WebSphere Adapters

バージョン 6 リリース 1

WebSphere Adapter for JDBC ユーザーズ・ガイド
バージョン 6 リリース 1

WebSphere Adapters

バージョン 6 リリース 1

WebSphere Adapter for JDBC ユーザーズ・ガイド
バージョン 6 リリース 1

お願い

本書および本書で紹介する製品をご使用になる前に、 263 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM WebSphere Adapter for JDBC バージョン 6、リリース 1、モディフィケーション 0 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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。

IBM 発行のマニュアルに関する情報のページ

`http://www.ibm.com/jp/manuals/`

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(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： WebSphere® Adapters
Version 6 Release 1
WebSphere Adapter for JDBC User Guide
Version 6 Release 1

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 2008.3

目次

第 1 章 WebSphere Adapter for JDBC の概要 1
このリリースの新機能 1
ハードウェアおよびソフトウェア要件 4
WebSphere Adapter for JDBC の技術的な概説 . . . 5
Outbound 処理 8
Inbound 処理 21
ビジネス・オブジェクト 26
ストアード・プロシージャーの概要 42
ストアード・プロシージャー・ビジネス・オブジェクトの概要 42
操作の代わりまたは追加で使用するストアード・プロシージャー 43
ストアード関数の概要 49
クエリー・ビジネス・オブジェクトの概要 . . . 50
バッチ SQL ビジネス・オブジェクトの概要 . . 51
外部サービス・ウィザード 51
標準の準拠 51
アクセシビリティー 51
インターネット・プロトコル・バージョン 6 (IPv6) 52

第 2 章 アダプター実装の計画 55
始める前に 55
セキュリティー 55
ユーザー認証 55
デプロイメント・オプション 57
クラスター環境での WebSphere Adapters 60
準備済みステートメントのキャッシュのサポート . . 61
バージョン 6.1.0 へのマイグレーション 62
マイグレーションに関する考慮事項 62
マイグレーションの実行 63
マイグレーションしない場合のバージョン 6.0.2 プロジェクトの更新 65

第 3 章 サンプルおよびチュートリアル 67

第 4 章 デプロイメントのためのモジュールの構成 69
モジュールの構成のためのロードマップ 69
イベント・ストアの作成 71
認証別名の作成 72
プロジェクトの作成 73
外部ソフトウェア依存関係の追加 75
外部サービス・ウィザードの接続プロパティーの設定 76
Outbound 処理のモジュールの構成 78
データベース・オブジェクトのディスカバリー . 79
ビジネス・オブジェクトの選択および構成 . . . 83
操作のグローバル・プロパティーの設定および wrapper ビジネス・オブジェクトの作成 . . . 103
デプロイメント・プロパティーの設定およびサービスの生成 106
構成の完了 113
Inbound 処理のモジュールの構成 114
データベース・オブジェクトのディスカバリー 114
ビジネス・オブジェクトの選択および構成 . . . 117
操作のグローバル・プロパティーの設定 . . . 126
デプロイメント・プロパティーの設定およびサービスの生成 128
構成の完了 137

第 5 章 アセンブリー・エディターによる対話仕様プロパティーの変更 139

第 6 章 モジュールのデプロイ 141
デプロイメント環境 141
テスト用のモジュールのデプロイ 141
Inbound 処理をテストするためのターゲット・コンポーネントの生成および接続 141
Outbound 操作のテスト準備 143
サーバーへのモジュールの追加 144
テスト・クライアントを使用した Outbound 処理用モジュールのテスト 145
実稼働のためのモジュールのデプロイ 146
サーバー上での外部ソフトウェア依存関係の追加 146
RAR ファイルのインストール (スタンドアロン・アダプターを使用するモジュールの場合のみ) 147
EAR ファイルとしてのモジュールのエクスポート 149
EAR ファイルのインストール 151

第 7 章 アダプター・モジュールの管理 153
組み込みアダプターの構成プロパティーの変更 . . 153
組み込みアダプターのリソース・アダプター・プロパティーの設定 153
組み込みアダプターの管理 (J2C) 接続ファクトリー・プロパティーの設定 155
組み込みアダプターのアクティベーション・スペック・プロパティーの設定 157
スタンドアロン・アダプターの構成プロパティーの変更 159
スタンドアロン・アダプターのリソース・アダプター・プロパティーの設定 159
スタンドアロン・アダプターの管理 (J2C) 接続ファクトリー・プロパティーの設定 160
スタンドアロン・アダプターのアクティベーション・スペック・プロパティーの設定 162
アダプターを使用するアプリケーションの開始 . . 164

アダプターを使用するアプリケーションの停止 . . 164
Performance Monitoring Infrastructure を使用したパフォーマンスのモニター. 165
Performance Monitoring Infrastructure の構成 . . 165
パフォーマンスに関する統計の表示 168
Common Event Infrastructure (CEI) を使用したトレースの使用可能化 169
トラブルシューティングとサポート 170
ロギングおよびトレースの構成 170
First Failure Data Capture (FFDC) サポート . . 174
ビジネス・フォールト 174
XAResourceNotAvailableException 179
セルフ・ヘルプ・リソース 180
一般的な問題の解決策 181

第 8 章 参照. 191
ビジネス・オブジェクト情報 191
ビジネス・オブジェクト属性 191
属性に関するアプリケーション固有情報 . . . 193
ビジネス・オブジェクト・レベルのアプリケーション固有情報 200
命名規則 204
Outbound 構成プロパティー 206
ウィザードの接続プロパティー 208
リソース・アダプター・プロパティー 213
管理接続ファクトリー・プロパティー 217
対話スペック・プロパティー 226
Inbound 構成プロパティー 227
ウィザードの接続プロパティー 229
リソース・アダプター・プロパティー 234
アクティベーション・スペック・プロパティー 238
グローバリゼーション 255
グローバリゼーションおよび双方向変換 . . . 255
双方向データ変換で使用可能なプロパティー . . 258
アダプター・メッセージ. 260
関連情報 260

特記事項. 263
プログラミング・インターフェース情報 265
商標 265

索引 267

第 1 章 WebSphere Adapter for JDBC の概要

WebSphere Adapter for JDBCにより、データベースとの情報交換を組み込んだ統合アプリケーションを作成できます。アプリケーションは、アダプターを使用することにより、要求をデータベースに送信すると共に、多くの場合 SQL コードを必要とせずにデータベースからイベントを受け取ることができます。

アダプターにより、WebSphere Process Server または WebSphere Enterprise Service Bus で実行されるアプリケーションと、データベースとの間で両方向の通信が可能になります。アプリケーションは、アダプターを使用して、多くの場合 SQL コードを書き込まずに、データベースにあるデータの読み取り、作成、変更、または削除要求を送信できます。アプリケーションから受け取った要求を処理するため、アダプターは SQL 照会またはストアード・プロシージャーを使用してデータベース表を更新します。アプリケーションは、データベースからのイベントも受信できます。例えば、特定のデータベース表が更新されたという通知を受け取ることができます。データベースに対する変更によって生じたイベントを処理するため、アダプターはイベントをアプリケーションに送信します。イベント通知を使用して、データベース更新を自動的に他のアプリケーションに伝搬できます。WebSphere Adapter for JDBC と別のアダプターによるイベント処理を結合することで、Siebel、PeopleSoft、Oracle などのエンタープライズ・アプリケーションに更新を自動的に伝搬できます。

アダプターは、さまざまなデータベース・ソフトウェアのベンダーとバージョンを統合する標準インターフェースを備えており、JDBC 2.0 以降の仕様をサポートする Java™ Database Connectivity (JDBC) ドライバーがインストールされているすべてのデータベース・サーバーをサポートします。そのようなサーバーの例としては、IBM® DB2®、Oracle、Microsoft® SQL Server、Sybase、Derby、および Informix® などがあります。アダプターは、ビジネス・オブジェクトを使用してアプリケーションとデータベース間でデータを交換するため、アプリケーションでは JDBC アプリケーション・プログラミング・インターフェース (API) を使用する必要がありません。*ビジネス・オブジェクト* とは、ビジネス機能やビジネス・エレメントを表すアプリケーション・データ (データベース表や SQL 照会の結果など) のコンテナーです。アダプターは、アプリケーションのデータ・フォーマットを認識し、データを処理し、操作を実行し、結果をそのフォーマットで戻すことができます。

このリリースの新機能

WebSphere Adapter for JDBC バージョン 6.1.0 では、アダプターの機能が拡張されています。このリリースでは、一部の機能が非推奨になっています。

この情報に関する最新情報は、WebSphere Adapters 製品サポート Web サイトから入手できます。更新情報や追加情報を確認するには、http://www.ibm.com/software/integration/wbiadapters/support/ を参照してください。

バージョン 6.1.0 では非推奨

非推奨の機能とは、サポートされているが使用が推奨されておらず、廃止される可能性がある機能です。バージョン 6.1.0 では非推奨になっている Adapter for JDBC の以前のバージョンの機能リストについては、63 ページの『非推奨機能』を参照してください。

バージョン 6.1.0の新機能

- バッチ SQL スクリプトのサポート (ユーザー指定 SQL INSERT、UPDATE、および DELETE ステートメント)

  これで、データベースに対して一連の INSERT、 UPDATE、および DELETE SQL ステートメントを実行するバッチ SQL スクリプトのビジネス・オブジェクトを作成できます。
- イベントでのオブジェクト取得のタイミングに対する変更

  イベントの対象であるデータベース・オブジェクトは、通知がエクスポートに送信された後に取得されます。この結果、発生した取得エラーの検出と通知は、エクスポートの通知の完了後まで据え置かれます。これは、アダプターがエクスポートに通知する前に取得エラーを検出可能な、バージョン 6.0.2 のアダプターでのイベント処理とは異なります。
- ビジネス・グラフおよび動詞はオプションになりました。

  バージョン 6.0.2 での各ビジネス・オブジェクトが含まれたビジネス・グラフが、オプションになりました。バージョン 6.0.2 で作成されたビジネス・オブジェクトのモジュール用、または、ApplyChanges Outbound 操作を使用する新規 バージョン 6.1.0 モジュール用にのみビジネス・グラフが必要になります。
- エンタープライズ・サービス・ディスカバリー・ウィザードの変更

  このウィザードの名前は 外部サービス・ウィザード に変更されました。また、機能と使用可能度が向上したことにより、アダプターと組み合わせて使用するビジネス・オブジェクトおよびサービスのディスカバー、作成および構成が容易になりました。このウィザードでは、以前はファイル・システム内または WebSphere Integration Developer 内で手動で実行していたいくつかの作業 (プロジェクトの作成、JDBC ドライバーのプロジェクトへのインポート、モジュールの作成など) の手順を順を追って示すようになりました。

  このウィザードでは、多くのプロパティーのデフォルト値を提示するようになり、特定の情報を入力しやすくなって、どのプロパティーが必須かを表示するようになりました。また、拡張プロパティーの心配をすることなくモジュールを構成できるようになりました。
- 双方向スクリプト処理のサポートが簡素化されました。
- ノード・レベルまたはスタンドアロンでのアダプターのデプロイメントのサポート
- ビジネス・フォールトのサポート

  アダプターはビジネス例外に対してビジネス・フォールトを生成するようになりました。これにより、それらのエラー条件に修正アクションを容易に割り当てることができます。

- アダプター RAR ファイルは WebSphere Integration Developer 内で使用可能です。つまり、アダプター RAR ファイルを別にインストールする必要はありません。アダプター・ファイルはウィザードによって自動的にプロジェクトにコピーされます。
- アダプターの資料は、WebSphere Integration Developer インフォメーション・センターのアダプターの構成および使用セクションにあります。
- ログに記録されるデータのカスタマイズ時に診断モジュールを支援するための情報および推奨アクションを示すために WebSphere Application Server 症状データベースに格納できる First Failure Data Capture (FFDC) 構造のサポート。
- IPv6 アドレスをサポートします。

バージョン 6.0.2、フィックスパック 2 で導入された機能

- ユーザー名とパスワード以外の接続プロパティー

  新しいプロパティーである DriverConnectionProperties は、追加接続プロパティーを指定するために、アクティベーション・スペック (Inbound 処理) および管理接続ファクトリー (Outbound 処理) で使用されます。
- 準備済みステートメントをキャッシュに格納するデータ・ソースのサポート

  データベースへの接続にデータ・ソースを使用する場合、準備済みステートメントのキャッシュ機能を使用して、Inbound および Outbound 処理のパフォーマンスを強化します。
- 外部サービス・ウィザードでのストアード・プロシージャー名の表示の改善

  ストアード・プロシージャー名の表示が読みやすくなったため、外部サービス・ウィザードでストアード・プロシージャーを容易に見つけることができます。
- ビジネス・オブジェクト階層のサポートの強化
  - ビジネス・オブジェクト・エディターではなく、外部サービス・ウィザードで、子ビジネス・オブジェクトの階層を作成します。これにより、親ビジネス・オブジェクトと子ビジネス・オブジェクトに外部キーが適切に設定されます。
  - 外部サービス・ウィザードでラッパー・ビジネス・オブジェクトを作成し、無関係なビジネス・オブジェクトを 1 つのビジネス・オブジェクト階層にまとめます。
- Oracle および MS SQL データベースのグローバル化データ型のサポート

  ビジネス・オブジェクト・エディターで、Oracle と MS SQL のグローバル化データ型の列の属性を手動で追加する必要はありません。データ型 NCHAR、NVARCHAR、NTEXT、TEXT、RAW、MONEY、および SMALLMONEY がサポートされています。
- Update、Delete、Retrieve、および RetrieveAll 操作での NULL 値のサポート

  アダプターは、データベース表で列値が NULL のレコードを更新または取得できます。
- クイック・スタート・チュートリアルの追加

Web で利用可能なクイック・スタート・チュートリアルが追加されました。このチュートリアルで、アダプターについて習得します。

バージョン 6.0.2 で導入された機能

- SQL データ型 CLOB (文字ラージ・オブジェクト) および BLOB (バイナリー・ラージ・オブジェクト) をサポートします。
- ストアード関数をサポートします。ストアード関数は、常に値を戻すという点を除き、ストアード・プロシージャーに類似しています。アダプターは、ReturnValue アプリケーション固有情報があるかどうかをチェックし、存在する場合はストアード関数を実行します。
- カスタム・イベント処理をサポートします。カスタム照会は、標準 SQL、ストアード・プロシージャー、またはストアード関数のいずれかを使用します。
- EDT は、「送達は 1 回のみ」に使用されなくなりました。この機能のために、アクティベーション・スペック・プロパティー AssuredOnceDelivery が新たに提供されます。
- ストアード・プロシージャーをビジネス・オブジェクトに関連付けるときに、新しいフィルターを使用して、エンタープライズ・メタデータ・ディスカバリーでストアード・プロシージャーのリストを絞り込むことができます。
- アダプターは、アクティベーション・スペック・プロパティー EventFilterType を使用して、処理するイベントをビジネス・オブジェクト・タイプによってフィルタリングすることができ、また、プロパティー FilterFutureEvents を使用して、タイム・スタンプによってイベントをフィルタリングすることができます。
- エンタープライズ・サービス・ディスカバリー中に、ユーザー指定 SELECT ステートメントからクエリー・ビジネス・オブジェクトを生成できます。
- ストアード・プロシージャーおよびストアード関数をサポートする Execute 操作を提供します。
- WebSphere Application Server で定義されたデータ・ソースを介したデータベース接続の確立をサポートします。DataSourceJNDIName プロパティーが、J2C 管理接続ファクトリーおよびアクティベーション・スペック・プロパティーに追加されました。
- Inbound 処理の高可用性をサポートします。詳しくは、60 ページの『クラスター環境での WebSphere Adapters』を参照してください。

ハードウェアおよびソフトウェア要件

WebSphere Adapters のハードウェアおよびソフトウェア要件は、以下のロケーションにある IBM Web サイトに記載されています。

WebSphere Adapters のハードウェアおよびソフトウェア要件: http://www.ibm.com/support/docview.wss?uid=swg27006249

追加情報

以下のリンク先には、アダプターの構成およびデプロイに必要になる場合がある追加情報が記載されています。

- WebSphere Business Integration Adapters と WebSphere Adapters の互換性一覧表には、使用するアダプターに必要なソフトウェアのサポートされるバージョンが

記載されています。この資料を参照するには、WebSphere Adapters のサポート・ページにアクセスし、**アップグレードの計画 (Planning upgrades)**: http://www.ibm.com/software/integration/wbiadapters/support/ の下にある互換性一覧表のリンクをクリックします。

- WebSphere Adapters のテクニカル・ノートには、製品資料に記載されていない次善策および追加情報が収録されています。アダプターのテクニカル・ノートを参照するには、Web ページ http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm にアクセスし、「**Product category**」リストからアダプターの名前を選択し、検索アイコンをクリックします。

WebSphere Adapter for JDBC の技術的な概説

アダプターは、JDBC アプリケーション・プログラミング・インターフェース (API) を介してアクセス可能なデータベースと、WebSphere Process Server または WebSphere Enterprise Service Bus で稼働するアプリケーションとの統合をサポートします。アダプターは、Java 2 Platform, Enterprise Edition (J2EE) コネクター・アーキテクチャー (JCA) の下で Outbound および Inbound 処理を提供し、Service Component Architecture (SCA) コンポーネントと統合します。

Outbound 処理により、アプリケーションはデータベースのデータに対してアクセスしたり変更を行ったりできます。アダプターは、アプリケーションからの要求を Outbound 操作に変換し、その操作を実行してデータベースのデータを作成、検索、更新、または削除したり、データベースに保管されたデータベース・プログラムを実行したりします。これらの要求を処理すると、対応するデータベース表の行が作成、検索、更新、または削除されます。またアダプターによって、データベース内に定義されたストアード・プロシージャーまたはストアード関数を実行でき、ユーザー定義の SELECT、INSERT、UPDATE、および DELETE ステートメントを実行することができます。アダプターを使用すると、同じデータベースに対して複数のアプリケーションを統合できます。

6 ページの図 1 は Outbound 処理のフローの概要を示したものです。WebSphere Process Server または WebSphere Enterprise Service Bus で実行中のアプリケーションが Outbound モジュール内のサービスを呼び出し、このサービスは 1 つ以上のビジネス・オブジェクトを処理する要求をアダプターに送信します。アダプターは JDBC API を使用してデータベース・サーバーに接続し、データベース内のテーブルやその他のオブジェクトにアクセスします。

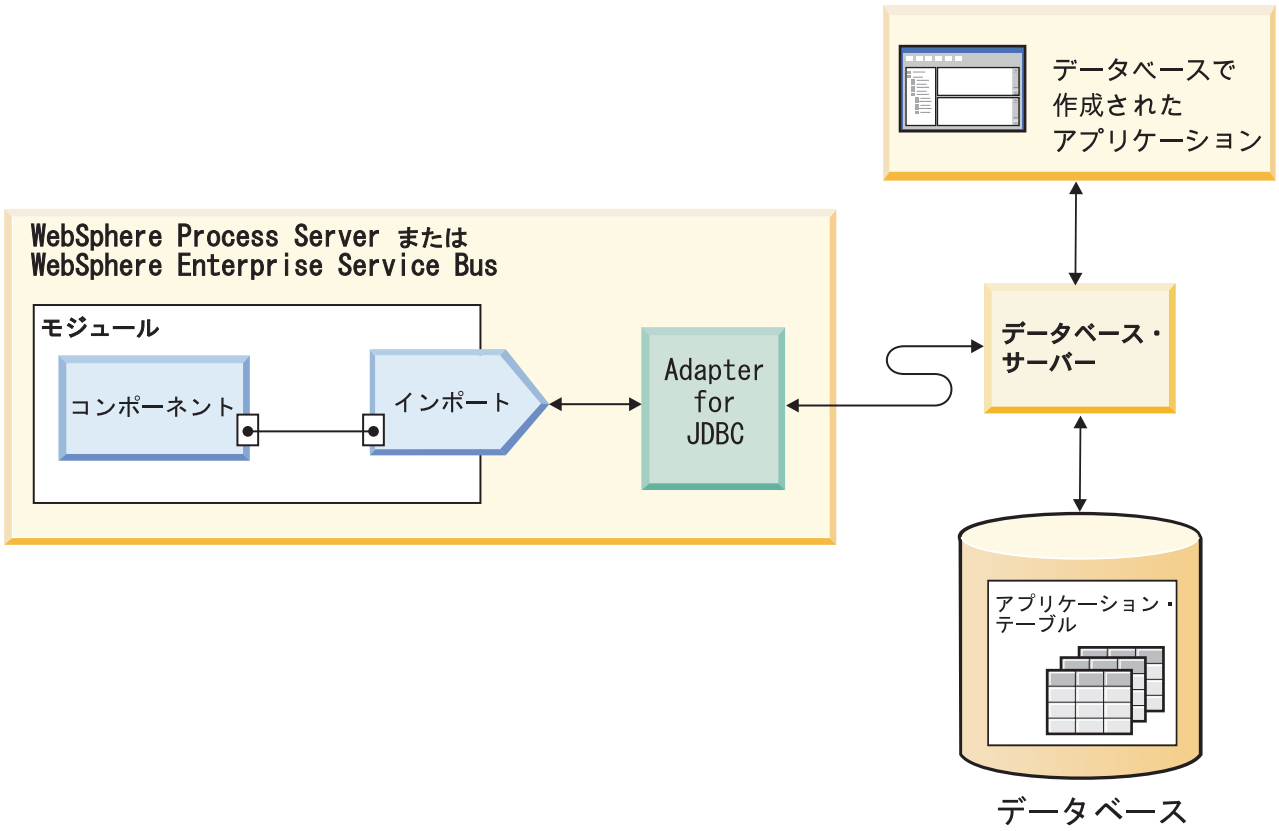


図 1. Outbound 要求の処理

Inbound 処理により、アプリケーションは、データベース内のオブジェクト変更時に通知を受け取ることができます。例えば、選択したデータベース表の行が作成、更新、または削除された場合にアプリケーションに通知できます。

7 ページの図 2 は Inbound 処理のフローの概要を示したものです。データベース・アプリケーションがデータベース内のテーブルを変更します。変更によってトリガーまたはその他の自動化機構が動作し、変更に関する情報でイベント・ストアを更新します。アダプターは定期的にイベント・ストアをポーリングし、イベントを取得および処理して、WebSphere Process Server または WebSphere Enterprise Service Bus で実行するアプリケーションの一部であるモジュールのエクスポートにイベントを送達します。

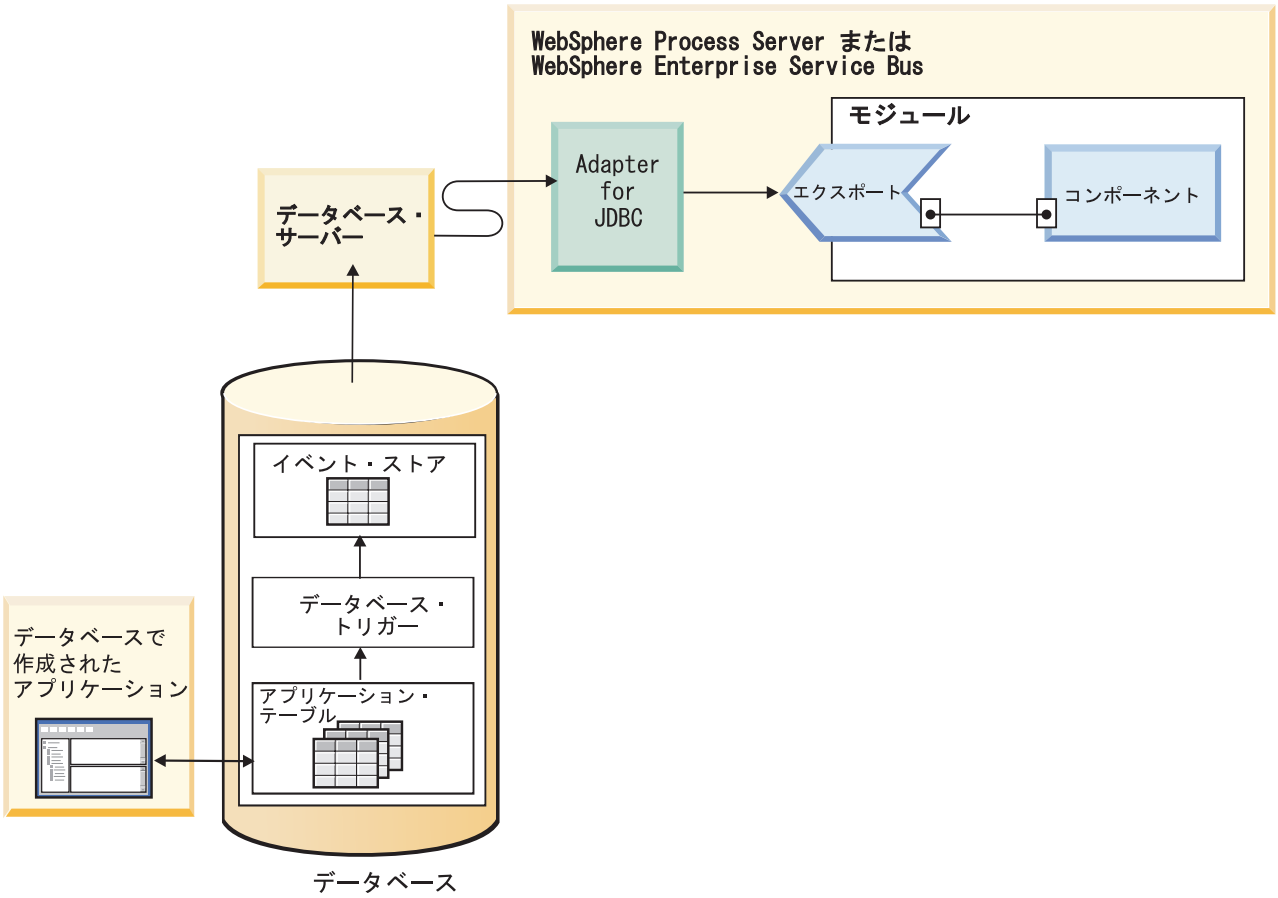


図 2. Inbound イベントの処理

アダプターは、次のいずれかの方法でイベントを処理できます。

- データベース・アプリケーションによって取り込まれたイベント・ストアを使用した標準イベント処理。
- ユーザー定義のデータベース・クエリーを使用したカスタム・イベント処理。

標準イベント処理 では、データベース内のテーブルのデータが変更されると、対応するイベントがキー値などの関連情報と共に、イベント・ストアと呼ばれるデータベース表に挿入されます。変更されたデータを取り込むために、それぞれのテーブルにトリガーを設定でき、また、Oracle データベースに対して提供される Oracle Change Data Capture のような他のメソッドを使用することもできます。アダプターは、イベント・ストアをポーリングし、一まとまりのイベントを検索します。イベントは、ビジネス・オブジェクト・タイプおよびタイム・スタンプ別にフィルターに掛けることができます。アダプターは各イベントを使用して、そのイベントによって変更されたビジネス・オブジェクトを含むビジネス・グラフまたはビジネス・オブジェクトを構成します。ビジネス・オブジェクトまたはビジネス・グラフはその後、特定のビジネス・オブジェクトを受信するように構成されたエクスポートにディスパッチされます。

カスタム・イベント処理 では、アダプターは、ユーザーが標準 SQL ステートメント、ストアード・プロシージャー、またはストアード関数として指定したクエリーを実行します。これらのアクションでは、クエリーによって返されるデータについ

て、結果セットが戻されます。結果セットの各行は、イベント・ストアの行に対応します。アダプターは、各イベントのビジネス・オブジェクトを構成して、それを特定のビジネス・オブジェクト用に構成された (またはビジネス・オブジェクトをサブスクライブする) エクスポート (エンドポイントとも呼ばれる) に送信します。

標準およびカスタムの両方のイベント処理について、アダプターがイベントをポーリングする頻度と、各ポーリング期間に取得するイベント数を指定することができます。

Outbound 処理

アプリケーション・コンポーネントがデータベースのデータを取得または変更する必要がある場合、アダプターは、アプリケーション・コンポーネントとデータベースの間のコネクターとして機能します。アダプターの一連の標準 Outbound 操作では、変更後イメージまたは差分スタイル・ビジネス・オブジェクトのいずれかが処理されます。また、アダプターは、Outbound 処理のためにローカル・トランザクションと XA (分散) トランザクションの両方をサポートしています。

アダプターのビジネス・オブジェクト・モデルでは、更新を行うビジネス・オブジェクトとして、変更後イメージと差分という 2 つのスタイルを使用します。*変更後イメージ* ・ビジネス・オブジェクトは、必要なすべての変更が行われた後のビジネス・オブジェクトの完全な状態を含んだものです。*差分* ビジネス・オブジェクトとは、キー値および変更対象のデータのみを含むビジネス・オブジェクトのことです。差分ビジネス・オブジェクトは、ビジネス・オブジェクトを更新する操作でのみ使用されます。

サポートされる操作

表 1 は、ビジネス・オブジェクトの各タイプでサポートされる Outbound 操作をリストしたもので、変更後イメージまたは差分のスタイルの処理をそれぞれサポートするかどうかを示しています。

表 1. ビジネス・オブジェクトの各タイプでサポートされる Outbound 操作

サポートされるビジネス・オブジェクト	操作	変更後イメージのサポート	差分のサポート
テーブルビューシノニムニックネーム	Create	はい	いいえ
	Update	はい	いいえ
	Delete	はい	いいえ
	Retrieve	該当なし	該当なし
	RetrieveAll	該当なし	該当なし
	ApplyChanges	はい	はい
ストアード・プロシージャー バッチ SQL	Execute	該当なし	該当なし
照会	RetrieveAll	該当なし	該当なし
ラッパー	Create	はい	いいえ
	Update	はい	いいえ
	Delete	はい	いいえ
	Retrieve	はい	いいえ

トランザクション管理

アダプターは、Outbound 処理のためにローカル・トランザクションと XA (分散) トランザクションの両方をサポートしています。このアダプターでは、トランザクションとは、データベースとの独立した相互作用です。トランザクションは、アトミックな単位で実行する、データベースへの複数の操作から構成されます。これらの操作は、データベースの他のユーザーによって同時に実行される操作の影響を受けるものではありません。

アダプターがトランザクションをサポートするのは、データベース・サーバーがトランザクションをサポートする場合のみです。サポートされるトランザクションのタイプは、ローカル・トランザクションと XA トランザクションです。

- *ローカル・トランザクション* では、あるコンポーネントが、単一データベースを使用したトランザクションの開始および終了を定義します。このトランザクションでは、1 フェーズ・コミット・プロトコルを使用します。
- *XA トランザクション* では、トランザクションは複数の異種データベースに渡ることができます。このトランザクションでは、グローバル・プロトコル (2 フェーズ・コミット・プロトコル) を使用します。

XA トランザクション

このアダプターは、Outbound 処理の XA トランザクションをサポートします。XA トランザクションのためにアダプターを構成する方法として、次のいずれかを選択します。

- XA トランザクションをサポートする JNDI データ・ソースの指定 (DataSourceJNDIName プロパティーを使用)
- XA データ・ソースおよびデータベースの指定 (XADataSourceName プロパティーおよび XADatabaseName プロパティーをそれぞれ使用)

DataSourceJNDIName プロパティーは、WebSphere Process Server または WebSphere Enterprise Service Bus 内部で作成されたデータ・ソースを表します。この名前は、XA データ・ソースまたは接続プール・データ・ソースを表します。サーバーで XA トランザクションをサポートする JNDI データ・ソースを定義し、アダプターの構成時にそのデータ・ソースを指定すると、アダプターでは、XA トランザクションをサポートするすべてのタイプのデータベースがサポートされます。XA データ・ソースとデータベースを使用する場合は、アダプターは DB2 および Oracle データベースでのみ XA トランザクションをサポートします。

Outbound 操作

アプリケーション・コンポーネントでは、データベースからの取得などのアクションを実行するために操作を使用します。アダプターは特定の Outbound 操作を提供します。サポートされる操作ごとにアダプターがビジネス・オブジェクトをどう処理するかについての詳細を説明します。

操作を実行するには、アダプターによって提供される標準 SQL ステートメントを使用するか、あるいは定義したストアード・プロシージャーを使用します。ストアード・プロシージャーを実行して操作を実行することや、または操作の前または後でのカスタム処理を行うことができます。それぞれの操作の実行方法は、各ビジネス・オブジェクト内で構成できます。

Create 操作:

Create 操作は、要求内のビジネス・オブジェクトに対応したデータベース表に行を作成します。階層ビジネス・オブジェクトの場合は、Create 操作によってビジネス・オブジェクトが再帰的に全探索され、階層内の各ビジネス・オブジェクトに対応する行が作成されます。

Create 操作を処理するため、アダプターは次の操作を実行します。

1. ビジネス・オブジェクトがラッパーであるかどうかを確認します。最上位ビジネス・オブジェクトがラッパー・ビジネス・オブジェクトの場合、アダプターはこのビジネス・オブジェクトを無視します。ラッパー・オブジェクトの行は作成されません。
2. 所有関係を伴う単一カーディナリティーの各子ビジネス・オブジェクトを、データベース内に再帰的に挿入します。つまり、アダプターは、子ビジネス・オブジェクトおよびその子孫にあたるビジネス・オブジェクトのすべてを作成します。

   ビジネス・オブジェクト定義上、ある属性がある単一カーディナリティーの子ビジネス・オブジェクトを表すものとされている場合に、その属性が空であると、アダプターはその属性を無視します。ただし、ビジネス・オブジェクト定義により、その属性が子を表すことが必要であるにもかかわらず、子を表していない場合には、アダプターはエラーを戻して処理を停止します。
3. 所有関係を伴わない単一カーディナリティーの各子ビジネス・オブジェクトの有無を検索し、確認します。子がデータベース内に存在しないことを示して、検索が失敗した場合、アダプターはエラーを戻して処理を停止します。Retrieve 操作が成功した場合、アダプターは子ビジネス・オブジェクトを再帰的に更新します。

   注: データベースに子ビジネス・オブジェクトが存在する場合に、このアプローチが正しく機能するには、子ビジネス・オブジェクト内の基本キー属性の相互参照が、Create 操作時に正しく行われる必要があります。アプリケーション・データベースに子ビジネス・オブジェクトが存在しない場合、基本キー属性は設定してはいけません。
4. 最上位ビジネス・オブジェクトをデータベース内に挿入するため、次の操作を実行します。
   a. 最上位ビジネス・オブジェクトの各外部キー値を、対応する単一カーディナリティーの子ビジネス・オブジェクトの基本キー値に設定します。子ビジネス・オブジェクトの値は、データベース・シーケンスまたはカウンター、あるいはデータベース自体によって、子の作成時に設定される場合があります。そのため、このステップでは、アダプターが親をデータベースに挿入する前に、親の外部キー値を正しいものにします。
   b. データベースによって自動的に設定される属性のそれぞれに対して、新しい固有 ID 値を生成します。データベース・シーケンスまたはカウンターの名前は、属性のアプリケーション固有情報に格納されます。属性にデータベース・シーケンスまたはカウンターが関連付けられている場合、アダプターによって生成された値により、アプリケーション・サーバーから渡された値が上書きされます。
   c. 最上位ビジネス・オブジェクトをデータベース内に挿入します。

5. 複数カーディナリティーの子ビジネス・オブジェクトのすべてを、次のように処理します。
   a. それぞれの子の外部キー値を、親に含まれる対応する基本キー属性の値を参照するように設定します。親の基本キー値は、親の作成時に生成されている可能性があります。そのため、ここでは、アダプターが子をデータベースに挿入する前に、それぞれの子の外部キー値を正しいものにします。
   b. 複数カーディナリティーの子ビジネス・オブジェクトのすべてを、データベースに挿入します。

Retrieve 操作:

Retrieve 操作では、データベースからビジネス・オブジェクト階層のデータが抽出されます。

Retrieve 操作を処理するため、アダプターは次の操作を実行します。

1. 受信した最上位ビジネス・オブジェクトから、すべての子ビジネス・オブジェクトを削除します。つまり、子のない最上位ビジネス・オブジェクトのコピーを作成します。
2. 最上位ビジネス・オブジェクトを、データベース内で検索します。
   - 最上位ビジネス・オブジェクトがラッパー・ビジネス・オブジェクトの場合、これは無視されます。ラッパー・ビジネス・オブジェクトの検索は実行されません。
   - 検索の結果戻された行が 1 つの場合、アダプターは処理を継続します。
   - 検索の結果戻された行がない場合 (目的の最上位ビジネス・オブジェクトがデータベース内に存在しないことを意味します)、アダプターは `RecordNotFoundException` エラーを戻します。
   - 検索の結果戻された行が複数ある場合、アダプターは `MultipleMatchingRecordsException` エラーを戻します。

   Retrieve 操作は基本キーのみを使用します。他の列は無視されます。
3. 複数カーディナリティーの子ビジネス・オブジェクトのすべてを、再帰的に検索します。

   注: アダプターは、ビジネス・オブジェクトの配列を取り込むときに、一意性を保証しません。一意性の保証は、データベース側で行われなければなりません。データベースから戻された子ビジネス・オブジェクトに重複があると、アダプターは、それらの重複する子を戻します。
4. 子ビジネス・オブジェクトが所有関係にあるかどうかに関係なく、各単一カーディナリティーの子を再帰的に検索します。

   注: 単一カーディナリティーの子ビジネス・オブジェクトはすべて、ビジネス・オブジェクト内での出現順序に従って、親ビジネス・オブジェクトが処理される前に処理されます。子オブジェクトに対する所有関係の有無は、処理シーケンスを決定しませんが、処理のタイプは決定します。

NULL データの取得

アダプターは、データベース表で列値が NULL のレコードを取得できます。例えば、Customer ビジネス・オブジェクトに、custid、ccode、fname、および lname という列があり、custid と ccode が複合キーを形成しているとします。複合キーとは、複数の属性を参照する基本キーであり、ビジネス・オブジェクトの一意性を定義するときに使用されます。ccode が NULL の Customer レコードを取得できます。アダプターは Retrieve 操作の SELECT ステートメントを次のように生成します。

```
select custid, ccode, fname, lname from customer where custid=? and ccode is null
```

RetrieveAll 操作:

アダプターは、RetrieveAll 操作を使用してデータベースからビジネス・オブジェクトの配列を検索します。アダプターの処理は、RetrieveAll 操作がデータベース表ビジネス・オブジェクトとユーザー指定 SQL ビジネス・オブジェクトのいずれを対象としているかによって異なります。

データベース表ビジネス・オブジェクトの場合

着信ビジネス・オブジェクト内に取り込まれているすべてのキー属性および非キー属性によって、取得のための選択基準が決まります。選択した属性によっては、アダプターは、データベースから最上位ビジネス・オブジェクトの複数の行を検索する場合もあります。着信ビジネス・オブジェクト内に属性が取り込まれていない場合は、データベース内のそれぞれの表からすべての行が検索されます。

生成されたビジネス・オブジェクトの名前が、データベースの表の名前と一致しています。例えば、データベースの Customer 表は、「Customer」というビジネス・オブジェクトを表しています。

ビジネス・オブジェクトの配列を取得するため、アダプターは次の操作を実行します。

1. 取得したすべての行についてコンテナーを構成します。コンテナー・ビジネス・オブジェクトの名前は、ビジネス・オブジェクト名にストリング「Container」を付加したものです。
2. ビジネス・グラフを使用するようモジュールが構成された場合 (オプション)、取得した各行について最上位のビジネス・グラフを構成します。ビジネス・グラフの名前は、ビジネス・オブジェクト名にストリング 『「BG」』 を付加したものです。
3. Retrieve 操作を使用してコンテナー内の各ビジネス・グラフを取得します。

以下の図は、RetrieveAll 操作で戻されるオブジェクトの構造 (ビジネス・グラフがある場合とない場合) を示しています。

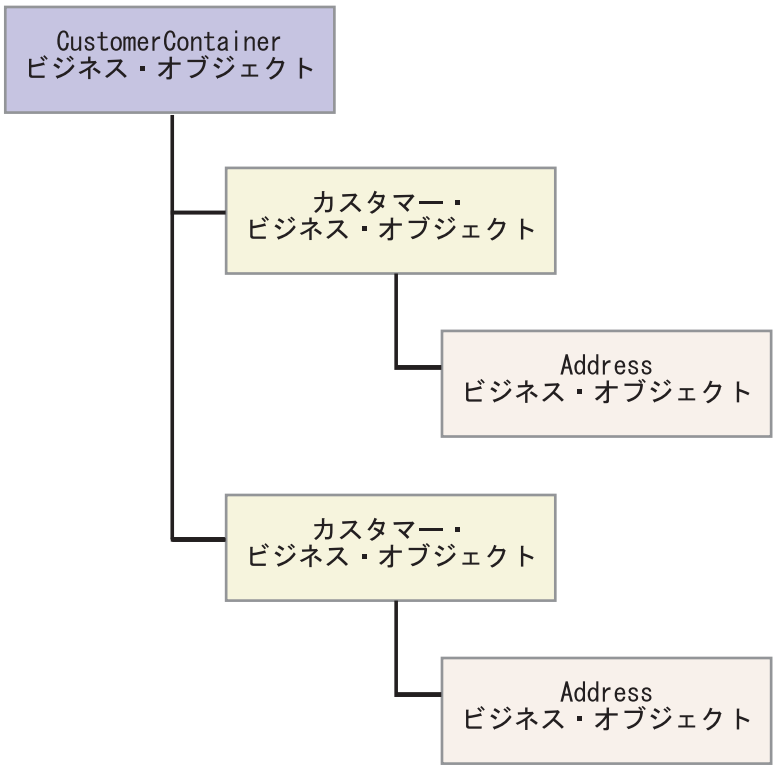


図 3. RetrieveAll 操作で戻されるビジネス・オブジェクトの構造 (オプションのビジネス・グラフなし)

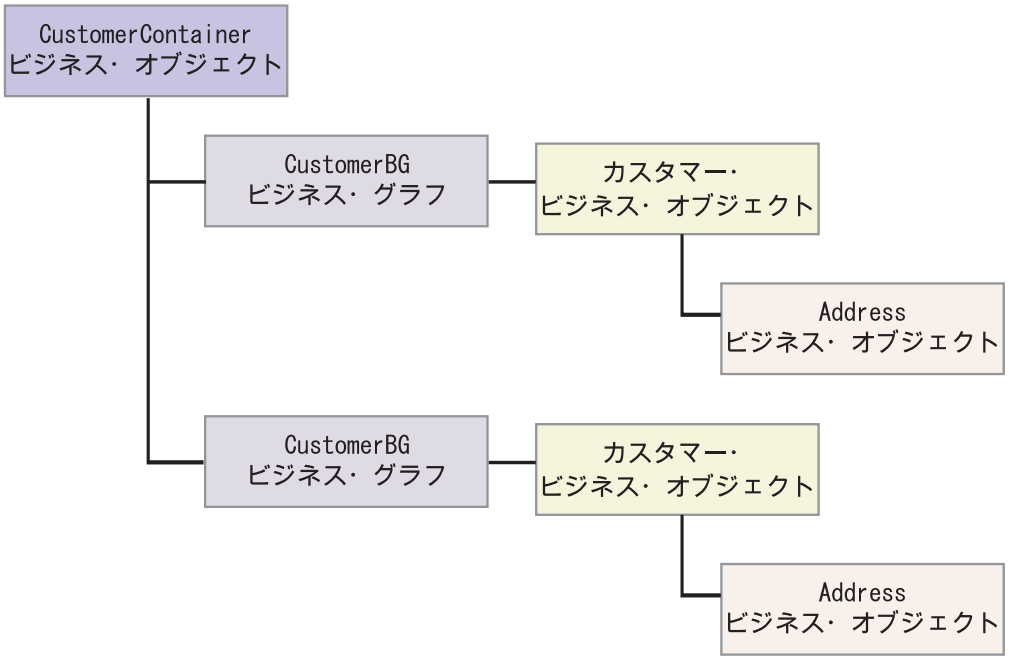


図 4. RetrieveAll 操作で戻されるビジネス・オブジェクトの構造 (オプションのビジネス・グラフあり)

RetrieveAll 操作によって、次のエラーが発生する可能性があります。

- `RecordNotFoundException` – 入力オブジェクトでデータが取り込まれている 1 つ以上のビジネス・オブジェクトが、エンタープライズ情報システムに存在していません。
- `MatchesExceededLimitException` – データベース内の一致するレコード数が、対話スペックで定義されている RetrieveAll 操作プロパティーの Maximum レコードの値を超えています。障害の MatchCount 属性には、アダプターがデータベースで検出した一致の実際の数が含まれているため、制限値を高くしたり、検索を適度に詳細化することができます。

  注: RetrieveAll 操作プロパティーの Maximum レコードを大きい数に設定すると、戻されるビジネス・オブジェクトのサイズと数によってはメモリー不足に関連する問題が発生する場合があります。
- `EISSystemException` – データベース (エンタープライズ情報システム (EIS)) から 1 つ以上の回復不能エラーが報告されました。

クエリー・ビジネス・オブジェクトの場合

ユーザー指定 SELECT ステートメント (クエリー・ビジネス・オブジェクト) に対して作成されたビジネス・オブジェクトも RetrieveAll 操作をサポートします。外部サービス・ウィザードは、ユーザー指定の SQL SELECT ステートメントを実行して、クエリー・ビジネス・オブジェクトの階層を作成することによって、クエリー・ビジネス・オブジェクトを生成します。オプションでビジネス・グラフを使用する場合、階層は図 5に示すようになります。

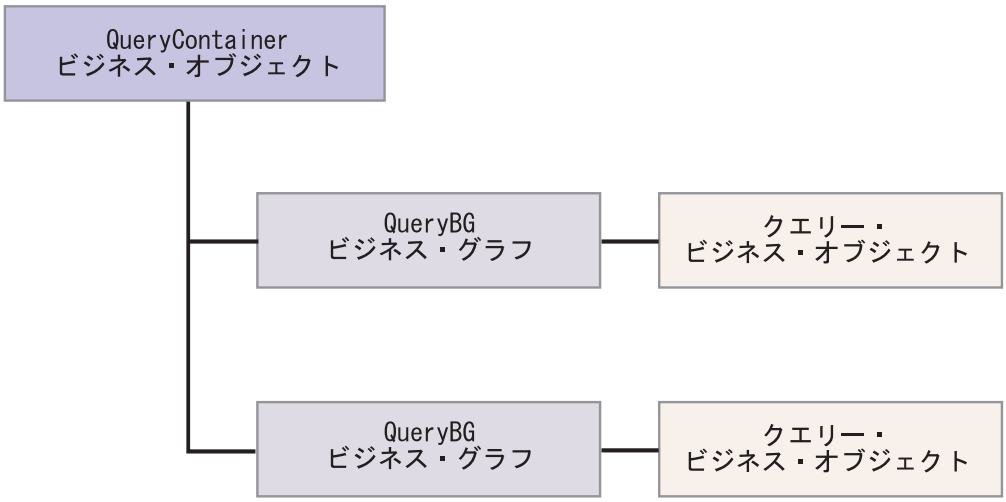


図 5. ユーザー指定のクエリー・ビジネス・オブジェクト

オプションでビジネス・グラフを使用しない場合、階層は 15 ページの図 6に示すようになります。

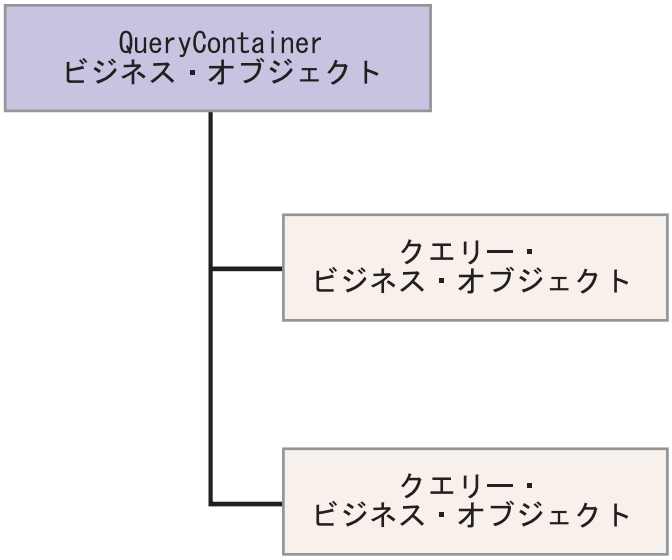


図 6. ユーザー指定のクエリー・ビジネス・オブジェクト

外部サービス・ウィザードによってユーザー指定 SELECT ステートメントに対して生成されたクエリー・ビジネス・オブジェクトを処理するために、アダプターは以下の操作を実行します。

1. クエリー・ビジネス・オブジェクトから SELECT SQL ステートメントを取得します。
2. クエリー・ビジネス・オブジェクトで動的 WHERE 文節が指定されているかどうかを検査します。
   - 動的 WHERE 文節がある場合、アダプターは、SELECT ステートメントのデフォルト WHERE 文節をその動的 WHERE 文節で置換します。
   - 動的 WHERE 文節がない場合、アダプターは、SELECT ステートメントのパラメーターをクエリー・ビジネス・オブジェクトで指定された対応する値で置換します。
3. SELECT ステートメントを実行します。
4. 返された結果セットを取得し、クエリー・ビジネス・オブジェクト値にデータベースから返されたデータを設定し、 14 ページの図 5に示した構造のコンテナー・ビジネス・オブジェクトを作成します。
5. クエリー・ビジネス・オブジェクトに子ビジネス・オブジェクトが定義されている場合、コンテナー内の各最上位クエリー･ビジネス・オブジェクトからなる階層全体を取得します (*下降検索*)。

   注: クエリー・ビジネス・オブジェクトを最上位ビジネス・オブジェクト以外にすることはできません。クエリー・ビジネス・オブジェクトが子クエリー・ビジネス・オブジェクトを持つことはできません。

NULL オブジェクトの取得

アダプターは、列値が NULL のレコードをデータベース表から取得できます。例えば、Customer ビジネス・オブジェクトに custid、ccode、fname、および lname という列があり、ccode は基本キーである必要はないとします。ccode が NULL の

Customer レコードをすべて検索する照会を実行できます。アダプターは、RetrieveAll 操作の選択照会を次のように生成します。

```
select custid, ccode, fname, lname from customer where custid=? and ccode is NULL
```

Update 操作:

Update 操作は、ソース・ビジネス・オブジェクトを、最上位のソース・ビジネス・オブジェクトで指定された基本キーを使用してデータベースから検索されたビジネス・オブジェクトと比較することによって実行されます。

階層ビジネス・オブジェクトの更新時に、アダプターは次の操作を実行します。

1. ソース・ビジネス・オブジェクトの基本キー値を使用して、データベース内の対応するエンティティーを検索します。検索されたビジネス・オブジェクトは、データベース内のデータの現在の状態を正確に表したものです。

   検索が失敗した場合 (最上位ビジネス・オブジェクトがデータベース内に存在しないことを意味します)、アダプターは `RecordNotFoundException` エラーを戻し、更新は失敗します。

   検索に成功した場合、アダプターは、検索されたビジネス・オブジェクトをソース・ビジネス・オブジェクトと比較して、どの子ビジネス・オブジェクトに関してデータベースに変更を加える必要があるかを判別します。ただし、アダプターはソース・ビジネス・オブジェクトの単純属性の値と検索されたビジネス・オブジェクトの単純属性の値を比較しません。アダプターは、非キーの単純属性すべての値を更新します。

   最上位ビジネス・オブジェクトのすべての単純属性がキーを表している場合、アダプターはその最上位ビジネス・オブジェクト用の更新照会を生成できません。この場合、アダプターは、警告を記録してから次に進みます。
2. 最上位ビジネス・オブジェクトの子のうち、単一カーディナリティーのものすべてを再帰的に更新します。

   ビジネス・オブジェクト定義上、ある属性がある子ビジネス・オブジェクトを表すことが必須である場合には、その子ビジネス・オブジェクトがソース・ビジネス・オブジェクトと検索されたビジネス・オブジェクトの両方に存在している必要があります。存在しない場合、Update 操作は失敗し、アダプターはエラーを戻します。

   アダプターでは、所有関係にある単一カーディナリティーの子を、次のいずれかの方法で処理します。
   - ソース・ビジネス・オブジェクトおよび検索したビジネス・オブジェクトの両方に子が存在する場合、アダプターは、データベース内の既存の子を更新するのではなく、既存の子を削除して新規の子を作成します。
   - その子がソース・ビジネス・オブジェクトには存在するにもかかわらず、検索されたビジネス・オブジェクトには存在しない場合、アダプターはデータベース内にその子を再帰的に作成します。
   - その子が検索されたビジネス・オブジェクトには存在するにもかかわらず、ソース・ビジネス・オブジェクトには存在しない場合、アダプターはデータベース内のその子を再帰的に削除します。

所有関係にない単一カーディナリティーの子に関しては、アダプターは、ソース・ビジネス・オブジェクトに存在するそのような子のすべてを、データベースから検索しようとします。アダプターは、子の検索に成功すると、その子ビジネス・オブジェクトにデータを取り込みますが、更新は行いません。これは、所有関係にない単一カーディナリティーの子はアダプターによって変更されることがないためです。

3. 検索されたビジネス・オブジェクトのすべての単純属性を更新します。ただし、ソース・ビジネス・オブジェクト内の対応する属性が指定されていない場合を除きます。

   更新されるビジネス・オブジェクトは一意である必要があるため、アダプターは、結果として 1 行のみが処理されることを確認します。複数の行が戻されている場合、アダプターはエラーを戻します。

   最上位ビジネス・オブジェクトがラッパー・ビジネス・オブジェクトの場合、これは無視されます。ラッパー・ビジネス・オブジェクトの更新は実行されません。

4. 検索されたビジネス・オブジェクトの複数カーディナリティーの子のそれぞれを、次のいずれかの方法で処理します。
   - その子がソース・ビジネス・オブジェクトの配列と検索されたビジネス・オブジェクトの配列の両方に存在する場合、アダプターはデータベース内でその子を再帰的に更新します。
   - その子がソース・ビジネス・オブジェクトの配列には存在しても、検索されたビジネス・オブジェクトの配列には存在しない場合、アダプターはデータベース内でその子を再帰的に作成します。
   - その子が検索されたビジネス・オブジェクトの配列には存在しても、ソース・ビジネス・オブジェクトの配列には存在しない場合、アダプターはデータベースからその子を再帰的に削除します。ただし、親に含まれているその子を表す属性のアプリケーション固有情報で、KeepRelationship プロパティーが `true` に設定されている場合を除きます。この場合、アダプターは、データベースからその子を削除しません。

NULL データと Update 操作

アダプターは、データベース表で列値が NULL のレコードを更新できます。例えば、Customer ビジネス・オブジェクトに、custid、ccode、fname、および lname という列があり、custid と ccode が複合キーを形成しているとします。複合キーとは、複数の属性を参照する基本キーであり、ビジネス・オブジェクトの一意性を定義するときに使用されます。ccode が NULL の Customer レコードを更新できます。アダプターにより、Update 操作の更新照会が次のように生成されます。

```
update customer set fname=?, lname=? where custid=? and ccode is null
```

ApplyChanges 操作:

ApplyChanges 操作では、ビジネス・オブジェクトの変更または削除のための差分および変更後イメージをサポートします。ApplyChanges 操作は、ビジネス・グラフを使用する場合にのみ使用可能です。

ビジネス・グラフの verb プロパティーを、create、update、delete などの操作の名前に設定した場合、アダプターは ApplyChanges 操作について変更後イメージ処理を実行します。例えば、verb を `create` に設定した場合、アダプターは ApplyChanges 操作を Create 操作と同様に処理します。

ビジネス・グラフで verb を設定しない場合、アダプターはビジネス・グラフの ChangeSummary を使用してビジネス・オブジェクトを更新します。このモードでは、ApplyChanges 操作は以下の点で Update 操作と異なります。

- ApplyChanges 操作では、更新の前に Retrieve 操作は実行されません。
- 着信ビジネス・オブジェクトとデータベース内のビジネス・オブジェクトの比較が行われません。
- 子はすべて、各子ビジネス・オブジェクトの ChangeSummary に設定されている操作に基づいて処理されます。子に操作が設定されていない場合、アダプターはエラーを戻します。

アダプターは、ChangeSummary からの階層ビジネス・オブジェクトの更新時に、以下のステップを実行します。このステップでは、ChangeSummary から、変更内容のみが処理されます。

1. 親オブジェクトの子のうち、単一カーディナリティーのものすべてを再帰的に処理します。ビジネス・オブジェクト仕様で必須にマークされている子は、Inbound オブジェクトに存在していなければなりません。存在しない場合、ApplyChanges 操作は失敗し、アダプターがエラーを戻します。
2. 親に含まれる外部キー値のうち、単一カーディナリティーの子の属性を参照するものすべてを、それぞれ対応する子の値に設定します。この処理が必要なのは、これ以前のステップで、単一カーディナリティーの子がデータベースに追加され、新しいシーケンス値が生成されている可能性があるためです。
3. 現在処理中のオブジェクトを、SQL UPDATE ステートメントまたはストアード・プロシージャーを使用して更新します。個々のビジネス・オブジェクトのすべての単純属性が更新されます。アダプターは UPDATE ステートメントにどの属性を追加しなければならないかを判断するのに、プロパティー・レベルの変更を使用しません。すべて更新されます。更新されているオブジェクトは固有であるため、アダプターは結果として 1 行のみが処理されていることをチェックします。複数の行が処理される場合、エラーが戻されます。
4. 現在のオブジェクトのカーディナリティー N のすべての子にある、親の属性を参照する外部キー値をすべて、対応する親の値に設定します。通常、これらの値はデータ・マッピング時に既に相互参照されています。ただし、これはカーディナリティーが N のコンテナーに含まれる新しい子には該当しない場合があります。このステップにより、カーディナリティーが N の子すべての外部キー値が正しい値になってから、それらの子の更新が行われることが徹底されます。
5. 現在のオブジェクトの、カーディナリティーが N のコンテナーをすべて更新します。

   子オブジェクトが処理されるときには、それぞれの子の操作が取得され、適切な操作が実行されます。ApplyChanges で子に対して許可される操作は、Create、Delete、および Update です。

- Create 操作が子で検出された場合、それが所有関係にある子であれば、検出された子がデータベース内に作成されます。所有関係にない子に関しては、検索により、データベースに存在するかどうかが確認されます。
- Delete 操作が子で検出された場合、子は削除されます。
- Update 操作が子で検出された場合、子はデータベース内で更新されます。

Delete 操作:

Delete 操作は、データベースからの着信ビジネス・オブジェクトのプルーニングと、その後の完全なビジネス・オブジェクトの検索によって実行されます。 Delete 操作は、その後階層内の各ビジネス・オブジェクトについて再帰的に適用されます。

Delete 操作では、ビジネス・オブジェクトのアプリケーション固有情報の StatusColumnName の値に応じて、物理削除と論理削除がサポートされます。ステータス列名の値が定義されている場合、アダプターは論理削除操作を実行します。ステータス列名の値が定義されていない場合、アダプターは物理削除操作を実行します。

物理削除

物理削除の場合、アダプターは次の処理を行います。

- 複数カーディナリティーの子ビジネス・オブジェクトすべてを再帰的に削除します。
- 最上位ビジネス・オブジェクトを削除します。

  最上位ビジネス・オブジェクトがラッパー・オブジェクトの場合、これは無視されます。ラッパー・ビジネス・オブジェクトの削除は実行されません。
- 所有関係にある単一カーディナリティーの子ビジネス・オブジェクトすべてを、再帰的に削除します。

論理削除

論理削除の場合、アダプターは次の処理を行います。

- ビジネス・オブジェクトの状況属性を、ビジネス・オブジェクト・レベルのアプリケーション固有情報によって指定されている値に設定する Update を発行します。アダプターでは、結果として 1 つのデータベース行だけが更新されることを確認します。それ以外の場合は、エラーを戻します。
- 所有関係にある単一カーディナリティーの子のすべて、および複数カーディナリティーの子のすべてに対し、論理削除を再帰的に実行します。アダプターは、所有関係にない単一カーディナリティーの子は削除しません。

NULL データと Delete 操作

アダプターは、データベース表で列値が NULL のレコードを削除できます。例えば、Customer ビジネス・オブジェクトに、custid、ccode、fname、および lname という列があり、custid と ccode が複合キーを形成しているとします。複合キーとは、複数の属性を参照する基本キーであり、ビジネス・オブジェクトの一意性を定

義するときに使用されます。ccode が NULL の Customer レコードを削除できます。アダプターは、Delete 操作の削除照会を次のように生成します。

```
delete from customer where custid=? and ccode is null
```

Execute 操作:

ストアード・プロシージャーおよびストアード関数を実行するときに Execute 操作が使用されます。外部サービス・ウィザードは、データベースのストアード・プロシージャーまたはストアード関数の定義に対応する必要なストアード・プロシージャー・ビジネス・オブジェクトを生成します。アダプターは、Execute 操作を使用してストアード・プロシージャー・ビジネス・オブジェクトを処理します。

ストアード・プロシージャー、そのストアード・プロシージャーから構成されるビジネス・オブジェクト、および Execute 操作でストアード・プロシージャー・ビジネス・オブジェクトを処理するためにアダプターが使用するステップの単純な例を以下に示します。

ストアード・プロシージャーの単純な例:

```
PROCEDURE testSP(IN int x,INOUT VARCHAR(10) msgSTR, OUT int status,
                 OUT struct outrec, OUT array retArr)
```

このプロシージャーは 2 つの結果セットを戻します。

このストアード・プロシージャーの場合に構成されるビジネス・オブジェクトの例を以下に示します。

```
BOLevel ASI
      SPName=testSP
      ResultSet=true
      MaxNumberOfResultSets=2
      ReturnValue = propName
                    ストアード・プロシージャーが関数の場合に戻されます。
                    戻り値が複合型 (array/struct/resultset) である場合は、
                    子ビジネス・オブジェクトに対応するプロパティー名になります。
          関数の場合にのみ定義されます。

プロパティー
      x Type=IP
      msgStr Type=IO
      status Type=OP
      outrec Type OP - outrec の子 BO、ASI ChildBOType = struct
      retarr Type OP - retArr の n カーディナリティー子 BO、ASI ChildBOType = array
      childBOName1 - 最初の結果セットに対する子 BO、ASI ChildBOType = resultset
      childBOName2 - 2 番目の結果セットに対する子 BO、ASI ChildBOType = resultset
```

Execute 操作でこのストアード・プロシージャー・ビジネス・オブジェクトを処理するために、アダプターは以下の処理を行います。

1. ストアード・プロシージャー呼び出し CALL testSP(x, msgStr, status, outrec) を構成します。
2. 呼び出し可能ステートメントで入力パラメーター x および msgStr を設定します。
3. 呼び出し可能ステートメントを実行します。
4. 戻り値を取得し (関数の場合)、それがスカラー値であればその値を適切な属性に設定します。複合値 (構造体や配列など) であれば子ビジネス・オブジェクトに設定します。

5. 最初の結果セットを取得し、ResultSet1 のコンテナーを作成します。
6. 2 番目の結果セットを取得し、ResultSet2 のコンテナーを作成します。
7. 出力パラメーター msgStr および status を取得し、ビジネス・オブジェクトで対応する属性を設定します。
8. 出力パラメーター outrec を取得し、outrec で返されたデータから子ビジネス・オブジェクトを作成します。outrec がネストされた構造体型である場合、アダプターは階層子ビジネス・オブジェクトを再帰的に作成してデータを格納します。
9. 出力パラメーター retArr を取得し、retArr で返されたデータから複数のカーディナリティーの子ビジネス・オブジェクトを作成します。retArr がネストされた配列型である場合、アダプターは階層子ビジネス・オブジェクトを再帰的に作成してデータを格納します。

Inbound 処理

アダプターは、イベント送達を行う Inbound イベント管理をサポートします。イベントは、データベース・アプリケーションか、またはユーザーが指定したカスタム照会の結果のいずれかによってデータが取り込まれたイベント・ストアで処理されます。アダプターによるイベントのポーリング頻度と、1 回あたりのエクスポートへの送信レコード数を制御できます。

アダプターは、次のいずれかの方法で変更確認のためのポーリングを実行します。

- 標準イベント処理。アダプターはイベント・ストアを調べ、データベース・アプリケーションにより格納されたイベントがあるかどうかを確認します。
- カスタム・イベント処理。アダプターはユーザー定義の照会、ストアード・プロシージャー、またはストアード関数を実行します。

最初に外部サービス・ウィザードを使用してアダプターを構成するときに標準またはカスタムのイベント処理をカスタマイズできます。または、後でサーバーの管理コンソールを使用してアクティベーション・スペック・プロパティーを変更することによってもイベント処理をカスタマイズすることができます。

イベントの対象であるデータベース・オブジェクトは、通知がエクスポートに送信された後に取得されます。この結果、発生した取得エラーの検出と通知は、エクスポートの通知の完了後まで据え置かれます。これは、アダプターがエクスポートに通知する前に取得エラーを検出可能な、バージョン 6.0.2 のアダプターでのイベント処理とは異なります。

標準イベント処理

標準イベント処理では、アダプターはイベントをポーリングする SQL 照会を指定して、イベントが一度で確実に送達されるようにします。

Oracle Change Data Capture などのデータベース・トリガーまたはツールは、データベース内のテーブルのレコードが作成、更新、または削除されたときに実行します。トリガーまたはその他のツールはイベント・レコードをイベント・ストアに書き込みます。イベント・ストアは、ポーリング・アダプターがイベント・レコードを処理できるまでイベント・レコードが保存される永続キャッシュです。イベン

ト・ストアはユーザー・テーブルと同じデータベース内にテーブルとして実装されます。これはアダプターによってアクセスされるデータベース・オブジェクトを含むテーブルです。

トリガーを定義するか、または他のツールをセットアップして、イベントの受け取りが必要な対象のデータベース表への変更をレポートする必要があります。アダプターには、アダプターのトリガーのセットアップ方法を示すサンプル・データベース・スクリプトがあります。サンプルは *WID_installation_dir*/ResourceAdapters/JDBC_*version*/samples/scripts ディレクトリーにあります。ここで *version* はアダプターのバージョン (`6.1.0.0_IF1` など) を識別します。 IBM DB2、IBM DB2 for z/OS®、Oracle、および Microsoft SQL Server に対応したサンプル・スクリプトが用意されています。

アダプターは「送達は 1 回のみ」を提供しています。これは、各イベントはエクスポートに 1 回だけ送達されることを保証するものです。モジュールについて「送達は 1 回 のみ」を有効にした場合、イベント・ストア内の各イベントにトランザクション ID (XID) が設定されます。イベントが処理対象として取得されると、イベント・ストア内でそのイベントの XID 値が更新されます。さらに、イベントが対応するエクスポートに送達され、その後イベント・ストアから削除されます。イベントが送達される前に、データベース接続が失われたか、アプリケーションが停止した場合、イベントは完全に処理されない可能性があります。この場合、イベントの再処理とエクスポートへの再送信が必要であることが XID 列によって示されます。データベース接続が再確立されるか、またはアダプターが再始動されると、アダプターは、イベント・ストアで XID 列に値を持つイベントがあるかどうかをチェックします。アダプターは、まずこれらのイベントを処理してから、ポーリング周期の間にその他のイベントをポーリングします。

アダプターは、すべてのイベントを処理するか、またはビジネス・オブジェクト・タイプによってイベントをフィルター操作できます。フィルターは、アクティベーション・スペック・プロパティー EventFilterType を使用して設定します。このプロパティーは、ビジネス・オブジェクト・タイプをコンマで区切ったリストを持ちます。プロパティーで指定されたタイプのみが処理されます。プロパティーに値が指定されていない場合、フィルターは適用されず、すべてのイベントが処理されます。アクティベーション・スペック・プロパティー FilterFutureEvents が true に設定されている場合、アダプターは、タイム・スタンプに基づいてイベントをフィルターに掛けます。アダプターは、各ポーリング周期のシステム時刻を各イベントのタイム・スタンプと比較します。イベントが将来発生するように設定されている場合は、その時刻になるまで処理されません。

カスタム・イベント処理

カスタム・イベント処理では、イベントをポーリングする SQL 照会またはストアード・プロシージャーを提供します。

カスタム・イベント処理を使用して、標準のイベント処理でイベント・ストアをポーリングするために使用する SQL 照会の代わりに、アダプターで実行するデータベース照会 (*カスタム・イベント照会*) を提供することによって、どのイベントをエクスポートに送達するかを制御します。カスタムのイベント照会では、必要なフィルタリングを実行する必要があります。カスタム・イベント処理が必要な場合、ウ

ィザードでオプションを選択するか、管理コンソールの EventQueryType アクティベーション・スペック・プロパティーを設定して指定します。

XID 値を格納するための標準イベント・ストアを作成した場合、カスタム・イベント処理は「送達は 1 回のみ」をサポートします。 アダプターは、カスタム・イベント照会によって返されたイベントをイベント・ストアに格納し、そのイベントを XID 値で更新します。アダプターは、標準のイベント処理と同じ方法でイベントを処理します。標準イベント・ストアを照会するカスタム照会は作成しないでください。これはアダプターが「送達は 1 回のみ」に構成されている場合、このテーブルにイベントが格納されるのは一時的であるためです。 さらにこの状況では、アダプターが、カスタム照会から取得したイベント ID 値をイベント・ストアに取り込むため、イベント・ストアではイベント ID 値の自動生成が行われてはいけないことになります。

カスタム・イベント処理を有効にするには、アダプターを使用するようにモジュールを構成するときウィザードで拡張オプションを選択するか、EventQueryType アクティベーション・スペック・プロパティーを設定します。

カスタム・イベント照会

カスタム・イベント照会の実行の指定は、ウィザードの拡張オプションでユーザー定義のイベント照会を提供するか、CustomEventQuery アクティベーション・スペック・プロパティーを設定して行います。次のプログラム・タイプのいずれかを指定してください。

- 標準の SQL ステートメント
- ストアード・プロシージャー
- ストアード関数

これらのプログラムはすべて、入力パラメーターとしてポーリング数量 (アダプターが実行時に提供するアクティベーション・スペック・プロパティー) を取ります。プログラムは他の入力パラメーターも受け取ることができます。これらのプログラムは、結果セットを戻す必要があります。この結果セットは、ポーリング数量に相当する数のレコードを含み、列 event_id、object_key、object_name、および object_function をこの順序で含んでいます。アダプターは結果セットからイベント・オブジェクトを生成し、イベントを処理します。

標準の SQL ステートメント

処理するイベントを選択する SQL SELECT ステートメントを指定できます。照会には、ポーリング数量の入力パラメーターとその他の入力パラメーターを指定できます。

ストアード・プロシージャー

カスタム照会は、ポーリング数量を入力として受け取り、結果セットのタイプの出力パラメーターを戻すストアード・プロシージャーである場合があります。ストアード・プロシージャーを指定するときには、次の構文を使用してください。

```
call procedure_name (?, ?)
```

ここで *procedure_name* は、実行する必要のあるストアード・プロシージャーの名前です。最初のパラメーターはポーリング数量を表し、2 番目のパラメーターは結果セットを表します。

ストアード・プロシージャーは、その他の入力パラメーターを受け入れることもでき、以下のように call ステートメント自体でそれらを提供します。

```
call procedure_name (25, ?, ?)
```

ストアード関数

カスタム照会は、ポーリング数量を入力として受け取り、結果セットを戻すストアード関数である場合もあります。ストアード関数を指定するときには、次の構文を使用してください。

```
? = call function_name (?)
```

ここで *function_name* は、実行する必要のあるストアード関数の名前です。最初のパラメーターは結果セットを表し、2 番目のパラメーターはポーリング数量を表します。

ストアード関数は、その他の入力パラメーターを受け入れることもでき、以下のように call ステートメント自体でそれらを提供します。

```
? = call function_name (?, 'abc')
```

カスタム更新照会およびカスタム削除照会

カスタム・イベント処理では、カスタム更新/削除照会を提供することができます。これらの照会は、各イベントの処理後に実行されます。ユーザーは通常、*更新照会* を使用して、データベース・レコードが以降のポーリング周期中に処理対象として取り出されないようにします。*削除照会* は、各イベントの処理後にデータベース・レコードを削除する必要がある場合に使用します。更新照会と削除照会は、どちらもオプションです。

更新照会と削除照会はそれぞれ CustomUpdateQuery および CustomDeleteQuery アクティベーション・スペック・プロパティーで指定されます。これらの照会は、標準 SQL ステートメント、ストアード・プロシージャー、またはストアード関数として入力できます。カスタム更新/削除照会の構文は、カスタム照会の構文と同じです。更新/削除照会は、イベント ID の入力パラメーターを取ります。アダプターは、実行時にイベント ID の値を提供します。照会には、これ以外の入力パラメーターを指定できます。このような入力パラメーターは、カスタム・イベント照会で説明した方法と同様に、照会構文自体に指定します。

イベント・ストア

イベント・ストアは、ポーリング・アダプターがイベント・レコードを処理できるまでイベント・レコードが保存される永続キャッシュです。アダプターは、Inbound 要求がシステム内を進行するときに、イベント・ストアを使用して Inbound 要求を追跡します。データベース・レコードの作成、更新、または削除が行われるたびに、アダプターはイベント・ストアのイベントの状況を更新します。各イベントの状況は、イベントがサーバーで構成済みエクスポートに渡されるまで、リカバリーの目的のために、アダプターによって継続的に更新されます。

アダプターは、イベント・ストアから定期的な間隔でイベント・レコードをポーリングします。ポーリング呼び出しごとに、多数のイベントがアダプターにより処理されます。イベントの処理順序は、優先順位の昇順およびイベント・タイム・スタンプの昇順です。各ポーリング周期において、アダプターはすべての新規イベントをピックアップします。新規イベントごとに、アダプターはイベントのオブジェクト・キー・フィールドに設定されている値を取得し、オブジェクト名フィールドに指定されている値に対応するビジネス・オブジェクトをロードします。オブジェクトのロード後、アダプターはオブジェクト・キー・フィールドに指定されている値に基づいて、ビジネス・オブジェクトの基本キー値を設定します。キーの設定後、アダプターはそれらのキーに基づいてオブジェクトの検索を実行します。取得された情報からビジネス・オブジェクトまたはオプションのビジネス・グラフが作成され、 エクスポートにパブリッシュされます。

ストアード・プロシージャーをビジネス・オブジェクトの RetrieveAll 操作に関連付けている場合は、ストアード・プロシージャーの入力パラメーターとビジネス・オブジェクト属性 (通常は基本キー) の間のマッピングを定義できます。そうしたマッピングを定義すると、アダプターは、ストアード・プロシージャーの入力パラメーターを設定し、ストアード・プロシージャーを呼び出し、ストアード・プロシージャーから得られた結果に基づいてオブジェクトにデータを取り込みます。

ストアード・プロシージャーとストアード関数の場合、RetrieveSP アプリケーション固有情報を使用して、ストアード・プロシージャー/ストアード関数の入力パラメーターと、ビジネス・オブジェクト属性 (一般に基本キーを使用) の間にマッピングを定義していると、アダプターはストアード・プロシージャーの入力パラメーターを設定し、ストアード・プロシージャーを呼び出し、ストアード・プロシージャーの結果に基づいてビジネス・オブジェクトにデータを取り込みます。

object_function 列の値が `Delete` で、オブジェクトが削除されたことを示す場合、そのオブジェクトはデータベースから取得されません。キーがデータ・オブジェクトとビジネス・オブジェクトに設定されるか、またはオプションのビジネス・グラフが作成されてエクスポートに送達されます。

イベントの通知が正常に完了すると、イベント・ストアからその項目が削除されます。失敗したイベントの場合は、エントリーがイベント・ストアに残り、event_status 列が -1 に設定されます。

表 2 に、イベント・ストアの表形式と内容の説明を示します。表 2.

