用することにより、 WORF で Web サービスをデプロイするときに生じるエラーの数を最小限に抑えます。

DADX 環境チェッカーのインストール

DADX 環境チェッカーは、コマンド行から呼び出される Java アプリケーションです。呼び出されると、エラー、警告、および成功の標識を含む出力ファイルが生成されます。出力テキスト・ファイルの名前は、ユーザーが定義します。名前が指定されない場合、標準出力が使用されます。

DADX 環境チェッカーは、WORF インストールの *tools¥lib* サブディレクトリーにあります。このツールのコードを含む JAR ファイルは、CheckersCommon.jar と DADXEnvChecker.jar です。システムに JRE または JDK バージョン 1.3.1 以上がインストールされていることを確認してください。 CLASSPATH を更新して、以下のアーカイブすべてを含めてください。

- WORF がインストールされている *tools¥lib* ディレクトリーにある、*CheckersCommon.jar*、*DADXEnvChecker.jar*、および *worf.jar*。
- *xerces.jar*。UNIX および Windows の場合、これらのファイルは、http://xml.apache.org/ でダウンロード可能な、 Xerces-J 2.0.2 のバイナリー配布に含まれています。 OS/390 および z/OS の場合、これらのファイルは、PTF UW95866 を適用した IBM XML Toolkit Version 1 Release 4 に含まれています。
- http://xml.apache.org でダウンロード可能な SOAP 2.3 のバイナリー配布か、WebSphere Application Server のインストール物に含まれる *soap.jar*。
- *j2ee.jar* バージョン 1.3 以降。このファイルは java.sun.com からダウンロードできます。
- *qname.jar*。このファイルは java.sun.com からダウンロードできます。
- *wsdl4j.jar*。このファイルは http://oss.software.ibm.com/developerworks/projects/wsdl4j からダウンロードできます。
- http://java.sun.com でダウンロード可能な JavaBeans Activation Framework 1.0.1 のバイナリー配布に含まれる *activation.jar*。
- http://java.sun.com でダウンロード可能な JavaMail 1.2 のバイナリー配布に含まれる *mail.jar*。
- WebSphere Application Server か、http://www.apache.org/ でダウンロード可能な Jakarta Tomcat Version 3.2.x から 4.0.3 以降の配布に含まれる *servlet.jar*。
- UNIX および Windows の場合: DB2 Universal Database をインストールした場所の /java ディレクトリーにある *db2java.zip*。 OS/390 および z/OS の場合: DB2 Universal Database を HFS でインストールしたときの classes/ サブディレクトリ

ーにある *db2j2classes.zip*。 *jcc.jar* も使用できます。 *group.properties* ファイルの dbDriver パラメーターは使用するドライバー・パッケージを決定します。

たとえば、Windows 環境で実行している場合、CLASSPATH を設定して以下のファイルを見つける必要があります。

```
CheckersCommon.jar;
DADXEnvChecker.jar;
worf.jar;
xerces.jar;
j2ee.jar
qname.jarwsdl4j.jar
soap.jar;
db2java.zip;
```

関連概念:

- 37 ページの『DADX ファイルの定義』

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』
- 264 ページの『DADX 環境チェッカーの実行』

DADX 環境チェッカーの実行

DADX 環境チェッカーは、Java プログラムです。 JDK バージョン 1.3.1 以上で実行できます。 DADX 環境チェッカーを実行する場合、次のコマンドを (1 行で) 実行します。

```
java com.ibm.etools.webservice.util.Check_install
   [-srv] [-schdir pathToSchemasDir]
   [-sch schemaLocations] [-out outputFile]
  fileToCheck
```

たとえば、*dxxworf.zip* を *c:¥dxxworf* ディレクトリーで解凍した場合、次のように入力して、c:¥tomcat¥webapps¥services に含まれるリソース・ファイルで DADX チェッカーを実行してから、出力を現行ディレクトリーの *myOutputFile.txt* に送信します。

```
java com.ibm.etools.webservice.util.Check_install
-srv -schdir c:¥dxxworf¥schemas
-out myOutputFile.txt c:¥tomcat¥webapps¥services
```

パラメーター

以下は、DADX 環境チェッカーの実行に使用できるパラメーターです。

-schdir *pathToSchemasDir*
: NST および DADX ファイルをチェックするのに使用されるスキーマが保管されるディレクトリーへの絶対パスを指定します。

-sch *schemaLocations*
: ファイルをチェックするときにパーサーが使用するスキーマのリストを指定します。 DADX チェッカーを使用すると、Xerces パーサーのプロパティーの値を指定できます。このプロパティーは、構文解析するファイルの妥当性検査を実行するときに必要な XML スキーマのロケーションを指定するのに使用できます。スキーマのロケーションを指定するときには、スキーマの

ターゲット・ネーム・スペースの名前 (たとえば、 *http://myschema*) を指定し、その後にスキーマの実際のロケーションを指定します。これは、ファイル・システムのパス (たとえば、*c:¥dir¥schema1.xsd*) か、有効な URL になります。ただし、XML 文書そのものに、スキーマのロケーションの宣言を含めることができます。この情報を提供するときには、XML 文書で schemaLocation 属性が使用されます。次に示すのは、XML 文書の先頭部分の一例です。

```
<purchaseReport
  xmlns="http://www.example.com/Report"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.com/Report
http://www.example.com/Report.xsd">
```

特定のネーム・スペースの場合、同じネーム・スペースに対して schemaLocation 属性が別のスキーマのロケーションを定義したとしても、パーサーは、パーサーのプロパティーを使用して定義したスキーマのロケーションを使用します。 schemaLocations の構文は、インスタンス文書の schemaLocation 属性の場合と同じです。たとえば、 *http://www.example.com file_name.xsd* のようになります。ユーザーは、複数の XML スキーマを指定できます。たとえば、-sch *http://www.example_1.com file_name_1.xsd http://www.example_2.com file_name_2.xsd* のようになります。

