BEA WebLogic Server™

FAQ 集

WebLogic Server バージョン 6.1
マニュアルの日付：2001 年 12 月 19 日

著作権

限定的権利条項

商標または登録商標

BEA WebLogic Server FAQ 集

パート番号	マニュアルの日付	ソフトウェアのバージョン
なし	2001 年 12 月 19 日	WebLogic Server バージョン 6.1

目次

4. FAQ: コード例

5. FAQ: dbKona

6. FAQ: EJB

7. FAQ: インストール

8. FAQ: Java

9. FAQ: J2EE コネクタ アーキテクチャ

10. FAQ: WebLogic JDBC

11. FAQ: WebLogic jDriver for Informix

12. FAQ: WebLogic jDriver for MSSQL Server

13. FAQ: WebLogic jDriver for Oracle

14. FAQ: JMS

15. FAQ: WebLogic メッセージング ブリッジ

16. FAQ: JTA

17. FAQ: プラグイン

18. FAQ: サーバ関連の質問

19. FAQ: サーバサイド Java（サーブレット）

20. FAQ: セキュリティ

21. FAQ: Web サービス

22. FAQ: 無線関連の質問

23. FAQ: XML

このマニュアルの内容

このマニュアルでは、BEA WebLogic Server™ でよく寄せられる質問について説明します。

このマニュアルの構成は次のとおりです。

- 第 1 章「FAQ: 管理とコンフィグレーション」では、WebLogic Server の管理およびコンフィグレーションに関する質問と回答を紹介します。
- 第 2 章「FAQ: アプレット」では、アプレットに関する質問と回答を紹介します。
- 第 3 章「FAQ: クラスタ化」では、クラスタ化に関する質問と回答を紹介します。
- 第 4 章「FAQ: コード例」では、コード例に関する質問と回答を紹介します。
- 第 5 章「FAQ: dbKona」では、dbKona に関する質問と回答を紹介します。
- 第 6 章「FAQ: EJB」では、EJB に関する質問と回答を紹介します。
- 第 7 章「FAQ: インストール」では、WebLogic Server のインストールに関する質問と回答を紹介します。
- 第 8 章「FAQ: Java」では、Java に関する質問と回答を紹介します。
- 第 9 章「FAQ: J2EE コネクタ アーキテクチャ」では、J2EE コネクタ アーキテクチャに関する質問と回答を紹介します。
- 第 10 章「FAQ: WebLogic JDBC」では、JDBC に関する質問と回答を紹介します。
- 第 11 章「FAQ: WebLogic jDriver for Informix」では、WebLogic jDriver for Informix に関する質問と回答を紹介します。
- 第 12 章「FAQ: WebLogic jDriver for MSSQL Server」では、WebLogic jDriver for MSSQL Server に関する質問と回答を紹介します。
- 第 13 章「FAQ: WebLogic jDriver for Oracle」では、WebLogic jDriver for Oracle に関する質問と回答を紹介します。

- 第 14 章「FAQ: JMS」では、JMS に関する質問と回答を紹介します。
- 第 16 章「FAQ: JTA」では、JTA に関する質問と回答を紹介します。
- 第 18 章「FAQ: サーバ関連の質問」では、WebLogic Server に関する一般的な質問と回答を紹介します。
- 第 19 章「FAQ: サーバサイド Java（サーブレット）」では、サーブレットに関する質問と回答を紹介します。
- 第 20 章「FAQ: セキュリティ」では、セキュリティに関する質問と回答を紹介します。
- 第 22 章「FAQ: 無線関連の質問」では、無線技術に関する質問と回答を紹介します。
- 第 23 章「FAQ: XML」では、XML に関する質問と回答を紹介します。

対象読者

このマニュアルは、Sun Microsystems の Java 2 Platform, Enterprise Edition（J2EE）を使った e- コマース アプリケーションを構築するアプリケーション開発者を対象としています。Web 技術、オブジェクト指向プログラミング技術、および Java プログラミング言語に読者が精通していることを前提として書かれてます。

e-docs Web サイト

BEA 製品のドキュメントは、BEA の Web サイトで入手できます。BEA のホーム ページで [製品のドキュメント] をクリックします。

このマニュアルの印刷方法

Web ブラウザの [ファイル | 印刷] オプションを使用すると、Web ブラウザからこのマニュアルのメイン トピックを一度に 1 つずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。WebLogic ServerPDF を Adobe Acrobat Reader で開くと、マニュアルの全体（または一部分）を書籍の形式で印刷できます。PDF を表示するには、WebLogic Server ドキュメントのホーム ページを開き、[ドキュメントのダウンロード] をクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader は、Adobe の Web サイト（http://www.adobe.co.jp）から無料で入手できます。

関連情報

BEA の Web サイトでは、WebLogic Server の全マニュアルを提供しています。

サポート情報

BEA のドキュメントに関するユーザからのフィードバックは弊社にとって非常に重要です。質問や意見などがあれば、電子メールで docsupport-jp@bea.com までお送りください。寄せられた意見については、ドキュメントを作成および改訂する BEA の専門の担当者が直に目を通します。

電子メールのメッセージには、ご使用のソフトウェア名とバージョン名、およびマニュアルのタイトルと作成日付をお書き添えください。本バージョンの BEA WebLogic Server について不明な点がある場合、または BEA WebLogic Server のインストールおよび動作に問題がある場合は、BEA WebSUPPORT（www.bea.com）を通じて BEA カスタマ サポートまでお問い合わせください。カスタマ サポートへの連絡方法については、製品パッケージに同梱されているカスタマ サポート カードにも記載されています。

カスタマ サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メール アドレス、電話番号、ファクス番号
- 会社の名前と住所
- お使いの機種とコード番号
- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

表記法	適用
〔Ctrl〕+〔Tab〕	同時に押すキーを示す。
斜体	強調または本のタイトルを示す。

等幅テキスト	コード サンプル、コマンドとそのオプション、Java クラス、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。
	例:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

斜体の等幅テキスト	コード内の変数を示す。 例: `String CustomerName;`

表記法	適用
すべて大文字のテキスト	デバイス名、環境変数、および論理演算子を示す。 例： LPT1 BEA_HOME OR
{ }	構文内の複数の選択肢を示す。
[ ]	構文内の任意指定の項目を示す。 例： `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
|	構文の中で相互に排他的な選択肢を区切る。 例： `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`
...	コマンドラインで以下のいずれかを示す。 ■ 引数を複数回繰り返すことができる。 ■ 任意指定の引数が省略されている。 ■ パラメータや値などの情報を追加入力できる。
. . .	コードサンプルまたは構文で項目が省略されていることを示す。

1 FAQ: 管理とコンフィグレーション

- config.xml ファイルを修正する際に気を付けることは何ですか。
- config.xml ファイルはどのように編集するのですか。

Q. config.xml ファイルを修正する際に気を付けることは何ですか。

A. config.xml ファイルを編集する際には、WebLogic Server Administration Console を使用することをお勧めします。config.xml ファイルを直に編集する必要が生じた場合は、以下の事項に注意してください。

- 編集の前には、必ず config.xml ファイルのコピーを保存します。編集後に起動を妨げるエラーが発生しても、ファイルを保存しておけば少なくともその保存バージョンには戻ることができます。
- ドメインがアクティブではないときは、ドメインの config.xml ファイルを手動で編集してください。ドメインがアクティブなときにコンフィグレーション ファイルを手動で編集した場合、加えた変更はシステムによって上書きされる可能性があります。さらに、ドメインがアクティブのときに手作業で行った変更は実行時にシステムで無視されます。
- 手作業で config.xml を編集するときには有効性の検証や値のチェックが行われないので、そのコンフィグレーション ファイルを初めてロードするとき、つまり管理サーバを再起動するときに型チェックが行われます。その時点で無効な XML または無効な属性値が検出された場合、ドメインは起動できません。

config.xml ファイルの修正の詳細については、『管理者ガイド』の「WebLogic Server の起動と停止」セクションの config.xml.booted ファイルに関する情報を参照してください。

Q. `config.xml` ファイルはどのように編集するのですか。

A. WebLogic Server およびクラスタから成るドメインの永続的なコンフィグレーションは、XML コンフィグレーション ファイル（`config.xml`）に格納されます。このファイルは、以下の方法で修正できます。

- ドメインのコンフィグレーションを修正またはモニタする適切な方法は、Administration Console を使用することです。
- WebLogic Server には、ドメイン リソースのコンフィグレーション属性を修正するためにプログラムで使用できるコンフィグレーション API（Application Programmatic Interface）が備わっています。
- ドメインの属性には、WebLogic Server コマンドライン ユーティリティを使用してアクセスできます。このユーティリティは、ドメイン管理を自動化するスクリプトを作成する場合に使用します。『管理者ガイド』の「WebLogic ドメインの管理用コマンド」を参照してください。

2 FAQ: アプレット

Q. アプレットの代わりに使用できるものは何ですか。

A. BEA では、J2EE プラットフォームの一部である HTTP サーブレットおよび JavaServer Pages（JSP）を利用したサーバサイド アプリケーションの使用をサポートしています。新しいアプリケーションを開発する前に、サーブレットまたは JSP の使用を検討することをお勧めします。サーブレットおよび JavaServer Pages（JSP）を利用している、設計の優れた対話型の Web ページは、より高速でより信頼性の高い Web サイトを実現します。現在アプレットを使用している場合は、Java Web Start を使用してほとんどを Java アプリケーションに変換でき、WebLogic Server の使用を継続することができます。詳細については、Sun の Java Web Start サイトを参照してください。

詳細については、『WebLogic Server アプレット ユーザーズ ガイド』を参照してください。

Q. ブラウザ アプレットに「ネイティブ」の 2 層ドライバを使用できますか。

A. できません。署名なしのアプレット内では、ネットワーク経由でネイティブ ライブラリをロードすることも、ローカルのファイル システムにアクセスすることも、アプレットのロード元ホスト以外のホストに接続することもできません。これらの制限は、無防備なユーザに対するアプレットの望ましくない動作を避けるため、アプレット セキュリティ マネージャにより強制されます。

アプレットから jDriver for Oracle を使用しようとすると、最初の制限の違反になります。非 Java の Oracle クライアント ライブラリに対して jDriver for Oracle から呼び出しを行うことができるようにするネイティブ（非 Java レイヤ）ライブラリをロードしようとすると、アプレットは失敗します。生成された例外を見る

と、アプレットの失敗は java.lang.System.loadLibrary で発生していることがわかります。これは、セキュリティ マネージャが、ローカル ライブラリのロードが試行されていると判断し、アプレットを停止させたためです。

ただし、アプレットでの JDBC 接続のために、WebLogic JTS または Pool ドライバを使用することはできます。これらの WebLogic 多層 JDBC ドライバのいずれかを使用するときは、WebLogic Server と DBMS 間の接続のために WebLogic jDriver for Oracle（または他の任意の 2 層 JDBC ドライバ）のコピーが必要です。

Q. ブラウザ アプレットがデータベースに接続できないのはなぜですか。

問題 : DBMS とのインタフェースとしてアプレットで WebLogic 多層ドライバを使用しています。ローカル マシンで Sun Appletviewer を使用してクラスを実行する場合は、何の問題もありません。しかし、Netscape ブラウザでアプレットを実行しようとすると、アプレットに接続できません。

A. Appletviewer では動作して Netscape で動作しない場合は、Netscape のセキュリティ制限に違反している可能性があります。この場合のセキュリティ制限は、アプレットはアプレットのダウンロード元サーバ以外のマシンに対してソケットを開くことができないというものです。この問題を解決するには、DBMS と同一ホストのアプレット コードを使用する必要があります。

また、アプレット CODEBASE と T3Client のコンストラクタで使用する IP ネーミング フォーマットは一致していなければなりません。つまり、一方でドット（.）表記を使用し、他方でドメイン名を使用することはできません。

Q. アプレットが Appletviewer では動作するのにブラウザで動作しないのはなぜですか。

問題 : 配布キットの examples ディレクトリにある 2 つのアプレットを試してみました。WebLogic クラスは、ローカル マシン（NT サーバ）と別のマシン（Windows 95 クライアント）にインストールしました。ブラウザは使用しておらず、単に Appletviewer でアプレットを実行しようとしているだけです。NT サーバから Appletviewer を起動したときはアプレットは正常に動作しましたが、Windows 95 クライアントからはまったく動作しません。

A. 考えられる原因は 2 つあります。CODEBASE タグがアプレットの HTML ファイルで正しく設定されていないか、クラス ファイルが HTTP サーバで正しくロードされていません。

アプレットが NT サーバで動作するのは、NT サーバに WebLogic 配布キットがインストールされているためです。アプレットは HTTP サーバから必要なクラスをロードするのに失敗しても、ローカルの CLASSPATH でクラスを探します。

しかし、Windows 95 クライアントからアプレットを実行しようとすると、アプレットは HTTP サーバからネットワーク経由でクラスをロードする必要があり、クラスが正しくインストールされていなければアプレットの実行は失敗します。

Q. アプレットで ClassFormatErrors が発生するのはなぜですか。

問題 : 配布キットをダウンロードし、HTTP サーバの DocumentRoot にクラスをコピーしました。そしてアプレットを作成し、Netscape サーバからの実行に成功しました。アプレットはサーバ ディレクトリの `\webz\ns-home\classes\applets\myapp.class` に配置し、次のように呼び出しました。

```
<APPLET
  CODEBASE=http://myserver.com/webz/ns-home/classes
  CODE=applets.myapp.class>
```

次に、ポート 7001 でリスンするように Administration Console で属性を設定し、そして WebLogic JDBC でアプレットを使用できるように WebLogic Server を HTTP マシンで起動しました。この場合は、次のように記述しました。

```
<APPLET
       CODEBASE=t3://myserver.com:7001/webz/ns-home/classes
       CODE=applets.myapp.class>
```

`CODEBASE` タグを WebLogic Server を指すように変更すると、ClassFormatErrors が表示されるようになりました。

A. この設定にはいくつかの問題があります。最も明らかな問題は、`CODEBASE` と関係があります。

1. アプレットの `CODEBASE` タグは、WebLogic Server ではなく HTTP サーバを指すように設定する必要があります。

2. `CODEBASE` タグで参照するディレクトリ パスは、HTTP サーバ上の絶対パスではなく、HTTP Document Root を基点とするパスです。この場合、`CODEBASE` タグには絶対パスが設定されています。「myapp」クラスが「applets」パッケージに含まれている場合、`CODEBASE` の正しい設定は次のようになります。

```
<APPLET
       CODEBASE=http://myserver.com/classes
       CODE=applets.myapp.class>
```

また、ClassFormatError は、HTTP サーバのコンフィグレーションに問題があることを示します。WebLogic またはアプレット クラスを HTTP サーバの適切なディレクトリにロードしていないか、または APPLET タグ内の CODEBASE または CODE を間違って指定している可能性があります。

また、アプレットを実行するマシンに WebLogic 配布キットをインストールした場合、アプレットはまずローカル CLASSPATH で WebLogic クラスを探すということも覚えておいてください。この場合、HTTP サーバから使用するようには、クラスはインストールされていません。HTTP コンフィグレーションを正確にテストするには、ローカル CLASSPATH で一時的に WebLogic クラスの名前を変更するか、別のマシンからアプレットを試す必要があります。

3 FAQ: クラスタ化

- WebLogic Server クラスタでスタブはどのように機能するのですか。
- 障害が発生し、スタブが WebLogic Server インスタンスに接続できない場合はどうなりますか。
- サーバは、別のサーバが利用できなくなったときに、どのようにしてそれを知るのですか。
- クラスタにサーバが追加されたとき、その通知はどのように行われるのですか。
- クライアントは、新しい WebLogic Server インスタンスのことをどのようにして知るのですか。
- 機能の停止したサーバへの DNS リクエストを、クライアントはどのように処理するのですか。
- 複数の CPU を搭載するマシン上で実行できる WebLogic Server の数はいくつですか。
- クラスタでのマルチキャスト用に別のネットワークを使用する必要がありますか。
- クラスタが「ハング」または「フリーズ」してしまった場合は、どうすればよいでしょうか。

Q. WebLogic Server クラスタでスタブはどのように機能するのですか。

A. WebLogic Server クラスタに接続して、クラスタ化されたオブジェクトをルック アップするクライアントは、そのオブジェクトのレプリカ対応スタブを取得します。このスタブには、そのオブジェクトの実装のホストとして使用可能なサーバ インスタンスのリストが入っています。スタブには、ホスト サーバ間で負荷を分散するためのロードバランシング ロジックも含まれています。

