IBM WebSphere Business Integration
Adapters

Adapter Development Kit ユーザーズ・ガイド

バージョン 2.4.0

IBM WebSphere Business Integration
Adapters

Adapter Development Kit ユーザーズ・ガイド

バージョン 2.4.0

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、11 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 Business Integration Adapter Development Kit バージョン 2.4.0、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。

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： IBM WebSphere Business Integration Adapters
Adapter Development Kit User Guide
Version 2.4.0

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.1

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

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

目次

本書について

IBM® WebSphere® Business Integration Adapter ポートフォリオは、先進の e-business テクノロジー、エンタープライズ・アプリケーション、およびレガシー/メインフレーム・システムを統合的に接続する機能を提供します。本製品には、コンポーネントをカスタマイズ、作成、および管理するためのツールとテンプレートが含まれており、これにより、ビジネス・プロセスの統合を実現します。

本書では、IBM WebSphere Business Integration Adapters 製品に含まれる Adapter Development Kit (ADK) 用のサンプルを構成して実行する方法について説明します。

対象読者

本書は、お客様のサイトで Adapter Development Kit を使用するコンサルタント、開発者、およびシステム管理者を対象にしています。

関連文書

この製品に付属する資料の完全セットで、すべての WebSphere Business Integration Adapters のインストールに共通な機能とコンポーネントについて説明します。また、特定のコンポーネントに関する参考資料も含まれています。

この製品に付属する資料の完全セットで、すべての WebSphere Business Integration Adapters のインストールに共通な機能とコンポーネントについて説明します。また、特定のコンポーネントに関する参考資料も含まれています。

以下のサイトから、関連資料をインストールすることができます。

- アダプターの一般情報、WebSphere Message Brokers (WebSphere MQ Integrator、WebSphere MQ Integrator Broker、WebSphere Business Integration Message Broker) でのアダプターの使用、WebSphere Application Server でのアダプターの使用については、次の IBM WebSphere Business Integration Adapters InfoCenter をご覧ください。
  http://www.ibm.com/websphere/integration/wbiadapters/infocenter
- WebSphere InterChange Server でのアダプターの使用については、次の IBM WebSphere InterChange Server InfoCenter をご覧ください。
  http://www.ibm.com/websphere/integration/wicserver/infocenter
  http://www.ibm.com/websphere/integration/wbicollaborations/infocenter
- WebSphere Message Brokers の詳細については、以下をご覧ください。
  http://www.ibm.com/software/integration/mqfamily/library/manualsa/
- WebSphere Application Server の詳細については、以下をご覧ください。
  http://www.ibm.com/software/webservers/appserv/library.html

これらのサイトでは、資料のダウンロード、インストール、および表示方法を簡単に説明しています。

表記上の規則

本書は下記の規則に従って編集されています。

courier フォント	コマンド名、ファイル名、入力情報、システムが画面に出力した情報など、記述されたとおりの値を示します。
イタリック *青字のテキスト*	初出語、変数名、または相互参照を示します。 オンライン・マニュアルでのみ表示される青字のテキストは、相互参照のハイパーリンクを示します。青字のテキストをクリックすると、参照先のオブジェクトにジャンプします。
{ }	構文の記述行の場合、中括弧 {} で囲まれた部分は、選択対象のオプションです。1 つのオプションのみを選択する必要があります。
|	構文の記述行の場合、パイプで区切られた部分は、選択対象のオプションです。1 つのオプションだけを選択する必要があります。
[ ]	構文の記述行の場合、大括弧 [] で囲まれた部分は、オプションのパラメーターです。
...	構文の記述行の場合、省略符号 ... は直前のパラメーターが繰り返されることを示します。例えば、option[,...] は、複数のオプションをコンマで区切って指定できることを意味します。
< >	不等号括弧は、名前の個々の要素を囲み、各要素を区別します (例: <server_name><*connector_name*>tmp.log)。
/、¥	本書では、ディレクトリー・パスの規則として円記号 (¥) を使用します。UNIX システムの場合、円記号の代わりにスラッシュ (/) を使用します。
%*text*% および $*text*	% 記号で囲まれたテキストは、Windows の text システム変数またはユーザー変数の値を示しています。UNIX 環境の場合、これと同等の表記は $*text* となります。これは、*text* UNIX 環境変数の値を示しています。