-out *outputFile*
出力テキスト・ファイルの名前を指定します。省略する場合、標準出力が使用されます。

-srv *fileToCheck* として渡された Web サービス・モジュール・ディレクトリー (たとえば、 c:¥tomcat¥webapps¥services) の下にあるすべての NST、DAD、および DADX ファイルに対して、チェックを実行しなければならないことを示します。このオプションを使用しない場合、チェックは、チェックするファイルとして渡される DADX ファイルと、他のリソース・ファイルに含まれる関連データに対してのみ実行されます。たとえば、この DADX ファイルで言及されている DAD ファイルがチェックされてから、これらの DAD ファイルで言及されている DTDID が NST ファイルでチェックされます。さらに、DADX ファイルに関連したデータだけが、NST ファイルおよび web.xml ファイルでチェックされます。

fileToCheck
パラメーター *-srv* を使用しない場合、 *fileToCheck* の値は、チェックされる DADX ファイルになります。パラメーター *-srv* を使用する場合、 *fileToCheck* 値は、 Web サービス・モジュールのルート・ディレクトリーになります。たとえば、services.war モジュールでは、 *services* として unzip された .war ファイルのルート・ディレクトリーです。

-help コマンド行のオプション情報を表示します。

-version
バージョン情報を表示します。

サンプル・ファイル

サンプル・ファイルは、dxxworf.zip の *tools¥samples* ディレクトリーにあります。 *DADXEnvChecker_sample.txt* は、Web サービス・モジュールで実行したチェックの

結果を示す出力テキスト・ファイルです。 DADX 環境チェッカーが、このファイルを生成します。チェッカーは、-out パラメーターで指定したファイル名 *DADXEnvChecker_sample.txt* を使用します。

関連概念:

- 37 ページの『DADX ファイルの定義』

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』
- 263 ページの『DADX 環境チェッカーのインストール』

出力テキスト・ファイルでのエラーおよび警告の表示

-srv パラメーターを使用する場合、エラー、警告、および成功の標識が、段落でグループ化されます。各段落は、チェックしたファイルに関連付けられています。各ファイルをチェックした結果は、ファイル名を指定した場合は出力ファイルに、ファイル名を指定しない場合は標準出力装置に表されます。

段落は、**groups** ディレクトリーで、パスかサブディレクトリーに従ってグループ化されます。次に示すのは、出力テキスト・ファイルの抜粋で、ここには、グループ /groups/dxx_sales_db に属するファイル sales_db.nst および getstart_xcollection.dad で実行されたチェックに対応するエラー・メッセージが示されています。

```
## Checking group: c:¥tomcat¥webapps¥services¥WEB-
INF¥classes¥groups¥dxx_sales_db
## Checking NST file: c:¥tomcat¥webapps¥services¥WEB-
INF¥classes¥groups¥dxx_sales_db¥sales_db.nst
INFO. Line 5: file "c:¥dxx¥samples¥dtd¥getstart.dtd" is accessible.
ERROR. Line 12: file "wrongDtd.dtd" CANNOT be found
either in the file system or in the database.
INFO. Line 8: file "getstart.dtd" is accessible.
## Checking DAD file: c:¥tomcat¥webapps¥services¥WEB-
INF¥classes¥groups¥dxx_sales_db¥getstart_xcollection.dad
WARNING. Line 4: DTDID "dtd_.dtd" CANNOT be found in the DTD_REF table.
INFO. Line 9: the DTDID "c:¥dxx¥samples¥dtd¥getstart.dtd"
has been declared in the NST file.
```

エラー、警告、および成功のメッセージは、エラー、警告、または成功のイベントが特定の行に関連していれば、特定の行番号で開始できます。出力テキストの行番号は、メッセージに関連付けられたチェック・エレメントが検出された、ファイル内の行番号を示します。段落内の出力に関連付けられた順序はありません。

関連資料:

- 270 ページの『DAD ファイルのエラーのチェック』
- 271 ページの『DADX ファイルのエラーのチェック』
- 267 ページの『DADX 環境チェッカーによるエラー・チェック』
- 263 ページの『DADX 環境チェッカーのインストール』
- 264 ページの『DADX 環境チェッカーの実行』
- 269 ページの『NST ファイルでのエラーの検査』
- 268 ページの『web.xml ファイルのエラーのチェック』

DADX 環境チェッカーによるエラー・チェック

-srv パラメーターを使用して DADX 環境チェッカーを呼び出す場合、 WEB-INF ディレクトリー内の web.xml ファイルに対して最初のチェックが実行されます。次に、DADX 環境チェッカーは、WEB-INF¥classes¥groups ディレクトリーの各グループ・ディレクトリーにある、以下のタイプのファイルに対してチェックを実行します。

- NST ファイル
- DAD ファイル
- DADX ファイル

-srv パラメーターを使用しないで DADX 環境チェッカーを呼び出す場合、チェック対象のファイルとして渡される DADX ファイルに対して最初のチェックが実行されます。次に、DADX 環境チェッカーは、この DADX ファイルで言及されている DAD ファイルをチェックします。さらに、DADX ファイルが属するグループの NST ファイルに対してチェックを実行します。最後に、DADX 環境チェッカーは、DADX ファイルを含む WEB-INF ディレクトリー内の web.xml ファイルをチェックします。

データベース・エラー・メッセージ

NST および DADX ファイルに対するいくつかのチェックでは、DADX 環境チェッカーは以下のアクションを実行します。

1. group.properties ファイルに含まれているデータを使用して、データベースへの接続の確立を試みる
2. グループが関連付けられているデータベースを照会する
3. エラーがないかどうかグループのファイルをチェックする

データベースへの接続が失敗した場合、DADX 環境チェッカーはエラー・メッセージを発行します。以下の例は典型的なエラー・メッセージを示しています。

```
Checking group: c:¥test¥jakarta-tomcat-3.2.2
##Checking group: c:¥tomcat¥webapps¥services
¥WEB-INF¥classes¥groups¥dxx_travel
WARNING. Connection error [IBM][CLI Driver]
SQL1013N The database alias name or database name
"TRAVELLL" could not be found.
SQLSTATE=42705
```

関連タスク:

- 76 ページの『group.properties ファイルのカスタマイズ』
- 70 ページの『web.xml ファイルおよび group.properties ファイルの定義』

関連資料:

- 270 ページの『DAD ファイルのエラーのチェック』
- 271 ページの『DADX ファイルのエラーのチェック』
- 269 ページの『NST ファイルでのエラーの検査』
- 268 ページの『web.xml ファイルのエラーのチェック』

web.xml ファイルのエラーのチェック

DADX 環境チェッカーは、Web サービス・モジュールのルート・ディレクトリー (この例では、 services) の下にある **WEB-INF¥web.xml** ファイルをチェックします。