Q. 障害が発生し、スタブが WebLogic Server インスタンスに接続できない場合はどうなりますか。

A. 障害が発生すると、スタブでは機能しなくなったサーバ インスタンスがリストから削除されます。リストにサーバが残っていない場合には、スタブは再度 DNS を使用して動作中のサーバを検索し、動作中のインスタンスの最新リストを取得します。さらにスタブでは、クラスタ内の利用可能なサーバ インスタンスのリストが定期的に更新されます。この更新によって、スタブでは新しいサーバがクラスタに追加されるのに応じて、それらを利用できるようになります。

Q. サーバは、別のサーバが利用できなくなったときに、どのようにしてそれを知るのですか。

A. WebLogic Server では、2 つのメカニズムを使用して、特定のサーバ インスタンスが利用できるのかどうかを確認します。

クラスタ内の各 WebLogic Server インスタンスは、マルチキャストを使って定期的に「ハートビート」メッセージをブロードキャストし、自分が利用可能であることを通知します。クラスタ内のサーバ インスタンスは、ハートビート メッセージをモニタすることで、他のサーバ インスタンスでの障害発生を検出します。あるサーバ インスタンスからのハートビートを 3 回連続して受信しないと、他のサーバ インスタンスはそのサーバ インスタンスをクラスタから削除します。

WebLogic Server は、サーバ インスタンスが利用できるかどうかの判断に、ソケット エラーも利用します。たとえば、サーバ インスタンス A がサーバ インスタンス B に対するソケットを開いていて、そのソケットが不意に閉じた場合、サーバ A はサーバ B がオフライン状態になったものと見なします。

Q. クラスタにサーバが追加されたとき、その通知はどのように行われるのですか。

A. クラスタに新しいインスタンスが加わるたびに、WebLogic Server クラスタは新しいサーバ インスタンスが利用可能であることをブロードキャストします。また、クラスタ対応スタブでも、利用可能なサーバ インスタンスのリストが定期的に更新されます。

Q. クライアントは、新しい WebLogic Server インスタンスのことをどのようにして知るのですか。

A. クライアントは、いったん JNDI ルックアップを済ませてオブジェクト参照を使い始めると、クラスタ対応スタブが利用可能なサーバのリストを更新した後でしか、新しいサーバ インスタンスの存在に気付きません。

Q. 機能の停止したサーバへの DNS リクエストを、クライアントはどのように処理するのですか。

A. サーバが機能しなくなった後、利用できないマシンに DNS がリクエストを送り続けると、帯域幅の浪費につながります。Java クライアント アプリケーションの場合、この問題は起動時にしか発生しません。WebLogic Server は DNS エントリをキャッシュし、利用できないエントリを削除して、機能しなくなったサーバにクライアントが再びアクセスしないようにします。

サーバの障害は、ブラウザ ベースのクライアントにとってはもっと大きい問題となる可能性があります。ブラウザ ベースのクライアントは常に DNS を利用するからです。ブラウザ ベースのクライアントで不必要な DNS リクエストを発行しないようにするには、Resonate、BigIP、Alteon、LocalDirector といったサードパーティのロード バランサを使用します。これらの製品は、複数の DNS アドレスを単一のアドレスに見せかけます。さらにこれらの製品は、ラウンドロビンよりも高度なロードバランシング オプションを提供するほか、機能しなくなったサーバを追跡して不必要なリクエストをルーティングしないようにします。

Q. 複数の CPU を搭載するマシン上で実行できる WebLogic Server の数はいくつですか。

A. 考え得るコンフィグレーションは多数あり、それぞれに利点と欠点があります。BEA WebLogic Server には、クラスタ内のサーバ インスタンス数に関する制限はありません。したがって、Sun Microsystems, Inc. の Sun Enterprise 10000 などの大規模マルチプロセッサ サーバは、大規模なクラスタまたは複数のクラスタのホストとなることができます。

ほとんどの場合、WebLogic Server クラスタは、2 つの CPU につき 1 つの WebLogic Server インスタンスの割合でデプロイするのが最適です。ただし、すべてのキャパシティ プランニングと同じように、サーバ インスタンスの最適数および分散方法を決定する場合は、対象となる Web アプリケーションで実際のデプロイメントを事前にテストする必要があります。詳細については、「複数の CPU を搭載するマシンでのパフォーマンスについての考慮事項」を参照してください。

Q. クラスタでのマルチキャスト用に別のネットワークを使用する必要がありますか。

A. いいえ。マルチキャスト トラフィックは、別のネットワークを必要とするほど重くはありません。

Q. クラスタが「ハング」または「フリーズ」してしまった場合は、どうすればよいでしょうか。

A. WebLogic Server クラスタが「フリーズ」した場合は、BEA テクニカル サポートに連絡する前に、スレッド ダンプや Java ガベージ コレクション メトリックなどの診断情報を収集する必要があります。詳細については、「診断情報の収集」を参照してください。

4 FAQ: コード例

- コード例はどこに存在しますか。
- コード例が動作しないのはなぜですか。
- Cloudscape の評価版は WebLogic でどのように使用するのですか。
- `build.cmd` スクリプトと `build.sh` スクリプトは依然として使用されていますか。
- ANT はどのように使用するのですか。

Q. コード例はどこに存在しますか。

A. コード例は、インストールされている場合、WebLogic Server の `\samples\examples` ディレクトリに格納されており、[スタート] メニューからアクセスできます。

Q. コード例が動作しないのはなぜですか。

A. 各コード例では、サンプル クラス ファイルの作成、サーバのコンフィグレーション、およびコード例の実行のための詳しい指示が用意されています。それぞれの指示を完全に実行するようにしてください。

通常、例に関する問題は実行環境に関係します。以下にトラブルシューティングのヒントを示します。

1. データベースを使用する場合は、`utils.dbping` ユーティリティを実行して、JDBC ドライバが正しくインストールおよびコンフィグレーションされていることを確認してください。
2. setEnv スクリプトを実行して、例を実行するシェルまたは DOS ウィンドウで CLASSPATH を正しく設定してください。詳細については、「開発環境の設定」を参照してください。
3. 例に関する指示をチェックして、コンパイルの前にコード内のユーザ固有の変数をすべて変更してください。

4. 例に関する指示に指定されているとおりに、-d オプションを使用してコンパイルし、クラス ファイルが適切なディレクトリに配置されるようにしてください。

例がアプレットである場合は、CODE と CODEBASE を調べるとともに、WebLogic Server が確実に動作しているようにしてください。

詳細については、WebLogic Server 配布キットの *samples\examples\examples.html* にある「WebLogic Server サンプル コード ガイド」を参照してください。

Q. Cloudscape の評価版は WebLogic でどのように使用するのですか。

A. Cloudscape データベース システムの評価版が WebLogic 配布キットに含まれていますが、コード例で使用できるようにするには正しく設定する必要があります。Cloudscape データベースを使用するすべてのコード例では、接続プールをコンフィグレーションするための指示が用意されています。詳細については、WebLogic Server 配布キットの *samples\eval\cloudscape\cloudscape.html* にある「WebLogic Server で Cloudscape Database を使用する」を参照してください。

Q. `build.cmd` スクリプトと `build.sh` スクリプトは依然として使用されていますか。

A. いいえ。これらは ANT に置き換えられました。

Q. ANT はどのように使用するのですか。

A. Windows では `setExamplesEnv.cmd` を、UNIX では `setExamplesEnv.sh` をそれぞれ実行して、サンプル ドメイン環境を設定します。サンプル ディレクトリに移動して「ANT」と入力すると、`build.xml` ファイルが作成されます。独自の構築スクリプトを作成する場合は、「`ANT -f myBuild.xml`」と入力して、構築スクリプトの名前を渡します。ここで *myBuild* は、作成する構築スクリプトの名前です。詳細については、Apache Web サイトを参照してください。

5 FAQ: dbKona

- JDBC と dbKona はどのように関連しているのですか。
- dbKona は、リモート DBMS と通信するためにベンダ特有のネイティブ ライブラリを使用しますか。
- dbKona の場合と、JDBC を介した直接の SQL の場合で、パフォーマンスはどのように違うのですか。
- 変更の保存の際、dbKona TableDataSet はどうなるのですか。

Q. JDBC と dbKona はどのように関連しているのですか。

A. Java Database Connectivity（JDBC）は JavaSoft の仕様です。この仕様は、データベースにアクセスする Java アプリケーションを記述するための標準的な方法を規定しています。Sun では、「より高いレベルのツールやインタフェースを構築できる共通の基盤」と JDBC を定義しています。*dbKona* は JDBC より高レベルのインタフェースですが、WebLogic jDriver JDBC ドライバなどの JDBC ドライバを必要とします。dbKona は、ベンダに依存しない快適なデータ アクセスを実現します。dbKona では、クエリ結果のクライアントサイド管理と自動 SQL 生成が可能です。dbKona オブジェクトを使用すれば、データベース データを操作するためにベンダ固有の知識を得る必要がありません。さらに、dbKona オブジェクトは JDBC インタフェースの準拠と相互運用性を実現します。たとえば、dbKona を使用すると、さまざまなデータベースと対話できる単一の Java アプリケーションを記述できます。

dbKona はプラグアンドプレイ API（さまざまな DBMS を同時にサポートする手段）として設計されているため、JDBC 仕様に準拠したどのベンダのドライバでも dbKona と組み合わせて使用できます。dbKona を使用するアプリケーションは、WebLogic の JDBC ドライバを含むあらゆる JDBC ドライバとスムーズに連携します。

WebLogic jDriver 製品と dbKona は、JDBC との互換性と dbKona のパワーを組み合わせて提供します。

Q. dbKona は、リモート DBMS と通信するためにベンダ特有のネイティブ ライブラリを使用しますか。

A. dbKona は、任意の JDBC ドライバを使用してリモート DBMS と通信できます。2 層環境では、Type 2 JDBC ドライバを使用する場合は、dbKona アプリケーションを使用する各クライアントに、ベンダが提供するライブラリ（または適切な ODBC ドライバ）が必要となります。あるいは、ベンダのライブラリを必要としない Type 4 JDBC ドライバ（pure-Java）を使用することもできます。この 3 層配置では、dbKona は上位レベルのインタフェースとして JDBC ドライバの上に位置します。Type 2 JDBC ドライバを使用する場合、JDBC ドライバはベンダのライブラリを呼び出してデータベースにアクセスします。

WebLogic プール、JTS、および RMI JDBC ドライバの場合、dbKona アプリケーションまたはアプレットの代わりに、WebLogic Server を介して JDBC ドライバを呼び出すので、ベンダ ライブラリはクライアントサイドには必要ありません。

Q. dbKona の場合と、JDBC を介した直接の SQL の場合で、パフォーマンスはどのように違うのですか。

A. dbKona は非常に軽量です。JDBC を直接使用した場合のデータ検索と比べて、パフォーマンスの低下はほとんどありません。その利点はオーバーヘッドをはるかに上回ります。よりシンプルで便利な API でデータにアクセスでき、サンプルを使用したクエリを通じて SQL の自動生成が可能になります。また、dbKona DataSets を使用すると、dbKona TableDataSet を WebLogic Server 側のクライアント ワークスペースに格納するなど、JDBC ResultSet では不可能なことができます。

クエリの結果を管理するツールを dbKona が提供するので、WebLogic の範囲内ではクライアントはむしろパフォーマンスの向上を*実感*するでしょう。先読みするために、データを WebLogic 上にキャッシュすることもできます。複数のクライアントで、クエリの結果を共有できます。「EventfulTableDataSet」を使用すると、同じテーブルを参照しているすべてのクライアントに対し、あるクライアントでの変更内容を反映させることも可能です。dbKona は、そのままの JDBC よりも強力なデータ処理が可能なので、クライアント アプリケーションを調整して最適なパフォーマンスを得ることができます。

Q. 変更の保存の際、dbKona TableDataSet はどうなるのですか。

A. `TableDataSet` に対する変更を保存した後、その結果はそのまま維持され、`fetchRecords()` メソッドを使用して引き続きレコードにアクセスできます。

`TableDataSet` の作成時に使用した JDBC 接続とは異なる JDBC 接続で `TableDataSet` を保存しても、`TableDataSet` 中のデータの整合性に影響はありません。もちろん、保存に使用した接続に、更新のための使用を不可能にするような何らかの履歴や状態（たとえば、更新のコミットを妨害したり、他の接続のロックによりデッドロックを起こすようなトランザクション状態である場合）が

あれば、必ずしも影響がないとはいえません。各接続がクライアントサイドで連携していても、DBMS では各接続を独立したアプリケーションであるとみなす必要があります。

6 FAQ: EJB

- JDBC コードがロールバックの SQLException を送出した理由は何ですか。
- Bean 管理の永続性メカニズムでは WebLogic JTS ドライバを使用する必要がありますか。
- create() メソッドまたは find() メソッドから、ポリモフィック型の応答がないのはなぜですか。
- EJB はクラスタ全体に均一にデプロイする必要がありますか。なぜでしょうか。

Q. JDBC コードがロールバックの SQLException を送出した理由は何ですか。

A. JDBC コードは次の例外を送出する場合があります。

```
"The coordinator has rolled back the transaction.
     No further JDBC access is allowed within this transaction."
```

WebLogic JTS JDBC ドライバがこの例外を送出するのは、現在の JDBC 接続トランザクションが JDBC 呼び出しの前または途中でロールバックされた場合です。この例外は、JDBC 接続が関わっていたトランザクションが JDBC 呼び出しの前または途中でロールバックされたことを示します。

ロールバックはトランザクションの一部である EJB の呼び出しで発生したか、トランザクションのタイムアウトが原因で発生しています。どちらの場合でも、トランザクションはロールバックされ、接続はプールに戻り、データベース リソースは解放されます。処理を進めるには、JTS JDBC 接続を閉じて、新しいトランザクションで再び開く必要があります。

Q. Bean 管理の永続性メカニズムでは WebLogic JTS ドライバを使用する必要がありますか。

A. Bean 管理の永続性には TxDataSource を使用することをお勧めします。

Q. create() メソッドまたは find() メソッドから、ポリモフィック型の応答がないのはなぜですか。

A. EJB 仕様ではこの動作は許されていません。weblogic.ejbc コンパイラは、この動作を厳密にチェックして、`create()` メソッドまたは `find()` メソッドからポリモフィック型の応答が戻るのを防止しています。

`create()` メソッドと `find()` メソッドがポリモフィックでない理由は、Java でコンストラクタがポリモフィックでない理由とほぼ同じです。派生クラスは、通常、基本クラスを認識しないか、正しく初期化することができません。

Q. EJB はクラスタ全体に均一にデプロイする必要がありますか。なぜでしょうか。

A. はい。WebLogic Server バージョン 6.0 から、EJB は以下の理由によりクラスタ全体に均一にデプロイされなければなりません。

- クラスタ化 EJB をシンプルに保つため。
- クロス サーバ呼び出しを回避することによって能率を上げるため。EJB がすべてのサーバにデプロイされていない場合、クロス サーバ呼び出しが発生する可能性が高まります。
- すべての EJB がローカルに使用できるようにするため。
- すべてのクラスタがデプロイ不可能な方法でロードされるようにするため。
- 各サーバは、ローカル JNDI ツリーにバインドされるよう各 EJB のクラスにアクセスしなければなりません。サーバのサブセットだけが EJB をデプロイした場合、他のサーバはその Bean のクラスをそれぞれのシステム クラスパスにロードする必要があります。これにより、Bean をアンデプロイすることができなくなります。

7 FAQ: インストール

- WebLogic Server 6.1 用に使用できるプラットフォームは何ですか。
- WinZip ファイルを使用して、Compaq Tru64 Unix 用の WebLogicServer 6.0 をダウンロードおよびインストールする方法は。
- WebLogic Server のインストール ファイルをダウンロードしましたが、インストール プログラムが実行できません。どうしたらよいですか。

Q. WebLogic Server 6.1 用に使用できるプラットフォームは何ですか。

A. Windows NT/2000 および Solaris です。

Q. WinZip ファイルを使用して、Compaq Tru64 Unix 用の WebLogicServer 6.0 をダウンロードおよびインストールする方法は。

A. WinZip ファイルは通常展開されて実行されるので、このファイルは `.jar` ファイルと同じようにクラスパスに挿入する必要があります。

この Java ベースのインストーラは、解凍せずに実行してください。

```
java -classpath weblogic600_tru64.zip install
```