表 2. イベント・ストア・データベース表の定義

列名	型	説明
XID	String	送達が 1 回のみの場合の固有トランザクション ID (XID) 値
event_id	Number	テーブルの基本キーである固有イベント ID。object_key と同じ値を設定できます。

表 2. イベント・ストア・データベース表の定義 (続き)

列名	型	説明
object_key	String	取得されるイベント・ストアのレコードのキーが含まれているストリング。 この列を NULL にすることはできません。 値を、1 つ以上の *key=value* ペアとして指定します。各ペアはセミコロン (;) で区切ります。 あるいは、基本キーの値のみをセミコロン (;) で区切って指定することもできます。この場合、ビジネス・オブジェクトでの基本キーの定義順序と同じ順序で値を指定する必要があります。
object_name	String	ビジネス・オブジェクトまたはビジネス・グラフの名前。ビジネス・オブジェクト (またはビジネス・グラフ内のビジネス・オブジェクト) は、階層ビジネス・オブジェクトであっても構いません。各ビジネス・オブジェクトまたはビジネス・グラフは、テーブルまたはビューを参照します。 この列を NULL にすることはできません。
object_function	String	イベントに対応した操作 (Delete、Create、Update など)。 この列を NULL にすることはできません。
event_priority	Number	イベント優先順位を識別します。この値は正整数でなければなりません。 この列を NULL にすることはできません。
event_time	Timestamp	イベントが生成された日時。形式は mm/dd/yyyy hh:mm:ss です。
event_status	Number	イベント状況。これは、新規イベントに最初に設定される値であり、アダプターがイベント処理時に更新します。状況の値は次のいずれかです。 • 0: 新規イベントを示します。 • 1: エクスポートに送達されたイベントを識別します。 • -1: イベント処理中にエラーが発生しました。 この列を NULL にすることはできません。
event_comment	String	イベントに関連付けられているコメント

ビジネス・オブジェクト

ビジネス・オブジェクトとは、データ、データ上で実行されるアクション、およびデータを処理するための追加の指示 (存在する場合) で構成される構造体のことです。WebSphere Adapter for JDBC は、ビジネス・オブジェクトを使用して、データベースのテーブルとビュー、データベース照会、ストアード・プロシージャー、およびストアード関数の結果を表現します。ビジネス・オブジェクトにより、データベースのオブジェクトの階層を作成し、無関係なテーブルをグループ化できます。コンポーネントはビジネス・オブジェクトを使用してアダプターと通信します。

アダプターによるビジネス・オブジェクトの使用法

統合アプリケーションは、ビジネス・オブジェクトを使用してデータベースにアクセスします。アダプターは、Outbound 要求のビジネス・オブジェクトを、データベースへアクセスするための JDBC API 呼び出しに変換します。Inbound イベントの場合、アダプターはイベントのデータをビジネス・オブジェクトに変換し、このビジネス・オブジェクトがアプリケーションに戻されます。

アダプターは、ビジネス・オブジェクトを使用してデータベース内の次のタイプのオブジェクトを表現します。

- テーブルとビュー
- シノニムとニックネーム
- ストアード・プロシージャーとストアード関数

一部のビジネス・オブジェクトは、データベース・オブジェクトを表現しません。以下のものがあります。

- バッチ SQL ビジネス・オブジェクト。このオブジェクトは、一連のユーザー定義の insert ステートメント、update ステートメント、および delete ステートメントを表します。
- クエリー・ビジネス・オブジェクト。これは、データベースに対して実行するユーザー定義 SQL 照会を表します。
- ラッパー・ビジネス・オブジェクト。このオブジェクトにより、無関係なテーブル・オブジェクトとビュー・オブジェクトを 1 つのビジネス・オブジェクトにグループ化できます。

ビジネス・オブジェクト内でのデータの表現方法

テーブルまたはビュー・ビジネス・オブジェクトの場合

テーブルまたはビューの各列は、テーブル・ビジネス・オブジェクトまたはビュー・ビジネス・オブジェクトの単純属性により表現されます。*単純属性* とは、String、Integer、または Date などの単一値を表す属性です。その他の属性は、子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表します。

同じビジネス・オブジェクトに含まれる単純属性を、別々のデータベース表に格納することはできませんが、次の状況が考えられます。

- データベース表に、対応するビジネス・オブジェクトに含まれる単純属性の数よりも多くの列が含まれる場合があります。つまり、データベースの列の一部が、ビジネス・オブジェクト内に表されていません。アプリケーションによるビジネス・オブジェクトの処理で必要な列のみを実際の設計に含めるようにします。
- ビジネス・オブジェクトに、対応するデータベース表に含まれる列の数よりも多くの単純属性が含まれる場合があります。つまり、ビジネス・オブジェクト内の属性の一部が、データベース内に表されていません。データベース内に列を持たない属性は、アプリケーション固有情報を持っていないか、デフォルト値が設定されているか、またはストアード・プロシージャーかストアード関数のパラメーターです。
- ビジネス・オブジェクトは、複数のデータベース表にまたがるビューを表すことができます。アダプターでは、Create、Update、および Delete 操作など、データ

ベースに対する変更によって起動されるイベントを処理するときに、このようなビジネス・オブジェクトを使用できます。ただし、ビジネス・オブジェクトの要求を処理する場合には、Retrieve および RetrieveAll 要求に対してのみ、このようなビジネス・オブジェクトを使用できます。

テーブル・ビジネス・オブジェクトには、対応するデータベース表に基本キーがない場合でも、常に基本キーが設定されています。アダプターは、テーブル・ビジネス・オブジェクトを取得するときに、基本キー属性で指定される列を使用します。アダプターは、複合の、つまり複数の基本キーが設定されている表をサポートします。 データベースに基本キーが 1 つ以上存在する場合、ウィザードは、テーブル・ビジネス・オブジェクトのそれらの列に基本キー・プロパティーを設定します。データベース表に基本キーが存在しない場合、外部サービス・ウィザードでは、そのビジネス・オブジェクトを構成するときに基本キー情報の入力を求めるプロンプトが出されます。シーケンスや ID 列などの固有データを含む列を指定してください。

テーブル・ビジネス・オブジェクトおよびビュー・ビジネス・オブジェクトは、Create、Update、Delete、Retrieve、RetrieveAll、および ApplyChanges Outbound 操作をサポートします。

図 7 に、ビジネス・オブジェクト・エディターに表示されたテーブル・ビジネス・オブジェクトを示します。このビジネス・オブジェクトでは、データベース表の列ごとに 1 つの属性が設定されています。表には子ビジネス・オブジェクトがないため、属性はすべて単純属性です。

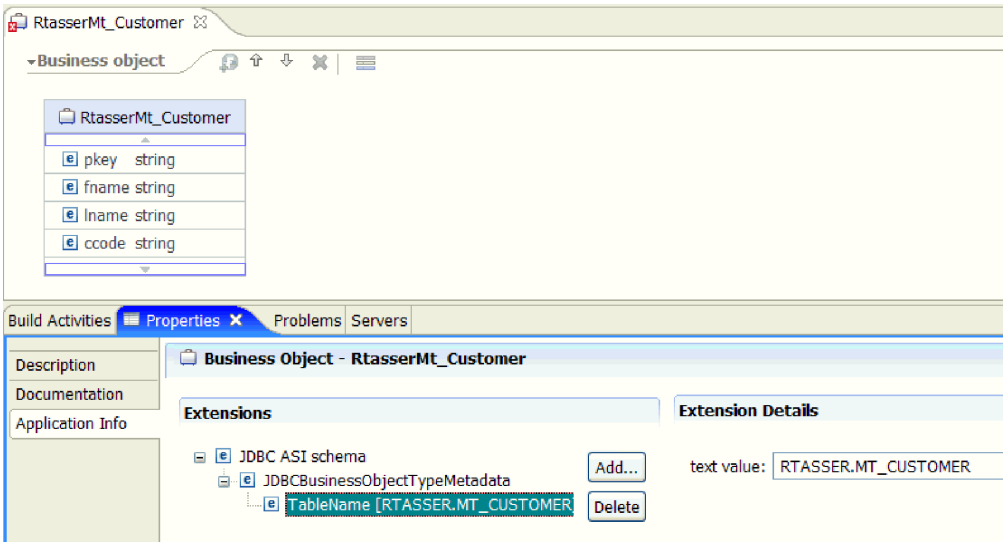


図 7. 子なしのテーブル・ビジネス・オブジェクト :

29 ページの図 8 に、子テーブル・ビジネス・オブジェクトが 1 つあるテーブル・ビジネス・オブジェクトを示します。このビジネス・オブジェクトでは、データベース表の列ごとの単純属性に加えて、子ビジネス・オブジェクトを指す複合属性が設定されています。

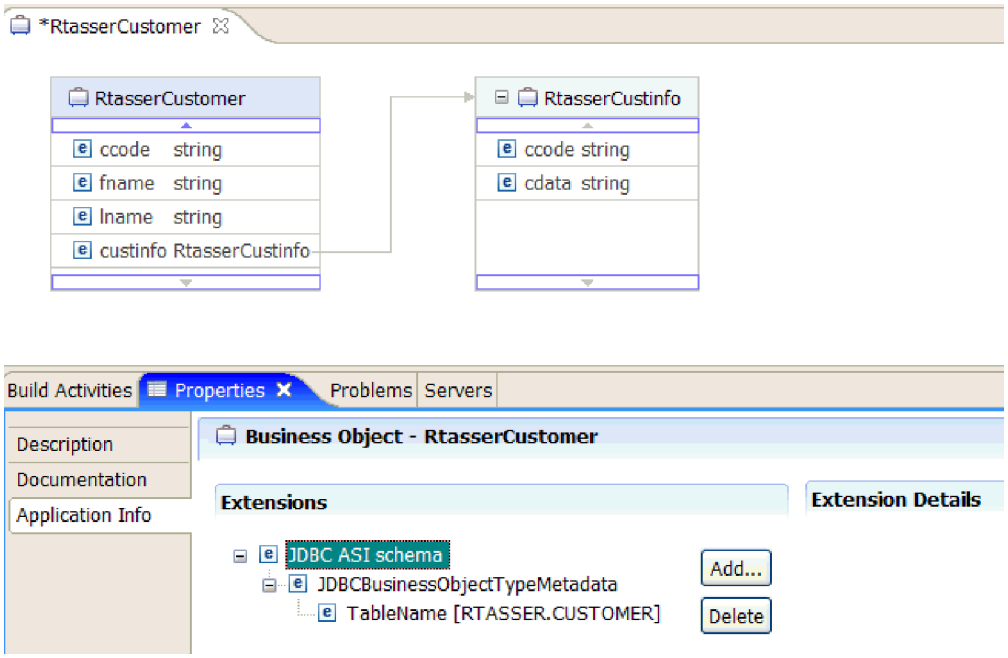


図 8. 子ビジネス・オブジェクトを 1 つ持つテーブル・ビジネス・オブジェクト:

ストアード・プロシージャー・ビジネス・オブジェクトとストアード関数ビジネス・オブジェクトの場合

ストアード・プロシージャーまたはストアード関数のビジネス・オブジェクトでは、ストアード・プロシージャーまたはストアード関数のすべての入力および出力パラメーターに、ビジネス・オブジェクトに対応する属性があります。入力または出力パラメーターのいずれかが、配列や構造体などの複合型である場合、対応するビジネス・オブジェクト属性は、配列または構造体の属性を含む子ビジネス・オブジェクトを持つ子ビジネス・オブジェクト型です。ストアード・プロシージャーが結果セットを戻した場合、戻された結果セットの属性を格納する子ビジネス・オブジェクトが作成されます。

ストアード・プロシージャーとストアード関数のビジネス・オブジェクトは、Execute Outbound 操作をサポートします。

ストアード・プロシージャー・ビジネス・オブジェクトの構造を下のサンプル・ファイルに示します。ビジネス・オブジェクト ScottStrtValues および ScottStrtValuesStrt が、1 つの入力タイプと 2 つの出力タイプを持つストアード・プロシージャーから生成されます。出力パラメーターの 1 つは、構造体データ型です。外部サービス・ウィザードによって、構造体型のビジネス・オブジェクト ScottStrtValuesStrt が生成され、子オブジェクトとして親ビジネス・オブジェクト ScottStrtValues に追加されます。親ビジネス・オブジェクト内の構造体型の属性については、ChildBOType アプリケーション固有情報が Struct に設定され、型が構造体であることを示します。ChildBOTypeName アプリケーション固有情報は、データベース内のユーザー定義構造体型の値に設定されます。次の例は、ストアード・プロシージャーのスキーマを示しています。

ScottStrtValues ビジネス・オブジェクトの例

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvalues" xmlns:scottstrtvalues=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvalues" xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
metadata" xmlns:scottstrtvaluesstrt=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvaluesstrt">
<import namespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvaluesstrt"
schemaLocation="ScottStrtvaluesStrt.xsd"/>
<import namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"
schemaLocation="JDBCASI.xsd"/>

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvalues">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
<jdbcasi:MaxNumOfRetRS>0</jdbcasi:MaxNumOfRetRS>
<jdbcasi:ResultSet>false</jdbcasi:ResultSet>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="pkey" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>IP</jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="fname" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>OP</jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="strt" type="scottstrtvaluesstrt:ScottStrtvaluesStrt"
minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>OP</jdbcasi:SPParameterType>
<jdbcasi:ChildBOType>STRUCT</jdbcasi:ChildBOType>
<jdbcasi:ChildBOTypeName>STRUCT1</jdbcasi:ChildBOTypeName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
```

```
</element>
</sequence>
</complexType>
</schema>
```

ScottStrtValuesStrt ビジネス・オブジェクトの例

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvaluesstrt" xmlns:scottstrtvaluesstrt=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvaluesstrt" xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<import namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"
schemaLocation="JDBCASI.xsd"/>

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvaluesStrt">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="name" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="title" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="dept_num" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

クエリー・ビジネス・オブジェクトの場合

データベース照会のビジネス・オブジェクトは、照会を実行する SQL ステートメントと、照会に必要なパラメーターを定義します。クエリー・ビジネス・オブジェクトでは、Outbound 操作 RetrieveAll がサポートされています。

例えば、次の SELECT ステートメントを実行するクエリー・ビジネス・オブジェクトがあるとします。

```
select C.pkey, C.fname, A.city from customer C, address A
    WHERE (C.pkey = A.custid) AND (C.fname like ?)
```

疑問符 (?) は、照会の入力パラメーターを示します。照会には複数のパラメーターを指定できます。各パラメーターは、SELECT ステートメントでは疑問符で示されています。サンプル・クエリー・ビジネス・オブジェクトの属性を表 3 に示します。クエリー・ビジネス・オブジェクトには、抽出される列ごとの単純属性、パラメーターごとの単純属性、およびパラメーター置換の後も WHERE 節を保持する、照会の WHERE 節の「プレースホルダー・オブジェクト」があります。

表 3. クエリー・ビジネス・オブジェクトの属性

ビジネス・オブジェクト属性	説明
pkey	Customer 表のデータベース列 PKEY に対応
fname	Customer 表のデータベース列 FNAME に対応
city	Address 表のデータベース列 CITY に対応
parameter1	パラメーター。SELECT ステートメントの ? (疑問符) ごとに 1 つのパラメーターがあります。複数のパラメーターを持つ SELECT ステートメントでは、後続のパラメーターに parameter2、parameter3 のようにして名前が付けられます。
jdbcwhereclause	WHERE 節のプレースホルダー・オブジェクト

以下の図に、ビジネス・オブジェクト・エディターに表示されたサンプル・クエリーのビジネス・オブジェクトを示します。

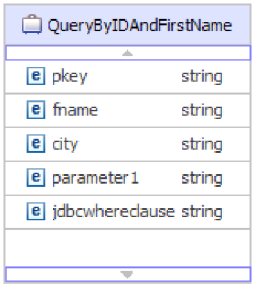


図 9. クエリー・ビジネス・オブジェクトの属性

この図は、クエリー・ビジネス・オブジェクト例のアプリケーション固有情報を示しています。SelectStatement アプリケーション固有情報には、SELECT ステートメントが含まれています。

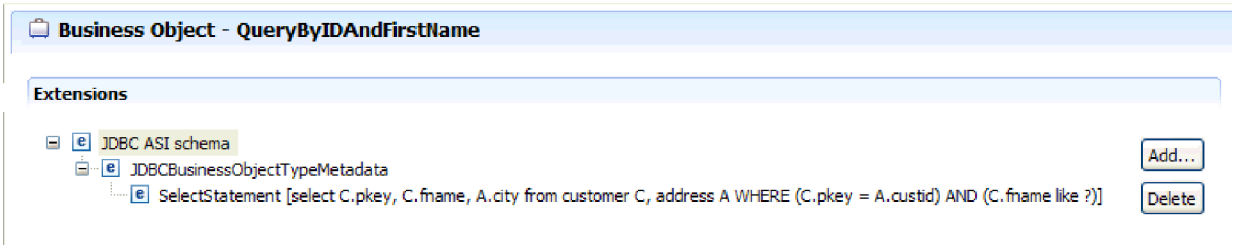


図 10. SELECT ステートメントは、ビジネス・オブジェクトのアプリケーション固有情報に保存されます。

バッチ SQL ビジネス・オブジェクトの場合

バッチ SQL ビジネス・オブジェクトは、データベース・アクションを実行する INSERT、UPDATE、および DELETE SQL ステートメントと、そのステートメントで必要とされるパラメーターを定義します。バッチ SQL ビジネス・オブジェクトでは、Outbound 操作 Execute がサポートされています。

例えば、次の INSERT および DELETE ステートメントを実行するバッチ SQL ビジネス・オブジェクトがあるとします。

```
Insert into customer (pkey,ccode,fname,lname) values(?,?,?,?);
Delete From Customer where pkey=?
```

各疑問符 (?) は、ステートメントのパラメーターを示します。バッチ SQL ビジネス・オブジェクト内の各ステートメントには複数のパラメーターを指定できます。各パラメーターは、ステートメントでは疑問符で示されています。バッチ SQL ビジネス・オブジェクトには複数のステートメントを含めることができ、それぞれが独自のパラメーター・セットを持ちます。図 11 に、それぞれが 1 つ以上のパラメーターを持つ INSERT ステートメントと DELETE ステートメントが定義されている、バッチ SQL ビジネス・オブジェクトのビジネス・オブジェクトの形式を示します。

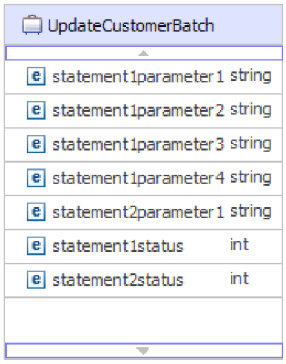


図 11. SQL ステートメントが 2 つのバッチ SQL ビジネス・オブジェクト

このビジネス・オブジェクトでは、statement1parameter1 や statement2parameter1 などの 各ステートメントのパラメーターごとに属性が設定されています。また、statement1status や statement2status などの、各ステートメントの状況に対する属性もあります。ステートメント自体は、 34 ページの図 12 に示すように、ビジネス・オブジェクトについてのアプリケーション固有情報として保管されます。

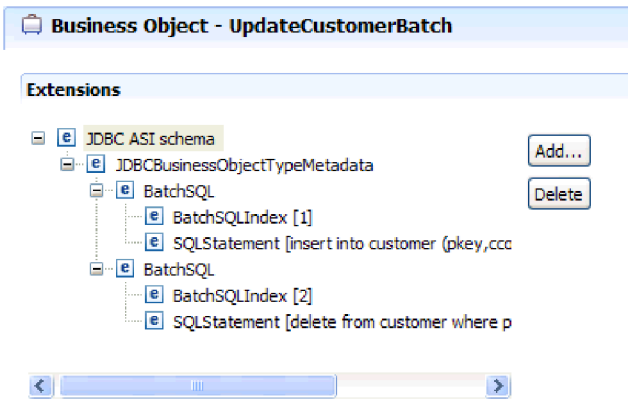


図 12. バッチ SQL ビジネス・オブジェクトのアプリケーション固有情報

ラッパー・ビジネス・オブジェクトの場合

ラッパー・ビジネス・オブジェクトにより、無関係の表の操作とビジネス・オブジェクトの表示を 1 回の操作で行うことができます。ラッパー・ビジネス・オブジェクトは、Create、Delete、Retrieve、および Update の Outbound 操作をサポートします。

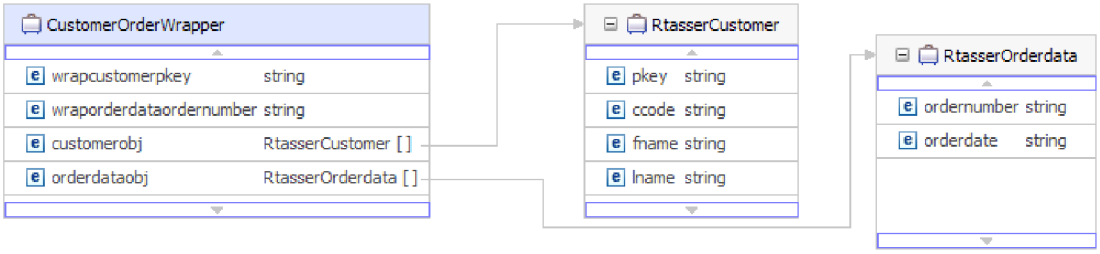


図 13. 2 つのテーブル・ビジネス・オブジェクトを含むラッパー・ビジネス・オブジェクト

ラッパー・ビジネス・オブジェクトには、子ビジネス・オブジェクトそれぞれの基本キーの単純属性が含まれています。フィールドの名前は「wrap」というストリングで、その後に、データベース表名とその表の基本キーの列名が続きます。ラッパー・ビジネス・オブジェクトには、テーブル・ビジネス・オブジェクトごとの複合属性も含まれています。属性の名前は、ストリング「obj」を付加した表名です。複合属性のタイプは、対応するテーブル・ビジネス・オブジェクトの名前です。

ビジネス・グラフ

アダプターの構成時に、ビジネス・グラフを生成するオプションを選択することもできます。バージョン 6.0.2 では、最上位レベルの各ビジネス・オブジェクトがビジネス・グラフに含まれていますが、このビジネス・オブジェクトには、実行する操作に関する追加情報を指定するために、バージョン 6.0.2 でアプリケーションが使用できる動詞が組み込まれています。バージョン 6.1.0 では、ビジネス・グラフが必要になるのは以下の状況に限られます。

- Outbound ApplyChanges 操作を使用する必要がある場合
- バージョン 6.1.0 より前のバージョンの WebSphere Integration Developer で作成されたモジュールにビジネス・オブジェクトを追加する場合

ビジネス・グラフが存在する場合、ビジネス・グラフは処理されますが、ApplyChanges 以外のすべての操作で動詞は無視されます。

ビジネス・オブジェクトが作成される仕組み

ビジネス・オブジェクトを作成するには、WebSphere Integration Developer から起動される外部サービス・ウィザードを使用します。このウィザードにより、データベースに接続し、データベース・オブジェクトがディスカバーされ、表示されます。ビジネス・オブジェクトを作成するデータベース・オブジェクトを選択します。例えば、調べるスキーマを指定します。指定されたスキーマで、テーブル、ビュー、ストアード・プロシージャー、ストアード関数、シノニム、およびニックネームを選択します。また、ビジネス・オブジェクトを追加で作成できます。例えば、データベースに対して実行されるユーザー定義の SELECT、INSERT、UPDATE、または DELETE ステートメントの結果を表すビジネス・オブジェクトを作成できます。 このウィザードでは、親子関係と、無関係なビジネス・オブジェクトをまとめるラッパーを使用してビジネス・オブジェクト階層を作成できます。

必要なビジネス・オブジェクトを指定し、これらのオブジェクトの階層を定義すると、ウィザードにより、選択されたオブジェクトを表すビジネス・オブジェクトが生成されます。また、アダプターに必要なその他の成果物も生成されます。

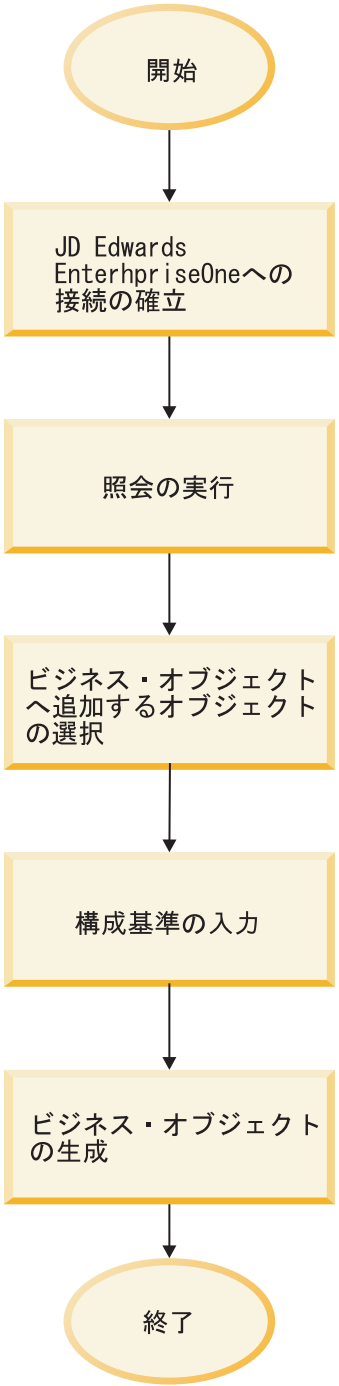


図 14. ビジネス・オブジェクトが作成される仕組み

場合によっては、ウィザードで親子関係を完全に構成できないこともあります。これらの関係の場合は、WebSphere Integration Developer から起動するビジネス・オブジェクト・エディターを使用して、ウィザードによって作成されたビジネス・オブジェクト階層の定義を変更または完了します。詳しくは、WebSphere Integration Developer インフォメーション・センター (リンク: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/in)で、ビジネス・オブジェクト・エディターによるビジネス・オブジェクトの変更方法を参照してください。

ビジネス・オブジェクト階層

階層型ビジネス・オブジェクトでの親子関係とデータ所有権を使用して、データベース表の間の関係を定義します。無関係な表は、ラッパー・ビジネス・オブジェクトでグループ化できます。

ビジネス・オブジェクトは、フラットまたは階層です。フラットのビジネス・オブジェクトでは、すべての属性が単純属性であり、データベース表の 1 行を表します。階層には、関連するビジネス・オブジェクトまたは無関係なビジネス・オブジェクトを含めることができます。関連するビジネス・オブジェクトには、所有関係を伴う親子関係または所有権を伴わない親子関係があります。無関係なビジネス・オブジェクトの場合、ラッパー・ビジネス・オブジェクトが使用されます。

階層 ビジネス・オブジェクトという用語は、あらゆるレベルの子ビジネス・オブジェクトをすべて含む、完全なビジネス・オブジェクトを指します。*個別* ビジネス・オブジェクトという用語は、そのビジネス・オブジェクトが子ビジネス・オブジェクトを含んでいるか、そのビジネス・オブジェクトが親ビジネス・オブジェクトに属しているかに関係なく、1 つのビジネス・オブジェクトを指します。個別ビジネス・オブジェクトは、複数のデータベース表にまたがるビューを表すことができます。*最上位* ビジネス・オブジェクトという用語は、階層の頂点にあり、それ自体は親ビジネス・オブジェクトを持たない個別ビジネス・オブジェクトを指します。

階層ビジネス・オブジェクトは、子ビジネス・オブジェクト、子ビジネス・オブジェクトの配列、またはその組み合わせを表す属性を持ちます。そして、子ビジネス・オブジェクトも、それぞれ自身の子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を持つことができます。この関係は階層の下に向かって続きます。

単一カーディナリティー関係 は、親ビジネス・オブジェクト内の属性が 1 つの子ビジネス・オブジェクトを表すときに発生します。この場合、その属性は、その子ビジネス・オブジェクトと同じタイプです。アダプターは、単一カーディナリティー関係と、所有権のない単一カーディナリティー関係およびデータをサポートします。

複数カーディナリティー関係 は、親ビジネス・オブジェクト内の属性が子ビジネス・オブジェクトの配列を表すときに発生します。この場合、その属性は、その子ビジネス・オブジェクトと同じタイプです。

以下のタイプの関係をビジネス・オブジェクトの間で使用して、データベース表を示す階層を定義します。

- 単一カーディナリティー関係
- 単一カーディナリティー関係および所有権のないデータ
- 複数カーディナリティー関係
- 複数の親を持つ子ビジネス・オブジェクト

また、無関係なビジネス・オブジェクトを 1 つのラッパー・ビジネス・オブジェクトにまとめることができます。

どちらのカーディナリティーのタイプでも、親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の関係は、その関係を保管するビジネス・オブジェクトのキー属性に含まれるアプリケーション固有情報によって記述されます。

ビジネス・オブジェクトの単一カーディナリティー関係:

単一カーディナリティー関係では、親ビジネス・オブジェクト内の属性が 1 つの子ビジネス・オブジェクトを表します。この場合、その属性は、その子ビジネス・オブジェクトと同じタイプです。アダプターは、単一カーディナリティー関係と、所有権のない単一カーディナリティー関係およびデータをサポートします。

単一カーディナリティー関係

通常、単一カーディナリティーの子ビジネス・オブジェクトを含むビジネス・オブジェクトには、関係を表すための属性が 2 つ以上含まれます。1 つの属性のタイプは、子ビジネス・オブジェクトのタイプと同じになります。もう 1 つの属性は、子の基本キーを、外部キーとして親に格納するための単純属性です。親には、子に含まれる基本キー属性と同数の外部キー属性が含まれます。

図 15 に、一般的な単一カーディナリティー関係を示します。この例では、ParentBOName オブジェクト内の FKey は、子の基本キーを含む単純属性であり、同様に、ParentBOName オブジェクト内にある Child(1) は、子ビジネス・オブジェクトを表す属性です。

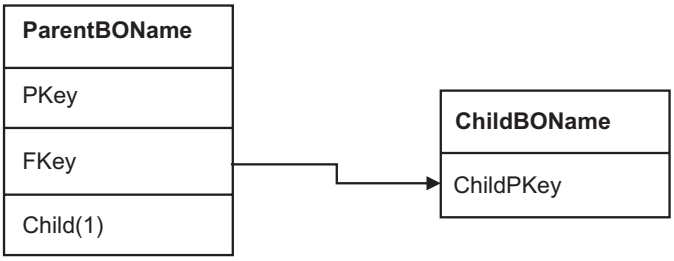


図 15. 典型的な単一カーディナリティー関係

関係を設定する外部キーが親に格納されるため、それぞれの親には特定のタイプの子ビジネス・オブジェクトを 1 つだけ格納できます。

親ビジネス・オブジェクトは、所有関係にある単一カーディナリティーの子と、所有関係にない単一カーディナリティーの子を持つことができます。参照テーブルは所有関係を伴わない関係に使用します。所有権は、アプリケーション固有情報 Ownership の値により示されます。

単一カーディナリティー関係および所有権のないデータ

通常、各親ビジネス・オブジェクトは、その親ビジネス・オブジェクトに含まれる子ビジネス・オブジェクトの内部のデータを所有しています。例えば、各 Customer ビジネス・オブジェクトが Address ビジネス・オブジェクトを 1 つ含んでいる場合に、新しいカスタマーが作成されると、Customer テーブルと Address テーブルの両方に新しい行が 1 行挿入されます。挿入された新しい住所は、その新しいカスタマ

ーに固有です。同様に、Customer テーブルからカスタマーを削除すると、Address テーブルからそのカスタマーの住所も削除されます。

ただし、複数の階層ビジネス・オブジェクトに同一のデータが含まれており、どのビジネス・オブジェクトもそのデータを所有していない場合があります。例えば、Address データベース表に、StateProvince ルックアップ・テーブルへの参照が含まれていると仮定します。このルックアップ・テーブルはほとんど更新されることがなく、住所データからは独立して保守されています。このため、住所データの作成または変更により、このルックアップ・テーブル内の都道府県データが影響を受けることはありません。ただし、StateProvince ビジネス・オブジェクトを Address ビジネス・オブジェクトと一緒に検索できるようにするには、StateProvince は Address の単一カーディナリティーの子であり、データの所有権のない関係を定義する必要があります。

データベース設計にルックアップ・テーブルが含まれている場合、ビジネス・オブジェクト設計はデータベース設計と少し異なります。この理由は、アダプターはテーブル・ビジネス・オブジェクトおよびその子テーブル・ビジネス・オブジェクトのためだけにデータを取得するためです。ルックアップ・テーブルを使用するには、テーブル間で単一カーディナリティーの親子関係を所有関係なしで作成する必要があります。StateProvince ルックアップ・テーブルはデータベース内の Address テーブルの子ではありませんが、対応する StateProvince ビジネス・オブジェクトは Address テーブル・ビジネス・オブジェクトの単一カーディナリティーの子です。これは、各アドレスに単一の都道府県が含まれているためです。しかし、Address ビジネス・オブジェクトは StateProvince ビジネス・オブジェクトを『所有』しません。住所を変更しても、都道府県のリストは変更されません。

アダプターが階層ビジネス・オブジェクトを Create、Delete、または Update 要求とともに受信した場合、アダプターは所有関係にない単一カーディナリティーの子ビジネス・オブジェクトの作成、削除、または更新を行いません。アダプターは、このようなビジネス・オブジェクトに対しては、Retrieve 操作のみを実行します。このような単一カーディナリティーのビジネス・オブジェクトの検索に失敗した場合、アダプターはエラーを戻して処理を停止します。ルックアップ・テーブルのビジネス・オブジェクトへの値の追加または変更は行いません。

非正規化データおよび所有権のないデータ

所有関係を伴わない包含関係には、静的参照テーブルの使用を容易にするだけでなく、正規化データと非正規化データを同期化するという別の機能があります。

正規化データから非正規化データへの同期: 所有関係を伴わない関係の場合、正規化アプリケーションから非正規化アプリケーションへの同期化を行うときに、データを作成または変更することができます。例えば、正規化されたソース・アプリケーションが、A と B という 2 つのテーブルにデータを格納するものとします。また、非正規化されている宛先アプリケーションでは、1 つのテーブルにすべてのデータが格納され、エンティティー A のそれぞれにエンティティー B のデータが重複して格納されるものとします。

この例では、テーブル B のデータの変更をソース・アプリケーションから宛先アプリケーションに同期化するには、テーブル B のデータが変更されるたびにテーブル A のイベントを起動する必要があります。さらに、テーブル B のデータはテーブ

ル A に重複して格納されているので、テーブル A の行ごとに、テーブル B で変更されたデータが含まれるビジネス・オブジェクトを送信しなければなりません。

注: 非正規化されたテーブルを更新する場合、1 行を更新した結果複数の行が変更されることがないように、レコードごとに固有キーを持たせるようにしてください。そのようなキーが存在しない場合、アダプターでは複数レコードが更新されたことを示すエラーが発生します。

非正規化データから正規化データへの同期: 非正規化されているソース・アプリケーションから正規化されている宛先アプリケーションにデータを同期化する場合、アダプターは、正規化されているアプリケーションに含まれる所有関係にないデータに関しては、作成、削除、または更新を行いません。

正規化されているアプリケーションにデータを同期化する場合、アダプターは、所有関係にない単一カーディナリティーの子をすべて無視します。そのような子のデータを作成、除去、または変更するには、データを手動で処理する必要があります。

複数カーディナリティー関係:

複数カーディナリティー関係では、親ビジネス・オブジェクト内の属性が子ビジネス・オブジェクトの配列を表します。属性は、子ビジネス・オブジェクトと同じ型です。アプリケーションが単一の子エンティティーを格納する場合を除き、関係を記述する外部キーは子に格納されます。親子関係は親に格納されます。

通常、子ビジネス・オブジェクトの配列を含むビジネス・オブジェクトには、関係を表す属性が 1 つだけ含まれており、通常はこの属性が基本キーになります。この属性のタイプは、子ビジネス・オブジェクトと同じタイプの配列です。親が複数の子を含むようにするため、関係を設定する外部キーは子に格納されます。

したがって、どの子にも、親の基本キーを外部キーとして含む単純属性が 1 つ以上存在します。子には、親に含まれる基本キー属性と同数の外部キー属性が含まれます。

関係を設定する外部キーが子に保管されるので、それぞれの親は、1 つ以上の子を持つことができます (子を持たないことも可能です)。

41 ページの図 16 に、複数カーディナリティー関係を示します。この例では、3 つの ChildBOName ボックス内の parentId は、親の基本キーを含む単純属性であり、ParentBOName ボックス内にある Child1 は、子ビジネス・オブジェクトの配列を表す属性です。

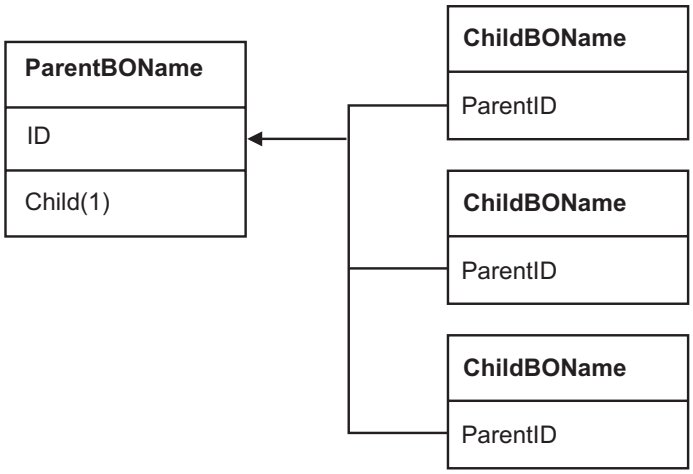


図 16. N>1 のビジネス・オブジェクトの複数カーディナリティーの関係

複数カーディナリティーの関係は、N=1 の関係である場合があります。アプリケーションによっては、親子関係を親ではなく子に格納するように、子エンティティーを 1 つ格納するものがあります。つまり、子には、親の基本キーに格納されている値と同一の値の外部キーが格納されます。

このタイプの関係がアプリケーションで使用されるのは、子のデータが親から独立して存在しておらず、親を介してのみそのデータにアクセスできる場合です。このような子のデータでは、子とその外部キー値を作成するために、親とその基本キー値があらかじめ存在していなければなりません。図 17 に、この関係タイプを示します。

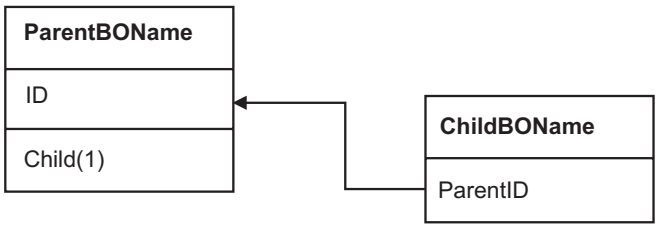


図 17. N=1 の場合の複数カーディナリティーの関係

複数の親テーブルを持つデータベース表:

データベース内の子テーブルが複数の親テーブルを持つ場合、アセンブリー・エディターを使用して追加の親ビジネス・オブジェクトを手動で構成する必要があります。外部サービス・ウィザードでは 1 つの親しか構成されません。

ビジネス・オブジェクト・スキーマ

ビジネス・オブジェクト・スキーマは、外部サービス・ウィザードを実行するときに選択したデータベース・オブジェクトから作成されます。各データベース・オブジェクトは最上位ビジネス・オブジェクトになります。

スキーマは、ビジネス・オブジェクト名とアプリケーション固有情報を定義します。ビジネス・オブジェクトと、その属性およびアプリケーション固有情報は、スキーマでは次のように表現されます。

- ビジネス・オブジェクトは、複合タイプの定義にマップします。
- ビジネス・オブジェクトのアプリケーション固有の情報は、複合タイプでの注釈に含まれます。
- ビジネス・オブジェクトの属性は、エレメント・タイプの定義にマップします。
- ビジネス・オブジェクト内の各プロパティーのアプリケーション固有情報は、エレメント・タイプに関する注釈に含まれます。

ビジネス・オブジェクトおよび属性のためのアプリケーション固有のプロパティーのテンプレートが、アダプターのメタデータ・スキーマに定義されています。このスキーマ・ファイルの名前は JDBCASI.xsd です。アダプターに対して生成されたスキーマ・ファイルの注釈には、このテンプレートへの参照が含まれます。

ストアード・プロシージャーの概要

ストアード・プロシージャーは、モジュールから Execute 操作で実行できるビジネス・オブジェクト、標準 SQL の代わりに実行して任意のビジネス・オブジェクトで操作できるビジネス・オブジェクト、または操作を実行する前または後に追加のアクションを実行できるビジネス・オブジェクトとすることができます。

ストアード・プロシージャーは、複数の SQL ステートメントのグループであり、1 つの論理単位を形成して特定のタスクを実行します。ストアード・プロシージャーは、アダプターがオブジェクトに対して実行する一連の操作または照会を、データベース・サーバー内にカプセル化したものです。アダプターは次のいずれかの方法でストアード・プロシージャーを使用します。

- データベースに対して実行するストアード・プロシージャー・ビジネス・オブジェクトを作成する
- ビジネス・オブジェクトの操作のために提供された SQL ステートメントを置換するか、または操作を実行する前または後にアクションを実行することによって、ビジネス・オブジェクトの操作を拡張する

ストアード・プロシージャー・ビジネス・オブジェクトの概要

データベースのストアード・プロシージャーまたはストアード関数に対応するストアード・プロシージャー・ビジネス・オブジェクトを作成することができます。次に Execute 操作を使用して、データベース内のデータに対してストアード・プロシージャーを実行できます。

外部サービス・ウィザードで、ストアード・プロシージャーまたはストアード関数を実行するストアード・プロシージャー・ビジネス・オブジェクトを作成できます。ウィザードではビジネス・オブジェクトを作成するために、データベース内のストアード・プロシージャーまたはストアード関数を検査します。ストアード・プロシージャー・ビジネス・オブジェクトでは、各パラメーターごとに属性が 1 つあります。

パラメーター属性が単純データ型の場合は、パラメーターのサンプル値についての属性が 1 つあります。ウィザードはストアード・プロシージャーを保存する前にス

トアード・プロシージャーを検証するとき、サンプル値を使用します。アダプターはストアード・プロシージャーからの結果を使用してパラメーターを検証し、返される結果セットの最大数を取得し、これらの結果セットのメタデータを使用して子ビジネス・オブジェクトを生成できるようにします。ストアード・プロシージャー・ビジネス・オブジェクトを検証すると、ウィザードではストアード・プロシージャー・ビジネス・オブジェクトの階層を自動的に生成します。

ストアード・プロシージャーに、Struct、Array、または ResultSet などの複合データ型である入出力パラメーターまたは戻り値パラメーターがある場合、これらの各パラメーターに対応するデータ型をウィザードで選択して、対応するユーザー定義の型の名前を指定する必要があります。Struct または Array 型のパラメーターの場合、プロパティー SPComplexParameterTypeName に保存されている対応するユーザー定義型名を指定する必要があります。

例えば、Struct_TEMP という名前の 1 つの Struct オブジェクトをデータベースに作成し、1 つの入力パラメーターとして型を設定する場合は、このプロパティーの値を `Struct_TEMP` に設定する必要があります。ウィザードはこの型名を使用して、対応する子ビジネス・オブジェクトの生成のためのメタデータを決定します。ストアード・プロシージャーが ResultSet を返す場合は、このストアード・プロシージャーから返される結果セットの数をプロパティー MaxNumberOfResultSets に設定する必要があります。この値は、アダプター・ランタイムによって処理される、返される結果セットの最大数を表します。

ストアード・プロシージャー・ビジネス・オブジェクトの場合、ウィザードは、ネストされた Struct または Array オブジェクトをサポートし、任意の数の層にわたるネストされた階層をサポートできます。ウィザードは、これらのネストされたすべての Struct または Array オブジェクトに対応する子ビジネス・オブジェクトを生成できます。

表 4. ストアード・プロシージャー・ビジネス・オブジェクトの複合データ型プロパティー

プロパティー名	型	説明

SPComplexParameterType	String	値は次のいずれかです。
		
		`Array`
		`ResultSetStruct`

SPComplexParameterTypeName	String	ユーザー定義型の名前。このプロパティーは、SPComplexParameterType の値が `Struct` または `Array` の場合に必要です。
MaxNumberOfResultSets	Integer	Adapter for JDBC ランタイムによって処理される、返される結果セットの最大数。ウィザードにより、この数のビジネス・オブジェクトが作成されます。

操作の代わりまたは追加で使用するストアード・プロシージャー

アダプターがデータベース内で、アダプターが操作を実行するのに使用する SQL ステートメントの代わりか、その前または後でストアード・プロシージャーを使用するよう指定することができます。各ビジネス・オブジェクトはそれぞれの操作で使用する、異なるセットのストアード・プロシージャーを持つことができます。

アダプターは、単純な SQL ステートメントを使用して、
Create、Update、Delete、Retrieve、または RetrieveAll 操作を実行できます。SQL ステートメントで使用される列名は、属性のアプリケーション固有の情報から取り出されます。WHERE 文節は、ビジネス・オブジェクトで指定されたキー値を使用して構成されます。各照会は、複数のテーブルにまたがることはできません。ただし、ビューに送ることはできます。しかし、アダプターによって提供された SQL ステートメントは、ストアード・プロシージャーおよびストアード関数を使用して置換または拡張することができます。

アダプターは以下の状況で、ストアード・プロシージャーまたはストアード関数を呼び出すことができます。

- ビジネス・オブジェクトを処理する前に、操作準備処理を行う。
- ビジネス・オブジェクトを処理した後で、操作後の処理を行う。
- 単純な Create、Update、Delete、Retrieve または RetrieveAll ステートメントを使用せずにビジネス・オブジェクトに対して一連の操作を実行する。

階層ビジネス・オブジェクトにおいて、その階層のビジネス・オブジェクトごとにストアード・プロシージャーを実行する場合は、ストアード・プロシージャーを、最上位ビジネス・オブジェクトと、ビジネス・オブジェクトの各子ビジネス・オブジェクトまたは配列に別々に関連付ける必要があります。ストアード・プロシージャーを最上位ビジネス・オブジェクトに関連付けても、各子ビジネス・オブジェクトに関連付けないと、その最上位ビジネス・オブジェクトはストアード・プロシージャーで処理されますが、子ビジネス・オブジェクトは標準 SQL 照会を使用して処理されます。

ストアード・プロシージャーのアプリケーション固有情報エレメント、各エレメントの目的および使用法を表 5 に示します。各エレメントの完全な説明は、表に続くセクションで説明します。ストアード・プロシージャーの定義例を 48 ページの『ストアード・プロシージャーの例』 に示します。

表 5. ストアード・プロシージャーについてのテーブルおよびビュー・ビジネス・オブジェクトのアプリケーション固有情報

記述名	エレメント名	目的
ストアード・プロシージャー・タイプ	StoredProcedureType	ストアード・プロシージャー・タイプは、使用するストアード・プロシージャーのタイプを定義します。これにより、ストアード・プロシージャーが呼び出される時点 (ビジネス・オブジェクトの処理前など) が決まります。
ストアード・プロシージャー名	StoredProcedureName	適切な StoredProcedureType と関連したストアード・プロシージャーの名前です。

表 5. ストアード・プロシージャーについてのテーブルおよびビュー・ビジネス・オブジェクトのアプリケーション固有情報 (続き)

記述名	エレメント名	目的
結果セット	ResultSet	この値は、ストアード・プロシージャーが結果セットを戻すかどうかを指定します。結果のセットが戻される場合は、結果のセットの行で戻される値を使用して、現在のビジネス・オブジェクトの複数カーディナリティーの子が作成されます。
パラメーター	Parameters	各 Parameters エレメントは、ストアード・プロシージャーまたはストアード関数の 1 つのパラメーターを記述します。
戻り値	ReturnValue	関数によって値が返されるため、プロシージャー呼び出しではなく関数呼び出しであることを示す値。

ストアード・プロシージャー・タイプ

ストアード・プロシージャー・タイプは、使用するストアード・プロシージャーのタイプを定義します。これにより、ストアード・プロシージャーが呼び出される時点 (ビジネス・オブジェクトの処理前など) が決まります。

表 6. 「ストアード・プロシージャー・タイプ」エレメントの特性

必須	はい
デフォルト	なし
使用可能な値	次のいずれかです。 • Before*Operation*SP • After*Operation*SP • *Operation*SP *Operation* には、操作名 (Create、Update、Delete、Retrieve、または RetrieveAll) の 1 つを指定します。
サポートされる双方向変換	いいえ
プロパティー・タイプ	String
使用上の注意	RetrieveAll に関連するストアード・プロシージャーのタイプは、最上位ビジネス・オブジェクトにのみ適用されます。 選択した任意のアプリケーション固有情報を StoredProcedureType プロパティーから除去できます。対応する操作のアプリケーション固有情報プロパティー・グループもすべて削除されます。

表 6. 「ストアード・プロシージャー・タイプ」エレメントの特性 (続き)

例	• CreateSP: 作成操作を実行します • UpdateSP: 更新操作を実行します • BeforeCreateSP: ビジネス・オブジェクトの作成前に実行します • AfterCreateSP: ビジネス・オブジェクトの作成後に実行します • AfterDeleteSP: ビジネス・オブジェクトの削除後に実行します

ストアード・プロシージャー名

適切な StoredProcedureType と関連したストアード・プロシージャーの名前です。

表 7. 「ストアード・プロシージャー名」エレメントの特性

必須	はい
デフォルト	なし
サポートされる双方向変換	はい
プロパティー・タイプ	String

結果セット

この値は、ストアード・プロシージャーが結果を戻すかどうかを決定します。結果のセットが戻される場合は、結果のセットの行で戻される値を使用して、現在のビジネス・オブジェクトの複数カーディナリティーの子が作成されます。

表 8. 「結果セット」エレメントの特性

必須	はい
デフォルト	なし
使用可能な値	`True` `False`
サポートされる双方向変換	いいえ
プロパティー・タイプ	Boolean
使用上の注意	Oracle ユーザーについて: ストアード・プロシージャーが結果セットを戻す場合、外部サービス・ウィザードが終了した後でビジネス・オブジェクト・エディターを使用して、この属性が `true` に設定されていることを確認してください。Oracle JDBC ドライバーがこの値を正しく戻さない場合があります。

Parameters

ストアード・プロシージャーまたはストアード関数のパラメーターごとに 1 つの Parameters エレメントがあります。各 Parameters エレメントは、1 つのパラメーターの名前と型を定義します。

表 9. 「Parameters」エレメントの特性

必須	はい
デフォルト	なし
内容	各 Parameters エレメントは、以下の情報を指定します。 • PropertyName: パラメーターとして受け渡すビジネス・オブジェクト属性の名前を指定します。 • Type: パラメーターのタイプ (以下のいずれかの値) を指定します。 – IP: 入力専用 – OP: 出力専用 – IO: 入出力 – RS: 結果セット
サポートされる双方向変換	いいえ
プロパティー・タイプ	String
使用上の注意	Oracle ストアード・プロシージャーの場合は、結果のセットが出力パラメーターとしてのみ戻されます。この場合、1 つのパラメーターのタイプが、結果セットを示す RS でなければなりません。

戻り値

関数によって値が返されるため、プロシージャー呼び出しではなく関数呼び出しであることを示す値。

表 10. 「戻り値」エレメントの特性

必須	いいえ
デフォルト	なし
使用可能な値	RS、ビジネス・オブジェクト属性の名前、または子ビジネス・オブジェクトの名前を指定できます。
サポートされる双方向変換	いいえ
プロパティー・タイプ	String

表 10. 「戻り値」エレメントの特性 (続き)

使用上の注意	戻り値が RS である場合、戻り値は結果セットです。この結果セットは、このビジネス・オブジェクトに対応する複数カーディナリティー・コンテナーの作成に使用されます。戻り値が属性名である場合、値はビジネス・オブジェクトの特定の属性に割り当てられます。属性が別の子ビジネス・オブジェクトである場合、アダプターはエラーを返します。 テーブルまたはビューから生成されたビジネス・オブジェクトにストアード・プロシージャーを関連付けるときに、ストアード・プロシージャーが関数である場合は、そのストアード・プロシージャーから値が返されます。 1 つの ReturnValue アプリケーション固有情報の値が操作のアプリケーション固有情報に追加されます。このアプリケーション固有情報が存在する場合は、関数によって値が返されるため、関数呼び出しであってプロシージャー呼び出しではないことが示されます。 このアプリケーション固有情報の値がビジネス・オブジェクト属性名である場合は、戻り値がビジネス・オブジェクトの特定の属性に割り当てられます。 このアプリケーション固有情報の値が別の子ビジネス・オブジェクトである場合、アダプター・ランタイムはエラーを返します。 要約すると、戻り値が単純データ型である場合はウィザードによって 1 つのビジネス・オブジェクト属性を戻り値にバインドでき、このアプリケーション固有情報の値がそのビジネス・オブジェクト属性の名前に設定されます。しかし、戻り値が結果セットである場合、ウィザードはこのアプリケーション固有情報の値を RS に設定します。 **注:** Oracle データベースの場合、結果セットは戻り値としてではなく、出力パラメーターとして返さなければなりません。出力パラメーターのタイプは RS に設定され、このパラメーターが結果セットを返すために使用されることを示します。 **重要:** 階層ビジネス・オブジェクトにおいて、その階層のビジネス・オブジェクトごとにストアード・プロシージャーを実行する場合は、ストアード・プロシージャーを、最上位ビジネス・オブジェクトと、ビジネス・オブジェクトの各子ビジネス・オブジェクトまたは配列に別々に関連付ける必要があります。ストアード・プロシージャーを最上位ビジネス・オブジェクトに関連付けても、各子ビジネス・オブジェクトに関連付けないと、その最上位ビジネス・オブジェクトはストアード・プロシージャーで処理されますが、子ビジネス・オブジェクトは標準 SQL 照会を使用して処理されます。

ストアード・プロシージャーの例

以下は、RtCustomer.xsd ファイルのカスタマー・ビジネス・オブジェクトの XML 定義を示した例で、Retrieve 操作の RetrieveSP および AfterRetrieveSP のストアード・プロシージャーの定義を表しています。アダプターは標準 SQL の代わりに RT.RETR_CUST ストアード・プロシージャーを実行して、テーブル・ビジネス・オブジェクトを取得します。ビジネス・オブジェクトを取得した後、アダプターは RT.CUSTINFO ストアード・プロシージャーを実行します。

```
<jdbcasi:JDBCBusinessObjectTypeMetadata
   xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:TableName>RTASSER.CUSTOMER</jdbcasi:TableName
```

```
    <jdbcasi:Operation> <jdbc asi:Name>Retrieve</jdbcasi:Name>
      <jdbcasi:StoredProcedures>
        <jdbcasi:StoredProcedureType>AfterRetrieveSP</jdbcasi:StoredProcedureType>
        <jdbcasi:StoredProcedureName>RT.CUSTINFO</jdbcasi:StoredProcedureName>
          <jdbcasi:Parameters>
            <jdbcasi:Type>IP</jdbcasi:Type>
            <jdbcasi:PropertyName>pkey</jdbcasi:PropertyName>
          </jdbcasi:Parameters>
          <jdbcasi:Parameters>
            <jdbcasi:Type>OP</jdbcasi:Type>
            <jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
          </jdbcasi:Parameters>
          <jdbcasi:Parameters>
            <jdbcasi:Type>OP</jdbcasi:Type>
            <jdbcasi:PropertyName>lname</jdbcasi:PropertyName>
          </jdbcasi:Parameters>
          <jdbcasi:Parameters>
            <jdbcasi:Type>OP</jdbcasi:Type>
            <jdbcasi:PropertyName>ccode</jdbcasi:PropertyName>
          </jdbcasi:Parameters>
        </jdbcasi:StoredProcedures>
        <jdbcasi:StoredProcedures>
        <jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
        <jdbcasi:StoredProcedureName>RT.RETR_CUST</jdbcasi:StoredProcedureName>
          <jdbcasi:Parameters>
            <jdbcasi:Type>IP</jdbcasi:Type>
            <jdbcasi:PropertyName>ccode</jdbcasi:PropertyName>
          </jdbcasi:Parameters>
          <jdbcasi:Parameters>
            <jdbcasi:Type>OP</jdbcasi:Type>
            <jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
          </jdbcasi:Parameters>
          <jdbcasi:Parameters>
            <jdbcasi:Type>OP</jdbcasi:Type>
            <jdbcasi:PropertyName>lname</jdbcasi:PropertyName>
          </jdbcasi:Parameters>
      </jdbcasi:StoredProcedures>
   </jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
```

ストアード関数の概要

データベースによっては、ストアード・プロシージャーの他にストアード関数をサポートしています。ストアード関数は、常に値を戻すという点を除き、ストアード・プロシージャーに類似しています。アダプターはこれらを同じようにサポートします。

Oracle データベースでは、アダプターはユーザーが CREATE FUNCTION ステートメントで作成するストアード関数をサポートします。この型の関数はユーザー定義関数 (UDF) と呼ばれることがありますが、この用語は一般的に、Java のストアード関数またはストアード・プロシージャーを指すことが多く、アダプターはこれらをサポートしません。

DB2 データベースでは、アダプターは値を戻すストアード・プロシージャーをサポートします。この機能を DB2 のユーザー定義関数と混同してはいけません。この用語はこの場合、SQL 言語の既存の組み込み関数への機能の拡張または追加のことを示しています。DB2 ユーザー定義関数は、アダプターで実行される操作、照会、およびバッチ SQL ステートメント用に作成した SQL 内で使用できますが、CREATE FUNCTION ステートメントで作成したユーザー定義関数は、通常の場合、ビジネス・オブジェクトとしてアダプターに示されません。

関数呼び出しの構文を次に示します。

```
? = call FunctionName parameter_list
```

次の構文のストアード・プロシージャー呼び出しと比較します。

```
call SPName parameter_list
```

ReturnValue ビジネス・オブジェクト・アプリケーション固有情報を使用して、戻り値を含む属性を指定します。ReturnValue について詳しくは、 200 ページの『ビジネス・オブジェクト・レベルのアプリケーション固有情報』を参照してください。

クエリー・ビジネス・オブジェクトの概要

クエリー・ビジネス・オブジェクトはユーザー定義の SELECT ステートメントをデータベースに対して実行して、ビジネス・オブジェクト内の一致するレコードを戻します。

外部サービス・ウィザードでは、データベースに対してユーザー定義 SELECT ステートメントを実行するクエリー・ビジネス・オブジェクトを作成できます。SELECT ステートメントを指定します。このとき、SELECT ステートメントの置換可能なパラメーターの代わりに ? (疑問符) を使用します。ウィザードに、各パラメーターのデータ型とサンプル値を指定できる領域が表示されます。サンプル値はデータベースのデータと一致している必要があります。これは、ウィザードでは SELECT ステートメントの結果を使用してクエリー・ビジネス・オブジェクトが作成されるためです。

ウィザードで照会の構成を保管する前に、これを検証します。検証時に、ウィザードでサンプル値を使用して SELECT ステートメントが実行されます。結果セットを取得した後、ウィザードはメタデータを分析し、すべての列の列名および列の型を取得します。返された結果セットの列ごとに、ウィザードは対応する属性を 1 つクエリー・ビジネス・オブジェクトに生成します。WHERE 節のパラメーターごとに、1 つの **jdbcwhereclause** 属性がクエリー・ビジネス・オブジェクトに作成され、この属性のデフォルト値にこの WHERE 節が設定されます。これらの属性は、デフォルトの WHERE 文節を置換する 1 つの動的 WHERE 文節を実行時に生成するために使用されます。

例えば、以下の SELECT ステートメントを指定したと仮定します。