次に示すのは web.xml ファイルの抜粋です。

```
<servlet>
<servlet-name>dxx_sales_db</servlet-name>
<servlet-class>com.ibm.etools.webservice.rt.dxx.servlet.DxxInvoker
        </servlet-class>
        <init-param>
          <param-name>faultListener</param-name>
<param-value>org.apache.soap.server.DOMFaultListener
          </param-value>
        </init-param>
        <load-on-startup>-1</load-on-startup>
    </servlet>
<servlet-mapping>
<servlet-name>dxx_sales_db</servlet-name>
<url-pattern>/sales/*</url-pattern>
    </servlet-mapping>
```

<servlet-class> タグは、<servlet> タグの直接の子タグで、値は com.ibm.etools.webservice.rt.isd.servlet.IsdInvoker か、com.ibm.etools.webservice.rt.dxx.servlet.DxxInvoker のいずれかになります。値が異なるときに、チェッカーはエラー・メッセージを示します。次の例は、web.xml 文書の <servlet-class> タグに対して実行されたチェックの結果を示しています。

```
INFO. Line 21: servlet class for
servlet "dxx_sales_db" is a correct servlet class.
ERROR. Line 31: servlet class
"com.ibm.etools.webservice.rt.dxx.servlet.OtherInvoker"
for servlet "dxx_sample"
is NOT a correct servlet class.
INFO. Line 41: servlet class
for servlet "dxx_travel"
is a correct servlet class.
```

各 <servlet-mapping> タグには、<servlet-name> タグが含まれますが、その値は、<servlet> タグの <servlet-name> タグの値と同じでなければなりません。そうでない場合、チェッカーは、次の例に示されているようなエラー・メッセージを示します。

```
ERROR. There is no <servlet>
tag declaring servlet
"isd_demos" mapped at line 50.
```

それ以外の場合、各 <servlet> タグには、同じサーブレット名が指定された、対応する <servlet-mapping> タグが必要です。 <servlet> タグに対応する <servlet-mapping> タグがない場合、チェッカーは、以下の種類のメッセージを示します。

```
ERROR. There is no
<servlet-mapping> tag
for servlet "dxx_travel" declared
at line 40.
```

各 <servlet-mapping> タグには、<url-pattern> タグも含まれ、その値としてユニークな値が指定されている必要があります。 2 つの <url-pattern> タグに同じ値が入っていれば、チェッカーは、次の例に示されているようなエラー・メッセージを示します。

```
ERROR. Line 56: "/sales/*" is already
declared as the URL pattern for servlet "isd_demos"
(see line 50).
```

関連タスク:

- 70 ページの『web.xml ファイルおよび group.properties ファイルの定義』

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』

NST ファイルでのエラーの検査

各グループ・ディレクトリーには、NST ファイルがあります。 NST ファイルは、グループのネーム・スペース表を宣言します。ここには、DTD ID 間の対応と、DTD から自動的に生成される XML スキーマのネーム・スペースとロケーションが含まれます。

次に示すのは NST ファイルの抜粋です。

```
<namespaceTable
xmlns="http://schemas.ibm.com/db2/dxx/nst">
<mapping dtdid="c:¥dxx¥samples¥dtd¥getstart.dtd"
      namespace="http://schemas.ibm.com/db2/dxx/samples/dtd/getstart.dtd"
    location="/dxx/samples/dtd/getstart.dtd/XSD"/>
    <mapping dtdid="getstart.dtd"
      namespace="http://schemas.myco.com/sales/getstart.dtd"
    location="/getstart.dtd/XSD"/>
```

DADX 環境チェッカーは、**nst.xsd** の正しいスキーマで、まず NST ファイルを検証します。これは、チェッカーによって報告された検証エラーの一例です。

```
ERROR. Validation error, in
"file:///c:/tomcat/webapps/services/WEB-
INF/classes/groups/dxx_sales_db/sales_db.nst",
line 8, column 35. cvc-complex-type.2.4.a:
Invalid content starting with element 'mappin'.
The content must match
'("http://schemas.ibm.com/db2/dxx/nst":mapping){0-UNBOUNDED}'.
ERROR. Validation error, in
"file:///c:/tomcat/webapps/services/WEB-
INF/classes/groups/dxx_sales_db/sales_db.nst",
line 17, column 32. cvc-complex-type.4: Attribute 'dtdid'
must appear on element 'mapping'.
ERROR. Validation error, in
"file:///c:/tomcat/webapps/services/WEB-
INF/classes/groups/dxx_sales_db/sales_db.nst",
line 17, column 32. Duplicate unique value
[ID Value: /order.dtd/XSD] declared for identity constraint
of element "namespaceTable".
```

最後に、チェッカーは、<mapping> エレメントの dtdid 属性が以下のいずれであるかチェックします。

- ファイル・システムで正確なパス
- db2xml.DTD_REF 表の列 DTDID に保管されている値

次の例は、NST ファイルの <mapping> エレメントに対するチェックの結果を示しています。

```
INFO. Line 5: file
"c:¥dxx¥samples¥dtd¥getstart.dtd" is accessible.
ERROR. Line 14: file
"wrongDtd.dtd" CANNOT be found either in
the file system or in the database.
```

関連資料:

- 270 ページの『DAD ファイルのエラーのチェック』
- 271 ページの『DADX ファイルのエラーのチェック』
- 267 ページの『DADX 環境チェッカーによるエラー・チェック』
- 263 ページの『DADX 環境チェッカーのインストール』
- 264 ページの『DADX 環境チェッカーの実行』
- 268 ページの『web.xml ファイルのエラーのチェック』

DAD ファイルのエラーのチェック

Document Access Definition (DAD) ファイルは、DB2 XML エクステンダーでサポートされる XML ファイルです。 DAD は、XML columns および XML collections という 2 つの選択的なアクセスおよび保管メソッドを使用して、 XML 文書を DB2 Universal Database 表に関連付けます。

次の例は、DAD ファイルの先頭部分を示しています。

```
<?xml
version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:¥dxx¥dtd¥dad.dtd">
<DAD>
<dtdid>c:¥dxx¥samples¥dtd¥getstart.dtd</dtdid>
<validation>NO</validation>
<Xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM
 "c:¥dxx¥samples¥dtd¥getstart.dtd"
</doctype>
<root_node>
<element_node name="Order">
...
```

DADX 環境チェッカーは、DTD **dad.dtd** に従って、まず DAD ファイルが有効かどうかをチェックします。 DAD の DOCTYPE 宣言で指定された dad.dtd へのパスが正しいことを確認する必要があります。