Q. WebLogic Server のインストール ファイルをダウンロードしましたが、インストール プログラムが実行できません。どうしたらよいですか。

A. ダウンロード中にインストール ファイルが破損した可能性があります。インストール ファイルに対してチェックサムを実行し、適切な値をテクニカル サポートに確認してください。

8 FAQ: Java

- プログラムのデバッグで支援を受けることができますか。
- Java 学習の材料はどこで入手できますか。
- JDK はどこで入手するのですか。
- CLASSPATH はどのように設定するのですか。
- コード例が動作しないのはなぜですか。
- Java エラー メッセージに関するヘルプはどこにあるのでしょうか。
- クライアントとサーバ間のメッセージで StackOverflowException が生成されるのはなぜですか。
- JIT を使用すれば Java アプリケーションの実行速度が上がりますか。
- WebLogic Server にバンドルされている JDK を再配布できますか。

Q. プログラムのデバッグで支援を受けることができますか。

A. 問題が BEA のソフトウェアに直接関係していない場合は、デバッグを支援する Java 開発ツールの使用、および Java の学習を手助けする書籍の購入やトレーニングの受講を提案します。プログラムにデバッグ機能を組み込む方法は多数存在し、Java プログラミングの優れたトレーニングを受けることはその方法を理解するための適切な出発点です。

Q. Java 学習の材料はどこで入手できますか。

A. Java に関しては書籍やオンライン リファレンスが多数あります。手始めに、JavaSoft の Web サイトでドキュメントの索引を参照してください。ここには、報告書や Java チュートリアルへのリンクがあります。Java 関連の書籍は、大規模なオンライン書籍販売サイトならどこでも見つけることができます。

Q. JDK はどこで入手するのですか。

A. WebLogic 6.1 には、JDK 1.3.1 がバンドルされています。テスト済みで、WebLogic ソフトウェアで使用できることが保証されている特定の JDK に関する

情報については、BEA の「プラットフォーム サポート」ページを参照してください。

どのバージョンの JDK を使用するのかを決めたら、JavaSoft の Web サイトに行ってください。多くのプラットフォーム ベンダが、自社のコンピュータ向けに最適化した JDK を提供しています。

Q. CLASSPATH はどのように設定するのですか。

A. CLASSPATH の設定は、何をしようとしているのかによって異なります。共通の作業は以下のとおりです。

- **WebLogic Server を起動します。**『管理者ガイド』の「WebLogic Server の起動と停止」で「クラスパス オプションの設定」を参照してください。また、WebLogic 配布キットにはサーバの起動に使用できるシェル スクリプトが含まれています。WebLogic Server 配布キットの *config* ディレクトリ内のドメイン ディレクトリに配置されているこれらのスクリプトは、サーバを起動する前にシェルで CLASSPATH 変数を自動的に設定します。
- **アプリケーション クラスをコンパイルするか、WebLogic Server ユーティリティを使用します。**『WebLogic Server アプリケーションの開発』の「WebLogic Server コンポーネントの開発」の「コンパイル用のクラスパスの設定」を参照してください。
- **WebLogic Server のコード例を使用します。**WebLogic Server 配布キットの `samples\examples\examples.html` にある「WebLogic Server サンプル コード ガイド」を参照してください。

Q. コード例が動作しないのはなぜですか。

A. 通常、例に関する問題は実行環境に関係します。以下にトラブルシューティングのヒントを示します。

1. データベースを使用する場合は、`utils.dbping` ユーティリティを実行して、JDBC ドライバが正しくインストールおよびコンフィグレーションされていることを確認してください。
2. setEnv スクリプトを実行して、例を実行するシェルまたは DOS ウィンドウで CLASSPATH を正しく設定してください。詳細については、「開発環境の設定」を参照してください。
3. 例に関する指示をチェックして、コンパイルの前にコード内のユーザ固有の変数をすべて変更してください。

4. 例に関する指示に指定されているとおりに、-d オプションを使用してコンパイルし、クラス ファイルが適切なディレクトリに配置されるようにしてください。

例がアプレットである場合は、CODE と CODEBASE を調べるとともに、WebLogic Server が確実に動作しているようにしてください。

Q. Java エラー メッセージに関するヘルプはどこにあるのでしょうか。

A. BEA に寄せられる質問の多くは一般的な Java エラー メッセージ関連であり、WebLogic に特有のものではありません。Java エラー メッセージに関する有益な情報を入手できる参照先を以下に示します。

参照先	説明
Sun の Java Developer Connection	このフォーラムには、エラー メッセージを含むさまざまな Java 関連トピックの質問と回答がある。すぐに結果を得たい場合には、[Search] ボックスを使用する。たとえば、[Search] ボックスに「classpath error」のように入力する
Compiler Error Messages	広範囲にわたるコンパイラ エラー メッセージのリスト。このリストには不評な NoClassDefFoundError も含まれている
Sun の Java API	Java API をチェックし、使用するクラスに例外の説明があるかどうかを確認できる

Q. クライアントとサーバ間のメッセージで StackOverflowException が生成されるのはなぜですか。

A. java.io.Serialization を使用して特別に大きなデータ構造を送信する場合は、Java またはネイティブ スタックのスレッド単位のサイズ制限を超えている可能性があります。スタック サイズは、以下のコマンドライン オプションで増やすことができます。

ネイティブ スタックのサイズを増やす場合は `-ss` *`Stacksize`*
Java スタックのサイズを増やす場合は `-oss` *`Stacksize`*

Stacksize では、整数に続いてキロバイトの「k」またはメガバイトの「m」を指定します。次に例を示します。

```
$java -ss156k (native)
$java -oss600k (Java)
```

デフォルトのネイティブ スタック サイズは 128KB で、最小値は 1000 バイトです。デフォルトの *java* スタック サイズは 400KB で、最小値は 1000 バイトです。

Q. JIT を使用すれば Java アプリケーションの実行速度が上がりますか。

A. Just-In-Time コンパイラを使用すれば一部の Java アプリケーションの実行速度は向上します。JIT は、生成したマシン コードをメモリに格納し、できる限りそれを再利用することで有効に作用します。たとえば、同じ処理をループで 1000 回実行する場合、コードの生成は 1 回だけなので、JIT を使用すればその処理のパフォーマンスが向上します。多くのネイティブ メソッドを使用するアプリケーションでは、pure-Java アプリケーションほどのパフォーマンスの向上は望めません。

JIT を使用する場合、スタック トレーシングを容易にするために、デバッグ中は JIT を使用しない方が良いでしょう。JIT を使用してパフォーマンスをテストする場合は、同じ呼び出しで同じテストを必ず複数回実行し、最初の結果は破棄して、アプリケーションが安定した状態で動作しているときにトランザクションにどのくらいの時間がかかるのかを確認してください。初回のコード実行時は、テストは長くかかります（コードが生成されるため）。

Q. WebLogic Server にバンドルされている JDK を再配布できますか。

A. BEA Systems は、独立ソフトウェア ベンダ（ISV）などの第三者に対して、WebLogic Server にバンドルされている JDK を一切の修正なしに再配布する権利を付与するための独占的権利を有しています。以下に、この一般表明に対する警告を示します。

- ISV は、JDK に含まれているあらゆる固有の凡例または注意を削除することができません。ISV は、JDK の逆コンパイル、逆アセンブル、復号化、抽出、またはその他のリバース エンジニアリングを行ってはなりません。JDK は、一部または全部を問わずリースまたは譲渡できません。
- ISV は、この再配布方針の説明と実質的に同様の条件で、その配布者と署名入りの契約を締結しなければなりません。
- ISV は、WebLogic Server を組み込む製品のエンド ユーザ ライセンス契約を必要とします。
- ISV 製品への JDK の組み込みには、JDK プロバイダによる JDK のメンテナンスとサポートとは含まれません。BEA Systems は、その ISV および配布者に対してメンテナンスとサポートを提供する単独の責任を負うものとしま

す。ISV は、その製品のエンド ユーザに対してメンテナンスとサポートを提供する単独の責任を負うものとします。

- BEA が WebLogic Server にバンドルした JDK とは異なる JDK を ISV が出荷する場合、その ISV はそれらのバンドル権を直接 Sun または HP から取得する必要があります。次に例を示します。

  BEA が WebLogic Server 6.0 と JDK 1.3 のみ、および WebLogic Server 5.1 と JDK 1.1 のみを出荷し、ISV がその統合製品で WebLogic Server 6.0 と JDK 1.1 を出荷したいとします。この場合、BEA がそのビジネス上の理由により WebLogic Server 6.0 に JDK 1.1 をバンドルすることを選択しない限り、ISV は BEA との契約または BEA と Sun との契約に基づいてその統合製品で JDK 1.1 を WebLogic Server 6.0 にバンドルして出荷することはできません。ただし、ISV は、JDK のバイナリ配布契約を Sun から独自に取得して、その契約に基づいて JDK 1.1 をその付加価値ソフトウェア ソリューション（ISV のアプリケーションと WebLogic Server 6.0 で構成される）にバンドルすることができます。

9 FAQ: J2EE コネクタ アーキテクチャ

Q. JNDI ツリーを参照しているときに次の例外を受け取るのはなぜですか。

```
isSerializable(class.javax.naming.Binding)
java.io.NotSerializableException:
java.io.PrintWriter at
java.io.ObjectOutputStream.OutputObject
```

A. Weblogic Server JNDI の実装では、オブジェクトは参照可能ではなくシリアライズ可能である必要があります。PrintWriter はシリアライズ可能ではないので、transient として宣言する必要があります。

Q. WebLogic J2EE コネクタ アーキテクチャの現在の実装で Cloudscape ではなく Oracle データベースを使用することはできますか。

A. 付属のサンプルには、任意のデータベース システムを表すリソース アダプタが含まれています。デフォルトでは、コンフィグレーションは Cloudscape を使用するよう設定されています。特に、`weblogic-ra.xml` ファイルのコンフィグレーション プロパティには Cloudscape データソースが設定されています。この設定は、Oracle の設定に置き換えることができます。

また、リソース アダプタ（特に `ManagedConnectionFactory`）は Oracle をサポートするよう実装されなければなりません。このサンプルに含まれているリソース アダプタは JDBC を利用するので、表すことができるようコンフィグレーションされている任意のデータベース システムをサポートできます。

Q. リソース アダプタ（`.rar`）を WebLogic Server にデプロイする場合、そのクラスは WebLogic クラスパスに配置されますか。

たとえば、EJB とリソース アダプタ（.rar）をデプロイしようとしていますが、EJB は CCI（Common Client Interface）に書き込むため、`.rar` に対する依存関係が存在しません。EJB クライアント アプリケーションは、`.rar` で定義されるパラメータ クラスとして送信 / マーシャリングを持っています。何らかの理由により、`.rar` が正常にデプロイされたにもかかわらず、EJB のクラス ローダ階層がこの `.rar` 固有クラスの定義を発見できません。EJB クライアントで次のエラーを受け取りました。

```
java.rmi.UnmarshalException: error unmarshalling arguments; nested
exception
is:
java.lang.ClassNotFoundException:
com.mycompany.InteractionSpecImpl
```

A. `com.myclientcompany.server.eai.InteractionSpecImpl` のインスタンスを引数として EJB に渡す場合、`appServer` は EJB のコンテキストでオブジェクトをデシリアライズ（アンマーシャリング）する必要があります。また、`ejb-jar`（`raTester.jar`）の内部にあるアンマーシャリング用の必須クラスを必要とします。このため、`interactionspecimpl` クラスを `ejb-jar` ファイルに含める場合、これらのクラスをサーバのクラスパスに含める必要はありません。

Q. サンプル EJB が `ConnectionFactory.getConnection()` を呼び出して EIS に接続するときに、WebLogic Server が `ManagedConnection.addConnectionEventListener()` 関数を呼び出すのはなぜですか。

A. これは必須であり、リソース アダプタとアプリケーション サーバ間の取り決めの一部です。

Q. EJB をコンパイルして CCI をサポートするリソース アダプタを使用するときに例外が送出されるのはなぜですか。

`resource-ref` で `javax.resource.cci.ConnectionFactory` を指定しましたが、EJB をコンパイルしようとすると次の例外を受け取ります。

```
weblogic.xml.process.SAXValidationException:
ejb-jar.enterprise-beans.session.resource-ref.res-type.
must be one of the values:
javax.sql.DataSource,javax.jms.QueueConnectionFactory,
javax.jms.TopicConnectionFactory,
java.net.URL,
javax.mail.Session
at
weblogic.ejb20.dd.xml.EjbJarLoader_EJB11.__post_84
```

A. `ejb-jar.xml` が EJB1.1 DTD ではなく EJB2.0 DTD を参照しているかどうかを確認してください。`ConnectionFactory resource ref` は、EJB 2.0 DTD だけでサポートされています。

Q. WebLogic J2EE コネクタ アーキテクチャではセキュリティはどのように扱われていますか。

A. WebLogic Server のリソース アダプタの現在のコンフィグレーションとパッケージ要件により、`weblogic-ra.xml` ファイルを手動で編集する必要があります。このため、security-principal-map エントリに指定する新しいパスワードはクリアテキスト形式で指定します。

BEA は、セキュリティ パスワードの保護がどのくらい重要であるかを認識しています。このため、`weblogic-ra.xml` ファイルに存在するすべてのパスワードを暗号化するための変換ツールを提供しています。変換ツールは、標準の `weblogic.jar` ファイルで提供されます。パスワード変換ツールの詳細については、『WebLogic J2EE コネクタ アーキテクチャ』（http://edocs.beasys.co.jp/e-docs/wls61/jconnector/index.html）の「コンフィグレーション」セクションの「パスワード変換ツール」を参照してください。

Q. BEA の `com.bea.adapter.dbms.cci.ConnectionImpl` は直接 `javax.resource.cci.Connection` を実装しません。この解決策はありますか。

A. はい。BEA `com.bea.adapter.dbms.cci.ConnectionImpl` は、`com.bea.adapter.cci.AbstractConnection` の拡張であり、これは Connection インタフェースを実装します。プロキシは最終派生クラス（`ConnectionImpl`）からのインタフェースを使用して構築されます。`dumpFamilyTree` 出力は、`ConnectionImpl` クラスの `getInterfaces` 呼び出しに Connection インタフェースが含まれていないことを示します。しかし、`AbstractConnection` の `getInterfaces` 呼び出しには、Connection インタフェースが含まれます。

この問題を解決するには、ConnectionImpl クラスが ra.xml ファイルに指定されているインタフェース クラスを直接実装する必要があります（特にこれが実装済みのクラスを拡張する場合、この文は冗長になる場合があります）。次に、アダプタを再構築し、もう一度テストを行ってみてください。

10 FAQ: WebLogic JDBC

Q. マルチプールはどのような場合に使用するのですか。

A. 単一サーバのコンフィグレーションで WebLogic Server を使用している場合は、データベース接続が失敗した場合のバックアップ マルチプールとして、またはロードバランシング マルチプールとしてマルチプールを使用できます。どちらかの方法しか選択できないので、マルチプールの主な目的を決める必要があります。詳細については、『WebLogic JDBC プログラミング ガイド』の「WebLogic JDBC 機能のコンフィグレーション」で「マルチプールの使い方」を参照してください。

Q. データベースが利用可能かどうかは、どのように確認できますか。

A. 基本的には、接続を試してみる以外にデータベースがダウンしているかどうかを確認する方法はありません。

また、データベースは接続を行ったり、その接続を使用した後に使用できなくなることがあります。コードは、予期しないエラーを処理できるように記述することをお勧めします。エラーは、データベースがダウンしたときにクライアントが何をしているかによってさまざまなかたちで発生します。詳細については、『WebLogic JDBC プログラミング ガイド』の「WebLogic JDBC 機能のコンフィグレーション」で接続プールとマルチプールの説明を参照してください。

11 FAQ: WebLogic jDriver for Informix

- WebLogic jDriver for Informix でマルチバイト文字セットを使用する方法を教えてください。
- Informix データベースへのマルチユーザ アクセスを許可するにはどうすれば良いですか。

Q. WebLogic jDriver for Informix でマルチバイト文字セットを使用する方法を教えてください。

A. 現在、WebLogic jDriver for Informix ドライバではマルチバイト文字セットはサポートされていません。今後のリリースではサポートされる可能性もあります。詳細については、『WebLogic jDriver for Informix のインストールと使い方』でコードセットのサポートに関する説明を参照してください。