```
select * from customer where fname=? and age=?
```

この WHERE 節には疑問符 (?) で示される 2 つのパラメーターがあります。最初のパラメーターのデータ型は **string** で、fname 列のデータ型と突き合わせます。2 番目のパラメーターのデータ型は **int** で、age 列と突き合わせます。データベースに、fname 列にストリング Mike、age 列に整数値 27 が含まれている顧客レコードがある場合、クエリー・ビジネス・オブジェクトの構成時にこれらの値をサンプル値として指定できます。ウィザードにより、戻される結果セットに対応するビジネス・オブジェクトが構成されます。

バッチ SQL ビジネス・オブジェクトの概要

バッチ SQL ビジネス・オブジェクトは 1 つ以上のユーザー定義の INSERT、UPDATE、および DELETE ステートメントを実行して、ステートメントの状況を戻します。

外部サービス・ウィザードでは、データベースに対してユーザー定義の INSERT、UPDATE、および DELETE ステートメントのセットを実行するバッチ SQL ビジネス・オブジェクトを作成できます。ステートメントを指定しますが、このとき、ステートメントの置換可能なパラメーターの代わりに ? (疑問符) を使用します。ウィザードに、各パラメーターのデータ型とサンプル値を指定できる領域が表示されます。ウィザードは構成を保存する前にビジネス・オブジェクトを検証するとき、サンプル値を使用します。

バッチ SQL ステートメントは UPDATE および DELETE ステートメントで動的 WHERE 節をサポートしません。アダプターは JOIN または SUBSELECT などの複合ステートメントを受け取りますが、アダプターではこれらのステートメントは構文解析されません。

バッチ SQL ビジネス・オブジェクトに単一の INSERT ステートメントが含まれる場合、自動生成 ID または挿入された行の識別値を検索できます。

ビジネス・オブジェクトの各 SQL ステートメントは状況値を戻しますが、これは Statement*N*Status という名前の属性に配置されます。例えば、ビジネス・オブジェクトの最初の SQL ステートメントの状況が Statement1Status に、2 番目のステートメントの状況が Statement2Status に配置されます。

単一の INSERT ステートメントが失敗すると、アダプターは例外をスローします。バッチ UPDATE のいずれかのステートメントが正しく実行されないと、JDBC ドライバーは java.sql.BatchUpdateException をスローします。これが発生すると、アダプターはトランザクションをロールバックして、どの SQL ステートメントもデータベースでコミットされないようにします。

外部サービス・ウィザード

WebSphere Integration Developer の外部サービス・ウィザードを使用して、データベースのオブジェクトをディスカバーし、バッチ SQL、照会、およびラッパー・ビジネス・オブジェクトを生成し、選択されているデータベース・オブジェクトからビジネス・オブジェクトを生成します。このウィザードでは、アダプターが Service Component Architecture (SCA) コンポーネントとして稼働できるようにするサービス成果物とモジュールも生成します。

標準の準拠

この製品は、アクセシビリティー標準やインターネット・プロトコル標準といった、いくつかの行政標準および業界標準に準拠しています。

アクセシビリティー

IBM は、年齢や能力を問わず、すべての人が便利に使用できる製品の提供に努めています。WebSphere Adapters は、完全にアクセス可能で、米国リハビリテーション

法第 508 条に準拠しています。アクセシビリティー機能を使用すると、運動障害または視覚障害など身体に障害を持つユーザーがソフトウェア・プロダクトを快適に操作できるようになります。これらの機能は、WebSphere Adapters のインストール機能や管理機能に組み込まれています。

管理

ランタイム管理コンソールは、エンタープライズ・アプリケーションのデプロイメントおよび管理用の基本インターフェースです。このコンソールは、標準の Web ブラウザー内に表示されます。Microsoft Internet Explorer や Netscape Browser などのアクセス可能な Web ブラウザーを使用すると、次のことが可能になります。

- スクリーン・リーダー・ソフトウェアとデジタル・スピーチ・シンセサイザーを使用して、画面上に表示されている内容を聞く
- IBM ViaVoice® などの音声認識ソフトウェアを使用したデータの入力とユーザー・インターフェースへのナビゲート
- マウスの代わりにキーボードを使用して機能を操作する

プロダクト機能は、提供されるグラフィカル・インターフェースではなく、標準テキスト・エディターおよびスクリプト・インターフェースまたはコマンド行インターフェースにより、構成および使用できます。

場合によっては、特定の製品機能についての文書に、その機能のアクセシビリティーについての追加情報が記載されています。

外部サービス・ウィザード

外部サービス・ウィザードは、モジュールを作成するのに使用する主コンポーネントです。WebSphere Integration Developer を通して使用可能な Eclipse プラグインとして実装されるこのウィザードは、完全にアクセス可能です。

キーボード・ナビゲーション

この製品では、標準の Microsoft Windows® ナビゲーション・キーを使用します。

IBM とアクセシビリティー

IBM のアクセシビリティーに対する取り組みについては、*IBM Accessibility Center* の Web サイト (http://www.ibm.com/able/) を参照してください。

インターネット・プロトコル・バージョン 6 (IPv6)

WebSphere Process Server および WebSphere Enterprise Service Bus は、Internet Protocol Version 6 (IPv6) の互換性について、WebSphere Application Server に依存しています。さらに、WebSphere Adapter for JDBC は、データベース・サーバーおよびそのサーバーへの接続に使用する JDBC ドライバーの IPv6 サポートに依存します。

データベース・サーバーおよび JDBC ドライバーが純粋な IPv6 をサポートする場合は、純粋な IPv6 機能をアダプターで使用できます。

IBM WebSphere Application Server バージョン 6.1.0 以上は、純粋なインターネット・プロトコル・バージョン 6.0 (IPv6) をサポートしています。

この互換性の WebSphere Application Server での詳細については、http://www.ibm.com/software/webservers/appserv/was/library/で、IPv6 サポートを参照してください。

IPv6 について詳しくは、http://www.ipv6.org を参照してください。

第 2 章 アダプター実装の計画

WebSphere Adapter for JDBC を使用する前に、作業者に必要な経験と、アダプターが稼働するサーバー環境について理解しておきます。アダプターをサーバー環境にデプロイする上での考慮事項を理解すると共に、クラスター・サーバー環境の使用によってアダプターのパフォーマンスおよび可用性を向上させる方法を検討します。

始める前に

モジュールの構成とデプロイを開始する前に、ビジネス・インテグレーションの概念、Java Database Connectivity (JDBC)、ご使用の環境のデータベース製品、および WebSphere Integration Developer および WebSphere Process Server または WebSphere Enterprise Service Bus の機能について十分に理解しておく必要があります。

WebSphere Adapter for JDBC を構成してデプロイするには、以下の概念、ツール、および作業に関する知識と経験が必要です。

- 構築するソリューションの業務要件。
- ご使用の環境の JDBC およびデータベース製品。これには、データ・アクセスの問題、トランザクション・モデル、および異種のリレーショナル・データベース、キュー、および Web サービス間の接続が含まれています。
- Service Component Architecture (SCA) プログラミング・モデルなどのビジネス・インテグレーションの概念およびモデル。
- 統合ソリューションに使用するサーバーの機能と要件。ホスト・サーバーの構成と管理の方法、および管理コンソールの使用によるプロパティー定義の設定と変更の方法、接続ファクトリーの構成方法、イベントの管理方法を理解しておく必要があります。
- WebSphere Integration Developer によって提供されるツールおよび機能。これらのツールの使用によるモジュールの作成方法、コンポーネントの接続およびテスト方法、その他の統合作業の実行方法を理解しておく必要があります。

セキュリティー

アダプターは、Java 2 セキュリティーの J2C 認証データ入力 (認証別名) 機能を使用して、ユーザー名およびパスワードの安全な認証機能を提供します。セキュリティー機能について詳しくは、WebSphere Process Server または WebSphere Enterprise Service Bus の資料を参照してください。

ユーザー認証

アダプターでは、データベースへの接続に必要なユーザー名およびパスワードを指定する方法がいくつかサポートされています。それぞれの方法の特徴および制限を理解した上で、ご使用のアプリケーションにとって適切なセキュリティー・レベルであり、かつ都合のよい方法を選択してください。

アダプターをアプリケーションに統合するには、以下の場合にユーザー名およびパスワードが必要になります。

- ユーザーがアダプターでアクセスできるオブジェクトおよびサービスに関する情報を抽出、つまりディスカバー するために外部サービス・ウィザードがデータベースに接続するとき。
- WebSphere Process Server または WebSphere Enterprise Service Bus での実行時に、アダプターが Outbound 要求および Inbound イベントを処理するためにデータベースに接続するとき。

ウィザードでの認証

外部サービス・ウィザードでは、両方の用途についての接続情報が求められます。ウィザードの実行中に使用するユーザー名およびパスワードは、アプリケーションをサーバーにデプロイするときとは別のものを使用できます。別の データベースに接続することもできます。ただし、2 つのデータベースのスキーマ名が同じである必要があります。例えば、Adapter for JDBC を使用するアプリケーションの開発および統合中は、実動データベースを使用しないことがあります。テスト・データベースを使用し、同じデータ形式で、より少ない数の模擬レコードを使用することにより、実動データベースのパフォーマンスに影響を与えることなく、また顧客データのプライバシー要件に起因する制限が生じることなく、アプリケーションを開発および統合できます。

ウィザードは、ディスカバリー・プロセス用に指定されたユーザー名およびパスワードをディスカバリー・プロセスでのみ使用します。これらは、ウィザードの完了後はアクセス不能になります。

実行時の認証

実行時、アダプターは、データベースに接続するためにユーザー名およびパスワードを提供する必要があります。ユーザー介入なしに接続するためには、アダプターは保管されているユーザー情報のコピーにアクセスしなければなりません。サーバー環境では、ユーザー情報の保管方法はいくつかあります。外部サービス・ウィザードでは、アダプターが次のいずれかの方法でユーザー情報を取得するように構成できます。

- アダプター・プロパティー
- データ・ソース
- J2C 認証別名

アダプター・プロパティーへのユーザー名およびパスワードの保管は、実行時にこの情報を提供するための直接的な方法です。外部サービス・ウィザードを使用してモジュールを構成するときに、このユーザー名およびパスワードを指定します。ユーザー名とパスワードを直接指定する方法はもっとも簡単なように見えますが、この方法には重要な制限があります。アダプター・プロパティーは暗号化されません。パスワードは、サーバー上で他のユーザーがアクセスできるフィールドに平文で保管されます。さらに、パスワードが変更された場合は、そのデータベースにアクセスするすべてのアダプター・インスタンスで、パスワードを更新しなければなりません。これは、アプリケーション EAR ファイルに組み込まれているアダプターだけでなく、サーバーに個別にインストールされたアダプターも該当します。

データ・ソースを使用する方法では、他のアプリケーション用にすでに確立された接続を使用します。例えば、複数のアプリケーションが同じユーザー名およびパスワードを使用して同じデータベースにアクセスする場合は、同じデータ・ソースを使用してそれらのアプリケーションをデプロイできます。ユーザー名およびパスワードを知るユーザーを、そのデータ・ソースにアプリケーションをデプロイする最初のユーザー、またはデータ・ソースを個別に定義する最初のユーザーのみに限定できます。

Java Authentication and Authorization Service (JAAS) で作成された J2C 認証別名を使用する方法は、堅固でセキュアなアプリケーション・デプロイ方法です。管理者は、システムにアクセスする必要がある 1 つ以上のアプリケーションで使用される認証別名を作成します。ユーザー名およびパスワードを知るユーザーを、その管理者のみに限定できます。管理者は、変更が必要な場合は単一の場所でパスワードを変更できます。

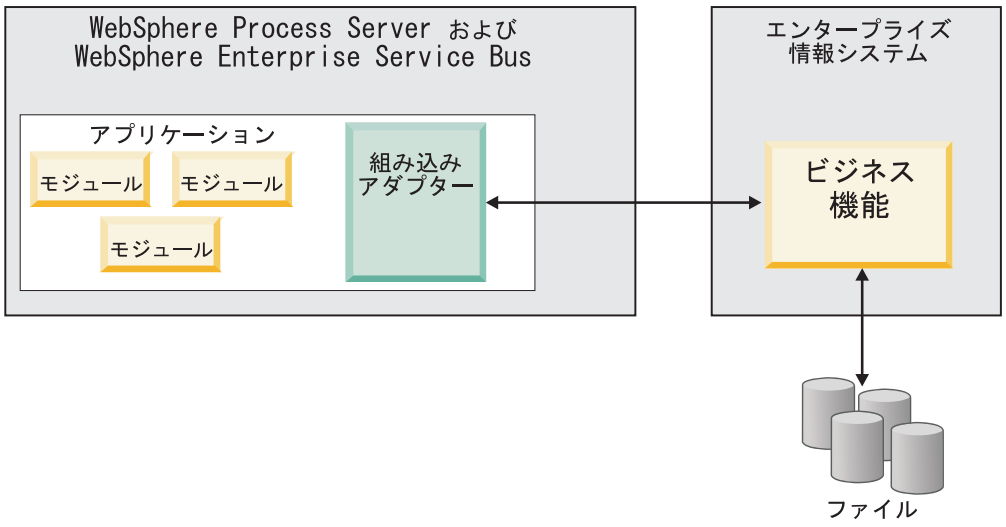
デプロイメント・オプション

デプロイされているアプリケーションの一部としてアダプターを組み込むか、RAR ファイルを単体でデプロイするかを選択できます。

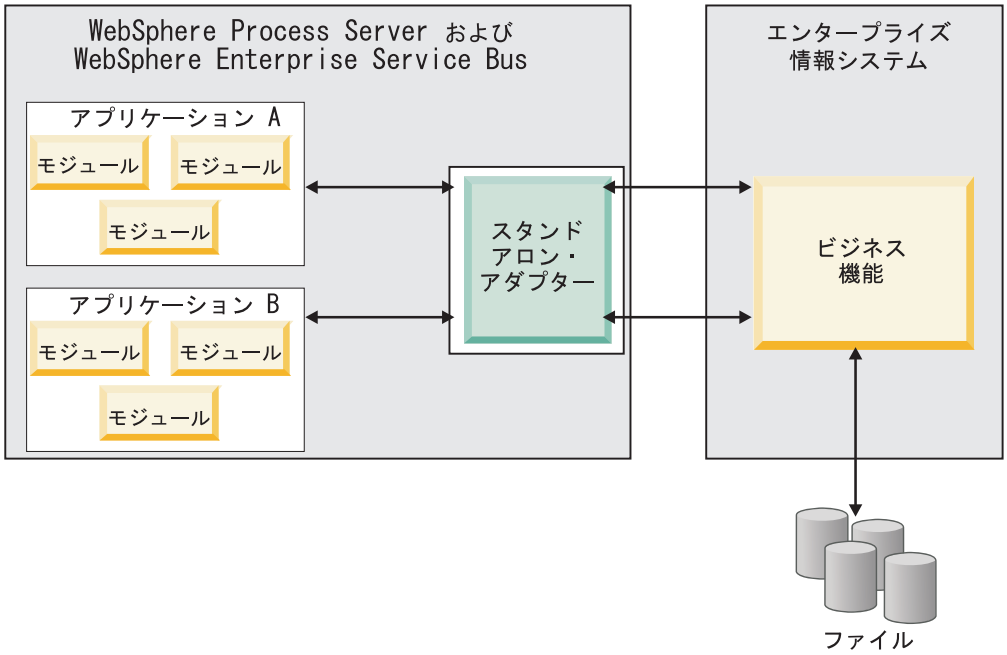
デプロイメント・オプションについて以下に説明します。

- 「**単一アプリケーションが使用するモジュールとともにデプロイする**」。アダプター・ファイルをモジュール内に組み込むと、モジュールをすべてのアプリケーション・サーバーにデプロイすることができます。組み込みアダプターを使用するのは、組み込みアダプターを使用するモジュールが 1 つある場合か、複数のモジュールでバージョンの異なるアダプターを実行する必要がある場合です。組み込みアダプターを使用すると、他のモジュールのアダプター・バージョンを変更することで、それらのモジュールを不安定にするリスクを生じることなく、1 つのモジュール内でアダプターをアップグレードできます。
- 「**複数アプリケーションが使用するサーバー上**」。モジュール内にアダプター・ファイルを組み込まない場合は、このモジュールを実行するアプリケーション・サーバーごとにモジュールをスタンドアロン・アダプターとしてインストールする必要があります。複数のモジュールが同じバージョンのアダプターを使用可能で、アダプターを中央の場所で管理する場合は、スタンドアロン・アダプターを使用します。スタンドアロン・アダプターの場合も、複数のモジュールに対して単一のアダプター・インスタンスを実行することにより、必要なリソースが軽減されます。

エンタープライズ・アーカイブ (EAR) ファイル内には、組み込みアダプターがバンドルされています。この組み込みアダプターは、一緒にパッケージされ、デプロイされたアプリケーションでのみ使用することができます。



スタンドアロン・アダプターを表すのは、スタンドアロンのリソース・アダプター・アーカイブ (RAR) ファイルです。これは、デプロイされた後、サーバー・インスタンス内のすべてのデプロイ済みアプリケーションから使用することができます。



ご使用のアプリケーションのプロジェクトを WebSphere Integration Developer を使用して作成する場合は、アダプターのパッケージ方法 (EAR ファイルによるバンドルまたはスタンドアロン RAR ファイルとして) を選択できます。この選択に応じて、アダプターをランタイム環境で使用する方法、および管理コンソールでのアダプターのプロパティーの表示の仕方が異なります。

アダプターをアプリケーションに組み込むか、スタンドアロン・モジュールとしてデプロイするかのどちらを選択するかは、アダプターの管理の仕方によって決まります。アダプターの 1 つのコピーのみを保持して、アダプターのアップグレード時

に複数のアプリケーションが中断してもかまわない場合は、アダプターをスタンドアロン・モジュールとしてデプロイすることが多くなります。

複数のバージョンを稼働させる計画があるため、アダプターのアップグレード時に起こる可能性のある中断により配慮する場合は、アダプターをアプリケーションに組み込むことになります。アダプターをアプリケーションに組み込む場合、アダプターのバージョンをアプリケーションのバージョンに関連付けて、単一のモジュールとして管理することができます。

アダプターのアプリケーションへの組み込みに関する考慮事項

アダプターをアプリケーションに組み込む計画がある場合は、以下の点を考慮してください。

- 組み込みアダプターには、クラス・ローダーの独立性があります。

  クラス・ローダーは、アプリケーションのパッケージ化、およびランタイム環境にデプロイされたパッケージ済みアプリケーションの動作に影響を与えます。クラス・ローダーの*分離* とは、アダプターは別のアプリケーションまたはモジュールからクラスを読み込むことができないという意味です。クラス・ローダーの分離機能により、異なるアプリケーションで、類似した名前の付いた 2 つのクラスによる相互干渉が防止されます。
- アダプターが組み込まれた各アプリケーションを、別々に管理する必要があります。

スタンドアロン・アダプターを使用する際の考慮事項

スタンドアロン・アダプターを使用する場合は、以下の点を考慮してください。

- スタンドアロン・アダプターには、クラス・ローダーの独立性がありません。

  スタンドアロン・アダプターにはクラス・ローダーの分離が存在しないため、ある特定の Java 成果物の 1 つのバージョンのみが実行され、その成果物のバージョンや順序は特定されません。例えば、スタンドアロン・アダプターを使用する場合は、*1* つの リソース・アダプター・バージョン、*1* つの アダプター・ファウンデーション・クラス (AFC) バージョン、または *1* つの サード・パーティー JAR バージョンのみが存在します。スタンドアロン・アダプターとしてデプロイされたアダプターはすべて、単一の AFC バージョンを共有し、1 つのアダプターのすべてのインスタンスは同じコードのバージョンを共有します。1 つのサード・パーティー・ライブラリーを使用するアダプター・インスタンスはすべて、そのライブラリーを共有しなければなりません。
- これらの共有成果物のいずれかを更新する場合、その成果物を使用するすべてのアプリケーションが影響を受けることになります。

  例えば、サーバー・バージョン X で動作しているアダプターを使用しているときに、クライアント・アプリケーションのバージョンをバージョン Y に更新すると、元のアプリケーションが動作しなくなることがあります。
- AFC には前のバージョンとの互換性がありますが、単体でデプロイされる各 RAR ファイルには、最新バージョンの AFC を入れておく必要があります。

スタンドアロン・アダプターのクラスパス内に JAR ファイルの複数のコピーがある場合、使用される JAR ファイルはランダムになります。このため、すべての JAR ファイルを最新バージョンにしておく必要があります。

クラスター環境での WebSphere Adapters

モジュールをクラスター化されたサーバー環境にデプロイすることで、アダプターのパフォーマンスおよび可用性を向上させることができます。スタンドアロン・アダプター、または組み込みアダプターのどちらを使用してモジュールをデプロイする場合も、モジュールは、クラスター内のすべてのサーバー内に複製されます。

WebSphere Process Server、WebSphere Application Server Network Deployment、および WebSphere Extended Deployment では、クラスター化された環境がサポートされます。クラスターとは、ワークロードの平衡を取り、高可用性とスケーラビリティーを提供するために、一緒に管理されるサーバー・グループのことです。サーバー・クラスターをセットアップするときには、デプロイメント・マネージャー・プロファイルを作成してください。デプロイメント・マネージャーのサブコンポーネントである HAManager により、アダプター・インスタンスを活動状態にするよう JCA (Java EE Connector Architecture) コンテナーに通知されます。JCA コンテナーにより、アダプター・インスタンスのランタイム環境が提供されます。クラスター環境の作成について詳しくは、リンク http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm_cluster_v61.html を参照してください。

必要に応じて、WebSphere Extended Deployment を使用して、クラスター環境内のアダプター・インスタンスのパフォーマンスを向上させることができます。WebSphere Extended Deployment は、WebSphere Application Server Network Deployment で使用される静的作業負荷マネージャーの代わりに、動的作業負荷マネージャーを使用することにより、WebSphere Application Server Network Deployment の機能を拡張します。動的作業負荷マネージャーは、要求による負荷の平衡化を動的に行うことによって、クラスター内のアダプター・インスタンスのパフォーマンスを最適化できます。これは、負荷の変動に応じて、アプリケーション・サーバー・インスタンスを自動的に停止したり始動したりできることを意味します。これにより、能力や構成が異なる複数のマシンが負荷の変動に一様に対処できるようになります。WebSphere Extended Deployment の利点について詳しくは、http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r1/index.jsp のリンクを参照してください。

クラスター化された環境では、アダプター・インスタンスは、Inbound 処理および Outbound 処理の両方を行えます。

Inbound 処理の高可用性

Inbound 処理は、データベースのデータを更新した結果、起動するイベントに基づいています。WebSphere Adapter for JDBC は、イベント・テーブルをポーリングすることで更新を検出するよう構成されます。その後、アダプターはイベントをそのエンドポイントにパブリッシュします。

モジュールをクラスターにデプロイすると、JCA (Java EE Connector Architecture) コンテナーにより、enableHASupport リソース・アダプター・プロパティーが検査さ

れます。enableHASupport プロパティーの値が真である場合 (デフォルトの設定)、すべてのアダプター・インスタンスはポリシー N のうちの 1 つを持つ HAManager に登録されます。 このポリシーは、アダプター・インスタンスのうちの 1 つのみがイベントのポーリングを開始することを意味します。 クラスター内のその他のアダプター・インスタンスが開始していても、それらのインスタンスは、アクティブなアダプター・インスタンスがイベントの処理を完了するまで、アクティブ・イベントに関して休止のままとなります。ポーリング・スレッドが開始しているサーバーが何らかの理由でシャットダウンした場合は、バックアップ・サーバーのいずれかで稼働しているアダプター・インスタンスが活動状態になります。

重要: enableHASupport プロパティーの設定は変更しないでください。

Outbound 処理の高可用性

クラスター化された環境では、Outbound 処理要求の実行に、複数のアダプター・インスタンスが使用可能です。そのため、Outbound 要求について WebSphere Adapter for JDBC と対話するアプリケーションが、ご使用の環境に複数存在する場合は、クラスター化された環境にモジュールをデプロイすることにより、パフォーマンスが向上することがあります。クラスター化された環境では、複数の Outbound 要求が同じレコードを処理しようとしない限り、複数の Outbound 要求を同時に処理することができます。

複数の Outbound 要求が、顧客の住所などの同じレコードを処理しようとした場合、WebSphere Application Server Network Deployment のワークロード管理機能により、その要求は、受信された順に使用可能なアダプター・インスタンスの間で分配されます。このため、クラスター化された環境では、この種の Outbound 要求は、単一サーバー環境内と同じように処理されます。つまり、1 つのアダプター・インスタンスが一度に処理するのは、1 つの Outbound 要求のみです。ワークロード管理について詳しくは、リンク http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm.html を参照してください。

準備済みステートメントのキャッシュのサポート

WebSphere Adapter for JDBC では、サーバーでの準備済みステートメントのキャッシュがサポートされています。これにより、Outbound/Inbound 操作または操作のバッチの実行にかかる時間が短縮されます。

アダプターは*準備済み*ステートメント を使用します。これは、1 回コンパイルされたが、繰り返し実行可能な SQL QUERY ステートメントが含まれている Java オブジェクトです。サーバーは、準備済みステートメントをキャッシュに格納することで、このステートメントの処理を最適化します。アダプターで準備済みステートメントのキャッシュを使用する場合は、管理コンソールでデータ・ソースを定義し、定義したデータ・ソースのキャッシュを有効にします。次に、以下のいずれかの方法で、このデータ・ソースを使用するようにアダプターを構成します。

- アダプターを初めて構成するときに、外部サービス・ウィザードを使用し、データ・ソースの JNDI 名を使用するように構成する
- 管理コンソールで DataSourceJNDIName プロパティーを設定する

バージョン 6.1.0 へのマイグレーション

WebSphere Adapter for JDBC のバージョン 6.1 へのマイグレーションを行うことにより、アダプターの前のバージョンから自動的にアップグレードします。さらに、アダプターの前のバージョンを組み込んだアプリケーションをマイグレーションできるため、このアプリケーションは、バージョン 6.1 が備えている機能や処理能力を活用できます。

マイグレーションに関する考慮事項

WebSphere Adapter for JDBC バージョン 6.1.0 には、既存のモジュールに影響する可能性がある更新が含まれています。

旧バージョンとの互換性

WebSphere Adapter for JDBC バージョン 6.1.0 は、バージョン 6.0.2 のアダプターと完全な互換性があり、カスタム・ビジネス・オブジェクト (XSD ファイル) で使用することができます。また、WebSphere Adapter for JDBC バージョン 6.0.2 を使用して作成されたデータ・バインディングとの互換性もあります。

バージョン 6.1 のアダプターはバージョン 6.0.2 と完全な互換性があるため、バージョン 6.0.2 のアダプターを利用したすべてのアプリケーションは、バージョン 6.1 にアップグレードするときに変更なしで実行できます。ただし、バージョン 6.1 のアダプターにあるフィーチャーおよび機能をアプリケーションで利用する場合は、マイグレーション・ウィザードを実行してください。

マイグレーション・ウィザードは、バージョン 6.0.2 のアダプターをバージョン 6.1 に置き換え (アップグレードし)、*ご使用のアプリケーションでバージョン 6.1 のフィーチャーおよび機能を使用可能にします。*

注: マイグレーション・ウィザードは、バージョン 6.1 のアダプターで使用できるマッパーやメディエーターなどの緩和コードを新規作成したり、既存のコードを変更したりしません。ご使用のアプリケーションにバージョン 6.0.2.x およびそれ以前のバージョンのアダプターが組み込まれているときに、バージョン 6.1.0 にアップグレードし、そのアプリケーションで 6.1 のフィーチャーおよび機能を利用する場合は、アップグレードするアプリケーションに変更を加えることが必要になる可能性があります。

1 つのモジュール内の*バージョン管理* に関して成果物に不整合がある場合は、このモジュール全体がそのようなものとしてマークを付けられ、マイグレーションで選択不可になります。バージョンの不整合は、プロジェクト破損の症状を示す可能性があるため、ワークスペース・ログに記録されます。

以下の方法はサポートされていません。

- WebSphere Integration Developer バージョン 6.1.0 と WebSphere Adapter for JDBC バージョン 6.0.2 との組み合わせでの外部サービス・ウィザードの実行
- WebSphere Integration Developer バージョン 6.0.2 と WebSphere Adapter for JDBC バージョン 6.1.0 との組み合わせでの外部サービス・ウィザードの実行

アップグレードするか、またはアップグレードしてマイグレーションするかどうかの決定

マイグレーション・ウィザードのデフォルト処理では、アダプターのアップグレードを実行してアプリケーション成果物をマイグレーションし、アプリケーションがバージョン 6.1 のアダプターのフィーチャーと機能を利用できるようにします。コネクター・プロジェクトを選択することによってコネクターのアップグレードを選択すると、ウィザードはマイグレーションに関連付けられた成果物を自動的に選択します。

バージョン 6.0.2 からバージョン 6.1 にアダプターをアップグレードしても、アダプターの成果物はマイグレーションしないようにする場合は、マイグレーション・ウィザードの該当するページでアダプターの成果物の選択を解除します。

アダプターの成果物が何も選択されていない状態でマイグレーション・ウィザードを実行すると、アダプターのインストールとアップグレードが行われますが、成果物はマイグレーションされず、アプリケーションではバージョン 6.1 のアダプターに存在するフィーチャーや機能を利用できなくなります。

最初にテスト環境でマイグレーション・ウィザードを実行

アダプターのマイグレーションでは、バージョン 6.1 の WebSphere Adapter for JDBC を利用するアプリケーションに変更を加える必要があるため、いつでもまず開発環境でマイグレーションを実行し、実稼働環境にデプロイする前にアプリケーションをテストしてください。

マイグレーション・ウィザードは開発環境に完全に統合されます。

非推奨機能

非推奨の機能とは、サポートされているが使用が推奨されておらず、廃止される可能性がある機能です。以前のバージョンの Adapter for JDBC の機能のうち、バージョン 6.1.0 で非推奨になった機能を以下に示します。

- ビジネス・グラフおよび動詞はオプションになりました。

  バージョン 6.0.2 での各ビジネス・オブジェクトが含まれたビジネス・グラフが、オプションになりました。バージョン 6.0.2 で作成されたビジネス・オブジェクトのモジュール用、または、ApplyChanges Outbound 操作を使用する新規 バージョン 6.1.0 モジュール用にのみビジネス・グラフが必要になります。
- UpdateWithDelete verb のサポート

マイグレーションの実行

バージョン 6.1.0 を使用するプロジェクトまたは EAR ファイルのマイグレーションを行うには、アダプター・マイグレーション・ウィザードを使用します。ツールが終了したらマイグレーションは完了するため、プロジェクトで作業したり、モジュールをデプロイしたりできます。

始める前に

『マイグレーションに関する*考慮事項*』の情報を見直します。

このタスクを実行する理由および時期

WebSphere Integration Developer でマイグレーションを実行するには、以下のステップを完了してください。

注: マイグレーションが完了すると、このモジュールは以前のバージョンの WebSphere Process Server、WebSphere Enterprise Service Bus、または WebSphere Integration Developer とは互換性がなくなります。

注: 以下の手順では、WebSphere Integration Developer の J2EE パースペクティブでコネクター・プロジェクトのコンテキスト・メニューからアダプター・マイグレーション・ウィザードを実行する方法について説明します。

注: 以下のいずれかの方法でもマイグレーションを実行できます。

- J2EE パースペクティブでプロジェクトを右クリックし、「**マイグレーション**」 → 「**プロジェクトのマイグレーション (Migrate project)**」を選択します。
- 問題ビューで、マイグレーション固有のメッセージを右クリックし、「**クイック・フィックス**」を選択して問題を解消します。

このタスクの手順

1. 既存のプロジェクトの場合は PI (プロジェクト交換) ファイルを、デプロイ済みアプリケーションの場合は EAR (エンタープライズ・アーカイブ) ファイルを、それぞれワークスペースにインポートします。
2. J2EE パースペクティブに切り替えます。
3. モジュールを右クリックし、「**マイグレーション**」 → 「**コネクター・プロジェクトの更新**」を選択します。
4. 「ようこそ」ページに表示されたタスクおよび警告を確認して、「**次へ**」を選択します。
5. 「プロジェクトの選択」ウィンドウで、「**次へ**」を選択します。

   デフォルトでは、ウィザードによりコネクター・プロジェクトとすべての依存プロジェクトのマイグレーションが行われます。プロジェクトに依存プロジェクトがあり、その時点ではそのうちの 1 つ以上の依存プロジェクトをマイグレーションしない場合は、「**依存アダプター・プロジェクト (Dependent adapter project)**」リストで、マイグレーションしない依存プロジェクトのチェック・ボックスの選択を解除します。このウィザードを再実行すれば、依存プロジェクトのマイグレーションを後で実行できます。以前にマイグレーション済みのプロジェクト、現行バージョンのプロジェクト、エラーのあるプロジェクトはマイグレーションの対象外であり、選択されません。
6. 「アダプターのマイグレーション」ウィンドウで、マイグレーションの変更内容を任意で確認できますが、選択項目を変更することはできません。「**終了**」をクリックします。
7. 問題ビューを参照して、マイグレーション・ウィザードからのメッセージ (先頭に CWPAD というストリングがあるメッセージ) の有無を確認します。
8. EAR ファイルをマイグレーションしている場合は、マイグレーション済みアダプターおよび成果物のある新規の EAR ファイルを作成して、WebSphere Process Server または WebSphere Enterprise Service Bus にデプロイすることもできま

す。 EAR ファイルのエクスポートおよびデプロイについて詳しくは、この資料で EAR ファイルについて説明しているトピックを参照してください。

結果

プロジェクトまたは EAR ファイルは、バージョン 6.1.0 へマイグレーションされます。アダプター・マイグレーション・ウィザードの終了後に外部サービス・ウィザードを実行する必要はありません。

マイグレーションしない場合のバージョン 6.0.2 プロジェクトの更新

アダプターをバージョン 6.0.2 からバージョン 6.1.0 にアップグレードする一方で、アダプター・プロジェクトの成果物をマイグレーションしないことを選択できます。

このタスクを実行する理由および時期

アダプターの内部名はバージョン 6.1.0 で変更されたため、WebSphere Integration Developer バージョン 6.1.0 でアダプター・ウィザードを使用するには、その前にバージョン 6.0.2 プロジェクトの成果物を更新して、新しい名前を使用する必要があります。バージョン 6.0.2 プロジェクトを更新するには、マイグレーション・ウィザードを使用します。次に、WebSphere Integration Developer のクイック・フィックス機能を使用して、プロジェクト成果物内のアダプター名を変更します。

このタスクの手順

1. プロジェクト交換 (PI) ファイルをワークスペースにインポートします。
2. J2EE パースペクティブで、プロジェクト名を右クリックして、「**マイグレーション**」 → 「**コネクター・プロジェクトの更新**」をクリックします。 アダプター・マイグレーション・ウィザードが開きます。
3. 「ようこそ」ページで、「**次へ**」をクリックします。
4. 「プロジェクトの選択」ウィンドウで、依存成果物プロジェクトの選択を解除し、「**終了**」をクリックします。
5. 問題ビューで、エラー・メッセージ「CWPADL77A1: IBM JDBC Adapter は ... と名前を変更する必要があります。(CWPADL77A1: The IBM JDBC Adapter must be renamed...)」を右クリックし、「**クイック・フィックス**」をクリックします。
6. 「クイック・フィックス」ウィンドウで、「**参照しているアダプターの名前を変更する (Rename the referenced adapter)**」というフィックスが選択されていることを確認し、「**OK**」をクリックします。
7. エラーが表示されたままの場合は、「**プロジェクト**」 → 「**クリーン**」をクリックし、更新直後のプロジェクトを選択して「**OK**」をクリックします。

結果

これで、プロジェクトを WebSphere Adapter for JDBC バージョン 6.1.0 で使用できるようになりました。

第 3 章 サンプルおよびチュートリアル

WebSphere Integration Developer のオンライン・サンプル/チュートリアル・ギャラリーには、WebSphere Adapters を使用するのに役立つサンプルおよびチュートリアルが置かれています。

オンライン・サンプル/チュートリアル・ギャラリーへのアクセス先のページは、以下のとおりです。

- WebSphere Integration Developer を始動すると表示される「ようこそ」ページ。WebSphere Adapter for JDBC のサンプルおよびチュートリアルを表示するには、「**取得**」をクリックします。表示されたカテゴリーをブラウズして、選択を行います。
- Web 上の http://publib.boulder.ibm.com/bpcsamp/index.html のページ。

第 4 章 デプロイメントのためのモジュールの構成

アダプターを WebSphere Process Server または WebSphere Enterprise Service Bus 上にデプロイできるように構成するには、WebSphere Integration Developer を使用して、アダプターをデプロイするときに EAR ファイルとしてエクスポートされるモジュールを作成します。次に、ディスカバーの対象となるビジネス・オブジェクトと、そのディスカバーを行うシステムを指定します。これらの手順が完了すると、外部サービスが正常に作成されます。

モジュールの構成のためのロードマップ

ランタイム環境で WebSphere Adapter for JDBC を使用できるようにするには、まずモジュールを構成する必要があります。このタスクの概要を理解すれば、タスクを達成するのに必要な手順を実行できるようになります。

WebSphere Integration Developer を使用することにより、使用するアダプターのモジュールを構成します。以下の図は、構成作業の流れを示しています。また、図の後に示す手順で、この作業の概要を説明します。これらの各ステップの実行方法の詳細については、このロードマップの後に記載するトピックを参照してください。

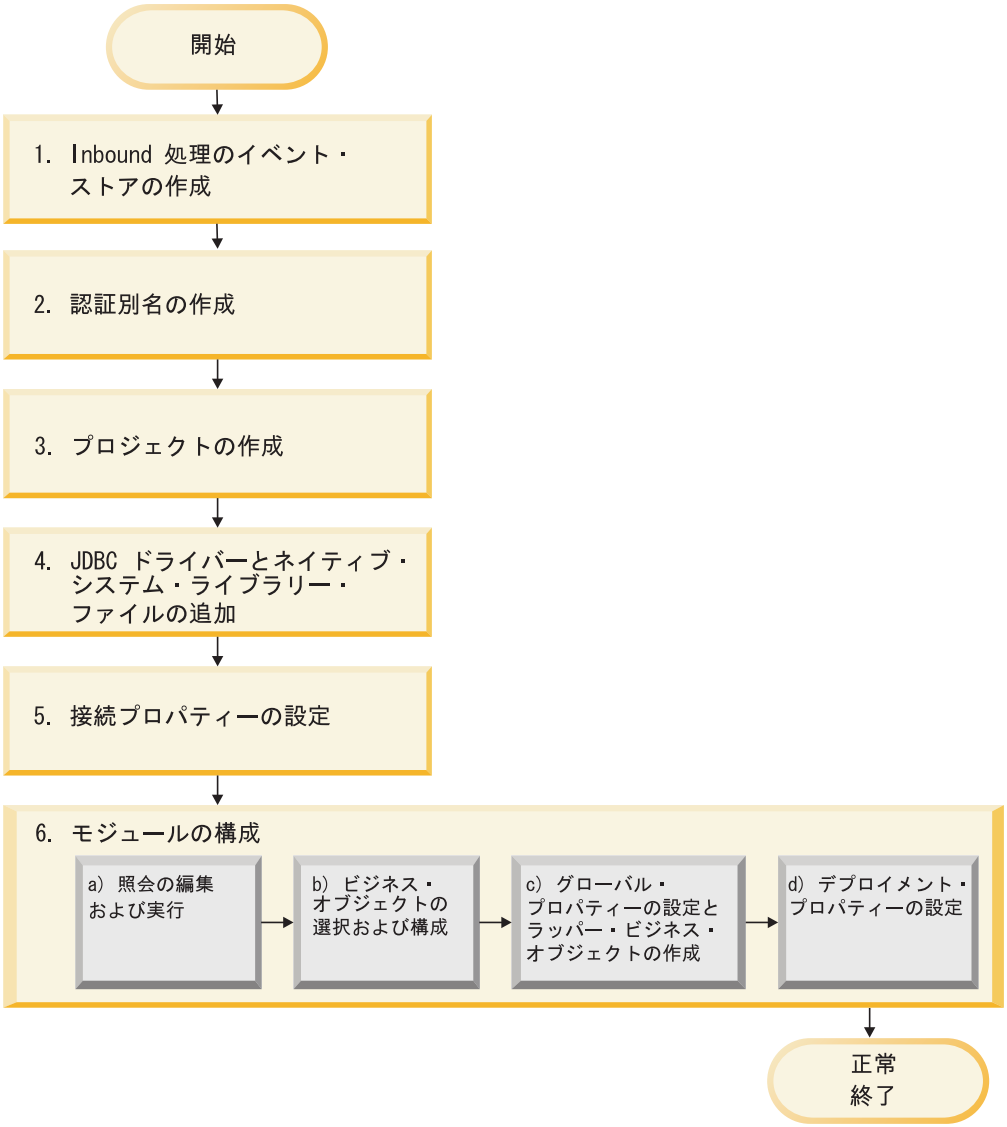


図 18. モジュールの構成のためのロードマップ

デプロイメント用のモジュールの構成

この作業は、次の概略的なステップから成ります。

1. Inbound 処理のイベント・ストアを作成します。
2. 暗号化したパスワードで データベースにアクセスするための認証別名を作成します。このステップは、パスワードおよび ID の取り扱いポリシーによってはオプションです。サーバー上の管理コンソールを使用して、この手順を実行してください。
3. プロジェクトを作成します。最初に、WebSphere Integration Developer で外部サービス・ウィザードを開始し、モジュールの作成とデプロイのプロセスを開始します。このウィザードはプロジェクトを作成します。これは、モジュールに関連付けられたファイルを編成するために使用されます。
4. WebSphere Adapter for JDBC が必要とする JDBC ドライバーおよびネイティブ・システム・ライブラリー・ファイルをプロジェクトに追加します。これらの

依存関係は、モジュールを EAR ファイルとしてエクスポートしてその EAR ファイルをサーバーにデプロイする場合にも必要です。

5. 外部サービス・ウィザードがオブジェクトおよびサービスをディスカバーするために データベースに接続するときに必要な接続プロパティーを設定します。
6. 外部サービス・ウィザードを使用して データベースからビジネス・オブジェクトおよびサービスを検出して選択し、ビジネス・オブジェクト定義および関連する成果物を生成することで、Inbound 処理用または Outbound 処理用のモジュールを構成します。
   a. アクセス可能なデータベース・オブジェクトをディスカバーする照会を編集して実行します。
   b. Inbound 処理用または Outbound 処理用のビジネス・オブジェクトを選択して構成します。
   c. 操作のグローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを作成します。
   d. 実行時にアダプターが データベースに接続するために使用するデプロイメント・プロパティーを設定します。次に、外部サービス・ウィザードを使用して新規モジュール (構成したビジネス・オブジェクトを含むモジュール)、import ファイルまたは export ファイル、およびサービス・インターフェースを保存することにより、サービスを生成します。

イベント・ストアの作成

アダプターが Inbound イベントを処理できるようにするには、事前にデータベースにイベント・ストアを作成する必要があります。必要に応じて、ユーザー・テーブル上にトリガーを設定して、イベント・テーブルを取り込むことができます。

このタスクを実行する理由および時期

イベントの Inbound 処理が必要な場合にのみ、このタスクを実行してください。イベントの報告対象のテーブルを含むデータベースにイベント・ストアを作成します。

このタスクの手順

1. イベント・ストアを作成します。 IBM DB2、IBM DB2 for z/OS、 Oracle、または MicrosoftSQL Server データベース用のイベント・ストアを作成するために、以下のサンプル・スクリプトが用意されています。
   - scripts_db2.sql
   - scripts_db2_zOS.sql
   - scripts_oracle.sql
   - scripts_mssql.sql

   これらのファイルは、 *WID_installation_dir*/ResourceAdapters/JDBC_*version*/samples/scripts ディレクトリーにあります。ここで、*WID_installation_dir* は、WebSphere Integration Developer のインストール・ディレクトリーであり、*version* はアダプターのバージョンを示します (例えば `6.1.0.0_IF1`)。
2. ユーザー・テーブルへの変更によって、イベント・ストアに保管されるイベントが自動的に生成されるように、必要に応じてトリガーをユーザー・テーブル上に

セットアップします。サンプル・スクリプトには、トリガーを作成して、イベント・ストアを取り込む方法についての例が含まれています。

結果

イベント・ストアは、イベント処理で使用可能です。

認証別名の作成

認証別名は、アダプターがデータベースへのアクセスに使用するパスワードを暗号化する機能です。認証別名を作成した後は、アダプターの構成時に (ユーザー ID とパスワードを直接入力する代わりに) その別名を使用できます。アダプターのプロパティーは暗号化されないため、パスワードを直接入力すると、他のユーザーが表示可能な平文でそのパスワードが保管されてしまいます。外部サービス・ウィザードでは、認証別名を使用するのがデフォルト選択です。

始める前に

認証別名を作成するには、WebSphere Process Server または WebSphere Enterprise Service Bus の管理コンソールへのアクセス権が必要です。データベースへの接続に使用するユーザー名およびパスワードも知っていなければなりません。

このタスクを実行する理由および時期

認証別名を使用すると、他のユーザーから見える可能性があるアダプター構成プロパティーに、パスワードを平文で保管する必要がなくなります。

以下の手順は、WebSphere Integration Developer によって管理コンソールにアクセスする方法を示しています。管理コンソールを直接使用する (WebSphere Integration Developer からアクセスしない) 場合は、管理コンソールにログインしてステップ 2 に進んでください。

このタスクの手順

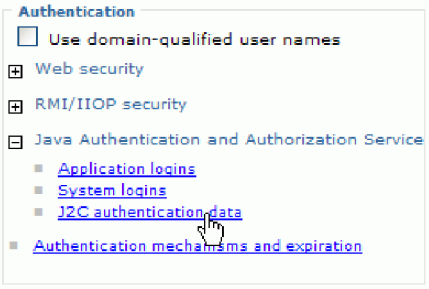
1. WebSphere Integration Developerを開始します。
   a. 「**スタート**」 → 「**プログラム**」 → 「**IBM Software Development Platform**」 → 「**IBM WebSphere Integration Developer 6.1**」 → 「**IBM WebSphere Integration Developer 6.1**」をクリックして、WebSphere Integration Developer を始動します。
   b. ワークスペースを指定するようにプロンプトが表示された場合は、デフォルト値を受け入れます。 (ワークスペースとは、WebSphere Integration Developer がプロジェクトを保管するディレクトリーのことです。)
   c. WebSphere Integration Developer ウィンドウが表示されたら、「**ビジネス・インテグレーション・パースペクティブへジャンプ**」をクリックします。
2. 管理コンソールを開始します。
   a. ビジネス・インテグレーション・パースペクティブで、「**サーバー**」タブをクリックします。
   b. サーバーで「**開始済み**」という状況が表示されない場合は、サーバーの名前 (例えば、「**WebSphere Process Server**」) を右クリックして、「**開始**」をクリックします。 サーバーの状況が `Started` になるのを待ちます。

c. サーバーの名前を右クリックし、「**管理コンソールの実行**」をクリックします。

d. 管理コンソールにログオンします。管理コンソールにユーザー ID およびパスワードが必要な場合は、ID およびパスワードを入力して、「**ログイン**」をクリックします。ユーザー ID およびパスワードが必要ない場合は、「**ログイン**」をクリックします。

3. 管理コンソールで、「**セキュリティー**」 → 「**管理、アプリケーション、およびインフラストラクチャーの保護**」をクリックします。

4. 「**認証 (Authentication)**」の下の「**Java Authentication and Authorization Service**」 → 「**J2C 認証データ (J2C Authentication data)**」をクリックします。



5. 認証別名を作成します。

a. 表示された J2C 認証別名のリストで、「**新規作成**」をクリックします。

b. 「**構成**」タブで、「**別名**」フィールドに認証別名の名前を入力します。

c. データベースへの接続の確立に必要なユーザー ID およびパスワードを入力します。

d. 別名の説明をオプションで入力します。

e. 「**OK**」をクリックします。

新規に作成された別名が表示されます。

別名のフルネームを、ノード名も含めてメモしておきます。このフルネームは、後続の構成ウィンドウで使用する名前です。

f. 「**保管**」をクリックします。

結果

ウィザードの後半でアダプター・プロパティーを構成するときに指定する認証別名が作成されました。

プロジェクトの作成

モジュールの作成とデプロイのプロセスを開始するには、WebSphere Integration Developer の外部サービス・ウィザードを開始します。このウィザードは、モジュールに関連付けられたファイルを編成するために使用される、コネクター・プロジェクトを作成します。

始める前に

データベースへの接続の確立に必要な情報を収集済みであることを確認します。例えば、データベースの名前または IP アドレスと、データベースにアクセスするためのユーザー ID およびパスワードが必要です。

このタスクを実行する理由および時期

既存のプロジェクトがある場合は、新規オブジェクトを作成する代わりに、そのプロジェクトを使用できます。ウィザードの開始前に選択してください。

このタスクの手順

1. WebSphere Integration Developer が現在実行されていない場合は、開始します。
   a. 「**スタート**」 → 「**プログラム**」 → 「**IBM Software Development Platform**」 → 「**IBM WebSphere Integration Developer 6.1**」 → 「**IBM WebSphere Integration Developer 6.1**」をクリックします。
   b. ワークスペースを指定するようにプロンプトが表示された場合は、デフォルト値を受け入れます。

      ワークスペースとは、WebSphere Integration Developer がプロジェクトを保管するディレクトリーのことです。
   c. WebSphere Integration Developer ウィンドウが表示されたら、「**ビジネス・インテグレーション・パースペクティブへジャンプ**」をクリックします。
2. 「**ファイル**」 → 「**新規**」 → 「**外部サービス**」をクリックして、外部サービス・ウィザードを開始します。
3. 新規外部サービス (New External Service)ウィンドウで、「**アダプター**」を選択して「**次へ**」をクリックします。
4. 「アダプターの選択」ウィンドウで、「**IBM WebSphere Adapter for JDBC (IBM : *version*)**」を選択します。*version* は、使用するアダプターのバージョン (例えば、6.1) です。
5. 「**次へ**」をクリックします。
6. アダプターのインポート (Adapter Import)ウィンドウで、「**コネクター・プロジェクト**」にあるデフォルト・プロジェクト名を受け入れるか、別の名前を入力します。
7. 「**ターゲット・ランタイム (Target runtime)**」で、モジュールをデプロイするサーバーのタイプを選択します。ウィザードは、そのサーバーに対して適切な成果物を作成します。
8. 「**次へ**」をクリックします。 必要なファイルおよびライブラリー (Required Files and Libraries)ウィンドウが表示されます。

結果

アダプターの RAR ファイルを含む新規のコネクター・プロジェクトが作成されます。プロジェクトは、ビジネス・インテグレーション・パースペクティブにリストされます。

次のタスク

外部サービス・ウィザードでの作業を続行します。次のステップでは、データベース固有のファイルをプロジェクトに追加します。

外部ソフトウェア依存関係の追加

アダプターがデータベースと通信できるようにするには、データベースの特定ファイルのコピーが必要です。外部サービス・ウィザードを使用して、JDBC ドライバーと必要なネイティブ・システム・ライブラリー・ファイルが格納されている JAR ファイルの場所を指定します。

始める前に

この作業を行うには、WebSphere Integration Developer で外部サービス・ウィザードを実行しておく必要があります。

このタスクを実行する理由および時期

モジュール構成時にこのタスクを実行するだけでなく、場合によっては、WebSphere Process Server または WebSphere Enterprise Service Bus にファイルをデプロイする必要があります。

このタスクの手順

1. データベース管理者またはデータベース・ソフトウェアの Web サイトから、ご使用のデータベース・ソフトウェアおよびオペレーティング・システムの JDBC ドライバー固有ファイルとネイティブ・ライブラリーを入手します。 必要なファイルの種類は、データベース・サーバーによって異なります。次の表に、一般的なデータベース・ソフトウェアに必要な JDBC ドライバー・ファイルをリストします。

表 11. 一般的なデータベース・ソフトウェアの JDBC ドライバー・ファイル

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
IBM DB2 Universal Database™ for Linux®、UNIX®、および Windows	IBM DB2 Universal (タイプ 4)	db2jcc.jar db2jcc_license_cu.jar	なし
IBM DB2 for z/OS	IBM DB2 Universal (タイプ 4)	db2jcc.jar db2jcc_license_cisuz.jar	なし
IBM DB2 for i5/OS®	IBM Toolbox for Java リモート・ドライバー	jt400.jardb2jcc_license_cisuz.jar	なし
	IBM DB2 Universal ドライバー	db2jcc.jar	なし
	IBM Toolkit for Java ネイティブ・ドライバー*	db2_classes.jar	なし
Oracle	Thin ドライバー	ojdbc14.jar	なし
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	なし

表 11. 一般的なデータベース・ソフトウェアの JDBC ドライバー・ファイル (続き)

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
* IBM Toolkit for Java ネイティブ・ドライバーを使用してアダプターの実行時にデータベースに接続できますが、ウィザードの実行中に使用して接続することはできません。ディスカバリー・プロセス中は、 IBM Toolbox for Java リモート・ドライバーか、または IBM DB2 Universal ドライバーを使用する必要があります。ただし、実行時にネイティブ・ドライバーを使用するようにモジュールを構成できます。この構成は、「サービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)」ウィンドウで行います。			

2. 「必要なファイルおよびライブラリー (Required Files and Libraries)」ウィンドウで、プロジェクトに必要な JDBC ドライバー固有ファイルの場所を指定します。ウィザードは、アダプターで使用するプロジェクトにこれらのファイルをインポートします。
   a. 「**JDBC ドライバー JAR ファイル**」で、「**追加**」をクリックして、JDBC ドライバー・ファイルを選択します。
   b. JDBC タイプ 2 ドライバーを使用する予定の場合は、「**システム・ライブラリー**」で「**追加**」をクリックして、データベース・サーバーへのアクセスに必要な、ネイティブ・ システム・ライブラリーを追加します。 JDBC タイプ 4 ドライバーを使用する場合は、このフィールドを空のままにします。
3. 「**次へ**」をクリックします。 処理指示 (Processing Direction)ウィンドウが表示されます。

結果

モジュールおよび組み込みアダプターに必要な JDBC ファイルがコネクター・プロジェクトに格納されました。

外部サービス・ウィザードでの作業を続行します。次のステップでは、ウィザードがデータベースに接続するために必要となる情報を設定します。

外部サービス・ウィザードの接続プロパティーの設定

外部サービス・ウィザードがデータベース・オブジェクトをディスカバーするためにデータベース・インスタンスに接続する際に使用する接続プロパティーを指定します。

始める前に

接続プロパティーを構成する前に、外部サービス・ウィザードを始動しておく必要があります。

このタスクを実行する理由および時期

外部サービス・ウィザードでは、ディスカバリーのためのデータベース接続やサービス記述の作成にこれらのプロパティーが必要となります。プロパティーについて詳しくは、 208 ページの『ウィザードの接続プロパティー』を参照してください。

このタスクの手順

1. 「処理指示 (Processing Direction)」ウィンドウで、「**Outbound**」または「**Inbound**」を選択し、「**次へ**」をクリックします。
2. ディスカバリー構成 (Discovery Configuration)ウィンドウで、ウィザードがデータベースへの接続に使用する接続プロパティーを指定します。
   a. データベース・ソフトウェアのリストで、ご使用の製品およびバージョンを選択します。 「**プロパティー (Properties)**」領域には、データベース固有の接続プロパティーを指定するためのフィールドが表示されます。

      注: IBM DB2 Version 9.1 for z/OS の場合、「**V8 (新機能モード) (V8 (New-Function Mode)**」を選択します。
   b. 「**JDBC ドライバー・タイプ**」で、使用する JDBC ドライバーのタイプを選択します。

      注: IBM DB2 for i5/OS の場合、 `AS/400 Toolbox for Java` または `IBM DB2 Universal` を選択して、データベース・オブジェクトをディスカバーします。実行時にサーバー上のローカル・アクセスでネイティブ・ドライバーを使用するように、後でモジュールを構成できます。
   c. 「**データベース**」に、データベース名を指定します。Oracle データベースの場合、これはシステム ID (SID) です。
   d. 「**ホスト名**」には、データベース・サーバーのホスト名または IP アドレスを指定します。IP アドレスを IPv6 形式で指定する場合は、アドレスを大括弧 ([]) で囲んでください。
   e. 「**ポート番号**」で、データベースに接続するためのポート番号を指定します。
   f. DB2、Oracle、および Microsoft SQL Server データベースで、「**JDBC ドライバー・タイプ**」で名前付きドライバーを選択した場合、ウィザードは「**JDBC ドライバー・クラス名**」に対してデフォルト値を指定し、他の接続フィールドから「**データベース URL**」を作成します。任意のデータベース・ソフトウェアおよび他の特定のドライバーについて、ドライバー 「**その他**」を選択した場合、ドライバー・クラス名およびデータベース URL を指定する必要があります (ただし、データベース URL の一部が入力済みの場合があります)。 210 ページの『データベース URL』および 211 ページの『JDBC ドライバー・クラス名』を参照してください。
   g. 「**追加の JDBC ドライバー接続プロパティー (Additional JDBC driver connection properties)**」で、データベース接続時に設定される追加プロパティーを指定します。1 組以上の *name*:*value* ペアをセミコロン文字 (;) で区切って指定します。 例えば、次のようになります。

      ```
      loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY
      ```

   接続情報はディスカバリー・プロセスでのみ使用されます。ウィザードの後のほうでは、実行時用として別の接続情報を指定できます。
3. ウィザードからデータベースに接続するために使用するユーザー名とパスワードを、「**ユーザー名**」および「**パスワード**」に入力します。 このユーザー名はディスカバリー・プロセスでのみ使用され、保存されません。ウィザードの後のほうでは、実行時用として別のユーザー名およびパスワード、または別の認証方式を指定できます。

4. 「**ビジネス・オブジェクト名のプレフィックス**」に、ビジネス・オブジェクト名の先頭に付けるストリングを入力します。
5. 実行時にアダプターの双方向言語サポートを有効にするには、以下の手順を実行します。
   a. 「**拡張**」をクリックします。
   b. 「**BiDi プロパティー**」で、「**BiDi 変換**」を選択します。
   c. 順序付けスキーマ、テキスト方向、対称スワッピング、文字シェーピング、および数字シェーピングの各プロパティーを設定して、双方向変換の実行方法を制御します。
6. ウィザードのログ・ファイルの場所またはログに格納される情報量を変更するには、「**ウィザードのロギング・プロパティーを変更する (Change logging properties for wizard)**」をクリックして、以下の情報を指定します。
   - 「**ログ・ファイル出力場所 (Log file output location)**」に、ウィザードのログ・ファイルの場所を指定します。
   - 「**ロギング・レベル (Logging level)**」に、記録するエラーの重大度を指定します。

   このログ情報はウィザードでのみ使用されます。実行時は、アダプターはサーバーの標準ログ・ファイルおよびトレース・ファイルにメッセージおよびトレース情報を書き込みます。
7. 「**次へ**」をクリックします。

   ウィザードに、com.ibm.adapter.framework.BaseException を示すエラー・ウィンドウが表示された場合、データベース・サーバーに接続できません。メッセージには、問題の考えられる原因について追加情報が含まれています。また、「**ログ・ファイル出力場所 (Log file output location)**」で指定されたディレクトリーにあるログを確認できます。接続情報が正しいことを確認してください。

結果

外部サービス・ウィザードは、データベースに接続し、「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウを表示します。

次のタスク

ウィザードでの作業を続行します。次のステップでは、データベースを調べて、ウィザードでビジネス・オブジェクトを作成する対象となるオブジェクトを見つけます。

Outbound 処理のモジュールの構成

アダプターを Outbound 処理に使用するようにモジュールを構成するには、WebSphere Integration Developer 内で外部サービス・ウィザードを使用して、 データベースからビジネス・オブジェクトおよびサービスを検出して選択し、ビジネス・オブジェクト定義および関連する成果物を生成します。

データベース・オブジェクトのディスカバリー

データベースに接続した後は、データベース・オブジェクトを検索するクエリーを実行します。ディスカバーされたオブジェクトのツリーを参照して、データベース内のオブジェクトの構造を理解します。また、フィルターを使用して、参照したいデータベース・オブジェクトのみを表示します。ユーザー定義データベース・クエリーおよびユーザー定義バッチ SQL ステートメントとして作成するビジネス・オブジェクトの数を定義します。

始める前に

データベースのアクセスに必要なプログラムのデータ要件を知っていなければなりません。例えば、データベースに関する以下の情報が必要になります。

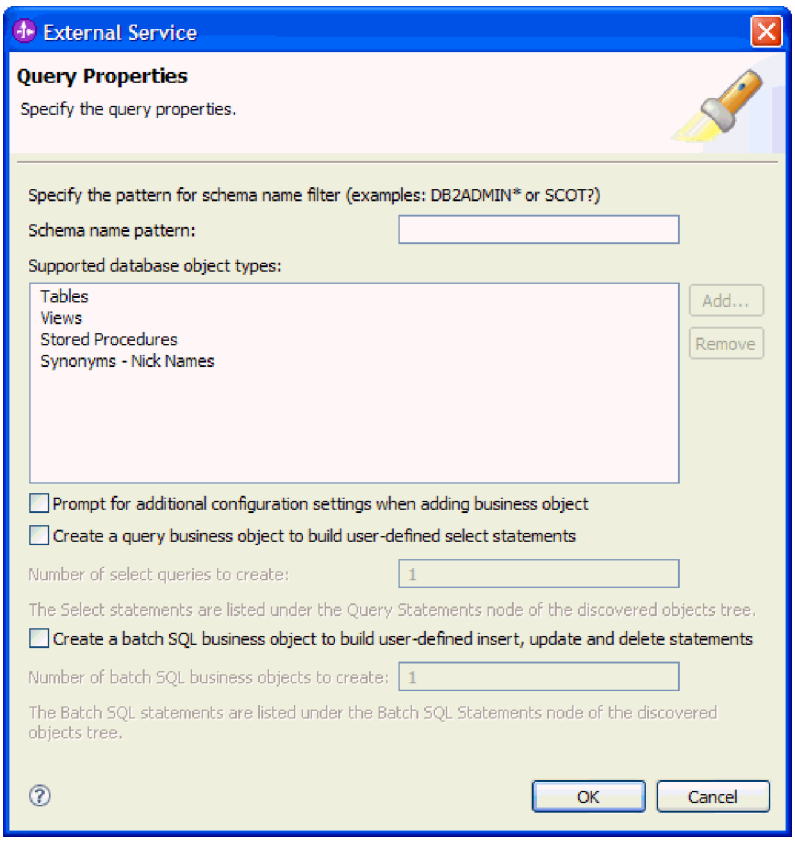
- モジュールはどのスキーマにアクセスする必要があるか
- これらのスキーマ内のどのタイプのデータベース・オブジェクトにアクセスする必要があるか
- アクセスするには、どのテーブル、ビュー、シノニムまたはニックネーム、ストアード・プロシージャー、ストアード関数が必要か
- 作成する必要があるクエリー・ビジネス・オブジェクトおよびバッチ SQL ビジネス・オブジェクトの数。パラメーター値およびそのパラメーターのサンプル・データベース値を含む

このタスクを実行する理由および時期

このタスクは、外部サービス・ウィザードのオブジェクトのディスカバリーと選択 (Object Discovery and Selection)で開始します。

このタスクの手順

1. オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウで、「**照会の編集 (Edit Query)**」をクリックします。 照会プロパティー (Query Properties)ウィンドウが表示されます。



以下の手順に従って、「照会プロパティー (Query Properties)」ウィンドウを使用して、次のタスクを実行します。

- データベース・スキーマのサブセットを検索することにより、検索時間を削減します。
- 1 つ以上のタイプのデータベース・オブジェクトを検索から除外します。
- データベース内の情報からは自動的に判別できないアプリケーション固有情報の入力をウィザードがユーザーに求めるようにします。
- 作成するクエリー・ビジネス・オブジェクトおよびバッチ SQL ビジネス・オブジェクトの数を指定します。

注: バージョン 6.0.2、フィックスパック 2 では、作成したいラッパー・ビジネス・オブジェクトの数もこのウィンドウで指定できました。バージョン 6.1.0 では、ウィザードは後で wrapper 情報の入力を求めます。

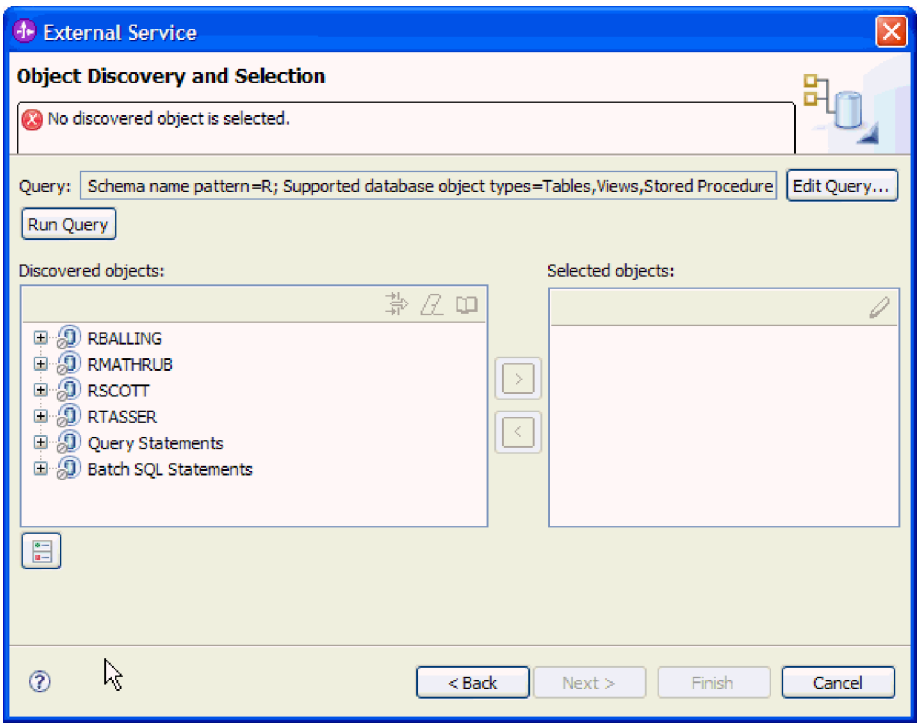
2. 取得されるデータベース・スキーマの数を制限するには、「**スキーマ名パターン**」にスキーマの名前またはスキーマ名パターンを入力します。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。 クエリーを実行すると、そのストリングで始まるスキーマか、またはスキーマ名パターンに一致するスキーマのみが表示されます。スキーマ名パターンを指定しない場合は、データベース内のすべてのスキーマが表示されます。データベースに多数のスキーマがある場合は、フィルターを使用してディスカバリー・プロセスを高速化できます。
3. 検索から 1 つ以上のタイプのオブジェクトを除外するには、除外したいオブジェクトのタイプ (テーブル、ビュー、ストアード・プロシージャーおよびスト

アード関数、シノニムまたはニックネーム) を「**サポートされるデータベース・オブジェクト・タイプ**」で選択して、「**除去 (Remove)**」をクリックします。元に戻したい場合は、「**追加**」をクリックして、オブジェクト・タイプをもう一度追加します。 特定のタイプのデータベース・オブジェクトにのみアクセスする必要がある場合は、必要のないオブジェクトを除外することで、ディスカバリー・プロセスを高速化できます。

4. 「**ビジネス・オブジェクトの追加時に追加構成設定のプロンプトを出す**」を選択します。 これにより、作成するビジネス・オブジェクトのリストにデータベース・オブジェクトを追加するたびに、そのオブジェクトに関してユーザーが構成可能なすべてのアプリケーション固有情報を入力するよう求められます。例えば、このオプションを選択した場合は、ビジネス・オブジェクトの単純親子階層を作成するプロセスが順を追って案内されます。テーブル・ビジネス・オブジェクトが、2 つの異なるテーブル (つまり、2 つの親ビジネス・オブジェクトを持つ) の属性を参照する 2 つの属性を持っている階層が必要な場合は、WebSphere Integration Developer から起動されるツールであるアセンブリー・エディターで構成を完了する必要があります。

   重要: このオプションを選択しない場合、ウィザードは必須情報のみを入力するようプロンプトを出します。アセンブリー・エディターを使用して、ビジネス・オブジェクトの構成を完成させてください。

5. ユーザー定義データベース・クエリーを実行するビジネス・オブジェクトを作成するには、「**ユーザー定義の select ステートメントを作成するためのクエリー・ビジネス・オブジェクトを作成する**」を選択して、作成するクエリー・ビジネス・オブジェクトの数を入力します。 ここではビジネス・オブジェクトの数のみを指定します。このビジネス・オブジェクトについては、後で名前およびその他の詳細の入力をウィザードから求められます。
6. 一連の SQL ステートメントを実行するビジネス・オブジェクトを作成するには、「**ユーザー定義の insert、update、および delete ステートメントを作成するためのバッチ SQL ビジネス・オブジェクトを作成する**」を選択して、作成するバッチ SQL ビジネス・オブジェクトの数を入力します。 ここではビジネス・オブジェクトの数のみを指定します。このビジネス・オブジェクトについては、後で名前およびその他の詳細の入力をウィザードから求められます。
7. 「**OK**」をクリックして、データベース・クエリーへの変更を保存します。
8. オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウで、「**クエリーの実行 (Run Query)**」をクリックします。これによりこのクエリーを使用してデータベース・オブジェクトがディスカバーされ、クエリーおよびバッチ SQL ビジネス・オブジェクトのテンプレートが作成されます。標準的なクエリーの実行結果を次の図に示します。



「**ディスカバーされたオブジェクト (Discovered objects)**」ペインには、ディスカバーされたデータベース・オブジェクトがリストされます。

9. 「**ディスカバーされたオブジェクト (Discovered objects)**」リストで、スキーマ・ノードおよびその下の「**テーブル**」、「**ビュー (Views)**」、「**ストアード・プロシージャー (Stored Procedures)**」、「**シノニム - ニックネーム (Synonyms - Nicknames)**」の各ノードを展開するには、「**+**」 (正符号) をクリックします。これにより、ウィザードによってディスカバーされたデータベース・オブジェクトが表示されます。
10. 「**クエリー・ステートメント (Query Statements)**」および「**バッチ SQL ステートメント (Batch SQL Statements)**」のノードを展開するには、「**+**」 (正符号) をクリックします。これによって、クエリー・ビジネス・オブジェクトおよびバッチ SQL ビジネス・オブジェクトのテンプレートが表示されます。

結果

アダプター、およびクエリー・ビジネス・オブジェクトとバッチ SQL ビジネス・オブジェクトのビジネス・オブジェクト・テンプレートを使用してアクセスできる、データベース・オブジェクトがウィザードによって表示されました。

次のタスク

外部サービス・ウィザードでの作業を続行します。次のステップでは、モジュールで使用するオブジェクトを選択し、各ビジネス・オブジェクトを構成して、ビジネス・オブジェクトの階層を作成します。

ビジネス・オブジェクトの選択および構成

外部サービス・ウィザードで検出されたデータベース・オブジェクトのリストと、指定した照会オブジェクト・テンプレートおよびバッチ SQL オブジェクト・テンプレートを使用することにより、引き続きこのウィザードを使用して、モジュールでアクセスする必要のあるデータベース・オブジェクトを選択します。次に、新規ビジネス・オブジェクトの構成情報を入力します。

このタスクを実行する理由および時期

オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウでは、オブジェクトを任意の順序で選択して構成できます。ただし、子テーブルを選択して構成するには、その前に親テーブルを選択して構成する必要があります。この制限を除けば、オブジェクトを個々に追加することも、複数のオブジェクトを一度に追加することも柔軟に行えます。「**ディスカバーされたオブジェクト (Discovered objects)**」リストのさまざまなノード内のオブジェクトを組み合わせることができます。例えば、テーブル・オブジェクト数個と、ストアード・プロシージャー・オブジェクト、およびクエリー・ステートメントを選択して、それらを同時に追加および構成することができます。

ビジネス・オブジェクトの選択および構成のおおまかな流れは以下のとおりです。

1. 「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウの「**検出オブジェクト (Discovered objects)**」リストで 1 つ以上のオブジェクトを選択します。
2. 「**>**」 (追加) ボタンをクリックします。
3. 「構成プロパティー (Configuration Properties)」ウィンドウが開きます。
   - 1 つのオブジェクトを選択すると、「構成プロパティー (Configuration Properties)」ウィンドウが 1 つ表示されます。

     このウィンドウで、ユーザーが構成可能な属性や、ウィザードによるデータベース検査では検出できないその他の情報をすべて入力したら、「**OK**」をクリックして構成を保存します。
   - 複数のオブジェクトを選択すると、「構成プロパティー (Configuration Properties)」ノートブックが表示されます。選択したオブジェクトごとに 1 ページずつ表示されます。

     各オブジェクトの名前を順にクリックします。ノートブックには、該当するオブジェクトを個別に選択した場合と同じ情報が表示されます。

     重要: すべてのオブジェクトの構成ページで操作が完了するまでは、ノートブックの「**OK**」はクリックしないでください。ウィザードは、必須フィールドすべてに入力されるまではノートブックを閉じませんが、オプション・フィールドに入力しなくてもノートブックを閉じることができます。ウィザードでオプション・フィールドを構成しない場合は、ビジネス・オブジェクト・エディターを使用して、ウィザードを終了した後にそれらのフィールドを構成する必要があります。
4. ウィザードにより、構成されたオブジェクトが「**選択済みオブジェクト (Selected objects)**」リストに追加されます。

ウィザードを終了しない限り、操作を繰り返してモジュールに必要なビジネス・オブジェクトを選択および構成できます。ただし、ウィザードを使用して既存のモジュールにオブジェクトを追加することはできないので、ウィザードを使用し始める前に、ビジネス・オブジェクトを使用するプログラムの要件を明確に理解してください。

テーブル、ビュー、およびシノニムまたはニックネームの選択および構成

モジュールで使用するテーブル、ビュー、およびシノニムまたはニックネームのビジネス・オブジェクトを選択および構成します。

始める前に

テーブル、ビュー、およびシノニムまたはニックネームのビジネス・オブジェクトを選択して構成するには、データベース内のデータの構造や、モジュールがどのデータベース・オブジェクトにアクセスする必要があるかを理解しなければなりません。これには、以下のタイプの情報が該当します。

- テーブル、ビュー、およびシノニムまたはニックネームの構造 (列や、データ型などの列属性を含む)。
- テーブル間の関係 (親子関係のカーディナリティーおよび所有権を含む)。

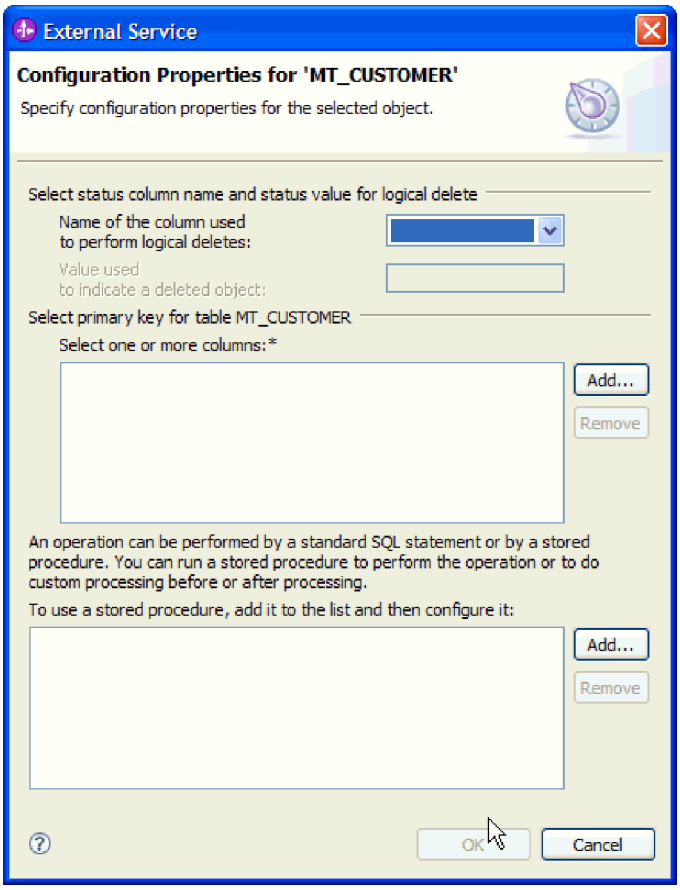
このタスクを実行する理由および時期

このタスクは、外部サービス・ウィザードを使用して実行されます。「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウから操作を開始し、「構成プロパティー (Configuration Properties)」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

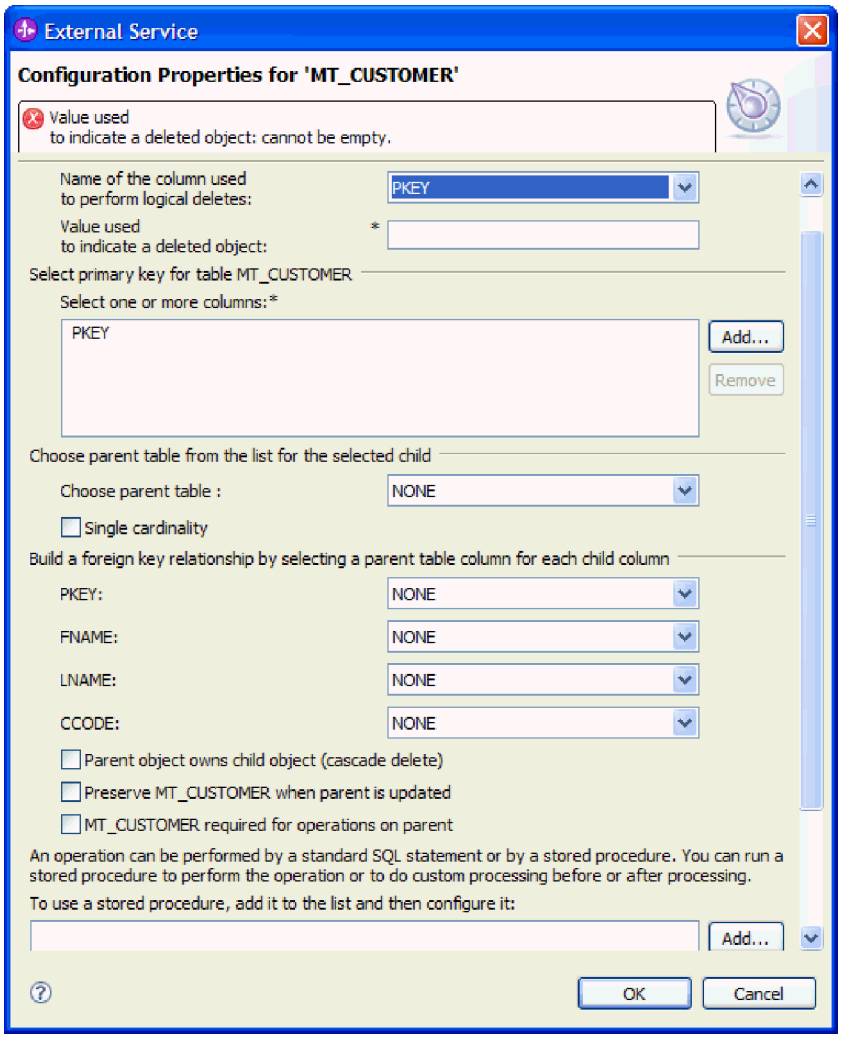
このタスクの手順

1. 「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウの「**検出オブジェクト (Discovered objects)**」リストで、テーブル、ビュー、またはシノニムを 1 つ以上選択し、「**>**」 (追加) ボタンをクリックします。オブジェクトが「**選択されたオブジェクト (Selected objects)**」リストに追加されます。

   以下の 2 つの図に、ビジネス・オブジェクト (テーブル、ビュー、シノニム、またはニックネーム) の標準的な「構成プロパティー (Configuration Properties)」ウィンドウを示します。以下の図に、選択する最初のテーブルまたはテーブル・グループの標準的なウィンドウを示します。



以下の図に、後続のテーブルの標準的なウィンドウを示します。少なくとも 1 つのテーブルを選択して構成した後は、後続テーブルの「構成プロパティー (Configuration Properties)」ウィンドウに、テーブル間の親子階層をオプションで定義することができる領域が表示されます。



オブジェクトの構成時には、拡張構成を必要とする選択を行うと、このウィンドウに追加のフィールドが表示されます。これにより、ウィンドウがスクロールすることがあります。必ずウィンドウのすべてのフィールドを確認してから、「**OK**」をクリックしてください。

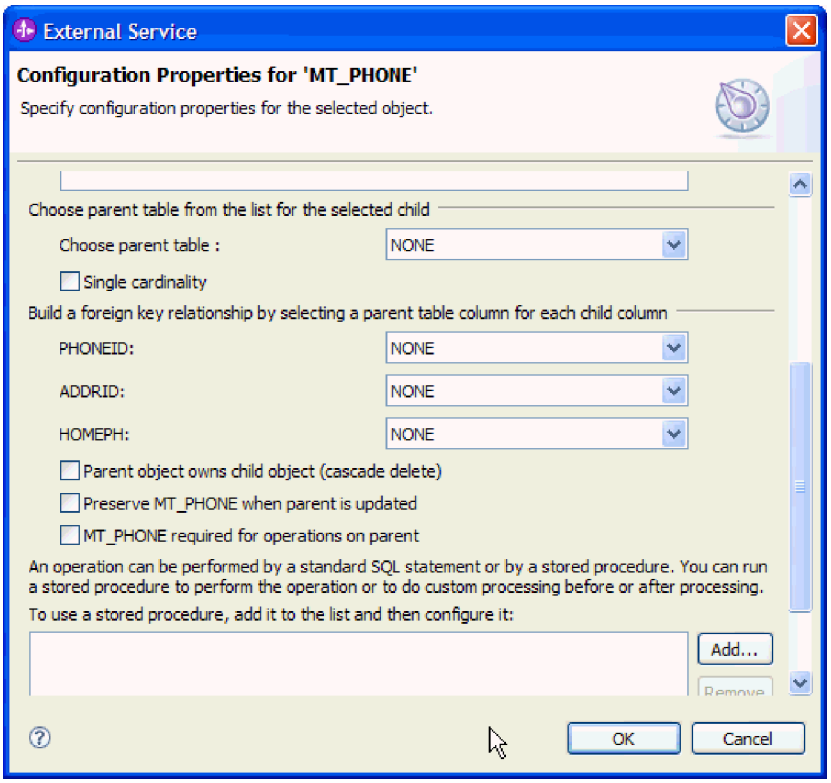
2. 論理削除を示すのに使用される列がテーブルにある場合は、次の手順に従います。
   a. 「**論理削除を実行するのに使用される列の名前**」で列名を選択します。
   b. 「**削除されたオブジェクトを示すために使用する値**」に、行が論理的に削除されていることを示す値を入力します。 この値については、データベース管理者に確認できます。
3. 「**テーブル *table_name* の基本キーの選択**」エリアは、データベース表に基本キーとして指定された列が存在しない場合にのみ表示されます。各テーブル・ビジネス・オブジェクトには、関連付けられたデータベース表にキーがない場合でも、基本キーが定義されている必要があります。データベースで基本キーが定義されている場合、ウィンドウのこのセクションは表示されません。

「**テーブル *table_name* の基本キーの選択**」で「**追加**」をクリックし、テーブル・ビジネス・オブジェクトの基本キーとして使用する列を選択して、「**OK**」をクリックします。テーブルに複合キーがある場合は、複数の列を選択できます。

4. オプションで、ビジネス・オブジェクト間の親子関係を定義します。

   重要: 親子階層を作成するには、親テーブルを最初に構成してから「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウに戻り、子テーブルを選択して構成します。

   以下の図に示す「構成プロパティー (Configuration Properties)」ウィンドウの領域を使用して、親子関係を構成します。これらのフィールドは、構成する最初のテーブルの場合には表示されません。



   a. 「**親テーブルの選択**」で、親テーブルとして使用するテーブルの名前を選択します。 リストに親テーブルが表示されていない場合は、親テーブルがまだ構成されていません。子オブジェクトを構成する前に、戻って親オブジェクトを構成してください。

   b. 関係のカーディナリティーを指定します。

      - テーブルとその親テーブルの間に単一カーディナリティー関係がある場合は、「**単一カーディナリティー**」を選択します。単一カーディナリティー関係では、親はこのタイプの子ビジネス・オブジェクトを 1 つのみ持つことができます。単一カーディナリティー関係は、所有関係を伴って実際の子を表すか、または所有関係を伴わずにルックアップ・テーブルまたはデータベース内の他の対等オブジェクトを表すために使用できます。

- テーブルが複数カーディナリティー関係の場合は、「**単一カーディナリティー**」を選択しないでください。複数カーディナリティー関係では、親がこのタイプの子ビジネス・オブジェクトの配列を持つことができます。

c. 親と子の間に外部キー関係を作成するため、子の列ごとに、親テーブルの外部キーであるかどうかを指定します。
   - 子の列が外部キーでない場合は、「なし」を選択します。
   - 子の列が外部キーの場合は、その子の列に対応する親テーブルの列を選択します。

     注: ウィザードは、1 つの親テーブルのみを構成できます。子テーブルに複数の親テーブルがある場合は、ビジネス・オブジェクト・エディターを使用して、ウィザードを終了した後に残りの親テーブルを構成する必要があります。

d. 親オブジェクトが子オブジェクトを所有している場合、データベース内の子オブジェクトは親が削除されるときに削除されます。子が親により所有されていることを示すには、「**親オブジェクトが子オブジェクトを所有する (カスケード削除)**」を選択します。あるいは、このオプションをクリアして、ルックアップ・テーブルなどの子オブジェクトが、親の削除時に削除されないようにします。

e. Update 操作の一環として子オブジェクトが削除されることのないようにするには、「**親の更新時に *child_table_name* を保持する**」を選択します。

   親テーブルが更新されると、アダプターは入力に存在する子ビジネス・オブジェクトを、データベースから返される子ビジネス・オブジェクトと比較します。デフォルトでは、アダプターは、入力ビジネス・オブジェクト内に存在しない、データベースから返されたすべての子オブジェクトを削除します。

f. デフォルトでは、子ビジネス・オブジェクトを指定せずに、親ビジネス・オブジェクトに対して操作を実行できます。親ビジネス・オブジェクトが変更対象として実行依頼されるときには、その親が子ビジネス・オブジェクトを必ず指定するように要求する場合は、「***Child_table_name* は、親に対する操作で必須**」を選択します。

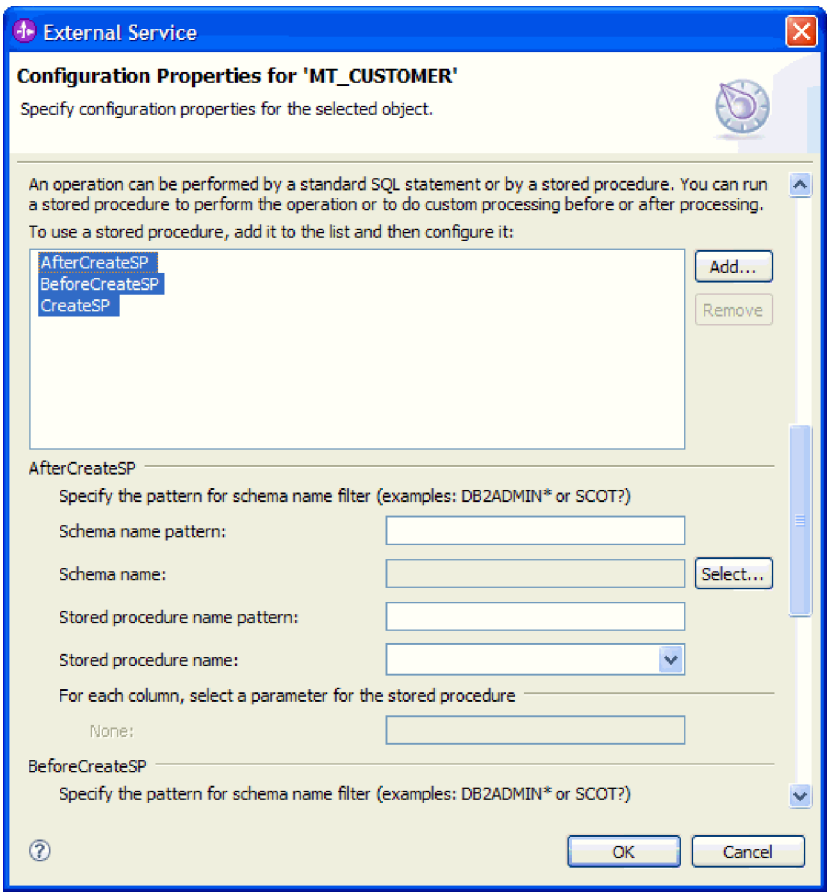
5. 操作を実行するには、アダプターによって生成される標準 SQL ステートメントを使用するか、あるいはデータベース内のストアード・プロシージャーまたはストアード関数を使用します。ストアード・プロシージャーまたはストアード関数を使用する場合は、以下の手順を実行します。

   a. 「**追加**」をクリックします。

   b. 追加 (Add)ウィンドウで、実行するストアード・プロシージャーのタイプを選択します。 操作ごとに、その操作を実行するストアード・プロシージャーと、操作の前後に実行するストアード・プロシージャーを選択できます。例えば、Create 操作の場合は、CreateSP、BeforeCreateSP、および AfterCreateSP のいずれかを指定できます。

   c. 「**OK**」をクリックします。選択したストアード・プロシージャーのタイプが「構成プロパティー (Configuration Properties)」ウィンドウに表示されます。このウィンドウは、各ストアード・プロシージャーの構成用の領域を表示するために拡張されます。新しい領域を表示するには、スクロールダウンする

必要がある場合もあります。



重要: 階層ビジネス・オブジェクトにおいて、その階層のビジネス・オブジェクトごとにストアード・プロシージャーを実行する場合は、ストアード・プロシージャーを、最上位ビジネス・オブジェクトと、ビジネス・オブジェクトの各子ビジネス・オブジェクトまたは配列に別々に関連付ける必要があります。ストアード・プロシージャーを最上位ビジネス・オブジェクトに関連付けても、各子ビジネス・オブジェクトに関連付けないと、その最上位ビジネス・オブジェクトはストアード・プロシージャーで処理されますが、子ビジネス・オブジェクトは標準 SQL 照会を使用して処理されます。

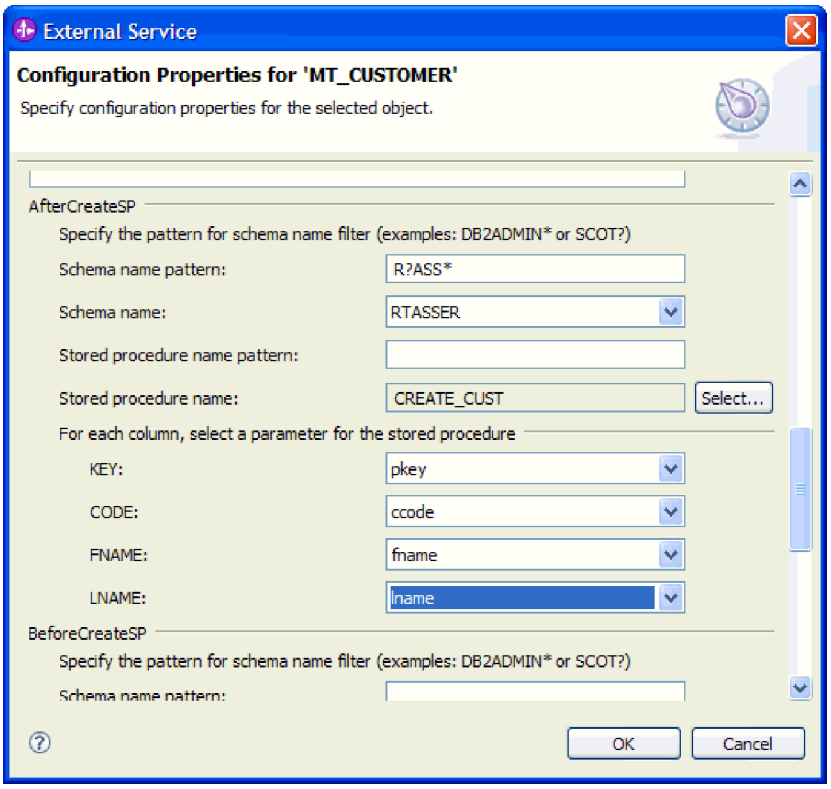
6. 選択したストアード・プロシージャー・タイプごとに、データベース内でのストアード・プロシージャーの名前を指定し、ビジネス・オブジェクトを構成します。
   a. ストアード・プロシージャーを含むスキーマの名前を指定します。 名前は 2 つの方法で指定できます。
      - データベースで検索するデータベース・スキーマをフィルタリングする。
        1) 「**スキーマ名パターン**」で、スキーマの名前またはスキーマ名パターンを入力します。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。名前は大文字小文字が区別されます。
        2) 「**スキーマ名**」で、目的のスキーマの名前を選択します。

- データベースでディスカバーされるすべてのスキーマのリストからスキーマを選択する。
  1) 「**スキーマ名パターン**」フィールドを空のままにします。
  2) 「**スキーマ名**」の横の「**選択 (Select)**」をクリックして、選択 (Select) ウィンドウを開きます。
  3) 選択 (Select)ウィンドウで、目的のスキーマの名前を選択します。オプションで、「**値**」に大/小文字を区別してパターンを入力し、リストを絞り込みます。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。
  4) 目的のスキーマがリストされている場合は、それを選択して「**OK**」をクリックします。

b. ストアード・プロシージャーまたはストアード関数の名前を指定します。 名前は 2 つの方法で指定できます。
   - データベースで検索するストアード・プロシージャーまたはストアード関数をフィルタリングする。
     1) 「**ストアード・プロシージャー名パターン**」で、ストアード・プロシージャーまたはストアード関数の名前を入力するか、または名前パターンを入力します。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。名前は大文字小文字が区別されます。
     2) 「**ストアード・プロシージャー名**」で、目的のプロシージャーの名前を選択します。
   - データベースでディスカバーされるすべてのストアード・プロシージャーのリストから名前を選択する。
     1) 「**ストアード・プロシージャー名パターン**」フィールドを空のままにします。
     2) 「**ストアード・プロシージャー名**」の横で、「**選択**」をクリックして「選択 (Select)」ウィンドウを開きます。
     3) 「選択 (Select)」ウィンドウで、目的のストアード・プロシージャーまたはストアード関数の名前を選択します。オプションで、「**値**」に大/小文字を区別してパターンを入力し、リストを絞り込みます。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。
     4) 目的のストアード・プロシージャーまたはストアード関数がリストにある場合は、それを選択して「**OK**」をクリックします。

   構成プロパティー (Configuration Properties)ウィンドウが拡張して、ストアード・プロシージャーを構成するための領域が表示されます。ウィザードは、データベース内のストアード・プロシージャーを調べることにより、パラメーターのリストを自動生成します。

c. ストアード・プロシージャーのパラメーターごと (左側) に、そのパラメーターでストアード・プロシージャーに渡すテーブル列 (右側) を選択します。次の図は、ストアード・プロシージャーを構成した後のウィンドウの一部分を示します。



7. ウィンドウのすべてのフィールドの操作が完了したら、「**OK**」をクリックします。ビジネス・オブジェクトの構成が保存されます。 定義したビジネス・オブジェクト (テーブル、ビュー、シノニム、およびニックネーム) が、「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウにリストされます。
8. 「**選択済みオブジェクト**」領域にあるオブジェクトの構成を変更するには、オブジェクト名を選択して、 (編集) アイコンをクリックします。

オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウでの作業を続行して、他のタイプのビジネス・オブジェクトを選択および構成します。必要なすべてのビジネス・オブジェクトを選択および構成したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

ストアード・プロシージャーおよびストアード関数の選択および構成

データベース内のストアード・プロシージャーおよびストアード関数に対応するビジネス・オブジェクトを選択および構成します。

始める前に

ストアード・プロシージャーまたはストアード関数のビジネス・オブジェクトを選択して構成するには、データベース内のデータの構造や、モジュールがどのオブジェクトにアクセスする必要があるかを理解しなければなりません。特に、モジュールがアクセスする必要があるストアード・プロシージャーまたはストアード関数に渡すパラメーターについて知る必要があります。

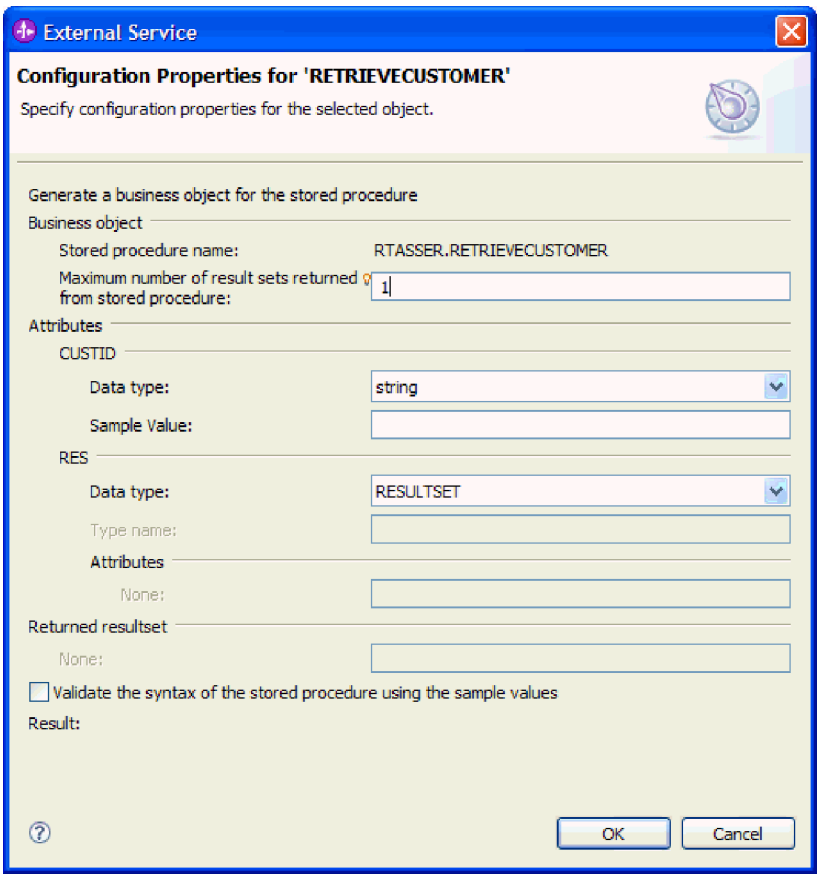
このタスクを実行する理由および時期

このタスクは、外部サービス・ウィザードを使用して実行されます。「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウから操作を開始し、「構成プロパティー (Configuration Properties)」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

このタスクの手順

1. オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウの「**ディスカバーされたオブジェクト (Discovered objects)**」リストで、目的のストアード・プロシージャーまたはストアード関数が含まれるスキーマのノードを展開して、「**ストアード・プロシージャー (Stored Procedures)**」ノードを展開します。
2. オプションで、データベース・オブジェクトをフィルターに掛けて、ツリー内のデータベース・オブジェクトを探しやすくすることができます。
   a. 「**ストアード・プロシージャー (Stored Procedures)**」をクリックして、「**ディスカバーされたオブジェクト (Discovered objects)**」ペインの上部にある  (フィルターの編集または作成) ボタンをクリックします。
   b. フィルター・プロパティー ウィンドウで、「**オブジェクト名フィルター**」内に文字パターンを入力します。 1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。
   c. 「**OK**」をクリックします。
3. 「**ストアード・プロシージャー (Stored Procedure)**」リストからオブジェクトを 1 つ以上選択し、「**>**」 (追加) ボタンをクリックして、「**選択されたオブジェクト (Selected objects)**」リストにオブジェクトを追加します。

   PL/SQL パッケージで定義されているストアード・プロシージャーの場合、そのストアード・プロシージャーは *SPName*(*PackageName*) の形式で表示されます。例えば、EMP_MGMT パッケージに CREATE_DEPT ストアード・プロシージャーが含まれている場合、このストアード・プロシージャーは CREATE_DEPT (EMP_MGMT) としてリストに表示されます。 「オブジェクト」の構成プロパティー (Configuration Properties for 'object')ウィンドウには、ストアード・プロシージャー・ビジネス・オブジェクトの属性がリストされます。この属性のリストには、ストアード・プロシージャーのパラメーターの名前とデータ型、および結果セットが戻される場合はその結果セットに関する情報が表示されます。



4. ストアード・プロシージャーが結果セットを戻す場合は、期待する最大数が「**ストアード・プロシージャーで返される結果セットの最大数**」に表示されていることを確認します。ウィザードは、結果を保持するための結果セット・ビジネス・オブジェクトをこの数だけ作成します。

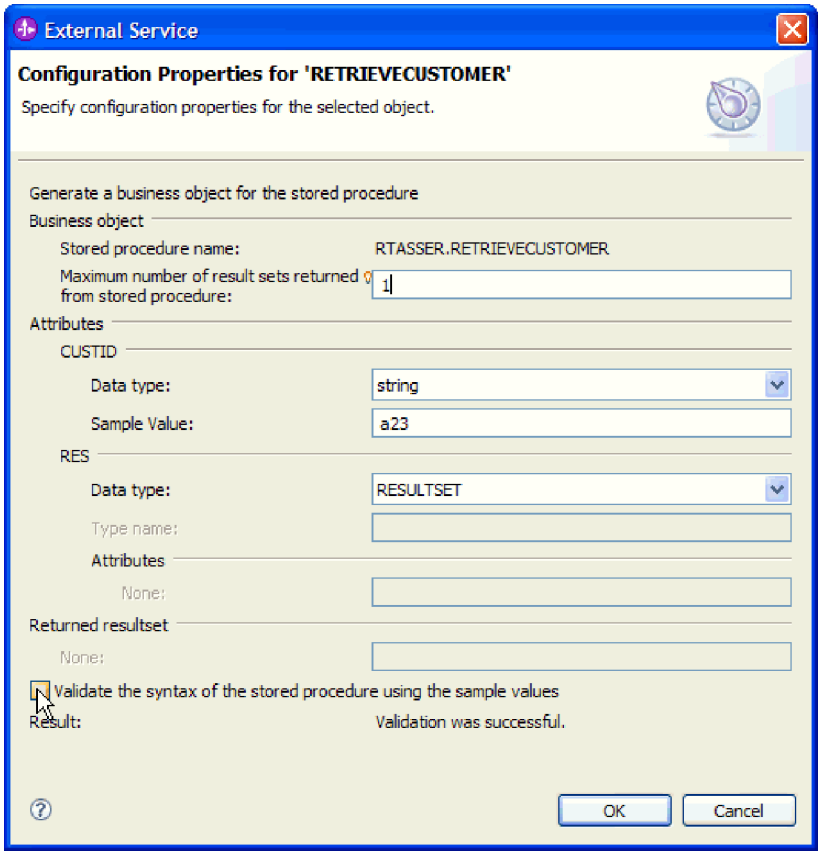
   注: Oracle データベースの場合は、検証後の結果セットの数が正しいことを確認してください。Oracle ドライバーは、必ず情報を返すとは限りません。数が正しくない場合は、検証後、「**OK**」をクリックしてウィンドウを終了する前に、数を設定してください。ウィザードを終了した後は、オプションで、ストアード・プロシージャー・ビジネス・オブジェクトの MaxNumOfRetRS アプリケーション固有情報の設定を検証できます。

5. 各パラメーターを構成します。
   a. 「**データ・タイプ**」に正しいデータ型が表示されていることを確認します。DB2 データベースの場合、ウィザードはパラメーターのデータ型を自動検出できます。他の種類のデータベースの場合は、データ型を手動で選択する必要があります。
   b. 属性が単純データ型の場合は、データベース内の実際の値を「**サンプル値**」に入力します。 例えば、パラメーターが顧客の姓を渡す場合は、データベース内の実際の顧客レコードに格納されている姓を入力する必要があります。
6. すべての属性を構成したら、「**サンプル値を使用してストアード・プロシージャーの構文を検証する**」をクリックします。 「**結果**」に検証結果が表示されます。

「**結果**」に「検証は失敗しました。」と表示された場合は、指定した情報に問題があります。「検証は失敗しました。」の後に表示されているデータベース・サーバーからのエラー・メッセージを参考にして、定義を訂正します。パラメーターおよびサンプル・データのデータ型が正しいことを確認してください。

ワークスペースの .metadata フォルダー内の .log ファイルには、問題に関する追加情報が含まれています。

以下の図に、ストアード・プロシージャーが検証された後のウィンドウを示します。



7. メッセージ「検証は成功しました。」が表示された場合は、「**OK**」をクリックして、ストアード・プロシージャー・ビジネス・オブジェクトの定義を保存します。

   重要: ストアード・プロシージャーまたはストアード関数が結果セットを返す場合は、検証が正常に終了するまで「**OK**」をクリックしないでください。ウィザードは、検証時に返された結果を使用して、その結果を保持するビジネス・オブジェクトを作成します。プロシージャーが検証しない場合、アダプターは実行時に結果セットを返すことができなくなります。
8. 「**選択済みオブジェクト**」領域にあるオブジェクトの構成を変更するには、オブジェクト名を選択して、 (編集) アイコンをクリックします。

結果

ストアード・プロシージャーおよびストアード関数として構成したビジネス・オブジェクトがオブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウにリストされます。

オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウでの作業を続行して、他のタイプのビジネス・オブジェクトを選択および構成します。必要なすべてのビジネス・オブジェクトを選択および構成したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

バッチ SQL ビジネス・オブジェクトの選択および構成

バッチ SQL ビジネス・オブジェクトを選択および構成します。バッチ SQL ビジネス・オブジェクトを使用して、データベース操作を実行する一連の INSERT、UPDATE、および DELETE SQL ステートメントを定義します。

始める前に

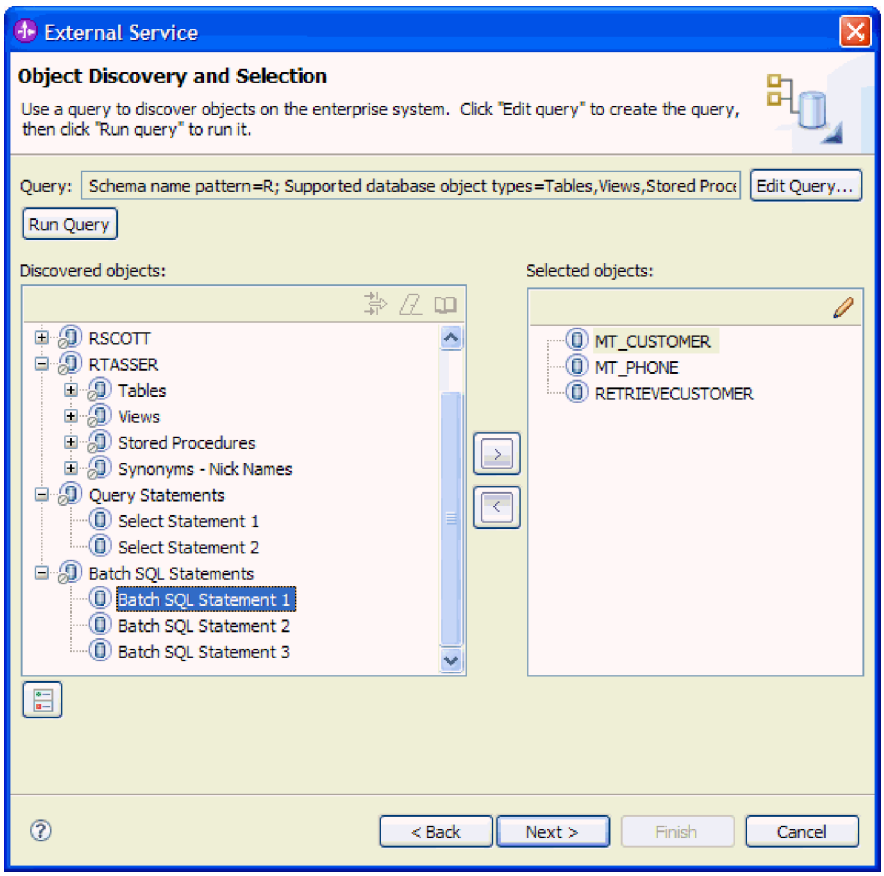
バッチ SQL ビジネス・オブジェクトを構成するには、テーブルおよびビューをはじめとする、データベース内のデータ構造がわかっていなければなりません。SQL ステートメントが処理するべき列の名前およびデータ型を知っておく必要があります。さらに、SQL INSERT、UPDATE、および DELETE ステートメントを記述できなければなりません。

このタスクを実行する理由および時期

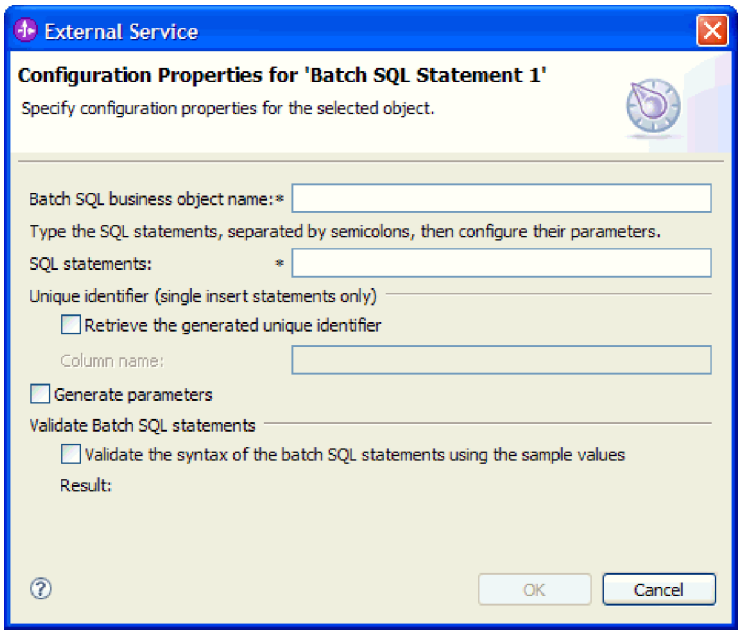
このタスクは、外部サービス・ウィザードを使用して実行されます。「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウから操作を開始し、「構成プロパティー (Configuration Properties)」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

このタスクの手順

1. オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウの「**ディスカバーされたオブジェクト (Discovered objects)**」リストで、「**バッチ SQL ステートメント (Batch SQL Statements)**」ノードを展開します。 このノードには、照会プロパティー (Query Properties)ウィンドウで要求した各バッチ SQL ビジネス・オブジェクトのオブジェクト・テンプレート「**バッチ SQL ステートメント (Batch SQL Statement) *n***」があります。例えば、前述のウィンドウでバッチ SQL ビジネス・オブジェクト数を 3 と指定した場合、「**ディスカバーされたオブジェクト (Discovered objects)**」リストには 3 つのオブジェクト・テンプレートが表示されます (次の図を参照)。



2. オブジェクト・テンプレートを 1 つ以上選択し、「>」 (追加) ボタンをクリックして、オブジェクトを「**選択されたオブジェクト (Selected objects)**」リストに追加します。 次の図は、バッチ SQL ビジネス・オブジェクトの構成プロパティー (Configuration Properties)ウィンドウを示します。このウィンドウは、「>」 (追加) をクリックすると開きます。



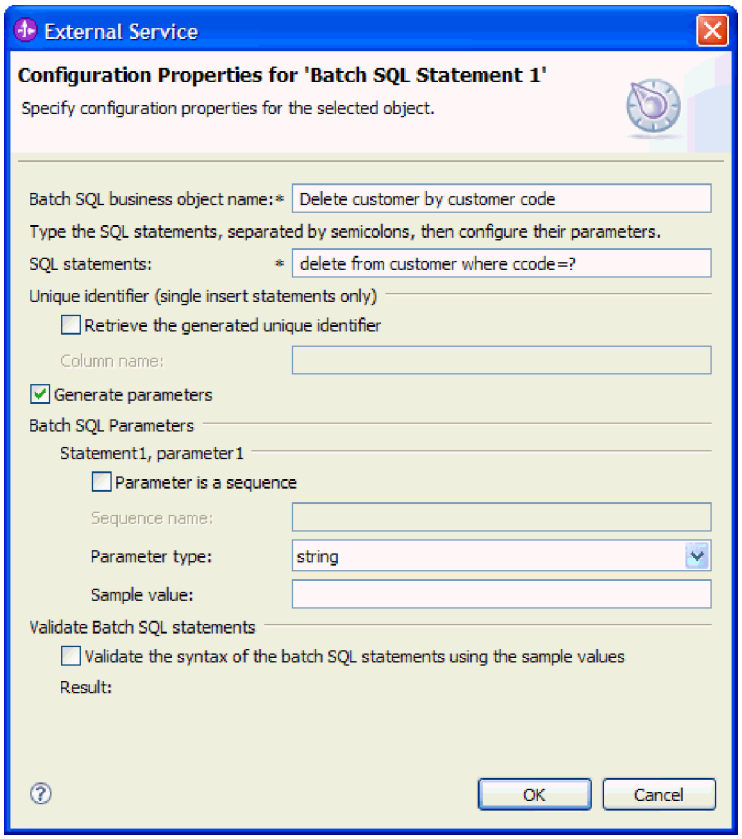
3. 「**バッチ SQL ビジネス・オブジェクト名**」に、ビジネス・オブジェクトの名前を入力します。 名前に空白を含めることはできませんが、各国語文字を使用することができます。
4. 「**SQL ステートメント**」に、1 つ以上の SQL INSERT、UPDATE、または DELETE ステートメントをセミコロン (**;**) で区切って入力します。ステートメント内の各パラメーターは疑問符 (**?**) で示します。 以下の例では、バッチ SQL ビジネス・オブジェクトの柔軟性が示されています。
   - insert into autoid (con1) values (″Smith″)
   - insert into customer (pkey, fname, lname, ccode) values (?, ?, ?, 12345)
   - update customer set fname=?, lname=? where custid=? and ccode is null
   - delete from customer where ccode like ?
   - insert into customer (pkey,ccode,fname,lname) values (?,?,?,?); delete from customer where pkey=?
5. DB2 または Microsoft SQL データベースで、単一の INSERT ステートメントを指定する場合は、オプションで、シーケンスにより自動生成された固有 ID をアダプターが取得するようにできます。ID を取得するようにビジネス・オブジェクトを構成するには、「**生成された固有 ID の取得**」を選択して、その ID が格納されている列の名前を入力します。

   このオプションは、単一の INSERT ステートメントを指定する場合で、かつ指定した列の ID を生成するようにデータベースが構成されている場合に限り有効です。
6. 「**パラメーターの生成**」をクリックします。 ウィンドウが拡張されて、各パラメーターを定義するための領域が表示されます。このため、ウィンドウをスクロールする必要があるかもしれません。ウィンドウを拡張すると、見やすくなります。パラメーターを構成するための領域には、「**ステートメント 1、パラメーター 1 (Statement 1, parameter 1)**」、「**ステートメント *n*、パラメーター *m*(Statement n, parameter m)**」 などのラベルが付いています。

   例えば、以下の SQL ステートメントを指定して、「**パラメーターの生成**」をクリックするとします。`Insert into customer (pkey,ccode,fname,lname) values (?,?,?,?); Delete from Customer where pkey=?`

   構成プロパティー (Configuration Properties)ウィンドウが拡張して、5 つのパラメーターが表示されます。最初のステートメント (`Insert`) には 4 つのパラメーターがあり、これらのパラメーターは「**ステートメント 1、パラメーター 1 (Statement 1, parameter 1)**」から「**ステートメント 1、パラメーター 4 (Statement 1, parameter 4)**」に対応します。2 番目のステートメント (`Delete`) には 1 つのパラメーターがあり、このパラメーターは「**ステートメント 2、パラメーター 1 (Statement 2, parameter 1)**」です。

   次の図は、1 つのパラメーターを持つ 1 つの SQL ステートメントの構成プロパティー (Configuration Properties)ウィンドウを示します。



7. 各パラメーターを、SQL ステートメントで指定した順に構成します。
   - パラメーターが DB2 または Oracle データベースのシーケンス列である場合:
     a. 「**パラメーターはシーケンス**」をクリックします。
     b. 「**シーケンス名**」に、シーケンス列の名前を入力します。

        シーケンス列は、整数データ型でなければならないため、「**パラメーター・タイプ**」は「int」に変更されます。

        シーケンス列にサンプル値は必要ありません。
   - パラメーターがシーケンス列でない場合:
     a. 「**パラメーターはシーケンス**」が選択されていないことを確認します。
     b. 「**パラメーター・タイプ**」で、パラメーターのデータ型を選択します。
     c. 「**サンプル値**」に、パラメーターのサンプル値を入力します。この値は、入力した SQL ステートメントの構文が正しいことを検証するために使用されます。

        INSERT ステートメントでは、パラメーターのデータ型と一致する任意の値を使用できます。

        UPDATE および DELETE ステートメントでは、データベースに存在する値を指定する必要があります。ウィザードは、このサンプル・データでステートメントを実行して、結果セットを取得し、この結果セットを使用して、バッチ SQL ビジネス・オブジェクトの属性を設定します。ウィザー

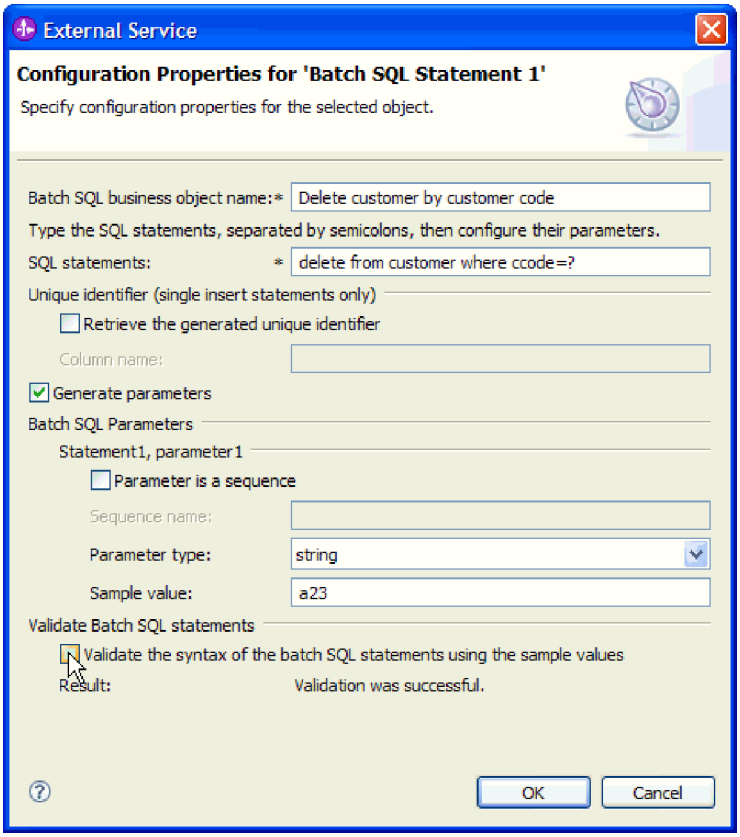
ドはステートメントを実行しますが、結果を COMMIT しないので、データベースのデータが更新または削除されることはありません。

例えば、顧客の姓が格納されている列に対応するパラメーターの場合は、データ型として string を選択し、サンプル値 Smith を指定します。

8. 「**バッチ SQL ステートメントの構文をサンプル値を使用して検証する**」をクリックします。 「**結果**」行に検証結果が表示されます。

「**結果**」に「検証は失敗しました。」と表示された場合は、指定した情報に問題があります。「検証は失敗しました。」の後に表示されているデータベース・サーバーからのエラー・メッセージを参考にして、定義を訂正します。SQL ステートメントの構文、およびパラメーターのデータ型を確認してください。UPDATE および DELETE ステートメントの場合は、サンプル・データがデータベースに存在することも確認してください。

次の図は、検証済みのバッチ SQL ビジネス・オブジェクトの「構成プロパティー (Configuration Properties)」ウィンドウを示します。



9. メッセージ「検証は成功しました。」が表示された場合は、「**OK**」をクリックして、バッチ SQL ビジネス・オブジェクトの定義を保存します。

結果

構成したバッチ SQL ビジネス・オブジェクトがオブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウにリストされます。

オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウでの作業を続行して、他のタイプのビジネス・オブジェクトを選択および構成します。必要なすべてのビジネス・オブジェクトを選択および構成したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

クエリー・ビジネス・オブジェクトの選択および構成

モジュールで使用するユーザー定義 SELECT ステートメントのクエリー・ビジネス・オブジェクトを選択および構成します。

始める前に

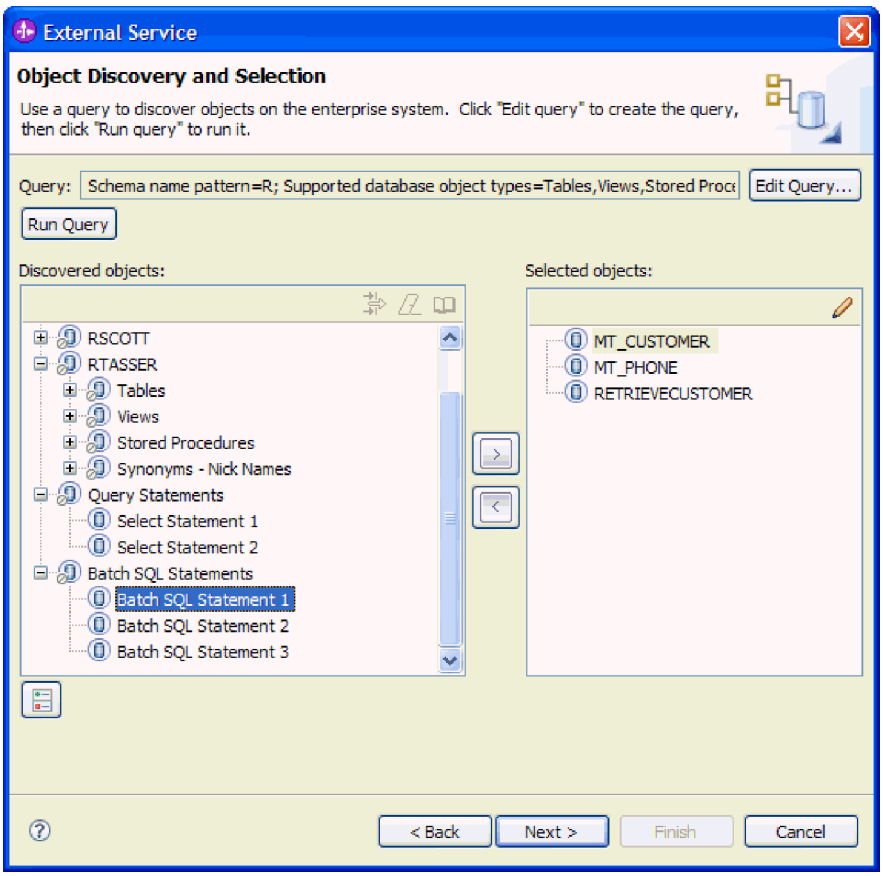
クエリー・ビジネス・オブジェクトを構成するには、テーブルおよびビューをはじめとする、データベース内のデータ構造がわかっていなければなりません。モジュールがアクセスするべき列の名前およびデータ型を知っておく必要があります。さらに、SQL SELECT ステートメントを記述できなければなりません。

このタスクを実行する理由および時期

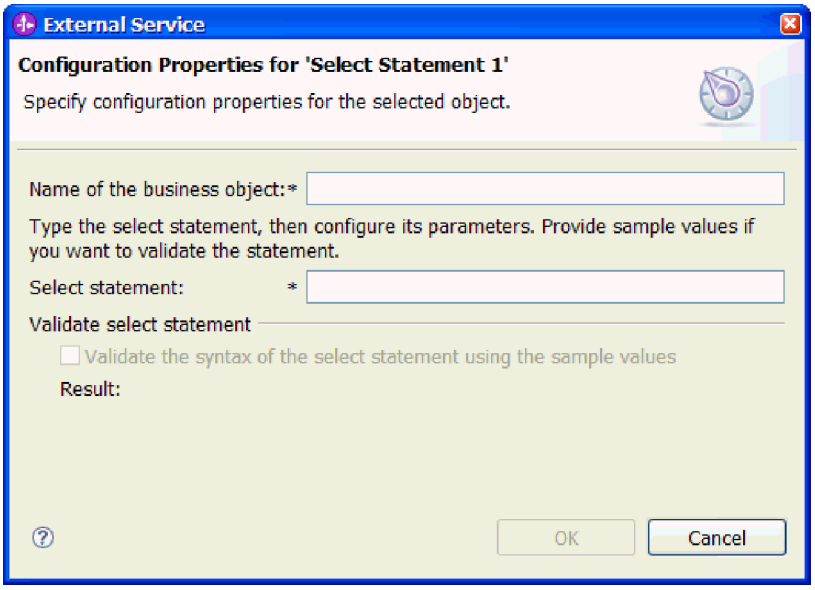
このタスクは、外部サービス・ウィザードを使用して実行されます。「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウから操作を開始し、「構成プロパティー (Configuration Properties)」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

このタスクの手順

1. オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウの 「**ディスカバーされたオブジェクト (Discovered objects)**」リストで、「**クエリー・ステートメント (Query Statements)**」 ノードを展開します。 このノードには、照会プロパティー (Query Properties)ウィンドウで要求した各クエリー・ビジネス・オブジェクトのオブジェクト・テンプレート「**Select ステートメント *n***」があります。例えば、前述のウィンドウでクエリー・ビジネス・オブジェクト数を 2 と指定した場合、「**ディスカバーされたオブジェクト (Discovered objects)**」リストには 2 つのオブジェクト・テンプレートが表示されます (次の図を参照)。

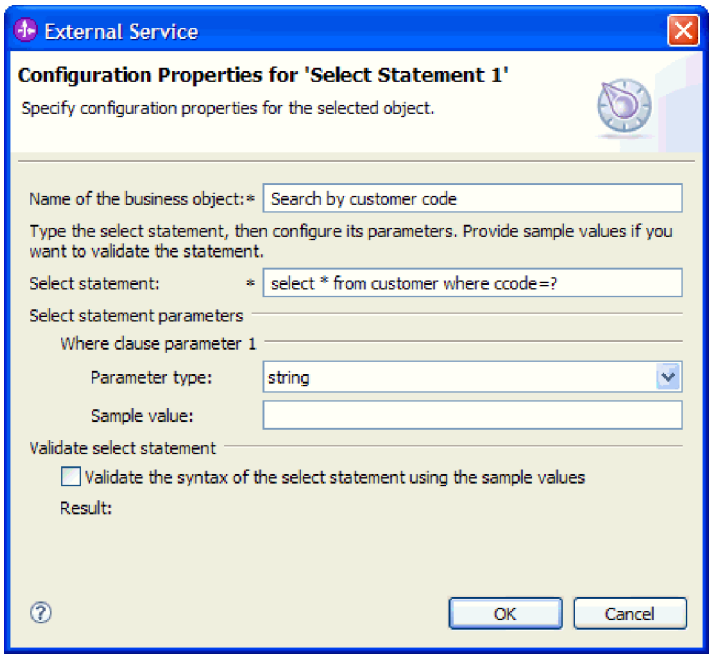


2. オブジェクト・テンプレートを 1 つ以上選択し、「>」 (追加) ボタンをクリックして、オブジェクトを「**選択されたオブジェクト (Selected objects)**」リストに追加します。 次の図は、クエリー・ビジネス・オブジェクトについて「>」 (追加) をクリックすると開く構成プロパティー (Configuration Properties)ウィンドウを示します。



3. 「**ビジネス・オブジェクトの名前**」に、ビジネス・オブジェクトの名前を入力します。 この名前には、スペースおよび各国語文字を使用できます。
4. 「**select ステートメント**」に、実行する SELECT ステートメントを入力します。各パラメーターは疑問符 (**?**) で示します。 次のサンプル SELECT ステートメントは、クエリー・ビジネス・オブジェクトの柔軟性を示しています。
   - `select * from customer where ccode=?`
   - `select * from customer where id=? and age=?`
   - `select * from customer where lname like ?`
   - `select C.pkey, C.fname, A.city from customer C, address A WHERE (C.pkey = A.custid) AND (C.fname like ?)`

   ? を入力するごとに、ウィンドウが拡張して、そのパラメーターの WHERE 節を定義するための領域が表示されます。次の図は、単一パラメーターを持つクエリー・ビジネス・オブジェクトの構成プロパティー (Configuration Properties)ウィンドウを示します。



5. 「**Where 節パラメーター *n***」に、SELECT ステートメントの各パラメーターに関する情報を指定します。
   a. 「**パラメーター・タイプ**」で、パラメーターのデータ型を選択します。
   b. 「**サンプル値**」に、パラメーターのサンプル値を入力します。

   例えば、顧客の姓が格納されている列に対応するパラメーターの場合は、データ型として `string` を選択し、サンプル値 `Smith` を指定します。
6. 「**サンプル値を使用して select ステートメントの構文を検証する**」をクリックします。「**結果**」に検証結果が表示されます。

   「**結果**」に「検証は失敗しました。」と表示された場合は、指定した情報に問題があります。「検証は失敗しました。」の後に表示されているデータベース・サ

ーバーからのエラー・メッセージを参考にして、定義を訂正します。SELECT ステートメントの構文、パラメーターのデータ型、およびサンプル・データを確認してください。

7. メッセージ「検証は成功しました。」が表示された場合は、「**OK**」をクリックして、クエリー・ビジネス・オブジェクトの定義を保存します。

結果

定義したクエリー・ビジネス・オブジェクトがオブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウにリストされます。

オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウでの作業を続行して、他のタイプのビジネス・オブジェクトを選択および構成します。必要なすべてのビジネス・オブジェクトを選択および構成したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

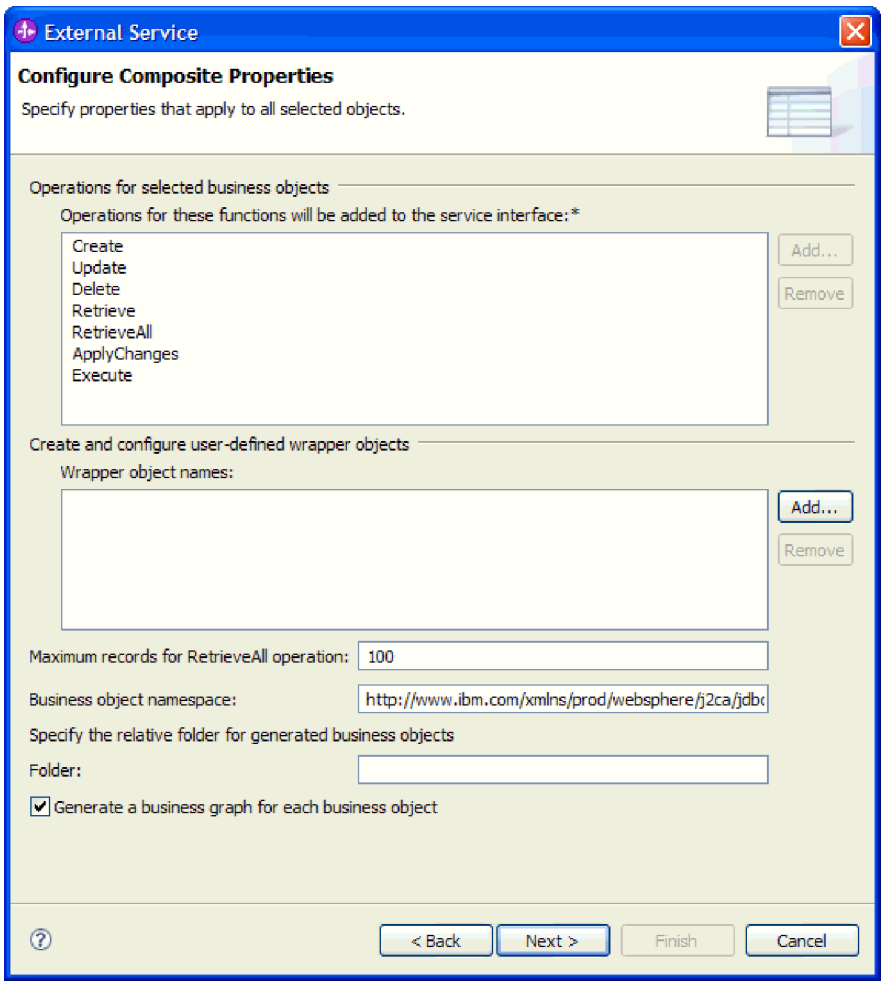
操作のグローバル・プロパティーの設定および wrapper ビジネス・オブジェクトの作成

外部サービス・ウィザードでデータベース・オブジェクトを選択した後、wrapper のビジネス・オブジェクトを定義し、すべてのビジネス・オブジェクトに適用するプロパティーを指定する必要があります。

このタスクの手順

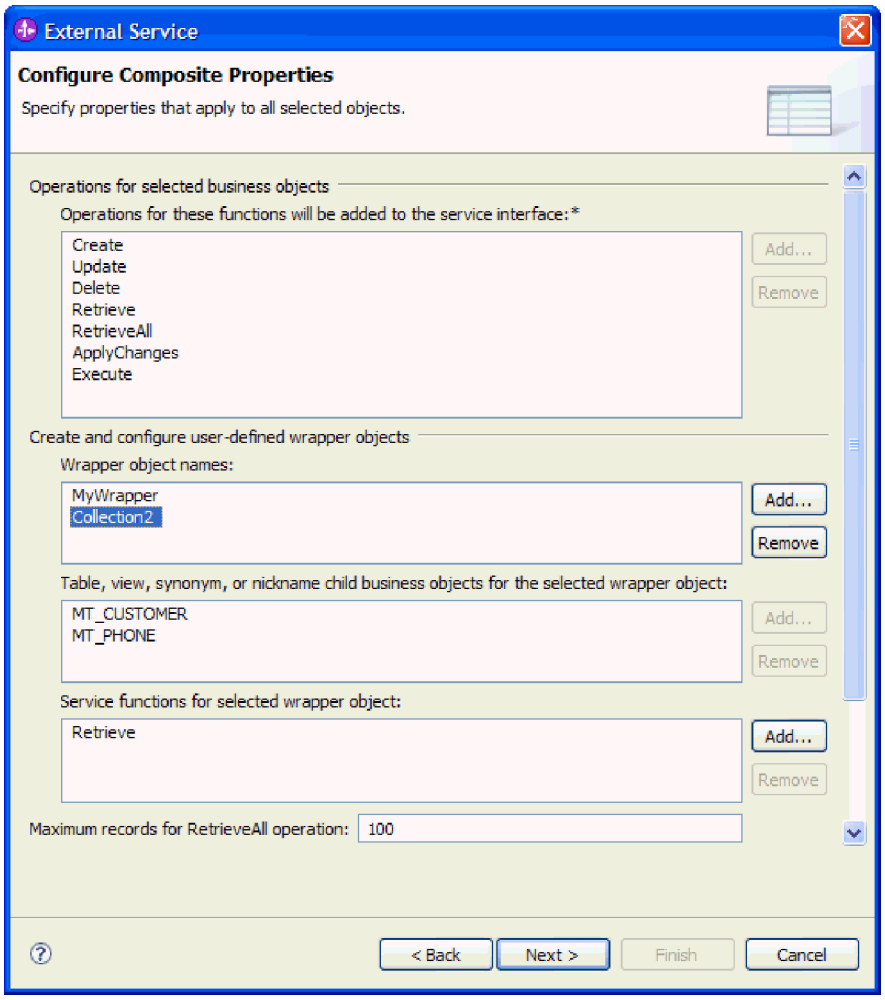
1. オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウの「**選択されたオブジェクト (Selected objects)**」リストに、アプリケーションで使用するすべてのビジネス・オブジェクト (wrapper ビジネス・オブジェクトを除く) が含まれている場合は、「**次へ**」をクリックします。
2. 複合プロパティーの構成 (Configure Composite Properties)ウィンドウで、操作リストを確認します。

   このウィンドウには、前のウィンドウで選択したすべてのビジネス・オブジェクトに対する Outbound サービスで、アダプターがサポートするすべての操作がリストされます。各ビジネス・オブジェクトがすべての操作をサポートするわけではありません。例えば、クエリー・ビジネス・オブジェクトは、RetrieveAll 操作のみをサポートします。ストアード・プロシージャーおよびバッチ SQL ビジネス・オブジェクトは、 Execute 操作のみをサポートします。



3. 不要な操作を除去するには、その操作名を選択して「**除去 (Remove)**」をクリックします。 元に戻したい場合は、「**追加**」をクリックして、除外した操作を復元します。
4. wrapper ビジネス・オブジェクトを作成するため、以下の手順を実行します。
   a. 「**Wrapper オブジェクト名**」領域で、「**追加**」をクリックします。
   b. 「**追加**」ウィンドウで、wrapper ビジネス・オブジェクトの名前を入力して「**OK**」をクリックします。 スペースを使用しないでください。名前には、各国語文字を使用できます。
   c. 「**選択した wrapper オブジェクトのテーブル、ビュー、シノニム、またはニックネーム子ビジネス・オブジェクト**」で、「**追加**」をクリックします。
   d. 「追加」ウィンドウで、wrapper に組み込むビジネス・オブジェクトを 1 つ以上選択して、「**OK**」をクリックします。
   e. 「**選択した wrapper オブジェクトのサービス機能**」で、「**追加**」をクリックします。
   f. 「追加」ウィンドウで、wrapper オブジェクトで実行する操作を 1 つ以上選択して、「**OK**」をクリックします。 RetrieveAll および ApplyChanges 操作は wrapper ビジネス・オブジェクトに適用されないため、リストされません。

g. 作成する wrapper ビジネス・オブジェクトごとに、この手順を繰り返します。 次の図は、2 つの wrapper ビジネス・オブジェクトが定義されている複合プロパティーの構成 (Configure Composite Properties)ウィンドウを示します。このウィンドウでは、一度に 1 つの wrapper ビジネス・オブジェクトについてのプロパティーが表示されます。



5. 「**RetrieveAll 操作の最大レコード数**」に、RetrieveAll 操作で検索するレコード数の上限を入力します。 デフォルト値は 100 です。このプロパティーについて詳しくは、 227 ページの『RetrieveAll 操作の最大レコード数』を参照してください。
6. 「**ビジネス・オブジェクト Namespace**」で、デフォルトのネーム・スペースを受け入れるか、別のネーム・スペースのフルネームを入力します。

   ネーム・スペースは、ビジネス・オブジェクト・スキーマを論理的に分離するため、ビジネス・オブジェクト名の前に付加されます。
7. オプションで、生成されたビジネス・オブジェクトが格納されるフォルダーの相対パスを「**フォルダー**」に入力します。
8. ビジネス・オブジェクトごとにビジネス・グラフを作成する場合は、「**ビジネス・オブジェクトごとにビジネス・グラフを生成**」をクリックします。 ビジネス・グラフは、以下の場合のみ必要になります。

- ApplyChanges 操作を使用する必要がある場合
- バージョン 6.1.0 より前のバージョンの WebSphere Integration Developer で作成されたモジュールにビジネス・オブジェクトを追加する場合

注: 前のバージョンの WebSphere Integration Developer によって作成されたモジュールにビジネス・オブジェクトを追加する場合は、このオプションを選択する必要があります。それ以外の場合は、インターフェースを再接続する必要があります。

9. 「**次へ**」をクリックします。

結果

wrapper ビジネス・オブジェクトを作成して、モジュール内のすべてのビジネス・オブジェクトに適用する情報を設定しました。

次のタスク

ウィザードでの作業を続行します。次のステップでは、実行時に使用するデプロイメント情報、およびサービスをモジュールとして保存するための情報を指定します。

デプロイメント・プロパティーの設定およびサービスの生成

モジュールのビジネス・オブジェクトを選択して構成した後、外部サービス・ウィザードを使用して、アダプターが特定のデータベースに接続するために使用するプロパティーを構成します。ウィザードは、すべての成果物とプロパティー値を保存する、新規のビジネス・インテグレーション・モジュールを作成します。

このタスクを実行する理由および時期

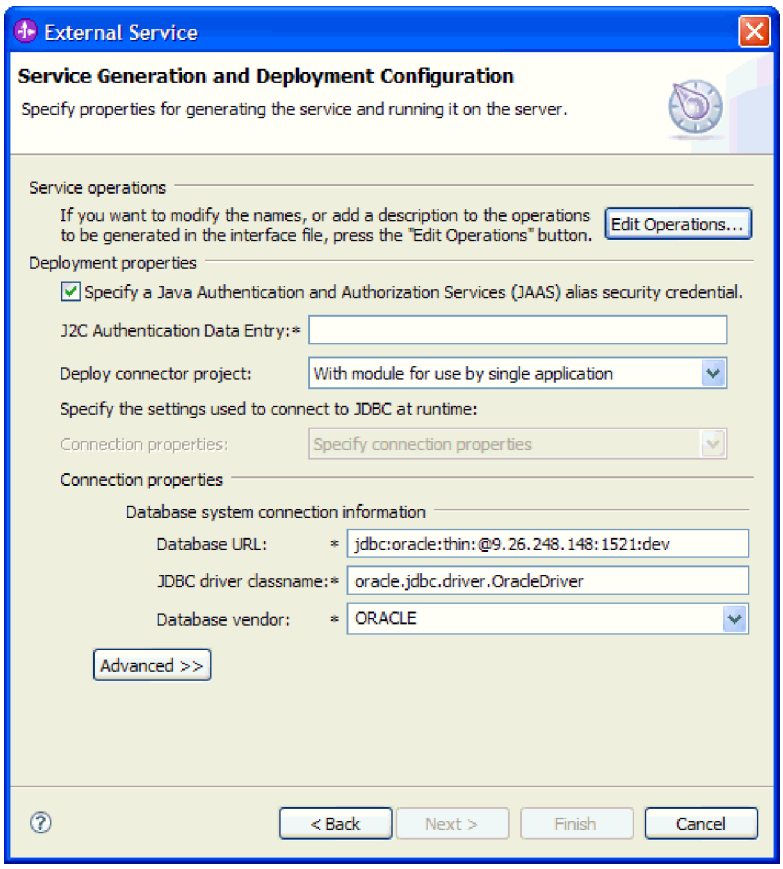
このタスクは、外部サービス・ウィザードのサービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)ウィンドウおよびサービス・ロケーション・プロパティー (Service Location Properties)ウィンドウを使用して実行されます。

このタスクの接続プロパティーは、ウィザードがデータベースに接続するために使用した値に初期化されます。他の値を使用するようにモジュールを構成するには、ここで値を変更します。例えば、実行時に i5/OS で IBM Toolkit for Java ネイティブ・ドライバーを使用するには、ここでドライバー情報を設定します。

このタスクの手順

1. サービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)ウィンドウで、「**操作の編集 (Edit Operations)**」をクリックして、作成するビジネス・オブジェクトの操作の名前を確認するか、操作の説明

を追加します。



2. アダプターが実行時にデータベース・ユーザー名およびパスワードを取得する方法を指定します。
   - J2C 認証別名を使用するには、「**Java 認証・承認サービス(JAAS) の別名のセキュリティー・クレデンシャルの指定 (Specify a Java Authentication and Authorization Services (JAAS) alias security credential)**」を選択し、「**J2C 認証データ入力 (J2C Authentication Data Entry)**」に別名の名前を入力します。

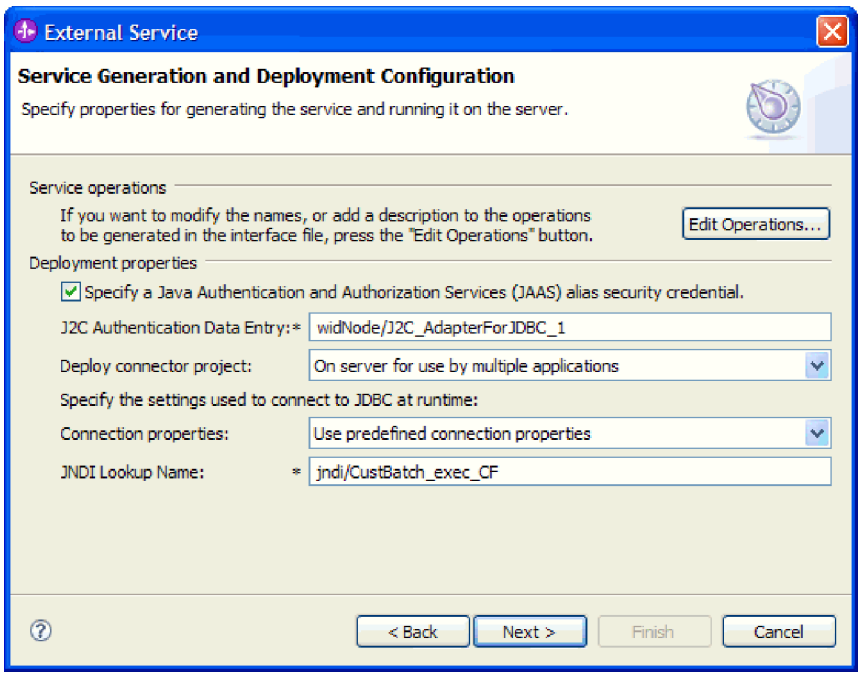
     既存の認証別名を指定することも、(モジュールをデプロイする前に) 認証別名を作成することもできます。名前は大文字小文字が区別されます。また、名前にはノード名が含まれます。
   - サーバー上の既存のデータ・ソースを使用するには、以下の手順を実行します。
     a. 「**Java 認証・承認サービス(JAAS) の別名のセキュリティー・クレデンシャルの指定 (Specify a Java Authentication and Authorization Services (JAAS) alias security credential)**」をクリアします。
     b. 「**拡張**」をクリックします。
     c. 「**代わりの方法で接続情報を指定する (Alternate ways to specify connection information)**」を展開します。
     d. 次のいずれかのフィールド・セットに値を入力します。
        - **DataSource JNDI 名**
        - **XA DataSource 名**および **XA データベース名**

- データベース・ユーザー名およびパスワードをアダプター・プロパティーに保管するよう指定するには、以下のようにします。
  a. 「**Java 認証・承認サービス(JAAS) の別名のセキュリティー・クレデンシャルの指定 (Specify a Java Authentication and Authorization Services (JAAS) alias security credential)**」をクリアします。
  b. 「**拡張**」をクリックします。
  c. 「**データベース・システム接続プロパティー (Database system connection properties)**」の下で、「**ユーザー名**」と「**パスワード**」を入力します。

  注: パスワードをここで指定すると、そのパスワードはアダプター・プロパティーに平文で保存されるので、非認証ユーザーから見られる可能性があります。

3. 「**コネクター・プロジェクトのデプロイ**」で、アダプター・ファイルをモジュールに組み込むかどうかを指定します。次の値のいずれかを選択してください。
   - 「**単一アプリケーションが使用するモジュールとともにデプロイする**」。アダプター・ファイルをモジュール内に組み込むと、モジュールをすべてのアプリケーション・サーバーにデプロイすることができます。組み込みアダプターを使用するのは、組み込みアダプターを使用するモジュールが 1 つある場合か、複数のモジュールでバージョンの異なるアダプターを実行する必要がある場合です。組み込みアダプターを使用すると、他のモジュールのアダプター・バージョンを変更することで、それらのモジュールを不安定にするリスクを生じることなく、1 つのモジュール内でアダプターをアップグレードできます。
   - 「**複数アプリケーションが使用するサーバー上**」。モジュール内にアダプター・ファイルを組み込まない場合は、このモジュールを実行するアプリケーション・サーバーごとにモジュールをスタンドアロン・アダプターとしてインストールする必要があります。複数のモジュールが同じバージョンのアダプターを使用可能で、アダプターを中央の場所で管理する場合は、スタンドアロン・アダプターを使用します。スタンドアロン・アダプターの場合も、複数のモジュールに対して単一のアダプター・インスタンスを実行することにより、必要なリソースが軽減されます。
4. 前のステップで「**複数アダプターが使用するサーバーにデプロイする (On server for use by multiple adapters)**」を指定した場合は、その接続プロパティーの指定方法を指定します。
   - サーバーで管理接続ファクトリーまたはアクティベーション・スペックを手動で作成および構成した場合、または同じ管理接続ファクトリーまたはアクティベーション・スペックのプロパティーを使用して同じデータベースに接続するアプリケーションをすでにデプロイ済みの場合は、その Java Naming and Directory Interface (JNDI) データ・ソースの名前を指定することによって、管理接続ファクトリーまたはアクティベーション・スペックを再利用できます。
     a. 「**接続プロパティー**」で、「**事前定義された接続プロパティーを使用する**」を選択します。
     b. 「**JNDI ルックアップ名**」に、既存の管理接続ファクトリーまたはアクティベーション・スペックの JNDI データ・ソースの名前を入力します。

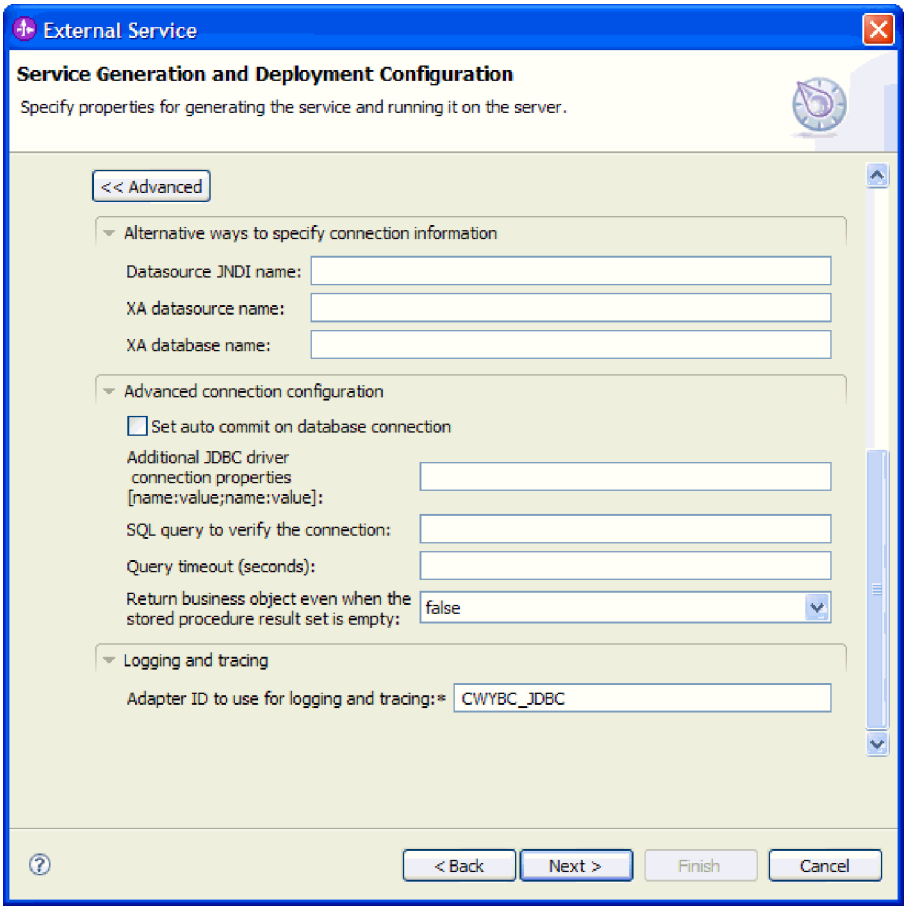
以下の図に、アダプターのスタンドアロン・デプロイメントで管理接続ファクトリーまたはアクティベーション・スペックを再利用する場合の標準的な設定を示します。



c. 「**次へ**」をクリックしてこのタスクを完了します。

- これが、特定のユーザー名とパスワードを使用してデータベースに接続する最初のアプリケーションである場合、または他のアプリケーションとは別々にユーザー名とパスワードを管理する場合は、「**接続プロパティーの指定**」を選択します。

5. 必須の接続プロパティーの値を確認し、必要に応じて変更します。 フィールドは、ウィザードの開始時に指定した接続情報で初期化されます。これらの値を変更して、別のユーザー名およびパスワードを実行時に指定できます。また、別のデータベースに接続できますが、スキーマ名は両方のデータベースで同じでなければなりません。接続プロパティーの形式はデータベース固有です。プロパティーについて詳しくは、217 ページの『管理接続ファクトリー・プロパティー』を参照してください。
6. オプションで、「**拡張**」をクリックして拡張プロパティーを指定します。 拡張セクションをそれぞれ展開して、プロパティーを確認します。以下の図は、「サービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)」ウィンドウの拡張プロパティーを示しています。

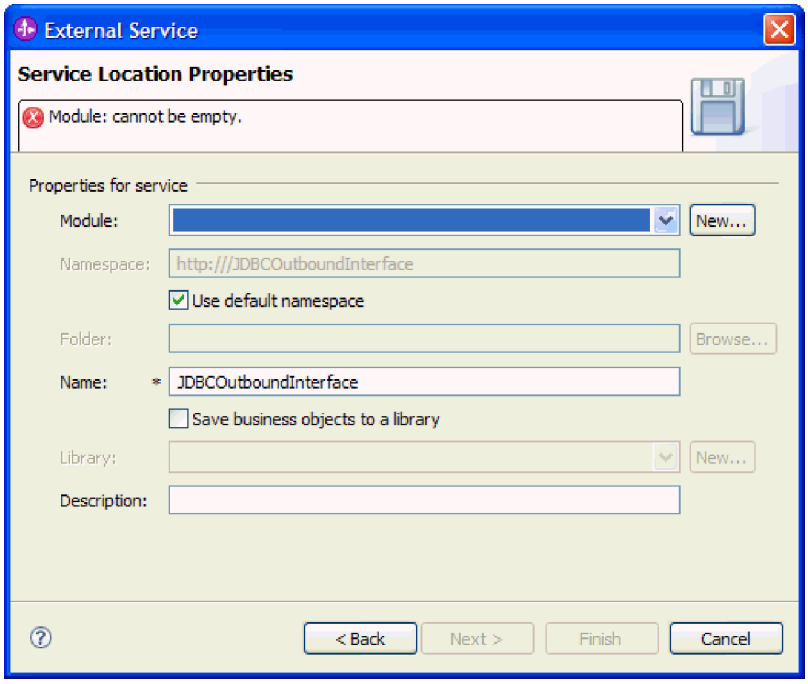


- 「**代わりの方法で接続情報を指定する (Alternate ways to specify connection information)**」で指定されるプロパティーについては、前のステップで説明済みです。
- **拡張接続構成**
  a. データベースの AUTOCOMMIT をオンにする場合は、「**データベース接続時に自動コミットを設定**」を選択します。このプロパティーについて詳しくは、219 ページの『自動コミット (Auto commit) (AutoCommit)』を参照してください。
  b. 「**追加の JDBC ドライバー接続プロパティー (Additional JDBC driver connection properties)**」を設定します。このプロパティーについて詳しくは、218 ページの『追加の JDBC ドライバー接続プロパティー (DriverConnectionProperties)』を参照してください。
  c. 「**クエリー・タイムアウト**」に、アダプターがデータベース・クエリーの応答を待つ時間 (秒) を入力します。このプロパティーについて詳しくは、224 ページの『クエリー・タイムアウト (QueryTimeOut)』を参照してください。
  d. 「**接続を検証するための SQL クエリー**」を設定します。このプロパティーについて詳しくは、225 ページの『接続を検証するための SQL 照会 (PingQuery)』を参照してください。
  e. 「**ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す**」を設定します。このプロパティーについて詳しくは、224 ページの

224 ページの『ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)』を参照してください。

- **ロギングおよびトレース**
  - 「**ロギングおよびトレースで使用するアダプター ID**」に、別のアダプター ID を入力します。このプロパティーについて詳しくは、 218 ページの『ロギングおよびトレースで使用するアダプター ID (AdapterID)』を参照してください。

7. 「**次へ**」をクリックします。 サービス・ロケーション・プロパティー (Service Location Properties)ウィンドウが表示されます。



8. サービス・ロケーション・プロパティー (Service Location Properties)ウィンドウで、作成するモジュールの名前を指定します。新規モジュールを指定することも、既存のモジュールを指定することもできます。
   - 目的のモジュール名が「**モジュール (Module)**」リストに表示されている場合は、その名前を選択します。

     重要: モジュールに、現在構成しているオブジェクトと同じ名前のインターフェースまたはビジネス・オブジェクトが含まれている場合、そのモジュールにある元のインターフェースまたはビジネス・オブジェクトは新しいバージョンによって置き換えられます。
   - それ以外の場合は、新規モジュールを作成します。
     a. 「**新規作成**」をクリックします。
     b. 統合プロジェクト (Integration Project)ウィンドウで、「**モジュール・プロジェクトの作成 (Create a module project)**」を選択して「**次へ**」をクリックします。
     c. モジュールウィンドウで、モジュールの名前を入力します。例えば、JDBCOutboundModule です。

d. サービス記述ファイル (.import ファイルおよび .wsdl ファイル) をモジュールのデフォルト・フォルダーに置きたい場合は、「**デフォルト・ロケーションを使用する (Use default location)**」を選択したままにします。モジュールの別のフォルダーを指定する場合は、このオプションをクリアし、「**参照**」をクリックして、「**ロケーション (Location)**」内の別のフォルダーを指定します。

e. ウィザードを閉じたときに WebSphere Integration Developer のアセンブリー・ダイアグラムでこのモジュールが自動的に開くようにする場合は、「**モジュールのアセンブリー・ダイアグラムを開く (Open module assembly diagram)**」を選択します。それ以外の場合は、このオプションをクリアします。

f. 「**終了**」をクリックすると、新規モジュールが作成されます。

9. ビジネス・オブジェクトに使用するネーム・スペースを指定します。
   - モジュールのビジネス・オブジェクトがデフォルトで得られるネーム・スペースを使用するようにする場合は、「**デフォルト Namespace を使用する (Use default namespace)**」を選択したままにします。
   - 別のネーム・スペースを指定するには、このオプションをクリアして、「**Namespace**」に別の値を入力します。
10. オプションで、新規モジュール内のサービス記述を保管するフォルダーを指定します。「**フォルダー**」にフォルダー名を入力するか、既存フォルダーを参照します。 フォルダー名を指定しない場合、成果物 (インポート・ファイル、XSD および WSDL ファイル) は、モジュールのルート・フォルダー (すなわちモジュール名のフォルダー) に保管されます。
11. 「**名前**」で、デフォルトのインポート名を受け入れるか、あるいは別の名前を入力します。
12. オプションで、ビジネス・オブジェクトをライブラリーに保管して、それを他のモジュールが使用できるようにする場合は、「**ビジネス・オブジェクトをライブラリーに保管する (Save business objects to a library)**」を選択して、ライブラリーの場所を「**ライブラリー (Library)**」に指定します。
13. オプションで、モジュールについて説明したコメントを「**説明**」に入力します。
14. プロパティーの設定が完了したら、「**終了**」をクリックします。
15. 「変更されたモデル (Model Changed)」ウィンドウが表示されたら、「**はい**」をクリックします。

結果

ウィザードは終了します。モジュールがプロジェクトに作成され、成果物が生成されます。

次のタスク

インスタンスによっては、構成を完了するためにアセンブリー・エディターを使用しなければならない場合があります。完了したら、モジュールをテストまたはデプロイできます。

構成の完了

場合によって、ビジネス・オブジェクトの構成を完了するために、手動による構成ステップが必要になります。

このタスクを実行する理由および時期

ウィザードによって生成された成果物をカスタマイズする必要がある場合に、このタスクを実行してください。以下の状態でこのタスクを行う可能性があります。

- ある列の CopyAttribute パラメーターの値を、他の列と同じ値に設定する場合。
- ビジネス・オブジェクトに対して属性を追加または除去する場合。例えば、参照する必要のないデータベース列に対応した単純な属性を除去することによって、ビジネス・オブジェクト設計を簡素化できます。
- 複数の親を持つテーブル・ビジネス・オブジェクトについて追加の親を構成する場合。ウィザードは、1 つのテーブル・ビジネス・オブジェクトにつき、親を 1 つのみ構成します。

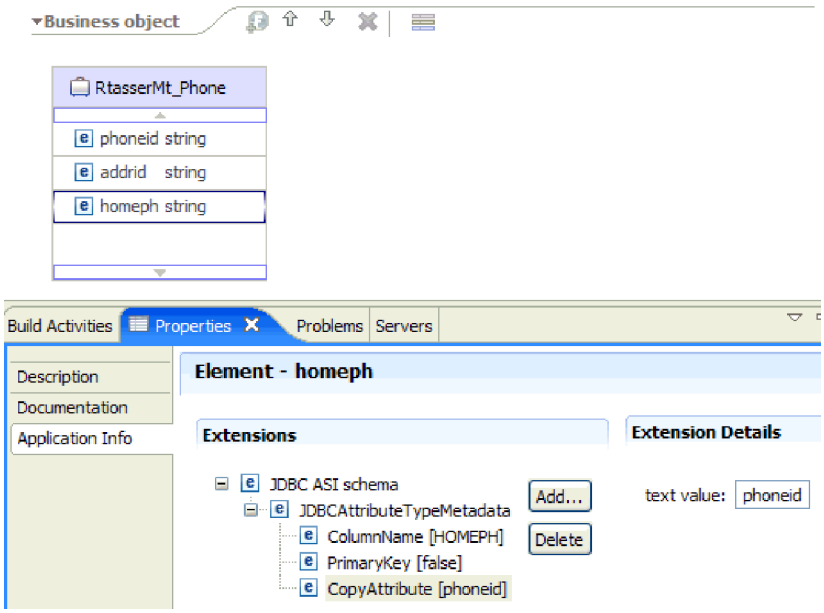
このトピックでは、CopyAttribute パラメーターをテーブル・ビジネス・オブジェクトに設定する、詳細な手順を説明します。ビジネス・オブジェクト構造に対する他の変更は、同じ手法を使用して、実行できます。

CopyAttribute パラメーターは、他の列の値およびアプリケーション固有情報を取り込む対象となる列についての属性のプロパティーに含まれます。例えば、テーブルの新しい行の `contact` 列に、`email` 列と同じ値を組み込むには、`contact` 属性の CopyAttribute パラメーターに `email` を設定します。WebSphere Integration Developer でアセンブリー・エディターを使用して、値を設定します。

このタスクの手順

1. WebSphere Integration Developer の「ビジネス・インテグレーション」パースペクティブで、モジュール名、「**データ・タイプ**」を展開し、テーブル・ビジネス・オブジェクトを見つけます。 ビジネス・オブジェクト名は、データベース・スキーマ名にデータベース表名を加えたものです。オプションのネーム・スペースが、名前の最初に含まれる場合もあります。
2. ビジネス・オブジェクト名を右クリックして、「**開く**」を選択します。 アセンブリー・エディターに、ビジネス・オブジェクトが表示され、それぞれの列に関するフィールドが表示されます。
3. アセンブリー・エディターで、他の列と突き合わせるために設定したい列を選択します。
4. 「プロパティー」ビューで、「アプリケーション情報」を選択します。 「プロパティー」ビューが表示されていない場合は、列の名前を右クリックして、「**プロパティーに表示 (Show in Properties)**」をクリックします。
5. 「**JDBC ASI スキーマ (JDBC ASI schema)**」を展開し、「**JDBCAttributeTypeMetadata**」を展開します。
6. 「**JDBCAttributeTypeMetadata**」を右クリックして、「**新規**」 → 「**jdbcasi:CopyAttribute**」を選択します。
7. 「**CopyAttribute**」プロパティーを選択します。
8. 「拡張子の詳細 (Extension Details)」領域で、コピーする情報を含む列の名前に、テキスト値を設定します。列は、現行ビジネス・オブジェクトに含めること

も、親のビジネス・オブジェクトに含めることもできます。現行ビジネス・オブジェクトの列からコピーするには、値を列の名前に設定します (phoneid など)。親ビジネス・オブジェクトの列からコピーするには、列の名前の接頭部に 2 つのピリオド (..) を使用します (..phone など)。以下の図は、CopyAttribute プロパティーが現行テーブルの列に設定されているアセンブリー・エディターを表しています。



結果

CopyAttribute プロパティーを使用して、別の列の情報に基づいてデータベース列のビジネス・オブジェクト属性およびプロパティーを設定するように、ビジネス・オブジェクトは構成済みです。

次のタスク

これで、モジュールをテストおよびデプロイできます。

Inbound 処理のモジュールの構成

アダプターを Inbound 処理に使用するようにモジュールを構成するには、WebSphere Integration Developer 内で 外部サービス・ウィザードを使用して、 データベースからビジネス・オブジェクトおよびサービスを検出して選択し、ビジネス・オブジェクト定義および関連する成果物を生成します。

データベース・オブジェクトのディスカバリー

接続プロパティーを構成した後は、データベース・オブジェクトを検索するクエリーを実行します。ディスカバーされたオブジェクトのツリーを参照して、データベース内のオブジェクトの構造を理解します。また、フィルターを使用して、参照したいデータベース・オブジェクトのみを表示します。

始める前に

データベースのアクセスに必要なプログラムのデータ要件を知っていなければなりません。例えば、データベースに関する以下の情報が必要になります。

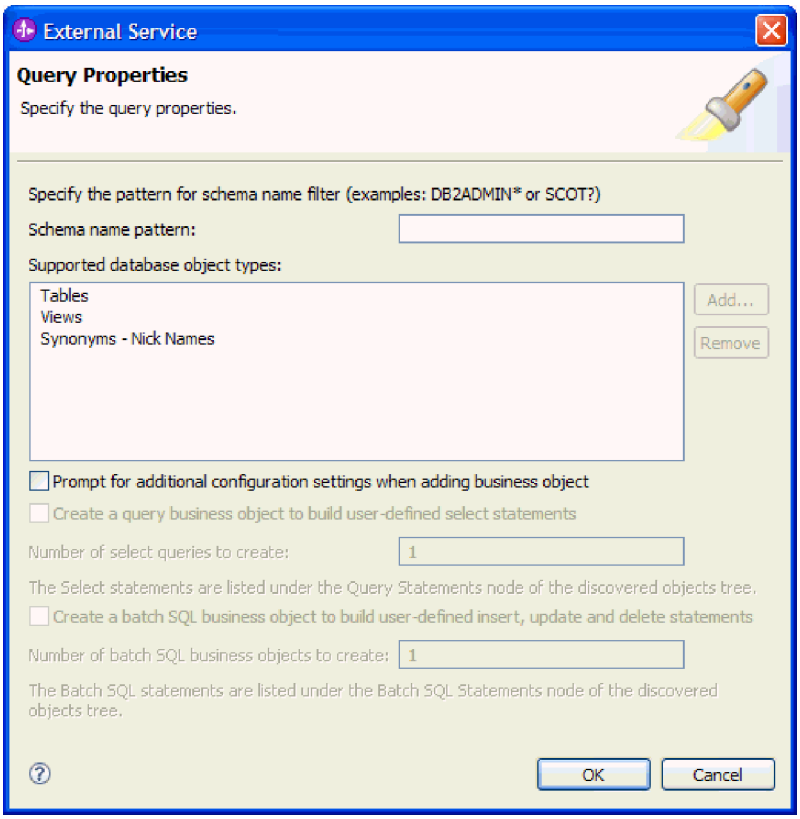
- モジュールはどのスキーマにアクセスする必要があるか
- これらのスキーマ内のどのタイプのデータベース・オブジェクトにアクセスする必要があるか

このタスクを実行する理由および時期

このタスクは、外部サービス・ウィザードのオブジェクトのディスカバリーと選択 (Object Discovery and Selection)で開始します。

このタスクの手順

1. オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウで、「**照会の編集 (Edit Query)**」をクリックします。 照会プロパティー (Query Properties)ウィンドウが表示されます。



無効なオプション「**ユーザー定義の select ステートメントを作成するためのクエリー・ビジネス・オブジェクトを作成する**」および「**ユーザー定義の insert、update、および delete ステートメントを作成するためのバッチ SQL ビジネス・オブジェクトを作成する**」が表示される場合があります。これらのオプションは、Outbound 処理でのみ有効です。

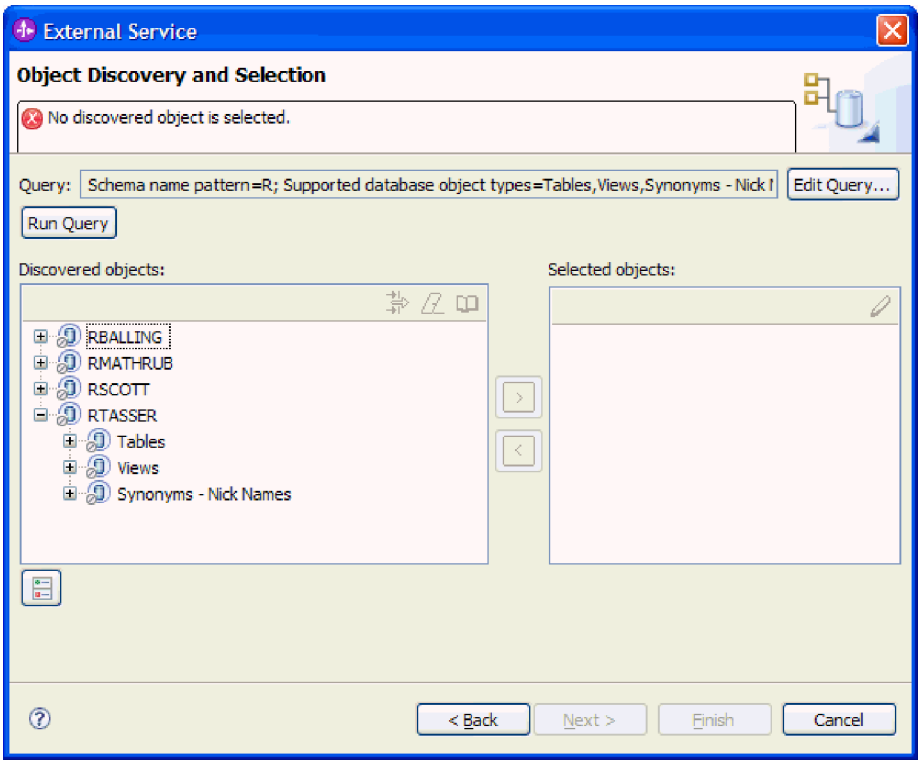
照会プロパティー (Query Properties)ウィンドウを使用して、次のタスクを実行します。

- データベース・スキーマのサブセットを検索することにより、検索時間を削減します。

- 1 つ以上のタイプのデータベース・オブジェクトを検索から除外します。
- データベース内の情報からは自動的に判別できないアプリケーション固有情報の入力をウィザードがユーザーに求めるようにします。

2. 取得されるデータベース・スキーマの数を制限するには、「**スキーマ名パターン**」にスキーマの名前またはスキーマ名パターンを入力します。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。 クエリーを実行すると、そのストリングで始まるスキーマか、またはスキーマ名パターンに一致するスキーマのみが表示されます。スキーマ名パターンを指定しない場合は、データベース内のすべてのスキーマが表示されます。データベースに多数のスキーマがある場合は、フィルターを使用してディスカバリー・プロセスを高速化できます。
3. 検索から 1 つ以上のタイプのオブジェクトを除外するには、除外したいオブジェクトのタイプ (テーブル、ビュー、シノニム、またはニックネーム) を「**サポートされるデータベース・オブジェクト・タイプ**」で選択して、「**除去 (Remove)**」をクリックします。元に戻したい場合は、「**追加**」をクリックして、オブジェクト・タイプをもう一度追加します。 アクセスする必要のないオブジェクト・タイプがデータベースに含まれている場合は、それらを除外すると、ディスカバリー・プロセスを高速化できます。
4. 「**ビジネス・オブジェクトの追加時に追加構成設定のプロンプトを出す**」を選択します。 これにより、作成するビジネス・オブジェクトのリストにデータベース・オブジェクトを追加するたびに、そのオブジェクトに関してユーザーが構成可能なすべてのアプリケーション固有情報を入力するよう求められます。例えば、このオプションを選択した場合は、ビジネス・オブジェクトの単純親子階層を作成するプロセスが順を追って案内されます。テーブル・ビジネス・オブジェクトが、2 つの異なるテーブル (つまり、2 つの親ビジネス・オブジェクトを持つ) の属性を参照する 2 つの属性を持っている階層が必要な場合は、WebSphere Integration Developer から起動されるツールであるアセンブリー・エディターで構成を完了する必要があります。

   重要: このオプションを選択しない場合、ウィザードは必須情報のみを入力するようプロンプトを出します。アセンブリー・エディターを使用して、ビジネス・オブジェクトの構成を完成させてください。
5. 「**OK**」をクリックして、クエリーへの変更を保存します。
6. 「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウで、「**クエリーの実行 (Run Query)**」をクリックします。これによりこのクエリーを使用してデータベース・オブジェクトがディスカバーされます。 標準的なクエリーの実行結果を次の図に示します。



「**ディスカバーされたオブジェクト (Discovered objects)**」ペインには、ディスカバーされたオブジェクトがリストされます。テーブル、ビュー、シノニム/ニックネームは、スキーマ名によってソートされます。

7. 「**ディスカバーされたオブジェクト (Discovered objects)**」リストで、スキーマ・ノードおよびその下の「**テーブル**」、「**ビュー (Views)**」、「**シノニム - ニックネーム (Synonyms - Nicknames)**」の各ノードを展開するには、「**+**」 (正符号) をクリックします。これにより、ウィザードによってディスカバーされたデータベース・オブジェクトが表示されます。

結果

アダプターを使用してアクセスできるデータベース・オブジェクトがウィザードによってディスカバーされました。

次のタスク

外部サービス・ウィザードでの作業を続行します。次のステップでは、モジュールで使用するオブジェクトを選択し、各ビジネス・オブジェクトを構成して、ビジネス・オブジェクトの階層を作成します。

ビジネス・オブジェクトの選択および構成

外部サービス・ウィザードで検出されたデータベース・オブジェクトのリストと、指定した照会オブジェクト・テンプレートおよびバッチ SQL オブジェクト・テンプレートを使用することにより、引き続きこのウィザードを使用して、モジュールでアクセスする必要のあるデータベース・オブジェクトを選択します。次に、新規ビジネス・オブジェクトの構成情報を入力します。

このタスクを実行する理由および時期

オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウでは、オブジェクトを任意の順序で選択して構成できます。ただし、子テーブルを選択して構成するには、その前に親テーブルを選択して構成する必要があります。この制限を除けば、オブジェクトを個々に追加することも、複数のオブジェクトを一度に追加することも柔軟に行えます。「**ディスカバーされたオブジェクト (Discovered objects)**」リストのさまざまなノード内のオブジェクトを組み合わせることができます。例えば、テーブルおよびビュー・オブジェクト数個とストアード・プロシージャー・オブジェクトを選択して、それらを同時に追加できます。

ビジネス・オブジェクトの選択および構成のおおまかな流れは以下のとおりです。

1. 「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウの「**検出オブジェクト (Discovered objects)**」リストで 1 つ以上のオブジェクトを選択します。
2. 「**>**」 (追加) ボタンをクリックします。
3. 「構成プロパティー (Configuration Properties)」ウィンドウが開きます。
   - 1 つのオブジェクトを選択すると、「構成プロパティー (Configuration Properties)」ウィンドウが 1 つ表示されます。

     このウィンドウで、ユーザーが構成可能な属性や、ウィザードによるデータベース検査では検出できないその他の情報をすべて入力したら、「**OK**」をクリックして構成を保存します。
   - 複数のオブジェクトを選択すると、「構成プロパティー (Configuration Properties)」ノートブックが表示されます。選択したオブジェクトごとに 1 ページずつ表示されます。

     各オブジェクトの名前を順にクリックします。ノートブックには、該当するオブジェクトを個別に選択した場合と同じ情報が表示されます。

     重要: すべてのオブジェクトの構成ページで操作が完了するまでは、ノートブックの「**OK**」はクリックしないでください。ウィザードは、必須フィールドすべてに入力されるまではノートブックを閉じませんが、オプション・フィールドに入力しなくてもノートブックを閉じることができます。ウィザードでオプション・フィールドを構成しない場合は、ビジネス・オブジェクト・エディターを使用して、ウィザードを終了した後にそれらのフィールドを構成する必要があります。
4. ウィザードにより、構成されたオブジェクトが「**選択済みオブジェクト (Selected objects)**」リストに追加されます。

ウィザードを終了しない限り、操作を繰り返してモジュールに必要なビジネス・オブジェクトを選択および構成できます。ただし、ウィザードを使用して既存のモジュールにオブジェクトを追加することはできないので、ウィザードを使用し始める前に、ビジネス・オブジェクトを使用するプログラムの要件を明確に理解してください。

テーブル、ビュー、およびシノニムまたはニックネームの選択および構成

モジュールで使用するテーブル、ビュー、およびシノニムまたはニックネームのビジネス・オブジェクトを選択および構成します。Inbound 処理の場合、これらはイベントで送達されるビジネス・オブジェクトです。

始める前に

テーブル、ビュー、およびシノニムまたはニックネームのビジネス・オブジェクトを選択して構成するには、データベース内のデータの構造や、モジュールがどのデータベース・オブジェクトにアクセスする必要があるかを理解しなければなりません。これには、以下のタイプの情報が該当します。

- テーブル、ビュー、およびシノニムまたはニックネームの構造 (列や、データ型などの列属性を含む)。
- テーブル間の関係 (親子関係のカーディナリティーおよび所有権を含む)。

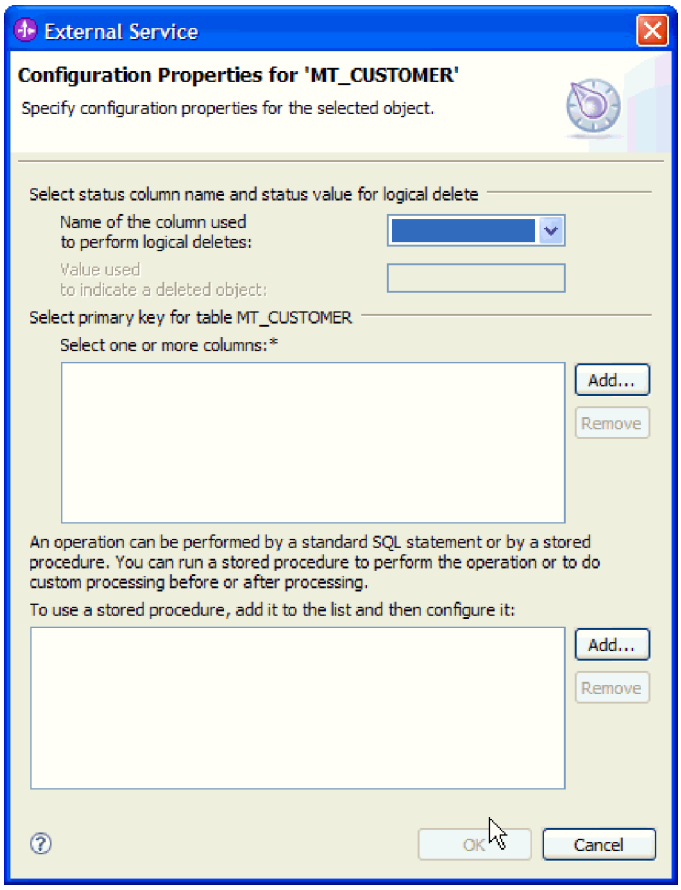
このタスクを実行する理由および時期

このタスクは、外部サービス・ウィザードを使用して実行されます。「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウから操作を開始し、「構成プロパティー (Configuration Properties)」ウィンドウ (構成するビジネス・オブジェクト固有のウィンドウ) で作業します。

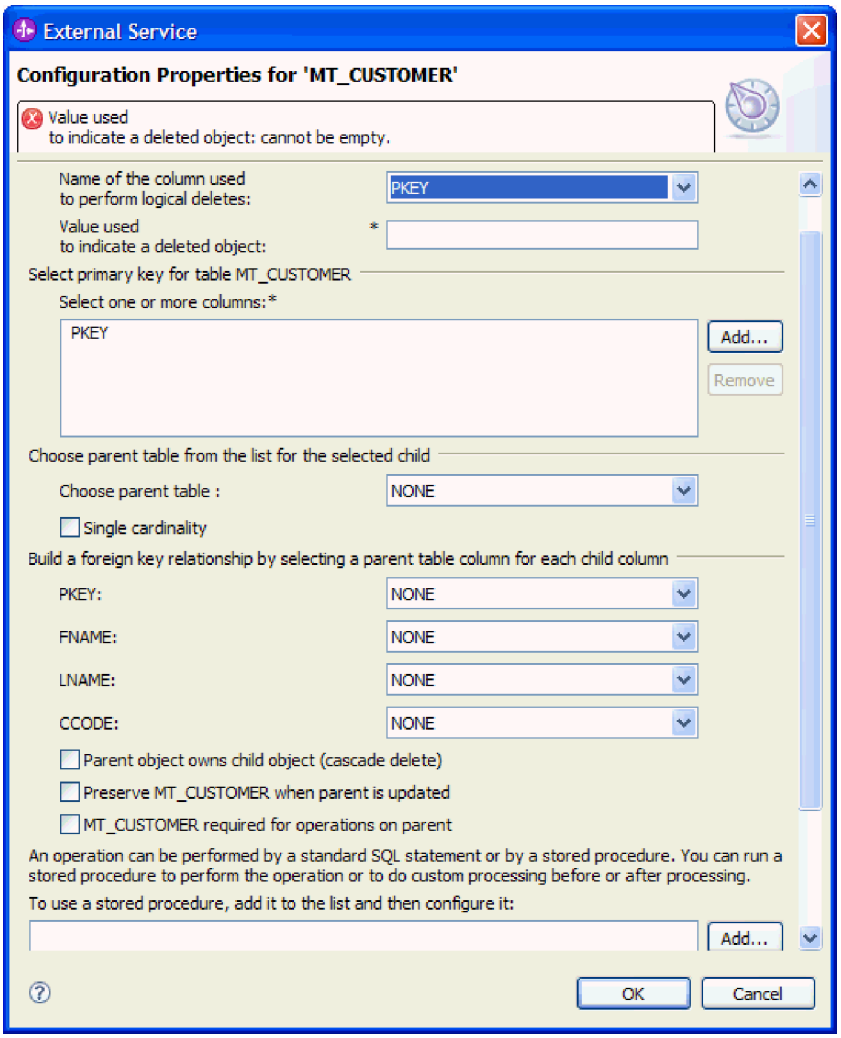
このタスクの手順

1. 「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウの「**検出オブジェクト (Discovered objects)**」リストで、テーブル、ビュー、またはシノニムを 1 つ以上選択し、「**>**」 (追加) ボタンをクリックします。オブジェクトが「**選択されたオブジェクト (Selected objects)**」リストに追加されます。

   以下の 2 つの図に、ビジネス・オブジェクト (テーブル、ビュー、シノニム、またはニックネーム) の標準的な「構成プロパティー (Configuration Properties)」ウィンドウを示します。以下の図に、選択する最初のテーブルまたはテーブル・グループの標準的なウィンドウを示します。



以下の図に、後続のテーブルの標準的なウィンドウを示します。少なくとも 1 つのテーブルを選択して構成した後は、後続テーブルの「構成プロパティー (Configuration Properties)」ウィンドウに、テーブル間の親子階層をオプションで定義することができる領域が表示されます。



オブジェクトの構成時には、拡張構成を必要とする選択を行うと、このウィンドウに追加のフィールドが表示されます。これにより、ウィンドウがスクロールすることがあります。必ずウィンドウのすべてのフィールドを確認してから、「**OK**」をクリックしてください。

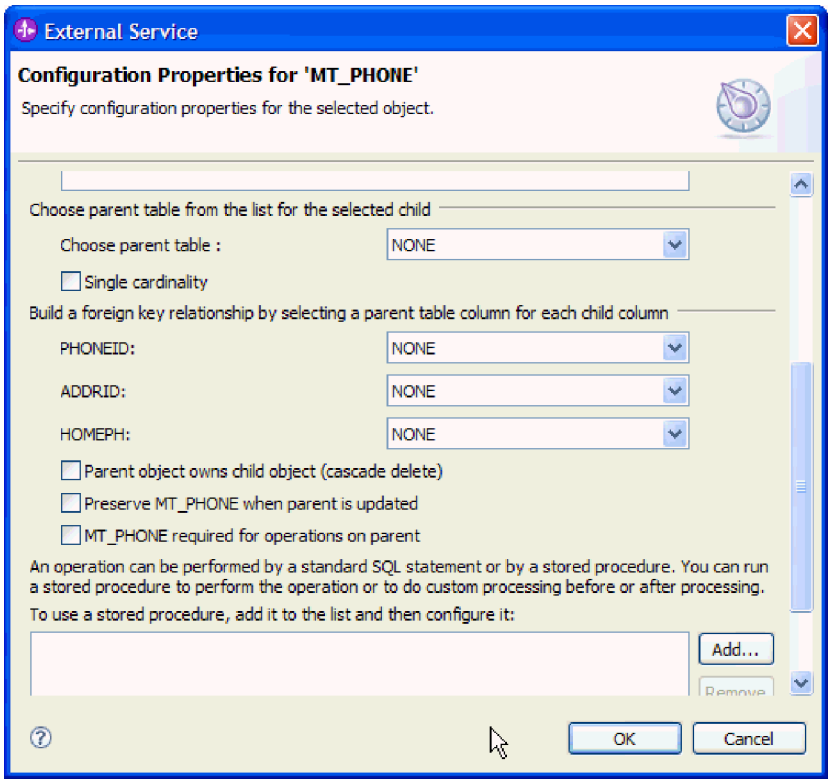
2. 論理削除を示すのに使用される列がテーブルにある場合は、次の手順に従います。
   a. 「**論理削除を実行するのに使用される列の名前**」で列名を選択します。
   b. 「**削除されたオブジェクトを示すために使用する値**」に、行が論理的に削除されていることを示す値を入力します。 この値については、データベース管理者に確認できます。
3. 「**テーブル *table_name* の基本キーの選択**」エリアは、データベース表に基本キーとして指定された列が存在しない場合にのみ表示されます。各テーブル・ビジネス・オブジェクトには、関連付けられたデータベース表にキーがない場合でも、基本キーが定義されている必要があります。データベースで基本キーが定義されている場合、ウィンドウのこのセクションは表示されません。

「**テーブル *table_name* の基本キーの選択**」で「**追加**」をクリックし、テーブル・ビジネス・オブジェクトの基本キーとして使用する列を選択して、「**OK**」をクリックします。テーブルに複合キーがある場合は、複数の列を選択できます。

4. オプションで、ビジネス・オブジェクト間の親子関係を定義します。

   重要: 親子階層を作成するには、親テーブルを最初に構成してから「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウに戻り、子テーブルを選択して構成します。

   以下の図に示す「構成プロパティー (Configuration Properties)」ウィンドウの領域を使用して、親子関係を構成します。これらのフィールドは、構成する最初のテーブルの場合には表示されません。



   a. 「**親テーブルの選択**」で、親テーブルとして使用するテーブルの名前を選択します。 リストに親テーブルが表示されていない場合は、親テーブルがまだ構成されていません。子オブジェクトを構成する前に、戻って親オブジェクトを構成してください。

   b. 関係のカーディナリティーを指定します。

      - テーブルとその親テーブルの間に単一カーディナリティー関係がある場合は、「**単一カーディナリティー**」を選択します。単一カーディナリティー関係では、親はこのタイプの子ビジネス・オブジェクトを 1 つのみ持つことができます。単一カーディナリティー関係は、所有関係を伴って実際の子を表すか、または所有関係を伴わずにルックアップ・テーブルまたはデータベース内の他の対等オブジェクトを表すために使用できます。

- テーブルが複数カーディナリティー関係の場合は、「**単一カーディナリティー**」を選択しないでください。複数カーディナリティー関係では、親がこのタイプの子ビジネス・オブジェクトの配列を持つことができます。

c. 親と子の間に外部キー関係を作成するため、子の列ごとに、親テーブルの外部キーであるかどうかを指定します。
   - 子の列が外部キーでない場合は、「なし」を選択します。
   - 子の列が外部キーの場合は、その子の列に対応する親テーブルの列を選択します。

     注: ウィザードは、1 つの親テーブルのみを構成できます。子テーブルに複数の親テーブルがある場合は、ビジネス・オブジェクト・エディターを使用して、ウィザードを終了した後に残りの親テーブルを構成する必要があります。

d. 親オブジェクトが子オブジェクトを所有している場合、データベース内の子オブジェクトは親が削除されるときに削除されます。子が親により所有されていることを示すには、「**親オブジェクトが子オブジェクトを所有する (カスケード削除)**」を選択します。あるいは、このオプションをクリアして、ルックアップ・テーブルなどの子オブジェクトが、親の削除時に削除されないようにします。

e. Update 操作の一環として子オブジェクトが削除されることのないようにするには、「**親の更新時に *child_table_name* を保持する**」を選択します。

   親テーブルが更新されると、アダプターは入力に存在する子ビジネス・オブジェクトを、データベースから返される子ビジネス・オブジェクトと比較します。デフォルトでは、アダプターは、入力ビジネス・オブジェクト内に存在しない、データベースから返されたすべての子オブジェクトを削除します。

f. デフォルトでは、子ビジネス・オブジェクトを指定せずに、親ビジネス・オブジェクトに対して操作を実行できます。親ビジネス・オブジェクトが変更対象として実行依頼されるときには、その親が子ビジネス・オブジェクトを必ず指定するように要求する場合は、「***Child_table_name* は、親に対する操作で必須**」を選択します。

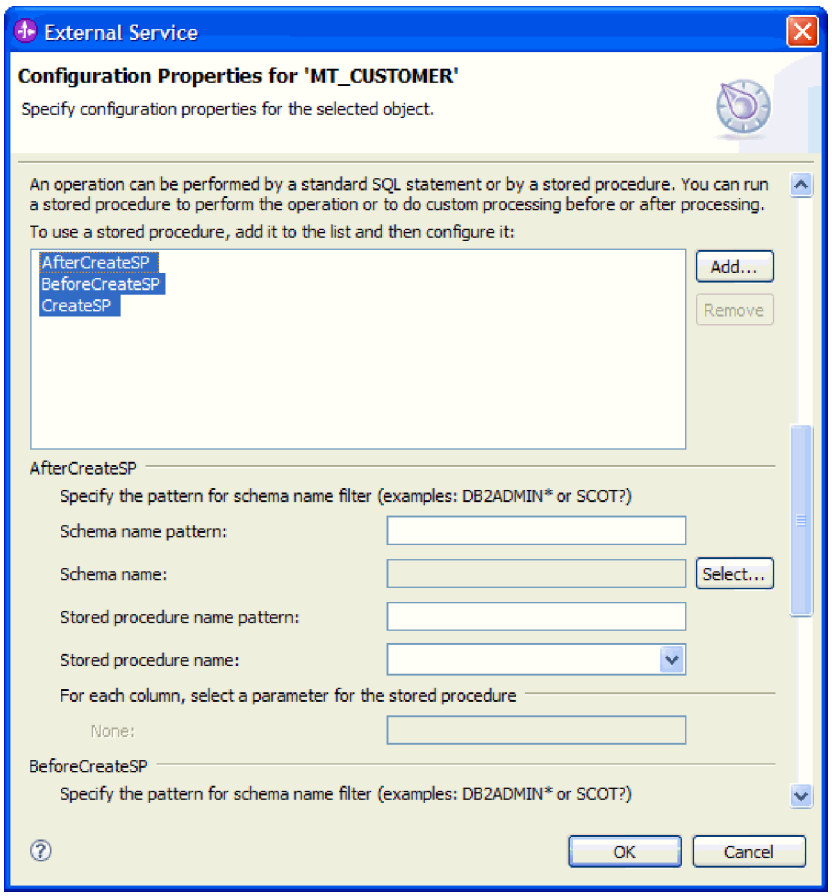
5. 操作を実行するには、アダプターによって生成される標準 SQL ステートメントを使用するか、あるいはデータベース内のストアード・プロシージャーまたはストアード関数を使用します。ストアード・プロシージャーまたはストアード関数を使用する場合は、以下の手順を実行します。

   a. 「**追加**」をクリックします。

   b. 追加 (Add)ウィンドウで、実行するストアード・プロシージャーのタイプを選択します。 操作ごとに、その操作を実行するストアード・プロシージャーと、操作の前後に実行するストアード・プロシージャーを選択できます。例えば、Create 操作の場合は、CreateSP、BeforeCreateSP、および AfterCreateSP のいずれかを指定できます。

   c. 「**OK**」をクリックします。選択したストアード・プロシージャーのタイプが「構成プロパティー (Configuration Properties)」ウィンドウに表示されます。このウィンドウは、各ストアード・プロシージャーの構成用の領域を表示するために拡張されます。新しい領域を表示するには、スクロールダウンする

必要がある場合もあります。



重要: 階層ビジネス・オブジェクトにおいて、その階層のビジネス・オブジェクトごとにストアード・プロシージャーを実行する場合は、ストアード・プロシージャーを、最上位ビジネス・オブジェクトと、ビジネス・オブジェクトの各子ビジネス・オブジェクトまたは配列に別々に関連付ける必要があります。ストアード・プロシージャーを最上位ビジネス・オブジェクトに関連付けても、各子ビジネス・オブジェクトに関連付けないと、その最上位ビジネス・オブジェクトはストアード・プロシージャーで処理されますが、子ビジネス・オブジェクトは標準 SQL 照会を使用して処理されます。

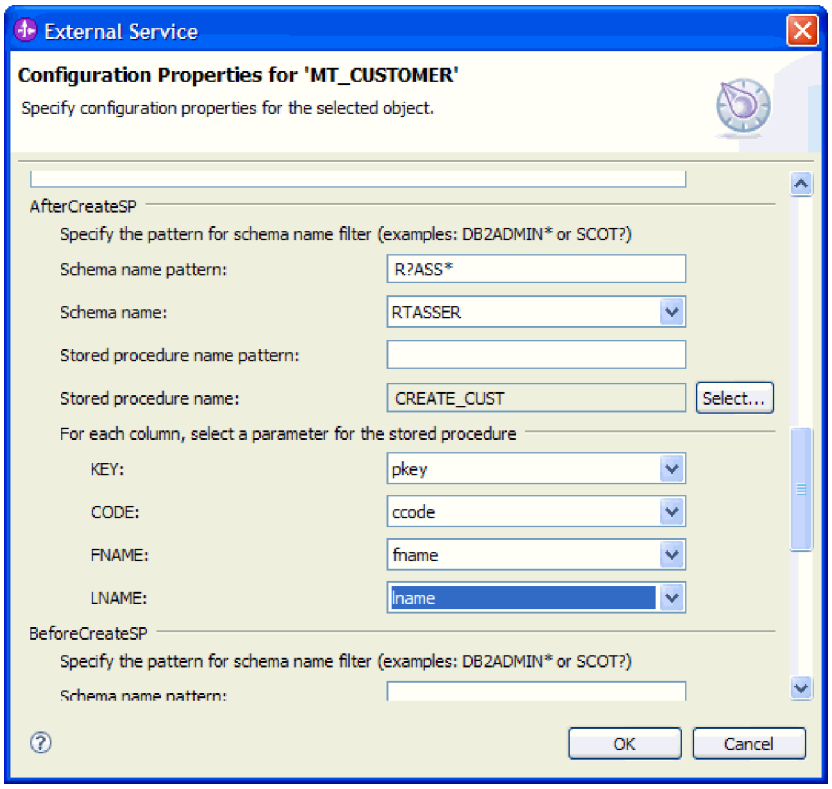
6. 選択したストアード・プロシージャー・タイプごとに、データベース内でのストアード・プロシージャーの名前を指定し、ビジネス・オブジェクトを構成します。
   a. ストアード・プロシージャーを含むスキーマの名前を指定します。 名前は 2 つの方法で指定できます。
      - データベースで検索するデータベース・スキーマをフィルタリングする。
        1) 「**スキーマ名パターン**」で、スキーマの名前またはスキーマ名パターンを入力します。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。名前は大文字小文字が区別されます。
        2) 「**スキーマ名**」で、目的のスキーマの名前を選択します。

- データベースでディスカバーされるすべてのスキーマのリストからスキーマを選択する。
  1) 「**スキーマ名パターン**」フィールドを空のままにします。
  2) 「**スキーマ名**」の横の「**選択 (Select)**」をクリックして、選択 (Select) ウィンドウを開きます。
  3) 選択 (Select)ウィンドウで、目的のスキーマの名前を選択します。オプションで、「**値**」に大/小文字を区別してパターンを入力し、リストを絞り込みます。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。
  4) 目的のスキーマがリストされている場合は、それを選択して「**OK**」をクリックします。

b. ストアード・プロシージャーまたはストアード関数の名前を指定します。 名前は 2 つの方法で指定できます。
   - データベースで検索するストアード・プロシージャーまたはストアード関数をフィルタリングする。
     1) 「**ストアード・プロシージャー名パターン**」で、ストアード・プロシージャーまたはストアード関数の名前を入力するか、または名前パターンを入力します。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。名前は大文字小文字が区別されます。
     2) 「**ストアード・プロシージャー名**」で、目的のプロシージャーの名前を選択します。
   - データベースでディスカバーされるすべてのストアード・プロシージャーのリストから名前を選択する。
     1) 「**ストアード・プロシージャー名パターン**」フィールドを空のままにします。
     2) 「**ストアード・プロシージャー名**」の横で、「**選択**」をクリックして「選択 (Select)」ウィンドウを開きます。
     3) 「選択 (Select)」ウィンドウで、目的のストアード・プロシージャーまたはストアード関数の名前を選択します。オプションで、「**値**」に大/小文字を区別してパターンを入力し、リストを絞り込みます。1 つの文字と突き合わせる場合は疑問符 (?) 文字を使用し、複数の文字と突き合わせる場合はアスタリスク (*) を使用します。
     4) 目的のストアード・プロシージャーまたはストアード関数がリストにある場合は、それを選択して「**OK**」をクリックします。

   構成プロパティー (Configuration Properties)ウィンドウが拡張して、ストアード・プロシージャーを構成するための領域が表示されます。ウィザードは、データベース内のストアード・プロシージャーを調べることにより、パラメーターのリストを自動生成します。

c. ストアード・プロシージャーのパラメーターごと (左側) に、そのパラメーターでストアード・プロシージャーに渡すテーブル列 (右側) を選択します。次の図は、ストアード・プロシージャーを構成した後のウィンドウの一部分を示します。



7. ウィンドウのすべてのフィールドの操作が完了したら、「**OK**」をクリックします。ビジネス・オブジェクトの構成が保存されます。 定義したビジネス・オブジェクト (テーブル、ビュー、シノニム、およびニックネーム) が、「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウにリストされます。
8. 「**選択済みオブジェクト**」領域にあるオブジェクトの構成を変更するには、オブジェクト名を選択して、 (編集) アイコンをクリックします。
9. 必要なすべてのビジネス・オブジェクトを選択および構成したら、「**次へ**」をクリックして、グローバル・プロパティーを設定し、ラッパー・ビジネス・オブジェクトを構成します。

次のタスク

「オブジェクトのディスカバリーと選択 (Object Discovery and Selection)」ウィンドウでの作業を続行して、他のタイプのビジネス・オブジェクトを選択および構成します。

操作のグローバル・プロパティーの設定

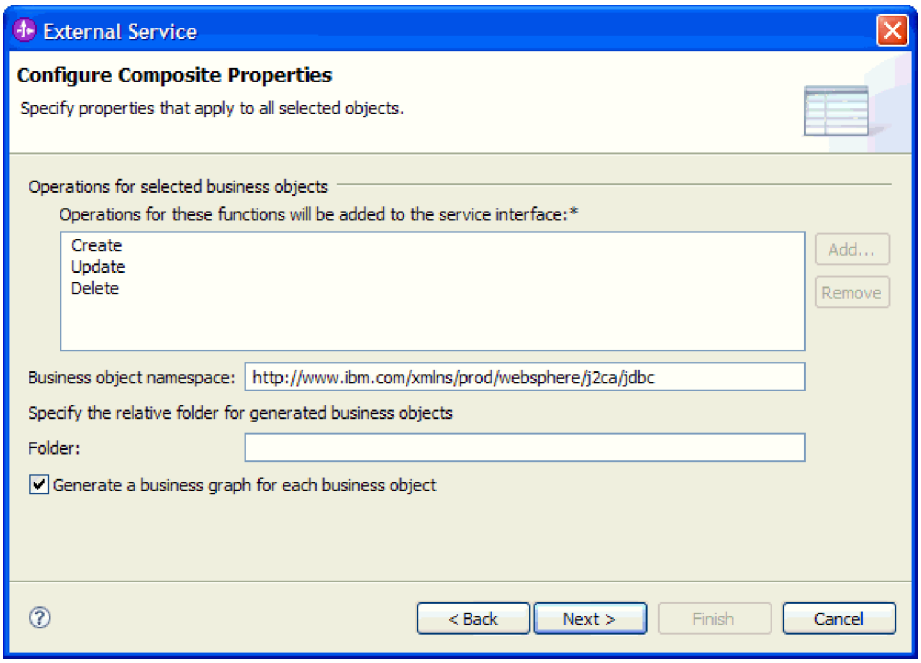
外部サービス・ウィザードでデータベース・オブジェクトを選択した後は、すべてのビジネス・オブジェクトに適用するプロパティーを指定する必要があります。

このタスクの手順

1. オブジェクトのディスカバリーと選択 (Object Discovery and Selection)ウィンドウの「**選択されたオブジェクト (Selected objects)**」リストに、アプリケーションで使用するすべてのビジネス・オブジェクト (wrapper ビジネス・オブジェクトを除く) が含まれている場合は、「**次へ**」をクリックします。

2. 複合プロパティーの構成 (Configure Composite Properties)ウィンドウで、操作リストを確認します。このリストには、アダプターが Inbound サービスでサポートする操作が含まれています。操作リストへの追加には、前のウィンドウで選択したすべてのビジネス・オブジェクトの操作が含まれます。

   指定された操作は、生成されるすべてのビジネス・オブジェクトに対して設定されます。



3. 不要な操作を除去するには、その操作名を選択して「**除去 (Remove)**」をクリックします。 元に戻したい場合は、「**追加**」をクリックして、除外した操作を復元します。
4. 「**ビジネス・オブジェクト Namespace**」で、デフォルトのネーム・スペースを受け入れるか、別のネーム・スペースのフルネームを入力します。

   ネーム・スペースは、ビジネス・オブジェクト・スキーマを論理的に分離するため、ビジネス・オブジェクト名の前に付加されます。このプロパティーについて詳しくは、 240 ページの『ビジネス・オブジェクト Namespace (BusinessObjectNameSpace)』を参照してください。
5. オプションで、生成されたビジネス・オブジェクトが格納されるフォルダーの相対パスを「**フォルダー**」に入力します。
6. ビジネス・オブジェクトごとにビジネス・グラフを作成する場合は、「**ビジネス・オブジェクトごとにビジネス・グラフを生成**」をクリックします。 ビジネス・グラフは、バージョン 6.1.0 より前のバージョンの WebSphere Integration Developer で作成されたモジュールにビジネス・オブジェクトを追加する場合のみ必要になります。

   注: 前のバージョンの WebSphere Integration Developer によって作成されたモジュールにビジネス・オブジェクトを追加する場合は、このオプションを選択する必要があります。それ以外の場合は、インターフェースを再接続する必要があります。

7. 完了したら、「**次へ**」をクリックします。

結果

モジュール内のすべてのビジネス・オブジェクトに適用する情報を設定しました。

次のタスク

ウィザードでの作業を続行します。次のステップでは、実行時に使用するデプロイメント情報、およびサービスをモジュールとして保存するための情報を指定します。

デプロイメント・プロパティーの設定およびサービスの生成

モジュールのビジネス・オブジェクトを選択して構成した後、外部サービス・ウィザードを使用して、アダプターが特定のデータベースに接続するために使用するプロパティーを構成します。ウィザードは、すべての成果物とプロパティー値を保存する、新規のビジネス・インテグレーション・モジュールを作成します。

このタスクを実行する理由および時期

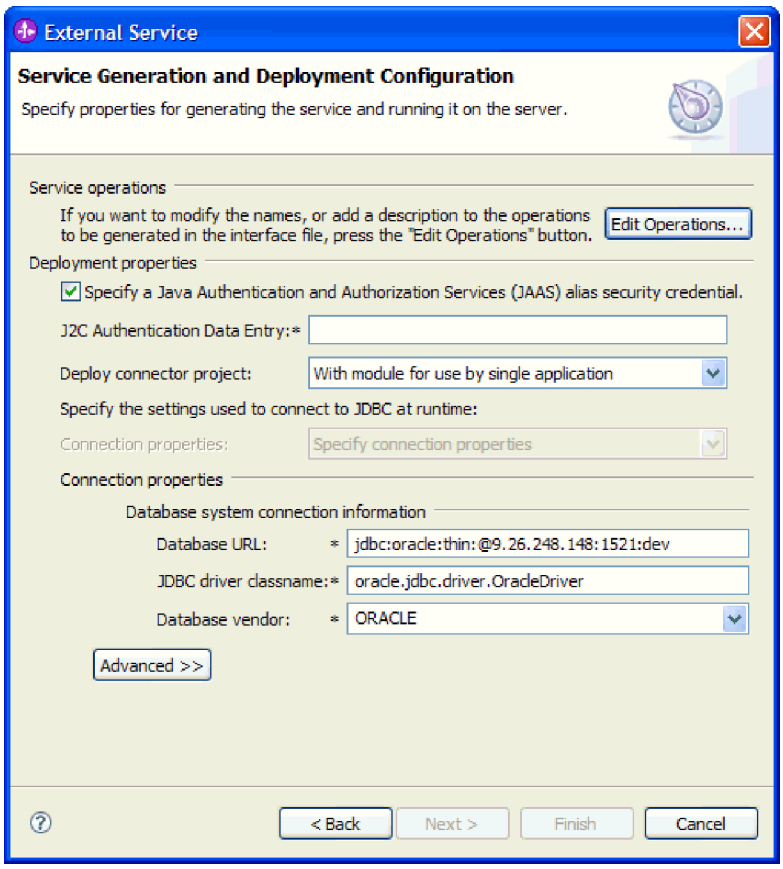
このタスクは、外部サービス・ウィザードのサービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)ウィンドウおよびサービス・ロケーション・プロパティー (Service Location Properties)ウィンドウを使用して実行されます。

このタスクの接続プロパティーは、ウィザードがデータベースに接続するために使用した値に初期化されます。他の値を使用するようにモジュールを構成するには、ここで値を変更します。例えば、実行時に i5/OS で IBM Toolkit for Java ネイティブ・ドライバーを使用するには、ここでドライバー情報を設定します。

このタスクの手順

1. サービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)ウィンドウで、「**操作の編集 (Edit Operations)**」をクリックして、作成するビジネス・オブジェクトの操作の名前を確認するか、操作の説明

を追加します。



2. アダプターが実行時にデータベース・ユーザー名およびパスワードを取得する方法を指定します。
   - J2C 認証別名を使用するには、「**Java 認証・承認サービス(JAAS) の別名のセキュリティー・クレデンシャルの指定 (Specify a Java Authentication and Authorization Services (JAAS) alias security credential)**」を選択し、「**J2C 認証データ入力 (J2C Authentication Data Entry)**」に別名の名前を入力します。

     既存の認証別名を指定することも、(モジュールをデプロイする前に) 認証別名を作成することもできます。名前は大文字小文字が区別されます。また、名前にはノード名が含まれます。
   - サーバー上の既存の Java Naming and Directory Interface (JNDI) データ・ソースに指定されたユーザー名およびパスワードを使用するには、以下のようにします。
     a. 「**Java 認証・承認サービス(JAAS) の別名のセキュリティー・クレデンシャルの指定 (Specify a Java Authentication and Authorization Services (JAAS) alias security credential)**」をクリアします。
     b. 「**拡張**」をクリックします。
     c. 「**拡張接続構成**」を展開します。
     d. 「**DataSource JNDI 名**」に、既存の JNDI データ・ソースの名前を入力します。詳しくは、242 ページの『データ・ソース JNDI 名 (DataSourceJNDIName)』を参照してください。

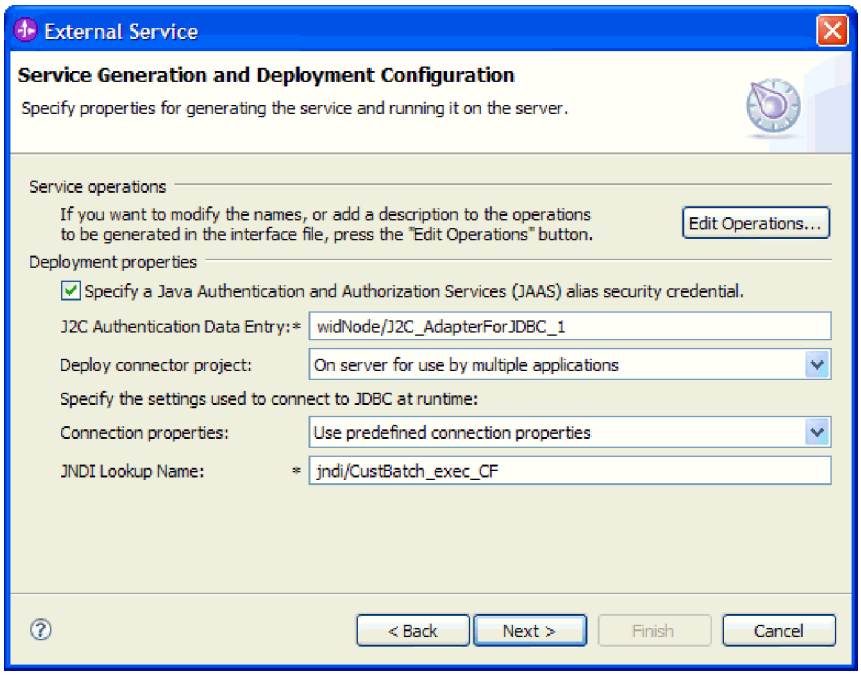
- データベース・ユーザー名およびパスワードをアダプター・プロパティーに保管するよう指定するには、以下のようにします。
  a. 「**Java 認証・承認サービス(JAAS) の別名のセキュリティー・クレデンシャルの指定 (Specify a Java Authentication and Authorization Services (JAAS) alias security credential)**」をクリアします。
  b. 「**拡張**」をクリックします。
  c. 「**データベース・システム接続情報**」の下に、「**ユーザー名**」および「**パスワード**」を入力します。詳しくは、 255 ページの『ユーザー名 (UserName)』および 251 ページの『パスワード (Password)』を参照してください。

  注: パスワードをここで指定すると、そのパスワードはアダプター・プロパティーに平文で保存されるので、非認証ユーザーから見られる可能性があります。

3. 「**コネクター・プロジェクトのデプロイ**」で、アダプター・ファイルをモジュールに組み込むかどうかを指定します。次の値のいずれかを選択してください。
   - 「**単一アプリケーションが使用するモジュールとともにデプロイする**」。アダプター・ファイルをモジュール内に組み込むと、モジュールをすべてのアプリケーション・サーバーにデプロイすることができます。組み込みアダプターを使用するのは、組み込みアダプターを使用するモジュールが 1 つある場合か、複数のモジュールでバージョンの異なるアダプターを実行する必要がある場合です。組み込みアダプターを使用すると、他のモジュールのアダプター・バージョンを変更することで、それらのモジュールを不安定にするリスクを生じることなく、1 つのモジュール内でアダプターをアップグレードできます。
   - 「**複数アプリケーションが使用するサーバー上**」。モジュール内にアダプター・ファイルを組み込まない場合は、このモジュールを実行するアプリケーション・サーバーごとにモジュールをスタンドアロン・アダプターとしてインストールする必要があります。複数のモジュールが同じバージョンのアダプターを使用可能で、アダプターを中央の場所で管理する場合は、スタンドアロン・アダプターを使用します。スタンドアロン・アダプターの場合も、複数のモジュールに対して単一のアダプター・インスタンスを実行することにより、必要なリソースが軽減されます。
4. 前のステップで「**複数アダプターが使用するサーバーにデプロイする (On server for use by multiple adapters)**」を指定した場合は、その接続プロパティーの指定方法を指定します。
   - サーバーで管理接続ファクトリーまたはアクティベーション・スペックを手動で作成および構成した場合、または同じ管理接続ファクトリーまたはアクティベーション・スペックのプロパティーを使用して同じデータベースに接続するアプリケーションをすでにデプロイ済みの場合は、その Java Naming and Directory Interface (JNDI) データ・ソースの名前を指定することによって、管理接続ファクトリーまたはアクティベーション・スペックを再利用できます。
     a. 「**接続プロパティー**」で、「**事前定義された接続プロパティーを使用する**」を選択します。

b. 「**JNDI ルックアップ名**」に、既存の管理接続ファクトリーまたはアクティベーション・スペックの JNDI データ・ソースの名前を入力します。

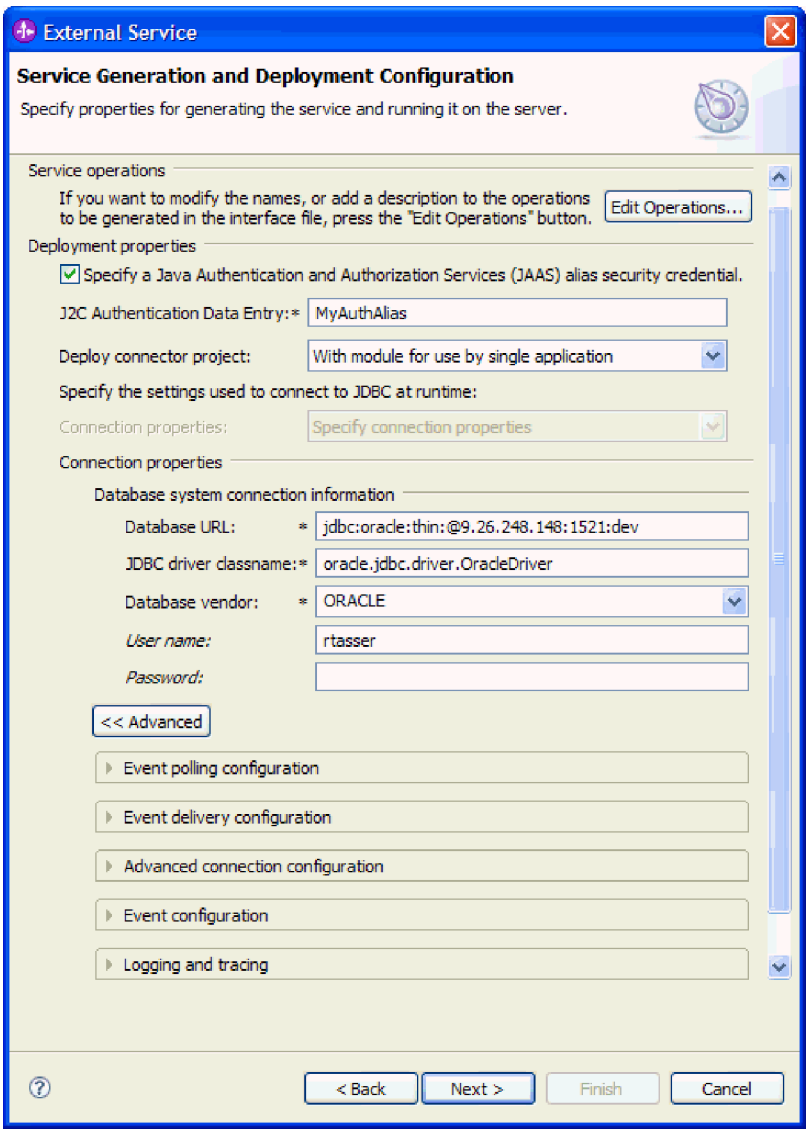
以下の図に、アダプターのスタンドアロン・デプロイメントで管理接続ファクトリーまたはアクティベーション・スペックを再利用する場合の標準的な設定を示します。



c. 「**次へ**」をクリックしてこのタスクを完了します。

- これが、特定のユーザー名とパスワードを使用してデータベースに接続する最初のアプリケーションである場合、または他のアプリケーションとは別々にユーザー名とパスワードを管理する場合は、「**接続プロパティーの指定**」を選択します。

5. 必須の接続プロパティーの値を確認し、必要に応じて変更します。 フィールドは、ウィザードの開始時に指定した接続情報で初期化されます。これらの値を変更して、別のユーザー名およびパスワードを実行時に指定できます。また、別のデータベースに接続できますが、スキーマ名は両方のデータベースで同じでなければなりません。接続プロパティーの形式はデータベース固有です。プロパティーについて詳しくは、 238 ページの『アクティベーション・スペック・プロパティー』を参照してください。
6. オプションで、「**拡張**」をクリックして拡張プロパティーを指定します。 拡張セクションをそれぞれ展開して、プロパティーを確認します。以下の図は、「サービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)」ウィンドウの拡張プロパティー・セクションを示しています。



- **イベント・ポーリング構成**
  a. 「**ポーリング期間の間隔**」に、アダプターがポーリング期間から次の期間まで待機する時間 (ミリ秒) を入力します。詳しくは、 251 ページの『ポーリング期間の間隔 (PollPeriod)』を参照してください。
  b. 「**ポーリング期間内の最大イベント数**」に、各ポーリング期間で送達するイベント数を入力します。詳しくは、 252 ページの『ポーリング期間内の最大イベント数 (PollQuantity)』を参照してください。
  c. 「**接続が失敗した場合の再試行間隔**」に、ポーリング中に接続が失敗してから接続を再試行するまでの待機時間 (ミリ秒) を入力します。詳しくは、 253 ページの『接続が失敗した場合の再試行間隔 (RetryInterval)』を参照してください。
  d. 「**システム接続を再試行する回数**」に、接続を再試行する回数を入力します。再試行がこの回数に達すると、ポーリング・エラーが報告されます。詳しくは、 253 ページの『システム接続を再試行する回数 (RetryLimit)』を参照してください。

e. ポーリング・エラーが発生したらアダプターを停止するようにしたい場合は、「**ポーリング時にエラーが検出された場合はアダプターを停止する**」を選択します。このオプションを選択しない場合、アダプターは例外をログに記録しますが、稼働し続けます。詳しくは、 254 ページの『ポーリング時にエラーが検出された場合はアダプターを停止する (StopPollingOnError)』を参照してください。

- **イベント送達構成**
  a. 「**送達のタイプ**」で、送達方法を選択します。この方式については、246 ページの『送達タイプ (DeliveryType)』で説明します。
  b. イベントの送達は一回のみで、かつ 1 つのエクスポートにのみ送達されるようにする場合は、「**送達は 1 回のみ**」を選択します。このオプションはパフォーマンスを低下させる可能性がありますが、イベント送達が重複したり欠落したりすることはありません。詳しくは、 246 ページの『イベントを一度のみ送達する (AssuredOnceDelivery)』を参照してください。
  c. デフォルトでは、アダプターはポーリング時に検出したすべてのイベントを処理します。現在時刻より後のタイム・スタンプを持つイベントを処理しないようにする場合は、「**将来のタイム・スタンプを持つイベントを処理しない**」を選択します。詳しくは、 246 ページの『将来のタイム・スタンプを持つイベントを処理しない (FilterFutureEvents)』を参照してください。
  d. 「**処理するイベント・タイプ**」に、イベント送達の対象とするビジネス・オブジェクトのリストをコンマで区切って入力します。すべてのビジネス・オブジェクト・タイプのイベントを受信する場合は、このフィールドをブランクのままにします。

     例えば、データベース内で Customer および Order テーブルが変更されたときにのみイベントを受信する (他のテーブルの変更時は受信しない) 場合は、このフィールドに `Customer,Order` を設定します。

     詳しくは、 248 ページの『処理するイベント・タイプ (EventTypeFilter)』を参照してください。
  e. 「**イベントを送達するための接続数**」の下に、イベント送達に使用する接続の最小数および最大数を指定します。詳しくは、 250 ページの『最小接続数 (Minimum connections) (MinimumConnections)』および 250 ページの『最大接続数 (Maximum connections) (MaximumConnections)』を参照してください。
- **拡張接続構成**
  a. 「**DataSource JNDI 名**」については、このトピックですでに説明しています。
  b. 「**追加の JDBC ドライバー接続プロパティー (Additional JDBC driver connection properties)**」を設定します。このプロパティーについて詳しくは、 240 ページの『追加の JDBC ドライバー接続プロパティー (DriverConnectionProperties)』を参照してください。
  c. 「**接続を検証するための SQL クエリー**」を設定します。このプロパティーについて詳しくは、 251 ページの『接続を検証するための SQL 照会 (PingQuery)』を参照してください。

d. 「**クエリー・タイムアウト**」に、アダプターがデータベース・クエリーの応答を待つ時間 (秒) を入力します。このプロパティーについて詳しくは、 252 ページの『クエリー・タイムアウト (QueryTimeOut)』を参照してください。

e. 「**ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す**」を設定します。このプロパティーについて詳しくは、253 ページの『ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)』を参照してください。

- **イベント構成**

a. 「**イベント順序**」で、イベントが取得および処理される順序を指定します。これは、イベント・テーブルの列名と、各列のソート順を制御するキーワードからなるコンマ区切りのリストです。昇順の場合は `asc` を使用し、降順の場合は `desc` を使用します。詳しくは、 247 ページの『イベント順序 (EventOrderBy)』を参照してください。

b. 「**イベント・テーブル名**」で、イベント・ストアを格納するテーブルのデフォルト名を受け入れるか、あるいは別のテーブル名を入力します。詳しくは、 248 ページの『イベント・テーブル名 (EventTableName)』を参照してください。

c. 「**ポーリング前に実行するストアード・プロシージャー**」に、実際のポーリング・クエリーが呼び出される前に実行するストアード・プロシージャーまたはストアード関数の名前を指定します。詳しくは、 255 ページの『ポーリング前に実行するストアード・プロシージャー (SPBeforePoll)』を参照してください。

d. 「**ポーリング後に実行するストアード・プロシージャー**」に、各ポーリング周期の後に実行するストアード・プロシージャーまたはストアード関数の名前を指定します。詳しくは、 254 ページの『ポーリング後に実行するストアード・プロシージャー (SPAfterPoll)』を参照してください。

e. 「**イベント処理用のイベント・クエリー・タイプ**」で、使用するイベント処理のタイプを選択します。

   – アダプターによって提供される標準イベント処理を使用するには、「**標準 (Standard)**」を選択します。

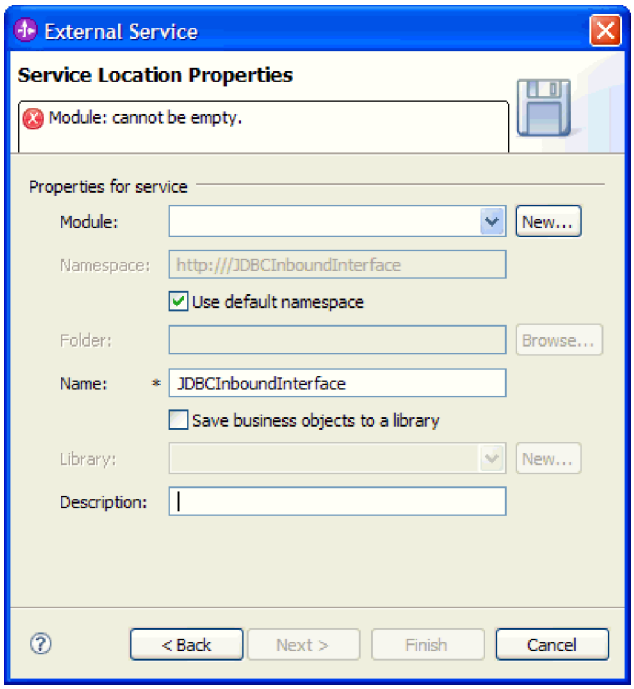
   – ユーザー独自のクエリーを提供してイベント処理をカスタマイズする場合は、「**ユーザー定義 (Dynamic) (User-Defined (Dynamic))**」を選択します。このオプションを選択する場合は、次の表で説明する追加フィールドに値を入力してください。

フィールド	指定内容	詳細情報
カスタム削除照会 (Custom delete query)	各イベントの処理後に実行され、イベントの送達後に削除可能なレコードを削除する照会、ストアード・プロシージャー、またはストアード関数の名前	241 ページの『カスタム削除照会 (Custom delete query) (CustomDeleteQuery)』
カスタム・イベント照会 (Custom event query)	イベントのポーリングを実行する照会、ストアード・プロシージャー、またはストアード関数の名前	241 ページの『カスタム・イベント照会 (Custom event query) (CustomEventQuery)』

フィールド	指定内容	詳細情報
カスタム更新照会 (Custom update query)	各イベントの処理後に実行され、後続のイベント・サイクルで処理対象としてイベントが選択されることを防ぐ照会、ストアード・プロシージャー、またはストアード関数の名前	242 ページの『カスタム更新照会 (Custom update query) (CustomUpdateQuery)』

- **ロギングおよびトレース**
  a. 「**ロギングおよびトレースで使用するアダプター ID**」に、別のアダプター ID を入力します。このプロパティーについて詳しくは、 215 ページの『ロギングおよびトレースで使用するアダプター ID (AdapterID)』を参照してください。

7. 「**次へ**」をクリックします。 サービス・ロケーション・プロパティー (Service Location Properties)ウィンドウが表示されます。



8. サービス・ロケーション・プロパティー (Service Location Properties)ウィンドウで、作成するモジュールの名前を指定します。新規モジュールを指定することも、既存のモジュールを指定することもできます。
   - 目的のモジュール名が「**モジュール (Module)**」リストに表示されている場合は、その名前を選択します。

     重要: モジュールに、現在構成しているオブジェクトと同じ名前のインターフェースまたはビジネス・オブジェクトが含まれている場合、そのモジュールにある元のインターフェースまたはビジネス・オブジェクトは新しいバージョンによって置き換えられます。
   - それ以外の場合は、新規モジュールを作成します。
     a. 「**新規作成**」をクリックします。

b. 統合プロジェクト (Integration Project)ウィンドウで、「**モジュール・プロジェクトの作成 (Create a module project)**」を選択して「**次へ**」をクリックします。

c. モジュールウィンドウで、モジュールの名前を入力します。例えば、JDBCInboundModule です。

d. サービス記述ファイル (.export ファイルおよび .wsdl ファイル) をモジュールのデフォルト・フォルダーに置きたい場合は、「**デフォルト・ロケーションを使用する (Use default location)**」を選択したままにします。モジュールの別のフォルダーを指定する場合は、このオプションをクリアし、「**参照**」をクリックして、「**ロケーション (Location)**」内の別のフォルダーを指定します。

e. ウィザードを閉じたときに WebSphere Integration Developer のアセンブリー・ダイアグラムでこのモジュールが自動的に開くようにする場合は、「**モジュールのアセンブリー・ダイアグラムを開く (Open module assembly diagram)**」を選択します。それ以外の場合は、このオプションをクリアします。

f. 「**終了**」をクリックすると、新規モジュールが作成されます。

9. ビジネス・オブジェクトに使用するネーム・スペースを指定します。
   - モジュールのビジネス・オブジェクトがデフォルトのネーム・スペースを使用するようにする場合は、「**デフォルト Namespace を使用する (Use default namespace)**」を選択したままにします。
   - 別のネーム・スペースを指定するには、このオプションをクリアして、「**Namespace**」に別の値を入力します。
10. オプションで、新規モジュール内のサービス記述を保管するフォルダーを指定します。「**フォルダー**」にフォルダー名を入力するか、既存フォルダーを参照します。 フォルダー名を指定しない場合、成果物 (エクスポート・ファイル、XSD および WSDL ファイル) は、モジュールのルート・フォルダー (すなわちモジュール名のフォルダー) に保管されます。
11. 「**名前**」で、デフォルトのインポート名を受け入れるか、あるいは別の名前を入力します。
12. オプションで、ビジネス・オブジェクトをライブラリーに保管して、それを他のモジュールが使用できるようにする場合は、「**ビジネス・オブジェクトをライブラリーに保管する (Save business objects to a library)**」を選択して、ライブラリーの場所を「**ライブラリー (Library)**」に指定します。
13. オプションで、モジュールについて説明したコメントを「**説明**」に入力します。
14. プロパティーの設定が完了したら、「**終了**」をクリックします。
15. 「変更されたモデル (Model Changed)」ウィンドウが表示されたら、「**はい**」をクリックします。

結果

ウィザードは終了します。モジュールがプロジェクトに作成され、成果物が生成されます。

次のタスク

インスタンスによっては、構成を完了するためにアセンブリー・エディターを使用しなければならない場合があります。完了したら、モジュールをテストまたはデプロイできます。

構成の完了

場合によって、ビジネス・オブジェクトの構成を完了するために、手動による構成ステップが必要になります。

このタスクを実行する理由および時期

ウィザードによって生成された成果物をカスタマイズする必要がある場合に、このタスクを実行してください。以下の状態でこのタスクを行う可能性があります。

- ある列の CopyAttribute パラメーターの値を、他の列と同じ値に設定する場合。
- ビジネス・オブジェクトに対して属性を追加または除去する場合。例えば、参照する必要のないデータベース列に対応した単純な属性を除去することによって、ビジネス・オブジェクト設計を簡素化できます。
- 複数の親を持つテーブル・ビジネス・オブジェクトについて追加の親を構成する場合。ウィザードは、1 つのテーブル・ビジネス・オブジェクトにつき、親を 1 つのみ構成します。

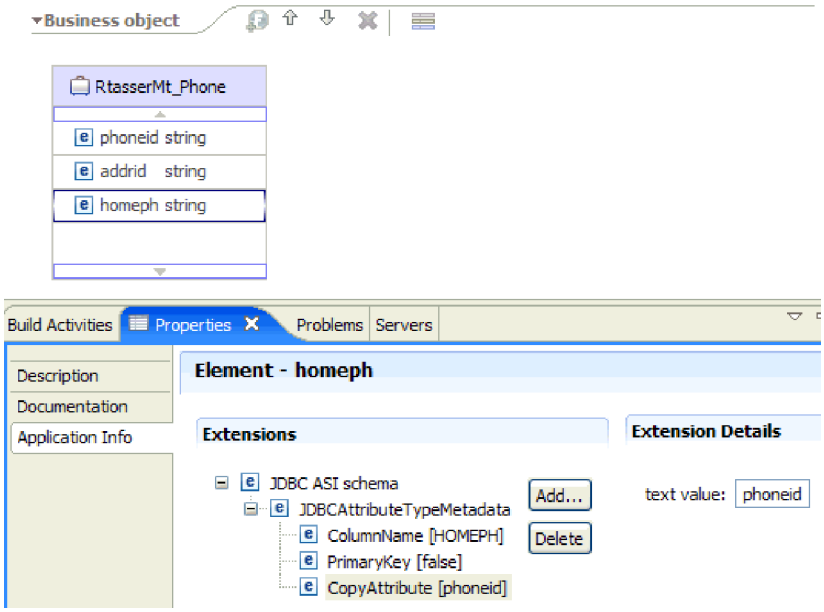
このトピックでは、CopyAttribute パラメーターをテーブル・ビジネス・オブジェクトに設定する、詳細な手順を説明します。ビジネス・オブジェクト構造に対する他の変更は、同じ手法を使用して、実行できます。

CopyAttribute パラメーターは、他の列の値およびアプリケーション固有情報を取り込む対象となる列についての属性のプロパティーに含まれます。例えば、テーブルの新しい行の `contact` 列に、`email` 列と同じ値を組み込むには、`contact` 属性の CopyAttribute パラメーターに `email` を設定します。WebSphere Integration Developer でアセンブリー・エディターを使用して、値を設定します。

このタスクの手順

1. WebSphere Integration Developer の「ビジネス・インテグレーション」パースペクティブで、モジュール名、「**データ・タイプ**」を展開し、テーブル・ビジネス・オブジェクトを見つけます。 ビジネス・オブジェクト名は、データベース・スキーマ名にデータベース表名を加えたものです。オプションのネーム・スペースが、名前の最初に含まれる場合もあります。
2. ビジネス・オブジェクト名を右クリックして、「**開く**」を選択します。 アセンブリー・エディターに、ビジネス・オブジェクトが表示され、それぞれの列に関するフィールドが表示されます。
3. アセンブリー・エディターで、他の列と突き合わせるために設定したい列を選択します。
4. 「プロパティー」ビューで、「アプリケーション情報」を選択します。 「プロパティー」ビューが表示されていない場合は、列の名前を右クリックして、「**プロパティーに表示 (Show in Properties)**」をクリックします。
5. 「**JDBC ASI スキーマ (JDBC ASI schema)**」を展開し、「**JDBCAttributeTypeMetadata**」を展開します。

6. 「**JDBCAttributeTypeMetadata**」を右クリックして、「**新規**」 → 「**jdbcasi:CopyAttribute**」を選択します。
7. 「**CopyAttribute**」プロパティーを選択します。
8. 「拡張子の詳細 (Extension Details)」領域で、コピーする情報を含む列の名前に、テキスト値を設定します。列は、現行ビジネス・オブジェクトに含めることも、親のビジネス・オブジェクトに含めることもできます。現行ビジネス・オブジェクトの列からコピーするには、値を列の名前に設定します (`phoneid` など)。親ビジネス・オブジェクトの列からコピーするには、列の名前の接頭部に 2 つのピリオド (..) を使用します (`..phone` など)。以下の図は、CopyAttribute プロパティーが現行テーブルの列に設定されているアセンブリー・エディターを表しています。



結果

CopyAttribute プロパティーを使用して、別の列の情報に基づいてデータベース列のビジネス・オブジェクト属性およびプロパティーを設定するように、ビジネス・オブジェクトは構成済みです。

次のタスク

これで、モジュールをテストおよびデプロイできます。

第 5 章 アセンブリー・エディターによる対話仕様プロパティーの変更

サービスの生成後にアダプター・モジュールの対話仕様プロパティーを変更するには、WebSphere Integration Developer のアセンブリー・エディターを使用します。

始める前に

外部サービス・ウィザード を使用して、アダプターのサービスを生成済みのはずです。

このタスクを実行する理由および時期

アダプターのサービスを生成後に、対話仕様プロパティーの変更が必要になる場合があります。対話仕様プロパティーはオプションですが、特定のビジネス・オブジェクトの特定の操作に対して、メソッド・レベルで設定されます。指定した値は、外部サービス・ウィザードによって生成されるすべての親ビジネス・オブジェクトのデフォルトとして表示されます。これらのプロパティーは、EAR ファイルをエクスポートする前に変更できます。アプリケーションをデプロイした後にこれらのプロパティーを変更することはできません。

対話仕様プロパティーを変更するには、以下の手順を実行します。

このタスクの手順

1. WebSphere Integration Developer の Business Integration パースペクティブで、モジュール名を展開します。
2. 「**アセンブリー・ダイアグラム**」を展開して、インターフェースをダブルクリックします。
3. アセンブリー・エディターでインターフェースをクリックします。(追加のクリックをしない限り、モジュールのプロパティーが表示されています。)
4. 「**プロパティー**」タブをクリックします。(ダイアグラム内でインターフェースを右クリックし、「**プロパティーを表示**」をクリックすることもできます。)
5. 「**バインディング**」で、「**メソッド・バインディング**」をクリックします。インターフェースのメソッドが、ビジネス・オブジェクトと操作の組み合わせごとに 1 つずつ表示されます。
6. 変更する対話仕様プロパティーを持つメソッドを選択します。
7. 「**汎用**」タブでプロパティーを変更します。変更する対話仕様プロパティーを持つメソッドごとにこの手順を繰り返します。

結果

アダプター・モジュールに関連付けられている対話仕様プロパティーが変更されました。

次のタスク

モジュールをデプロイします。

第 6 章 モジュールのデプロイ

モジュールをデプロイし、モジュールおよびアダプターを構成するファイルを、実稼働またはテストのための動作環境に配置します。WebSphere Integration Developer では、統合テスト環境は、インストール時に選択したテスト環境プロファイルに応じて、WebSphere Process Server または WebSphere Enterprise Service Bus、あるいはその両方に対する実行時サポート機能を備えています。

デプロイメント環境

モジュールおよびアダプターのデプロイ先には、テスト環境と実稼働環境があります。

WebSphere Integration Developer では、モジュールをテスト環境内の 1 つ以上のサーバーにデプロイできます。通常は、これがビジネス・インテグレーション・モジュールの実行およびテストを行うための最も一般的な手法です。ただし、WebSphere Process Server 上または WebSphere Enterprise Service Bus 上で管理コンソールまたはコマンド行ツールを使用して、サーバーへのデプロイメント用のモジュールを EAR ファイルとしてエクスポートすることもできます。

テスト用のモジュールのデプロイ

WebSphere Integration Developer では、組み込みアダプターを内蔵するモジュールをテスト環境にデプロイし、サーバー構成の編集、サーバーの始動および停止、モジュール・コードのテストによるエラー有無の確認などの作業を実行できるサーバー・ツールと連携できます。テストは通常、コンポーネントのインターフェース操作について実行されますが、このテストを実行すると、コンポーネントが正しく実装され、参照先が正しく接続されているかどうかを判断できます。

Inbound 処理をテストするためのターゲット・コンポーネントの生成および接続

Inbound 処理用のアダプターが組み込まれているモジュールをテスト環境にデプロイする前に、まずターゲット・コンポーネントを生成して接続する必要があります。このターゲット・コンポーネントは、アダプターがイベントを送信する宛先 として機能します。

始める前に

外部サービス・ウィザードを使用してエクスポート・モジュールを生成してあるはずです。

このタスクを実行する理由および時期

Inbound 処理のためにターゲット・コンポーネントを生成して接続する必要があるのは、テスト環境のみです。実稼働環境でアダプターを配置する際には必要ありません。

ターゲット・コンポーネントは、イベントを受信します。 WebSphere Integration Developer のアセンブリー・エディターを使用して、エクスポート・コンポーネントを (2 つのコンポーネントを接続している) ターゲット・コンポーネントに*接続* します。アダプターはこのワイヤーを使用して、(エクスポート・コンポーネントからターゲット・コンポーネントへ) イベント・データを受け渡します。

このタスクの手順

1. ターゲット・コンポーネントを作成します。
   a. WebSphere Integration Developer の Business Integration パースペクティブで、「**アセンブリー・ダイアグラム**」を展開して、エクスポート・コンポーネントをダブルクリックします。 デフォルト値を変更しなかった場合、エクスポート・コンポーネントの名前は、ご使用のアダプター + **InboundInterface** になります。

      インターフェースにより、呼び出すことができる操作と渡されるデータ (入力引数、戻り値、例外など) が指定されます。 **InboundInterface** コンポーネントには、Inbound 処理をサポートするためにアダプターが必要とする操作が格納されています。また、このコンポーネントは外部サービス・ウィザードを実行すると作成されます。
   b. 「**コンポーネント**」を展開して「**型なしコンポーネント**」を選択し、そのコンポーネントをアセンブリー・ダイアグラムまでドラッグして、新規コンポーネントを作成します。

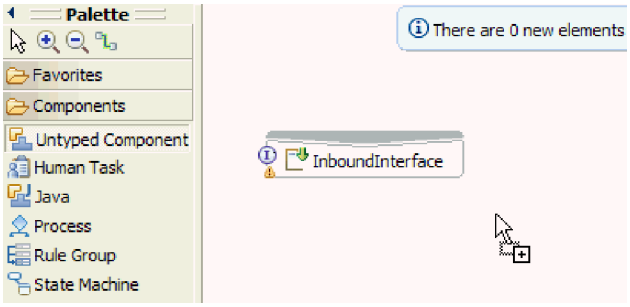


図 19. アセンブリー・ダイアグラムへのコンポーネントの追加

      カーソルが配置アイコンに変わります。
   c. コンポーネントをクリックして、そのコンポーネントをアセンブリー・ダイアグラムに表示します。
2. コンポーネントを接続します。
   a. エクスポート・コンポーネントをクリックして、新規コンポーネントにドラッグします。 これにより、次の図に示すように、エクスポート・コンポーネントから新規コンポーネントへ線を引くことができます。

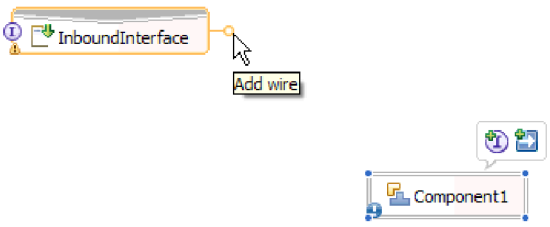


図 20. ワイヤー・アイコンの選択

b. アセンブリー・ダイアグラムを保存します。 「**ファイル**」 → 「**保管**」をクリックします。

3. 新規コンポーネントの実装を生成します。

a. 新規コンポーネントを右クリックして、「**実装の生成**」 → 「**Java**」を選択します。

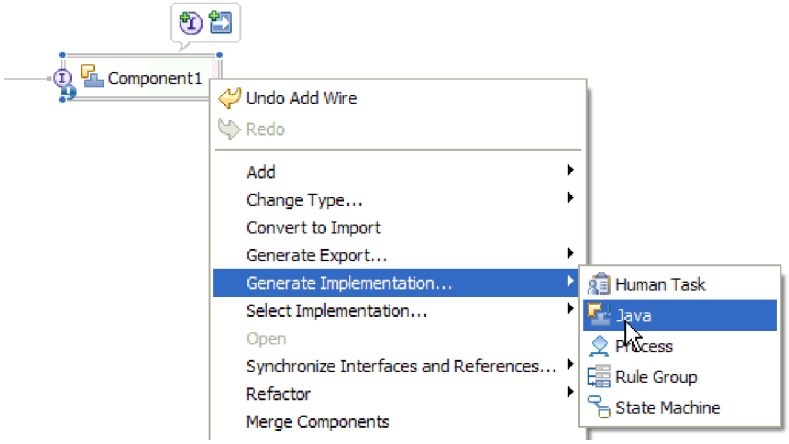


図 21. Java 実装環境の生成

b. 「**(デフォルト・パッケージ)**」を選択して、「**OK**」をクリックします。 これにより、インバウンド・モジュールのエンドポイントが作成されます。

別のタブに Java 実装環境が表示されます。

c. **オプション:** print ステートメントを追加して、エンドポイント・メソッドのそれぞれのエンドポイントで受信したデータ・オブジェクトを出力します。

d. 「**ファイル**」 → 「**保管**」をクリックして、変更内容を保存します。

次のタスク

テストを行うモジュールのデプロイを続行します。

Outbound 操作のテスト準備

WebSphere Integration Developer テスト・クライアントでご使用のモジュールの Outbound 処理をテストする前に、ビジネス・オブジェクトのいくつかを変更しなければならない場合があります。

このタスクを実行する理由および時期

このステップは、WebSphere Integration Developer テスト・クライアントで実行されます。まだ開いていない場合は、プロジェクト名を右クリックして、「**テスト**」 → 「**モジュールのテスト**」をクリックし、ビジネス・インテグレーション・パースペクティブから開きます。

- **クエリー・ビジネス・オブジェクト**

  ご使用のクエリー・ビジネス・オブジェクトが WHERE 節なしで作成されていた場合 (例えば、`Select * from Customer` などの SELECT ステートメントで定義されていた場合)、テスト・クライアントでテストする前に、クエリー・ビジネス・オブジェクトの jdbcwhereclause 属性を設定解除します。
- **テーブル、ビュー、およびシノニムまたはニックネーム・ビジネス・オブジェクト**

  RetrieveAll 操作をテストする前に、テストの一環として設定しない値を持つ、あらゆる属性を設定解除する必要があります。
- **クエリー・ビジネス・オブジェクト**

  RetrieveAll 操作をテストする前に、テストの一環として設定しない値を持つ、あらゆる属性を設定解除する必要があります。

サーバーへのモジュールの追加

WebSphere Integration Developer では、モジュールをテスト環境内の 1 つ以上のサーバーに追加できます。

始める前に

テストしているモジュールがアダプターを使用して Inbound 処理を実行する場合は、アダプターによるイベント送信先となるターゲット・コンポーネント を生成して、そこに接続する必要があります。

このタスクを実行する理由および時期

ご使用のモジュールと、モジュールによるアダプターの使用をテストするため、そのモジュールをサーバーに追加する必要があります。

このタスクの手順

1. *条件:* 「**サーバー・ビュー**」にサーバーがない場合は、以下の手順を実行し、新規サーバーを追加して定義します。
   a. カーソルを「**サーバー・ビュー**」の内側に置き、右クリックして、「**新規**」 → 「**サーバー**」を選択します。
   b. 「新規サーバーの定義」ウィンドウで、サーバー・タイプを選択します。
   c. サーバーの設定値を構成します。
   d. 「**終了**」をクリックして、サーバーを公開します。
2. サーバーにモジュールを追加します。
   a. 「サーバー・ビュー」に切り替えます。 WebSphere Integration Developer で、「**ウィンドウ**」 → 「**ビューの表示**」 → 「**サーバー**」を選択します。

a. サーバーを始動します。 WebSphere Integration Developer 画面の右下のペインにある「サーバー」タブで、サーバーを右クリックし、「**開始**」を選択します。

3. サーバーの状況が「*開始済み*」である場合は、サーバーを右クリックし、「**プロジェクトの追加および除去**」を選択します。
4. 「プロジェクトの追加および除去」画面で、対象のプロジェクトを選択して「**追加**」をクリックします。 プロジェクトは、「**使用可能プロジェクト**」のリストから「**構成プロジェクト**」のリストに移動します。
5. 「**終了**」をクリックします。 これにより、モジュールがサーバーにデプロイされます。

   モジュールがサーバーに追加されている間に、右下のペインの「コンソール」タブに、ログが表示されます。

次のタスク

モジュールおよびアダプターの機能をテストします。

テスト・クライアントを使用した Outbound 処理用モジュールのテスト

Outbound 処理用のアセンブル済みモジュールおよびアダプターを、WebSphere Integration Developer の統合テスト・クライアントを使用してテストします。

始める前に

最初に、モジュールをサーバーに追加する必要があります。

このタスクを実行する理由および時期

モジュールのテストは、通常、コンポーネントのインターフェース操作について実行されますが、このテストを実行すると、コンポーネントが正しく実装され、参照先が正しく接続されているかどうかを判断できます。

このタスクの手順

1. テストするモジュールを選択し、右クリックして、「**テスト**」 → 「**テスト・モジュール**」を選択します。
2. テスト・クライアントを使用したモジュールのテストについて詳しくは、WebSphere Integration Developer インフォメーション・センターの『モジュールおよびコンポーネントのテスト』のトピックを参照してください。

次のタスク

ご使用のモジュールおよびアダプターのテスト結果に納得したら、モジュールおよびアダプターを実稼働環境にデプロイできます。

実稼働のためのモジュールのデプロイ

外部サービス・ウィザード を使用して作成したモジュールを、実稼働環境で WebSphere Process Server または WebSphere Enterprise Service Bus にデプロイする処理は、2 段階構成になっています。最初に、WebSphere Integration Developer 内にモジュールをエンタープライズ・アーカイブ (EAR) ファイルの形でエクスポートします。次に、WebSphere Process Server 管理コンソールを使用して、EAR ファイルをデプロイします。

サーバー上での外部ソフトウェア依存関係の追加

アダプターがデータベースと通信するには、 WebSphere Process Server または WebSphere Enterprise Service Bus サーバーにインストールされた特定のファイルが必要です。

始める前に

データベース が、 WebSphere Process Server または WebSphere Enterprise Service Bus と同じコンピューター・システムにインストール済みの場合、このタスクを実行する必要はありません。アダプターは、既にこのファイルを使用できます。

このタスクを実行する理由および時期

スタンドアロン・アダプターが通信するには、データベース の JDBC ドライバー・ファイルおよびネイティブ・システム・ライブラリーが必要です。組み込みアダプターには、ネイティブ・システム・ライブラリーが必要です。

このタスクの手順

1. データベース管理者またはデータベース・ソフトウェアの Web サイトから、ご使用のデータベース・ソフトウェアおよびオペレーティング・システムの JDBC ドライバー固有ファイルとネイティブ・ライブラリーを入手します。 必要なファイルの種類は、データベース・サーバーによって異なります。次の表に、一般的なデータベース・ソフトウェアで必要とされるファイルのリストを示します。次の表、データベース・サーバー。次の表に、一般的なデータベース・ソフトウェアに必要な JDBC ドライバー・ファイルをリストします。

表 12. 一般的なデータベース・ソフトウェアの JDBC ドライバー・ファイル

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
IBM DB2 Universal Database for Linux、UNIX、および Windows	IBM DB2 Universal (タイプ 4)	db2jcc.jar db2jcc_license_cu.jar	なし
IBM DB2 for z/OS	IBM DB2 Universal (タイプ 4)	db2jcc.jar db2jcc_license_cisuz.jar	なし

表 12. 一般的なデータベース・ソフトウェアの JDBC ドライバー・ファイル (続き)

データベース・ソフトウェア	ドライバー	JDBC ドライバー・ファイル	ネイティブ・システム・ライブラリー
IBM DB2 for i5/OS	IBM Toolbox for Java リモート・ドライバー	jt400.jar	なし
	IBM DB2 Universal ドライバー	db2jcc.jar db2jcc_license_cisuz.jar	なし
	IBM Toolkit for Java ネイティブ・ドライバー*	db2_classes.jar	なし
Oracle	Thin ドライバー	ojdbc14.jar	なし
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	なし
* IBM Toolkit for Java ネイティブ・ドライバーを使用してアダプターの実行時にデータベースに接続できますが、ウィザードの実行中に使用して接続することはできません。ディスカバリー・プロセス中は、IBM Toolbox for Java リモート・ドライバーか、または IBM DB2 Universal ドライバーを使用する必要があります。ただし、実行時にネイティブ・ドライバーを使用するようにモジュールを構成できます。この構成は、「サービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)」ウィンドウで行います。			

2. スタンドアロン・アダプターの場合、JDBC ドライバー・ファイルを WebSphere Process Server または WebSphere Enterprise Service Bus にコピーしてください。
3. タイプ 2 ドライバーを使用するようにモジュールが構成済みの場合は、ネイティブ・システム・ライブラリーをサーバーにコピーします。 このステップは、スタンドアロン・アダプターおよび組み込みアダプターの両方に必要です。

RAR ファイルのインストール (スタンドアロン・アダプターを使用するモジュールの場合のみ)

アダプターをモジュールに組み込まないが、サーバー・インスタンス内にデプロイされているすべてのアプリケーションに対してアダプターを使用可能にする場合は、アダプターを RAR ファイルの形式でアプリケーション・サーバーにインストールすることが必要になります。RAR ファイルとは、Java 2 Connector (J2C) アーキテクチャーに合わせてリソース・アダプターを圧縮するときに使用する Java アーカイブ (JAR) ファイルのことです。

始める前に

外部サービス・ウィザードの「サービス生成およびデプロイメント構成 (Service Generation and Deployment Configuration)」ウィンドウで、「**コネクター・プロジェクトのデプロイ**」を「**複数アダプターが使用するサーバー上 (On server for use by multiple adapters)**」に設定してあるはずです。

このタスクを実行する理由および時期

アダプターを RAR ファイルの形式でインストールすると、そのアダプターは、サーバー・ランタイムで実行されているすべての J2EE アプリケーション・コンポーネントで使用可能になります。

このタスクの手順

1. 管理コンソールを開始します。
2. 「**リソース**」 → 「**リソース・アダプター**」 → 「**リソース・アダプター**」をクリックします。
3. 「リソース・アダプター」ページで、「**RAR のインストール**」をクリックします。

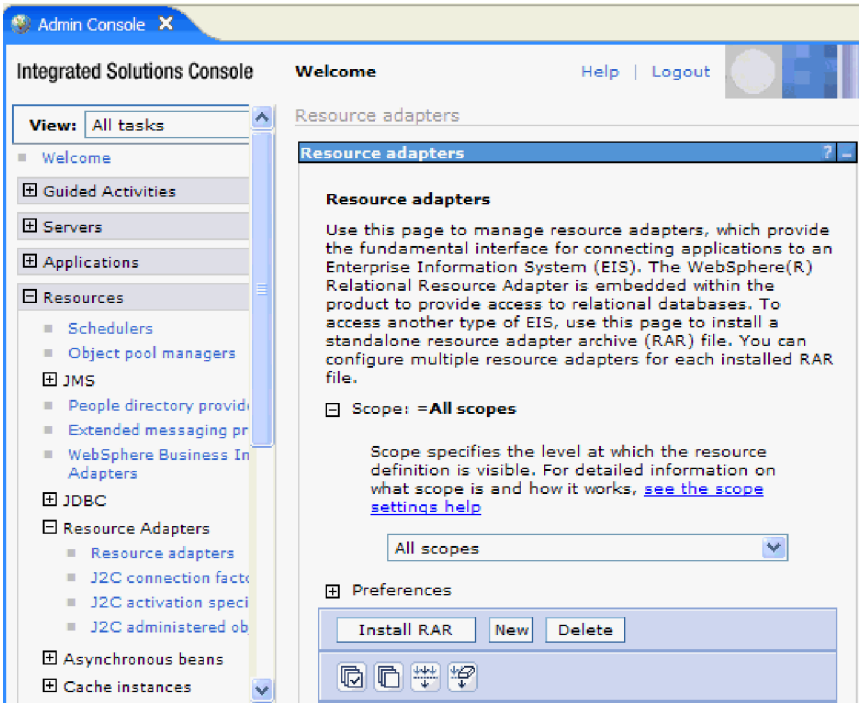


図 22. 「リソース・アダプター」ページの「RAR のインストール」ボタン

4. 「RAR ファイルのインストール」ページで、「**参照**」をクリックし、ご使用のアダプターの RAR ファイルへ移動します。

   RAR ファイルは、通常、*WID_installation_directory*/ResourceAdapters/*adapter_name*/deploy/*adapter*.rarのパスにインストールされます。
5. 「**次へ**」をクリックします。
6. 「リソース・アダプター」ページで、必要に応じてアダプターの名前を変更し、説明を追加します。
7. 「**OK**」をクリックします。
8. ページの上部にある「**メッセージ**」ボックスで「**保管**」をクリックします。

次のタスク

次の手順は、サーバーにデプロイできる EAR ファイルとしてモジュールをエクスポートすることです。

EAR ファイルとしてのモジュールのエクスポート

WebSphere Integration Developer を使用して、モジュールを EAR ファイルとしてエクスポートします。EAR ファイルを作成することによって、モジュールのすべての内容を WebSphere Process Server または WebSphere Enterprise Service Bus に容易にデプロイできる形式で取り込みます。

始める前に

モジュールを EAR ファイルとしてエクスポートするには、事前にサービスと通信するためのモジュールを作成しておく必要があります。このモジュールを、WebSphere Integration Developer ビジネス・インテグレーション・パースペクティブ内に表示する必要があります。

このタスクを実行する理由および時期

モジュールを EAR ファイルとしてエクスポートするには、以下の手順を実行します。

このタスクの手順

1. モジュールを右クリックして、「**エクスポート**」を選択します。
2. 「選択」ウィンドウで、「**J2EE**」を展開します。
3. 「**EAR ファイル**」を選択して、「**次へ**」をクリックします。

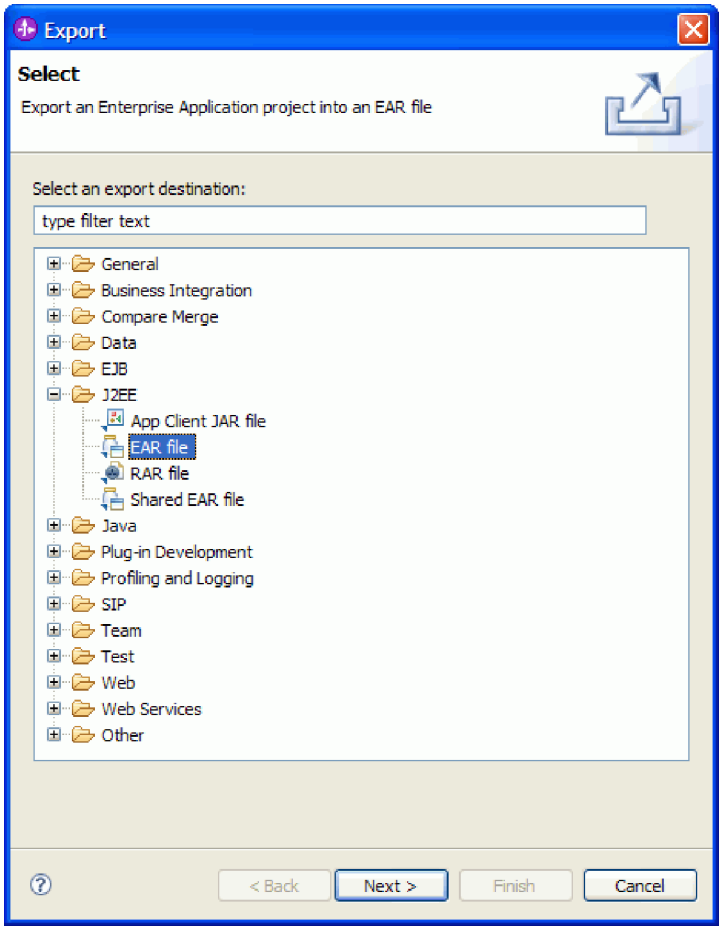


*図 23. 「選択」ウィンドウからの「**EAR ファイル**」の選択*

4. オプション: 正しい EAR アプリケーションを選択します。 EAR アプリケーションにはモジュールと同じ名前が付けられますが、名前の末尾に「App」が追加されます。
5. EAR ファイルを格納するローカル・ファイル・システム上で、「**参照**」を選択してフォルダーを参照します。
6. 必要に応じて、ソース・ファイルをエクスポートする場合は、「**ソース・ファイルのエクスポート**」を選択します。 このオプションは、EAR ファイルのほかにソース・ファイルをエクスポートする場合に表示されます。ソース・ファイルには、Java コンポーネント、データ・マップなどに関連付けられているファイルがあります。
7. 既存のファイルを上書きする場合は、「**既存のファイルを上書き (Overwrite an existing file)**」をクリックします。
8. 「**終了**」をクリックします。

結果

モジュールの内容が EAR ファイルとしてエクスポートされます。

次のタスク

このモジュールを管理コンソールにインストールします。これにより、モジュールが WebSphere Process Server にデプロイされます。

EAR ファイルのインストール

EAR ファイルのインストールは、デプロイメント・プロセスの最終手順です。EAR ファイルをサーバーにインストールして実行すると、EAR ファイルの一部として組み込まれているアダプターが、インストール済みアプリケーションの一部として稼働します。

始める前に

モジュールを WebSphere Process Server にインストールするには、その前にモジュールを EAR ファイルとしてエクスポートしておく必要があります。

このタスクを実行する理由および時期

EAR ファイルをインストールするには、次の手順を実行します。アダプター・モジュール・アプリケーションのクラスター化については、http://www.ibm.com/software/webservers/appserv/was/library/を参照してください。

このタスクの手順

1. サーバー・インスタンスを右クリックし、「**管理コンソールの実行**」を選択して、WebSphere Process Server 管理コンソールを開きます。
2. 管理コンソール・ウィンドウで、「**アプリケーション**」 → 「**新規アプリケーションのインストール**」をクリックします。

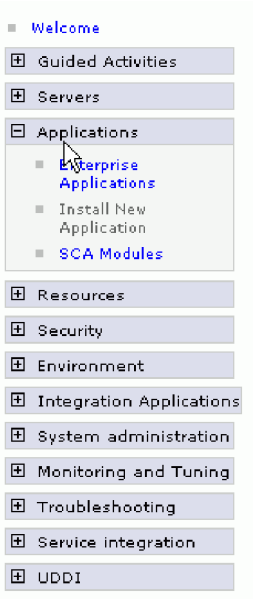


図 24. 「アプリケーション・インストールの準備」ウィンドウ

3. 「**参照**」をクリックして、EAR ファイルを位置指定し、「**次へ**」をクリックします。 EAR ファイル名は、モジュール名の後に「App」が付いたものです。
4. オプション: クラスター化された環境にデプロイする場合は、以下の手順を実行します。

a. 「**ステップ 2: サーバーにモジュールをマップ**」ウィンドウで、モジュールを選択します。
b. サーバー・クラスターの名前を選択します。
c. 「**適用 (Apply)**」をクリックします。

5. 「**次へ**」をクリックして、「要約」を開きます。すべての設定が正しいことを確認して、「**終了**」をクリックします。
6. オプション: 認証別名を使用している場合は、以下の手順を実行します。
   a. 「**セキュリティー**」を展開して、「**ビジネス・インテグレーションの認証別名 (Business Integration Authentication Aliases)**」を選択します。
   b. 構成する認証別名を選択します。 認証別名の構成を変更するための管理者権限またはオペレーター権限を持っている必要があります。
   c. オプション: 「**ユーザー名**」を入力します (まだ入力されていない場合)。
   d. 「**パスワード**」を入力します (まだ入力されていない場合)。
   e. 「**確認パスワード (Confirm Password)**」フィールドに再度パスワードを入力します (まだ入力されていない場合)。
   f. 「**OK**」をクリックします。

結果

この時点で、プロジェクトがデプロイメントされ、「エンタープライズ・アプリケーション」ウィンドウが表示されます。

次のタスク

いずれかのプロパティーを設定または再設定する場合、あるいは、アダプター・プロジェクトのアプリケーションをクラスター化したい場合は、トラブルシューティング・ツールを構成する前に、管理コンソールを使用して対応する変更を行ってください。

第 7 章 アダプター・モジュールの管理

アダプターをスタンドアロンのデプロイメントで稼働している場合は、アダプター・モジュールの開始、停止、モニター、およびトラブルシューティングには、サーバーの管理コンソールを使用します。組み込みアダプターを使用しているアプリケーションでは、アプリケーションの開始時または停止時にアダプター・モジュールが開始または停止します。

組み込みアダプターの構成プロパティーの変更

アダプターをモジュールの一部としてデプロイした後に構成プロパティーを変更するには、実行時環境の管理コンソールを使用します。リソース・アダプター・プロパティー (一般的なアダプター操作に使用)、管理接続ファクトリー・プロパティー (Outbound 処理に使用)、およびアクティベーション・スペック・プロパティー (Inbound 処理に使用) を更新できます。

組み込みアダプターのリソース・アダプター・プロパティーの設定

アダプターをモジュールの一部としてデプロイした後に、このアダプターのリソース・アダプター・プロパティーを設定するには、管理コンソールを使用します。構成するプロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプター・モジュールを WebSphere Process Server または WebSphere Enterprise Service Bus 上にデプロイする必要があります。

このタスクを実行する理由および時期

カスタム・プロパティーとは、すべての WebSphere アダプターが共用するデフォルト構成プロパティーです。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

このタスクの手順

1. 管理コンソールを開始します。
2. 「**アプリケーション**」の下で、「**エンタープライズ・アプリケーション**」を選択します。
3. 「**エンタープライズ・アプリケーション**」リストから、プロパティーを変更するアダプター・モジュールの名前をクリックします。
4. 「**モジュール**」の下で、「**モジュールの管理**」をクリックします。

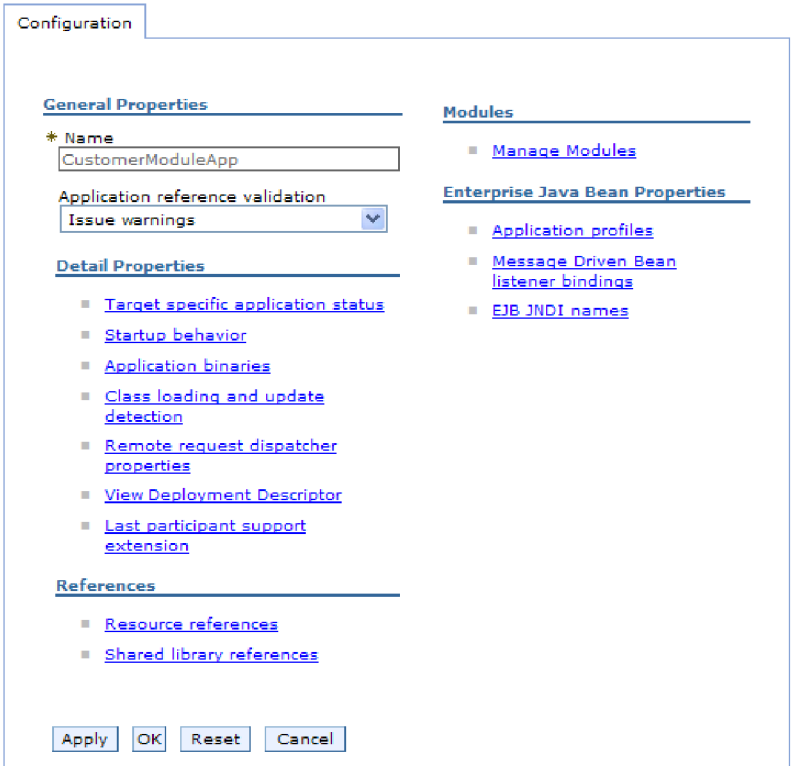


図 25. 「構成」タブでの「モジュールの管理」の選択

5. 「**IBM WebSphere Adapter for JDBC**」をクリックします。
6. 「**追加プロパティー**」リストから、「**リソース・アダプター**」をクリックします。
7. 次のページで、「**追加プロパティー**」リストから、「**カスタム・プロパティー**」をクリックします。
8. 変更するプロパティーごとに、以下の手順を実行します。

   注: ここで示すプロパティーについて詳しくは、 213 ページの『リソース・アダプター・プロパティー』を参照してください。

   a. プロパティーの名前をクリックします。
   b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。

      例えば、「**logNumberOfFiles**」をクリックすると、次のページが表示されます。

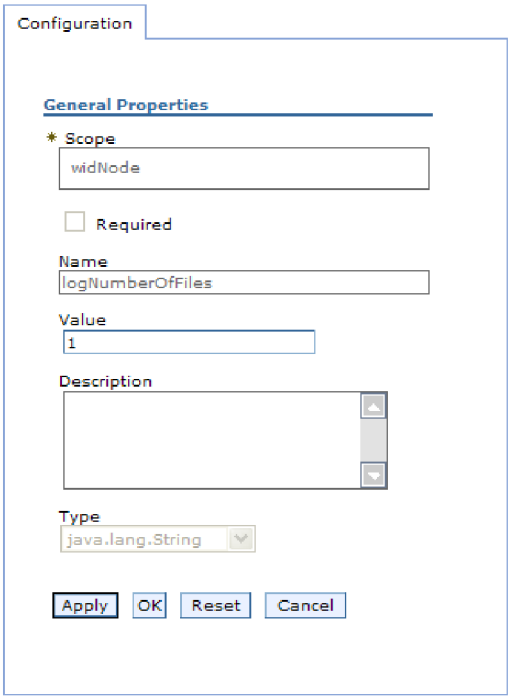


図 26. logNumberOfFiles プロパティーの「構成」タブ

「**値**」フィールドの数値を変更して、プロパティーの説明を追加できます。

c. 「**OK**」をクリックします。

9. ウィンドウの上部にある「**メッセージ**」ボックス内の「**保管**」リンクをクリックします。

結果

アダプター・モジュールに関連付けられているリソース・アダプター・プロパティーが変更されました。

組み込みアダプターの管理 (J2C) 接続ファクトリー・プロパティーの設定

アダプターをモジュールの一部としてデプロイした後に、このアダプターの管理接続ファクトリー・プロパティーを設定するには、管理コンソールを使用します。構成するプロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプター・モジュールを WebSphere Process Server または WebSphere Enterprise Service Bus 上にデプロイする必要があります。

このタスクを実行する理由および時期

管理接続ファクトリー・プロパティーは、ターゲット・データベースのインスタンスを構成する場合に使用します。

注: 管理コンソール内では、このプロパティーを「J2C 接続ファクトリー・プロパティー」と呼びます。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

このタスクの手順

1. 管理コンソールを開始します。
2. 「**アプリケーション**」の下で、「**エンタープライズ・アプリケーション**」を選択します。
3. 「**エンタープライズ・アプリケーション**」リストから、プロパティーを変更するアダプター・モジュールの名前をクリックします。
4. 「**モジュール**」の下で、「**モジュールの管理**」をクリックします。

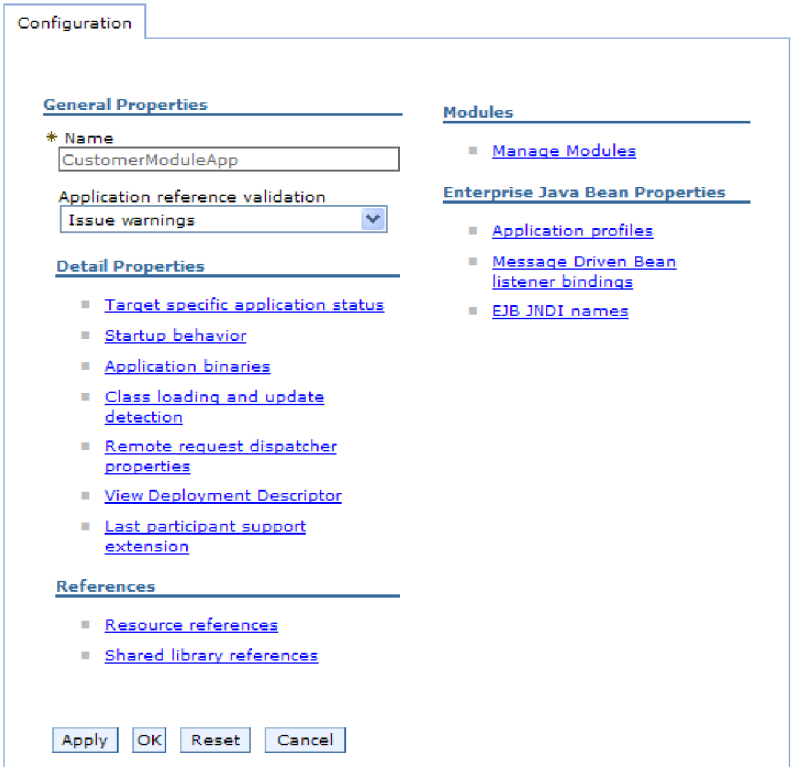


図 27. 「構成」タブでの「モジュールの管理」の選択

5. 「**IBM WebSphere Adapter for JDBC**」をクリックします。
6. 「**追加プロパティー**」リストから、「**リソース・アダプター**」をクリックします。
7. 次のページで、「**追加プロパティー**」リストから「**J2C 接続ファクトリー**」をクリックします。
8. アダプター・モジュールに関連付けられた接続ファクトリーの名前をクリックします。
9. 「**追加プロパティー**」リストで、「**カスタム・プロパティー**」をクリックします。

カスタム・プロパティーは、Adapter for JDBC に固有の J2C 接続ファクトリー・プロパティーです。接続プールおよび拡張接続ファクトリー・プロパティーは、ユーザーが独自にアダプターを作成する場合に構成するプロパティーです。

10. 変更するプロパティーごとに、以下の手順を実行します。

    注: ここで示すプロパティーについて詳しくは、217 ページの『管理接続ファクトリー・プロパティー』を参照してください。

    a. プロパティーの名前をクリックします。
    b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。
    c. 「**OK**」をクリックします。
11. ウィンドウの上部にある「**メッセージ**」ボックス内の「**保管**」リンクをクリックします。

結果

アダプター・モジュールに関連付けられた管理接続ファクトリー・プロパティーが変更されます。

組み込みアダプターのアクティベーション・スペック・プロパティーの設定

アダプターをモジュールの一部としてデプロイした後に、そのアダプターのアクティベーション・スペック・プロパティーを設定するには、管理コンソールを使用します。構成するメッセージ・エンドポイント・プロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプター・モジュールを WebSphere Process Server または WebSphere Enterprise Service Bus 上にデプロイする必要があります。

このタスクを実行する理由および時期

アクティベーション・スペック・プロパティーは、エンドポイントを Inbound 処理用に構成する場合に使用します。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

このタスクの手順

1. 管理コンソールを開始します。
2. 「**アプリケーション**」の下で、「**エンタープライズ・アプリケーション**」を選択します。
3. 「**エンタープライズ・アプリケーション**」リストから、プロパティーを変更するアダプター・モジュールの名前をクリックします。
4. 「**モジュール**」の下で、「**モジュールの管理**」をクリックします。

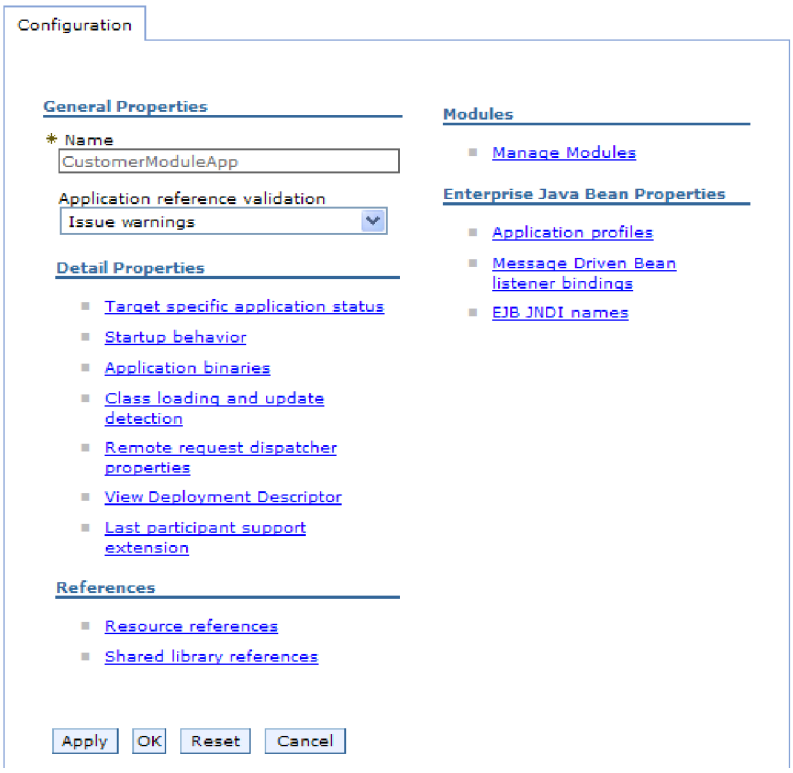


図 28. 「構成」タブでの「モジュールの管理」の選択

5. 「**IBM WebSphere Adapter for JDBC**」をクリックします。
6. 「**追加プロパティー**」リストから、「**リソース・アダプター**」をクリックします。
7. 次のページで、「**追加プロパティー**」リストから、「**J2C アクティベーション・スペック**」をクリックします。
8. アダプター・モジュールに関連付けられているアクティベーション・スペックの名前をクリックします。
9. 「**追加プロパティー**」リストから、「**J2C アクティベーション・スペックのカスタム・プロパティー**」をクリックします。
10. 変更するプロパティーごとに、以下の手順を実行します。

    注: ここで示すプロパティーについて詳しくは、238 ページの『アクティベーション・スペック・プロパティー』を参照してください。

    a. プロパティーの名前をクリックします。
    b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。
    c. 「**OK**」をクリックします。
11. ウィンドウの上部にある「**メッセージ**」ボックス内の「**保管**」リンクをクリックします。

結果

アダプター・モジュールに関連付けられているアクティベーション・スペック・プロパティーが変更されました。

スタンドアロン・アダプターの構成プロパティーの変更

スタンドアロン・アダプターのインストール後に構成プロパティーを設定するには、実行時環境の管理コンソールを使用します。アダプターに関する一般的な情報を入力して、(汎用のアダプター操作に使用される) リソース・アダプター・プロパティーを設定します。アダプターを Outbound 操作に使用する場合は、接続ファクトリーを作成して、それに対してプロパティーを設定します。アダプターを Inbound 操作に使用する場合は、アクティベーション・スペックを作成して、それに対してプロパティーを設定します。

スタンドアロン・アダプターのリソース・アダプター・プロパティーの設定

スタンドアロン・アダプターを WebSphere Process Server または WebSphere Enterprise Service Bus にインストールした後に、そのアダプターのリソース・アダプター・プロパティーを設定するには、管理コンソールを使用します。構成するプロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプターを WebSphere Process Server または WebSphere Enterprise Service Bus にインストールしておく必要があります。

このタスクを実行する理由および時期

カスタム・プロパティーとは、すべての WebSphere アダプターが共用するデフォルト構成プロパティーです。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

このタスクの手順

1. 管理コンソールを開始します。
2. 「**リソース**」 → 「**リソース・アダプター**」 → 「**リソース・アダプター**」をクリックします。
3. 「リソース・アダプター」ページで、「**IBM WebSphere Adapter for JDBC**」をクリックします。
4. 「**追加プロパティー**」リストで、「**カスタム・プロパティー**」をクリックします。
5. 変更するプロパティーごとに、以下の手順を実行します。

   注: ここで示すプロパティーについて詳しくは、 213 ページの『リソース・アダプター・プロパティー』を参照してください。

   a. プロパティーの名前をクリックします。
   b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。

例えば、「**logNumberOfFiles**」をクリックすると、次のページが表示されます。

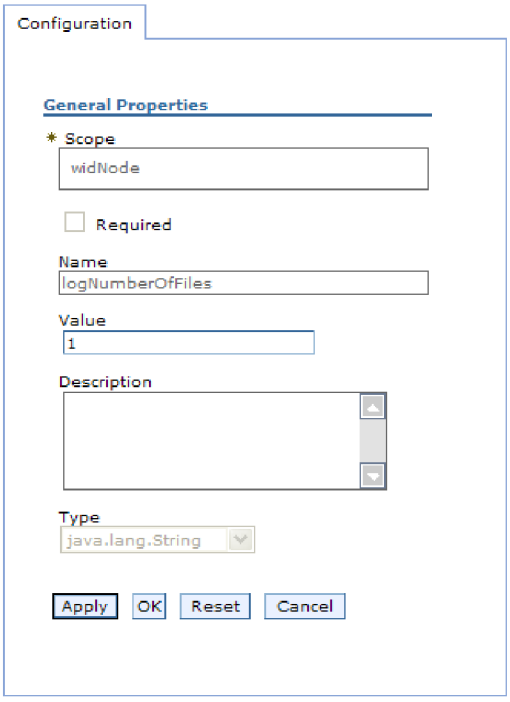


図 29. logNumberOfFiles プロパティーの「構成」タブ

「**値**」フィールドの数値を変更して、プロパティーの説明を追加できます。

c. 「**OK**」をクリックします。

6. ページの上部にある「**メッセージ**」ボックスで「**保管**」をクリックします。

結果

アダプターに関連付けられているリソース・アダプター・プロパティーが変更されました。

スタンドアロン・アダプターの管理 (J2C) 接続ファクトリー・プロパティーの設定

スタンドアロン・アダプターを WebSphere Process Server または WebSphere Enterprise Service Bus にインストールした後に、そのアダプターの管理接続ファクトリー・プロパティーを設定するには、管理コンソールを使用します。構成するプロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプターを WebSphere Process Server または WebSphere Enterprise Service Bus にインストールしておく必要があります。

このタスクを実行する理由および時期

管理接続ファクトリー・プロパティーは、ターゲット・データベースのインスタンスを構成する場合に使用します。

注: 管理コンソール内では、このプロパティーを「J2C 接続ファクトリー・プロパティー」と呼びます。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

このタスクの手順

1. 管理コンソールを開始します。
2. 「**リソース**」 → 「**リソース・アダプター**」 → 「**リソース・アダプター**」をクリックします。
3. 「リソース・アダプター」ページで、「**IBM WebSphere Adapter for JDBC**」をクリックします。
4. 「**追加プロパティー**」リストで、「**J2C 接続ファクトリー**」をクリックします。
5. 既存の接続ファクトリーを使用する場合は、手順 6 に進んでください。

   注: 外部サービス・ウィザードを使用してアダプター・モジュールを構成したときに「**事前定義された接続プロパティーを使用する**」を選択していた場合は、接続ファクトリーを作成する必要はありません。

   接続ファクトリーを作成する場合は、以下の手順を実行します。

   a. 「**新規作成**」をクリックします。
   b. 「**構成**」タブの「**一般プロパティー**」セクションで、接続ファクトリーの名前を入力します。例えば、`AdapterCF` と入力できます。
   c. 「**JNDI 名**」に値を入力します。例えば、`com/eis/AdapterCF` と入力できます。
   d. 「**コンポーネント管理認証別名**」リストから認証別名を選択します。
   e. 「**OK**」をクリックします。
   f. ページの上部にある「**メッセージ**」ボックスで「**保管**」をクリックします。

   新規に作成された接続ファクトリーが表示されます。

図 30. 接続ファクトリーのリスト

6. 接続ファクトリーのリストから、使用するものをクリックします。
7. 「**追加プロパティー**」リストで、「**カスタム・プロパティー**」をクリックします。

カスタム・プロパティーは、Adapter for JDBC に固有の J2C 接続ファクトリー・プロパティーです。接続プールおよび拡張接続ファクトリー・プロパティーは、ユーザーが独自にアダプターを作成する場合に構成するプロパティーです。

8. 変更するプロパティーごとに、以下の手順を実行します。

   注: ここで示すプロパティーについて詳しくは、217 ページの『管理接続ファクトリー・プロパティー』を参照してください。

   a. プロパティーの名前をクリックします。
   b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。
   c. 「**OK**」をクリックします。
9. プロパティーの設定が終了したら、「**適用**」をクリックします。
10. ウィンドウの上部にある「**メッセージ**」ボックスで「**保管**」をクリックします。

結果

アダプターに関連付けられている管理接続ファクトリー・プロパティーが設定されます。

スタンドアロン・アダプターのアクティベーション・スペック・プロパティーの設定

スタンドアロン・アダプターを WebSphere Process Server または WebSphere Enterprise Service Bus にインストールした後に、そのアダプターのアクティベーション・スペック・プロパティーを設定するには、管理コンソールを使用します。構成するメッセージ・エンドポイント・プロパティーの名前を選択してから、その値を変更または設定します。

始める前に

アダプターを WebSphere Process Server または WebSphere Enterprise Service Bus にインストールしておく必要があります。

このタスクを実行する理由および時期

アクティベーション・スペック・プロパティーは、エンドポイントを Inbound 処理用に構成する場合に使用します。

管理コンソールを使用してプロパティーを構成するには、以下の手順を実行します。

このタスクの手順

1. 管理コンソールを開始します。
2. 「**リソース**」 → 「**リソース・アダプター**」 → 「**リソース・アダプター**」をクリックします。
3. 「リソース・アダプター」ページで、「**IBM WebSphere Adapter for JDBC**」をクリックします。

4. 「**追加プロパティー**」リストで、「**J2C アクティベーション・スペック**」をクリックします。
5. 既存のアクティベーション・スペックを使用する場合は、手順 6 に進んでください。

   注: 外部サービス・ウィザードを使用してアダプター・モジュールを構成したときに「**事前定義された接続プロパティーを使用する**」を選択していた場合は、アクティベーション・スペックを作成する必要はありません。

   アクティベーション・スペックを作成する場合は、以下の手順を実行します。
   a. 「**新規作成**」をクリックします。
   b. 「**構成**」タブの「**一般プロパティー**」セクションで、アクティベーション・スペックの名前を入力します。例えば、AdapterAS と入力できます。
   c. 「**JNDI 名**」に値を入力します。例えば、com/eis/AdapterAS と入力できます。
   d. 「**認証別名**」リストから認証別名を選択します。
   e. メッセージ・リスナー・タイプを選択します。
   f. 「**OK**」をクリックします。
   g. ページの上部にある「**メッセージ**」ボックスで「**保管**」をクリックします。

   新規に作成されたアクティベーション・スペックが表示されます。
6. アクティベーション・スペックのリストから、使用するものをクリックします。
7. 「追加プロパティー」リストから、「**J2C アクティベーション・スペックのカスタム・プロパティー**」をクリックします。
8. 設定するプロパティーごとに、次の手順を実行します。

   注: ここで示すプロパティーについて詳しくは、238 ページの『アクティベーション・スペック・プロパティー』を参照してください。
   a. プロパティーの名前をクリックします。
   b. 「**値**」フィールドの値の内容を変更するか、フィールドが空の場合は値を入力します。
   c. 「**OK**」をクリックします。
9. プロパティーの設定が終了したら、「**適用**」をクリックします。
10. ページの上部にある「**メッセージ**」ボックスで「**保管**」をクリックします。

結果

アダプターに関連付けられたアクティベーション・スペック・プロパティーが設定されます。

アダプターを使用するアプリケーションの開始

アダプターを使用するアプリケーションを開始するには、サーバーの管理コンソールを使用します。デフォルトでは、サーバーが始動すると、アプリケーションは自動的に開始します。

このタスクを実行する理由および時期

アプリケーションが使用するのが組み込みアダプターの場合でもスタンドアロン・アダプターの場合でも、アプリケーションを開始するには、以下の手順に従います。組み込みアダプターを使用するアプリケーションの場合、アダプターはアプリケーションの開始時に開始されます。スタンドアロン・アダプターを使用するアプリケーションの場合、アダプターはアプリケーション・サーバーの始動時に開始されます。

このタスクの手順

1. 管理コンソールで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックします。

   注: 管理コンソールには、「Integrated Solutions Console」というラベルが付いています。
2. 開始するアプリケーションのチェック・ボックスを選択します。アプリケーション名は、インストールした EAR ファイルの名前からファイル拡張子 .EAR を除いたものです。
3. 「**開始**」をクリックします。

結果

アプリケーションの状況が「開始済み」に変化し、アプリケーションが開始されたことを示すメッセージが管理コンソールの上部に表示されます。

アダプターを使用するアプリケーションの停止

アダプターを使用するアプリケーションを停止するには、サーバーの管理コンソールを使用します。デフォルトでは、サーバーが停止すると、アプリケーションは自動的に停止します。

このタスクを実行する理由および時期

アプリケーションが使用するのが組み込みアダプターの場合でもスタンドアロン・アダプターの場合でも、アプリケーションを停止するには、以下の手順に従います。アプリケーションと組み込みアダプターの組み合わせの場合、アダプターはアプリケーションの停止時に停止します。スタンドアロン・アダプターを使用するアプリケーションの場合、アダプターはアプリケーション・サーバーの停止時に停止します。

このタスクの手順

1. 管理コンソールで、「**アプリケーション**」 → 「**エンタープライズ・アプリケーション**」をクリックします。

注: 管理コンソールには、「Integrated Solutions Console」というラベルが付いています。

2. 停止するアプリケーションのチェック・ボックスを選択します。アプリケーション名は、インストールした EAR ファイルの名前からファイル拡張子 .EAR を除いたものです。
3. 「**停止 (Stop)**」をクリックします。

結果

アプリケーションの状況が「停止」に変化し、アプリケーションが停止したことを示すメッセージが管理コンソールの上部に表示されます。

Performance Monitoring Infrastructure を使用したパフォーマンスのモニター

Performance Monitoring Infrastructure (PMI) は、管理コンソールの機能の 1 つで、これを使用すると、実稼働環境内で Adapter for JDBC を含む、コンポーネントのパフォーマンスを動的にモニターすることができます。PMI は、サーバー内のさまざまなコンポーネントから、平均応答時間や要求の総数などのアダプターのパフォーマンス・データを収集して、そのデータをツリー構造に編成します。このデータは、Tivoli® Performance Viewer (WebSphere Process Server の管理コンソールに統合されているグラフィカル・モニター・ツール) を通して表示することができます。

このタスクを実行する理由および時期

PMI により、以下の時点のデータを収集することによって、アダプターのパフォーマンスをモニターすることができます。

- Outbound 処理時。Outbound 要求をモニターします。
- Inbound イベントの取り出し時。イベント・テーブルからのイベントの取り出しをモニターします。
- Inbound イベントの送達時。エンドポイント (1 つまたは複数の) へのイベントの送達をモニターします。

使用するアダプター用に PMI を使用可能に設定し、構成するためには、まず、トレース機能の詳細レベルを設定し、パフォーマンス・データの収集元となるいくつかのイベントを実行する必要があります。

ご使用のアダプター環境の全体的なパフォーマンスをモニターし、それを向上させるために PMI を役立てる方法について詳しくは、WebSphere Application Server の Web サイト (http://www.ibm.com/software/webservers/appserv/was/library/) で PMI を検索してください。

Performance Monitoring Infrastructure の構成

Performance Monitoring Infrastructure (PMI) を、アダプターのパフォーマンス・データ (平均応答時間や要求の総数など) を収集するように構成することができます。使用するアダプター用に PMI を構成した後、Tivoli Performance Viewer を使用してアダプターのパフォーマンスをモニターすることができます。

始める前に

使用するアダプター用に PMI を構成するためには、まず、トレース機能の詳細レベルを設定し、パフォーマンス・データの収集元となるいくつかのイベントを実行する必要があります。

1. トレース機能を使用可能にしてイベント・データを受け取るためには、トレース・レベルを fine、finer、finest、または all のいずれかに設定する必要があります。*=info の後に、コロンとストリングを追加します。例えば、次のように入力します。

   ```
   *=info: WBILocationMonitor.CEI.ResourceAdapter.
   *=finest: WBILocationMonitor.LOG.ResourceAdapter.*=finest:
   ```

   トレース・レベルの設定方法については、169 ページの『Common Event Infrastructure (CEI) を使用したトレースの使用可能化』を参照してください。
2. 1 つ以上の Outbound 要求 または Inbound イベントを生成して、構成可能なパフォーマンス・データを生成します。

このタスクの手順

1. アダプターに対して PMI を使用可能にします。
   a. 管理コンソールで、「**モニターおよびチューニング**」を展開してから、「**Performance Monitoring Infrastructure (PMI)**」を選択します。
   b. サーバーのリストから、ご使用のサーバーの名前をクリックします。
   c. 「構成」タブを選択してから、「**Performance Monitoring (PMI) を使用可能にする (Enable Performance Monitoring (PMI))**」チェック・ボックスを選択します。
   d. 「**カスタム**」を選択して、選択的に統計を使用可能または使用不可に設定します。

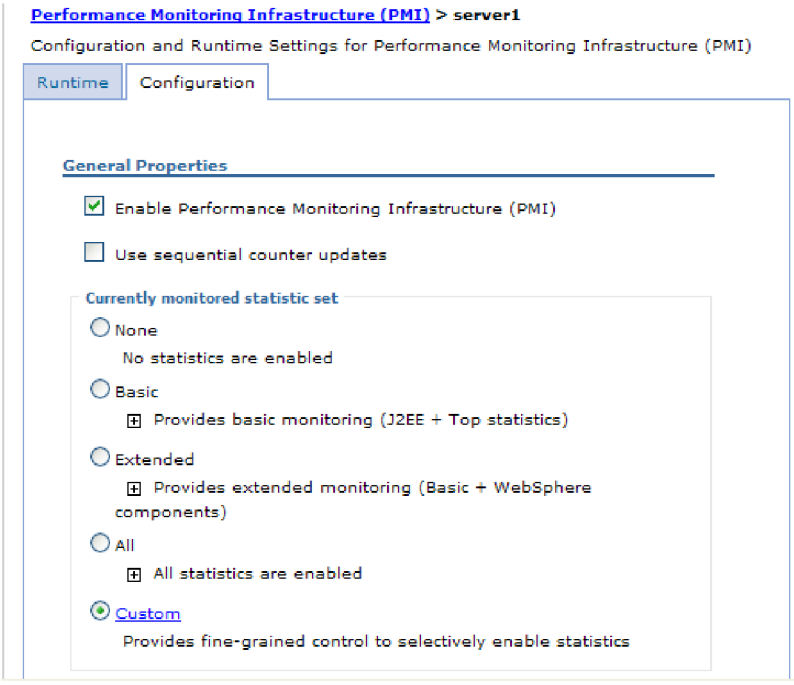


図 31. Performance Monitoring Infrastructure の使用可能化

e. 「**適用**」または「**OK**」をクリックします。

f. 「**保管**」をクリックします。 これで、PMI が使用可能になりました。

2. アダプター用に PMI を構成します。

a. 管理コンソールで、「**モニターおよびチューニング**」を展開してから、「**Performance Monitoring Infrastructure (PMI)**」を選択します。

b. サーバーのリストから、ご使用のサーバーの名前をクリックします。

c. 「**カスタム**」を選択します。

d. 「**ランタイム**」タブを選択します。 以下の図は、「ランタイム」タブを示しています。

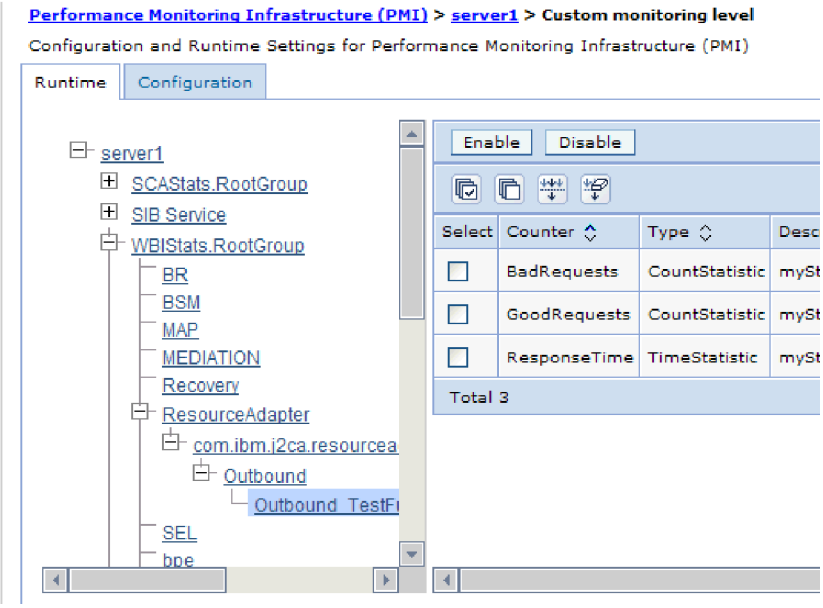


図 32. PMI の構成に使用される「ランタイム」タブ

e. 「**WBIStats.RootGroup**」をクリックします。 これは、ルート・グループで収集されるデータ用の PMI サブモジュールです。この例では、ルート・グループに WBIStats という名前を使用しています。

f. 「**ResourceAdapter**」をクリックします。 これは、JCA アダプターについて収集されるデータ用のサブモジュールです。

g. アダプターの名前をクリックして、モニターするプロセスを選択します。

h. 右側のペインで、収集する統計のチェック・ボックスを選択してから、「**使用可能**」をクリックします。

結果

PMI がアダプター用に構成されます。

次のタスク

これで、アダプターのパフォーマンス統計を表示することができるようになりました。

パフォーマンスに関する統計の表示

アダプターのパフォーマンス・データは、グラフィカル・モニター・ツール Tivoli Performance Viewer を使用して表示することができます。Tivoli Performance Viewer は、WebSphere Process Server の管理コンソールに組み込まれています。

始める前に

アダプター用の Performance Monitoring Infrastructure の構成。

このタスクの手順

1. 管理コンソールで、「**モニターおよびチューニング**」を展開し、「**Performance Viewer**」を展開した後、「**現行アクティビティー**」を選択します。

2. サーバーのリストにて、ご使用のサーバーの名前をクリックします。
3. サーバー名の下で、「**パフォーマンス・モジュール**」を展開します。
4. 「**WBIStatsRootGroup**」をクリックします。
5. 「**ResourceAdapter**」およびアダプター・モジュールの名前をクリックします。
6. 複数のプロセスがある場合は、統計を表示させるプロセスのチェック・ボックスを選択します。

結果

右側のパネルに統計が表示されます。「**グラフの表示**」をクリックして、データのグラフを表示するか、または「**表の表示**」をクリックして、統計を表形式で表示することができます。以下の図では、アダプターのパフォーマンス統計をグラフの形で表示しています。

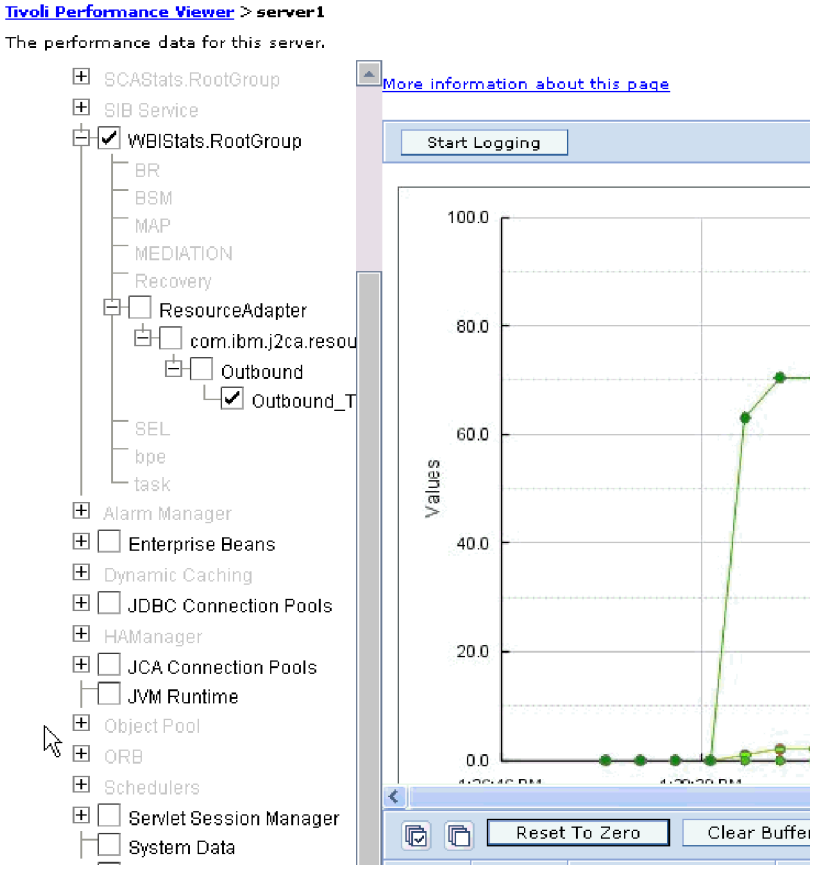


図 33. グラフ表示によるアダプターのパフォーマンス統計

Common Event Infrastructure (CEI) を使用したトレースの使用可能化

アダプターは、サーバー内に組み込まれたコンポーネントである Common Event Infrastructure を使用して、ポーリング周期の開始または停止などの重要なビジネス・イベントに関するデータを通知できます。イベント・データの書き込み先は、構成設定に応じてデータベースまたはトレース・ログ・ファイルになります。

このタスクの手順

1. 管理コンソールにて、「**トラブルシューティング**」をクリックします。
2. 「**ログおよびトレース**」を選択します。
3. サーバーのリストにて、ご使用のサーバーの名前をクリックします。
4. 「**ログ詳細レベルの変更**」ボックスで、アダプターによるイベント・データの書き込み先にする CEI データベースの名前 (例えば、WBIEventMonitor.CEI.ResourceAdapter.*) またはトレース・ログ・ファイルの名前 (例えば、WBIEventMonitor.LOG.ResourceAdapter.*) をクリックします。
5. アダプターを使用してデータベースまたはトレース・ログ・ファイルに書き込むビジネス・イベントの詳細レベルを選択し、(必要に応じて) メッセージおよびトレースに関連付けられている詳細レベルの細分度を調整します。
   - **ロギングなし**。イベント・ロギングをオフにします。
   - **メッセージのみ**。アダプターはイベントを通知します。
   - **すべてのメッセージおよびトレース**。アダプターは、イベントの詳細を通知します。
   - **メッセージとトレースのレベル**。イベントに関連付けられているビジネス・オブジェクト・ペイロードについてアダプターが通知する詳細度を制御するための設定です。詳細度を調整する場合は、以下のいずれかを選択してください。

     詳細 - 中。アダプターはイベントを通知しますが、ビジネス・オブジェクト・ペイロードについては通知しません。

     詳細 - 高。アダプターは、イベントおよびビジネス・オブジェクト・ペイロードの説明を通知します。

     詳細 - 最高。アダプターは、イベントおよびすべてのビジネス・オブジェクト・ペイロードを通知します。
6. 「**OK**」をクリックします。

結果

イベント・ロギングが使用可能になります。CEI 項目は、トレース・ログ・ファイル内で参照できます。または、管理コンソール内で Common Base Event Browser を使用して表示することもできます。

トラブルシューティングとサポート

共通のトラブルシューティング手法とセルフ・ヘルプ情報は、問題を迅速に識別して解決するのに役立ちます。

ロギングおよびトレースの構成

要件に合うようロギングおよびトレースを構成します。アダプターのロギングを使用可能にし、イベント処理の状況を制御します。アダプターのログ・ファイル名およびトレース・ファイル名を変更して、ほかのログ・ファイルおよびトレース・ファイルと区別します。

ロギング・プロパティーの構成

ログを使用可能にし、ログの出力プロパティー (ログのロケーション、詳細レベル、出力形式など) を設定するには、管理コンソールを使用します。

このタスクを実行する理由および時期

アダプターでモニター対象イベントをログに記録できるようにするには、モニターしたいサービス・コンポーネントのイベント・ポイント、イベントごとに必要となる詳細レベル、およびイベントをログにパブリッシュするのに使用する出力のフォーマットを指定する必要があります。管理コンソールを使用して、次のタスクを実行します。

- 特定のイベント・ログを使用可能または使用不可に設定する
- ログの詳細レベルを指定する
- ログ・ファイルの保管場所と保持数を指定する。
- ログの出力形式を指定する。

  ログ・アナライザーの出力形式を設定した場合は、ログ・アナライザー・ツール (プロセス・サーバーに同梱されるアプリケーション) を使用して、トレース出力を開くことができます。これは、2 つの異なるサーバー・プロセスからのトレースを相関しようとする場合に便利です。なぜなら、これにより、ログ・アナライザーのマージ機能が使用できるからです。

プロセス・サーバー (サービス・コンポーネントとイベント・ポイントを含む) のモニターの詳細については、ご使用のプロセス・サーバーの資料を参照してください。

ログ構成は、静的または動的に変更できます。アプリケーション・サーバーを開始または再始動すると、静的構成が有効になります。動的構成 (実行時構成) の変更は、直ちに適用されます。

ログが作成されると、そのログの詳細レベルが構成データから設定されます。特定のログ名に対して、構成データが使用可能でない場合、そのログのレベルは、ログの親から取得されます。親ログに構成データが存在しない場合、そのログの親が確認される、という具合に、ヌル以外のレベル値があるログが見つかるまでツリーを上昇します。ログのレベルを変更すると、その変更はログの子に伝搬されます。また、必要に応じて、ログの子からその子へと変更が再帰的に伝搬されます。

ロギングを使用可能にし、ログの出力プロパティーを設定するには、以下の手順を実行します。

このタスクの手順

1. 管理コンソールのナビゲーション・ペインで、「**サーバー**」 → 「**アプリケーション・サーバー**」をクリックします。
2. 作業したいサーバーの名前をクリックします。
3. 「**トラブルシューティング**」で「**ログおよびトレース**」をクリックします。
4. 「**ログ詳細レベルの変更**」をクリックします。
5. 変更を有効にするには、以下を行います。
   - 構成を静的に変更する場合は、「**構成**」タブをクリックします。

- 構成を動的に変更する場合は、「**ランタイム**」タブをクリックします。

6. 変更したいロギング・レベルのパッケージの名前をクリックします。
   WebSphere Adapters 用のパッケージ名は、**com.ibm.j2ca** で始まります。
   - アダプターの基本コンポーネントの場合は、**com.ibm.j2ca.base** を選択します。
   - アダプターの基本コンポーネントとすべてのデプロイ済みアダプターの場合は、 **com.ibm.j2ca.base.*** を選択します。
   - Adapter for JDBC の場合のみ、**com.ibm.j2ca.jdbc** パッケージを選択します。
7. ロギング・レベルを選択します。

ロギング・レベル	説明
致命的	タスクを続行できない。または、コンポーネントが機能しない。
重大	タスクを続行できないが、コンポーネントは機能する。このロギング・レベルには、差し迫った致命的エラーを示す (すなわち、リソースが枯渇寸前であることを強く示唆する) 状況も含まれる。
警告	潜在的なエラーが発生したか、重大エラーが差し迫っている。このロギング・レベルには、例えばリソース・リークの可能性など、進行性の障害を示す状況も含まれる。
監査	サーバーの状態やリソースに影響を与える重大なイベントが発生した。
情報	タスクが稼働中である。このロギング・レベルには、タスクの全体的な進行を概説する一般情報が含まれる。
構成	構成の状況が報告されるか、構成変更が発生した。
詳細	サブタスクが稼働中である。このロギング・レベルには、サブタスクの進行を詳細に説明した一般情報が含まれる。

8. 「**適用 (Apply)**」をクリックします。
9. 「**OK**」をクリックします。
10. 静的な構成変更を有効にするには、プロセス・サーバーを停止し、再始動します。

結果

これ以降、ログ項目には、選択したアダプター・コンポーネントについての指定したレベルの情報が格納されます。

ログ・ファイル名およびトレース・ファイル名の変更

アダプター・ログおよびトレース情報を他のプロセスとは分離して保持するには、管理コンソールを使用してファイル名を変更します。デフォルトでは、プロセス・サーバー上にあるすべてのプロセスおよびアプリケーションのログ情報およびトレース情報は、それぞれ SystemOut.log ファイルおよび trace.log ファイルに書き込まれます。

始める前に

アダプター・モジュールをアプリケーション・サーバーにデプロイした後は、ログ・ファイル名およびトレース・ファイル名はいつでも変更できます。

このタスクを実行する理由および時期

ログ・ファイルおよびトレース・ファイルは、静的または動的に変更できます。アプリケーション・サーバーを開始または再始動すると、静的変更が有効になります。動的変更またはランタイム構成変更は、即座に適用されます。

ログ・ファイルおよびトレース・ファイルは、*install_root*/profiles/*profile_name*/logs/*server_name* フォルダーにあります。

ログ・ファイル名およびトレース・ファイル名を設定または変更するには、次の手順を実行します。

このタスクの手順

1. 管理コンソールのナビゲーション・ペインで、「**アプリケーション**」>「**エンタープライズ・アプリケーション**」を選択します。
2. 「エンタープライズ・アプリケーション」リストから、アダプター・アプリケーションの名前をクリックします。これは、アダプターの EAR ファイルの名前から .ear ファイル拡張子を除いたものです。 例えば、EAR ファイルの名前が Accounting_OutboundApp.ear である場合は、**Accounting_OutboundApp** をクリックします。
3. 「構成」タブの「モジュール」リストから、「**モジュールの管理**」をクリックします。
4. モジュールのリストで、IBM WebSphere Adapter for JDBC をクリックします。
5. 「構成」タブの「追加プロパティー」の下で、「**リソース・アダプター**」をクリックします。
6. 「構成」タブの「追加プロパティー」の下で、「**カスタム・プロパティー**」をクリックします。
7. 「カスタム・プロパティー」テーブル内で、ファイル名を変更します。
   a. 「**logFilename**」をクリックして、ログ・ファイルの名前を変更します。あるいは、「**traceFilename**」をクリックして、トレース・ファイルの名前を変更します。
   b. 「構成」タブで、「**値**」フィールドに新しい名前を入力します。 デフォルトでは、ログ・ファイルの名前は SystemOut.log、トレース・ファイルの名前は trace.log になります。
   c. 「**適用**」または「**OK**」をクリックします。 変更内容がローカル・マシン上に保存されます。
   d. 変更内容をサーバー上のマスター構成に保存するには、次のいずれかの手順を実行します。
      - **静的変更**: サーバーを停止してから再始動します。この方法では、変更を行うことは可能ですが、サーバーを停止してから始動するまで、行った変更は有効になりません。
      - **動的変更**: 「カスタム・プロパティー」テーブルの上にあるメッセージ・ボックス内にある「**保管**」リンクをクリックします。プロンプトが出されたら、再度「**保管**」をクリックします。この方法では、行った変更をすぐに有効にすることができます。

First Failure Data Capture (FFDC) サポート

アダプターは、WebSphere Process Server または WebSphere Enterprise Service Bus の実行時に発生する障害や重大なソフトウェアの問題の永続的な記録を提供する First Failure Data Capture (FFDC) をサポートしています。

FFDC 機能はバックグラウンドで実行され、実行時に発生するイベントやエラーを収集します。この機能はさまざまな障害を相互に関連付ける手段を提供するため、この機能を利用すると、ソフトウェアは、ある 1 つの障害の影響をその原因に結びつけ、その結果、障害の根本原因を素早く突き止めることが容易になります。取り込まれたデータは、アダプターの実行時に発生した例外処理を識別するときに使用できます。

問題が発生すると、例外メッセージおよびコンテキスト・データがアダプターによってログ・ファイルに書き込まれます。このログ・ファイルは *install_root*/profiles/*profile*/logs/ffdc ディレクトリーに置かれます。

First Failure Data Capture (FFDC) について詳しくは、WebSphere Process Server または WebSphere Enterprise Service Bus の資料を参照してください。

ビジネス・フォールト

アダプターは、予想される例外で Outbound サービス記述で宣言されている例外であるビジネス・フォールトか、インポートをサポートします。ビジネス・フォールトは、ビジネス・ルールの違反または制約違反の結果としてビジネス・プロセスの予測可能なポイントに発生します。

WebSphere Process Server と WebSphere Enterprise Service Bus は他のタイプのフォールトをサポートしますが、アダプターはビジネス・フォールトのみを生成します。この資料ではビジネス・フォールトを単にフォールトと呼びます。すべての例外がフォールトになるわけではありません。フォールトは、アクション可能なエラー、つまり、アプリケーションの終了を必要としないリカバリー・アクションが可能なエラーに対して生成されます。例えば、アダプターで必要なデータが含まれていない Outbound 処理のビジネス・オブジェクトを受け取るか、またはアダプターで Outbound 処理中にエラーが発生した場合に、アダプターによってフォールトが生成されます。

フォールト・ビジネス・オブジェクト

外部サービス・ウィザードは、アダプターが生成可能なフォールトごとにビジネス・オブジェクトを作成します。またウィザードは、175 ページの図 34 に示すように、message、errorCode、および primarySetKey 属性などのすべてのフォールトに共通の情報を含む WBIFault スーパーセット・ビジネス・オブジェクトを作成します。

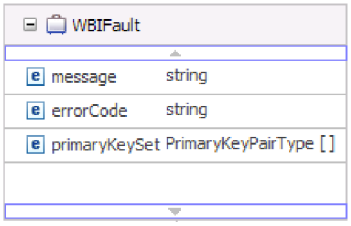


図 34. WBIFault ビジネス・オブジェクトの構造

フォールトによっては、エラーに関する追加情報を提供する matchCount 属性を含むものもあります。それ以外の場合には、フォールトを処理するために必要な情報すべてが WBIFault に含まれています。

ウィザードは、以下のフォールト・ビジネス・オブジェクトを作成します。

- IntegrityConstraintFault

  Create 操作の処理時に、データベースが整合性制約違反に関する SQLException 例外をスローすると、アダプターは整合性制約フォールトをスローします。例えば外部キーが見つからないと、アダプターからこのフォールトがスローされます。

- MatchesExceededLimitFault

  RetrieveAll 操作の処理を実行する場合、データベース照会から返されたレコード数が対話仕様内のレコードの最大数プロパティーを超えている場合は、アダプターによりこのフォールトがスローされます。

  返せるレコード数を増やすには、RetrieveAll 操作用の対話仕様プロパティー内の MaxRecords プロパティーの値を増やします。

  このフォールトのビジネス・オブジェクトには 1 つのプロパティー matchCount があり、このプロパティーは一致した数が含まれるストリングです。

- MissingDataFault

  Outbound 操作に受け渡されたビジネス・オブジェクトに、必要なすべての属性がない場合は、アダプターによりこのフォールトがスローされます。

- MultipleMatchingRecordsFault

  Retrieve 操作の処理時に、照会で指定されたキーのレコードが複数返された場合に、アダプターによりこのフォールトがスローされます。このフォールトのビジネス・オブジェクトには 1 つのプロパティー matchCount があり、このプロパティーは一致した数が含まれるストリングです。

- ObjectNotFoundFault

  Create 操作と Update 操作の処理時に、単一カーディナリティーの子オブジェクトの所有権が false の場合、アダプターはこの子オブジェクトを取得します。取得操作で何も戻されないと、アダプターからこのフォールトがスローされます。

- RecordNotFoundFault

データ検索操作の処理時に、指定されたキーのレコードがデータベース内に見つからなかった場合に、アダプターによりこのフォールトがスローされます。このフォールトは、Delete、Update、Retrieveおよび RetrieveAll の各操作に対して発生する可能性があります。

- UniqueConstraintFault

  Create 操作の処理時に、固有制約違反のために データベースから SQLException 例外を受け取った場合に、アダプターによりこのフォールトがスローされます。

フォールト処理のモジュールの構成

ご使用のモジュールをビジネス・フォールトをサポートするように構成するには、外部サービス・ウィザードを使用してモジュールを構成しておく必要があります。

フォールト処理を使用可能にするには、モジュールの .import ファイルと WSDL ファイルに変更を加える必要があります。フォールトは、バインディング・レベルまたはメソッド・レベルのいずれかで構成できます。バインディング・レベルで変更を行う場合は、インポートのすべてのメソッドにその変更が適用されます。メソッド・バインディング・レベルで変更を行う場合には、メソッドごとに異なるフォールトを構成できます。

表 13に、フォールトごとのフォールト名とフォールト・バインディングをリストします。モジュールを構成するときには、フォールト名とフォールト・バインディング・クラスを使用してください。

表 13. 各フォールトのフォールト名とフォールト・バインディング・クラス

フォールト名	関連付けられたフォールト・バインディング・クラス
INTEGRITY_CONSTRAINT_VIOLATION	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
MATCHES_EXCEEDED_LIMIT	com.ibm.j2ca.extension.emd.runtime.MatchingFaultDataBinding
MISSING_DATA	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
MULTIPLE_MATCHING_RECORDS	com.ibm.j2ca.extension.emd.runtime.MatchingFaultDataBinding
OBJECT_NOTFOUND_EXCEPTION	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
RECORD_NOT_FOUND	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
UNIQUECONSTRAINT_VIOLATION	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl

1. .import ファイルを編集して、バインディング・レベルまたはメソッド・レベルのいずれかでフォールトを構成します。
   - バインディング・レベルでフォールトを構成するには、以下の手順を実行します。
     a. バインディング・セクションで、faultSelector 属性およびフォールト・セレクターの名前を追加します。フォールト・セレクターの名前は、com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl です。
     b. 使用可能にするフォールトごとに、<faultBinding> エレメントを追加します。このエレメントで、表 13 にあるフォールト名とフォールト・データ・バインディング・クラス名を指定します。

次の .import ファイルは、すべてのメソッドで構成された MULTIPLE_MATCHING_RECORDS フォールトと RECORD_NOT_FOUND フォールトを示しています。**太字**は、フォールト処理を使用可能にするために加えられた変更を示します。