本リリースの新機能

「Adapter Development Kit ユーザーズ・ガイド」は、WebSphere Business Integration Adapter Framework バージョン 2.4 リリースで新たに作成された資料です。

第 1 章 TwineBall

TwineBall サンプルは、コネクター、ODA、およびアプリケーションから構成されており、アダプター開発の基本原則のいくつかが理解できるように設計されています。このアプリケーションは TwineBall と呼ばれ、以下のものを備えています。

- 基本的なデータ指向操作 (作成、検索、更新、および削除など) のための RMI インターフェース
- Object Discovery Agent に対するエンティティー定義情報 (組み込み済み)

対象読者

本書は、カスタム・アダプターの開発を検討している熟練した Java 開発者向けに作成されています。アダプターの開発を始める前に、ドキュメンテーション Infocenter 内の「*IBM WebSphere コネクター開発ガイド (Java 用)*」および「*IBM WebSphere ビジネス・オブジェクト開発ガイド*」も学習する必要があります。

アダプターの概要については、ドキュメンテーション Infocenter 内の「*Technical Introduction to WebSphere Business Integration Adapters*」を参照してください。

アダプター・アーキテクチャー

TwineBall サーバーは、DB2 内に作成されたデータベースを使用してデータを保管します。このサーバーは多数の異なるエンティティー定義を処理できますが、最も多く使用される表は以下のとおりです。

- Customer
- Address
- Order
- OrderLine

2 ページの図 1 は、TwineBall アーキテクチャーを示しています。

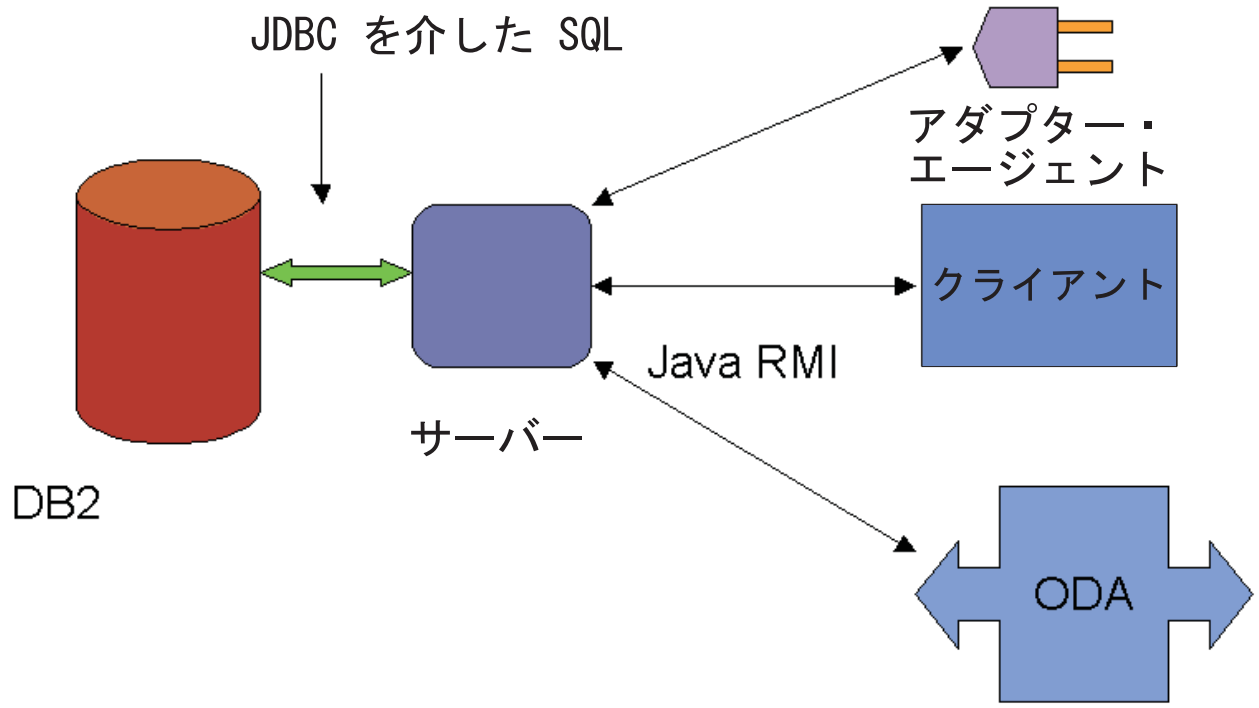

図 1. 元の TwineBall アーキテクチャー

イベント・デリバリーは、ポーリング機構によって提供されます。ポーリングを機能させるには、DB2 にイベント表とアーカイブ表を作成し、トリガーも用意しておく必要があります。