次に、チェッカーは、<dtdid> タグの値を入手します (存在する場合)。このタグの値が db2xml.DTD_REF 表の列 DTDID に保管されている値と一致しない場合、チェッカーは警告を発行します。 DAD の <validation> タグに *YES* の値が含まれる場合、チェッカーは、次のようなエラー・メッセージを発行します。

```
## Checking DAD file:
c:¥tomcat¥webapps¥services¥WEB-INF
¥classes¥groups¥dxx_sales_db¥order.dad
ERROR. Line 4: DTDID "wrongDtd.dtd"
CANNOT be found in the DTD_REF table.
```

次に、チェッカーは、DAD ファイルが Xcollection と Xcolumn のどちらを宣言しているか判別します。 Xcollection を宣言している場合、<doctype> エレメントで指定された DTD が取り出されます。 DADX 環境チェッカーは、この DTD が NST ファイルで宣言されていることをチェックします。

次の例は、同じグループに属する Xcolumn および Xcollection DAD のチェック結果を示しています。

```
## Checking DAD file:
c:¥tomcat¥webapps¥services¥WEB-
INF¥classes¥groups¥dxx_sales_db¥getstart_xcolumn.dad
INFO. Line 4: DTDID "getstart.dtd" was found in the DTD_REF table.

## Checking DAD file: c:¥tomcat¥webapps¥services¥WEB-
INF¥classes¥groups¥dxx_sales_db¥order-public.dad
INFO. Line 4: DTDID "order.dtd" was found in the DTD_REF table.
ERROR. Line 8: the DTDID "order.dtd" has NOT been
declared in the NST file.
```

さらに、DAD チェッカーを使用して、DAD ファイルに対する他のチェックを実行することも可能です。 DAD チェッカーは、やはり dxxworf.zip の *tools¥lib* ディレクトリーにある独立したツールです。詳細は、WebSphere アプリケーション開発の Web サイトで、 DAD チェッカー・ツールについての資料を参照してください。

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』
- 145 ページの『Web サービスのサンプル - PartOrders.dadx』

DADX ファイルのエラーのチェック

Document Access Definition Extension (DADX) は、データベースにアクセスする Web サービスを迅速に作成するためのテクノロジーです。 DADX では、標準の SQL ステートメント SELECT、INSERT、UPDATE、DELETE、および CALL、そして DB2 XML エクステンダーのストアード・プロシージャーを使用して、 Web サービスの操作を定義します。

次に示すのは DADX ファイルの抜粋です。

```
<?xml version="1.0"?>
<DADX xmlns="http://schemas.ibm.com/db2/dxx/dadx"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<operation name="find">
<documentation >
Returns the parts from order #1 with price > 20000.
    </documentation>
<retrieveXML>
<DAD_ref>getstart_xcollection.dad</DAD_ref>
<no_override/>
</retrieveXML>
</operation>
<operation name="findByMinPrice">
<retrieveXML>
<collection_name>
getstart_xcollection.dad
</collection_name>
<no_override/>
```

```
<parameter name="minprice"
type="xsd:decrimal"/>
</retrieveXML>
</operation>
```

DADX 環境チェッカーは、スキーマである dadx.xsd に従って、まず DADX ファイルを検証します。次に、チェッカーは <DAD_REF> または <collection_name> タグの値を入手し、それらのタグの値が以下のとおりであるかチェックします。

- <DAD_REF> タグの場合、ファイル・システムの DAD ファイルへの正しいパスかどうか
- <collection_name> の場合、使用可能な集合の名前。これは、表 db2xml.xml_usage の列 COL_NAME に保管された値です。

次の例は、DADX ファイルで実行されたチェックの結果を示しています。

```
## Checking DADX file: c:¥tomcat¥webapps¥services¥WEB-
INF¥classes¥groups¥dxx_sales_db¥PartOrders.dadx
ERROR. Validation error, in "file:///c:/tomcat/webapps/services/WEB-
INF/classes/groups/dxx_sales_db/PartOrders.dadx",
line 8, column 67. cvc-complex-type.2.4.c:
The matching wildcard is strict, but no declaration
can be found for element 'as'.
INFO. Line 16: for operation "find",
DAD "getstart_xcollection.dad" was found.
ERROR. Line 26: for operation "findAll",
DAD "non_existing_dad.dad" was NOT found.
INFO. Line 44: for operation "findByColor",
DAD "getstart_xcollection.dad" was found.
INFO. Line 65: for operation "findByMinPrice",
DAD "getstart_xcollection.dad" was found.
```

<operation> タグに、子として <DAD_REF> または <collection_name> タグがない場合、次の例に示されているように、チェッカーはこの操作に対してチェックが実行されなかったことを示すメッセージを発行します。

```
##
Checking DADX file: c:¥tomcat¥webapps¥services¥WEB-
INF¥classes¥groups¥dxx_sample¥HelloSample.dadx
INFO. Line 10: no <DAD_ref> or <collection_name>
elements to check for operation "listDepartments".
```

さらに、DADX 環境チェッカーは、WORF が DADX ファイルで宣言されたパラメーターの並列化機構を検出できるかどうかもチェックします。並列化機構は、ネットワーク接続を介して受け取った XML メッセージを、指定した変数またはオブジェクトに再構成します。すべての <parameter> タグで、タイプ属性の値は、非直列化できるタイプでなければなりません。特定タイプの並列化機構が見つけられない場合、チェッカーは、次の例に示されているようなエラー・メッセージを示します。

```
ERROR. Line 13: no deserializer was found
to deserialize a
 "http://www.w3.org/2001/XMLSchema:ssstring", using encoding
"http://schemas.xmlsoap.org/soap/encoding/".
```

関連概念:

- 37 ページの『DADX ファイルの定義』

関連資料:

- 267 ページの『DADX 環境チェッカーによるエラー・チェック』

付録 C. DADX ファイルの XML スキーマ

次の Extensible Markup Language (XML) スキーマ *dadx.xsd* で、 DADX を説明します。すべての WORF スキーマ・ファイルは、samples ディレクトリーの一部である dxxworf.zip ファイルにあります。

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://schemas.ibm.com/db2/dxx/dadx"
  xmlns:dadx="http://schemas.ibm.com/db2/dxx/dadx"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" xml:lang="en">
 <annotation>
  <documentation>
      A Document Accession Definition Extension (DADX)
       document defines a Web Service
       that is implemented by operations that
       access a relational database and that optionally use
       stored procedures, types and functions provided
       by the DB2 XML Extender.
    </documentation>
 </annotation>
 <element name="DADX">
  <annotation>
   <documentation>
        Defines a Web Service.
        The Web Service is described by an optional
        WSDL documentation element.
        The Web Service may implement a set of WSDL
        bindings defined elsewhere.
        The Web Service consists of one or more
        uniquely named operations.
      </documentation>
  </annotation>
```

図 77. DADX スキーマ (1/17)