```
<esbBinding xsi:type="eis:EISImportBinding"
dataBindingType="com.ibm.j2ca.jdbc.emd.databinding.JDBCDataBindingGenerator"
 faultSelector="com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl">
    <resourceAdapter name="JDBCXAApp.IBM WebSphere Adapter for JDBC"
type="com.ibm.j2ca.jdbc.JDBCResourceAdapter" version="6.1">
       <properties>
         <adapterID>CWYBC_JDBC</adapterID>
       </properties>
    </resourceAdapter>
    <faultBinding fault="MULTIPLE_MATCHING_RECORDS"
          faultBindingType="com.ibm.j2ca.extension.emd.runtime.MatchingFa
ultDataBinding"/>
    <faultBinding fault="RECORD_NOT_FOUND"
          faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultDa
taBindingImpl"/>
```

- メソッド・レベルでフォールトを構成するには、以下の手順を実行します。
  a. フォールトに関連付けるメソッドのメソッド・バインディング・セクションで、フォールト・セレクターの名前を追加します。フォールト・セレクターの値は com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl です。
  b. メソッド・バインディング・セクションでフォールト・バインディング・エレメントを追加します。 176 ページの表 13 にあるフォールト名と対応するフォールト・データ・バインディング・クラス名を使用してください。

  次の .import ファイルは、retrieveRtasserCustomerBG メソッドに対してのみ構成された MULTIPLE_MATCHING_RECORDS フォールトと RECORD_NOT_FOUND フォールトを示しています。**太字**は、フォールト処理を使用可能にするために加えられた変更を示します。