アダプターと ODA は、それぞれ Java RMI 経由でこのサーバーに接続します。またアダプターと ODA は、サーバー名、ポート、およびサービス名を示すサーバー URL をそれぞれ取得します。

標準のサーバー URL は `RMI://localhost:2112/TwineBall` です。

また、TwineBall アダプターはサーバーとは独立して実行することもできます。この場合、TwineBall アダプターは、C++ で作成された代わりの TwineBall API インプリメンテーションを呼び出します。アダプターは、リモート RMI を呼び出す代わりに、C++ 共用ライブラリーをロードし、その内部に対して呼び出しを行います。詳細については、6 ページの『JNI の使用』を参照してください。

TwineBall サーバー

TwineBall サーバー・コンポーネントを以下に示します。

- **TwineBallConnection**: データベース・ロジックの大半が含まれます。
- **TwineBallServer**: RMI クライアントへの接続を提供します。
- **TwineBallInterface**: 呼び出し先のアダプター、ODA、またはその他のクライアント用の API を提供します。
- **TwineBallConnectionPool**: TwineBall 接続をプールします (API の一部と見なされます)。

アダプター

アダプター・コンポーネントを以下に示します。

- **AdapterAgent**: 初期化と終了のロジックを提供することにより、CWConnectorAgent を拡張します。
- **AdapterBOHandler**: CWConnectorBOHandler を拡張し、サービス呼び出し要求を処理します。
- **AdapterEventStore**: イベント・デリバリー・ロジックを提供することにより、CWConnectorEventStore を拡張します。

Object Discovery Agent

ODA コンポーネントを以下に示します。

- **ObjectDiscoveryAgent**: 初期化機能を提供することにより ODKAgentBase を拡張し、ObjectFinder と ObjectAnalyzer を呼び出します。
- **ObjectFinder**: getTreeNodes のロジックをインプリメントします。
- **ObjectAnalyzer**: generateDefs のロジックをインプリメントします。

インストール済みのディレクトリー

TwineBall サンプルとともにいくつかのディレクトリーがインストールされます。それらを以下に示します。

表 1. ファイル・ディレクトリー

ディレクトリー	目的
/dependencies	TwineBall サーバーの実行に必要なプロパティー・ファイルとポリシー・ファイルです。
/documentation	サンプルの使用方法に関する README ファイルが含まれます。
/JNI_supplement	JNI を使用してアダプターで C++ API を呼び出す方法を示すコードが含まれます。
/Repository	アダプターおよび ODA 用のメッセージ・ファイルが含まれます。
/Samples	アダプターのビジネス・オブジェクトが含まれます。
/setup	設定用スクリプトとサーバー、アダプター、および Object Discovery Agent の実行用スクリプトが含まれます。

Java ソース・パッケージ

Java リソースを含むパッケージは、以下のように構成されています。

表 2. Java リソース

リソース	内容
com.ibm.wbia.TwineBall.Adapter	TwineBall アダプター
com.ibm.wbia.TwineBall.oda	TwineBall Object Discovery Agent
com.ibm.wbia.TwineBall.Server	TwineBall サーバーおよび API
com.ibm.wbia.TwineBall.jnibridge	JNI の補足ファイル (スタブ・ライブラリーおよびベクトル・ブリッジ)

起動と実行

TwineBall サーバー、アダプター、および ODA は、Windows でのみテストされています。その他のプラットフォーム上でも動作する可能性はありますが、そのテストはユーザーの課題として残しておきます。

TwineBall サンプルを実行するには、JDK をインストールしておく必要があります (1.3.1 を推奨)。

データベースの設定

create、restore、update、delete (CRUD) 操作、および真のメタデータ主導型の設計を説明するために、TwineBall サーバーは、実際のデータベースを使用してエンティティーを定義しデータを保管します。そのために、いくつかの追加設定が必要となります。この作業を容易に行うためにスクリプトが用意されていますが、スクリプトをご使用のシステムに適合させるには、若干の変更が必要となります。