Q. Informix データベースへのマルチユーザ アクセスを許可するにはどうすれば良いですか。

A. WebLogic jDriver for Informix ドライバを使用してマルチユーザ アプリケーションを実行しているときに、エラー 271（`Could not insert new row into the table`）および 243（`Could not position within a table table-name`）が表示される場合は、まず、`bcheck` または `secheck` ユーティリティを使用してテーブル ファイルが壊れていないこと、または不完全ではないことを確認します。ファイルが正常である場合は、テーブル ファイルまたはテーブル内のレコードがロックされている可能性があります。これはマルチユーザ プログラムでは正常なイベントです。通常は、現在のトランザクションをロールバックし、少し待ってからトランザクションを再実行すると回復できます。以下に、具体的な回復手順を紹介します。

対話型の SQL の場合は、操作をやり直します。

- C-ISAM プログラムの場合は、プログラム ロジックを調べて、確実にこの状況を処理できるようにします。`ISEXCLLOCK` フラグを `ISOPEN` に渡すことによってテーブルに排他的にアクセスできます。
- SQL プログラムの場合は、プログラム ロジックを調べて、確実にこの状況を処理できるようにします。`SET LOCK MODE TO WAIT` 文を使用します。一

括更新については、LOCK TABLE 文、および DATABASE 文の EXCLUSIVE 句を参照してください。

12 FAQ: WebLogic jDriver for MSSQL Server

- Weblogic JDriver for MSSSQL Server は、NT/WIN2K の信頼された接続を使用してデータベース サーバに接続できますか。
- SQL Server 2000 の複数のインスタンスを持つマシン上で動作する SQL Server インスタンスに接続するにはどうすればよいですか。

Q. Weblogic JDriver for MSSSQL Server は、NT/WIN2K の信頼された接続を使用してデータベース サーバに接続できますか。

A. 当社のドライバは、信頼された接続をサポートしていません。信頼された接続をサポートすると、ドライバは Type 4 ではなくなってしまいます。

Q. SQL Server 2000 の複数のインスタンスを持つマシン上で動作する SQL Server インスタンスに接続するにはどうすればよいですか。

A. MS SQL Server の各インスタンスは、異なるポートをリスンする必要があります。このため、`getConnection()` メソッドに渡すプロパティでポート番号を使用するか、接続プールの場合は、以下のプロパティにポートのプロパティを指定できます。

```
server=machineName
port=instancePort
```

各 MS SQL Server インスタンスが実行されているポート番号を見つけるには、サーバ ネットワーク ユーティリティ（Microsoft SQL Server プログラム グループにある）を実行し、サーバ インスタンスを選択し、TCP/IP を選択して、プロパティ ボタンをクリックします。

13 FAQ: WebLogic jDriver for Oracle

- Oracle 8 で FOR UPDATE を使うと ORA-01002 エラーが発生するのはなぜですか。
- OCIW32.dll エラーの原因は何ですか。
- WebLogic jDriver for Oracle はどのトランザクション アイソレーション レベルをサポートしているのですか。
- WebLogic jDriver for Oracle ドライバで Unicode コードセットを使用するにはどうするのですか。
- WebLogic jDriver for Oracle および接続プールと共に OS 認証を使用するにはどうすれば良いですか。
- ResultSet.getObject() ではどの型のオブジェクトが返されるのですか。
- WebLogic Server で生成される Oracle データベース接続の数を制限するには、どうすればいいですか。
- パラメータをとらない Oracle ストアド プロシージャは、どのように呼び出すのですか。
- PreparedStatement の文字列値はどのようにバインドするのですか。
- WebLogic jDriver for Oracle を使用し、8 ビット文字セットを使用していますが、期待した文字が表示されません。何が問題なのでしょうか。
- Oracle で利用可能なコードセットは、どうしたらわかりますか。
- 「ORA」SQLException はどのように調べるのですか。
- エラー「ORA-6502」は何を意味するのですか。
- ORA-12705 のテキストを取り出そうとするとエラーが発生するのはなぜですか。

- Oracle のデータベース リンクを使ってデータベースを更新するとリソースが足りなくなってしまうのはなぜですか。
- t3dbping の実行時にエラーを防止する方法を教えてください。
- CLOB/NCLOB カラムからマルチバイト文字をアクセス使用とすると「ORA-03120」エラーを受け取るのはなぜですか。
- PreparedStatement クラスを実行すると「TRUNC fails: ORA-00932: inconsistent datatypes」エラーが発生するのはなぜですか。

Q. Oracle 8 で FOR UPDATE を使うと ORA-01002 エラーが発生するのはなぜですか。

A. Oracle 8 サーバでは、AUTOCOMMIT が**オン**の状態（JDBC を使用する場合のデフォルト）で FOR UPDATE 文を使用すると「`ORA-01002 フェッチ順序が無効です`」というエラー メッセージが生成されます。この問題は、Solaris の Oracle 8.0 と 8.1、そして、Windows NT の Oracle 8.1 で起こることがわかっています。AUTOCOMMIT を**オフ**にすれば、このエラーは起きなくなります。この問題は、Oracle 8 サーバの変更によるものなので、詳しい情報については Oracle のサポートに連絡してください。

Q. OCIW32.dll エラーの原因は何ですか。

A. Oracle 用 JDBC ドライバを使用すると、「The ordinal 40 could not be loaded in the dynamic link library OCIW32.dll.」というエラー メッセージが表示される場合があります。この問題は、システム ディレクトリにある OCIW32.DLL が旧バージョンであることが原因です。一部のプログラムは、実行時に使用するために、このファイルをシステム ディレクトリにインストールします。このファイルをシステム ディレクトリから削除すれば、このエラーは発生しなくなります。

Q. WebLogic jDriver for Oracle はどのトランザクション アイソレーション レベルをサポートしているのですか。

A. サーブレット アプリケーションは、Oracle Thin ドライバを使用して BLOB フィールドが含まれているデータベースにアクセスする場合があります。この場合、WebLogic jDriver for Oracle をインストールして使用するときに、同じコードが以下の例外によってエラーとなります。

```
com.roguewave.jdbtools.v2_0.LoginFailureException:
TRANSACTION_READ_UNCOMMITTED isolation level not allowed
  The Stack Trace:
com.roguewave.jdbtools.v2_0.LoginFailureException:
TRANSACTION_READ_UNCOMMITTED isolation level not allowed
  at
  com.roguewave.jdbtools.v2_0.jdbc.JDBCServer.createConnection
```

```
(JDBCServer.java :46)
at com.roguewave.jdbtools.v2_0.ConnectionPool.getConnection_
(ConnectionPool.jav a:412)
at com.roguewave.jdbtools.v2_0.ConnectionPool.getConnection
(ConnectionPool.java :109)
```

コードで、Isolation_level を 1 に設定すると、RogueWave JDBCServer クラスを呼び出しています。Oracle Thin ドライバは問題なく動作しますが、WebLogic jDriver for Oracle ではエラーとなります。

WebLogic jDriver for Oracle では、以下のトランザクション アイソレーション レベルがサポートされています。

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

Oracle のマニュアルによると、Oracle DBMS では上の 2 つのアイソレーション レベルしかサポートされていません。他の JDBC ドライバと違い、WebLogic のドライバは、サポートされていないアイソレーション レベルを使用しようとした場合に例外を送出します。一部のドライバは、サポートされていないアイソレーション レベルを設定しようとした場合に、例外を生成することなく無視します。サポートされていないアイソレーション イベントの設定を Oracle Thin ドライバが無視するのかどうかテストすることをお勧めします。

Q. WebLogic jDriver for Oracle ドライバで Unicode コードセットを使用するにはどうするのですか。

A. Unicode コードセットを使用するには、以下のようにします。

1. Oracle のインストール時に、適切なコードセットをインストールします。最初のインストール時にコードセットをインストールしなかった場合は、Oracle インストーラを再実行し、適切なコードセットをインストールする必要があります。

2. JDBC ドライバを実行している環境で、NLS_LANG 変数を定義します。WebLogic Server を起動するシェルで、適切なコードセットを NLS_LANG に割り当てることで定義できます。

インターナショナライゼーションのサポートの詳細については、開発者ガイドを参照してください。Unicode の一般的な情報については、Unicode の Web サイトを参照してください。Unicode 言語の略称については、JavaSoft の Web サイトを参照してください。

Q. WebLogic jDriver for Oracle および接続プールと共に OS 認証を使用するにはどうすれば良いですか。

A. OS 認証を接続プールで使用するということは、WebLogic Server を起動したユーザのユーザ ID を使用することになります。OS 認証は NT および UNIX で利用できますが、Solaris では利用できません。つまり、データベース セキュリティは WebLogic のセキュリティに厳密に依存します。したがって、WebLogic Server にクライアント接続してプールにアクセスできる場合は、データベースにアクセスできるということです。

Oracle ではプロセスのオーナを使用して、接続しようとしているユーザを判断するので、これは WebLogic jDriver for Oracle で実行できます。WebLogic JDBC の場合は、常に WebLogic Server を起動したユーザです。

この機能を使用するように Oracle のインスタンスを設定するには、DBA は以下の基本的な手順に従って操作する必要があります。詳しい手順については、Oracle のマニュアルを参照してください。

1. 次の行を INIT[sid].ORA ファイルに追加します。

   ```
   OS_AUTHENT_PREFIX = OPS$
   ```

   文字列「OPS$」は、DBA の判断で使用してください。

2. Oracle サーバに SYSTEM としてログインします。

3. `OPS$`*`userid`* という名前のユーザを作成します。*userid* は、オペレーティング システムのログイン ID にしてください。このユーザには、CONNECT や RESOURCE といった標準的な特権を割り当てます。

4. ユーザ ID を設定したら、ユーザ名プロパティとして「"/"」、パスワード プロパティとして「""」を指定して WebLogic jDriver for Oracle に接続できます。次に、`dbping` ユーティリティでこの接続をテストする例を示します。

   ```
   $ java utils.dbping ORACLE "/" "" myserver
   ```

   次に、WebLogic jDriver for Oracle のコード例を示します。

   ```
   Properties props = new Properties();
     props.put("user",       "/");
     props.put("password",   "");
     props.put("server",     "myserver");

     Class.forName("weblogic.jdbc.oci.Driver").newInstance();
     Connection conn = myDriver.connect("jdbc:weblogic:oracle",
        props);
   ```

5. Administration Console を使用して、接続プールの属性を設定します。次のコードは、WebLogic jDriver for Oracle を使用して JDBC 接続プールをコンフィグレーションする例です。

```
<JDBCConnectionPool
    Name="myPool"
    Targets="myserver,server1"
    DriverName="weblogic.jdbc.oci.Driver"
    InitialCapacity="1"

  MaxCapacity="10"
  CapacityIncrement="2"
  Properties="databaseName=myOracleDB"
```

Q. ResultSet.getObject() ではどの型のオブジェクトが返されるのですか。

A. WebLogic jDriver for Oracle は、取り出したデータの精度を維持する Java オブジェクトを常に返します。WebLogic jDriver for Oracle は、`getObject()` メソッドで以下を返します。

- 型が NUMBER(*n*) や NUMBER(*m,n*) の列の場合 : 定義された列の精度を Double で表すことができる場合は Double を返し、それ以外の場合は BigDecimal を返します。
- 型が NUMBER の列の場合 : 精度が明確でないため、各行の実際の値に基づいて、返す Java の型が判断されます。型は、行によって異なる場合があります。小数部がゼロで、値を整数で表現できる場合は Integer が返されます。

たとえば、1.0000 の場合は Integer が返されます。123456789123.00000 のような値の場合は Long が返されます。小数部がゼロでない値の場合、その値の精度を Double で表すことができる場合は Double、それ以外の場合は BigDecimal が返されます。

Q. WebLogic Server で生成される Oracle データベース接続の数を制限するには、どうすればいいですか。

A. クライアントの要求に応じて WebLogic Server が生成する Oracle データベース接続の数を制限する場合は、接続プールを使用できます。接続プールを使用すると、複数の T3 アプリケーションで固定数のデータベース接続を共有することができます。接続プールの設定方法については、『WebLogic JDBC プログラミング ガイド』を参照してください。

Q. パラメータをとらない Oracle ストアド プロシージャは、どのように呼び出すのですか。

A. 次の構文を使用してください。

```
CallableStatement cstmt = conn.prepareCall("Begin procName;
    END;");
cstmt.execute();
```

procName は、Oracle ストアド プロシージャの名前です。これは標準的な Oracle SQL 構文であり、どの Oracle DBMS でも動作します。次の構文も使用できます。

```
CallableStatement cstmt = conn.prepareCall("{call procName};");
 cstmt.execute();
```

このコードは、Java 拡張 SQL 仕様に準拠しており、Oracle だけでなくすべての DBMS で動作します。

Q. PreparedStatement の文字列値はどのようにバインドするのですか。

A. 文で String をバインドするために、PreparedStatement クラスを取得しようとしていると仮定します。setString() メソッドは、機能しないように見えます。PreparedStatement は次のように設定されています。

```
String pstmt = "select n_name from n_table where n_name LIKE
'?%'";
PreparedStatement ps = conn.prepareStatement(pstmt);
ps.setString(1, "SMIT");
ResultSet rs = ps.executeQuery();
```

このコードは機能しません。なぜなら、String で完全な値を指定し（引用符は埋め込まない）、引用符のない疑問符（?）にバインドする必要があるためです。正しいコードは次のようになります。

```
String matchvalue = "smit%";
String pstmt = "select n_name from n_table where n_name LIKE ?";
PreparedStatement ps = conn.prepareStatement(pstmt);

ps.setString(1, matchvalue);
ResultSet rs = ps.executeQuery();
```

Q. WebLogic jDriver for Oracle を使用し、8 ビット文字セットを使用していますが、期待した文字が表示されません。何が問題なのでしょうか。

A. 8 ビット文字セットの Oracle データベースを Solaris で使用している場合は、クライアントで `NLS_LANG` を適切な値に設定してください。`NLS_LANG` を設定しないと、デフォルトで 7 ビット ASCII 文字セットになり、ASCII 128 より大きい文字は適当な文字にマップされます。たとえば、á、à、â はすべて a にマップされます。その他の文字は、疑問符（?）にマップされます。

Q. Oracle で利用可能なコードセットは、どうしたらわかりますか。

A. Oracle で現在使用できるコードセットを調べるには、SQLPlus からコマンドラインで次の SQL クエリを実行します。

```
SQL> SELECT value FROM v$nls_valid_values
         WHERE parameter='CHARACTERSET';
```

現在システムにインストールされているすべてのコードセットのリストが返されます。このリストは、次のように表示されます。

```
VALUE
----------------------------------------------------------
US7ASCII
WE8DEC
WE8HP
US8PC437
WE8EBCDIC37
WE8EBCDIC500
WE8EBCDIC285
...
```

クエリの値を特定のコードセットに制限するには、次のような SQL クエリを使用します。

```
SQL> SELECT value FROM v$nls_valid_values
         WHERE parameter='CHARACTERSET' and VALUE='AL24UTFFSS';
```

そのコードセットがインストールされている場合は、次のようなリストが表示されます。

```
VALUE
----------------------------------------------------------
AL24UTFFSS
```

その他のコードセットを追加するには、Oracle のインストール ツールを使用します。詳細については、Oracle に問い合わせてください。

Q. 「ORA」SQLException はどのように調べるのですか。

A. WebLogic jDriver for Oracle アプリケーションでは、次のような SQLException が生成されることがあります。

```
java.sql.SQLException:ORA-12536:TNS:operation would block
```

Oracle エラーは、`oerr` コマンドを使用して調べることができます。たとえば、エラー ORA-12536 の説明は次のコマンドで表示できます。

```
> oerr ora 12536
```

Q. エラー「ORA-6502」は何を意味するのですか。

A. CallableStatement の `OUTPUT` パラメータにバインドされる文字列のデフォルト長は 128 文字です。バインド パラメータに割り当てた値がこの長さを超えると、このエラーが発生します。

バインド パラメータの値の長さは、明示的な長さを `scale` 引数を使って `CallableStatement.registerOutputParameter()` メソッドに渡すことによって調節できます。

Q. ORA-12705 のテキストを取り出そうとするとエラーが発生するのはなぜですか。

A. このエラーは、`ORACLE_HOME` 環境変数を適切に設定していないという、よくあるミスから発生します。WebLogic jDriver for Oracle を使用するには、Oracle クライアント ソフトウェアをインストールし、`ORACLE_HOME` を設定する必要があります。