```
<methodBinding inDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.r
tassercustomerbg.RtasserCustomerBGDataBinding"
   method="retrieveRtasserCustomerBG"
   outDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.rtassercustom
erbg.RtasserCustomerBGDataBinding"
   faultSelector="com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl">
      <interaction>
        <properties>
          <functionName>Retrieve</functionName>
        </properties>
      </interaction>
      <faultBinding fault="MULTIPLE_MATCHING_RECORDS"
         faultBindingType="com.ibm.j2ca.extension.emd.runtime.MatchingFau
ltDataBinding"/>
      <faultBinding fault="RECORD_NOT_FOUND"
         faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultDataB
indingImpl"/>
</methodBinding>
```

2. フォールトのターゲット・ネーム・スペースを決定します。使用可能にするフォールトごとに、次のようにしてネーム・スペースを決定します。
   a. テキスト・エディターでフォールト・スキーマ (XSD ファイル) を開きます。
   b. ターゲット・ネーム・スペースを見つけます。ターゲット・ネーム・スペースは、フォールト・スキーマの以下の部分に**太字**で示されています。

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://com/ibm/j2ca/fault/afcfault"
```

```
   xmlns:basefault="http://com/ibm/j2ca/fault">
<import namespace="http://com/ibm/j2ca/fault" schemaLocation="WBIFault.xsd"/>

. . .
```