データベースを設定するには、以下の手順を実行します。

1. WebSphere Business Integration Adapter Framework または WebSphere InterChange Server をインストールし、その前提ソフトウェアをすべてインストールします。
2. IBM DB2 バージョン 8、フィックスパック 2 をインストールします。
3. オペレーティング・システム内に次のユーザー ID を作成します。
   TwineBall、TwineBallAdapter。
   これに対して次のパスワードを設定します。

   ```
   sample42
   ```
4. Twine データベースを作成します。setup/SQL ディレクトリーに Twine-database.bat と Twine-database.sql が入っていることを確認します。db2admin ID のパスワードと一致するように Twine-database.sql を編集します。
5. Twine-database.bat スクリプトを実行します。エラー・メッセージは表示されないはずです。もしメッセージが表示された場合は、それを解決してから続行してください。
6. TwineBall サーバーで必要な表を作成します。setup/SQL ディレクトリーに TwineBall-Schema.bat と TwineBall-Schema.SQL が入っていることを確認します。
7. TwineBall-Schema.bat スクリプトを実行します。エラーが発生した場合は、それを解決してから続行してください。
8. イベント表とアーカイブ表を設定します。setup/SQL ディレクトリーに WBIA-Events.sql と WBIA-Events.bat が入っていることを確認します。
9. WBIA-Events.bat スクリプトを実行します。エラーが発生した場合は、それを解決してから続行してください。

サーバーの実行

サーバーを実行するには、以下の手順を実行します。

1. すべての Java ソースを IDE 内のプロジェクトにインポートすることにより、/src ディレクトリーと JNI_Supplement ディレクトリー内の Java ソース・ツリー全体をコンパイルします。

InterChange Server または Adapter Framework の /lib ディレクトリーに入っているすべての JAR ファイルが CLASSPATH に含まれるように IDE を構成します。

2. Adapter Framework または InterChange Server インストールの connectors ディレクトリーの下に、TwineBall という名前のディレクトリーを作成します。
3. 標準の Java パッケージ構造を使用して、すべてのコンパイル済みクラス・ファイルをこのディレクトリーにエクスポートします (com.ibm.sample.MyClass は com/ibm/sample/Myclass.class として保管されています)。
4. setup¥scripts ディレクトリー内のすべてのファイルもこのフォルダーにコピーします。通常、アプリケーションはアダプターと異なるディレクトリーに配置されますが、このように構成するとインストールが容易になります。
5. 新しく作成した TwineBall ディレクトリーをシステムの CLASSPATH に追加します。これにより、Java ランタイムがサーバー・クラスを検出できるようになります。
6. Java 開発キットの bin ディレクトリーが CLASSPATH に含まれていることを確認します。runrmic.bat ファイルを実行します。これにより、必要なスタブと RMI 通信に必要なスケルトンが生成されます。
7. TwineBall ディレクトリーの中に Properties という名前の新しいディレクトリーを作成します。
8. dependencies ディレクトリー内のファイル TwineBall.policy および TwineBall.db2.properties をこのディレクトリーに配置し、TwineBall.db2.properties の名前を TwineBall.properties に変更します。
9. TwineBall.properties ファイルを編集し、server.name をご使用のシステム名に変更します。例えば、マシン名が JEFFB である場合、server.name は //JEFFB:2112/TwineBallServer となります。
10. runserver スクリプトを実行し、TwineBall サーバーを始動します。サーバーが動作しているときは、「サーバー ... が登録されました (Server ... has been registered)」というメッセージが表示されます。
11. サーバーをウィンドウ内で実行させておきます。ODA とアダプターは、RMI 経由でサーバーに接続します。

TwineBall コネクターの始動

TwineBall コネクターを始動するには、以下の手順を実行します。

1. Adapter Framework または InterChange Server インストールの /connectors ディレクトリーの中に、TwineBall という名前のディレクトリーを作成します。
2. WebSphere InterChange Server の Adapter Framework を実行しない場合は、このディレクトリーに構成ファイルを配置します。
   InterChange Server を実行する場合は、構成ファイルをインポートします。
3. ブローカー用のアダプターを実行するために必要な WebSphere MQ キューを作成します。InterChange Server がブローカーである場合は、IDL トランスポートが使用できるため、この作業は不要です。
4. Connector Configurator を始動し、MessageFileName 標準プロパティーを「TwineBallAdapter.txt」に変更します。