システムにインストールされていない言語とコードセットの組み合わせで WebLogic jDriver for Oracle のインターナショナライゼーション機能を使用しようとした場合にも、このエラー メッセージが表示されます。適切なエラー テキストの ORA-12705 エラーが表示される場合は、`NLS_LANG` を適切に設定していないか、システムに正しいコードセットをインストールしていないかのどちらかです。

Q. Oracle のデータベース リンクを使ってデータベースを更新するとリソースが足りなくなってしまうのはなぜですか。

A. Oracle のデータベース リンクを使用してデータベースを更新する場合、終了時に結果セットと文を閉じても、「maximum number of temporary table locks exceeded」というエラーが表示される場合があります。

データベース リンクは、リモート データベースのテーブルやビューにアクセスできるようにするローカル データベースのオブジェクトです。データベース リンクは Oracle サーバによって管理されるので、ドライバではリソースの使用を管理できません。実際にはリンクが生じて処理が実行されますが（その他のプロセスでは、作成されたレコードが表示可能）、接続が終了するまでリソースは解放されません。データベース リンクを削除し、JDBC ドライバを使用して選択、挿入、および更新を実行することで解決してください。

Q. t3dbping の実行時にエラーを防止する方法を教えてください。

A. UNIX で Oracle データベース接続をテストする場合は、SQL*PLUS を実行し、`utils.dbping` を使用してデータベースに正常に ping を実行できます。ただし、多層 `utils.t3dbping` ユーティリティを使用すると、ORA-12154 エラーメッセージが表示されます。

まず、ORACLE_HOME 環境変数が Oracle のインストール先を指すように正しく設定します。この変数は、WebLogic サーバが動作している環境で設定しなければなりません。

C シェルで、次のコマンドを発行します。

```
$ setenv ORACLE_HOME path
```

path は、Oracle のインストール先へのパスです。

Bourne シェルで、次のコマンドを発行します。

```
$ ORACLE_HOME=path
$ export ORACLE_HOME
```

path は、Oracle のインストール先へのパスです。2 層 `utils.dbping` ユーティリティを使用してデータベースに ping を実行する場合は、JDBC ドライバがデータベース クライアント ライブラリをロードし、データベースへの接続を確立します。多層 `utils.t3dbping` ユーティリティを使用する場合は、WebLogic Server が 2 層ドライバをロードし、それを使用してデータベース接続を確立します。どちらの場合でも、同じメソッドを使用してデータベースに接続します。SQL*PLUS が機能するのは、クライアント ライブラリを見つけるのに ORACLE_HOME を必要としないからです。

まだ問題がある場合は、次の手順を試してみてください。

1. コマンド シェルを開きます。
2. このシェルで 2 層バージョンの `utils.dbping` を実行します。
3. このシェルでコマンドラインから WebLogic を起動します。

   ```
   $ java -ms32m -mx32m weblogic.server
   ```

4. 2 番目のコマンド シェルを開きます。
5. 最初のコマンドシェルで動作しているサーバに対して、2 番目のシェルで `utils.t3dbping` を実行します。

この方法で解決しない場合は、これらのコマンドの出力を WebLogic テクニカル サポートに送ってください。

Q. CLOB/NCLOB カラムからマルチバイト文字をアクセス使用とすると「ORA-03120」エラーを受け取るのはなぜですか。

A. CLOB/NCLOB カラムの CLOB の長さを取得するときに OCI レイヤから実際の長さより大きい値が返されると、超過した文字をアクセスすることにより ORA-03120 エラーが発生します。この問題は、Oracle 8.1.6.3 を使用することによって解消されます。

Q. PreparedStatement クラスを実行すると「TRUNC fails: ORA-00932: inconsistent datatypes」エラーが発生するのはなぜですか。

A. Oracle Metalink Bug Database Doc ID: 144784.1 によれば、暗黙的なデータ型キャストの不存在により、OCI はバインド変数が CHAR データ型であると見なします。SQL 文がバインド変数を DATE データ型として使用し、一方 OCI はそれが CHAR であると考えた場合、SQL パーサにはデータ型の矛盾が発生します。これを解決するには、データ変換関数を明示的に使用して、問題の生じたクエリに含まれるバインド変数を変換します。たとえば、次の選択文字列があるとします。

```
String st = "select count(*) from simple_table where
TRUNC(mydate) = TRUNC(?)";
```

これは、次のように変更する必要があります。

```
String st = "select count(*) from simple_table where
TRUNC(mydate) = TRUNC(TO_DATE(?))";
```

Q. ResultSet、Statement、Connection の各オブジェクトをすべて閉じても、「ORA-01000: 最大オープン カーソル数を超えました」というエラーが発生するのはなぜですか。

A. これは Oracle の問題です。Oracle のマニュアルによれば、動的カーソルは、セッション内の異なる実行間で開いたままになり、プロシージャが閉じてもカーソルを閉じることはできません。この問題を回避するには、データベースで使用できるオープン カーソルの数を増やすか、または接続プールをリセットします (接続プールのデータベース接続を閉じて開き直す)。

接続プールをリセットするには、Administration Console を使って、接続プールを対象から除外した後、再び対象として設定します。JMX API から `reset()` メソッドを使用して、または WebLogic Server コマンドライン インタフェースの `RESET_POOL` コマンドを使用して、同じ操作を行うこともできます。詳細については、『WebLogic JDBC プログラミング ガイド』を参照してください。

14 FAQ: JMS

WebLogic JMS 6.1 製品 BEA

- WebLogic JMS がユニークである理由は何ですか。
- WLS 6.1 の新しい JMS 機能は何ですか。
- WLS 6.0 サービスパック 1 で修正された問題、およびまだ存在している既知の問題は何ですか。
- WLS 5.X JMS クライアントは WLS 6.X と対話できますか。また、その反対も可能ですか。
- WLS JMS への C/C++ インタフェースは存在しますか。
- クライアントをサポートするための weblogic.jar の小型軽量バージョンはありますか。
- JMS の詳細情報はどこで参照できますか。

コンフィグレーション

- WLS を起動して JMS をコンフィグレーションする方法を教えてください。
- JMS のコンフィグレーションはどのように行うのですか。
- WebLogic リリース 5.1 でサポートされていたデフォルト接続ファクトリはまだ使用できますか。
- `JMSSession.createTopic` または `JMSSession.createQueue` が WLS JMS 6.1 で送り先の作成に失敗するのはなぜですか（5.1 では正常に作成されます）。
- キューまたはトピックのリストをプログラマティックに取得するにはどのようにすればよいですか。
- 一時的な送り先はどのように使用するのですか。
- MBean を使用して実行時統計を印刷する方法を教えてください。
- 2 つの JMS サーバで同じ永続ストレージを共有できますか。

永続ストレージ

- WebLogic JMS ではどのタイプの JDBC データベースがサポートされているのですか。
- JMS でサードパーティの JDBC ドライバを使用する方法を教えてください。
- JDBC データベースで障害が発生した場合はどうなるのですか。
- 永続性はどのように使用するのですか。
- ファイル ストアと JDBC ストアの比較はどのように行えばよいですか。

管理

- WLS JMS 6.1 サーバ / 送り先メッセージの最大値としきい値はどのように機能するのですか。
- JDBC をコンフィグレーションして JMS JDBC ストアが自動的に回復するようにする方法を教えてください。
- WebLogic JMS はクラスタ化をサポートしていますか。
- アプリケーションがどの WebLogic Server で実行されるかを制御する方法を教えてください。
- WLS 6.1 での JMS クラスタ化の価値はなんですか。
- 手動のフェイルオーバはどのように実行するのですか。
- WLS JMS サーバは、クローズまたは失われた接続、クラッシュ、およびその他の問題を検出して、それらから回復できますか。
- WLS T3 プロトコルを使用する必要はありますか。
- HTTP トンネリングはどのようにして行いますか。
- WLS JMS は SSL2 をサポートしていますか。
- 非 WLS JMS プロバイダと WLS を統合する方法を教えてください。

トランザクション サポート

- 2 フェーズ トランザクションまたはグローバル トランザクションは、WebLogic JMS とどのように関連するのですか。

- JMS 処理がユーザ トランザクションの一部にならない（トランザクション内で呼び出されるが、適切にロール バックされない）のはなぜでしょうか。トランザクションの問題を追跡するにはどのようにすればよいでしょうか。
- アプリケーションがトランザクションの結果に関係なく JMS 処理を正常に実行する方法を教えてください。
- トランザクション内で `acknowledge()` が呼び出された場合、何が起こりますか。
- トランザクションを必要とする EJB から非トランザクションの `TopicSession` を使用するときにエラーが発生するのはなぜですか。
- 他の作業に使用しているのと同じデータベース上に JMS JDBC ストアがある場合、1 フェーズ コミットを使用できますか。
- XAResource と WLS を統合して、別のリソース マネージャで JTA トランザクションを取得する方法を教えてください。
- XA ドライバまたは TX データ ソースを使用して JMS を起動するときに例外が発生するのはなぜですか。
- WLS JMS は `XAResource` 互換ですか。
- コンテナ管理トランザクション内で送信したメッセージを受信できないのはなぜですか。
- ロールバックまたは回復されるメッセージはどのようになるのですか。

JMS プログラミングの慣習

- メッセージを保留にしておいて、後で確認応答することは可能ですか？
- ソートされたキューはどのように使用するのですか。
- メッセージ優先度に基づくソートはどのように機能するのですか。
- 送信されるメッセージに追いつかないリスナをどのように処理すればよいでしょうか。
- スレッド ダンプを取得して問題を追跡する方法を教えてください。
- クライアント識別子はユニークにする必要がありますか。
- メッセージはコピー / 値か参照のどちらによって渡されますか。

- キューを管理して特定のメッセージを参照および削除する方法を教えてください。
- キューをクローズして、サーバの次回の起動時にメッセージがリロードされないようにする方法を教えてください。
- オブジェクト メッセージの受信後にそれが null として出力されるのはなぜですか。
- メッセージはどのような順序でコンシューマに配信されるのですか。
- 接続ファクトリを見つけようとしているときに例外が送出されるのはなぜですか。
- メッセージ セレクタの使用を避ける必要があるのはなぜですか。
- メッセージ セレクタ（通常相関 ID に基づくフィルタ処理）を使用して実際にメッセージを受信するリスナを決定することによって、複数のキュー レシーバが同じキューをリスンすることは可能ですか。
- 1 つのアプリケーションがあるキューにリスナとして 1 つのオブジェクトを持っており、他のアプリケーションがそのキューのメッセージをリスンできるようにキューを作成する方法はありますか。
- `javax.jms.Message.setJMSPriority`、`DeliveryMode`、`Destination`、`TimeStamp`、または `Expiration` を使用するときに設定値が機能しないのはなぜですか。
- JMS クライアントをマルチ スレッド化する場合は、どのような注意が必要ですか。
- 複数のトピックをサブスクライブするにはアプリケーションをどのように設定すればよいですか。
- `receive()` 呼び出しのブロックおよび非同期の `receive()` 呼び出しはどのように利用するのですか。
- `receive()` 呼び出しをブロックするときに注意することは何ですか。
- `NO_ACKNOWLEDGE` 確認応答モードの目的は何ですか。
- マルチキャスト サブスクライバを使用するのはどのような場合ですか。
- サーバ セッション プールと接続コンシューマは、どのような場合に使用するのですか。

- `onMessage()` メソッド呼び出し内で `close()` メソッドを発行するにはどのようにすればよいですか、また、`close()` メソッドのセマンティクスは何ですか。
- XML メッセージをパブリッシュするにはどのようにすればよいですか。
- JMS をアプレットで使用するにはどのようにすればよいですか。
- スタートアップ クラスを使用して JMS オブジェクトを初期化し、後でそれを参照するにはどのようにすればよいですか。
- メッセージ リスナ内からのメッセージの送受信は可能ですか。
- プロデューサ プールはどのように作成するのですか。
- コンソール内の保留中のメッセージとは何ですか。
- ejb-jar.xml のメッセージ選択で「より小さい」または「より大きい」を使用する方法を教えてください。
- 一定数のサブスクライバに対するセッションを増やした方がよいですか、減らした方がよいですか。
- 外部 JMS メッセージで外部送り先は処理されますか。
- アプリケーション サーバ内でスレッドの作成や初期化の実行などを行うための標準的な方法を教えてください。
- トピック A.B と 2 番目のトピック A.B.C に名前を付けたときに JNDI の問題が発生するのはなぜですか。
- トピック メッセージを処理するためにネットワーク間で送信されるメッセージの数はどのくらいですか。
- `XPATH` セレクタとはどのようなものですか。
- JMS を使用して要求 / 応答を処理する方法を教えてください。
- メッセージをキューに戻して処理するにはどうすればよいですか。
- キューまたはトピック接続が開始されてから新しいセッションとサブスクライバをそれらに追加することはできますか。
- プロデューサがコンシューマより高速であるため `java.lang.OutOfMemoryError` を受け取った場合、何を行えばよいですか。

- さまざまな接続ファクトリがあるのはなぜですか。
- 接続とセッションはどのように割り当てればよいですか。
- アプリケーションは、アプリケーション サーバがダウンしているかどうかをどのように知るのですか。
- Visual Cafe 4.1 を使用して WebLogic Server をデバッグする方法を教えてください。
- `setMessageSelect(String s)` を使用して、`TopicConsumer` の既存のセレクタを動的に変更する方法はありますか。
- 非同期メッセージのデッドロックを回避するにはどうすればよいですか。

メッセージ駆動型 Bean

- メッセージ駆動型 Bean の利点は何ですか。
- メッセージ駆動型 Bean の同時実行性はどのように機能するのですか。
- MDB はメッセージ プロデューサ、またはプロデューサとコンシューマの両方になれますか。
- MDB が恒久サブスクリプションを使用する場合、MDB がデプロイされないときにメッセージは蓄積されますか。
- 非 WebLogic Server JMS プロバイダの送り先を使用して MDB を駆動する方法を教えてください。
- 外部 JMS プロバイダを使用してトランザクション対応 MDB を駆動できますか。
- JTA トランザクションを MDB で使用するにはどのようにすればよいですか。
- サーバ セッション プールとメッセージ駆動型 Bean を比較したいのですが。

Q. WebLogic JMS がユニークである理由は何ですか。

A. WebLogic JMS をユニークにしている特長は以下のとおりです。

- JavaSoft JMS 仕様バージョン 1.0.2 の厳密な準拠
- 確かな信頼性、スケーラビリティ、およびパフォーマンス
- アプリケーション サーバの WebLogic Server との統合

- クラスタ化のサポート（2 層または 3 層）。クライアントで送り先の位置は意識されません。
- JMS アプリケーションと他のリソース マネージャ（主にデータベース）との相互運用を実現する 2 フェーズ トランザクション。JMS アプリケーションは、Java Transaction API（JTA）を使用する他の Java API とのトランザクションに参加できます。
- ファイルベースまたは JDBC ベースによるメッセージの永続ストレージ
- メッセージ駆動型 Bean
- IP マルチキャスト アドレスを使用して、選択したホストのグループにメッセージを配信できるようにするマルチキャストのサポート
- NO_ACKNOWLEDGE 確認応答モード
- Extensible Markup Language（XML）メッセージ
- サーバまたは送り先に基づくメッセージの割り当て（オプション）
- 複数のキュー ソート オプション
- 信頼性の高いカスタマ サポート

Q. WLS 6.1 の新しい JMS 機能は何ですか。

A. WLS JMS 6.1 の新機能は以下のとおりです。

- 配信時間 ― この機能を使用すると、メッセージのプロデューサは、各メッセージに配信時間を関連付けることができます。配信時間が設定されたメッセージは、配信されるまで（配信時間まで）本質的にユーザからは見えません。このため、将来の特定の時間に作業スケジュールを設定できます。
- 再配信の遅延 ― 外部の条件によって、アプリケーションのメッセージ処理が妨げられる場合があります。この場合、アプリケーションはメッセージをロールバックまたは回復して、そのメッセージを後で再び受信できるようにします。JMS がそのメッセージを即座にアプリケーションに再配信した場合、外部条件が不明確であるため、アプリケーションは依然としてメッセージを処理できない場合があります。この機能は、ロールバックまたは回復されたメッセージの配信を遅らせることによって外部条件が明確になる時間を確保して、アプリケーションにメッセージを処理する機会を与えます。
- 再配信の制限 ― この機能を使用すると、メッセージング システムは問題のあるメッセージを処理できるようになります。ユーザが指定した回数だけ