フォールトには、すべて同じターゲット・ネーム・スペースを指定することも、別々のターゲット・ネーム・スペースを指定することもできます。

3. WSDL ファイルを編集してサービスのフォールトを宣言します。これらの変更を行ったサンプル WSDL ファイルをリストの終わりに示します。
   a. <definitions> エレメントで、フォールト・スキーマ・ファイルから取得した情報を使用して、フォールト・ネーム・スペースごとにネーム・スペースを追加します。すべてのフォールト・スキーマで同じ targetNamespace が使用されている場合は、別名を 1 つだけ追加します。別々の targetNamespace が使用されている場合は、固有のネーム・スペースごとに別名を 1 つ追加します。
   b. <xsd:import> エレメントを作成して、使用可能にする各フォールトのスキーマをインポートします。
   c. フォールト・タイプごとにインポート・ステートメントを宣言します。type=*alias*:*faultBOName*.xsd の複合型を解決するためにステップ 3a で定義した、正しい別名を使用していることを確認してください。
   d. フォールト・タイプのそれぞれで、メッセージ・タグを宣言します。
   e. フォールトを処理する各メソッドにフォールト宣言を追加します。

以下の WSDL ファイルでは、MULTIPLE_MATCHING_RECORDS フォールトと RECORD_NOT_FOUND フォールトを定義しています。**太字**は、フォールト処理を使用可能にするために加えられた変更を示します。

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:RtasserCustomerBG="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtassercusto
merbg"
  xmlns:RtasserCustomerContainer="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtass
ercustomercontainer"
  xmlns:intf="http://JDBCXA/JDBCOutboundInterface"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

ステップ 3a

```
  xmlns:fault="http://com/ibm/j2ca/fault/afcfault"
     name="JDBCOutboundInterface.wsdl"
     targetNamespace="http://JDBCXA/JDBCOutboundInterface">
  <types>
<xsd:schema
  xmlns:tns="http://JDBCXA/JDBCOutboundInterface"
  xmlns:xsd1="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtassercustomerbg"
  xmlns:xsd2="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtassercustomercontainer"
  elementFormDefault="qualified"
  targetNamespace="http://JDBCXA/JDBCOutboundInterface"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:import
   namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtassercustomerbg"
   schemaLocation="RtasserCustomerBG.xsd"/>
<xsd:import
   namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtasseracustomercontainer"
   schemaLocation="RtasserCustomerContainer.xsd"/>
```

ステップ 3b

```
<xsd:import namespace="http://com/ibm/j2ca/fault/afcfault"
   schemaLocation="MultipleMatchingRecordsFault.xsd"/>
<xsd:import namespace="http://com/ibm/j2ca/fault/afcfault"
   schemaLocation="RecordNotFoundFault.xsd"/>

. . .
```

ステップ 3c (178 ページ)

```
<xsd:element name="multipleMatchingRecordsFaultX">
 <xsd:complexType>
  <xsd:sequence>
  <xsd:element name="multipleMatchingRecordsFaultElement"
       type="fault:MultipleMatchingRecordsFault"/>
   </xsd:sequence>
 </xsd:complexType>
</xsd:element>

<xsd:element name="recordNotFoundFaultX">
  <xsd:complexType>
   <xsd:sequence>
   <xsd:element name="recordNotFoundFaultElement"
        type="fault:RecordNotFoundFault"/>
    </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
</types>

. . .
```

ステップ 3d (178 ページ)

```
<message name="multipleMatchingRecordsFault">
  <part element="intf:multipleMatchingRecordsFaultX"
        name="multipleMatchingRecordsFaultPart"/>
</message>
<message name="recordNotFoundFault">
  <part element="intf:recordNotFoundFaultX"
        name="recordNotFoundFaultPart"/>
</message>
<portType name="JDBCOutboundInterface">

. . .

<operation name="retrieveRtasserCustomerBG">
<input message="intf:retrieveRtasserCustomerBGRequest"
       name="retrieveRtasserCustomerBGRequest"/>
<output message="intf:retrieveRtasserCustomerBGResponse"
        name="retrieveRtasserCustomerBGResponse"/>
```

ステップ 3e (178 ページ)

```
<fault message="intf:multipleMatchingRecordsFault"
     name="multipleMatchingRecordsFaultFault" />
<fault message="intf:recordNotFoundFault"
     name="recordNotFoundFaultFault" />
</operation>
</portType>
</definitions>
```

XAResourceNotAvailableException

com.ibm.ws.Transaction.XAResourceNotAvailableException 例外の報告がプロセス・サーバーのログに繰り返し含まれているときは、トランザクション・ログを除去し、問題を訂正してください。

症状:

アダプターが始動すると、プロセス・サーバーのログ・ファイルに以下の例外が繰り返し記録されます。

com.ibm.ws.Transaction.XAResourceNotAvailableException

問題:

プロセス・サーバーがリソースのトランザクションをコミットまたはロールバックしている間に、そのリソースが除去されました。アダプターは、始動するとトランザクションのリカバリーを試みますが、リソースが除去されているため、それができません。

解決策:

この問題を訂正するには、以下の手順を実行します。

1. プロセス・サーバーを停止します。
2. そのトランザクションを含むトランザクション・ログ・ファイルを削除します。例外トレース内の情報を使用して、トランザクションを識別します。これにより、サーバーは、それらのトランザクションのリカバリーを試みないようになります。

   注: テスト環境または開発環境では、通常はトランザクション・ログをすべて削除できます。 WebSphere Integration Developer では、トランザクション・ログ・ディレクトリー *server_install_directory*¥profiles¥*profile_name*¥tranlog に含まれるファイルとサブディレクトリーを削除します。

   実稼働環境では、処理する必要のないイベントを表すトランザクションのみを削除します。これを行う方法の一つは、アダプターを再インストールし、使用した元のイベント・データベースをそのアダプターに参照させ、不要なトランザクションのみを削除することです。もう一つの方法は、以下のディレクトリー内の log1 ファイルまたは log2 ファイルからトランザクションを削除することです。

   server_install_directory¥profiles¥*profile_name*¥tranlog¥*node_name*¥wps¥*server_name*¥transaction¥tranlog
3. プロセス・サーバーを始動します。

セルフ・ヘルプ・リソース

IBM ソフトウェア・サポートのリソースは、最新のサポート情報やテクニカル文書を入手したり、サポート・ツールやフィックスをダウンロードしたり、 WebSphere Adapters の問題を回避したりするために使用することができます。また、セルフ・ヘルプ・リソースは、アダプターに関連する問題を診断するのに役立ち、IBM ソフトウェア・サポートへの連絡方法についての情報を提供します。

サポート Web サイト

WebSphere Adapters ソフトウェアのサポート Web サイト (http://www.ibm.com/software/integration/wbiadapters/support/) では、WebSphere Adapters の学習、使用、およびトラブルシューティングに役立つ多数のリソースへのリンクを提供しています。以下の種類のリソースがあります。

- フラッシュ (製品に関する警告)
- 製品のインフォメーション・センター、マニュアル、IBM Redbooks®、およびホワイト・ペーパーなどの技術情報。
- 教育関連のオファリング
- テクニカル・ノート

推奨フィックス

適用することが望ましい推奨フィックスのリストは、http://www.ibm.com/support/docview.wss?fdoc=aimadp&rs=695&uid=swg27010397にあります。

テクニカル・ノート

テクニカル・ノートは、Adapter for JDBC に関する最新の資料を提供します。以下のトピックがあります。

- 問題とそれに対する現在使用可能な解決策
- よくある質問に対する答え
- アダプターのインストール、構成、使用法、トラブルシューティングに関する手引きとなる情報
- *IBM* ソフトウェア・サポート・ハンドブック

WebSphere Adapters のテクニカル・ノートのリストについては、以下のアドレスにアクセスしてください。

http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm

IBM Support Assistant のプラグイン

Adapter for JDBC では、IBM Support Assistant のプラグインを提供します。これは、無料の保守容易性ローカル・ソフトウェア・ワークベンチです。IBM Support Assistant のインストールおよび使用については、以下のアドレスにアクセスしてください。

http://www.ibm.com/software/support/isa/

一般的な問題の解決策

ご使用のデータベースで WebSphere Adapter for JDBC を実行するときに発生する可能性のある問題の一部をその解決策および予備手段と共に説明します。これらの問題および解決策は、ソフトウェア・サポート Web サイトの技術情報として文書化されている問題や解決策を補足するものです。

WebSphere Adaptersについての技術情報の詳細なリストについては、http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm を参照してください。

テスト・クライアントでの RetrieveAll 操作の RecordNotFoundException

問題

WebSphere Integration Developer テスト・クライアントで RetrieveAll 操作を実行するときには、照会からのデータが予期されると RecordNotFoundException 例外が生成されます。次のメッセージが生成されます。`RecordNotFoundException: EIS にレコードが見つかりません (RecordNotFoundException: Record not found in EIS)`

原因

この例外は、SELECT ステートメントの WHERE 節にビジネス・オブジェクトの属性すべてが設定されていない場合に発生する可能性があります。属性を空白 (デフォルト値) にすることは、値を明示的に設定解除することと同じではありません。

解決策

テスト・クライアントで、必要とされる属性の値を <unset> に設定します。RetrieveAll 操作を繰り返します。再び例外が生成される場合は、データベース表に一致するレコードが存在しないと考えられます。

Oracle 9i または 10g データベースに 4K 以上の CLOB データ型を挿入できない

問題

4K 以上の CLOB (文字ラージ・オブジェクト) 値を Oracle 9i または 10g データベースに挿入しようとすると、次の例外が生成されます。

- Oracle 9i: ResourceAdapt E com.ibm.j2ca.jdbc.JDBCDBOperationHandler executePreparedCUDStatement CWYBC0301E: データベースでの操作が SQL 例外で失敗しました。失敗の理由は ソケットから読み取るデータがありません (No more data to read from socket) です。
- Oracle 10g: ResourceAdapt E com.ibm.j2ca.jdbc.JDBCDBOperationHandler executePreparedCUDStatement CWYBC0301E: データベースでの操作が SQL 例外で失敗しました。失敗の理由は ORA-01460: インプリメントされていないまたは不当な変換が要求されました (ORA-01460: unimplemented or unreasonable conversion requested) です。

原因

4K を超える CLOB を正しくサポートしない古いドライバーを使用しています。

解決策

Oracle 10.1.0.2 以降のリリースの Oracle シン・ドライバーを使用してください。

生成された一部のビジネス・オブジェクトには Oracle データベース・オブジェクトの属性がない

問題

Oracle データベース・オブジェクトから生成される一部のビジネス・オブジェクトに関して、生成されたビジネス・オブジェクトにテーブル列の属性がありません。

原因

特定の条件下では、Oracle JDBC ドライバーはデータベース・オブジェクトの列情報を返しません。この問題については、以下のバグを現在 Oracle に提出しています。

- 2281705。シノニムがある場合、DATABASEMETADA.GETCOLUMNS は基盤となるテーブルを返さない

- 2696213。JDBC GETPROCEDURECOLUMNS はプロシージャーのシノニムの列を返さない

さらに、他のスキーマ内のオブジェクトを参照するプライベート・シノニムが使用されている場合は、列情報が返されません。

解決策

シノニムを持つテーブルの場合は、テーブルのシノニムを使用してビジネス・オブジェクトを生成します。

プロシージャーのシノニムの場合は、シノニムの基盤となるオリジナルのプロシージャーを使用してビジネス・オブジェクトを生成します。

他のスキーマ内のオブジェクトを参照するプライベート・シノニムの場合は、オリジナルのテーブルを使用するか、現在のスキーマにシノニムを作成します。

Outbound 処理中に ResourceException 例外が発生する

ResourceException 例外を受け取った場合は、根本原因のフィールドを調べて原因を判別します。よくある問題の根本原因を以下に示します。

- SQLException 例外

  SQLException 例外にテキスト `UserID or password is invalid` が含まれている場合は、Outbound 接続に指定されたユーザー ID またはパスワードが正しくありません。

  例えば、次のように表示されます。

  ```
  javax.resource.ResourceException: [ibm][db2][jcc][t4][2013][11249] Connection
  authorization failure occurred.  Reason: User ID or Password invalid.
  ```

- ConnectException 例外

  ConnectException 例外のテキストに `connection refused` または `could not establish connection to the server` のようなテキストが含まれている場合は、データベース・サーバーが作動可能でないか、接続を妨げるネットワーク上の問題が存在する可能性があります。

  例えば、次のように表示されます。

  ```
  javax.resource.ResourceException: [ibm][db2][jcc][t4][2043][11550] Exception
  java.net.ConnectException: Error opening socket to server /9.26.237.55 on port
  50,000 with message: Connection refused: connect.
  ```

Inbound 処理中に ResourceException 例外が発生する

この例外は、データベース接続時に繰り返される問題があることを示します。イベントをポーリングするには、アダプターはデータベースに接続する必要があります。接続が失敗した場合、アダプターは、一定の時間 (この値は構成可能) 待機してから、接続を再試行します。再試行が一定の回数 (この値は構成可能) になると、アダプターはポーリングを停止します。アダプターはポーリングを停止すると、ResourceException 例外を生成します。

UniqueConstraintViolation フォールト、MultiMatchingRecordsException フォールト

外部サービス・ウィザードの始動時にクラス・ローダー違反が発生する

問題

データ・パースペクティブでデータベースへの接続を使用した後に、外部サービス・ウィザードを使用することができません。ウィザードの 2 番目のパネルの最後に、例外

com.ibm.adapter.framework.api.ImportException が生成されます。
理由: pc:0 でクラス・ロードの制約に違反がありました
(クラス: oracle/jdbc/driver/OracleConnection
メソッド: getWrapper()Loracle/jdbc/OracleConnection;)。

このエラーは、以下の状態のどちらでも発生します。

- 外部サービス・ウィザードを通じてデータベースへの接続を確立すると、データ・パースペクティブからデータベースに接続しようとしたときにエラーが発生します。
- データ・パースペクティブを通じてデータベースへの接続を確立すると、外部サービス・ウィザードからデータベースに接続しようとしたときにエラーが発生します。

原因

データ・パースペクティブとウィザードは独自のクラス・ローダーを使用するため、エラーが発生します。DLL (JDBC ドライバーが使用するネイティブ・ライブラリー) は、データ・パースペクティブにいったんロードすると、ウィザードに再度ロードできなくなります。JVM には、どの時点でも 1 つのクラス・ローダーのみがネイティブ・ライブラリーをロード可能であるという、固有の制約事項があります。そのため、クラス・ローダー A が DLL B をロードした場合、クラス・ローダー A が解放され、ガーベッジ・コレクションが実行されるまで、その他のクラス・ローダーは DLL B をロードできません。実際にはユーザーはガーベッジ・コレクションを制御できないため、通常、別のクラス・ローダーが DLL B をロードするようにしたい場合は、JVM を再始動する必要があるということになります。この制限は、既知のものであり、WebSphere Application Server では文書化されています。

解決策

このエラーが発生したときは、WebSphere Integration Developer を再始動することが唯一の解決策です。

Oracle 10g と共に XA を使用すると接続のクローズのエラーが発生する

問題

Oracle 10g を使用して XA トランザクションを実行するために Adapter for JDBC が使用されると、アダプターは Closed Connection (接続のクローズ) 例外 javax.resource.ResourceException: Closed Connection を生成します。

原因

これは、Oracle 10g データベース・ドライバーの既知の問題です。この問題については、Oracle で「3488761 Connection closed error from OracleConnection.getConnection() - 10G drivers」というバグが記録されています。

解決策

このバグは、Oracle 10g リリース 2 ドライバーで修正されています。予備手段としては、Oracle 9i JDBC Thin ドライバーを使用して、XA トランザクションのためにデータベースに接続します。

Oracle でトランザクションを開始中にエラーが発生する

問題

Adapter for JDBCを使用して、Oracle データベースを使用する XA トランザクションを実行すると、次のエラーが生成されます。 (WTRN0078E: トランザクション・マネージャーがトランザクションのリソースで start を呼び出そうとして、エラーが発生しました。エラー・コードは XAER_RMERR でした。(WTRN0078E: An attempt by the transaction manager to call start on a transactional resource has resulted in an error。The error code was XAER_RMERR.)