5. アダプターを始動する際の通常の構文を使用して、コネクターの始動を試みます。以下に例を示します。

```
start_TwineBall TwineBall WMQI  -c
C:¥IBM¥WebSphereAdapters¥connectors¥TwineBall¥TwineBall.cfg
```

6. アダプターをテストするには、InterChange Server 内にパススルー・コラボレーションを設定するか、非 InterChange Server モードでデリバリーおよびサービス呼び出しキューを反転します。ただし、これらの手順は本書の範囲を超えています。

TwineBall ODA の始動

TwineBall ODA を始動するには、以下の手順を実行します。

- start_twineballODA.bat ファイルを実行し、ODA の始動を試みます。
- Business Object Designer を通じて ODA を検出し使用することを試みます。
- ビジネス・オブジェクト作成元のノードを選択するときは、作成するすべてのビジネス・オブジェクトも同時に選択してください。ビジネス・オブジェクトを明確に選択しないと、子ビジネス・オブジェクトが作成されません。

JNI の使用

Java アダプターを作成して Java Native Interface (JNI) を使用すると、C++ API を呼び出すことができます。詳細については、7 ページの『第 2 章 Java Native Interface (JNI)』を参照してください。

第 2 章 Java Native Interface (JNI)

この章では、Java アダプターで Java Native Interface (JNI) を使用し、C++ アプリケーションとの統合を行う方法について説明します。

C++ API を使用してアプリケーションを利用するには、基本的に 3 つの方法があります。

- CDK を使用して C++ アダプターを作成する。
- Java アダプターを作成し、JNI (Java Native Interface) を使用して C++ API を呼び出す。
- アダプターは作成せず、テクノロジー・アダプター (MQSeries や COM 用のアダプター) を使用する。

ここでは 2 番目の方法について説明し、この方法を使用するサンプル・アダプターを確認します。

注: この例の目的は、JNI を使用する Java アダプターの作成方法とサンプル JNI アダプターの動作を確認することです。この方法を理解して活用するには、JNI のインプリメンテーション手法についての知識が前提条件として必要です。

JNI は、Java アプリケーションから C/C++ ライブラリーを呼び出せるように設計されています。以下の例では、Java を使用して C++ API を呼び出します。この API は、TwineBall (Java アダプター) で C++ API を呼び出す方法が理解できるようにインプリメントされます。

API との対話

TwineBall アダプターは、アプリケーション・サーバーを呼び出す際に、必ず TwineBall インターフェースを使用します。 Java RMI では、リモート・インターフェースの呼び出しは、JVM によって自動的にマーシャルされます。図 2 は、Java RMI を介した元の対話を示しています。