メッセージがコンシューマに配信され、コンシューマから戻されると、そのメッセージは配信不能と見なされてリダイレクトされます。リダイレクトされたメッセージは、コンフィグレーションされた ErrorDestination に送信されるか、または単純に削除されます。

- WLS 6.0 より（場合によっては大幅に）高速化されています。
- メッセージ ページング – WLS 6.1 サービス パック 2 から利用できるようになったこの機能を使えば、メッセージ負荷が指定のしきい値に達した時点で仮想メモリから永続ストレージにメッセージをスワップ アウトすることで、メッセージ負荷がピークのときに貴重な仮想メモリを解放できます。パフォーマンスの点から見ると、今日の企業アプリケーションで必要とされる大きなメッセージ空間を使用する WebLogic Server の実装は、この機能から多大な恩恵を受けます。

  詳細については、『WebLogic Server 管理者ガイド』の「JMS の管理」を参照してください。

Q. WLS 6.0 サービス パック 1 で修正された問題、およびまだ存在している既知の問題は何ですか。

A. このトピックに関する資料については、http://developer.bea.com/docs/jmsjta.jsp を参照してください。

Q. WLS 5.X JMS クライアントは WLS 6.X と対話できますか。また、その反対も可能ですか。

A. いいえ、これは現在サポートされていません。基本的に、RMI には相互運用性がありません。また、JMS は実質的に異なるプロトコルを使用します。

Q. WLS JMS への C/C++ インタフェースは存在しますか。

A. いいえ、これはサポートされていません。

- JNI を使用して、独自のインタフェースを記述してください。
- C/C++ クライアントが JMS メッセージを作成するために呼び出すサーブレットを設定します。C++ では複数スレッドを生成し、複数ポストを使用してメッセージを http を介して投稿する必要があります。

Q. クライアントをサポートするための weblogic.jar の小型軽量バージョンはありますか。

A. WebLogic Server でシン クライアント アプリケーションを作成するための現在のオプションについては、http://dev2dev.bea.com/resourcelibrary/whitepapers.jsp?highlight=whitepapers の「WebLogic Thin Client」ホワイト ペーパー (`WebLogicThinClient.zip`) を参照してください。

weblogic.jar の独自の軽量バージョンを作成する方法については、news://newsgroups.bea.com/3ad4ad17@newsgroups.bea.com も参照してください。

Q. JMS の詳細情報はどこで参照できますか。

以下のリンクを利用すれば、JMS の詳細情報を参照できます。

『WebLogic JMS プログラマーズ ガイド』

『WebLogic Server 管理者ガイド』の「JMS の管理」

Sun Microsystems の JMS 仕様

BEA の dev2dev Web サイト

AskBEA

BEA Newsgroup サーバーで利用できる WebLogic JMS “weblogic.developer.interest.jms” ニュースグループ。

Q. WLS を起動して JMS をコンフィグレーションする方法を教えてください。

A. NT では、[スタート | プログラム | BEA Weblogic E-Business Platform | Weblogic Server 6.X | Start Default Server] を選択し、管理者パスワードを入力して WLS 6.X を起動します。

NT で JMS をコンフィグレーションするには、[スタート | プログラム | BEA Weblogic E-Business Platform | Weblogic Server 6.X | Start Default Console] を選択してコンソールを起動します。

1. 左側のコンソール ツリー ビューで、[JMS] を選択します。
2. 永続メッセージが必要な場合、最初にストアを作成します。そのためには、まず［ストア］を選択します。右側のウィンドウで、ファイル ストアの [新しい JMSFile Store のコンフィグレーション] を選択し、そのファイル ストアに名前を付け、ディレクトリを指定して ［作成］を選択します。
   JDBCStore が必要な場合、最初に JDBC 接続プールを作成する必要があります。そのためには、ツリー ビューで JDBC を選択し、[接続プール]、[新し

い JDBC Connection Pool のコンフィグレーション] を選択します。[対象] を選択し、対象サーバを選択し、右矢印を選択して、[適用] をクリックします。次に、[ストア]、[新しい JMSJDBCStore のコンフィグレーション] に戻ります。

3. テンプレートが必要な場合、最初にテンプレートを作成します。そのためには、[テンプレート] を選択します。一時キューを作成するには、テンプレートが必要です。[新しい JMS Template のコンフィグレーション] を選択し、名前を指定して、[作成] を選択します。次に、[しきい値と割当] タブまたは [オーバライド] タブに移動します。変更が済んだら、[適用] を選択します。

4. [サーバ] を選択します。[新しい JMSServer のコンフィグレーション] を選択し、その名前を指定し、ストアを選択し（作成済みの場合）、テンプレートを選択して（作成済みの場合）、最後に [作成] を選択します。次に、他のタブに移動し、変更を行って [適用] をクリックします。特に、[対象] を選択し、対象サーバを選択し、右矢印を選択して、[適用] をクリックします。これが、JMS が起動するサーバです。

5. 送り先の作成 - 左パネルのツリー ビューから、[JMS] の左の + を選択し、[サーバ] の左の + を選択し、対象サーバの左の + を選択して、[送り先] を選択します。[新しい JMSQueue のコンフィグレーション] または [新しい JMSTopic のコンフィグレーション] を選択し、最初のページに値を入力して [作成] をクリックします。次に、他のタブで値を入力して [適用] をクリックします。

6. 接続ファクトリの作成 - 左側のツリー ビューで、[JMS] を開きます。[接続ファクトリ] を選択します。右パネルで [新しい JMS Connection Factory のコンフィグレーション] を選択します。接続ファクトリ名と JNDI 名を入力します。右下隅にある [作成] を選択します。[対象] タブを選択します。接続ファクトリをデプロイするサーバを選択します。右矢印を選択すると、そのサーバが移動して選択されます。最後に、右下隅にある [適用] を選択します。

Q. JMS のコンフィグレーションはどのように行うのですか。

A. JMS のセキュリティを設定するための適切な方法は、コンソールに移動し、ツリー ビューで ACL を選択して、アクセス制御リストを作成することです。

1. ACL 名を選択します。これは、weblogic.jms.queue.QUEUENAME または weblogic.jms.topic.TOPICNAME です。

2. ［作成］を選択します。

3. 送信または受信の新しいパーミッションを入力します。

4. ［作成］を選択します。

5. ユーザまたはグループのカンマ区切りリストを入力します。

6. ［Grant Permission］を選択します。

7. [saved to the realm implementation] を選択して、変更を保存します。

8. ［はい］を選択します。

これで、fileRealm.properties ファイルの行が次のように更新されます。

```
acl.send.weblogic.jms.queue.TestQueue1=user1
acl.receive.weblogic.jms.queue.TestQueue1=user1
```

キューまたはトピックの ACL が存在しない場合、セキュリティは開放状態です。

また、JNDI コンテキストにアクセスするための ACL も存在します。JNDI コンテキストは、最初に JMS にアクセスするために不可欠です。詳細については、JNDI のドキュメントを参照してください。

Q. WebLogic リリース 5.1 でサポートされていたデフォルト接続ファクトリはまだ使用できますか。

A. はい。ただし、以下の 2 つのデフォルト接続ファクトリに対する名前は非推奨となっています。

```
javax.jms.QueueConnectionFactory
javax.jms.TopicConnectionFactory.
```

ただし、これらの接続ファクトリは、下位互換性のためにこのリリースでも定義されており、使用することができます。

WebLogic JMS 6.1 では、デフォルトで次の 1 つの接続ファクトリが用意されています。

```
weblogic.jms.ConnectionFactory
```

JMS デフォルト接続ファクトリを有効にする必要があります。コンソールの [（サーバ名）｜チューニング] を選択して、[デフォルト JMS 接続ファクトリを有効化] チェック ボックスをクリックします。

Administration Console を使用すれば、ユーザ定義の接続ファクトリも指定できます。

Q. JMSSession.createTopic または JMSSession.createQueue が WLS JMS 6.1 で送り先の作成に失敗するのはなぜですか（5.1 では正常に作成されます）。

A. WLS 5.1 では、createTopic() または createQueue() はデータベースに永続的な送り先を（存在しない場合に）作成しますが、weblogic.properties ファイルは修正しません。

JavaSoft JMS 仕様バージョン 1.0.2 によれば、createQueue() および createTopic() は送り先を動的に作成するためのものではありません。これらは、JNDI ルックアップの代わりに文字列名を使用して参照される送り先を検索するために使用されます。送り先は、最初に config.xml ファイルに存在する必要があります。この変更は、WLS 6.0 のドキュメントに記載されています。この動作が前のリリースとは異なるからです。WLS JMS ヘルパー クラス（weblogic.jms.extensions.JMSHelper）またはコンソールを使用すると、実行時に送り先を作成できます（6.0 にはバグがあったため、サーバの再起動時に問題が発生しました。これはサービス パック 1 で修正されています）。これらのメカニズムにより送り先が作成され、コンフィグレーション ファイルも修正されます。

JMSHelper クラスの詳細については、『WebLogic JMS プログラマーズ ガイド』の「送り先の動的作成」を参照してください。

次のプログラムにより、トピックが作成されます。

```
import java.io.*;
import java.util.Hashtable;
import javax.jms.*;
import javax.naming.*;
import weblogic.jms.extensions.JMSHelper;


class t {
public final static String
JNDI_FACTORY="weblogic.jndi.WLInitialContextFactory";
public final static String JMS_SERVER_NAME="TestJMSServer";
public final static String DEST_JNDI_PREFIX="javax.destination.";


static public void main(String [] args) throws Exception {
try {
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, JNDI_FACTORY);
env.put(Context.PROVIDER_URL, "t3://localhost:7001");
Context ctx = new InitialContext(env);


String topicName = "JMSHelperTestQueue01";
String topicJNDI = DEST_JNDI_PREFIX + topicName;
System.out.println("topic name=" + topicName + ", jndi=" +
topicJNDI);
JMSHelper.createPermanentTopicAsync(ctx, JMS_SERVER_NAME,
```

```
topicName,
topicJNDI);
} catch (JMSException e) {
    e.printStackTrace();
      }
   }
}
```

Q. キューまたはトピックのリストをプログラマティックに取得するにはどのようにすればよいですか。

A. 次のプログラムは Mbean を使用します。

```
import weblogic.management.*;
import weblogic.management.configuration.*;


InitialContext ic = new InitialContext();
MBeanHome home = (MBeanHome)ic.lookup(MBeanHome.ADMIN_JNDI_NAME);
for(Iterator i = o.getMBeansByType("JMSTopic").iterator();
i.hasNext(); ){
WebLogicMBean wmb = (WebLogicMBean)i.next();
System.out.println("topic name found: " + wmb.getName());
}


for(Iterator i = o.getMBeansByType("JMSQueue").iterator();
i.hasNext(); ){
WebLogicMBean wmb = (WebLogicMBean)i.next();
System.out.println("queue name found: " + wmb.getName());
}
```

Q. 一時的な送り先はどのように使用するのですか。

A. 一時的な送り先を作成するすべての JMSServer にテンプレートを作成する必要があります。TemporaryTemplate をサポートするために複数の JMSServer エントリを指定できます。これらの JMSServer の間でロード バランシングが行われ、一時的な送り先が設定されます。JMS のコンフィグレーション方法については、「WLS を起動して JMS をコンフィグレーションする方法を教えてください」を参照してください。作成されるテンプレート定義は、次のようになります。

```
<JMSTemplate  Name="MyTemplate"/>
```

JMSServer は次のように定義されます。

```
<JMSServer Name="MyJMSServer" TemporaryTemplate="MyTemplate"
Targets="MyServer" >
```

テンプレート名の後に、テンプレートに追加したい任意のキュー / トピックを入れることができます（JNDI 名とトピック マルチキャスト設定は含みません）。このテンプレートは最も外側のレベルに存在します。このため、<JMSServer> の中でネストすることはできません。

一時的な送り先は、作成する接続によってのみ消費されます。トピックを使用する場合、一時的なトピックを作成して、そのトピックにサブスクライブします。誰かにその一時的なトピックにパブリッシュしてもらうには、その人に自分のトピックを教える必要があります。その人にメッセージを送信して、一時的なトピックを JMSReplyTo フィールドに挿入できます。TemporaryTopic の作成者とサブスクライバは、同じでなければなりません。

```
import javax.jms.TopicSession;
TemporaryTopic myTopic = mySession.createTemporaryTopic();
TopicSubscriber = mySession.createSubscriber(myTopic);
```

一時的なトピックは、名前を取得せず、他の接続によってサブスクライブできません。一時的なトピックを作成する場合、JMS プロバイダは `javax.jms.Topic` を返します。そのトピックは、他の当事者（トピックにパブリッシュする人たち）に公開する必要があります。そのためには、`JMSReplyTo` フィールドにそのトピックを入れて、それらの当事者が応答できるようにします。一般に、そのトピックは他の誰もサブスクライブできません。トピックは、自分の好きなように公開できます。トピックは、シリアル化（外部化）できます。このため、ファイルを介して RMI 呼び出しで受け渡し、JNDI 名にバインドできます。つまり、サブスクライバ サイドでトピックを作成し、他人がパブリッシュできるようにそれを公開できます。同じ接続で複数のサブスクライバを取得し、複数のセッションを使用して並行処理を取得できます。

Q. MBean を使用して実行時統計を印刷する方法を教えてください。

A. ニュースグループの記事、news://newsgroups.bea.com/3B3B77A9.CDCE3954@not.my.address.com に、実行時 MBean に基づいて JMS 統計を印刷するプログラムが掲載されています。

Q. 2 つの JMS サーバで同じ永続ストレージを共有できますか。

A. できません。各 JMS サーバは、それぞれ独自の永続ストレージを使用する必要があります。ファイルベースの 2 つの JMS 永続ストレージは同じディレクトリを共有できますが、それらのメッセージは別々のファイルに格納されます。この場合、ファイル名では別々のプレフィックスが使用されます。

JDBC ベースの 2 つの JMS 永続ストレージは同じデータベースを共有できますが、データベース テーブルのプレフィックス名に別々のものを使用するようにコンフィグレーションする必要があります。JDBC プレフィックス名のコンフィグレーションの詳細については、Administration Console オンライン ヘルプの「JMS JDBC ストア」を参照してください。同じプレフィックス名でコンフィグレーションすると、永続的メッセージは壊れるか、失われてしまいます。

Q. WebLogic JMS ではどのタイプの JDBC データベースがサポートされているのですか。

A. JMS データベースには、JDBC ドライバからアクセスできる任意のデータベースを指定できます。WebLogic では、以下のデータベースがサポートされており、それぞれに対応する JDBC ドライバが用意されています。

- Cloudscape
- Informix
- Microsoft SQL （MSSQL）Server （バージョン 6.5、7）
- Oracle（バージョン 8.1.6）
- Sybase （バージョン 12)

Q. JMS でサードパーティの JDBC ドライバを使用する方法を教えてください。

使用する JDBC ドライバが、WebLogic JMS でサポートされる JDBC データベースに関する質問のドライバ リストに含まれていない場合は、JMS が必要とするテーブルを手動で作成する必要があります。

注意： WebLogic Server では、上のリストに含まれている JDBC ドライバだけがサポートされます。他の JDBC ドライバはサポートされません。

`weblogic.jar` ファイルの `weblogic\jms\ddl` ディレクトリに配置されている `.ddl` ファイルは、テンプレートとして使用できます。JDK で用意されている jar ユーティリティを使用すると、次のコマンドでそれらを `weblogic\jms\ddl` ディレクトリに抽出できます。

```
jar xf weblogic.jar weblogic\jms\ddl
```

注意： 2 番目のパラメータ（`weblogic\jms\ddl`）を省略すると、jar ファイル全体が抽出されます。

JDBC ストア用のデータベース テーブルを手動で作成するには、『WebLogic JMS プログラマーズ ガイド』の「JDBC データベース ユーティリティ」の手順に従ってください。

JDBC ストアの代わりにファイル ストアを使用する方法もあります。ファイル ストアはコンフィグレーションが簡単であり、パフォーマンスが大幅に向上することもあります。

Q. JDBC データベースで障害が発生した場合はどうなるのですか。

JDBC ストア テーブルの削除および再作成の手順、またはデータベース テーブルの手動作成の手順については、『WebLogic JMS プログラマーズ ガイド』の「JDBC データベース ユーティリティ」に説明してあります。

Q. 永続性はどのように使用するのですか。
A. 次のガイドラインを使用してください。

1. 使用する JMSServer にストアがコンフィグレーションされていることを確認します。`config.xml` ファイルの JMSServer コンフィグレーション エントリには次の形式の行が含まれている必要があります。