原因

Oracle データベース・サーバーで XA トランザクションをサポートするには、いくつかのコマンドを実行する必要があります。

解決策

Oracle ディレクトリーに含まれる 2 つのスクリプトを実行する必要があります。これらのスクリプトを実行するために必要な許可を持つためには、SYSOPER または SYSDBA として Oracle にログインしていなければならないため、おそらくこのアクティビティーは、Oracle データベース管理者が実行する必要があります。スクリプトは次のとおりです。

```
<ORACLE_HOME>javavm¥install
file: initxa.sql
file: initjvm.sql
```

initxa.sql スクリプトは、データベースを XA 用に構成します。正常に実行されると、データベースは XA 向けに構成されます。このスクリプトは、一度目の試行で正常に実行される場合もあります。ただし、データベースのメモリー・スペースの一部が小さすぎるために、正常に実行されない可能性が高くなっています。

これを修正するには、initjvm.sql スクリプトを実行します。このスクリプトもおそらく失敗しますが、失敗したときに、どのパラメーターを調整する必要があるかが示されます。パラメーターは、次のファイルに格納されます。

```
<ORACLE_HOME>¥database
file: init<DATABASE_SID>.ora
```

表 14に、通常増加させる必要のある 2 つのパラメーターを示します。ご使用の特定のデータベース構成では、異なるパラメーターの調整が必要となる可能性があります。

表 14. 標準的なパラメーター・サイズ

パラメーター名	最小値
java_pool_size	12000000
shared_pool_size	24000000

アダプターを使用して JDBC (タイプ 2 またはタイプ 4) ユニバーサル・ドライバーにより IBM DB2 for z/OS に接続する

問題と原因

DB2 for z/OS は、位置インデックスをデフォルトで使用し、列名を使用しないことによって、Adapter for JDBCが使用するストアード・プロシージャー・メタデータの照会をサポートします。解決策では、z/OS プラットフォームで DB2 と共に Adapter for JDBC を使用する手順を示しています。

解決策

Adapter for JDBCを使用して DB2 for z/OS に接続するには、以下の接続要件が満たされていることを確認してください。

- ユニバーサル JDBC ドライバーの物理表現は、db2jcc.jar ファイルです。このファイルへのパスが、クラス・パスに設定されていることを確認してください。
- データベース URL: タイプ 2 またはタイプ 4 のどちらのドライバーを使用するかを決定するには、接続の形式を検討します。

  タイプ 2: jdbc:db2:database
  (例: jdbc:db2:MyDB。MyDB はデータベース名。)

  タイプ 4: jdbc:db2://server:port/database
  (例: jdbc:db2://9.182.15.129:50000/MyDB。MyDB はデータベース名。)
- ドライバー・クラス: com.ibm.db2.jcc.DB2Driver。

  タイプ 2 ドライバーとタイプ 4 ドライバーの両方で、ドライバー・クラスは同一です。
- クラス・パスに db2jcc_license_cisuz.jar ファイルのパスを設定します。

  タイプ 2 ドライバーとタイプ 4 ドライバーの両方で、ライセンス JAR ファイルは同一です。DB2 for z/OS サーバーおよび DB2 for i5/OS サーバーにアクセスするには、有効な DB2 Connect™ のライセンスが必要です。DB2 クライアントは、DB2 Connect ライセンスがなければ、zSeries® サーバーおよび iSeries® サーバーへの直接接続を提供しません。

  DB2 Connect のライセンス交付および使用法については、以下のページを参照してください。

http://www-128.ibm.com/developerworks/db2/library/techarticle/0303zikopoulos1/0303zikopoulos1.html
http://www-128.ibm.com/developerworks/db2/library/techarticle/0301zikopoulos/0301zikopoulos.html

ウィザードを使用することによるストアード・プロシージャーのメタデータのインポートでは、問題が発生する可能性があります。Adapter for JDBC を使用して、ストアード・プロシージャーを使用し、DB2 からメタデータをインポートするには、DB2 を以下のステップで説明するように再構成する必要があります。前述のステップに加えて、以下のステップを実行してください。

- DB2 に APAR の PQ62695、PQ55393、PQ56616、PQ54605、PQ46183、および PQ62139 を適用します。
- アダプターでストアード・プロシージャーを使用したい場合は、下記のステップを実行します。これは、PQ62695 のフィックスの一部です。このフィックスは、JDBC および ODBC 仕様で文書化されているスキーマ・メタデータ API に対応する結果セットを生成することが可能なストアード・プロシージャーを導入しています。

  これらのプロシージャーは、DB2 Universal Driver で提供される JDBC および ODBC ドライバーが使用します。以下のステップを実行して、ストアード・プロシージャーのサポートを使用可能にします。

  1. APAR を適用します。
  2. ZPARM アセンブリー・ジョブ DSNTIJUZ 内の DESCSTAT 変数の値を確認します。 DESCSTAT 変数の値が NO である場合は、YES に変更します。

     注: DESCSTAT のデフォルトは、V7 では NO ですが、V8 では YES に変更されました。
  3. ZPARM モジュールを再アセンブルし、再初期化します。
  4. DSNTIJMS という名前の JCL ジョブを実行します。このメンバーは、db2prefix.SDSNSAMP データ・セット内にあります。
  5. DB2 を再始動します。

リモート DB2 データベースを用いたアウトバウンド・サポートのための XA トランザクションの使用

これにより、リモート DB2 データベースと共に Adapter for JDBC を使用する XA トランザクション・サポートのための手順、データベース・バージョン、および構成要件が提供されます。

Universal Driver を使用した Adapter for JDBC での XA トランザクションの使用

Adapter for JDBC および Universal ドライバーで XA トランザクションを使用してリモート DB2 データベースに接続するには、以下のバージョンのソフトウェアおよび構成プロパティーが必要です。

- DB2 バージョン: 8.2 以降
- JDBC ドライバー: UDB ドライバー (db2jcc.jar) タイプ 4
- XA データ・ソース名: com.ibm.db2.jcc.DB2XADataSource

- XA データベース名: これは、ローカル DB2 クライアントで構成されたリモート・データベース別名です。
- データベース URL: jdbc:db2://hostname:port/databasename
- JDBC ドライバー・クラス: com.ibm.db2.jcc.DB2Driver

リモート DB2 データベースでの XA トランザクションの使用

リモート DB2 データベースの追加

1. DB2 サーバーで db2admin (*DB2_InstallPath*¥SQLLIB¥BIN) コマンドを実行します。
2. DB2 構成アシスタントを開きます。
3. 「**表示 (View)**」 → 「**拡張表示 (Advanced View)**」に移動します。

次のステップを順番に実行してください。

1. **リモート・システムの追加**
   a. 「**システム**」タブを選択します。
   b. メニューから、「**選択済み**」 → 「**システムの追加 (Add System)**」を選択します。
   c. 「**システム名 (System name)**」フィールドで、ターゲット・データベースが配置されている物理マシン、サーバーシステム、またはワークステーションを指定します。サーバー・システム上のシステム名は、DB2SYSTEM DAS 構成パラメーターによって定義されます。ユーザーはこの値を使用する必要があります。
   d. 「**ホスト名**」フィールドに、ホスト名か、またはターゲット・データベースが存在するインターネット・プロトコル (IP) アドレスを入力します。
   e. 「**ノード名 (Node name)**」フィールドで、データベースが配置されているリモート・ノードのローカル・ニックネームを指定します。選択するノード名は、ノード・ディレクトリーまたは管理ノード・ディレクトリーに存在していないものにする必要があります。
   f. オペレーティング・システムを選択して、「**OK**」をクリックします。
2. **インスタンス・ノードの追加**
   a. 「**インスタンス・ノード (Instance Nodes)**」タブを選択します。
   b. メニューから、「**選択済み**」 → 「**インスタンス・ノードの追加 (Add Instance Node)**」を選択します。
   c. 「**システム名 (System name)**」フィールドで、ターゲット・データベースが配置されている物理マシン、サーバーシステム、またはワークステーションを指定します。リモート・システムの追加タスクで追加したシステムを選択します。
   d. 「**インスタンス名**」フィールドに、ターゲット・データベースが配置されているインスタンスの名前 (DB2 など) を入力します。
   e. 「**インスタンス・ノード名 (Instance node name)**」フィールドで、データベースが配置されているカタログされたシステム (ノード) の固有のニックネームを指定します。選択するノード名は、ノード・ディレクトリーまたは管理ノード・ディレクトリーに存在していないものにする必要があります。

f. オペレーティング・システムを選択して、ホスト名を入力します。リモート・システムの追加タスクのステップ 4 と同じホスト名を使用します。

g. リモート DB2 インスタンスが実行されているポート番号を入力します。

h. 「**OK**」をクリックします。

3. **データベースの追加**

a. 「**データベース**」タブを選択します。

b. メニューから、「**選択済み**」 → 「**データベースの追加**」を選択します。

c. 「**インスタンス・ノード (Instance node)**」フィールドで、インスタンス・ノードの追加タスクで作成したインスタンスを選択します。「**データベース名 (Database name)**」フィールドで追加するデータベースの名前を指定します。

d. 「**別名 (Alias)**」フィールドで、ワークステーション上で実行中のアプリケーションが使用できるローカル・ニックネームを指定します。何も入力されない場合、別名はデータベース名と同じになります。別名は、固有でなければなりません。

注: この別名値を、アダプターの XADatabaseName プロパティーに入力する必要があります。

4. **データベース接続のテスト**

a. 「**データベース**」タブを選択します。

b. データベースの追加タスクで追加されたデータベースを選択します。

c. メニューから、「**選択済み**」 → 「**接続のテスト**」を選択します。

d. 「**CLI**」チェック・ボックスを選択し、ユーザー ID およびパスワードを入力し、「**接続のテスト**」をクリックします。これにより、接続の成功が戻されます。

イベント・テーブルのトランザクション (XID) 列を調べる

アダプターが送達は 1 回のみとして構成されている場合は、XID 列と共に状況列を使用して、イベントが処理されたかどうかを判別します。

- XID 列に 0 が含まれている場合、イベントはまだ処理対象として選出されていません。
- XID 列にトランザクション ID が含まれている (つまり 0 ではない) 場合、アダプターはイベントの処理を開始しましたが、処理は完了していません。イベントの処理中にアダプターまたはアプリケーション・サーバーが異常終了した場合は、この組み合わせが表示される可能性があります。トランザクション・マネージャーはリカバリー時にこれらのトランザクションを COMMIT または ROLLBACK します。

第 8 章 参照

ビジネス・オブジェクト、アダプター・プロパティー (エンタープライズ・サービス・ディスカバリー・プロパティー、リソース・アダプター・プロパティー、管理 (J2C) 接続ファクトリー・プロパティー、アクティベーション・スペック・プロパティー、および対話仕様プロパティー)、メッセージ、および関連製品情報に関する詳細情報は参照用に提供されます。

ビジネス・オブジェクト情報

ビジネス・オブジェクトは、アダプターによるビジネス・オブジェクトの処理方法、およびビジネス・オブジェクトに対して実行される操作に関するアプリケーション固有情報 (メタデータ) を格納する構造です。ビジネス・オブジェクトの名前は、アダプターの命名規則に従って、外部サービス・ウィザードが生成します。

ビジネス・オブジェクト属性

ビジネス・オブジェクト属性は、ビジネス・オブジェクトの内容を定義するものであり、データベース・オブジェクトの列リストから作成されます。各属性は、名前、型、カーディナリティーなどのプロパティーを持ちます。外部サービス・ウィザードで、列名に属性名が設定されます。アダプターでは、属性のカーディナリティー、型、およびアプリケーション固有情報が追加されます。

ビジネス・オブジェクトは、属性で指定されるデータの単なるコンテナーです。データベースのデータの構造はビジネス・オブジェクトによって定義されますが、データベースのデータはビジネス・オブジェクト属性内にあります。

表 15 に、ビジネス・オブジェクト属性のプロパティーをリストし、それらの解釈および設定値について説明します。

表 15. 属性プロパティー

プロパティー	解釈と設定値
Cardinality	ビジネス・オブジェクトのカーディナリティーを示す整数。1 つの子ビジネス・オブジェクトまたは複数の子ビジネス・オブジェクトを表すビジネス・オブジェクト属性はそれぞれ、カーディナリティーの値が単一 (1) または複数 (制限のない整数) になります。 単一カーディナリティー関係および複数カーディナリティー関係の両方で、親ビジネス・オブジェクトと子ビジネス・オブジェクトの間の関係は、その関係を保管するビジネス・オブジェクトのキー属性に含まれるアプリケーション固有情報によって記述されます。
Foreign Key	カーディナリティーが *n* の子ビジネス・オブジェクトの配列が検索されると、SELECT ステートメントの WHERE 文節で外部キーが使用されます。 RetrieveAll 操作は、キーおよび外部キーの使用を指定変更します。**注:** アダプターでは、子ビジネス・オブジェクトを表す属性を外部キーとして指定することについては、サポートしていません。

表 15. 属性プロパティー (続き)

プロパティー	解釈と設定値
Name	このプロパティーは、属性が単純属性の場合は、属性の固有の名前。属性が子ビジネス・オブジェクトの場合は、ビジネス・オブジェクトの名前を表します。
MinOccurs MaxOccurs	列が基本キーではなく、かつ NULL 値を取れない場合、MinOccurs および MaxOccurs 属性は必須であり、値は 1 以上に設定されます。
Primary Key	この属性が基本キーかどうかを示します。 どのビジネス・オブジェクトでも、1 つ以上の単純属性が基本キーに指定されなければなりません。 単純属性の基本キー・プロパティーを `true` に設定すると、アダプターは、ビジネス・オブジェクトの処理中に生成する SELECT および SQL UPDATE の各ステートメントの WHERE 文節にその属性を追加します。RetrieveAll 操作は、基本キーおよび外部キーの使用を指定変更します。 **注:** アダプターでは、子ビジネス・オブジェクトまたは子ビジネス・オブジェクトの配列を表す属性を基本キー属性として指定することについてはサポートしていません。
必須	属性が値を含む必要があるかどうかを指定します。カーディナリティーが単一 (1) のコンテナーに対して、このプロパティーが `true` に設定されている場合、アダプターでは、その親ビジネス・オブジェクトが、この属性に対応する子ビジネス・オブジェクトを含んでいる必要があります。Create、Update、および Delete 操作でアダプターに渡されるビジネス・オブジェクトは、子ビジネス・オブジェクトも含んでいなければなりません。単純属性のカーディナリティーは単一 (1) で、コンテナー属性のカーディナリティーは複数 (n) です。ビジネス・オブジェクトが必須属性に対して有効な値またはデフォルト値を持っていないと、アダプターでは Create 操作が失敗します。このオブジェクトに対するデータベースからの検索時に使用可能なデータがない場合も、Create 操作は失敗します。
型	単純属性の場合、このプロパティーは属性の型 (Integer、String、Date、Timestamp、Boolean、Double、Float など) を指定します。サポートされる単純属性の型と、それらがマップされるデータベース・オブジェクトの JDBC タイプを 193 ページの表 16 に示します。 子ビジネス・オブジェクトを指定する属性の場合、このプロパティーはビジネス・オブジェクトの名前を指定します。

JDBC メタデータとして戻される各データベース・オブジェクトのタイプは、 193 ページの表 16 のリストにあるようにビジネス・オブジェクト属性タイプにマップされます。リストされている JDBC タイプのみがアダプターでサポートされます。リストされていないタイプの列は、ビジネス・オブジェクトに追加されません。その場合は、問題を説明する通知メッセージが生成されます (例: テーブル *yyyy* の列名 *xxxx* のタイプはサポートされていません。ビジネス・オブジェクトには追加されません)。

表 16. JDBC メタデータ列タイプとビジネス・オブジェクト属性タイプ

JDBC メタデータの列タイプ	ビジネス・オブジェクト属性のタイプ
BIT	Boolean
CHAR LONGVARCHAR VARCHAR	String
INTEGER NUMERIC SMALLINT TINYINT BIGINT	Int
TIME TIMESTAMP DATE	String
DECIMAL	String
DOUBLE FLOAT	Double
REAL	Float
BLOB	hexBinary
CLOB	String
BINARY VARBINARY LONGBINARY	hexBinary
NCHAR NVARCHAR NTEXT	String
TEXT	String
RAW	String
MONEY SMALLMONEY	String

属性に関するアプリケーション固有情報

ビジネス・オブジェクト属性のアプリケーション固有情報 (ASI) は、属性が単純属性であるか、子ビジネス・オブジェクト (または子ビジネス・オブジェクトの配列) を表す属性であるかによって異なります。子を表す属性のアプリケーション固有情報は、親子関係が子に格納されるか親に格納されるかによっても異なります。

単純属性のアプリケーション固有情報

単純属性では、アプリケーション固有情報の形式は、いくつかのパラメーターとその値で構成されています。単純属性に必要な唯一のパラメーターは列名です。単純属性のアプリケーション固有情報については、 194 ページの表 17で説明します。

表 17. 単純属性のアプリケーション固有情報

パラメーター	型	説明	デフォルト値
BLOB	Boolean	この属性に対応するデータベース列が BLOB データ型であるかどうかを示します。BLOB データの表示中、アダプターはバイト数を 16 進値で表示します。属性の型は hexBinary です。 True の場合は、列のデータ型は BLOB です。	なし
ByteArray	Boolean	列がバイナリー・データ型であるかどうかを示します。 True の場合、アダプターはデータベースでバイナリー・データを読み取りおよび書き込みし、そのデータをストリングとしてアプリケーション・サーバーに送信します。アダプターは、ビジネス・オブジェクトにバイナリー・データを設定します。属性の型は hexBinary です。	False
ChildBOType	String	属性が複合データ型の場合は、このアプリケーション固有情報を使用して実際の型を指定します。 • Struct • Array • ResultSet	なし
ChildBOTypeName	String	ChildBOType アプリケーション固有情報の値が Struct または Array の場合、このパラメーターの値はユーザー定義型の名前を表します。この値は大文字小文字が区別されます。	
CLOB	Boolean	この属性に対応するデータベース列が CLOB データ型であるかどうかを示します。この値は、String 型の属性にのみ適用されます。 True の場合は、列のデータ型は CLOB です。 CLOB 属性は String 型を持ち、この長さを使用して CLOB の長さを定義します。	なし
ColumnName	String	この属性に対応するデータベース列の名前。 これが唯一の必須パラメーターです。	なし

表 17. 単純属性のアプリケーション固有情報 (続き)

パラメーター	型	説明	デフォルト値
CopyAttribute	String	同一ビジネス・オブジェクトまたは親ビジネス・オブジェクト内から別の属性名を参照するユーザー指定値。 アプリケーション固有情報で設定されている値が、同一ビジネス・オブジェクト内の別の属性の名前を参照している場合、アダプターは、Create 操作でビジネス・オブジェクトをデータベースに追加する前に、その別の属性の値を使用してこの属性の値 (アプリケーション固有情報が定義されている属性) を設定します。 例えば、テーブルの新しい行の `contact` 列に、`email` 列と同じ値を組み込むには、`contact` 属性の CopyAttribute パラメーターに `email` を設定します。 値から子ビジネス・オブジェクトの属性を参照することはできませんが、親ビジネス・オブジェクト内の属性は参照できます。このためには、属性名の前に 2 つのピリオドを付加します。例えば、親ビジネス・オブジェクト内の `ccode` 属性を参照するには、`..ccode` と指定します。 アプリケーション固有情報にこのパラメーターを指定しないと、アダプターは、別の属性から値をコピーせずに現行属性の値を使用します。	なし
DateType	String	対応するエレメントが日付、時刻、またはタイム・スタンプであることを指定します。次の値のいずれかを指定してください。 • Date • Time • Timestamp DateType 型の属性の値を設定するときには、次の形式で設定します。 • Date には yyyy-MM-dd を使用します。 • Time には hh:mm:ss を使用します。 • Timestamp には yyyy-MM-dd hh:mm:ss を使用します。	なし

表 17. 単純属性のアプリケーション固有情報 (続き)

パラメーター	型	説明	デフォルト値
FixedChar	Boolean	テーブル内の列が VARCHAR 型ではなく CHAR 型である場合に、属性を固定長とするかどうかを指定します。例えば `true` に設定すると、ある特定の属性が CHAR 型の列にリンクされている場合、アダプターはデータベースを照会するときに、属性値をその属性の最大長までブランクで埋めます。 ビジネス・オブジェクトの XSD ファイルでは、このパラメーターは手動で更新する必要があります。WebSphere Integration Developer のビジネス・オブジェクト・エディターを使用して XSD ファイルを編集します。更新後は XSD ファイル内で検証エラーが発生していないことを確認してください。この表に続くこのパラメーターのコード例を参照してください。 XSD ファイルに FixedChar 型のアプリケーション固有情報を指定する例をこの表の後に示します。	`false`
ForeignKey	String	このプロパティーの値は、親子関係が親ビジネス・オブジェクトに格納されるか、子ビジネス・オブジェクトに格納されるかによって異なります。 関係が親に格納される場合、子ビジネス・オブジェクトのタイプと、外部キー *(Child_BO_name/Child_Property_Name)* として使用される子ビジネス・オブジェクト内の属性の名前の両方がこの値に含まれます。 この関係が子に保管される場合は、外部キーとして使用される親ビジネス・オブジェクト内の属性の名前のみを含むように値を設定します。 属性が外部キーではない場合は、アプリケーション固有情報にこのパラメーターを含めないでください。	なし
OrderBy	String	値が指定され、属性が子ビジネス・オブジェクト内に存在する場合は、アダプターは、検索照会の ORDER BY 文節でその属性の値を使用します。 アダプターは、子ビジネス・オブジェクトを昇順 (ASC) または降順 (DESC) で検索することができます。このパラメーターがアプリケーション固有情報に含まれていない場合、アダプターは検索順序を指定しません。	なし
PrimaryKey	Boolean	この属性に関連付けられている列が、データベース内の対応するテーブルの基本キーである場合、PrimaryKey は `True` に設定されます。	なし

表 17. 単純属性のアプリケーション固有情報 (続き)

パラメーター	型	説明	デフォルト値
SPParameterType	String	ストアード・プロシージャーのタイプを指定します。 使用可能な値は、以下のとおりです。 • IP (入力のみ) • OP (出力のみ) • IO (入出力) • RS (結果セット)	なし
UniqueIdentifier (UID)	String	アダプターは、ビジネス・オブジェクトの固有 ID の生成に、このパラメーターを使用します。シーケンスと ID 列の生成がサポートされます。 このパラメーターの形式を以下に示します。 UID=AUTO|*Sequence_Name* シーケンスの場合、UID 属性にシーケンス名を設定します。シーケンスを定義できるのは、DB2 および Oracle データベースのみです。 ID 列の場合、UID 属性に AUTO を設定します。 ID 列を定義できるのは、DB2 および Microsoft SQL Server です。 属性で固有 ID が必要とされない場合は、このパラメーターをアプリケーション固有情報に含めないでください。	なし

属性のアプリケーション固有情報の形式は、XSDファイルの以下の実例セクションに示します。

XSD ファイルの例

```
   <jdbcasi:ColumnName>pkey</jdbcasi:ColumnName>
   <jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
   <jdbcasi:FixedChar>true</jdbcasi:FixedChar>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
<simpleType>
   <restriction base="string">
   <maxLength value="10"/>
   </restriction>
</simpleType>
</element>
<element name="custCode" type="string">
<annotation>
<appinfo source="WBI">
<jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:ColumnName>ccode</jdbcasi:ColumnName>
   <jdbcasi:ForeignKey>custinfoObj/custCode</jdbcasi:ForeignKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="firstName" type="string">
<annotation>
<appinfo source="WBI">
<jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:ColumnName>fname</jdbcasi:ColumnName>
```

```
        </jdbcasi:JDBCAttributeTypeMetadata>
        </appinfo>
        </annotation>
        </element>
        <element name="lastName" type="string">
        <annotation>
        <appinfo source="WBI">
        <jdbcasi:JDBCAttributeTypeMetadata
        xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
           <jdbcasi:ColumnName>lname</jdbcasi:ColumnName>
        </jdbcasi:JDBCAttributeTypeMetadata>
        </appinfo>
        </annotation>
        </element>
```

ビジネス・オブジェクトの XSD ファイル内の FixedChar パラメーターの例

```
<element name="primaryKey">
<annotation>
<appinfo source="WBI">
        <jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
                <jdbcasi:ColumnName>pkey</jdbcasi:ColumnName>
                <jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
                <jdbcasi:FixedChar>true</jdbcasi:FixedChar>
        </jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
<simpleType>
        <restriction base="string">
                <maxLength value="10"/>
        </restriction>
</simpleType>
</element>
```

型が子ビジネス・オブジェクトの属性のアプリケーション固有情報

子ビジネス・オブジェクトを参照する属性には、2 つのアプリケーション固有情報パラメーターが使用されます (単純属性ではなく複合属性)。このアプリケーション固有情報を設定する場合は、表 18に示すパラメーターを指定します。

表 18. 型が子ビジネス・オブジェクトの属性のアプリケーション固有情報

パラメーター	型	説明	デフォルト値
KeepRelationship	Boolean	True の場合、このパラメーターは Update 操作中に子ビジネス・オブジェクトを削除しないようにします。	なし
Ownership	Boolean	このパラメーターは、子ビジネス・オブジェクトが親によって所有されることを指定します。True の場合、子ビジネス・オブジェクトに対する Create、 Update、および Delete 操作が許可されます。False の場合、どの更新も子ビジネス・オブジェクトには適用できません。親が作成されると、データベース内で関係の整合性が保持されるように、子が存在するかどうかが検証されます。	なし

ビジネス・オブジェクトの XSD ファイル内の ownership の例

```
<element minOccurs="0" name="addressObj" type="bons0:OutboundRtasserAddress"
 maxOccurs="unbounded">
  <annotation>
    <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
    <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
         <jdbcasi:Ownership>true</jdbcasi:Ownership>
```

```
    </jdbcasi:JDBCAttributeTypeMetadata>
    </appinfo>
  </annotation>
</element>


<element minOccurs="0" name="custinfoObj" type="bons1:OutboundRtasserCustinfo"
maxOccurs="1">
 <annotation>
  <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
    <jdbcasi:Ownership>false</jdbcasi:Ownership>
   </jdbcasi:JDBCAttributeTypeMetadata>
  </appinfo>
 </annotation>
</element>
```

単一または複数カーディナリティー子ビジネス・オブジェクトの XSD 定義ファイルの例をここに示します。エレメント custInfoObj は単一カーディナリティー子ビジネス・オブジェクトで、addressObj は複数カーディナリティー子ビジネス・オブジェクトです。

単一または複数カーディナリティー子ビジネス・オブジェクトの XSD ファイルのもう 1 つの例

```
<element name="addressObj" minOccurs="1" type="Address:Address"
maxOccurs="unbounded">
                        <annotation>
                        <appinfo source="WBI">
                        <pasi:JDBCAttributeTypeMetadata xmlns:pasi=
"urn:app:jdbc:asi">
                                <pasi:Ownership>true</pasi:Ownership>
                        </pasi:JDBCAttributeTypeMetadata>
                        </appinfo>
                        </annotation>
                        </element>
                        <element name="custInfoObj" minOccurs="0" type=
"CustInfo:CustInfo" maxOccurs="1">
                        <annotation>
                        <appinfo source="WBI">
                        <pasi:JDBCAttributeTypeMetadata xmlns:pasi=
"urn:app:jdbc:asi">
                                <pasi:Ownership>false</pasi:Ownership>
                        </pasi:JDBCAttributeTypeMetadata>
                        </appinfo>
                        </annotation>
                        </element>
```

操作のアプリケーション固有情報

アダプターは、操作レベルのアプリケーション固有情報を使用してデータベース内の情報の取得や更新などの操作を実行します。アダプターは、ビジネス・オブジェクトでの指定に従って、SQL 照会、ストアード・プロシージャー、またはストアード関数を使用してデータベース表を取得および更新します。

ビジネス・オブジェクトにストアード・プロシージャーまたはストアード関数を追加する場合、操作レベルのアプリケーション固有情報 (ASI) を、200 ページの表 19 に指定されているとおりに設定します。

表 19. 操作に関するアプリケーション固有情報

操作 ASI の StoredProcedure のパラメーター・エレメント	ウィザードによって設定	説明
Parameters	はい	ストアード・プロシージャーのパラメーターをリストします。
PropertyName	はい	選択したビジネス・オブジェクト属性の名前に設定します。
ResultSet	いいえ	ストアード・プロシージャーから結果セットが返される場合は、ビジネス・オブジェクト定義でこのパラメーターを `True` に設定します。
ReturnValue	はい	ストアード・プロシージャーに戻り値がある場合は、このパラメーターには次のいずれかの値が設定されます。 • ストリング `RS`。この値は、プロシージャーから結果セットが戻され、この結果セットを使用して、このビジネス・オブジェクトに対応する複数カーディナリティー・コンテナーが作成されることを示します。 • ビジネス・オブジェクト属性の名前。この値は、プロシージャーから戻される値が、実行時にビジネス・オブジェクトの特定の属性に割り当てられることを示します。 属性が別の子ビジネス・オブジェクトである場合、アダプターはエラーを返します。
StoredProcedure	はい	ストアード・プロシージャー名に設定します。
StoredProcedureType	はい	タイプのリストから選択します。有効なストアード・プロシージャー・タイプについては、45 ページの『ストアード・プロシージャー・タイプ』のセクションを参照してください。
型	はい	ストアード・プロシージャーのパラメーターのタイプを設定します。使用可能な値は、以下のとおりです。 • IP (入力のみ) • OP (出力のみ) • IO (入出力) • RS (結果セット)

ビジネス・オブジェクト・レベルのアプリケーション固有情報

ビジネス・オブジェクト定義内のアプリケーション固有情報は、アダプターに対し、ビジネス・オブジェクトの処理方法に関するアプリケーション依存の指示を与えるものです。アダプターでは、ビジネス・オブジェクト、またはビジネス・オブジェクトの属性あるいは操作から取得したアプリケーション固有情報を解析して、Create、Update、Retrieve、および Delete 操作のための照会を生成します。

テーブルおよびビュー・ビジネス・オブジェクトのアプリケーション固有情報

ビジネス・オブジェクト・レベルのアプリケーション固有情報は、対応するデータベース表の名前を指定するとき、および物理削除または論理削除操作の実行に必要な情報を指定するときに使用されます。

外部サービス・ウィザードは、アプリケーション固有情報属性 TableName に、*SchemaName.TableName* という形式の値を設定します。物理/論理 Delete 操作の実行に必要な情報の入力を求めるプロンプトが出され、表 20 に示すビジネス・オブジェクト・レベルのアプリケーション固有情報が設定されます。

表 20. テーブル・ビジネス・オブジェクトのビジネス・オブジェクト・アプリケーション固有情報 (ASI)

アプリケーション固有情報	型	説明
TableName	String	このビジネス・オブジェクトに対応するデータベース表の名前。
ステータス列名	String	アダプターがテーブルのデータを論理的に削除するか、または物理的に削除するかを示します。 StatusColumnName パラメーターが設定されていない場合、データは物理的に削除されます。このパラメーターが設定されている場合は、論理的に削除された行を示す列の名前がパラメーターにより指定されます。外部サービス・ウィザードでテーブル・オブジェクトを選択するときに、このパラメーターを指定します。 このパラメーターは、Update 操作と Delete 操作の両方に適用されます。
ステータス値	String	列が論理的に削除されることを示す値。外部サービス・ウィザードでテーブル・オブジェクトを選択するときに、この値を指定します。

Update 操作または Delete 操作に対応して、アダプターが論理削除と物理削除操作のどちらを実行するかを判別する方法を示すため、Customer ビジネス・オブジェクトに、表 21に示すビジネス・オブジェクトのアプリケーション固有情報が含まれているとします。

表 21. テーブル・ビジネス・オブジェクトのビジネス・オブジェクト・アプリケーション固有情報のパラメーターの例

アプリケーション固有情報	値
TableName	customer
ステータス列名	status
ステータス値	deleted

アダプターが、カスタマーの削除要求を受信したとします。ビジネス・オブジェクトのアプリケーション固有情報に StatusColumnName パラメーターが含まれるため、アダプターは論理削除操作を実行します。このため、StatusValue パラメーター

に指定されているストリング「deleted」を、StatusColumnName パラメーターで指定されている列である status 列に挿入します。

このような要求により、アダプターは次の SQL ステートメントを発行します。

```
UPDATE customer set status = 'deleted' where pkey = . . . .
```

ただし、StausColumnName パラメーターが設定されていない場合は、顧客レコードは物理的に削除されます。アダプターは次の SQL ステートメントを実行します。

```
DELETE customer where pkey = . . . .
```

ストアード・プロシージャー・ビジネス・オブジェクトのアプリケーション固有情報

ストアード・プロシージャーに基づくビジネス・オブジェクトの場合、外部サービス・ウィザードでビジネス・オブジェクト・レベルのアプリケーション固有情報 SPName が *SchemaName* + *SPName* の形の値に設定されます。設定されるビジネス・オブジェクト・レベルのアプリケーション固有情報を表 22に示します。ビジネス・オブジェクトの属性は、ストアード・プロシージャーの入出力パラメーターに基づいて作成されます。ストアード・プロシージャーが 1 つの戻り値を持つ場合は、対応するビジネス・オブジェクト属性が作成されます。戻り値または入出力パラメーターに複合データ型がある場合、ウィザードによってそれらの子ビジネス・オブジェクトが作成されます。

外部サービス・ウィザードでのデータベース・オブジェクトのディスカバリーは、ネストされた構造体および配列をサポートできます。返される結果セットからこれらの子ビジネス・オブジェクトが生成される場合、それらの子ビジネス・オブジェクトの名前の形式は *Prefix* + *SchemaName* + *SPName* + `RetRS` + *Number* になります。例えば、あるストアード・プロシージャーが 2 つの結果セットを返す場合、ウィザードはそれらに対応する 2 つの子ビジネス・オブジェクトを作成します。それぞれの名前は、*Prefix* + *SchemaName* + *SPName* + `RetRS1` と *Prefix* + *SchemaName* + *SPName* + `RetRS2` です。

子ビジネス・オブジェクトが、複合データ型の ResultSet、Struct、または Array を持つ入出力パラメーターから生成される場合、これらの子ビジネス・オブジェクト名の形式は *Prefix+SchemaName+SPName+ParameterName* になります。ネストされた構造体および配列に対応する子ビジネス・オブジェクトの場合、ビジネス・オブジェクト名の形式は *Prefix+SchemaName+SPName+ParameterName+ColumnName*になります。

表 22. ストアード・プロシージャーに基づくビジネス・オブジェクトの場合のビジネス・オブジェクトのアプリケーション固有情報 (ASI)

アプリケーション固有情報	型	説明
SPName	String	ストアード・プロシージャーまたはストアード関数の名前
ResultSet	Boolean	ストアード・プロシージャーまたはストアード関数が結果セットを戻すかどうかを示します。`true` の場合は、ストアード・プロシージャーが 1 つ以上の結果セットを戻します。`false` の場合は、ストアード・プロシージャーまたはストアード関数は結果セットを戻しません。
MaxNumberOfRetRS	String	アダプター・ランタイムによって処理される返される結果セットの最大数

表 22. ストアード・プロシージャーに基づくビジネス・オブジェクトの場合のビジネス・オブジェクトのアプリケーション固有情報 (ASI) (続き)

アプリケーション固有情報	型	説明
ReturnValue	String	ストアード・プロシージャーに戻り値がある場合は、対応するビジネス・オブジェクト属性の名前に設定されます。戻り値が単純データ型の場合は、属性も単純データ型です。戻り値が結果セットの場合、この属性は子ビジネス・オブジェクトを指します。

クエリー・ビジネス・オブジェクトのアプリケーション固有情報

クエリー・ビジネス・オブジェクトには、ビジネス・オブジェクト・レベルのアプリケーション固有情報が 1 つあります。表 23 にこの情報を示します。

表 23. クエリー・ビジネス・オブジェクトのビジネス・オブジェクト・アプリケーション固有情報 (ASI)

アプリケーション固有情報	型	説明
SelectStatement	String	照会を実行する完全な SELECT ステートメント。外部サービス・ウィザードでこのステートメントを指定します。

また、外部サービス・ウィザードでは、すべてのビジネス・オブジェクトが最上位であるため、すべてのビジネス・オブジェクトのビジネス・グラフも生成されます。ビジネス・グラフの名前は、ビジネス・オブジェクト名に「BG」が付いたものです。例えば、JDBCSchema1Customer という名前のビジネス・オブジェクトのビジネス・グラフの名前は JDBCSchema1CustomerBG となります。ビジネス・オブジェクトに設定された操作はビジネス・グラフにも設定されます。

ウィザードは、ストアード・プロシージャー・ビジネス・オブジェクトを生成する場合に、必要であれば ResultSet、Struct、Array などの子ビジネス・オブジェクトを作成します。テーブル・ビジネス・オブジェクト間の親子関係は、Business Object Editor を使用して手動で作成します。

ウィザードは、同義語がストアード・プロシージャーのものであっても、同義語/ニックネームに基づくビジネス・オブジェクトもテーブルおよびビューに基づくオブジェクトのように処理します。

バッチ SQL ビジネス・オブジェクトのアプリケーション固有情報

バッチ SQL ビジネス・オブジェクトには、以下のアプリケーション固有情報が含まれています。

表 24. バッチ SQL ビジネス・オブジェクトのビジネス・オブジェクト・アプリケーション固有情報 (ASI)

アプリケーション固有情報	型	説明
BatchSQLIndex	String	SQL ステートメントの順序を指定します。例えば、ユーザーが 1 つのバッチ SQL ビジネス・オブジェクトに 3 つのステートメントを指定する場合 (セミコロン区切り)、最初のステートメントの索引が 1、2 番目の索引が 2、最後の索引が 3 となります。

SQLStatement	String	ユーザーに指定された 1 つの INSERT、UPDATE、および DELETE SQL ステートメントを含みます。ユーザーが 1 つのビジネス・オブジェクトに複数の SQL ステートメントを指定した場合、それぞれが別の SQLStatement エレメントに保管されます。例えば、次のようになります。
		
		`Delete From Customer where pkey=?`
		
		`Insert into customer (pkey,ccode,fname,lname values(?,?,?,?)`

ラッパー・ビジネス・オブジェクトのアプリケーション固有情報

ラッパー・ビジネス・オブジェクトの場合、ラッパー・アプリケーション固有情報が追加され、`True` に設定されます。ラッパー・ビジネス・オブジェクトのビジネス・オブジェクト・レベルでは、その他のアプリケーション固有情報は不要です。

アプリケーション固有情報	型	説明
Wrapper	Boolean	ビジネス・オブジェクトがラッパー・ビジネス・オブジェクトであるかどうかを示します。 Wrapper が `True` の場合、その他のビジネス・オブジェクト・レベル ASI は不要です。

命名規則

外部サービス・ウィザードでは、ビジネス・オブジェクトの生成時に、アダプターの命名規則に従った名前がビジネス・オブジェクトに指定されます。一般に、ビジネス・オブジェクト名はビジネス・オブジェクトの構造を示しています。

外部サービス・ウィザードは、ビジネス・オブジェクト名の作成時に、ビジネス・オブジェクト名のアンダースコアー (_) 以外の特殊文字を、`U` とその後にその文字の Unicode 番号を続けたストリングに置き換えます。例えば、データベースの Order_Item テーブルのビジネス・オブジェクト名は Order_Item です。Shipping-Address テーブルのビジネス・オブジェクト名は ShippingU45Address です。

ビジネス・オブジェクト名には、アダプターまたはデータベースを意味する値は含まれません。つまり、ビジネス・オブジェクト名から情報や意味が派生することはありません。名前が別の名前で置換された場合でも、アダプターの動作は同じです。

ビジネス・オブジェクト名はデータベース固有のメタデータを扱います。名前のプレフィックスに JDBC や *%AppName%* のようなストリングを使用すると、アプリケーション固有と汎用の 2 つのタイプのビジネス・オブジェクトを区別するのに役立ちます。名前の残りの部分で、ビジネス・オブジェクトが表すテーブルまたはストアード・プロシージャーを説明することができます。例えば、Human Resources (HR) のようなデータベース・アプリケーションの Employee テーブル用のビジネス・オブジェクト定義を生成する場合、相当するビジネス・オブジェクト名は HREmployee です。

データベース照会、バッチ SQL ステートメント、およびラッパーの場合のビジネス・オブジェクトなど、データベース・オブジェクトに対応しないビジネス・オブジェクトでは、別のクエリー・ビジネス・オブジェクトの名前と重複しない名前を使用するように注意してください。重複していると、名前が上書きされます。

ビジネス・オブジェクト名ではグローバル化文字がサポートされています。

WebSphere Integration Developer のリファクタリング機能を使用して、ビジネス・オブジェクトの名前を変更することができます。詳細については、WebSphere Integration Developer の資料を参照してください。

ウィザードでビジネス・オブジェクトに対して使用される命名規則を以下の表に示します。

表 25. ビジネス・オブジェクトの命名規則

エレメント	命名規則
ビジネス・グラフ	親ビジネス・オブジェクトが含まれているビジネス・グラフの名前は、含まれるビジネス・オブジェクトの名前の後にストリング「BG」を付加したものになります。例えば、SalesCustomer ビジネス・オブジェクトを含むビジネス・グラフの名前は SalesCustomerBG になります。
以下の項目に対応するビジネス・オブジェクト: • テーブル • ビュー • ストアード・プロシージャー • ストアード関数 • シノニムとニックネーム	テーブル、ビュー、ストアード・プロシージャー、シノニムおよびニックネームに基づくビジネス・オブジェクトの場合、外部サービス・ウィザードは、ビジネス・オブジェクト名として *Prefix* + *SchemaName* + *ObjectName* という形式の名前を生成します。ここで、 • *Prefix* はプレフィックスという名前の外部サービス接続プロパティーで指定された値です。プレフィックスは必須ではありません。指定しない場合は、ビジネス・オブジェクト名にプレフィックスが追加されません。 • *SchemaName* は、オブジェクトが属するスキーマの名前です。 • *ObjectName* はテーブル、ビュー、ストアード・プロシージャー、ストアード関数、またはシノニム/ニックネームの名前です。 例えば、Sales スキーマで Customer 表の Campaign12 プレフィックスを使用する場合、ビジネス・オブジェクト名は Campaign12SalesCustomer となります。

表 25. ビジネス・オブジェクトの命名規則 (続き)

エレメント	命名規則
クエリー・ビジネス・オブジェクト	クエリー・ビジネス・オブジェクトの場合、外部サービス・ウィザードは、ビジネス・オブジェクト名として *Prefix* + *QueryBOName* という形式の名前を生成します。ここで、 • *Prefix* は、ウィザードで指定するプレフィックスです。プレフィックスは必須ではありません。指定しない場合は、ビジネス・オブジェクト名にプレフィックスが追加されません。 • *QueryBOName* は、ウィザードでビジネス・オブジェクトを構成したときに指定した値です。
BatchSQL ビジネス・オブジェクト	batchSQL ビジネス・オブジェクトの場合、外部サービス・ウィザードは、ビジネス・オブジェクト名として *Prefix* + *BatchSQLBOName* という形式の名前を生成します。ここで、 • *Prefix* は、ウィザードで指定するプレフィックスです。プレフィックスは必須ではありません。指定しない場合は、ビジネス・オブジェクト名にプレフィックスが追加されません。 • *BatchSQLBOName* は、ウィザードでビジネス・オブジェクトを構成したときに指定した名前です。
ラッパー・ビジネス・オブジェクト	ラッパー・ビジネス・オブジェクトの場合、外部サービス・ウィザードは、ビジネス・オブジェクト名として *Prefix* + *WrapperBOName* という形式の名前を生成します。ここで、 • *Prefix* は、ウィザードで指定するプレフィックスです。プレフィックスは必須ではありません。指定しない場合は、ビジネス・オブジェクト名にプレフィックスが追加されません。 • *WrapperBOName* は、ウィザードでビジネス・オブジェクトを構成したときに指定した名前です。

Outbound 構成プロパティー

WebSphere Adapter for JDBC には、オブジェクトやサービスを生成したり作成したりするときに、外部サービス・ウィザードを使用して設定する、いくつかの種類の Outbound 接続構成プロパティーがあります。リソース・アダプターおよび管理接続ファクトリーのプロパティーは、WebSphere Process Server にモジュールをデプロイした後に、WebSphere Integration Developer 管理コンソールまたは WebSphere Process Server 管理コンソールを使用して変更できますが、外部サービス・ウィザードの接続プロパティーは、デプロイメント後に変更することはできません。

プロパティーの詳細についてのガイド

WebSphere Adapter for JDBC を構成するときに使用されるプロパティーは、リソース・アダプター・プロパティーや管理接続ファクトリー・プロパティーなど、それぞれの構成プロパティーのトピックに記載されている表で詳細に説明されています。これらの表を使用しやすくするため、参照する各行の情報を以下に説明します。

次の表では、構成プロパティーの表に表示される場合がある各行の意味を説明します。

行	説明
必須	アダプターが動作するためには、必須フィールド (プロパティー) に値が必要です。必須プロパティーに対しては、外部サービス・ウィザードがデフォルト値を提供する場合があります。 外部サービス・ウィザードの*必須フィールドからデフォルト値を除去*しても、デフォルト*値は変更されません*。必須フィールドに値がまったく入っていないと、外部サービス・ウィザードは、その割り当て済みのデフォルト値を使用してフィールドを処理し、そのデフォルト値は管理コンソールにも表示されます。 可能な値は「**はい**」および「**いいえ**」です。 プロパティーは、他のプロパティーが特定の値の場合のみ必須となることがあります。その場合は、表にこの依存関係が記載されます。例えば以下のようになります。 • EventQueryType プロパティーが Dynamic に設定された場合は「はい」 • Oracle データベースの場合は「はい」
使用可能な値	プロパティーで選択可能な値をリストして説明します。
デフォルト	外部サービス・ウィザードによって設定される事前定義値。プロパティーが必須の場合は、デフォルト値を受け入れるか、ユーザーが値を指定する必要があります。プロパティーにデフォルト値がない場合、表には「デフォルト値なし」と記載されます。 `None` という語は、受け入れ可能なデフォルト値です。デフォルト値がないという意味ではありません。
計測単位	プロパティーの計測単位を指定します (例: キロバイト、秒)。
プロパティー・タイプ	プロパティー・タイプを示します。有効なプロパティー・タイプとしては、以下のものがあります。 • Boolean • String • Integer
使用法	プロパティーに適用される場合がある使用の条件または制限について記述します。制限の記載例を以下に示します。 WebSphere Application Server バージョン 6.40 またはそれ以前では、パスワードに以下の制限があります。 • 大文字である必要があります • 長さが 8 文字である必要があります WebSphere Application Server バージョン 6.40 よりも後のバージョンでは、パスワードの制限が以下のように変更されました。 • 大文字小文字を区別しません • 長さが 40 文字まで可能です このセクションでは、このプロパティーに影響を及ぼす他のプロパティー、またはこのプロパティーによって影響を受ける他のプロパティーをリストし、その条件付き関係の内容を説明します。
例	プロパティー値のサンプルを示します。例: 「言語が JA (日本語) に設定された場合、コード・ページ番号は `8000` に設定されます。」

行	説明
グローバル化	グローバル化されたプロパティーには各国語サポートが備わっているので、値を各国語で設定できます。 有効な値は「**はい**」および「**いいえ**」です。
BIDI 対応	プロパティーが双方向 (bidi) 処理でサポートされるかどうかを示します。双方向処理とは、1 つのファイルに左から右 (ヘブライ語やアラビア語など) と右から左 (URL やファイル・パスなど) の両方の意味内容を含むデータを処理するタスクを指します。 有効な値は「**はい**」および「**いいえ**」です。

ウィザードの接続プロパティー

外部サービス接続プロパティーは、外部サービス・ウィザード (ビジネス・オブジェクト作成ツール) とデータベース間の接続を確立するために使用されます。これらのプロパティーにより、接続構成、双方向変換プロパティー、およびウィザードのログ記録オプションなどが指定されます。接続の確立後に、ウィザードは、ビジネス・オブジェクトの作成に必要なメタデータをデータベース内でディスカバーできます。

データベース内でオブジェクトをディスカバーするためにウィザードで指定したプロパティーの一部は、ウィザードで後で指定する実行時プロパティーの初期値として使用されます。このようなプロパティーには、リソース・アダプター、管理接続ファクトリー、およびアクティベーション・スペックのプロパティーがあります。

外部サービス・ウィザードの接続プロパティーとその目的を以下の表に示します。各プロパティーの完全な説明は、表に続くセクションで説明します。後続セクションのプロパティー詳細表の見方について詳しくは、206 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 26. 外部サービス・ウィザードの接続プロパティー

ウィザードのプロパティー名	説明
追加の JDBC ドライバー接続プロパティー	JDBC ドライバーを使用してデータベースへ接続するときに使用される UserName および Password プロパティー以外の追加プロパティー
209 ページの『データベース』	データベース名
209 ページの『データベース・ソフトウェア』	アダプターがアクセスするデータベース管理ソフトウェアの名前とバージョン
データベース URL	データベースへの接続に使用されるデータベース URL
211 ページの『ホスト名』	データベース・サーバーのホスト名または IP アドレス
211 ページの『JDBC ドライバー・クラス名』	JDBC ドライバー・クラスの名前
211 ページの『JDBC ドライバー・タイプ』	使用する JDBC ドライバーのタイプ
パスワード	対応するユーザー名のパスワード
212 ページの『ポート番号』	データベース・インスタンスへ接続するポートのポート番号
212 ページの『ビジネス・オブジェクト名のプレフィックス』	ビジネス・オブジェクト名に追加されるプレフィックス

表 26. 外部サービス・ウィザードの接続プロパティー (続き)

ウィザードのプロパティー名	説明
ユーザー名	データベース・ユーザー名

外部サービス・ウィザードは、双方向接続プロパティーを使用して、エンタープライズ情報システムに渡すデータに適切な双方向変換を適用します。

追加の JDBC ドライバー接続プロパティー

このプロパティーには、JDBC ドライバーを使用したデータベースへの接続に関する追加情報が含まれています。

表 27. ドライバー接続プロパティーの詳細

必須	いいえ
使用可能な値	データベース接続プロパティーはデータベース固有です。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これらの接続プロパティーは、UserName および Password プロパティーと共に使用されます。これにより、アダプターが使用するデータベース接続をカスタマイズできます。 接続プロパティーは、1 つ以上の *name*:*value* ペアとして指定し、ペアとペアの間はセミコロン (;) で区切ります。
例	このプロパティーに以下の値を設定すると、ログイン・タイムアウト間隔が指定され、データベース接続が読み取り専用になり、セキュリティー・メカニズムが設定されます。 `loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY`
グローバル化	はい
BIDI 対応	いいえ

データベース

このプロパティーは、データベースの名前を指定します。

表 28. 「データベース」の詳細

必須	はい
デフォルト	デフォルト値はデータベースによって異なります。
プロパティー・タイプ	String
使用法	これは、アクセスするデータベースの名前です。Oracle データベースの場合、これはデータベースを識別するシステム ID (SID) です。
グローバル化	はい
BIDI 対応	はい

データベース・ソフトウェア

このプロパティーは、アダプターがアクセスするデータベースを管理するデータベース管理ソフトウェアを指定します。

表 29. 「データベース・ソフトウェア」の詳細

行	説明
必須	はい
使用可能な値	このプロパティーは、一般的なデータベース・ソフトウェアを名前およびバージョン番号ごとにリストします。ご使用のソフトウェアがリストされていない場合は、「汎用 JDBC (Generic JDBC)」を選択してください。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードは、このプロパティーの値を使用して他のプロパティーのデフォルト値を設定し、データベース固有の選択リストを生成します。例えば、「DB2 USB Version 9.1」を選択すると、ウィザードの JDBC ドライバー・クラス・フィールドには、そのバージョンの DB2 UDB によってサポートされる JDBC ドライバーのみが表示されます。「Oracle 10」を選択すると、異なる JDBC ドライバーのセットが表示されます。
グローバル化	はい
BIDI 対応	はい

データベース URL

このプロパティーは、データベース接続を作成するための JDBC ドライバー固有の URL を指定します。

表 30. 「データベース URL」の詳細

必須	特定の JDBC ドライバーの場合は「はい」
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これは、使用するデータベース・ソフトウェアと JDBC ドライバーに固有の値です。 データベース・サーバーで IPv6 がサポートされている場合は、データベース URL のホスト名部分を IPv6 形式で指定できます。IP アドレスを大括弧 ([]) で囲んでください。

例	一般的なデータベース・サーバーの標準的な値を以下に示します。
	
	DB2 Universal (タイプ 4) JDBC ドライバー
	`jdbc:db2://www.example.com:50000/DB`
	
	DB2 Universal JDBC ドライバー (IPv6 アドレス)
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	
	DB2 Universal Database タイプ・ドライバー (ローカル接続用)
	`jdbc:db2:TEST`
	
	DB2 Universal Database タイプ 2 ドライバー (リモート接続用)
	`jdbc:db2://www.example.com:50000/TEST`
	
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

グローバル化	はい
BIDI 対応	はい

ホスト名

このプロパティーは、データベース・サーバーのホスト名または IP アドレスを指定します。

表 31. 「ホスト名」の詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーは、データベース・サーバーのホスト名または IP アドレスを指定します。データベース・サーバーにより IPv6 がサポートされている場合は、ホスト名を IPv6 形式で指定できます。
グローバル化	はい
BIDI 対応	はい

JDBC ドライバー・クラス名

このプロパティーは、JDBC ドライバー・クラスの名前を指定します。

表 32. 「JDBC ドライバー・クラス名」の詳細

行	説明
必須	はい
使用可能な値	使用可能な値は、データベースのタイプとバージョンによって異なります。ウィザードには、既知のドライバーのリストが表示されます。
デフォルト	デフォルトは、データベースのタイプとバージョンによって異なります。
プロパティー・タイプ	String
使用法	これは JDBC ドライバーのクラス名です。ウィザードには、選択した JDBC ドライバー・タイプのデフォルト・クラス名が表示されますが、必要に応じて別のクラス名を入力できます。クラス名は、ウィザードの開始時に指定した JDBC ドライバー・ファイルに記述されている必要があります。
グローバル化	はい
BIDI 対応	いいえ

JDBC ドライバー・タイプ

このプロパティーは、使用する JDBC ドライバーのタイプを指定します。

表 33. 「JDBC ドライバー・タイプ」の詳細

行	説明
必須	はい
使用可能な値	使用可能な値は、データベースのタイプとバージョンによって異なります。ウィザードには、既知のドライバーのリストが表示されます。
デフォルト	デフォルトは、データベースのタイプとバージョンによって異なります。

表 33. 「JDBC ドライバー・タイプ」の詳細 (続き)

行	説明
プロパティー・タイプ	String
使用法	これは、使用する JDBC ドライバーのタイプです。根本的な点は使用するドライバーがタイプ 2 とタイプ 4 (ユニバーサル) のどちらであるかですが、各データベース・システムでは、ドライバーにデータベース・システム固有の名前が使用されています。各データベース・システムの既知のドライバーのリストがウィザードに表示されます。使用するドライバーがリストにない場合は、「その他」を選択します。このフィールドの情報は、ウィザードの開始時に指定した JDBC ドライバー・ファイルと一致している必要があります。
グローバル化	はい
BIDI 対応	いいえ

パスワード (Password)

このプロパティーは、データベース・ユーザーのユーザー名に対するパスワードを指定します。

表 34. 「パスワード」の詳細

必須	DataSourceJNDIName または XADataSourceName プロパティーが設定されていない場合は必須。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを設定すると、DataSourceJNDIName または XADataSourceName プロパティーを使用してサーバーのデータ・ソースに指定されているパスワードが上書きされます。
グローバル化	はい
BIDI 対応	はい

ポート番号

このプロパティーは、データベース・インスタンスのポート番号を指定します。

表 35. 「ポート番号」の詳細

必須	はい
デフォルト	デフォルト値はデータベースによって異なります。また、デフォルト値はウィザードにより初期化されます。
プロパティー・タイプ	String
使用法	これは、データベース・インスタンスへ接続するポートのポート番号です。
グローバル化	はい
BIDI 対応	いいえ

ビジネス・オブジェクト名のプレフィックス

ビジネス・オブジェクトの名前に追加されるプレフィックス。

表 36. 「プレフィックス」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	プレフィックスを使用して、ビジネス・オブジェクトのタイプを容易に区別できるようにします。
例	汎用ビジネス・オブジェクトにプレフィックス JDBC を指定し、アプリケーション固有のビジネス・オブジェクトに %AppName% を指定する場合があります。
グローバル化	はい
BIDI 対応	いいえ

ユーザー名 (UserName)

このプロパティーは、データベース接続に使用するデータベース・ユーザー名を指定します。

表 37. 「ユーザー名」の詳細

必須	DataSourceJNDIName または XADataSourceName プロパティーが設定されていない場合は必須。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを設定すると、DataSourceJNDIName または XADataSourceName プロパティーを使用してサーバーのデータ・ソースに指定されているユーザー名が上書きされます。
グローバル化	はい
BIDI 対応	はい

リソース・アダプター・プロパティー

リソース・アダプター・プロパティーは、アダプターの一般的な操作 (ビジネス・オブジェクトのネーム・スペースの指定など) を制御します。リソース・アダプター・プロパティーは、アダプターの構成時に外部サービス・ウィザードを使用して設定します。アダプターのデプロイ後に、これらのプロパティーを変更するには、管理コンソールを使用します。

以下に示すロギングおよびトレースのプロパティーは、バージョン 6.1.0 には必要なくなりましたが、旧バージョンとの互換性を維持するためにサポートされています。

- ログ・ファイル最大サイズ
- ログ・ファイル名
- ログ・ファイル数
- トレース・ファイル最大サイズ
- トレース・ファイル名
- トレース・ファイル数

バージョン 6.0.2 で Inbound 処理と Outbound 処理の両方に対して指定されている次のプロパティーは、バージョン 6.1.0 では Inbound 処理にのみ適用されます。Outbound 処理に適用されるプロパティーは、管理接続ファクトリー・プロパティー・グループに属しています。

- Ping クエリー
- クエリー・タイムアウト
- ReturnDummyBOForSP

BusinessObjectNameSpaceプロパティーは、アクティベーション・スペック・プロパティーに移動しました。

リソース・アダプター・プロパティーとその目的を次の表に示します。各プロパティーの完全な説明は、表に続くセクションで説明します。プロパティー詳細表の見方について詳しくは、206 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 38. Adapter for JDBC 用のリソース・アダプター・プロパティー

名前		
ウィザード内	**管理コンソール内**	**説明**
ロギングおよびトレースで使用するアダプター ID	AdapterID	CEI イベントおよび PMI イベントのアダプター・インスタンスをロギングおよびトレースを基準にして識別します。
データベース・ベンダー	DatabaseVendor	アダプターが特殊な処理に使用するデータベースのタイプ
クエリー・タイムアウト	QueryTimeOut	すべての SQL ステートメントでの照会の最大実行時間 (秒数)
ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す	ReturnDummyBOForSP	結果セットが空の場合に出力パラメーターを返すかどうかを指定します。
(なし)	enableHASupport	このプロパティーは変更しないでください。
(なし)	ログ・ファイル最大サイズ	旧バージョンとの互換性を維持するためにサポートされています。
(なし)	LogFilename	旧バージョンとの互換性を維持するためにサポートされています。
(なし)	ログ・ファイル数	旧バージョンとの互換性を維持するためにサポートされています。
接続を検証するための SQL 照会	PingQuery	データベースへの接続の有効性をテストするのに使用する SQL 照会
(なし)	トレース・ファイル最大サイズ	旧バージョンとの互換性を維持するためにサポートされています。
(なし)	トレース・ファイル名	旧バージョンとの互換性を維持するためにサポートされています。
(なし)	トレース・ファイル数	旧バージョンとの互換性を維持するためにサポートされています。

ロギングおよびトレースで使用するアダプター ID (AdapterID)

このプロパティーは、アダプターの特定のデプロイメント (インスタンス) を識別する場合に使用します。

表 39. 「ロギングおよびトレースで使用するアダプター ID」の詳細

必須	はい
デフォルト	CWYBC_JDBC
プロパティー・タイプ	String
使用法	このプロパティーは、PMI イベントのアダプター・インスタンスを識別する場合に使用します。アダプターのインスタンスを複数デプロイする場合は、このプロパティーをアダプターのインスタンスごとに固有な値に設定します。 Inbound 処理の場合、このプロパティーはリソース・アダプター・プロパティーから取り出します。Outbound 処理の場合は、管理接続ファクトリー・プロパティーから取り出します。
グローバル化	はい
BIDI 対応	いいえ

データベース・ベンダー (DatabaseVendor)

このプロパティーは、使用されるデータベースのタイプを指定します。このタイプは、データベース・ベンダー名により決まります。

表 40. 「データベース・ベンダー」の詳細

必須	はい

使用可能な値	`DB2`
	`MSSQLServer`
	`Oracle`
	`Others`

デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	一部の SQL ステートメントでは特殊な処理が必要であり、この処理はデータベース・タイプに応じて異なります。例えば、Oracle の Struct データ型と Array データ型には特殊な処理が必要です。このプロパティーは、使用する RDBMS を指定します。これにより、データベース・タイプが決まります。 ご使用のデータベースのベンダーに対応する値を指定します。 • `DB2` (IBM DB2 データベース) • `Oracle` (Oracle データベース) • `MSSQLServer` (Microsoft SQL Server データベース) • `Others` (その他のすべてのデータベース・タイプ) その他のデータベース・タイプの場合、アダプターは特殊な処理を一切実行しません。JDBCDriverClass プロパティーに指定されているドライバーが正しいことを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

高可用性サポートを使用可能にする (Enable high availability support) (enableHASupport)

このプロパティーは変更しないでください。true に設定してください。

クエリー・タイムアウト (QueryTimeOut)

このプロパティーは、すべての SQL ステートメントでの照会の最大実行時間を秒数で指定します。

表 41. 「クエリー・タイムアウト」の詳細

必須	いいえ
デフォルト	デフォルト値なし
計測単位	秒
プロパティー・タイプ	整数
使用法	照会の処理に、指定された秒数より長い時間が必要な場合は、データベースにより、キャプチャーされる SQL 例外が生成されます。関連付けられているメッセージがログ・ファイルに記録されます。 値を指定しない場合は、照会のタイムアウトが設定されません。
グローバル化	はい
BIDI 対応	いいえ

ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)

このプロパティーは、結果セットが空の場合に出力パラメーターを返すかどうかを指定します。

表 42. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細

必須	いいえ
使用可能な値	True False
デフォルト	False
プロパティー・タイプ	Boolean
使用法	ストアード・プロシージャー取得 (RetrieveSP) 操作では、結果セットが戻されます。結果のセットが空であり、ReturnDummyBOForSP プロパティーが False に設定されている場合は、ビジネス・オブジェクトが作成されず、プロシージャー呼び出しから返される出力パラメーターを取得できません。 ただし、ReturnDummyBOForSP が True の場合は、ダミー・ビジネス・オブジェクトが作成され、対応する属性の出力パラメーターと入出力パラメーターの値が取り込まれます。
グローバル化	はい
BIDI 対応	いいえ

接続を検証するための SQL 照会 (PingQuery)

このプロパティーは、データベースへの接続の有効性をテストするために使用される SQL 照会を指定します。

表 43. 「ping クエリー (Ping query)」の詳細

必須	いいえ
プロパティー・タイプ	String
デフォルト	デフォルト値なし
使用法	このプロパティーには、アダプターがデータベースに接続できるかどうかを判断するために実行する SQL 照会ステートメントが含まれています。
グローバル化	いいえ
BIDI 対応	いいえ

管理接続ファクトリー・プロパティー

管理接続ファクトリー・プロパティーは、データベースとの Outbound 接続インスタンスを作成するために、アダプターがランタイムに使用します。

アダプターの構成時に、外部サービス・ウィザードを使用して管理接続ファクトリー・プロパティーを設定します。プロパティーは、WebSphere Integration Developer アセンブリー・エディターを使用して、またはデプロイメント後に WebSphere Process Server または WebSphere Enterprise Service Bus の管理コンソールを使用して変更できます。

管理接続ファクトリー・プロパティーのリストと説明を以下の表に示します。各プロパティーの完全な説明は、表に続くセクションで説明します。後続セクションのプロパティー詳細表の見方について詳しくは、206 ページの『プロパティーの詳細についてのガイド』を参照してください。

注: 外部サービス・ウィザードはこれらのプロパティーを管理接続ファクトリー・プロパティーとして参照し、管理コンソールは J2C 接続ファクトリー・プロパティーとして参照します。

表 44. Adapter for JDBC の管理接続ファクトリー・プロパティー

プロパティー名		
ウィザード内	**管理コンソール内**	**説明**
アダプター ID	AdapterID	CEI イベントおよび PMI イベントのアダプター・インスタンスをロギングおよびトレースを基準にして識別します。
追加の JDBC ドライバー接続プロパティー	DriverConnectionProperties	JDBC ドライバーを使用してデータベースへ接続するときに使用される UserName および Password プロパティー以外の追加プロパティー
自動コミット (Auto commit)	自動コミット	接続に使用する AutoCommit 値。
データ・ソース JNDI 名	DataSourceJNDIName	データベース接続の確立に使用される JNDI データ・ソースの名前
データベース URL	DatabaseURL	データベースへの接続に使用されるデータベース URL

表 44. Adapter for JDBC の管理接続ファクトリー・プロパティー (続き)

プロパティー名		
ウィザード内	**管理コンソール内**	**説明**
データベース・ベンダー	DatabaseVendor	アダプターが特殊な処理に使用するデータベースのタイプ
JDBC ドライバー・クラス (JDBC driver class)	JDBCDriverClass	データベース接続に使用される JDBC ドライバーのクラス名
パスワード	Password	対応するユーザー名のパスワード
クエリー・タイムアウト	QueryTimeOut	すべての SQL ステートメントでの照会の最大実行時間 (秒数)
ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す	ReturnDummyBOForSP	結果セットが空の場合に出力パラメーターを返すかどうかを指定します。
接続への SQL 照会	PingQuery	データベースへの接続の有効性をテストするのに使用する SQL 照会
ユーザー名	UserName	データベース・ユーザー名
XA データ・ソース名	XADataSourceName	XA (分散) トランザクションを実行するためにデータベースへの接続を確立するときに使用される XA データ・ソースの名前
XA データベース名	XADatabaseName	XA 接続で使用するデータベース名

ロギングおよびトレースで使用するアダプター ID (AdapterID)

このプロパティーは、アダプターの特定のデプロイメント (インスタンス) を識別する場合に使用します。

表 45. 「ロギングおよびトレースで使用するアダプター ID」の詳細

必須	はい
デフォルト	CWYBC_JDBC
プロパティー・タイプ	String
使用法	このプロパティーは、PMI イベントのアダプター・インスタンスを識別する場合に使用します。アダプターのインスタンスを複数デプロイする場合は、このプロパティーをアダプターのインスタンスごとに固有な値に設定します。 Inbound 処理の場合、このプロパティーはリソース・アダプター・プロパティーから取り出します。Outbound 処理の場合は、管理接続ファクトリー・プロパティーから取り出します。
グローバル化	はい
BIDI 対応	いいえ

追加の JDBC ドライバー接続プロパティー (DriverConnectionProperties)

このプロパティーには、JDBC ドライバーを使用したデータベースへの接続に関する追加情報が含まれています。

表 46. ドライバー接続プロパティーの詳細

必須	いいえ
使用可能な値	データベース接続プロパティーはデータベース固有です。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これらの接続プロパティーは、UserName および Password プロパティーと共に使用されます。これにより、アダプターが使用するデータベース接続をカスタマイズできます。 接続プロパティーは、1 つ以上の *name*:*value* ペアとして指定し、ペアとペアの間はセミコロン (;) で区切ります。
例	このプロパティーに以下の値を設定すると、ログイン・タイムアウト間隔が指定され、データベース接続が読み取り専用になり、セキュリティー・メカニズムが設定されます。 `loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY`
グローバル化	はい
BIDI 対応	いいえ

自動コミット (Auto commit) (AutoCommit)

このプロパティーは、接続に AutoCommit を設定するかどうかを指定します。

表 47. 「自動コミット (Auto commit)」の詳細

必須	いいえ
使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean
使用法	このプロパティーは、XA (分散) トランザクションでは無視されます。
グローバル化	いいえ
BIDI 対応	いいえ

データ・ソース JNDI 名 (DataSourceJNDIName)

このプロパティーは、データベース接続を確立するときに使用される JNDI データ・ソースの名前を指定します。

表 48. 「データ・ソース JNDI 名」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 48. 「データ・ソース JNDI 名」の詳細 (続き)

使用法	このプロパティーを使用して、ターゲット・データベースの接続情報を指定する WebSphere Process Server または WebSphere Enterprise Service Bus のデータ・ソースの JNDI 名を指定します。 Inbound 操作および Outbound 操作のパフォーマンスを改善するには、準備済みステートメントをキャッシュするために使用可能に設定されているデータ・ソースの名前を指定します。 ユーザー名プロパティーとパスワード・プロパティーも設定されている場合、データ・ソースのユーザー名とパスワードはこれらのプロパティーにより上書きされます。 データベース接続プロパティーは次の順序で使用されます。 1. DataSourceJNDIName プロパティーが設定されている場合、アダプターはこのプロパティーを使用してデータベースへの接続を確立します。 UserName プロパティーと Password プロパティーも設定されている場合は、データ・ソースで設定されているユーザー名とパスワードはこれらのプロパティーにより上書きされます。 2. DataSourceJNDIName プロパティーが設定されておらず、XADataSourceName プロパティーと XADatabaseName プロパティーが設定されている場合は、アダプターはこれらのプロパティーを使用して接続を確立します。 DataSourceJNDIName プロパティーは、XA データ・ソースまたは接続プール・データ・ソースを表します。 XA トランザクションをサポートするサーバーで JNDI データ・ソースを定義し、その後アダプターの構成時にそのデータ・ソースを指定する場合は、XA トランザクションをサポートするすべてのタイプのデータベースに接続できます。XA データ・ソースとデータベースを使用する場合は、アダプターは DB2 および Oracle データベースでのみ XA トランザクションをサポートします。 3. DataSourceJNDIName、XADataSourceName、および XADatabaseName プロパティーがすべて設定されていない場合は、アダプターは DatabaseURL、JDBCDriverClass、UserName、および Password プロパティーを使用して接続を確立します。 データ・ソース JNDI 名前プロパティーを、管理接続ファクトリーの JNDI 名、またはサーバーのアクティベーション・スペックと混同しないでください。以下のリストに、JNDI 名のタイプ間での重要な違いを示します。 • データ・ソース JNDI 名 – データベースへの接続を指定する – ユーザー名とパスワードをアダプターのプロパティーに保存する代わりに使用される – アダプター・プロパティーとして保存される • 管理接続ファクトリーまたはアクティベーション・スペックの JNDI 名 – サーバーの管理接続ファクトリーまたはアクティベーション・スペックへの接続を指定する – ウィザードで各管理接続ファクトリーまたはアクティベーション・スペック・プロパティーの値を指定する代わりに使用される – インポート・ファイルの接続ターゲットとして保存される
グローバル化	はい
BIDI 対応	いいえ

データベース URL (DatabaseURL)

このプロパティーは、データベース接続を作成するための JDBC ドライバー固有の URL を指定します。

表 49. 「データベース URL」の詳細

必須	次のいずれかのプロパティー・セットが設定されている場合を除き必須。 • DataSourceJNDIName プロパティー • XADataSourceName および XADatabaseName プロパティー
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードのデータベース固有のフィールドに情報を入力し、データベース URL を作成します。例えば、DB2 データベースのデータベース URL は、データベース名、サーバー・ホスト名、およびデータベース・ポート番号で構成されています。管理コンソールで、データベース URL 値全体を入力します。 データベース・サーバーで IPv6 がサポートされている場合は、データベース URL のホスト名部分を IPv6 形式で指定できます。 データベース接続プロパティーは次の順序で使用されます。 1. DataSourceJNDIName プロパティーが設定されている場合、アダプターはこのプロパティーを使用してデータベースへの接続を確立します。 UserName プロパティーと Password プロパティーも設定されている場合は、データ・ソースで設定されているユーザー名とパスワードはこれらのプロパティーにより上書きされます。 2. DataSourceJNDIName プロパティーが設定されておらず、XADataSourceName プロパティーと XADatabaseName プロパティーが設定されている場合は、アダプターはこれらのプロパティーを使用して接続を確立します。 DataSourceJNDIName プロパティーは、XA データ・ソースまたは接続プール・データ・ソースを表します。 XA トランザクションをサポートするサーバーで JNDI データ・ソースを定義し、その後アダプターの構成時にそのデータ・ソースを指定する場合は、XA トランザクションをサポートするすべてのタイプのデータベースに接続できます。XA データ・ソースとデータベースを使用する場合は、アダプターは DB2 および Oracle データベースでのみ XA トランザクションをサポートします。 3. DataSourceJNDIName、XADataSourceName、および XADatabaseName プロパティーがすべて設定されていない場合は、アダプターは DatabaseURL、JDBCDriverClass、UserName、および Password プロパティーを使用して接続を確立します。 ホスト名を IPv6 形式の IP アドレスとして指定する場合は、その IP アドレスを大括弧 ([]) で囲んでください。

表 49. 「データベース URL」の詳細 (続き)

例	一般的なデータベース・サーバーの標準的な値を以下に示します。
	DB2 Universal (タイプ 4) JDBC ドライバー
	`jdbc:db2://www.example.com:50000/DB`
	DB2 Universal JDBC ドライバー (IPv6 アドレス)
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	DB2 Universal Database タイプ 2 ドライバー (ローカル接続用)
	jdbc:db2:TEST
	DB2 Universal Database タイプ 2 ドライバー (リモート接続用)
	jdbc:db2://www.example.com:50000/TEST
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

グローバル化	はい
BIDI 対応	はい

データベース・ベンダー (DatabaseVendor)

このプロパティーは、使用されるデータベースのタイプを指定します。このタイプは、データベース・ベンダー名により決まります。

表 50. 「データベース・ベンダー」の詳細

必須	はい

使用可能な値	`DB2`
	`MSSQLServer`
	`Oracle`
	`Others`

デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	一部の SQL ステートメントでは特殊な処理が必要であり、この処理はデータベース・タイプに応じて異なります。例えば、Oracle の Struct データ型と Array データ型には特殊な処理が必要です。このプロパティーは、使用する RDBMS を指定します。これにより、データベース・タイプが決まります。 ご使用のデータベースのベンダーに対応する値を指定します。 • `DB2` (IBM DB2 データベース) • `Oracle` (Oracle データベース) • `MSSQLServer` (Microsoft SQL Server データベース) • `Others` (その他のすべてのデータベース・タイプ) その他のデータベース・タイプの場合、アダプターは特殊な処理を一切実行しません。JDBCDriverClass プロパティーに指定されているドライバーが正しいことを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

JDBC ドライバー・クラス (JDBC driver class) (JDBCDriverClass)

このプロパティーは、データベース接続に使用される JDBC ドライバーのクラス名を指定します。

表 51. 「JDBC ドライバー・クラス (JDBC driver class)」の詳細

必須	DataSourceJNDIName プロパティーが設定されていない場合は必須
使用可能な値	値はデータベースによって異なります。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードでは、共通データベース・ソフトウェアとドライバー (IBM DB2、Oracle、および MicrosoftSQL の最新バージョンのタイプ 4 ドライバーなど) の組み合わせを選択すると、ドライバーが自動的に指定されます。ほとんどのデータベース・ソフトウェアのほとんどのタイプ 2 ドライバーでは、データベース・クラス名を指定する必要があります。 タイプ 2 ドライバーまたは汎用ドライバーを選択する場合は、JDBC ドライバー・クラス名を入力する必要があります。例えば、DB2 Universal Database タイプ 2 ドライバーの場合、クラス名は `COM.ibm.db2.jdbc.app.DB2Driver` です。 管理コンソールで、ドライバーのデータベース固有名を入力してください。 DataSourceJNDIName プロパティーが設定されている場合、このプロパティーは無視されます。

例	**外部サービス・ウィザード内:**
	• ユニバーサル (タイプ 4) JDBC ドライバーを使用して DB2 データベースに接続するには、「`IBM DB2 Universal`」を選択します。
	• DB2 ユニバーサルのタイプ 2 ドライバーを使用して DB2 データベースに接続するには、「`Other`」を選択します。
	• タイプ 4 ドライバーを使用して Oracle 10 データベースに接続するには、「`Oracle Thin Driver`」を選択します。
	
	管理コンソール内
	
	DB2 Universal Database タイプ 2 ドライバー
	`COM.ibm.db2.jdbc.app.DB2Driver`
	
	DB2 Universal Database タイプ 4 ドライバー
	`com.ibm.db2.jcc.DB2Driver`
	
	Oracle Thin JDBC ドライバー
	`oracle.jdbc.driver.OracleDriver`
	
	IBM Toolkit for Java リモート・ドライバー (i5/OS 用)
	`com.ibm.as400.access.AS400JDBCDriver`
	
	IBM WebSphere Connect JDBC ドライバー (Microsoft SQL Server 用)
	`com.ibm.websphere.jdbc.sqlserver.SQLServerDriver`

グローバル化	いいえ
BIDI 対応	いいえ

パスワード (Password)

このプロパティーは、データベース・ユーザーのユーザー名に対するパスワードを指定します。

表 52. 「パスワード」の詳細

必須	DataSourceJNDIName または XADataSourceName プロパティーが設定されていない場合は必須。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを設定すると、DataSourceJNDIName または XADataSourceName プロパティーを使用してサーバーのデータ・ソースに指定されているパスワードが上書きされます。
グローバル化	はい
BIDI 対応	はい

クエリー・タイムアウト (QueryTimeOut)

このプロパティーは、すべての SQL ステートメントでの照会の最大実行時間を秒数で指定します。

表 53. 「クエリー・タイムアウト」の詳細

必須	いいえ
デフォルト	デフォルト値なし
計測単位	秒
プロパティー・タイプ	整数
使用法	照会の処理に、指定された秒数より長い時間が必要な場合は、データベースにより、キャプチャーされる SQL 例外が生成されます。関連付けられているメッセージがログ・ファイルに記録されます。 値を指定しない場合は、照会のタイムアウトが設定されません。
グローバル化	はい
BIDI 対応	いいえ

ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)

このプロパティーは、結果セットが空の場合に出力パラメーターを返すかどうかを指定します。

表 54. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細

必須	いいえ
使用可能な値	`True` `False`
デフォルト	`False`
プロパティー・タイプ	Boolean

表 54. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細 (続き)

使用法	ストアード・プロシージャー取得 (RetrieveSP) 操作では、結果セットが戻されます。結果のセットが空であり、ReturnDummyBOForSP プロパティーが `False` に設定されている場合は、ビジネス・オブジェクトが作成されず、プロシージャー呼び出しから返される出力パラメーターを取得できません。 ただし、ReturnDummyBOForSP が `True` の場合は、ダミー・ビジネス・オブジェクトが作成され、対応する属性の出力パラメーターと入出力パラメーターの値が取り込まれます。
グローバル化	はい
BIDI 対応	いいえ

接続を検証するための SQL 照会 (PingQuery)

このプロパティーは、データベースへの接続の有効性をテストするために使用される SQL 照会を指定します。

表 55. 「ping クエリー (Ping query)」の詳細

必須	いいえ
プロパティー・タイプ	String
デフォルト	デフォルト値なし
使用法	このプロパティーには、アダプターがデータベースに接続できるかどうかを判断するために実行する SQL 照会ステートメントが含まれています。
グローバル化	いいえ
BIDI 対応	いいえ

ユーザー名 (UserName)

このプロパティーは、データベース接続に使用するデータベース・ユーザー名を指定します。

表 56. 「ユーザー名」の詳細

必須	DataSourceJNDIName または XADataSourceName プロパティーが設定されていない場合は必須。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを設定すると、DataSourceJNDIName または XADataSourceName プロパティーを使用してサーバーのデータ・ソースに指定されているユーザー名が上書きされます。
グローバル化	はい
BIDI 対応	はい

XA データ・ソース名 (XADataSourceName)

このプロパティーでは、XA (分散) トランザクションを実行するためにデータベース接続を確立するときに使用される XA データ・ソースの名前を指定します。

表 57. 「XA データ・ソース名」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	DB2 データベースへの XA 接続を確立するため、このプロパティーと XADatabaseName プロパティーが組み合わせて使用されます。 Oracle データベースへの XA 接続を確立する場合は、このプロパティーが使用され、XADatabaseName プロパティーは使用されません。 DataSourceJNDIName プロパティーが指定されている場合、このプロパティーは無視されます。

例	タイプ 2 JDBC ドライバー (db2java.zip) を備えた DB2 データベースの標準的な値:
	
	`COM.ibm.db2.jdbc.DB2XADataSource`
	
	タイプ 4 JDBC ドライバー (db2jcc.jar) を備えた DB2 データベースの標準的な値:
	
	`com.ibm.db2.jcc.DB2XADataSource`
	
	Oracle データベースの標準的な値:
	
	`oracle.jdbc.xa.client.OracleXADataSource`

グローバル化	いいえ
BIDI 対応	いいえ

XA データベース名 (XADatabaseName)