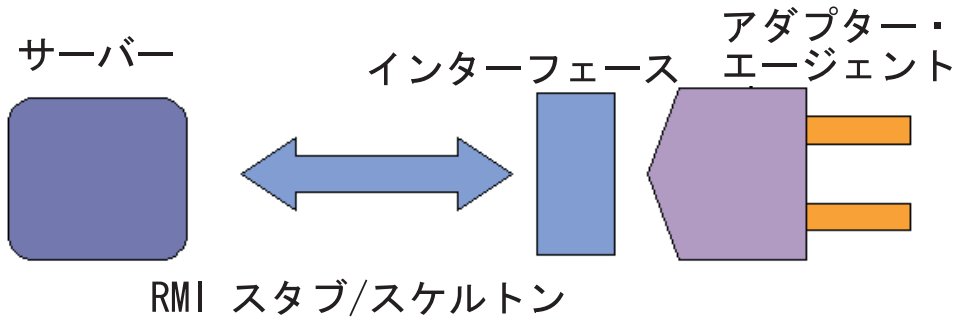

図 2. RMI を介した対話

JNI を説明するために、図のサーバー側を C++ 共用ライブラリーで置き換えます。これにより、図は 図 3 のようになります。

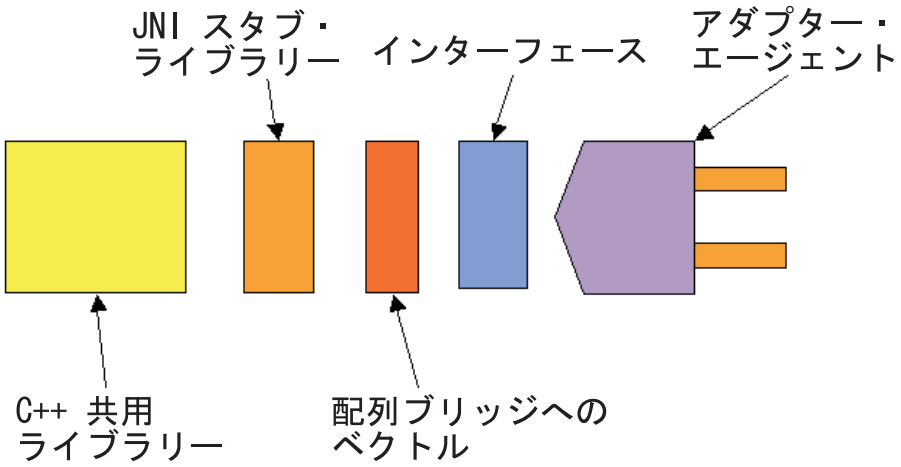

図 3. 変更した TwineBall アーキテクチャー

明確に定義された Java インターフェースがすでにアプリケーションを呼び出していたため、アダプター側のコードをほとんど変更することなく、このインターフェースのインプリメンテーションを変更できました。そのため、JNI を使用するアダプターと Java RMI を使用するアダプターは同じアダプター・コードで説明できます。変更するのは、インターフェースの背後のインプリメンテーションのみです。

TwineBall インターフェースでは、Java ベクトルが多用されます。ベクトルは、C++ の世界に変換するのではなく、ストリング配列に変換することにしました。これは、VectorToArrayBridge で行うことです。アダプターで Java ユーティリティー・クラス使用するほうが便利な場合は、同様の方法を使用できます。

JNI スタブ・ライブラリーでは、ネイティブ・メソッドを宣言します。これは、C++ でインプリメントされたメソッドを Java の世界に接続するものです。

この場合の C++ 共用ライブラリーは、単一の C++ モジュールを使用してインプリメントされます。ただし、この共用ライブラリーには JNI.H ヘッダー・ファイルが含まれるため、Java 型が認識されます。 C++ 空間では、基本 Java 型と C++ 型との間の変換が今でも幅広く行われています。既存の C++ アプリケーション/API に接続するには、Java だけではなく、新しい C++ コードも作成する必要があります。

注意していただきたいのは、データベースはもはや含まれていないという点です。この例では、インターフェースは事前に決めた戻り値を使用してインプリメントされます。

サンプルの JNI ソース・コードには、以下のコンポーネントが含まれます。

表 3. JNI ソース・コードのコンポーネント

ファイル名	機能
Makefile.PLATFORM	特定プラットフォーム上で C++ 共用ライブラリーを作成します。
com_ibm_wbia_TwineBall_jnibridge_JNIStubLibrary.cc	C++ 共用ライブラリーのソース・コード。C++「グルー・コード」が含まれます。
com_ibm_wbia_TwineBall_jnibridge_JNIStubLibrary.h	C++ 共用ライブラリー用のヘッダー・ファイル。
com.ibm.wbia.TwineBall.jnibridge.JNIStubLibrary	JNI スタブ・ライブラリー。
com.ibm.wbia.TwineBall.jnibridge.VectorToArrayBridge	配列ブリッジへのベクトル。
com.ibm.wbia.TwineBall.jnibridge.test.TestDriver	JNIStubLibrary 用のテスト・ドライバー。

TwineBall JNI インプリメンテーションのコンパイルと実行

共用ライブラリーを作成するには Make を使用します。ターゲット・プラットフォームのビルド環境が完備していることを確認してください。各プラットフォームの Make ファイルには、どのコンパイラー向けに共用ライブラリーを作成するかについての情報が含まれています。該当するコンパイラーと Make がインストールされている場合、ライブラリーをコンパイルするのに必要なことは、ファイルをターゲット・プラットフォーム上に抽出し、`make -f Makefile.Platform` を入力することだけです (ここで `Platform` は、希望するターゲット・プラットフォームを示します)。

TwineBall アダプターを JNI ブリッジとともに実行するには、以下の手順を実行します。

1. ターゲット・プラットフォーム上に WebSphere InterChange Server または WebSphere Business Integration Adapter Framework をインストールします。
2. TwineBall サーバーおよびアダプター用にコンパイルしたクラスを `%CROSSWORLDS%/connectors/twineball` ディレクトリーに配置します。
3. 「dependencies」内のメッセージ・ファイルを「messages」フォルダーに配置します。
4. 共用ライブラリーと `start_twineball.sh` スクリプトを `%CROSSWORLDS%/connectors/twineball` ディレクトリーに配置します。TwineBall データベースまたはサーバーを設定する必要はありません。なぜなら、この例ではアダプターは JNI ブリッジを代わりに呼び出すからです。
5. twineball.cfg をテンプレートとして使用してアダプターを構成し、特別なストリング `JNIStub` をサーバー URL として使用します。
6. 始動スクリプトを使用して、通常どおりアダプターを始動します。