   ```
   Store="<YOUR-STORE-NAME>"
   ```

   ストアがコンフィグレーションされていない JMS が起動すると、ユーザがそのストアを必要としていないと見なされ、永続メッセージが自動的に非永続的メッセージにダウングレードされます（JMS 1.0.2 に対して指定されているとおり）。

2. 「Message.setJMSDeliveryMode」を使用しないようにします。こればベンダ専用のメソッドなので上書きされます。

3. 以下のいずれかを呼び出す必要があります。

   ```
   QueueSender.send(msg, deliveryMode, ...)
   ```

   または

   ```
   QueueSender.setDeliveryMode(deliveryMode)
   ```

   または

   `config.xml` ファイルで接続ファクトリに対する `DefaultDeliveryMode` モードを永続的に設定します（`QueueSender.setDeliver/send` はこの値をオーバーライドします）。同様に、トピックの場合は `TopicPublisher` を介してこれを設定します。

4. `config.xml` ファイルで送り先にタイする「DeliveryModeOverride」が非永続的に設定されていないことを確認します。

5. `pub/sub` を使用する場合、恒久サブスクリプションだけがメッセージを永続させます。非恒久サブスクリプションは、メッセージを永続させる必要がありません。これらは定義によってサーバの存続期間中にのみ存在するからです。

6. JDBC を使用する場合、JMS サーバの起動時に JDBC テーブル、JMSSTATE、および JMSSTORE が自動的に作成されます。テーブルの作成に使用される DDL ファイルは、`weblogic\jms\ddl` の `weblogic.jar` に格納されます。以下のコンフィグレーション例は、Oracle 用の JDBC ストアを示したものです（JDK 1.3 で WLS 6.1 を実行するにはクライアント バージョン 8.1.7 以上が必要です）。テーブルを手動で作成する（および既存のテーブルを削除する）には、前述の質問で説明したとおり、`java utils.Schema` を実行します。

JMS のコンフィグレーション方法については、「WLS を起動して JMS をコンフィグレーションする方法を教えてください」を参照してください。

以下に、JMS がコンフィグレーションされた `config.xml` ファイルのサンプルを示します。このサンプルは、実際に使用するファイルとほぼ同じであるはずです。JMS でデータベースの代わりにファイル ストアを使用する場合、`JMSServer` セクションの `JDBCStore` を `FileStore` に変更します。

```
<Server Name="myserver"
ListenPort="7001" DefaultProtocol="t3"
ThreadPoolSize="8" >
</Server>
<Security  Realm="defaultRealm"
GuestDisabled="false" />
<Realm Name="defaultRealm"
FileRealm="defaultFileRealm" />
<FileRealm Name="defaultFileRealm"
/>
<JMSServer Name="TestJMSServer"
TemporaryTemplate="TestTemplate1"
Targets="myserver" Store="JDBCStore">
<JMSQueue Name="TestQueue1"
JNDIName="jms.queue.TestQueue1"
Template="TestTemplate1"
/>
</JMSServer>
<JMSTemplate Name="TestTemplate1"
/>
<JMSFileStore Name="FileStore"
Directory="myfilestore"
JMSServer="TestJMSServer"
/>
<JMSJDBCStore Name="JDBCStore"
ConnectionPool="testpool2"
JMSServer="TestJMSServer"
/>
<JDBCConnectionPool Name="testpool2"
Targets="myserver"
URL="jdbc:weblogic:oracle"
DriverName="weblogic.jdbc.oci.Driver"
InitialCapacity="0"
```

```
MaxCapacity="1"
CapacityIncrement="1"
Properties="user=SCOTT;password=tiger;server=bay816"
/>
</Domain>
```

次に、構築時にトピック メッセージを送信するサンプル クラスを示します。

```
import javax.naming.*;
import javax.jms.*;
import java.util.Hashtable;

public class t

{
public final static String DESTINATION="jms.topic.TestTopic1";

private TopicConnectionFactory connectionFactory;
private TopicConnection        connection;
private TopicSession           session;
private TopicPublisher         producer;
private TextMessage            message;
private Topic                  destination;

public t()
{
try {
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.WLInitialContextFactory");
env.put(Context.PROVIDER_URL, "t3://localhost:7001");
 InitialContext ctx = new InitialContext(env);
 destination = (Topic) ctx.lookup(DESTINATION);
 connectionFactory = (TopicConnectionFactory)
 ctx.lookup("javax.jms.TopicConnectionFactory");
 connection = (TopicConnection)
connectionFactory.createTopicConnection();
 session = (TopicSession) connection.createTopicSession(false,
 Session.AUTO_ACKNOWLEDGE);
 producer = (TopicPublisher) session.createPublisher(destination);
 producer.setDeliveryMode(DeliveryMode.PERSISTENT);
 message = (TextMessage) session.createTextMessage();
 message.setText("hello world");
 producer.publish(message);
} catch (Exception e) {
}
}
}
```

Q. ファイル ストアと JDBC ストアの比較はどのように行えばよいですか。

A. 以下に、ファイル ストアと JDBC ストアの類似点と相違点を挙げます。

- どちらも同じトランザクション セマンティクスを持っています。これには、トランザクションのロール バック（受信したメッセージをキューに戻すなど）が含まれます。
- どちらも同じアプリケーション インタフェースを備えています（アプリケーション コードに違いがない）。
- ファイル ストアの方が非常に高速です。
- JDBC の方が障害の回復が簡単です。これは、JDBC インタフェースが任意のクライアント マシンからデータベースにアクセスできるからです。ファイル ストアの場合、ディスクが共有または移行されている必要があります。
- ファイル ストアの信頼性は、ディスクと O/S の信頼性に限定されます。このため、Veritas または RAID 5 システム上で実行する必要があります。データベースの信頼性はこれより高くなります。
- ファイル ストアの方がより多くのメモリを必要としますが、必ずしも大量に必要とするわけではありません。必要なメモリ量は、アプリケーションの動作が不安定な場合にファイル ストアがどのようにフラグメント化されるのかによって決まります。
- FIFO の場合、ファイル ストアはフラグメント化されません。
- ファイル ストアは、ネットワーク トラフィックをさらに生成することがありません。データベース ストアは、データベース サーバが異なる JVM またはマシン上に存在する場合はトラフィックをさらに生成します。

Q. WLS JMS 6.1 サーバ / 送り先メッセージの最大値としきい値はどのように機能するのですか。

A. バイトとメッセージの最大値はフロー制御ではなく割り当てです。メッセージの割り当てにより、WebLogic JMS サーバがメッセージで一杯になり、メモリが不足して、予期しない結果になることが防止されます。割り当てに到達すると、JMS は（ブロッキングではなく）`ResourceAllocationException` によってそれ以上の送信を防ぎます。割り当ては、個々の送り先またはサーバ全体に設定できます。

同様に、しきい値もフロー制御ではありません。ただし、しきい値は割り当てよりもそのアプリケーションに適しています。しきい値は、超過した場合にメッセージをコンソールに記録するための設定値に過ぎません。これにより、ユーザは対処に遅れたことを知ることができます。

接続ファクトリに対するメッセージの最大数の設定は割り当てではありません。これは、サーバから送信されてから非同期コンシューマが見るまでの間に存在する未処理メッセージの最大数を指定します。このデフォルト値は 10 です。

Q. JDBC をコンフィグレーションして JMS JDBC ストアが自動的に回復するようにする方法を教えてください。

A. 複数の顧客からの報告によると、JDBC ストアを使用しており、DBMS がダウンして復帰した場合、JMS は WLS をシャットダウンして再起動するまで JDBC ストアを使用しないという問題があります。この問題を回避するには、JMSJDBCStore に関連付けられている JDBC 接続プールの次の属性をコンフィグレーションします。

```
TestConnectionsOnReserve="true"\
TestTableName="[[[catalog.]schema.]prefix]JMSState"
```

これらが設定されていないと、JDBC リソースがダウンして復帰した場合、JMS は WLS をシャットダウンしてから再起動するまで接続プールを使用できません。これは、WLS 6.0 SP02 と WLS 6.1 でテスト済みです。

Q. WebLogic JMS はクラスタ化をサポートしていますか。

A. WebLogic JMS では、クラスタ内のあらゆるサーバから送り先へのクラスタワイドで透過的なアクセスがサポートされています。システム管理者は、複数の接続ファクトリをコンフィグレーションし、対象を使用してそれらを WebLogic サーバに割り当てることで、クラスタ内のあらゆるサーバから送り先へのクラスタワイドで透過的なアクセスを確立できます。各接続ファクトリは、複数の WebLogic サーバにデプロイできます。

アプリケーションでは、Java Naming and Directory Interface（JNDI）を使用して接続ファクトリをルックアップし、JMS サーバとの通信を確立するための接続を作成します。各 JMS サーバでは、複数の送り先に対する要求が処理されます。JMS サーバで処理されない送り先への要求は、適切なサーバに転送されます。

クラスタ内のさまざまなノード上の複数の JMS サーバに異なる名前が付けられている場合、それらのサーバをコンフィグレーションできます。また、さまざまな JMS サーバに送り先を割り当てることができます。

ただし、JNDI でエントリをレプリケートする際に伝播遅延が発生するという問題を認識しておく必要があります。あるノードにデプロイされる MDB が別のノード上の送り先を参照する場合、デプロイメントは失敗し、javax.naming.NamingException 例外が送出されます。この問題が発生するのは、サーバがまだリモート サーバ（JMS サーバ）からの JNDI と同期化されておらず、このため MDB のデプロイメントの一部としての送り先の JNDI ルックアッ

プが失敗するためです。この解決策のひとつは、各 MDB がローカルの送り先を参照するようにすることです。、また、サーバの起動（プラス JNDI 伝播の遅延）後に MDB をデプロイするという方法もあります。MDB のデプロイ前にメッセージを失うことを回避するには、恒久サブスクリプションを使用します。この問題は、WLS 6.1 では修正されています。WLS 6.1 では、MDB はデプロイされ、送り先が使用可能になるまで再接続が試行されます。ただし、これはローカル以外の JMS 送り先を参照する EJB については依然として問題であることに注意してください。

接続ファクトリの詳細については、『WebLogic JMS プログラマーズ ガイド』の「WebLogic JMS の基礎」を参照してください。WebLogic クラスタの詳細については、『WebLogic Server Clusters ユーザーズガイド』を参照してください。

Q. アプリケーションがどの WebLogic Server で実行されるかを制御する方法を教えてください。

A. システム管理者は、接続ファクトリのコンフィグレーション時に対象を指定することで、アプリケーションをどの WebLogic Server で実行するのかを指定できます。各接続ファクトリは、複数の WebLogic サーバにデプロイできます。

注意： デフォルトの接続ファクトリを使用する場合は、接続ファクトリがデプロイされる WebLogic サーバを指定できません。特定の WebLogic サーバを対象にする場合は、新しい接続ファクトリを作成し、適切な JMS サーバの対象を指定してください。

Q. WLS 6.1 での JMS クラスタ化の価値はなんですか。

A. コンシューマとプロデューサは、送り先が実際にどこに存在するかに関係なく、クラスタ内の任意のサーバからその送り先にアクセスできます。これにより、場所の透過性が実現されます。ただし、場所の透過性（レプリケートされた JNDI を使用する）を実現するには、クラスタのメンバのいずれか（管理サーバ以外）の JMSServer をコンフィグレーションする必要があります。

特定のプロデューサまたはコンシューマが存在するサーバがダウンした場合でも、他のサーバは送り先にアクセスできます。しかし、送り先がダウンした場合、すべてのプロデューサおよびコンシューマは役に立たなくなります。JMS サーバがダウンした場合、そのサーバが関与しているすべてのトランザクションがロール バックされます。JMS は、クラスタ内のダウンしたサーバ以外のサーバで引き続き正常に動作します。そのサーバが復帰すると、そのサーバが再び JMS サービスを提供します。サーバが復帰すると、メッセージがコンシューマに送信されるか、またはコンシューマによって取得されます。サーバを再起動すれば、正常に動作するはずです。

たとえば、トピックがサーバ B にデプロイされ、サブスクライバがサーバ A を介し、パブリッシャがサーバ B を介してそのトピックにアクセスするとします。この場合、サーバ A がダウンすると、パブリッシャはメッセージをパブリッシュできますが、サブスクライバはメッセージを受信できません（サーバ B に接続していない場合）。

将来のリリースでは、WLS JMS は、送り先が存在するサーバがダウンしたときに自動移行およびフェイルオーバを処理するようになります。

Q. 手動のフェイルオーバはどのように実行するのですか。

A. WebLogic Server の障害からの回復手順、手動フェイルオーバの実行手順、およびプログラミングの考慮事項については、『管理者ガイド』の「JMS の管理」を参照してください。

Q. WLS JMS サーバは、クローズまたは失われた接続、クラッシュ、およびその他の問題を検出して、それらから回復できますか。

A. はい、クライアントがそのリソースをクローズした、ネットワーク接続に失敗した、およびその中に JVM が存在する場合、WLS 6.1 マルチキャスト pub/sub であっても、JMS サーバは「ピア」の損失を検出して回復します。

Q. WLS T3 プロトコルを使用する必要はありますか。

A. J2EE は、インタフェースを標準化しています。WebLogic の RMI 仕様の実装では、T3 という独自の通信プロトコルが使用されます。Javasoft の RMI の参照実装では、JRMP という独自のプロトコルが使用されます。WebLogic が T3 を開発した理由は、エンタープライズクラスの分散オブジェクト システムを Java で構築するためのスケーラブルで効率的なプロトコルが必要であったためです。

T3 は WebLogic 独自のプロトコルですが、ユーザのアプリケーション コードは T3 について何も知る必要はありません。このため、これについて心配する必要はありません。「WebLogic 固有の文字列」（`PROVIDER_URL`、`INITIAL_CONTEXT_FACTORY`、など）をプロパティ ファイル（またはどこか）に外部化します。これにより、コードを完全に移植可能にして、プロパティ ファイル内のそれらのみを変更するだけで、コードを取得して別の J2EE アプリケーション サーバで実行できるようになります。

Q. HTTP トンネリングはどのようにして行いますか。

A. HTTP トンネリング（すべてのメッセージを HTTP に包み込んでファイアウォールを通過させること）を使用する場合、`TunnelingEnabled="true"` を

config.xml ファイルの <Server> 定義に追加するか、コンソール上の該当するボックスをオンにする必要があります。次に、InitialContext を取得するときに、Context.PROVIDER_URL 用に t3://localhost:7001 ではなく http://localhost:7001 というような URL を使用します。SSL を使用して HTTP トンネリングを行う場合、https://localhost:7002 を使用します（https は HTTP トンネリングと SSL を使用します。7002 はコンフィグレーションしたセキュアなポートです）。これを行うと、パフォーマンスが低下します。このため、必要な場合にのみ（ファイアウォールを通過する必要がある場合など）トンネリングを使用してください。

Q. WLS JMS は SSL2 をサポートしていますか。

A. はい、SSL は WLS JMS でサポートされています。SSL2 は、初期 JNDI コンテキストをルックアップするときに、「t3:」の代わりに「t3s:」で始まる URL を使用することによって、自動的に使用されます。

Q. 非 WLS JMS プロバイダと WLS を統合する方法を教えてください。

A. MQ Series、IBus MessageServer、Fiorano、および SonicMQ との統合については、ftp://edownload:BUY_ME@ftpna2.bea.com/pub/downloads/jmsproviders.doc を参照してください。

Q. 2 フェーズ トランザクションまたはグローバル トランザクションは、WebLogic JMS とどのように関連するのですか。

A. 2 フェーズ トランザクションまたはグローバル トランザクションを使用すると、複数のリソース マネージャ（EJB、データベース、JMS サーバなど）が 1 つのトランザクションに参加できます。

たとえばクライアントは、2 フェーズ トランザクションを使用して、1 つの JMS サーバ（サーバ A）のキューから別の JMS サーバ（サーバ B）のキューにメッセージを送信できます。各サーバには、固有の永続ストレージがあります。トランザクションがコミットされると、メッセージがサーバ B で表示可能になります。トランザクションがロールバックされた場合、メッセージはサーバ A のキューに戻されます。