```
<complexType>
   <sequence>
    <element ref="dadx:documentation"
             minOccurs="0" maxOccurs="unbounded"/>
    <choice>
     <element ref="dadx:DQS"
                 minOccurs="0"/>
     <sequence>
      <element ref="dadx:implements" minOccurs="0"/>
      <element ref="dadx:result_set_metadata"
                   minOccurs="0" maxOccurs="unbounded"/>
      <element ref="dadx:operation"
                   maxOccurs="unbounded"/>
     </sequence>
    </choice>
   </sequence>
  </complexType>
  <key name="result_set_metadataNames">
   <selector xpath="dadx:result_set_metadata"/>
   <field xpath="@name"/>
  </key>
  <keyref name="resultSetMetatdata"
         refer="dadx:result_set_metadataNames">
   <selector xpath="dadx:operation/dadx:call/dadx:result_set"/>
   <field xpath="@metadata"/>
  </keyref>
  <unique name="operationNames">
   <selector xpath="dadx:operation"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

図 77. DADX スキーマ (2/17)

```
<element name="DQS">
  <annotation>
   <documentation>
        Defines the DQS tag.
      </documentation>
  </annotation>
  <complexType/>
 </element>
```

図 77. DADX スキーマ (3/17)

```
<element name="documentation">
 <annotation>
  <documentation>
       Defines WSDL documentation for the Web service or an operation.
     </documentation>
 </annotation>
 <complexType mixed="true">
  <choice minOccurs="0" maxOccurs="unbounded">
   <any minOccurs="0" maxOccurs="unbounded"/>
  </choice>
  <anyAttribute/>
 </complexType>
</element>
```

図 77. DADX スキーマ (4/17)

```
<element name="implements">
  <annotation>
   <documentation>
       Defines the namespace and location of a set of WSDL bindings
       defined elsewhere. This information is imported into the
       WSDL document generated for this Web Service.
     </documentation>
  </annotation>
  <complexType>
   <attribute name="namespace"
            type="anyURI" use="required"/>
   <attribute name="location"
            type="anyURI" use="required"/>
  </complexType>
 </element>
```

図 77. DADX スキーマ (5/17)

```
<element name="result_set_metadata">
  <annotation>
   <documentation>
        Defines the metadata for a result set returned
        by a stored procedure.
        Each metadata element defines a global element
        in the WSDL for the Web Service.
        The metatdata name defines the name of its global element.
        The metadata rowName defines the element name of each row.
        The metadata contains one or more column definitions.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element ref="dadx:column"
             maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name"
              type="NCName"
              use="required"/>
   <attribute name="rowName"
              type="NCName"
              use="required"/>
  </complexType>
 </element>
```

図 77. DADX スキーマ (6/17)

```
<element name="column">
  <annotation>
   <documentation>
        Defines the metadata for a column of a result set
        returned by a stored procedure.
        The column name, type, and nullability must match the values
           returned by the JDBC result set metadata at runtime.
        A column is considered to be nullable unless it is explicitly
           defined to not accept nulls.
        If the "nullable" attribute is absent then
        the column is considered to not be nullable.
        The element name associated with the column isdefined
           by the value of the "as" attribute if present,
           or the column name otherwise.
        The element may contain an XML document, in which case
           it must have an "element" attribute that
           defines the XML Schema name
        of its root element.
      </documentation>
  </annotation>
  <complexType>
   <attribute name="name"
              type="string"
              use="required"/>
   <attribute name="type"
              type="dadx:columnType"
              use="required"/>
   <attribute name="nullable"
              type="boolean"/>
   <attribute name="as"
              type="string"/>
   <attribute name="element"
              type="QName"/>
  </complexType>
 </element>
```

図 77. DADX スキーマ (7/17)

```
 <simpleType name="columnType">
  <restriction base="string">
   <enumeration value="BIGINT"/>
   <enumeration value="CHAR"/>
   <enumeration value="CLOB"/>
   <enumeration value="DATE"/>
   <enumeration value="DECIMAL"/>
   <enumeration value="DOUBLE"/>
   <enumeration value="FLOAT"/>
   <enumeration value="INTEGER"/>
   <enumeration value="NUMERIC"/>
   <enumeration value="REAL"/>
   <enumeration value="SMALLINT"/>
   <enumeration value="TIME"/>
   <enumeration value="TIMESTAMP"/>
   <enumeration value="TINYINT"/>
   <enumeration value="VARCHAR"/>
  </restriction>
 </simpleType>
```

図 77. DADX スキーマ (8/17)

```
<element name="operation">
  <annotation>
   <documentation>
        Defines an operation of the Web Service.
        Each operation has a unique name and is
        optionally described
        by WSDL documentation.
        An operation is defined using one of the
        supported operators.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element ref="dadx:documentation"
             minOccurs="0"/>
    <choice>
     <element ref="dadx:retrieveXML"/>
     <element ref="dadx:storeXML"/>
     <element ref="dadx:query"/>
     <element ref="dadx:update"/>
     <element ref="dadx:call"/>
    </choice>
   </sequence>
   <attribute name="name"
           type="NCName"
           use="required"/>
  </complexType>
 </element>
```

図 77. DADX スキーマ (9/17)

```
<element name="retrieveXML">
  <annotation>
   <documentation>
        Retrieves a set of XML documents by composing
        them from relational data.
        This operator requires the DB2 XML Extender.
        The mapping from relational data to XML is defined by a
        Document Access Definition (DAD) which can be specified
        by refering to either a resource file or the name
        of an XML Collection
        that has been previously enabled in the database.
        The DAD must define an XML Collection and can use
        either SQL
        or RDB mapping. The DAD behavior may be modified by
        an override.
        If no override is desired, the no_override element
        must be used.
        Otherwise, the SQL_override element must be used
        for SQL mapping and the
        XML_override element must be used for RDB mapping.
        In either case, the
        override string may contain input parameters using
        the host variable syntax.
        The name and type of all parameters must be defined in a list of
        parameter elements that are uniquely named within this operation.
        </documentation>
  </annotation>
  <complexType>
   <sequence>
    <choice>
     <element ref="dadx:DAD_ref"/>
     <element ref="dadx:collection_name"/>
    </choice>
    <choice>
     <element name="no_override">
      <complexType/>
     </element>
     <element name="SQL_override"
                 type="string"/>
     <element name="XML_override"
                 type="string"/>
    </choice>
    <element ref="dadx:parameter"
              minOccurs="0"
              maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <unique name="retrieveXmlParameterNames">
   <selector xpath="dadx:parameter"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

図 77. DADX スキーマ (10/17)