このプロパティーは、XA 接続に使用するデータベースの名前を指定します。

表 58. 「XA データベース名」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	DB2 データベースへの XA 接続を確立するため、このプロパティーと XADataSourceName プロパティーが組み合わせて使用されます。Oracle データベースの場合は、このプロパティーは不要です。 DataSourceJNDIName プロパティーが指定されている場合、このプロパティーは無視されます。
グローバル化	はい
BIDI 対応	はい

対話スペック・プロパティー

対話スペック (InteractionSpec) プロパティーは、操作の対話を制御します。アダプターを構成するときに、外部サービス・ウィザードによって対話スペック・プロパティーが設定されます。一般に、このプロパティーを変更する必要はありません。ただし、Outbound 操作に関する一部のプロパティーはユーザーが変更できます。例えば、RetrieveAll 操作が返す情報が十分でない場合は、RetrieveAll によって返されるレコードの最大数を指定する対話スペック・プロパティーの値を大きくすることができます。アプリケーションのデプロイ後にこれらのプロパティーを変更するに

は、WebSphere Integration Developer のアセンブリー・エディターを使用します。プロパティーは、インポートのメソッド・バインディングに存在します。

表 59 に、ユーザーが設定する対話スペック・プロパティーをリストし、説明します。後続セクションのプロパティー詳細表の見方について詳しくは、 206 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 59. Adapter for JDBC 用の対話スペック・プロパティー

プロパティー名	説明
『RetrieveAll 操作の最大レコード数』	RetrieveAll 操作時に返す結果のセットの最大数

RetrieveAll 操作の最大レコード数

このプロパティーは、RetrieveAll 操作で返されるレコードの最大数を指定します。

表 60. RetrieveAll 操作の最大レコード数の詳細

必須	はい
デフォルト	100
使用法	データベースでの一致数がこのプロパティーの値を超えると、アダプターは、MatchesExceededLimitException 例外と MatchesExceededLimitFault フォールトをスローします。RetrieveAll 操作がレコードのすべてを返さない場合は、この値を大きくしてください。メモリー不足になる場合は、値を小さくしてください。
プロパティー・タイプ	Integer
グローバル化	いいえ
BIDI 対応	いいえ

Inbound 構成プロパティー

WebSphere Adapter for JDBC には、オブジェクトやサービスを生成したり作成したりするときに、外部サービス・ウィザードを使用して設定する、いくつかの種類の Inbound 接続構成プロパティーがあります。リソース・アダプターおよびアクティベーション・スペックのプロパティーは、モジュールをデプロイした後に WebSphere Integration Developer 管理コンソールまたは WebSphere Process Server 管理コンソールを使用して変更できますが、外部サービス・ウィザードの接続プロパティーは、デプロイメント後に変更することはできません。

プロパティーの詳細についてのガイド

WebSphere Adapter for JDBC を構成するときに使用されるプロパティーは、リソース・アダプター・プロパティーや管理接続ファクトリー・プロパティーなど、それぞれの構成プロパティーのトピックに記載されている表で詳細に説明されています。これらの表を使用しやすくするため、参照する各行の情報を以下に説明します。

次の表では、構成プロパティーの表に表示される場合がある各行の意味を説明します。

行	説明
必須	アダプターが動作するためには、必須フィールド (プロパティー) に値が必要です。必須プロパティーに対しては、外部サービス・ウィザードがデフォルト値を提供する場合があります。 外部サービス・ウィザードの*必須フィールドからデフォルト値を除去*しても、デフォルト値は変更されません。必須フィールドに値がまったく入っていないと、外部サービス・ウィザードは、その割り当て済みのデフォルト値を使用してフィールドを処理し、そのデフォルト値は管理コンソールにも表示されます。 可能な値は「**はい**」および「**いいえ**」です。 プロパティーは、他のプロパティーが特定の値の場合のみ必須となることがあります。その場合は、表にこの依存関係が記載されます。例えば以下のようになります。 • EventQueryType プロパティーが Dynamic に設定された場合は「はい」 • Oracle データベースの場合は「はい」
使用可能な値	プロパティーで選択可能な値をリストして説明します。
デフォルト	外部サービス・ウィザードによって設定される事前定義値。プロパティーが必須の場合は、デフォルト値を受け入れるか、ユーザーが値を指定する必要があります。プロパティーにデフォルト値がない場合、表には「デフォルト値なし」と記載されます。 `None` という語は、受け入れ可能なデフォルト値です。デフォルト値がないという意味ではありません。
計測単位	プロパティーの計測単位を指定します (例: キロバイト、秒)。
プロパティー・タイプ	プロパティー・タイプを示します。有効なプロパティー・タイプとしては、以下のものがあります。 • Boolean • String • Integer
使用法	プロパティーに適用される場合がある使用の条件または制限について記述します。制限の記載例を以下に示します。 WebSphere Application Server バージョン 6.40 またはそれ以前では、パスワードに以下の制限があります。 • 大文字である必要があります • 長さが 8 文字である必要があります WebSphere Application Server バージョン 6.40 よりも後のバージョンでは、パスワードの制限が以下のように変更されました。 • 大文字小文字を区別しません • 長さが 40 文字まで可能です このセクションでは、このプロパティーに影響を及ぼす他のプロパティー、またはこのプロパティーによって影響を受ける他のプロパティーをリストし、その条件付き関係の内容を説明します。
例	プロパティー値のサンプルを示します。例: 「言語が JA (日本語) に設定された場合、コード・ページ番号は `8000` に設定されます。」

行	説明
グローバル化	グローバル化されたプロパティーには各国語サポートが備わっているので、値を各国語で設定できます。 有効な値は「**はい**」および「**いいえ**」です。
BIDI 対応	プロパティーが双方向 (bidi) 処理でサポートされるかどうかを示します。双方向処理とは、1 つのファイルに左から右 (ヘブライ語やアラビア語など) と右から左 (URL やファイル・パスなど) の両方の意味内容を含むデータを処理するタスクを指します。 有効な値は「**はい**」および「**いいえ**」です。

ウィザードの接続プロパティー

外部サービス接続プロパティーは、外部サービス・ウィザード (ビジネス・オブジェクト作成ツール) とデータベース間の接続を確立するために使用されます。これらのプロパティーにより、接続構成、双方向変換プロパティー、およびウィザードのログ記録オプションなどが指定されます。接続の確立後に、ウィザードは、ビジネス・オブジェクトの作成に必要なメタデータをデータベース内でディスカバーできます。

データベース内でオブジェクトをディスカバーするためにウィザードで指定したプロパティーの一部は、ウィザードで後で指定する実行時プロパティーの初期値として使用されます。このようなプロパティーには、リソース・アダプター、管理接続ファクトリー、およびアクティベーション・スペックのプロパティーがあります。

外部サービス・ウィザードの接続プロパティーとその目的を以下の表に示します。各プロパティーの完全な説明は、表に続くセクションで説明します。後続セクションのプロパティー詳細表の見方について詳しくは、206 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 61. 外部サービス・ウィザードの接続プロパティー

ウィザードのプロパティー名	説明
追加の JDBC ドライバー接続プロパティー	JDBC ドライバーを使用してデータベースへ接続するときに使用される UserName および Password プロパティー以外の追加プロパティー
230 ページの『データベース』	データベース名
230 ページの『データベース・ソフトウェア』	アダプターがアクセスするデータベース管理ソフトウェアの名前とバージョン
データベース URL	データベースへの接続に使用されるデータベース URL
232 ページの『ホスト名』	データベース・サーバーのホスト名または IP アドレス
232 ページの『JDBC ドライバー・クラス名』	JDBC ドライバー・クラスの名前
232 ページの『JDBC ドライバー・タイプ』	使用する JDBC ドライバーのタイプ
パスワード	対応するユーザー名のパスワード
233 ページの『ポート番号』	データベース・インスタンスへ接続するポートのポート番号
233 ページの『ビジネス・オブジェクト名のプレフィックス』	ビジネス・オブジェクト名に追加されるプレフィックス

表 61. 外部サービス・ウィザードの接続プロパティー (続き)

ウィザードのプロパティー名	説明
ユーザー名	データベース・ユーザー名

外部サービス・ウィザードは、双方向接続プロパティーを使用して、エンタープライズ情報システムに渡すデータに適切な双方向変換を適用します。

追加の JDBC ドライバー接続プロパティー

このプロパティーには、JDBC ドライバーを使用したデータベースへの接続に関する追加情報が含まれています。

表 62. ドライバー接続プロパティーの詳細

必須	いいえ
使用可能な値	データベース接続プロパティーはデータベース固有です。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これらの接続プロパティーは、UserName および Password プロパティーと共に使用されます。これにより、アダプターが使用するデータベース接続をカスタマイズできます。 接続プロパティーは、1 つ以上の *name*:*value* ペアとして指定し、ペアとペアの間はセミコロン (;) で区切ります。
例	このプロパティーに以下の値を設定すると、ログイン・タイムアウト間隔が指定され、データベース接続が読み取り専用になり、セキュリティー・メカニズムが設定されます。 `loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY`
グローバル化	はい
BIDI 対応	いいえ

データベース

このプロパティーは、データベースの名前を指定します。

表 63. 「データベース」の詳細

必須	はい
デフォルト	デフォルト値はデータベースによって異なります。
プロパティー・タイプ	String
使用法	これは、アクセスするデータベースの名前です。Oracle データベースの場合、これはデータベースを識別するシステム ID (SID) です。
グローバル化	はい
BIDI 対応	はい

データベース・ソフトウェア

このプロパティーは、アダプターがアクセスするデータベースを管理するデータベース管理ソフトウェアを指定します。

表 64. 「データベース・ソフトウェア」の詳細

行	説明
必須	はい
使用可能な値	このプロパティーは、一般的なデータベース・ソフトウェアを名前およびバージョン番号ごとにリストします。ご使用のソフトウェアがリストされていない場合は、「汎用 JDBC (Generic JDBC)」を選択してください。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードは、このプロパティーの値を使用して他のプロパティーのデフォルト値を設定し、データベース固有の選択リストを生成します。例えば、「DB2 USB Version 9.1」を選択すると、ウィザードの JDBC ドライバー・クラス・フィールドには、そのバージョンの DB2 UDB によってサポートされる JDBC ドライバーのみが表示されます。「Oracle 10」を選択すると、異なる JDBC ドライバーのセットが表示されます。
グローバル化	はい
BIDI 対応	はい

データベース URL

このプロパティーは、データベース接続を作成するための JDBC ドライバー固有の URL を指定します。

表 65. 「データベース URL」の詳細

必須	特定の JDBC ドライバーの場合は「はい」
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これは、使用するデータベース・ソフトウェアと JDBC ドライバーに固有の値です。 データベース・サーバーで IPv6 がサポートされている場合は、データベース URL のホスト名部分を IPv6 形式で指定できます。IP アドレスを大括弧 ([]) で囲んでください。

例	一般的なデータベース・サーバーの標準的な値を以下に示します。
	DB2 Universal (タイプ 4) JDBC ドライバー
	`jdbc:db2://www.example.com:50000/DB`
	DB2 Universal JDBC ドライバー (IPv6 アドレス)
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	DB2 Universal Database タイプ・ドライバー (ローカル接続用)
	`jdbc:db2:TEST`
	DB2 Universal Database タイプ 2 ドライバー (リモート接続用)
	`jdbc:db2://www.example.com:50000/TEST`
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

グローバル化	はい
BIDI 対応	はい

ホスト名

このプロパティーは、データベース・サーバーのホスト名または IP アドレスを指定します。

表 66. 「ホスト名」の詳細

行	説明
必須	はい
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーは、データベース・サーバーのホスト名または IP アドレスを指定します。データベース・サーバーにより IPv6 がサポートされている場合は、ホスト名を IPv6 形式で指定できます。
グローバル化	はい
BIDI 対応	はい

JDBC ドライバー・クラス名

このプロパティーは、JDBC ドライバー・クラスの名前を指定します。

表 67. 「JDBC ドライバー・クラス名」の詳細

行	説明
必須	はい
使用可能な値	使用可能な値は、データベースのタイプとバージョンによって異なります。ウィザードには、既知のドライバーのリストが表示されます。
デフォルト	デフォルトは、データベースのタイプとバージョンによって異なります。
プロパティー・タイプ	String
使用法	これは JDBC ドライバーのクラス名です。ウィザードには、選択した JDBC ドライバー・タイプのデフォルト・クラス名が表示されますが、必要に応じて別のクラス名を入力できます。クラス名は、ウィザードの開始時に指定した JDBC ドライバー・ファイルに記述されている必要があります。
グローバル化	はい
BIDI 対応	いいえ

JDBC ドライバー・タイプ

このプロパティーは、使用する JDBC ドライバーのタイプを指定します。

表 68. 「JDBC ドライバー・タイプ」の詳細

行	説明
必須	はい
使用可能な値	使用可能な値は、データベースのタイプとバージョンによって異なります。ウィザードには、既知のドライバーのリストが表示されます。
デフォルト	デフォルトは、データベースのタイプとバージョンによって異なります。

表 68. 「JDBC ドライバー・タイプ」の詳細 (続き)

行	説明
プロパティー・タイプ	String
使用法	これは、使用する JDBC ドライバーのタイプです。根本的な点は使用するドライバーがタイプ 2 とタイプ 4 (ユニバーサル) のどちらであるかですが、各データベース・システムでは、ドライバーにデータベース・システム固有の名前が使用されています。各データベース・システムの既知のドライバーのリストがウィザードに表示されます。使用するドライバーがリストにない場合は、「その他」を選択します。このフィールドの情報は、ウィザードの開始時に指定した JDBC ドライバー・ファイルと一致している必要があります。
グローバル化	はい
BIDI 対応	いいえ

パスワード (Password)

このプロパティーは、データベース・ユーザーのユーザー名に対するパスワードを指定します。

表 69. 「パスワード」の詳細

必須	DataSourceJNDIName または XADataSourceName プロパティーが設定されていない場合は必須。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを設定すると、DataSourceJNDIName または XADataSourceName プロパティーを使用してサーバーのデータ・ソースに指定されているパスワードが上書きされます。
グローバル化	はい
BIDI 対応	はい

ポート番号

このプロパティーは、データベース・インスタンスのポート番号を指定します。

表 70. 「ポート番号」の詳細

必須	はい
デフォルト	デフォルト値はデータベースによって異なります。また、デフォルト値はウィザードにより初期化されます。
プロパティー・タイプ	String
使用法	これは、データベース・インスタンスへ接続するポートのポート番号です。
グローバル化	はい
BIDI 対応	いいえ

ビジネス・オブジェクト名のプレフィックス

ビジネス・オブジェクトの名前に追加されるプレフィックス。

表 71. 「プレフィックス」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	プレフィックスを使用して、ビジネス・オブジェクトのタイプを容易に区別できるようにします。
例	汎用ビジネス・オブジェクトにプレフィックス JDBC を指定し、アプリケーション固有のビジネス・オブジェクトに %AppName% を指定する場合があります。
グローバル化	はい
BIDI 対応	いいえ

ユーザー名 (UserName)

このプロパティーは、データベース接続に使用するデータベース・ユーザー名を指定します。

表 72. 「ユーザー名」の詳細

必須	DataSourceJNDIName または XADataSourceName プロパティーが設定されていない場合は必須。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを設定すると、DataSourceJNDIName または XADataSourceName プロパティーを使用してサーバーのデータ・ソースに指定されているユーザー名が上書きされます。
グローバル化	はい
BIDI 対応	はい

リソース・アダプター・プロパティー

リソース・アダプター・プロパティーは、アダプターの一般的な操作 (ビジネス・オブジェクトのネーム・スペースの指定など) を制御します。リソース・アダプター・プロパティーは、アダプターの構成時に外部サービス・ウィザードを使用して設定します。アダプターのデプロイ後に、これらのプロパティーを変更するには、管理コンソールを使用します。

以下に示すロギングおよびトレースのプロパティーは、バージョン 6.1.0 には必要なくなりましたが、旧バージョンとの互換性を維持するためにサポートされています。

- ログ・ファイル最大サイズ
- ログ・ファイル名
- ログ・ファイル数
- トレース・ファイル最大サイズ
- トレース・ファイル名
- トレース・ファイル数

バージョン 6.0.2 で Inbound 処理と Outbound 処理の両方に対して指定されている次のプロパティーは、バージョン 6.1.0 では Inbound 処理にのみ適用されます。Outbound 処理に適用されるプロパティーは、管理接続ファクトリー・プロパティー・グループに属しています。

- Ping クエリー
- クエリー・タイムアウト
- ReturnDummyBOForSP

BusinessObjectNameSpaceプロパティーは、アクティベーション・スペック・プロパティーに移動しました。

リソース・アダプター・プロパティーとその目的を次の表に示します。各プロパティーの完全な説明は、表に続くセクションで説明します。プロパティー詳細表の見方について詳しくは、206 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 73. Adapter for JDBC 用のリソース・アダプター・プロパティー

名前		説明
ウィザード内	**管理コンソール内**	
ロギングおよびトレースで使用するアダプター ID	AdapterID	CEI イベントおよび PMI イベントのアダプター・インスタンスをロギングおよびトレースを基準にして識別します。
データベース・ベンダー	DatabaseVendor	アダプターが特殊な処理に使用するデータベースのタイプ
クエリー・タイムアウト	QueryTimeOut	すべての SQL ステートメントでの照会の最大実行時間 (秒数)
ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す	ReturnDummyBOForSP	結果セットが空の場合に出力パラメーターを返すかどうかを指定します。
(なし)	enableHASupport	このプロパティーは変更しないでください。
(なし)	ログ・ファイル最大サイズ	旧バージョンとの互換性を維持するためにサポートされています。
(なし)	LogFilename	旧バージョンとの互換性を維持するためにサポートされています。
(なし)	ログ・ファイル数	旧バージョンとの互換性を維持するためにサポートされています。
接続を検証するための SQL 照会	PingQuery	データベースへの接続の有効性をテストするのに使用する SQL 照会
(なし)	トレース・ファイル最大サイズ	旧バージョンとの互換性を維持するためにサポートされています。
(なし)	トレース・ファイル名	旧バージョンとの互換性を維持するためにサポートされています。
(なし)	トレース・ファイル数	旧バージョンとの互換性を維持するためにサポートされています。

ロギングおよびトレースで使用するアダプター ID (AdapterID)

このプロパティーは、アダプターの特定のデプロイメント (インスタンス) を識別する場合に使用します。

表 74. 「ロギングおよびトレースで使用するアダプター ID」の詳細

必須	はい
デフォルト	CWYBC_JDBC
プロパティー・タイプ	String
使用法	このプロパティーは、PMI イベントのアダプター・インスタンスを識別する場合に使用します。アダプターのインスタンスを複数デプロイする場合は、このプロパティーをアダプターのインスタンスごとに固有な値に設定します。 Inbound 処理の場合、このプロパティーはリソース・アダプター・プロパティーから取り出します。Outbound 処理の場合は、管理接続ファクトリー・プロパティーから取り出します。
グローバル化	はい
BIDI 対応	いいえ

データベース・ベンダー (DatabaseVendor)

このプロパティーは、使用されるデータベースのタイプを指定します。このタイプは、データベース・ベンダー名により決まります。

表 75. 「データベース・ベンダー」の詳細

必須	はい

使用可能な値	`DB2`
	`MSSQLServer`
	`Oracle`
	`Others`

デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	一部の SQL ステートメントでは特殊な処理が必要であり、この処理はデータベース・タイプに応じて異なります。例えば、Oracle の Struct データ型と Array データ型には特殊な処理が必要です。このプロパティーは、使用する RDBMS を指定します。これにより、データベース・タイプが決まります。 ご使用のデータベースのベンダーに対応する値を指定します。 • `DB2` (IBM DB2 データベース) • `Oracle` (Oracle データベース) • `MSSQLServer` (Microsoft SQL Server データベース) • `Others` (その他のすべてのデータベース・タイプ) その他のデータベース・タイプの場合、アダプターは特殊な処理を一切実行しません。JDBCDriverClass プロパティーに指定されているドライバーが正しいことを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

高可用性サポートを使用可能にする (Enable high availability support) (enableHASupport)

このプロパティーは変更しないでください。true に設定してください。

クエリー・タイムアウト (QueryTimeOut)

このプロパティーは、すべての SQL ステートメントでの照会の最大実行時間を秒数で指定します。

表 76. 「クエリー・タイムアウト」の詳細

必須	いいえ
デフォルト	デフォルト値なし
計測単位	秒
プロパティー・タイプ	整数
使用法	照会の処理に、指定された秒数より長い時間が必要な場合は、データベースにより、キャプチャーされる SQL 例外が生成されます。関連付けられているメッセージがログ・ファイルに記録されます。 値を指定しない場合は、照会のタイムアウトが設定されません。
グローバル化	はい
BIDI 対応	いいえ

ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)

このプロパティーは、結果セットが空の場合に出力パラメーターを返すかどうかを指定します。

表 77. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細

必須	いいえ
使用可能な値	True False
デフォルト	False
プロパティー・タイプ	Boolean
使用法	ストアード・プロシージャー取得 (RetrieveSP) 操作では、結果セットが戻されます。結果のセットが空であり、ReturnDummyBOForSP プロパティーが False に設定されている場合は、ビジネス・オブジェクトが作成されず、プロシージャー呼び出しから返される出力パラメーターを取得できません。 ただし、ReturnDummyBOForSP が True の場合は、ダミー・ビジネス・オブジェクトが作成され、対応する属性の出力パラメーターと入出力パラメーターの値が取り込まれます。
グローバル化	はい
BIDI 対応	いいえ

接続を検証するための SQL 照会 (PingQuery)

このプロパティーは、データベースへの接続の有効性をテストするために使用される SQL 照会を指定します。

表 78. 「ping クエリー (Ping query)」の詳細

必須	いいえ
プロパティー・タイプ	String
デフォルト	デフォルト値なし
使用法	このプロパティーには、アダプターがデータベースに接続できるかどうかを判断するために実行する SQL 照会ステートメントが含まれています。
グローバル化	いいえ
BIDI 対応	いいえ

アクティベーション・スペック・プロパティー

アクティベーション・スペック・プロパティーは、エクスポート用の Inbound イベント処理の構成情報を保持するプロパティーです。

アクティベーション・スペック・プロパティーは、アダプターの構成時に外部サービス・ウィザードを使用して設定します。また、WebSphere Integration Developer アセンブリー・エディターを使用して、またはデプロイメント後に WebSphere Process Server または WebSphere Enterprise Service Bus の管理コンソールを使用して変更できます。

以下の表に、アクティベーション・スペック・プロパティーとその説明を示します。各プロパティーの完全な説明は、表に続くセクションで説明します。後続セクションのプロパティー詳細表の見方について詳しくは、206 ページの『プロパティーの詳細についてのガイド』を参照してください。

表 79. Adapter for JDBC のアクティブ化仕様プロパティー

プロパティー名		
ウィザード内	**管理コンソール内**	**説明**
追加の JDBC ドライバー接続プロパティー	DriverConnectionProperties	JDBC ドライバーを使用してデータベースへ接続するときに使用される UserName および Password プロパティー以外の追加プロパティー
ビジネス・オブジェクト Namespace	BusinessObjectNameSpace	ビジネス・オブジェクト定義のネーム・スペース
カスタム削除照会 (Custom delete query)	CustomDeleteQuery	各イベントの処理後に実行され、イベントの送達後に削除可能なレコードを削除する照会、ストアード・プロシージャー、またはストアード関数の名前
カスタム・イベント照会 (Custom event query)	CustomEventQuery	イベントのポーリングを実行する照会、ストアード・プロシージャー、またはストアード関数の名前
カスタム更新照会 (Custom update query)	CustomUpdateQuery	各イベントの処理後に実行され、後続のイベント・サイクルで処理対象としてイベントが選択されることを防ぐ照会、ストアード・プロシージャー、またはストアード関数の名前

表 79. *Adapter for JDBC* のアクティブ化仕様プロパティー *(続き)*

プロパティー名		
ウィザード内	**管理コンソール内**	**説明**
DataSource JNDI 名	DataSourceJNDIName	データベース接続の確立に使用される JNDI データ・ソースの名前
データベース URL	DatabaseURL	データベースへの接続に使用されるデータベース URL
データベース・ベンダー	DatabaseVendor	アダプターが特殊な処理に使用するデータベースのタイプ
将来のタイム・スタンプを持つイベントを処理しない	FilterFutureEvents	アダプターが各イベントのタイム・スタンプをシステム時刻と比較することによって、将来のイベントをフィルターで除去するかどうかを指定します。
イベントを一度のみ送達する	AssuredOnceDelivery	アダプターにより、1 回のイベント送達を確保する機能が提供されるかどうかを指定します。
イベント順序	イベント順序	イベントが取得および処理される順序
イベント・クエリー・タイプ (Event query type)	EventQueryType	標準イベント・ストアまたはカスタム照会のいずれを使用するかを決定します
イベント・テーブル名	イベント・テーブル名	データベースが Inbound 処理のために生成するイベントを格納するデータベース・テーブルの名前
処理するイベント・タイプ	EventTypeFilter	どのイベントをアダプターが配信するかをアダプターに示す、区切り文字で区切られているイベント・タイプのリスト。
ポーリング期間の間隔	ポーリング間隔	ポーリング期間中にアダプターが待機する時間の長さ
JDBC ドライバー・クラス (JDBC driver class)	JDBCDriverClass	データベース接続に使用される JDBC ドライバーのクラス名
最大接続数 (Maximum connections)	MaximumConnections	アダプターが Inbound イベント送達に使用できる接続の最大数
最小接続数 (Minimum connections)	MinimumConnections	アダプターが Inbound イベント送達に使用できる接続の最小数
システム接続を再試行する回数	RetryLimit	エラーが発生したあと、アダプターが Inbound 接続の再確立を試行する回数。
パスワード	Password	データベースからイベントを検索する際にユーザーを認可するためのパスワード。
ポーリング数量 (Poll quantity)	ポーリング数量	各ポーリング期間中にアダプターがエクスポートに配信するイベント数
クエリー・タイムアウト	QueryTimeOut	すべての SQL ステートメントでの照会の最大実行時間 (秒数)
接続が失敗した場合の再試行間隔	RetryInterval	Inbound 操作時のエラー後、新規接続を確立しようとする試行間にアダプターが待機する時間の長さ
RetrieveSP に対しダミー・ビジネス・オブジェクトを戻す (Return dummy business object for RetrieveSP)	ReturnDummyBOForSP	結果セットが空の場合に出力パラメーターを返すかどうかを指定します。
「接続を検証するための SQL クエリー」	PingQuery	データベースへの接続の有効性をテストするのに使用する SQL 照会

表 79. Adapter for JDBC のアクティブ化仕様プロパティー (続き)

プロパティー名		説明
ウィザード内	管理コンソール内	
ポーリング時にエラーが検出された場合はアダプターを停止する	StopPollingOnError	ポーリング時にアダプターがエラーを検出した場合、アダプターがイベントのポーリングを停止するかどうかを指定します。
ポーリング後に実行するストアード・プロシージャー (Stored procedure to run after polling)	SPAfterPoll	各ポーリング周期の終了ごとに実行するストアード・プロシージャーの名前
ポーリング前に実行するストアード・プロシージャー (Stored procedure to run before polling)	SPBeforePoll	実際のポーリング照会の呼び出しの前に実行するストアード・プロシージャーの名前
送達のタイプ	DeliveryType	イベントがアダプターによってエクスポートに配信される順序を指定します。
ユーザー名	UserName	Inbound イベントに使用するデータベース・ユーザー名

追加の JDBC ドライバー接続プロパティー (DriverConnectionProperties)

このプロパティーには、JDBC ドライバーを使用したデータベースへの接続に関する追加情報が含まれています。

表 80. ドライバー接続プロパティーの詳細

必須	いいえ
使用可能な値	データベース接続プロパティーはデータベース固有です。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	これらの接続プロパティーは、UserName および Password プロパティーと共に使用されます。これにより、アダプターが使用するデータベース接続をカスタマイズできます。 接続プロパティーは、1 つ以上の *name*:*value* ペアとして指定し、ペアとペアの間はセミコロン (;) で区切ります。
例	このプロパティーに以下の値を設定すると、ログイン・タイムアウト間隔が指定され、データベース接続が読み取り専用になり、セキュリティー・メカニズムが設定されます。 `loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY`
グローバル化	はい
BIDI 対応	いいえ

ビジネス・オブジェクト Namespace (BusinessObjectNameSpace)

このプロパティーは、ビジネス・オブジェクト定義のネーム・スペースを指定します。

表 81. 「ビジネス・オブジェクト *Namespace*」プロパティーの特性

必須	はい
デフォルト	`http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc`
プロパティー・タイプ	String
使用法	この値は、ビジネス・オブジェクト名を論理的に分離するため、ビジネス・オブジェクト名の前に追加されます。
例	デフォルト・ネーム・スペースを使用した`Schema1Customer` ビジネス・オブジェクトの例は `http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/Schema1Customer` です。
BIDI 対応	いいえ

カスタム削除照会 (Custom delete query) (CustomDeleteQuery)

このプロパティーを使用して、イベントの送達後に削除可能なレコードを削除するために各イベントが処理された後に実行する、SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 82. 「カスタム削除照会 *(Custom delete query)*」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを使用して、EventQueryType プロパティーが `Dynamic` に設定されている場合に実行する SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。
グローバル化	はい
BIDI 対応	はい

カスタム・イベント照会 (Custom event query) (CustomEventQuery)

このプロパティーを使用して、カスタム・イベント処理においてイベントをポーリングするために実行する、SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 83. 「カスタム・イベント照会 *(Custom event query)*」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを使用して、EventQueryType プロパティーが `Dynamic` に設定されている場合の各ポーリング・サイクル中に実行する、SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 83. 「カスタム・イベント照会 (Custom event query)」の詳細 (続き)

例	次の例において、カスタム・イベント照会は、状況列の値が 0 になっている MY_EVENT_TABLE イベント・ストアにあるすべてのレコードのイベント ID、オブジェクト・キー、およびオブジェクト名を返す SQL ステートメントを実行します。
	`select event_id, object_key, object_name from MY_EVENT_TABLE where status = 0`
	次の例では、Oracle データベースの場合に限り、返されるイベント・レコードを PollQuantity プロパティーの値に制限します。
	`select event_id, object_key, object_name from MY_EVENT_TABLEwhere status = 0 and rownum < POLL QUANTITY`
	次の例では、2 つのパラメーターを持つストアード・プロシージャーを実行します。
	`CALL MY_EVENT_STORED_PROC (?,?)`
	以下の例では、1 つのパラメーターと 1 つの戻り値を持つストアード関数を実行します。
	`? = CALL MY_EVENT_FUNCTION(?)`

グローバル化	はい
BIDI 対応	はい

カスタム更新照会 (Custom update query) (CustomUpdateQuery)

このプロパティーを使用して、同じイベントが後続のイベント周期で処理対象として取り出されないように、各イベントの処理後に実行する SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。

表 84. 「カスタム更新照会 (Custom update query)」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを使用して、EventQueryType プロパティーが Dynamic に設定されている場合に実行する SQL ステートメント、ストアード・プロシージャー、またはストアード関数を指定します。
グローバル化	はい
BIDI 対応	はい

データ・ソース JNDI 名 (DataSourceJNDIName)

このプロパティーは、データベース接続を確立するときに使用される JNDI データ・ソースの名前を指定します。

表 85. 「データ・ソース JNDI 名」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 85. 「データ・ソース JNDI 名」の詳細 (続き)

使用法	このプロパティーを使用して、ターゲット・データベースの接続情報を指定する WebSphere Process Server または WebSphere Enterprise Service Bus のデータ・ソースの JNDI 名を指定します。 Inbound 操作および Outbound 操作のパフォーマンスを改善するには、準備済みステートメントをキャッシュするために使用可能に設定されているデータ・ソースの名前を指定します。 ユーザー名プロパティーとパスワード・プロパティーも設定されている場合、データ・ソースのユーザー名とパスワードはこれらのプロパティーにより上書きされます。 データベース接続プロパティーは次の順序で使用されます。 1. DataSourceJNDIName プロパティーが設定されている場合、アダプターはこのプロパティーを使用してデータベースへの接続を確立します。 UserName プロパティーと Password プロパティーも設定されている場合は、データ・ソースで設定されているユーザー名とパスワードはこれらのプロパティーにより上書きされます。 2. DataSourceJNDIName プロパティーが設定されておらず、XADataSourceName プロパティーと XADatabaseName プロパティーが設定されている場合は、アダプターはこれらのプロパティーを使用して接続を確立します。 DataSourceJNDIName プロパティーは、XA データ・ソースまたは接続プール・データ・ソースを表します。 XA トランザクションをサポートするサーバーで JNDI データ・ソースを定義し、その後アダプターの構成時にそのデータ・ソースを指定する場合は、XA トランザクションをサポートするすべてのタイプのデータベースに接続できます。XA データ・ソースとデータベースを使用する場合は、アダプターは DB2 および Oracle データベースでのみ XA トランザクションをサポートします。 3. DataSourceJNDIName、XADataSourceName、および XADatabaseName プロパティーがすべて設定されていない場合は、アダプターは DatabaseURL、JDBCDriverClass、UserName、および Password プロパティーを使用して接続を確立します。 データ・ソース JNDI 名前プロパティーを、管理接続ファクトリーの JNDI 名、またはサーバーのアクティベーション・スペックと混同しないでください。以下のリストに、JNDI 名のタイプ間での重要な違いを示します。 • データ・ソース JNDI 名 – データベースへの接続を指定する – ユーザー名とパスワードをアダプターのプロパティーに保存する代わりに使用される – アダプター・プロパティーとして保存される • 管理接続ファクトリーまたはアクティベーション・スペックの JNDI 名 – サーバーの管理接続ファクトリーまたはアクティベーション・スペックへの接続を指定する – ウィザードで各管理接続ファクトリーまたはアクティベーション・スペック・プロパティーの値を指定する代わりに使用される – インポート・ファイルの接続ターゲットとして保存される
グローバル化	はい
BIDI 対応	いいえ

データベース URL (DatabaseURL)

このプロパティーは、データベース接続を作成するための JDBC ドライバー固有の URL を指定します。

表 86. 「データベース URL」の詳細

必須	次のいずれかのプロパティー・セットが設定されている場合を除き必須。 • DataSourceJNDIName プロパティー • XADataSourceName および XADatabaseName プロパティー
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードのデータベース固有のフィールドに情報を入力し、データベース URL を作成します。例えば、DB2 データベースのデータベース URL は、データベース名、サーバー・ホスト名、およびデータベース・ポート番号で構成されています。管理コンソールで、データベース URL 値全体を入力します。 データベース・サーバーで IPv6 がサポートされている場合は、データベース URL のホスト名部分を IPv6 形式で指定できます。 データベース接続プロパティーは次の順序で使用されます。 1. DataSourceJNDIName プロパティーが設定されている場合、アダプターはこのプロパティーを使用してデータベースへの接続を確立します。 UserName プロパティーと Password プロパティーも設定されている場合は、データ・ソースで設定されているユーザー名とパスワードはこれらのプロパティーにより上書きされます。 2. DataSourceJNDIName プロパティーが設定されておらず、XADataSourceName プロパティーと XADatabaseName プロパティーが設定されている場合は、アダプターはこれらのプロパティーを使用して接続を確立します。 DataSourceJNDIName プロパティーは、XA データ・ソースまたは接続プール・データ・ソースを表します。 XA トランザクションをサポートするサーバーで JNDI データ・ソースを定義し、その後アダプターの構成時にそのデータ・ソースを指定する場合は、XA トランザクションをサポートするすべてのタイプのデータベースに接続できます。XA データ・ソースとデータベースを使用する場合は、アダプターは DB2 および Oracle データベースでのみ XA トランザクションをサポートします。 3. DataSourceJNDIName、XADataSourceName、および XADatabaseName プロパティーがすべて設定されていない場合は、アダプターは DatabaseURL、JDBCDriverClass、UserName、および Password プロパティーを使用して接続を確立します。 ホスト名を IPv6 形式の IP アドレスとして指定する場合は、その IP アドレスを大括弧 ([]) で囲んでください。

表 86. 「データベース URL」の詳細 (続き)

例	一般的なデータベース・サーバーの標準的な値を以下に示します。
	DB2 Universal (タイプ 4) JDBC ドライバー
	`jdbc:db2://www.example.com:50000/DB`
	DB2 Universal JDBC ドライバー (IPv6 アドレス)
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	DB2 Universal Database タイプ 2 ドライバー (ローカル接続用)
	jdbc:db2:TEST
	DB2 Universal Database タイプ 2 ドライバー (リモート接続用)
	jdbc:db2://www.example.com:50000/TEST
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

グローバル化	はい
BIDI 対応	はい

データベース・ベンダー (DatabaseVendor)

このプロパティーは、使用されるデータベースのタイプを指定します。このタイプは、データベース・ベンダー名により決まります。

表 87. 「データベース・ベンダー」の詳細

必須	はい

使用可能な値	`DB2`
	`MSSQLServer`
	`Oracle`
	`Others`

デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	一部の SQL ステートメントでは特殊な処理が必要であり、この処理はデータベース・タイプに応じて異なります。例えば、Oracle の Struct データ型と Array データ型には特殊な処理が必要です。このプロパティーは、使用する RDBMS を指定します。これにより、データベース・タイプが決まります。 ご使用のデータベースのベンダーに対応する値を指定します。 • `DB2` (IBM DB2 データベース) • `Oracle` (Oracle データベース) • `MSSQLServer` (Microsoft SQL Server データベース) • `Others` (その他のすべてのデータベース・タイプ) その他のデータベース・タイプの場合、アダプターは特殊な処理を一切実行しません。JDBCDriverClass プロパティーに指定されているドライバーが正しいことを確認してください。
グローバル化	いいえ
BIDI 対応	いいえ

送達タイプ (DeliveryType)

このプロパティーでは、イベントがアダプターによってエクスポートに配信される順序を指定します。

表 88. 「送達タイプ」の詳細

必須	いいえ
使用可能な値	ORDERED UNORDERED
デフォルト	ORDERED
プロパティー・タイプ	String
使用法	以下の値がサポートされています。 • ORDERED: アダプターは、一度に 1 つのイベントをエクスポートに配信します。 • UNORDERED: アダプターは、一度にすべてのイベントをエクスポートに配信します。
グローバル化	いいえ
BIDI 対応	いいえ

将来のタイム・スタンプを持つイベントを処理しない (FilterFutureEvents)

このプロパティーでは、アダプターが各イベントのタイム・スタンプをシステム時刻と比較することによって、将来のイベントをフィルターで除去するかどうかを指定します。

表 89. 「将来のタイム・スタンプを持つイベントを処理しない」の詳細

必須	はい
使用可能な値	True False
デフォルト	False
プロパティー・タイプ	Boolean
使用法	True に設定すると、アダプターは各イベントの時刻をシステム時刻と比較します。イベント時刻がシステム時刻より後の時刻である場合、そのイベントは配信されません。 False に設定すると、アダプターはすべてのイベントを配信します。
グローバル化	いいえ
BIDI 対応	いいえ

イベントを一度のみ送達する (AssuredOnceDelivery)

このプロパティーでは、Inbound イベントに対して、「イベントを一度のみ送達する」の機能を提供するかどうかを指定します。

表 90. 「イベントを一度のみ送達する」の詳細

必須	はい

表 90. 「イベントを一度のみ送達する」の詳細 (続き)

使用可能な値	`True` `False`
デフォルト	`True`
プロパティー・タイプ	Boolean
使用法	このプロパティーを `True` に設定すると、アダプターにより、1 回のイベント送達を確保する機能が提供されます。つまり、各イベントは 1 回のみ配信されます。値を `False` にすると、1 回のイベント送達を確保する機能は提供されませんが、パフォーマンスは向上します。 このプロパティーを `True` に設定すると、アダプターにより、トランザクション (XID) 情報のイベント・ストアへの保管が試行されます。このプロパティーを `False` に設定した場合は、アダプターではこの情報の保管は行われません。 このプロパティーは、エクスポート・コンポーネントがトランザクションの対象である場合にのみ使用されます。そうでない場合は、このプロパティーの値に関係なく、トランザクションを使用することはできません。
グローバル化	いいえ
BIDI 対応	いいえ

イベント順序 (EventOrderBy)

イベントが取得および処理される順序。

表 91. 「イベント順序」の詳細

必須	いいえ
使用可能な値	イベント・ストアの列名をコンマ (,) で区切ったリストと、順序属性 `asc` および `desc`
デフォルト	`event_time, event_priority`
プロパティー・タイプ	String
使用法	イベント・ストアの列名をコンマで区切って記述したリストと、オプションの属性 (昇順または降順) を指定します。

例	並べ替えの第 1 条件として時刻、第 2 条件として優先順位を使用して並べ替えられたイベントを表示するには、次のように指定します。
	`event_time, event_priority`
	
	並べ替えの第 1 条件としてオブジェクト名 (昇順)、第 2 条件として時間 (降順) を使用して並べ替えられたイベントを表示するには、次のように指定します。
	`object_name asc, event_time desc`

グローバル化	はい
BIDI 対応	はい

イベント・クエリー・タイプ (Event query type) (EventQueryType)

このプロパティーは、標準照会処理とカスタム照会処理のどちらを使用するかを指定します。

表 92. 「イベント・クエリー・タイプ (Event query type)」の詳細

必須	はい
使用可能な値	`Standard` `Dynamic`
デフォルト	`Standard`
プロパティー・タイプ	String
使用法	有効な値は、標準イベント処理用の `Standard` とカスタム・イベント処理用の `Dynamic` です。 このプロパティーが `Dynamic` に設定されていると、CustomEventQuery、CustomUpdateQuery、および CustomDeleteQuery プロパティーが使用されます。このプロパティーが `Standard` に設定されていると、これらのプロパティーは無視されます。
グローバル化	いいえ
BIDI 対応	いいえ

イベント・テーブル名 (EventTableName)

このプロパティーは、Inbound 処理に使用されるイベント・ストアが格納されているターゲット・データベース内のテーブルの名前を指定します。

表 93. 「イベント・テーブル名」の詳細

必須	はい
デフォルト	`WBIA_JDBC_EventStore`
プロパティー・タイプ	String
使用法	アダプターの構成を開始する前に、イベント・ストアを作成してください。 標準イベント処理では、イベントはトリガーまたはその他のメカニズムを介してデータベースにより生成されます。カスタム照会処理では、アダプターはカスタム照会の結果を受信すると、イベントをイベント・ストアに保存します。
グローバル化	はい
BIDI 対応	はい

処理するイベント・タイプ (EventTypeFilter)

このプロパティーには、どのイベントをアダプターが配信するかをアダプターに示す、区切り文字で区切られているイベント・タイプのリストが入っています。

表 94. 「処理するイベント・タイプ」の詳細

必須	いいえ
使用可能な値	ビジネス・オブジェクト・タイプのコンマ (,) 区切りのリスト
デフォルト	`NULL`
プロパティー・タイプ	String

表 94. 「処理するイベント・タイプ」の詳細 (続き)

使用法	イベントは、ビジネス・オブジェクト・タイプ別にフィルタリングされます。このプロパティーを設定すると、アダプターは、リスト内に存在するイベントのみを配信するようになります。値が `null` の場合は、フィルターが適用されず、すべてのイベントはエクスポートに配信されることを示しています。 このプロパティーは、標準のイベント処理に対してのみ適用されます。カスタムのイベント処理の場合は、カスタムのイベント照会により、必要なフィルタリングを実行する必要があります。
例	Customer ビジネス・オブジェクトおよび Order ビジネス・オブジェクトに関連するイベントのみを受信するには、次の値を指定します。 `Customer,Order`
グローバル化	いいえ
BIDI 対応	いいえ

JDBC ドライバー・クラス (JDBC driver class) (JDBCDriverClass)

このプロパティーは、データベース接続に使用される JDBC ドライバーのクラス名を指定します。

表 95. 「JDBC ドライバー・クラス (JDBC driver class)」の詳細

必須	DataSourceJNDIName プロパティーが設定されていない場合は必須
使用可能な値	値はデータベースによって異なります。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	外部サービス・ウィザードでは、共通データベース・ソフトウェアとドライバー (IBM DB2、Oracle、および MicrosoftSQL の最新バージョンのタイプ 4 ドライバーなど) の組み合わせを選択すると、ドライバーが自動的に指定されます。ほとんどのデータベース・ソフトウェアのほとんどのタイプ 2 ドライバーでは、データベース・クラス名を指定する必要があります。 タイプ 2 ドライバーまたは汎用ドライバーを選択する場合は、JDBC ドライバー・クラス名を入力する必要があります。例えば、DB2 Universal Database タイプ 2 ドライバーの場合、クラス名は `COM.ibm.db2.jdbc.app.DB2Driver` です。 管理コンソールで、ドライバーのデータベース固有名を入力してください。 DataSourceJNDIName プロパティーが設定されている場合、このプロパティーは無視されます。

表 95. 「JDBC ドライバー・クラス (JDBC driver class)」の詳細 (続き)

例	**外部サービス・ウィザード内:** • ユニバーサル (タイプ 4) JDBC ドライバーを使用して DB2 データベースに接続するには、「IBM DB2 Universal」を選択します。 • DB2 ユニバーサルのタイプ 2 ドライバーを使用して DB2 データベースに接続するには、「Other」を選択します。 • タイプ 4 ドライバーを使用して Oracle 10 データベースに接続するには、「Oracle Thin Driver」を選択します。 **管理コンソール内** **DB2 Universal Database タイプ 2 ドライバー** COM.ibm.db2.jdbc.app.DB2Driver **DB2 Universal Database タイプ 4 ドライバー** com.ibm.db2.jcc.DB2Driver **Oracle Thin JDBC ドライバー** oracle.jdbc.driver.OracleDriver **IBM Toolkit for Java リモート・ドライバー (i5/OS 用)** com.ibm.as400.access.AS400JDBCDriver **IBM WebSphere Connect JDBC ドライバー (Microsoft SQL Server 用)** com.ibm.websphere.jdbc.sqlserver.SQLServerDriver
グローバル化	いいえ
BIDI 対応	いいえ

最大接続数 (Maximum connections) (MaximumConnections)

このプロパティーでは、アダプターが Inbound イベント送達に使用できる接続の最大数を指定します。

表 96. 「最大接続数 (Maximum connections)」の詳細

必須	いいえ
デフォルト	1
プロパティー・タイプ	Integer
使用法	正の値のみが有効です。アダプターは、1 より小さい正の入力値を 1 であるとみなします。このプロパティーに対して負の値または 1 を入力すると、実行時エラーが発生することがあります。
グローバル化	いいえ
BIDI 対応	いいえ

最小接続数 (Minimum connections) (MinimumConnections)

このプロパティーでは、アダプターが Inbound イベント送達に使用できる接続の最小数を指定します。

表 97. 「最小接続数 (Minimum connections)」の詳細

必須	いいえ

表 97. 「最小接続数 (Minimum connections)」の詳細 (続き)

デフォルト	1
プロパティー・タイプ	Integer
使用法	正の値のみが有効です。1 より小さい値は、アダプターによって 1 として処理されます。このプロパティーに対して負の値または 1 を入力すると、実行時エラーが発生することがあります。
グローバル化	いいえ
BIDI 対応	いいえ

パスワード (Password)

このプロパティーは、データベース・ユーザーのユーザー名に対するパスワードを指定します。

表 98. 「パスワード」の詳細

必須	DataSourceJNDIName または XADataSourceName プロパティーが設定されていない場合は必須。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを設定すると、DataSourceJNDIName または XADataSourceName プロパティーを使用してサーバーのデータ・ソースに指定されているパスワードが上書きされます。
グローバル化	はい
BIDI 対応	はい

接続を検証するための SQL 照会 (PingQuery)

このプロパティーは、データベースへの接続の有効性をテストするために使用される SQL 照会を指定します。

表 99. 「ping クエリー (Ping query)」の詳細

必須	いいえ
プロパティー・タイプ	String
デフォルト	デフォルト値なし
使用法	このプロパティーには、アダプターがデータベースに接続できるかどうかを判断するために実行する SQL 照会ステートメントが含まれています。
グローバル化	いいえ
BIDI 対応	いいえ

ポーリング期間の間隔 (PollPeriod)

このプロパティーでは、ポーリング期間中にアダプターが待機する時間の長さを指定します。

表 100. 「ポーリング期間の間隔」の詳細

必須	はい

表 100. 「ポーリング期間の間隔」の詳細 (続き)

使用可能な値	0 以上の整数
デフォルト	2000
計測単位	ミリ秒
プロパティー・タイプ	Integer
使用法	ポーリング期間は一定の割合で確立されます。つまり、ポーリング周期の実行が何らかの理由で遅延すると (例えば、前のポーリング周期が完了するまでに予想より時間がかかった場合)、遅延によって失った時間を取り戻すために次のポーリング周期がすぐに開始されます。
グローバル化	いいえ
BIDI 対応	いいえ

ポーリング期間内の最大イベント数 (PollQuantity)

このプロパティーでは、各ポーリング期間中にアダプターがエクスポートに配信するイベント数を指定します。

表 101. 「ポーリング期間内の最大イベント数」の詳細

必須	はい
デフォルト	10
プロパティー・タイプ	Integer
使用法	値は 0 より大きくする必要があります。この値を大きくすると、ポーリング期間ごとに処理される イベントの数が増加し、アダプターのパフォーマンス効率が低下する場合があります。この値を小さくすると、ポーリング期間ごとに処理される イベントの数が減少し、アダプターのパフォーマンスが若干向上することがあります。
グローバル化	いいえ
BIDI 対応	いいえ

クエリー・タイムアウト (QueryTimeOut)

このプロパティーは、すべての SQL ステートメントでの照会の最大実行時間を秒数で指定します。

表 102. 「クエリー・タイムアウト」の詳細

必須	いいえ
デフォルト	デフォルト値なし
計測単位	秒
プロパティー・タイプ	整数
使用法	照会の処理に、指定された秒数より長い時間が必要な場合は、データベースにより、キャプチャーされる SQL 例外が生成されます。関連付けられているメッセージがログ・ファイルに記録されます。 値を指定しない場合は、照会のタイムアウトが設定されません。
グローバル化	はい
BIDI 対応	いいえ

接続が失敗した場合の再試行間隔 (RetryInterval)

このプロパティーでは、アダプターが Inbound 接続に関連したエラーを検出した場合に、アダプターが新規接続を確立しようとするまで待機する時間の長さを指定します。

表 103. 再試行間隔の詳細

必須	はい
デフォルト	2000
計測単位	ミリ秒
プロパティー・タイプ	Integer
使用法	正の値のみが有効です。このプロパティーでは、アダプターが Inbound 接続に関連したエラーを検出した場合に、アダプターが新規接続を確立しようとするまで待機する時間の長さを指定します。
グローバル化	はい
BIDI 対応	いいえ

システム接続を再試行する回数 (RetryLimit)

このプロパティーでは、アダプターが Inbound 接続の再確立を試行する回数を指定します。

表 104. 「システム接続を再試行する回数」の詳細

必須	いいえ
使用可能な値	正整数
デフォルト	0
プロパティー・タイプ	Integer
使用法	正の値のみが有効です。 このプロパティーでは、アダプターが Inbound 接続に関連したエラーを検出した場合に、アダプターが接続を再開しようとする回数を指定します。値を 0 にすると、再試行回数は無限になります。
グローバル化	はい
BIDI 対応	いいえ

ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す (ReturnDummyBOForSP)

このプロパティーは、結果セットが空の場合に出力パラメーターを返すかどうかを指定します。

表 105. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細

必須	いいえ

表 105. 「ストアード・プロシージャーの結果セットが空の場合にもビジネス・オブジェクトを返す」の詳細 (続き)

使用可能な値	True False
デフォルト	False
プロパティー・タイプ	Boolean
使用法	ストアード・プロシージャー取得 (RetrieveSP) 操作では、結果セットが戻されます。結果のセットが空であり、ReturnDummyBOForSP プロパティーが False に設定されている場合は、ビジネス・オブジェクトが作成されず、プロシージャー呼び出しから返される出力パラメーターを取得できません。 ただし、ReturnDummyBOForSP が True の場合は、ダミー・ビジネス・オブジェクトが作成され、対応する属性の出力パラメーターと入出力パラメーターの値が取り込まれます。
グローバル化	はい
BIDI 対応	いいえ

ポーリング時にエラーが検出された場合はアダプターを停止する (StopPollingOnError)

このプロパティーでは、ポーリング時にアダプターがエラーを検出した場合、アダプターがイベントのポーリングを停止するかどうかを指定します。

表 106. 「ポーリング時にエラーが検出された場合はアダプターを停止する」の詳細

必須	いいえ
使用可能な値	True False
デフォルト	False
プロパティー・タイプ	Boolean
使用法	このプロパティーを True に設定した場合、アダプターはエラーを検出するとポーリングを停止します。 このプロパティーを False に設定した場合、アダプターはポーリング時にエラーを検出すると例外をログに記録し、ポーリングを続行します。
グローバル化	いいえ
BIDI 対応	いいえ

ポーリング後に実行するストアード・プロシージャー (SPAfterPoll)

このプロパティーは、各ポーリング周期の完了後に実行するストアード・プロシージャーまたはストアード関数の名前を指定します。

表 107. 「ポーリング後に実行するストアード・プロシージャー」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String

表 107. 「ポーリング後に実行するストアード・プロシージャー」の詳細 (続き)

使用法	ストアード・プロシージャーは、ポーリング数量のパラメーターを 1 つだけ取ります。
グローバル化	はい
BIDI 対応	はい

ポーリング前に実行するストアード・プロシージャー (SPBeforePoll)

このプロパティーは、実際のポーリング照会の呼び出し前に実行するストアード・プロシージャーまたはストアード関数の名前を指定します。

表 108. 「ポーリング前に実行するストアード・プロシージャー」の詳細

必須	いいえ
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	ストアード・プロシージャーは、ポーリング数量のパラメーターを 1 つだけ取ります。
グローバル化	はい
BIDI 対応	はい

ユーザー名 (UserName)

このプロパティーは、データベース接続に使用するデータベース・ユーザー名を指定します。

表 109. 「ユーザー名」の詳細

必須	DataSourceJNDIName または XADataSourceName プロパティーが設定されていない場合は必須。
デフォルト	デフォルト値なし
プロパティー・タイプ	String
使用法	このプロパティーを設定すると、DataSourceJNDIName または XADataSourceName プロパティーを使用してサーバーのデータ・ソースに指定されているユーザー名が上書きされます。
グローバル化	はい
BIDI 対応	はい

グローバリゼーション

WebSphere Adapter for JDBC は、複数の言語および国/地域別環境で使用することができる、国際化されたアプリケーションです。アダプターは、文字セット・サポートおよびホスト・サーバーのロケールに基づいて、メッセージ・テキストを適切な言語で送信します。アダプターは、統合コンポーネント間の双方向スクリプト・データの変換をサポートします。

グローバリゼーションおよび双方向変換

アダプターは、1 バイト文字セットとマルチバイト文字セットをサポートし、メッセージ・テキストを指定された言語で配信できるようにグローバル化されています。アダプターは双方向変換も実行します。双方向変換とは、1 つのファイルに左

から右 (ヘブライ語やアラビア語など) と右から左 (URL やファイル・パスなど) の両方の意味内容を含むデータを処理するタスクを指します。

グローバリゼーション

グローバル化されたソフトウェア・アプリケーションは、言語環境や国/地域別環境が単一ではなく複数の環境で使用することを目的として設計され、開発されています。WebSphere Adapters、WebSphere Integration Developer、WebSphere Process Server、および WebSphere Enterprise Service Bus は、Java で作成されています。Java 仮想マシン (JVM) 内の Java ランタイム環境は、 Unicode 文字コード・セットでデータを表現します。Unicode には、最もよく知られた文字コード・セット (単一バイトとマルチバイトの両方) の文字エンコードが含まれています。そのため、これらの統合システム・コンポーネント間でデータを転送するときに文字を変換する必要はありません。

エラー・メッセージや通知メッセージを適切な言語や個々の国や地域に合った形でログに記録するために、アダプターは稼働先システムのロケールを使用します。

双方向変換

アラビア語やヘブライ語などの言語は右から左に書きますが、テキストには左から右に書かれる部分も埋め込まれるため、双方向スクリプトになります。ソフトウェア・アプリケーションが双方向スクリプトを表示および処理する方法は複数あります。 WebSphere Process Server と WebSphere Enterprise Service Bus では、Windows 標準形式が使用されますが、WebSphere Process Server または WebSphere Enterprise Service Bus とデータを交換するエンタープライズ情報システムでは他の形式を使用できます。WebSphere Adapters は、2 つのシステム間でやり取りされる双方向スクリプト・データの変換を行うことによって、トランザクションの両側でデータが正確に処理および表示されるようにします。

双方向形式

WebSphere Process Server および WebSphere Enterprise Service Bus は、ILYNN (暗黙、左から右、オン、オフ、公称) の双方向形式を使用します。これは Windows によって使用される形式です。エンタープライズ情報システムが別の形式を使用する場合、アダプターは、データを WebSphere Process Server または WebSphere Enterprise Service Bus に導入する前に形式を変換します。

双方向形式は 5 つの属性から構成されます。双方向プロパティーを設定する場合は、これらの各属性に値を割り当ててください。属性および設定値を以下の表に示します。

表 110. 双方向形式の属性

文字の位置	目的	値	説明	デフォルト設定
1	スキーマの配列	I または V	暗黙 (Implicit) (論理 (Logical))、または表示 (Visual)	I

表 110. 双方向形式の属性 (続き)

文字の位置	目的	値	説明	デフォルト設定
2	方向	L R C D	左から右 右から左 コンテキスト LTR コンテキスト RTL	L
3	対称スワッピング	Y または N	対称スワッピングがオンまたはオフ	Y
4	シェーピング	S N I M F B	テキストの形状を指定する (Text is shaped) テキストの形状を指定しない (Text is not shaped) 語頭形 (Initial shaping) 語中形 (Middle shaping) 語尾形 (Final shaping) 独立形 (Isolated shaping)	N
5	数字シェーピング	H C N	ヒンディ語 コンテキスト 公称	N

アダプターは、データを左から右の論理形式に変換してから WebSphere Process Server または WebSphere Enterprise Service Bus に送信します。

双方向プロパティーの使用

複数の双方向プロパティーを使用して、コンテンツ・データとメタデータの両方の変換を制御できます。特別な双方向プロパティーを設定してコンテンツ・データまたはメタデータのいずれかを双方向変換から除外するか、変換中に特別な処理が必要なデータを識別できます。

以下の表で、4 種類の双方向プロパティーについて説明します。

表 111. 双方向プロパティーのタイプ

プロパティー・タイプ	データ変換
EIS	コンテンツ・データ (エンタープライズ情報システム、つまりデータベースによって送信されるデータ) の形式を制御します。
メタデータ	メタデータ (コンテンツ・データに関する情報を提供するデータ) の形式を制御します。
スキップ	変換から除外するコンテンツまたはメタデータを識別します。
特殊形式	変換処理中に異なる処理を必要とするファイル・パスまたは URL などの特定のテキストを識別する。コンテンツ・データまたはメタデータのいずれかに設定できる。

以下の領域で双方向変換を制御するプロパティーを設定できます。

- **リソース・アダプター・プロパティー:** これらのプロパティーは、TurnBiDiOff プロパティー (アダプター・インスタンスが双方向変換を実行するかどうかを制御します) などのデフォルト構成設定を格納します。これらのプロパティーを構成するには、サーバーの管理コンソールを使用します。
- **管理接続ファクトリー・プロパティー:** これらのプロパティーは、エンタープライズ情報システムとの Outbound 接続インスタンスを作成するためにランタイムに使用されます。管理接続ファクトリー・プロパティーは作成された後、デプロイメント記述子に格納されます。
- **アクティベーション・スペック・プロパティー:** これらのプロパティーは、メッセージ・エンドポイントに対する Inbound イベント処理構成情報を保持します。外部サービス・ウィザードを使用するか、またはサーバーの管理コンソールを使用して設定します。

プロパティーのスコープと検索機構

アダプターの双方向プロパティーに値を設定すると、アダプターは双方向変換を実行します。実行時には、プロパティー設定の階層の継承と、検索機構に依存するロジックを使用します。

リソース・アダプター内で定義されたプロパティーは階層のトップにありますが、他の領域で定義されたプロパティーまたはビジネス・オブジェクト内で注釈が付けられたプロパティーは下位の階層にあります。このため、例えば、リソース・アダプターの EIS タイプの双方向プロパティーのみに値を設定すると、Inbound (アクティベーション・スペック) トランザクションと Outbound (管理接続ファクトリー) トランザクションのいずれで発生するかにかかわらず、定義された EIS タイプの双方向プロパティーを必要とする変換によってそれらの値が継承および使用されます。

しかし、リソース・アダプターとアクティベーション・スペックの両方の EIS タイプの双方向プロパティーに値を設定した場合、Inbound トランザクションから発生した変換は、アクティベーション・スペックに設定された値を使用します。

処理ロジックでは、変換時に使用する双方向プロパティーの値を、検索機構を使用して検索します。検索機構は、変換が生じるレベルから検索を開始し、適切なプロパティー・タイプを持つ定義済みの値を対象に、階層の上位に向かって検索を進めます。検出された最初の有効値が使用されます。階層の検索は、子から親の方向にのみ進行します。兄弟は検索の対象になりません。

双方向データ変換で使用可能なプロパティー

WebSphere Adapter for JDBC には、双方向データ変換で使用可能な各種構成プロパティーがあります。

アダプターは、クライアント・アプリケーションとデータベースの間での双方向データの交換を有効にします。これは、データベースのデータの双方向形式が、ランタイム環境で使用される双方向形式と異なる場合でも有効になります。アダプターの構成時、およびビジネス・オブジェクトのアプリケーション固有情報では、双方向文字を使用できます。双方向サポートで使用可能なプロパティーとアプリケーション固有情報を以下に示します。

- 構成プロパティー

  - アクティベーション・スペック・プロパティー
  - 外部サービス・ウィザードの接続プロパティー
  - 管理接続ファクトリー・プロパティー
- アプリケーション固有情報
  - ビジネス・オブジェクト・レベル ASI
  - 操作レベル ASI
  - 属性レベル ASI

以降のセクションでは、双方向変換で使用可能な特定の構成プロパティーとアプリケーション固有情報をリストします。

ウィザードで使用される接続プロパティー

双方向スクリプト・データ変換で使用可能な、外部サービス・ウィザードの接続プロパティーを以下に示します。

- ユーザー名
- パスワード

管理接続ファクトリー・プロパティー

双方向スクリプト・データ変換で使用可能な管理接続プロパティーを以下に示します。

- 追加の JDBC ドライバー接続プロパティー
- データベース URL
- パスワード
- ユーザー名
- XA データベース名

アクティベーション・スペック・プロパティー

双方向スクリプト・データ変換で使用可能なアクティベーション・スペック・プロパティーを以下に示します。

- カスタム削除照会 (Custom delete query)
- カスタム・イベント照会 (Custom event query)
- カスタム更新照会 (Custom update query)
- 追加の JDBC ドライバー接続プロパティー
- データベース URL
- イベント順序
- イベント・テーブル名
- パスワード
- ポーリング前に実行するストアード・プロシージャー (Stored procedure to run before polling)
- ポーリング後に実行するストアード・プロシージャー (Stored procedure to run after polling)
- ユーザー名

ビジネス・オブジェクトのアプリケーション固有情報

双方向スクリプト・データ変換で使用可能なビジネス・オブジェクトのアプリケーション固有情報パラメーターを以下に示します。

- TableName
- StatusColumnName
- SPName
- SelectStatement

操作に関するアプリケーション固有情報

双方向スクリプト・データ変換で使用可能な操作のアプリケーション固有情報パラメーターを以下に示します。

- StoredProcedureName
- Parameters の PropertyName

属性に関するアプリケーション固有情報

双方向スクリプト・データ変換で使用可能な属性のアプリケーション固有情報パラメーターを以下に示します。

- ColumnName

アダプター・メッセージ

WebSphere Adapter for JDBC によって送出されたメッセージを以下の場所で表示します。

メッセージのリンク先は http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.wbit.610.help.messages.doc/messages.html です。

表示される Web ページには、メッセージ接頭語のリストがあります。メッセージ接頭語をクリックすると、以下に示すように、その接頭語があるすべてのメッセージを参照できます。

- 接頭語 CWYBC があるメッセージの送出元は WebSphere Adapter for JDBC です。
- 接頭語 CWYBS があるメッセージの送出元はアダプター・ファウンデーション・クラスであり、これらはすべてのアダプターにより使用されます。

関連情報

以下の、インフォメーション・センター、IBM Redbooks および Web ページには、WebSphere Adapter for JDBC の関連情報が含まれています。

サンプルおよびチュートリアル

WebSphere Integration Developer のオンライン・サンプル/チュートリアル・ギャラリーには、WebSphere Adapters を使用するのに役立つサンプルおよびチュートリアルが置かれています。オンライン・サンプル/チュートリアル・ギャラリーへのアクセス先のページは、以下のとおりです。

- WebSphere Integration Developer を始動すると表示される「ようこそ」ページ。WebSphere Adapter for JDBC のサンプルおよびチュートリアルを表示するには、「**取得**」をクリックします。表示されたカテゴリーをブラウズして、選択を行います。
- Web 上の http://publib.boulder.ibm.com/bpcsamp/index.html のページ。

情報リソース

- WebSphere Business Process Management の情報リソース Web ページ (http://www14.software.ibm.com/webapp/wsbroker/redirect?version=pix&product=wps-dist&topic=bpmroadmaps) には、記事、Redbooks、資料、および研修用資料へのリンクが組み込まれており、WebSphere Adapters を習得するのに役立ちます。
- WebSphere Adapters ライブラリーのページ (http://www.ibm.com/software/integration/wbiadapters/library/infocenter/) には、資料の全バージョンへのリンクが組み込まれています。

関連製品の情報

- WebSphere Business Process Management バージョン 6.1.0 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp)。ここには、WebSphere Process Server、WebSphere Enterprise Service Bus、および WebSphere Integration Developer の情報が記載されています。
- WebSphere Adapters バージョン 6.0.2 インフォメーション・センター: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.wsadapters602.doc/welcome_top_wsa602.html
- WebSphere Adapters バージョン 6.0 インフォメーション・センター: http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/topic/com.ibm.wsadapters.doc/welcome_wsa.html
- WebSphere Business Integration Adapters インフォメーション・センター: http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp?topic=/com.ibm.wbi_adapters.doc/welcome_adapters.htm

developerWorks® リソース

- WebSphere Adapter Toolkit
- WebSphere Business Integration ゾーン (WebSphere business integration zone)

サポートおよび支援

- WebSphere Adapters テクニカル・サポート: http://www.ibm.com/software/integration/wbiadapters/support/
- WebSphere Adapters テクニカル・ノート: http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm 「**Product category**」リストで、アダプターの名前を選択して、「**Go**」をクリックします。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

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。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

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 CorporationDepartment 2Z4A/SOM1
294 Route 100
Somers, NY 10589-0100
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。

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も、次のように、著作権表示を入れていただく必要があります。「(c) (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。 (c) Copyright IBM Corp. _年を入れる_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを書くことができます。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

警告:

診断、修正、調整情報は、変更される場合がありますので、プログラミング・インターフェースとしては使用しないでください。

商標

IBM、IBM LOGO、developerWorks、i5/OS、OS/400、Redbooks、Tivoli、ViaVoice、WebSphere、および z/OS は、International Business Machines Corporation の米国およびその他の国における商標です。

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

Microsoft および Windows は、Microsoft 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。

この製品には、Eclipse Project (http://www.eclipse.org) により開発されたソフトウェアが含まれています。

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]

アクセシビリティー
  外部サービス・ウィザード 52
  管理コンソール 52
  キーボード 52
  ショートカット・キー 52
  IBM アクセシビリティー・センター 52
アクティベーション・スペック・プロパティー
  管理コンソールでの設定 157, 162
  リスト 238
アセンブリー・エディター、アダプター・ アプリケーション固有情報の変更 113, 137
アダプター
  プロジェクト、作成 73
アダプター実装
  セキュリティー 55
アダプターのテクニカル・ノート 261
アダプターのパフォーマンス 165
アダプター・アプリケーション
  開始 164
  停止 164
アダプター・アプリケーションの開始 164
アダプター・アプリケーションの停止 164
アダプター・メッセージ 260
アプリケーション固有情報 201
  オブジェクトへの追加 81, 116
  型が子ビジネス・オブジェクトの属性 198
  単純属性 193
一覧表、互換性 4
イベント処理
  カスタム 7, 22
  標準 7, 21
イベント・ストア 25
イベント・ストアのセットアップ 71
インターネット・プロトコル・バージョン 6.0 (IPv6) 52

[カ行]

カーディナリティー 37, 191
階層ビジネス・オブジェクト 37
外部サービス接続プロパティー 208, 229
外部サービス・ウィザード
  アクセシビリティー 52
  始動 76
  接続プロパティー 76
外部サービス・ウィザード *(続き)*
  認証 56
外部サービス・ディスカバリー
  属性情報 191
カスタム照会
  ストアード関数 24
  ストアード・プロシージャー 23
  標準の SQL 23
カスタム・プロパティー
  アクティベーション・スペック 157, 162
  管理接続ファクトリー 155, 160
  リソース・アダプター 153, 159
管理 (J2C) 接続ファクトリー・プロパティー
  管理コンソールでの設定 155, 160
管理接続ファクトリー・プロパティー
  詳細 217
  XA データ・ソース名 225
  XADataSourceName 225
関連情報 260
関連製品、情報 260
キーボード 52
旧バージョンとの互換性 62
クエリー・ビジネス・オブジェクト
  構造 32
  SELECT ステートメントからの生成 50
組み込みアダプター
  アクティベーション・スペック・プロパティー、設定 157
  管理接続ファクトリー・プロパティー、設定 155
  使用する場合の考慮事項 59
  説明 57
  リソース・アダプター・プロパティー、設定 153
クラスター環境
  説明 60
  デプロイ 60
  Inbound 処理 60
  Outbound 処理 61
研修、WebSphere Adapters 260
高可用性環境
  説明 60
  デプロイ 60
  Inbound 処理 60
  Outbound 処理 61
構成
  トレース 171
  ロギング 171
  Performance Monitoring Infrastructure (PMI) 166
構成の概要 70
後方互換性
  プロジェクト 65
  プロジェクト交換ファイル 65
互換性一覧表 4

コネクター・プロジェクト 73
コンポーネントの接続 141

[サ行]

再試行制限のプロパティー名 253
サポート
  概要 170
  セルフ・ヘルプ・リソース 180
  テクニカル 261
サンプル 67
実行時環境
  認証 56
  EAR ファイルのデプロイ先 146
実装環境、Java 143
準備済みステートメントのキャッシュ 61
ショートカット・キー 52
スタンドアロン・アダプター
  アクティベーション・スペック・プロパティー、設定 162
  管理接続ファクトリー・プロパティー、設定 160
  使用する場合の考慮事項 59
  説明 57
  リソース・アダプター・プロパティー、設定 159
ストアード関数
  概要 50
ストアード・プロシージャー 20
  概要 42
  定義 42
  定義の例 48
  ビジネス・オブジェクトの構造 29
  SQL ステートメント 42
ストアード・プロシージャー・ビジネス・オブジェクト 42
セキュリティー機能
  アダプター 55
  Java 2 セキュリティー 55
セルフ・ヘルプ・リソース 180
操作
  ApplyChanges 18
  Create 10
  Delete 19
  Execute 20
  Retrieve 11
  RetrieveAll 12
  Update 16
送達は 1 回のみ 22
属性タイプ、ビジネス・オブジェクト 192
属性プロパティー 191
ソフトウェア依存関係 75
ソフトウェア要件 4

[タ行]

ターゲット・コンポーネント 141
対話仕様プロパティー
  変更 139
対話スペック・プロパティー 227
チュートリアル 67
データ型
  複合 43
テーブル
  ビジネス・オブジェクトの構造 27
テクニカル・サポート 261
テクニカル・ノート 4, 180, 261
テクニカル・ノート、WebSphere Adapters 260
テスト環境
  デプロイ先 141, 144
  モジュールの追加先 144
  モジュールのテスト 145
デバッグ
  セルフ・ヘルプ・リソース 180
  XAResourceNotAvailableException 例外 179
デプロイメント
  オプション 57
  環境 141
  実稼働環境への 146
  テスト環境への 141
トラブルシューティング
  概要 170
  セルフ・ヘルプ・リソース 180
  XAResourceNotAvailableException 例外 179
トランザクション 9
  DataSourceJNDIName を使用した 9
トランザクション、「XA トランザクションおよびローカル・トランザクション」も参照 9
トレース
  管理コンソールによるプロパティーの構成 171
トレース・ファイル
  使用可能化 171
  詳細レベル 171
  使用不可化 171
  場所 173
  ファイル名の変更 172

[ナ行]

認証
  外部サービス・ウィザード 56
  実行時 56
  説明 56
認証別名 72

[ハ行]

ハードウェアおよびソフトウェア要件 4
ハードウェア要件 4
パッケージ・ファイル、アダプターの 172
バッチ SQL ビジネス・オブジェクト 51
  構造 33
パフォーマンス
  準備済みステートメントのキャッシュ 61

パフォーマンスに関する統計　168
パフォーマンスのモニター　165
ビジネス・オブジェクト　27, 201
　カーディナリティー　37
　照会　50
　ストアード・プロシージャー　42
　属性　191
　属性タイプ　192
　バッチ SQL　51
　表示方法　82, 117
　複合 キー　113, 137
　複数の親　113, 137
　命名規則　204
ビジネス・オブジェクト情報　191
ビジネス・オブジェクトの構造　27
　クエリー・ビジネス・オブジェクトの場合　32
　ストアード・プロシージャー・ビジネス・オブジェクトの場合　29
　テーブルまたはビュー・ビジネス・オブジェクトの場合　27
　バッチ SQL ビジネス・オブジェクトの場合　33
　ラッパー・ビジネス・オブジェクトの場合　34
ビジネス・オブジェクトの命名規則　204
ビジネス・グラフ　7
ビジネス・フォールト　174
非推奨機能　62
ビュー
　ビジネス・オブジェクトの構造　27
標準の準拠　51
ファイル
　SystemOut.log ログ・ファイル　172
　trace.log トレース・ファイル　172
フォールト
　説明　174
複合データ型　43
フラット・ビジネス・オブジェクト　37
プロジェクト交換 (PI) ファイル
　マイグレーションなしでの更新　65
プロパティー
　アクティベーション・スペック　157, 162
　　リスト　238
　外部サービス接続　208, 229
　管理 (J2C) 接続ファクトリー　155, 160
　構成プロパティー
　　Inbound　227
　　Outbound　206
　リソース・アダプター　153, 159
　Inbound 構成　227
　Outbound 構成　206
分散トランザクション、*XA* トランザクションを参照　226
変更後イメージ　8
ポーリング　25

［マ行］

マイグレーションに関する考慮事項　62
メタデータ選択プロパティー
　指定方法 (Inbound)　126
　指定方法 (Outbound)　103
メッセージ、アダプター　260
モジュールの構成のためのロードマップ　69
問題判別
　一般的な問題の解決策　181
　セルフ・ヘルプ・リソース　180
　XAResourceNotAvailableException 例外　179

［ヤ行］

ユーザー定義関数　49
ユーザー・テーブル上のトリガー　71
要件、ハードウェアおよびソフトウェア　4

［ラ行］

ラッパー・ビジネス・オブジェクト
　構造　34
　作成　104
リソース・アダプター・アーカイブ (RAR) ファイル
　サーバーへのインストール　147
　説明　147
リソース・アダプター・プロパティー
　管理コンソールでの設定　153, 159
　詳細　214, 235
例外
　XAResourceNotAvailableException　179
ローカル・トランザクション　9
ロギング
　管理コンソールによるプロパティーの構成　171
ログ・アナライザー　171
ログ・ファイル
　使用可能化　171
　詳細レベル　171
　使用不可化　171
　場所　173
　ファイル名の変更　172

A

Adapter for JDBC
　アクセシビリティー　52
　管理　153
　標準の準拠　51
Adapter for JDBC モジュール
　開始　164
　停止　164
　EAR ファイルとしてのエクスポート　149
　EAR ファイルのサーバーへのインストール　151
ApplyChanges 操作　18

C

CEI (Common Event Infrastructure) 169
Common Event Infrastructure (CEI) 169
Create 操作 10

D

DataSourceJNDIName 9
Delete 操作 19
delta 8
developerWorks 261
developerWorks リソース、WebSphere Adapters 260

E

EAR ファイル
  エクスポート 149
  サーバーへのインストール 151
EAR ファイルとしてのモジュールのエクスポート 149
EAR ファイルのインストール 151
enableHASupport プロパティー 60
Execute 操作 20

F

FFDC (First Failure Data Capture) 174
First Failure Data Capture (FFDC) 174
foreign key 191

I

IBM WebSphere Adapter Toolkit 261
Inbound 構成プロパティー 227
Inbound 処理 6
IPv6 52

J

J2C 接続ファクトリー
  管理接続ファクトリーを*参照* 217
Java 2 セキュリティー 55
Java 実装環境 143
JDBC ドライバー・ファイル 75

N

NULL オブジェクト
  取得 15

O

Outbound 構成プロパティー 206
Outbound 処理 5
Outbound 操作
  リスト 8

P

Performance Monitoring Infrastructure (PMI)
  構成 166
  説明 165
  パフォーマンスに関する統計の表示 168
PMI (Performance Monitoring Infrastructure)
  構成 166
  説明 165
  パフォーマンスに関する統計の表示 168
primary key 191

R

RAR (リソース・アダプター・アーカイブ) ファイル
  サーバーへのインストール 147
  説明 147
Redbooks、WebSphere Adapters 260
Retrieve 操作 11
RetrieveAll 操作
  データベース表ビジネス・オブジェクトの場合 12
  ユーザー指定クエリー・ビジネス・オブジェクトの場合 14

S

SystemOut.log ファイル 172

T

trace.log ファイル 172

U

UDF、「ユーザー定義関数」を参照 49
Update 操作 16

W

WebSphere Adapters バージョン 6.0 情報 261
WebSphere Adapters バージョン 6.0.2 情報 261
WebSphere Application Server 情報 261
WebSphere Business Integration Adapters の情報 261
WebSphere Business Process Management バージョン 6.1.0 情報 261
WebSphere Enterprise Service Bus
  情報 261
  デプロイ先 146
WebSphere Extended Deployment 60
WebSphere Integration Developer
  情報 261
  テスト環境 141

WebSphere Process Server
情報 261
デプロイ先 146

X

XA トランザクション 9
DB2 データベース 9
Oracle データベース 9
XA データ・ソース名 226
XAResourceNotAvailableException 179

Printed in Japan

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