アダプター用の JNI バックエンドは、「TwineBallCustomer」ビジネス・オブジェクト用のサービス呼び出しとイベント・デリバリーを両方インプリメントしています。TwineBallCustomer ビジネス・オブジェクトは、「sample business objects」フォルダーに入っています。

JNI アダプターの作成

JNI アダプターを作成するには、以下の手順を実行します。

1. 既存の C++ API を調べ、Java で定義されていないデータ構造があればそれをメモします。これらのデータ構造は、C++ グルー・コード内で変換する必要があります。
2. Java の内部でメソッドにできるかぎり一致するネイティブ宣言を指定して JNI スタブ・ライブラリーを作成します。これは、アダプターが使用する API 内で公開されるすべての関数について行う必要があります。
3. `javah` を実行し、スタブ・ライブラリー用の C++ ヘッダー・ファイルを作成します (使用例:
   `javah com.ibm.wbia.TwineBall.jnibridge.JNIStubLibrary`)。
4. 生成される C/C++ のヘッダーにすべての関数をインプリメントします。このコードは、JNI 関数シグニチャーと API を結び付けるために使用します。
5. テスト・ドライバー・プログラムを使用して、JNI スタブ・ライブラリーの関数をすべてテストします。
6. JNI スタブ・ライブラリーを Java API として使用し、Java アダプターの場合と同様にアダプターのコーディングを開始します。

注:

1. TwineBall で確認したように、JNI レイヤーの両側で型変換を実行することが適切な場合があります。
2. 文字セットは非常に重要です。この例で Java ユニコード・ストリングを C++ ストリングに変換するときは、それらを UTF-8 に変換しています。アプリケーションで UTF-8 が処理されない場合は、それらを特定のコード・ページに変換する必要があります。これを行うときは、十分に注意を払ってください。可能性のあるすべてのユニコード文字を処理するか、または適切な方法で失敗するようにコードを作成してください。
3. HP-UX の多くのバージョンでは、C++ に対して JNI を実行する際には、共用ライブラリー内の `_main()` 関数を宣言して呼び出す必要があります。これにより、C++ ランタイムを初期化する指示がオペレーティング・システムに伝えられます。Win32 上で同じコードは失敗します。したがって、この移植問題には、`#defines` が必要となる場合があります。

スレッド・セーフティー

スレッド化については、以下の点に注意してください。

- デフォルトでは Java アダプターはマルチスレッド化されていることを忘れないでください。アダプターは、複数のスレッド上のサービス呼び出しを複数同時に処理するように要求される場合があります。ネイティブ・コードがスレッド・セーフでない場合は、始動スクリプトに `-t` オプションを指定して、アダプターを単一スレッド化する必要があります。
- C++ 空間では、あるスレッドから別のスレッドに Java オブジェクトを渡さないでください (アダプターでこれを行う必要はほとんどありませんが)。
- グローバル・アクセスが可能なオブジェクトの同期化については、標準的な規則が適用されます。
- あるスレッドから別のスレッドに `JNIEnv*` を渡したり、2 つの異なるスレッドで同じになっているポインターに依存してはいけません。
  JVM は、各スレッドごとに異なる JNI 環境を作成します。ネイティブ・コードでは、現在のスレッドで有効なポインタのみを使用する必要があります。

付録. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM RTP Laboratory
3039 Cornwallis Road
P.O. BOX 12195
Raleigh, NC 27709-2195
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。

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。

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。

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

以下は、IBM Corporation の商標です。

IBM
IBM ロゴ
AIX
CrossWorlds
DB2
DB2 Universal Database
MQIntegrator
MQSeries
Tivoli
WebSphere

Lotus、Domino、Lotus Notes、および Notes Mail は、IBM Corporation の商標です。

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

MMX、Pentium および ProShare は、Intel Corporation の米国およびその他の国における商標です。

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

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

IBM WebSphere Business Integration Adapter Framework V2.4.0

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