```
<element name="storeXML">
  <annotation>
   <documentation>
        Stores an XML document by decomposing it into relational data.
        This operator requires the DB2 XML Extender.
        The mapping from relational data to XML is defined by a
        Document Access Definition (DAD) which can be specified
        by refering to either a resource file or the name of
        an XML Collection
        that has been previously enabled in the database.
        The DAD must define an XML Collection and
        must use RDB mapping.
        </documentation>
  </annotation>
  <complexType>
   <choice>
    <element ref="dadx:DAD_ref"/>
    <element ref="dadx:collection_name"/>
   </choice>
  </complexType>
 </element>
```

図 77. DADX スキーマ (11/17)

```
<element name="query">
  <annotation>
   <documentation>
        Retrieves a set of relational data using an
        SQL SELECT statement.
        The result set must consist of uniquely named columns.
        If any result set column contains XML documents,
        the XML document type must be
        defined using an XML_result element.
        The statement may contain input parameters using
        the host variable syntax.
        The input parameters must be defined by a list of
        parameter elements that are
        uniquely named within this operation.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element name="SQL_query"
               type="string"/>
    <element ref="dadx:XML_result"
               minOccurs="0"
               maxOccurs="unbounded"/>
    <element ref="dadx:parameter"
               minOccurs="0"
               maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <unique name="XML_resultNames">
   <selector xpath="dadx:XML_result"/>
   <field xpath="@name"/>
  </unique>
  <unique name="queryParameterNames">
   <selector xpath="dadx:parameter"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

図 77. DADX スキーマ (12/17)

```
<element name="update">
  <annotation>
   <documentation>
        Updates a relational table using an SQL INSERT,
        UPDATE, or DELETE statement and
        reports the number of rows affected.
        The statement may contain input parameters
        using the host variable syntax.
        The input parameters must be defined by a list of
        parameter elements that are
        uniquely named within this operation.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element name="SQL_update"
                type="string"/>
    <element ref="dadx:parameter"
                minOccurs="0"
                maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <unique name="updateParameterNames">
   <selector xpath="dadx:parameter"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

図 77. DADX スキーマ (13/17)

```
<element name="call">
  <annotation>
   <documentation>
        Calls a stored procedure.
        The call statement contains in, out, and
        in/out parameters using host variable syntax.
        The parameters are defined by a list of parameter
        elements that are uniquely named
        within the operation.
      </documentation>
  </annotation>
  <complexType>
   <sequence>
    <element name="SQL_call"
             type="string"/>
    <element ref="dadx:parameter"
             minOccurs="0"
             maxOccurs="unbounded"/>
    <element ref="dadx:result_set"
             minOccurs="0"
             maxOccurs="unbounded"/>
   </sequence>
  </complexType>
  <unique name="callParameterNames">
   <selector xpath="dadx:parameter"/>
   <field xpath="@name"/>
  </unique>
  <unique name="callResultSetNames">
   <selector xpath="dadx:result_set"/>
   <field xpath="@name"/>
  </unique>
 </element>
```

図 77. DADX スキーマ (14/17)

```
<element name="result_set">
  <annotation>
   <documentation>
        Defines a result set.
        The name defines the element name of the result
        set and becomes part of the output message.
        The metatdata name refers to a result set metadata
        element defined in the same document.
      </documentation>
  </annotation>
  <complexType>
   <attribute name="name"
            type="NCName" use="required"/>
   <attribute name="metadata"
            type="NCName" use="required"/>
  </complexType>
 </element>
 <element name="DAD_ref"
           type="string"/>
 <element name="collection_name"
           type="string"/>
```

図 77. DADX スキーマ (15/17)

```
<element name="parameter">
  <annotation>
   <documentation>
        Defines a named parameter. A parameter
        must have it contents defined either by
        an XML Schema element or type, but not both.
        The parameter kind in one of in,
        out, or in/out, with in being the default.
      </documentation>
  </annotation>
  <complexType>
   <attribute name="name"
            type="NCName"
            use="required"/>
   <attribute name="element"
            type="QName"/>
   <attribute name="type"
            type="QName"/>
   <attribute name="kind"
            type="dadx:parameterKindType"
            default="in"/>
  </complexType>
 </element>
 <simpleType name="parameterKindType">
  <restriction base="string">
   <enumeration value="in"/>
   <enumeration value="out"/>
   <enumeration value="in/out"/>
  </restriction>
 </simpleType>
```

図 77. DADX スキーマ (16/17)

```
<element name="XML_result">
  <annotation>
   <documentation>
        Defines a named column that contains XML documents.
        The document type
        must be defined by the XML Schema element
        of its root.
      </documentation>
  </annotation>
  <complexType>
   <attribute name="name"
            type="NCName"
            use="required"/>
   <attribute name="element"
            type="QName"
            use="required"/>
  </complexType>
 </element>
</schema>
```

図 77. DADX スキーマ (17/17)

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 37 ページの『DADX ファイルの定義』

関連タスク:

- 100 ページの『文書タイプ定義を XML スキーマに変換する』

関連資料:

- 94 ページの『DADX 操作の例』

付録 D. Web サービスのエンコード・アルゴリズム

これは、group.properties ファイル内のパスワードをエンコードおよびデコードするアルゴリズムです。

1. クリア・テキストの情報を、UTF-8 文字エンコードを使用して、データ・バイトのシーケンスに変換します。 L をデータ・バイト・シーケンスの長さとします。
2. データ・タイプをより長いデータ・バイトである data8 (つまり、8 倍の長さ) に変換します。 data8 のバイト k を次のように計算します。 $k = j * L + i$。ここで 0 <= i < L および 0 <= j < 8 です。まず、データ・バイト i のビット j をマスクします。次に、これを k で *排他 OR* 演算します。このステップにより、各データ・バイトのビットは、data8 シーケンスの長さ全体に行き渡ります。
3. 標準の base64 エンコード・アルゴリズムを data8 に適用します。このステップにより、バイトは印刷可能文字としてレンダリングされ、さらに 3 分の 4 (4/3) の係数で長さが長くなります。
4. エンコードしたストリングの先頭に「encoded:」を付け、エンコードされたことを示します。

関連概念:

- 78 ページの『文書アクセス定義拡張ファイルを使用した Web サービスの定義』
- 37 ページの『DADX ファイルの定義』
- 126 ページの『Web サービス・アプリケーションのテスト - シナリオ』
- 40 ページの『Web サービス・プロセスの概要』

関連資料:

- 289 ページの『付録 E. Web サービスのコマンド・リファレンス』

付録 E. Web サービスのコマンド・リファレンス

このセクションでは、WORF 内で特定の関数を実行するときに使用できるコマンドを紹介します。

Encoder

group.properties ファイルのパスワードをエンコードまたはデコードします。

- エンコードの例 (worf.jar が CLASSPATH にリストされているものとする):

```
java com.ibm.etools.webservice.rt.util.Encoder
     -in group.properties -out group.properties
```

- デコードの例 (worf.jar が CLASSPATH にリストされているものとする):

```
java com.ibm.etools.webservice.rt.util.Encoder
     -action decode -in group.properties -out group.properties
```

Dadx2Dd

DADX ファイルからデプロイメント記述子を生成します。

- 例:

```
java com.ibm.etools.webservice.rt.dadx.Dadx2Dd
```

Check_install

DADX ファイルを検証します。

- 例:

```
java com.ibm.etools.webservice.util.Check_install
   [-srv] [-schdir pathToSchemasDir]
   [-sch schemaLocations]  [-out outputFile] fileToCheck
```

dadchecker

DAD ファイルを検証します。このコマンドで使用するパラメーターの詳細は、http://www.ibm.com/software/data/db2 /extenders/xmlext/download/beta/dadcheck_rn.html を参照してください。

- 例:

```
java dadchecker.Check_dad_xml  [-dad | -xml] [-all]
    [-dup dupName] [-enc encoding][-dtd dtdPath]
    [-xstruct xmlDocument] [-out outputFile] fileToCheck
```

関連概念:

- 31 ページの『Web サービス・プロバイダーとしての DB2 の使用の概要 - WORF』
- 37 ページの『Web サービス・プロバイダーのフィーチャー』

関連タスク:

- 152 ページの『デプロイメント記述子の生成』

関連資料:

- 287 ページの『付録 D. Web サービスのエンコード・アルゴリズム』
- 145 ページの『Web サービスのサンプル - PartOrders.dadx』

DB2 Information Integrator 技術文書

以下のトピックについて説明します。

- DB2 Information Integrator PDF 資料
- DB2 Information Integrator リリース情報およびインストール要件
- DB2 Information Integrator 資料のフィックスパック

DB2 Information Integrator の資料

DB2 PDF Documentation CD には、 DB2 Information Integrator ライブラリーと DB2 Universal Database ライブラリーの資料が、 PDF ファイルで収められています。 *DB2 PDF Documentation* CD の構成は、次のとおりです。

- Windows オペレーティング・システム: *x:¥doc¥%L*
- UNIX オペレーティング・システム: */cdrom/doc/%L/*

ここで、

- *x* は Windows CD ドライブ名
- *cdrom* は CD の UNIX マウント・ポイント
- *%L* は使用する資料のロケール (たとえば、*en_US*)

次の表は、PDF 資料にアクセスするときに使用できる言語 ID をリストしています。

表 30. 言語ロケールおよび ID

言語	ロケール	ID	言語	ロケール	ID
アラビア語	ar_AA	w	日本語	ja_JP	j
ブラジル・ポルトガル語	pt_BR	b	韓国語	ko_KR	k
ブルガリア語	bg_BG	u	ノルウェー語	no_NO	n
クロアチア語	hr_HR	9	ポーランド語	pl_PL	p
チェコ語	cs_CZ	x	ポルトガル語	pt_PT	v
デンマーク語	da_DK	d	ルーマニア語	ro_RO	8
オランダ語	nl_NL	q	ロシア語	ru_RU	r
英語	en_US	e	中国語 (簡体字)	zh_CN	c
フィンランド語	fi_FI	y	スロバキア語	sk_SK	7
フランス語	fr_FR	f	スロベニア語	sl_SI	l
ドイツ語	de_DE	g	スペイン語	es_ES	z
ギリシャ語	el_GR	a	スウェーデン語	sv_SE	s
ハンガリー語	hu_HU	h	中国語 (繁体字)	zh_TW	t
イタリア語	it_IT	i	トルコ語	tr_TR	m

各 PDF ファイル名の 6 番目の文字は、資料の言語バージョンを示しています (下記の表を参照)。たとえば、iiyige80 というファイル名は、英語版の「*IBM DB2 Information Integrator インストール・ガイド*」であることを示し、 iiyigg80 というファイル名は、同じ資料のドイツ語版を示しています。

次の表は、DB2 Information Integrator で使用可能な PDF 資料をリストしています。

表 31. DB2 Information Integrator 資料

名前	資料番号	PDF ファイル名
「*IBM DB2 Information Integrator アプリケーション開発者向けガイド*」	SC88-9609-01	iiyfsx81
「*IBM DB2 Information Integrator ラッパー開発における C++ API リファレンス*」	SC88-9921-01	iiywrx80
「*IBM DB2 Information Integrator データ・ソース構成ガイド*」	オンライン専用	
「*IBM DB2 Information Integrator フェデレーテッド・システム・ガイド*」	SC88-9614-01	iiyfpx81
「*IBM DB2 Information Integrator インストール・ガイド*」	SC88-9562-01	iiyigx81
「*IBM DB2 Information Integrator レプリケーションとイベント・パブリッシング入門*」	GC88-9895-00	db2gpx80
「*IBM DB2 Information Integrator ラッパー開発における Java API リファレンス*」	SC88-9922-01	iiyjrx80
「*IBM DB2 Information Integrator マイグレーション・ガイド*」	SC88-9610-01	iiymgx81
「*IBM DB2 Information Integrator レプリケーションとイベント・パブリッシング ガイドおよびリファレンス*」	SC88-9893-00	db2qrx80
「*IBM DB2 Information Integrator SQL レプリケーション・ガイドおよびリファレンス*」	SC88-9163-02	db2e0x82
「*IBM DB2 Information Integrator ラッパー開発者向けガイド*」	SC88-9923-01	iiywdx80

リリース情報およびインストール要件

リリース情報およびインストール要件では、製品のリリースおよびフィックスパックのレベルに固有の情報を提供します。また、各リリースおよびフィックスパックに組み込まれているドキュメンテーション更新のサマリーも提供します。

リリース情報およびインストール要件は、テキスト形式および HTML 形式で製品 CD から入手できます。

- Windows オペレーティング・システムの場合: *x:¥doc¥%L*
- UNIX オペレーティング・システムの場合: */cdrom/doc/%L/*

各部分の意味は以下のとおりです。

- *x* は Windows の CD ドライブ名を表します。
- *cdrom* は CD のマウント・ポイントを表します (UNIX)。

- *%L* は使用する資料のロケールです (*en_US* など)。

次の表は、リリース情報およびインストール要件のロケーションを示しています。

表 32. リリース情報およびインストール要件

資料名	ファイル名	位置
IBM DB2 Information Integrator リリース・ノート	ReleaseNotes	• DB2 インフォメーション・センターにあるリリース情報 • Web 上で http://www.ibm.com/software/data/integration/db2ii/support.html にアクセスします。
IBM DB2 Information Integrator インストール要件	Prereqs	• 製品 CD • DB2 Information Integration Installation Launchpad

DB2 Information Integrator ドキュメンテーション・フィックスパック

IBM は、DB2 インフォメーション・センターの資料のフィックスパックやその他の資料更新を定期的に発行しています。 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) にアクセスすれば、常に最新の情報が掲載されます。 DB2 インフォメーション・センターをローカル・インストールしている場合、更新記事を表示するには、まず手動で更新をインストールしてください。新しい情報が発表されたときに資料を更新することにより、*DB2 インフォメーション・センター CD* からインストールした情報を更新することができます。

インフォメーション・センターの方が、PDF 資料やハードコピー資料よりも頻繁に更新されます。 DB2 の最新の技術情報を入手するには、資料更新が発行されたときにそれをインストールするか、または www.ibm.com サイトの DB2 インフォメーション・センターにアクセスしてください。

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 ユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 296 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 296 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 296 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

アクセスしやすい表示

DB2 ツールには、視力の弱いユーザー、その他の視力障害をもつユーザーのためにアクセシビリティーを向上させる機能が備わっています。これらのアクセシビリティー拡張機能には、フォント・プロパティーのカスタマイズを可能にする機能も含まれています。

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、メニューおよびテキストのフォントを変更する: Common GUI help を参照してください。

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

DB2 形式は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。 XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン・ドキュメンテーションにアクセスする場合にのみ使用できます。

関連概念:

- 「*Infrastructure Topics (DB2 Common Files)*」の『ドット 10 進シンタックス・ダイアグラム』

関連タスク:

- 『キーボード・ショートカットおよびアクセラレーター: Common GUI help』
- 『メニューおよびテキストのフォントを変更する: Common GUI help』

参照文献

- *WebSphere: WebSphere Solution Bundles: Implementation and Integration Guide*, SG24-6550
- *MQSeries: アプリケーション・メッセージング・インターフェース*, SC88-9227-00
- *Information Integrator: Planning, Installation and Configuration Guide*
- *Information Integrator: フェデレーテッド・システム・ガイド*
- *IBM DB2 Universal Database: Life Sciences Data Connect 計画、インストール、および構成のガイド*, GC88-9173
- *IBM DB2 Universal Database: XML Extender 管理およびプログラミングのガイド*, SC88-9172
- *IBM DB2 Universal Database: データウェアハウス・センター 管理ガイド*, SC88-9165
- *IBM DB2 Universal Database: レプリケーション・ガイドおよびリファレンス*, SC88-9163
- *IBM Enterprise Information Portal for Multiplatforms: Enterprise Information Portal の計画およびインストール*, GC88-8707
- *IBM Enterprise Information Portal for Multiplatforms: Enterprise Information Portal の管理*, SC88-8709
- *IMS: Administration Guide: Database Manager*, SC26-9419-02
- *IBM WebSphere: Application Server V4 for z/OS and OS/390: インストールおよびカスタマイズ*, GA88-8652-02
- *IBM WebSphere: Application Server V4.0.1 for z/OS and OS/390: システム管理スクリプト API*, SA88-8657
- *DB2 XML Extender: 管理およびプログラミングのガイド*
- *DB2 XML Extender: DB2 XML Extender Administration and Programming, Version 7.2 Release Notes*
- *Using WSDL in a UDDI Registry 1.07*
- *WebSphere Handbook*
- *Web Services Description Language (WSDL) 1.1*
- *Dynamic e-business: The next stage of e-business and Web services*
- *Web services zone*

特記事項

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

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

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

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

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

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

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

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

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

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

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

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

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

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

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

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケ

ーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

商標

以下は、IBM Corporation の商標です。

IBM
AIX
DB2
DB2 Extenders
DB2 Universal Database
Domino
Informix
Intelligent Miner
Lotus
OS/390
UNIX
WebSphere
z/OS

以下は、それぞれ各社の商標または登録商標です。

Java は、Sun Inc. の米国およびその他の国における商標です。

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

Intel、Intel Inside (ロゴ)、MMX および Pentium は、Intel Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

A

C

D

E

G

H

I

J

L

M

N

O

P

R

S

T

U

W

X

Z

IBM と連絡を取る

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Information Integrator についての情報は、電話または Web から入手することができます。

米国にお住まいの場合は、以下のいずれかの番号にお問い合わせください。

- 製品の注文または一般情報の入手: 1-800-IBM-CALL (1-800-426-2255)
- 資料の注文: 1-800-879-2755

Web 上で www.ibm.com/software/data/integration/db2ii/support.html にアクセスします。このサイトには、以下の最新情報が記載されています。

- テクニカル・ライブラリー
- 資料の注文
- クライアント・ダウンロード
- ニュースグループ
- フィックスパック
- ニュース
- Web リソースへのリンク

資料についてのコメント

お客様のフィードバックは IBM が良質な情報を提供する助けになります。この資料や他の DB2 Information Integrator の資料についてのコメントをお送りください。コメントの送付には、以下のいずれかの方法を利用することができます。

- www.ibm.com/software/data/rcf で、オンラインの読者コメント・フォームを使用して送信する。
- E メールで comments@us.ibm.com にコメントを送信する。お送りいただく情報には、製品の名前、製品のバージョン番号、および資料の名前と部品番号 (該当する場合) を含めてください。特定の本文についてコメントする場合は、本文の位置 (たとえば、タイトル、表の番号、またはページ番号) を含めてください。

Printed in Japan

SC88-9609-01

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