注意： 両方のキューが同じ JMS サーバにある場合は、1 フェーズ トランザクションを使用します。

Q. JMS 処理がユーザ トランザクションの一部にならない（トランザクション内で呼び出されるが、適切にロール バックされない）のはなぜでしょうか。トランザクションの問題を追跡するにはどのようにすればよいでしょうか。

A. 通常、この問題は、設計により外部のグローバルなトランザクションを無視するトランザクション セッションを明示的に使用することによって発生します。

トランザクション JMS セッションは、常に独自の内部トランザクションを持ちます。これは、呼び出し側が持っているトランザクション コンテキストによって影響されません。

また、この問題は、「UserTransactionsEnabled」が false に設定されている接続ファクトリを使用することによっても発生します。

1. 以下の 2 つのインポート行を追加して、現在のスレッドがトランザクションに存在するかどうかをチェックできます。

   ```
   import javax.transaction.*
   import weblogic.transaction.*;
   ```

   また、以下の行を追加します（各処理の直前および直後など）。

   ```
   Transaction tran = TxHelper.getTransaction();
   System.out.println(tran);
   System.out.println(TxHelper.status2String(tran.getStatus()));
   ```

   これにより、いつ新しいトランザクションが開始し、関連付けが発生するのかを明確に理解できます。

2. JMS メッセージを送信するスレッドがトランザクションに関連付けられるようにします。createQueueSession または createTopicSession の最初のパラメータを false に設定することによって、コードがトランザクション セッションを使用していないことをチェックします。接続およびセッションの作成は、トランザクションと直交しています。トランザクションは、その前または後に開始できます。メッセージを送信または受信する前に、トランザクションを開始するだけで済みます。

3. config.xml ファイルで接続ファクトリの UserTransactionsEnabled フラグが明示的に true に設定されていることを確認してください。この値のユーザ コンフィグレーション接続ファクトリのデフォルト値は false であるからです。コンフィグレーション済みの接続ファクトリの 1 つを使用する場合、次のように設定します。

   `weblogic.jms.ConnectionFactory` は、ユーザ トランザクションを無効化します。このため、ユーザ トランザクションが必要な場合には使用しないでください。

   `javax.jms.QueueConnectionFactory` および `javax.jms.TopicConnectionFactory` はユーザ トランザクションを有効化します。

4. 次の追加プロパティを使用してサーバを起動することによって、JTA 処理を追跡できます。

   ```
   -Dweblogic.Debug.DebugJMSXA=true
   ```

以下のようなトレース文がログに表示されます。

```
XA ! XA(3163720,487900) <RM-isTransactional() ret=true>
```

これを使用すると、JMS をトランザクションに関連付けることができます。

Q. アプリケーションがトランザクションの結果に関係なく JMS 処理を正常に実行する方法を教えてください。

A. 基本的に、JMS 処理がトランザクション セッションを使用して行われるか、またはトランザクションが次のようにサスペンド / 無効化されなければなりません（以下の 1 つまたは複数を行います）。

1. JMS 呼び出しを行う前に現在のトランザクションをサスペンドし、その完了後に再開します。コードは次のようになります。

```
import javax.transaction.Transaction;
import javax.transaction.TransactionManager;

TransactionManager tranManager=
TxHelper.getTransactionManager();
Transaction saveTx = null;
try {
saveTx = tranManager.suspend();
... JMS 処理を行う。トランザクションには参加しない
} finally {
// 中断されたトランザクションは常に再開する必要がある
if (saveTx != null) tranManager.resume(saveTx);
}
```

2. createQueueSession または createTopicSession の最初のパラメータに true を指定することによって、トランザクション セッションを使用します。

3. 接続ファクトリを使用し、ユーザ トランザクションを無効にします。つまり、config.xml ファイルで接続ファクトリの UserTransactionsEnabled フラグが明示的に false に設定されていることを確認するか、この値のユーザ コンフィグレーション接続ファクトリのデフォルト値の false を使用します。コンフィグレーション済みの接続ファクトリ `weblogic.jms.ConnectionFactory` はユーザ トランザクションを無効にします。

トランザクション JMS セッションは、常に独自の内部トランザクションを持ちます。これは、呼び出し側が持っているトランザクション コンテキストによって影響されません。非トランザクション JMS セッションはより複雑です。WLS 6.1 のデフォルト ファクトリ `weblogic.jms.ConnectionFactory` を使用する場合、セッションはユーザ トランザクションに参加しません。これは、

UserTransactionsEnabled フラグが「False」に設定されているからです。非推奨のデフォルト ファクトリの javax.jms.QueueConnectionFactory または javax.jms.TopicConnectionFactory を使用する場合、または独自のファクトリを定義して UserTransactionsEnabled フラグを「True」に設定した場合、JMS セッションは外部のトランザクションに参加します（外部トランザクションが存在し、JMS セッションがトランザクション セッションではない場合）。

Q. トランザクション内で acknowledge() が呼び出された場合、何が起こりますか。

A. JMS 仕様により、トランザクションの中では acknowledgeMode は無視されます。このため、トランザクションの中で acknowledge() メソッドが呼び出された場合、それは無視されます。

Q. トランザクションを必要とする EJB から非トランザクションの TopicSession を使用するときにエラーが発生するのはなぜですか。

A. あるトランザクションに 2 つのリソース（JMS やデータベースなど）が参加するときには、そのトランザクションは 2 フェーズとなります。使用しているデータベース ドライバが XA 互換ではないため、通常 2 フェーズ トランザクションには参加できません。これを解決するには、XA 互換ドライバを使用するか、または JDBCTxDataSource 値をコンフィグレーションして enableTwoPhaseCommit を true に設定します。後者の場合、ヒューリスティック エラーが発生する場合があるので注意してください。JMS を現在のトランザクションに参加させたくない場合、「アプリケーションがトランザクションの結果に関係なく JMS 処理を正常に実行する方法を教えてください」を参照してください。

Q. 他の作業に使用しているのと同じデータベース上に JMS JDBC ストアがある場合、1 フェーズ コミットを使用できますか。

A. 使用できません。WLS 6.1 は、JMS は自身のリソース マネージャです。つまり、JMS 自体が XAResource を実装し、メッセージがデータベースに格納されている場合でも、データベースに依存せずにトランザクションを処理します。このため、JMS とデータベースを使用するときには、そのデータベースが JMS メッセージが格納されているデータベースと同じである場合でも、常に 2 フェーズ コミットになります。

JMS キューと同じサーバ上にデータベース作業に使用する接続プールが存在していれば、パフォーマンスが向上します。トランザクションは依然として 2 フェーズですが、より小さいオーバーヘッドで処理されるからです。また、JMS JDBC ストアではなく JMS ファイル ストアを使用することによっても、パフォーマンスが向上する場合があります。

Q. XAResource と WLS を統合して、別のリソース マネージャで JTA トランザクションを取得する方法を教えてください。

A. MQ Series の統合については、http://developer.bea.com/docs/jmsjta.jsp を参照してください。

Q. XA ドライバまたは TX データ ソースを使用して JMS を起動するときに例外が発生するのはなぜですか。

A. JMS で TX データ ソースを使用することはできません。JMS は、非 XA リソース ドライバを使用する JDBC 接続プールを使用する必要があります（XA ドライバまたは JTS ドライバは使用できません）。`enableTwoPhaseCommit` オプションは設定しないでください。JMS は JDBC ドライバの上で XA をサポートします。

Q. WLS JMS は `XAResource` 互換ですか。

A. はい。バージョン 6.0 では、WLS JMS は JTA 仕様に定義されているとおりに `XAResource` インタフェースを実装していました。バージョン 6.1 では、`XAConnection`、`XAConnectionFactory`、`XAQueueConnection`、`XAQueueConnectionFactory`、`XAQueueSession`、`XASession`、`XATopicConnection`、`XATopicConnectionFactory`、および `XATopicSession` の各メソッドも実装されています。これらは、JMS 仕様ではオプションとして定義されています。これらのメソッドは、`XAResource` インタフェースの一部ではありません。`XAResource` インタフェースは、トランザクション マネージャが JTA トランザクションを管理するために必要な開始、準備、コミットのようなメソッドを持ち、これらは WLS JMS によって提供されます。JMS の送信または受信を JTA トランザクション（開始 / コミット）でラップした場合、JMS の処理はトランザクションの一部となります（トランザクション セッションを使用しない場合）。

このため、`XAQueueConnectionFactory` を使用して `XAQueueConnection` および `XAQueueSession` を取得して `getXAResource()` を呼び出すことはできません。ただし、WLS JMS は WLS JTA によって自動的に登録されるため、これを心配する必要はありません。

また、WLS JMS はトランザクションを駆動する別のトランザクション マネージャ（少なくともドキュメント化されているインタフェース）では使用できません。

Q. コンテナ管理トランザクション内で送信したメッセージを受信できないのはなぜですか。

A. コンテナ管理トランザクションを使用している場合、EJB から送信された元のメッセージは送信されません。以下に、何が起こるかを示します。

1. コンテナがトランザクションを起動します。
2. メソッドを起動します。
3. 新しいメッセージを生成します。
4. メッセージを送信します（実際には送信されず、トランザクションのコミットまでバッファに格納されます）。
5. キュー上でブロック受信を行います。
6. メソッドを終了します。
7. 元のメッセージが送信されなかったため、トランザクションのコミットは行われません。これは、過去のブロック受信を取得できないからです。

これを解決するには、Bean 管理のトランザクションを使用するか、または送信および受信をを 2 つの独立したメソッドに分割します。

Q. ロールバックまたは回復されるメッセージはどのようになるのですか。

A. メッセージがロールバックまたは回復されると、そのメッセージが要求されます。キューおよびそのキューのその他のメッセージのソート順序に基づいてメッセージがキューに入れられます。

順序が設定されていないキュー（一般的）では、他のメッセージが同時に回復されない限り、通常メッセージはキューのヘッダに置かれます（キューが到着時間（FIFO）でソートとされているため）。ただし、ソート順序は競合を解決します。

メッセージは可能な限り最初に利用可能なコンシューマに再配信されます。要求時にコンシューマが存在しない場合、コンシューマが現れるまでそのままそこに置かれます。

WLS 6.1 から、メッセージの最大要求回数を設定できるようになりました。また、要求後にメッセージが使用できなくなるまでの遅延時間も指定できます。メッセージが最大要求回数に達した場合、エラー送り先に置かれるか（コンフィグレーションされている場合）、または通知することなく削除されます。

Q. メッセージを保留にしておいて、後で確認応答することは可能ですか？

A. これを行うためのプリミティブは存在しません。以下に、可能な解決策を 2 つ示します。

1 つは、次のように複数のセッションを使用することです。

```
while (true) {
  Create a session, subscribe to one message on durable
```

```
subscription
  Close session
  Save session reference in memory
  To acknowledge the message, find the session reference and call
  acknowledge() on it.
}
```

もう 1 つは、トランザクションを使用して、次のように作業をサスペンドすることです。

```
start transaction
while(true) {
  message = receive();
  if (message is one that I can handle)
  process the message
  commit
} else {
  suspend transaction
  put transaction aside with message
  start transaction
 }
}
```

メッセージを「確認応答」するには、次のようにします。

```
resume user transaction
commit
```

メッセージを「回復」するには、次のようにします。

```
resume user transaction
rollback
```

サスペンドするたびに、トランザクションをメッセージとともにスタックまたはリストに置いて、後で処理またはロールバックできるようにする必要があります。この解決策は、未処理トランザクションが大量に蓄積する可能性があるため、高いオーバーヘッドとなります。トランザクションはタイムアウトを持っており、自身でロールバックする場合があるため、メッセージを（異なるトランザクションで）再び取得する可能性があることに注意してください。. また、未処理にしておくトランザクションの数には実際的な制限があることにも注意してください。デフォルトの制限は約 10000 です。最終的には、スタック / リストに戻ってトランザクションをコミット / ロールバックします。トランザクション参照（`javax.transaction.Transaction`）はシリアル化できないことに注意してください。

Q. ソートされたキューはどのように使用するのですか。

A. 送り先キーは、特定の送り先に対してソート順を定義する場合に使用されます。送り先キーとしては、メッセージ ヘッダまたはプロパティ フィールドを使用できます。有効なメッセージ ヘッダおよびプロパティ フィールドのリストについては、『WebLogic JMS プログラマーズ ガイド』を参照してください。

キューは、送り先キーを基準に昇順または降順でソートできます。送り先は、送り先キーが JMSMessageID メッセージ ヘッダ フィールドの昇順として定義されている場合は先入れ先出しと判断され、降順として定義されている場合は後入れ先出しと判断されます。JMSMessageID ヘッダ フィールドに定義されるキーは、キー リストで定義されている最後のキーでなければなりません。

送り先をソートするために、複数の送り先キーを定義できます。

送り先キーを作成するには、Administration Console の [送り先キー] ノードを使用します。詳細については、Administration Console オンライン ヘルプを参照してください。

Q. メッセージ優先度に基づくソートはどのように機能するのですか。

A. 第 1 に、送り先にキーを追加する必要があります（デフォルトではそれらはソートされていません）。そのためには、`JMSPriority` をキーとして選択します。最も高い優先度を 0 にする場合は、キーを昇順にします。最も高い優先度を 9 にする場合は、キーを降順にします。

第 2 に、優先度は（メッセージではなく）プロデューサまたはセンダ側で設定される必要があります。

第 3 に、優先度のソートは、待ち状態のメッセージがキューに複数存在する場合に機能します。レシーバが常にセンダに追いついていれば、メッセージは到着した順序で処理されます。

Q. 送信されるメッセージに追いつかないリスナをどのように処理すればよいでしょうか。

A. 以下に、いくつかのガイドラインを挙げます。

- 複数のリスナの使用を検討してください。
- リスナでの処理の削減を検討してください（「System.out」を呼び出さないなど）。
- 送信は、一般に受信より高速です。すべてのメッセージを確認応答せず、クライアント肯定応答を使用していくつかのメッセージを受信するまで確認応答を延期することによって、受信のオーバーヘッドを低減することを検討してください。非同期リスナの場合、確認応答の「境界」に入らない場合、追

加のコード ロジックがなければ、最後のいくつかのメッセージは肯定応答されない場合があります（10 のうち 9 を受信するが、10 番目は受信しません）。

Q. スレッド ダンプを取得して問題を追跡する方法を教えてください。

A. スレッド ダンプを取得する方法は次のとおりです。

- コマンド ラインから次のコマンドを実行してみます（\bea\wlserver6.1\config\mydomain\ で setEnv スクリプトの実行後）。

  ```
  java weblogic.Admin -url t3://localhost:7001 THREAD_DUMP
  ```

- Windows では、コンソール ウィンドウで、〔Ctrl〕+〔Break〕を入力します。
- Unix では、`kill -3` を使用してサーバに通知します。

Q. クライアント識別子はユニークにする必要がありますか。

A. 恒久サブスクライバを使用すると、2 つの異なるクライアントからサブスクライブし、同じ接続ファクトリを使用し、その接続ファクトリがコンフィグレーションされた `clientID` を持つ場合、`TopicConnection` を作成するたびにユニークな `clientID` を設定する必要があります。各接続は、ユニークな `clientID` を必要とします。`clientID` を使用して接続ファクトリをコンフィグレーションした場合、その接続ファクトリの `TopicConnection` は、その接続がクローズされるまで 1 つしか作成できません。

Q. メッセージはコピー / 値か参照のどちらによって渡されますか。

A. メッセージは、アプリケーションから見た場合、値によって渡されます（コピーが作成されます）。内部的には、メッセージ渡しを最適化するためにあらゆる努力が払われます。

プロデューサが JMS サーバと同じ JVM 上に存在する場合、メッセージのコピーが作成されてから、そのメッセージが実際に送り先に置かれます。コンシューマが JMS サーバと同じ JVM 上に存在する場合、メッセージのコピーが作成されてから、そのメッセージがコンシューマに渡されます。これにより、実際に送り先に置かれるメッセージが保存されます。参照渡しは、JMS 仕様の違反です。特に、ある人間がメッセージを作成し、そのメッセージを自分の空間で修正した場合、それは既に送信されたメッセージに影響を与えてはなりません。したがって、送信されるメッセージはコピーでなければなりません。これは受信側にも当てはまります。誰かがメッセージを消費し、プロパティの削除、ヘッダ フィールドの設定、何らかの変更などを行った場合、その変更は他のコンシューマが受信するメッセージに影響を与えてはなりません。したがって、同じ JVM 上のレ

シーバに渡されるメッセージもコピーです（レシーバごとに 1 つ）。また、レシーバに参照を渡し、そのレシーバがメッセージを修正してロールバックまたは回復を行った場合、実際に受信する人間にとって、送り先にあるメッセージは変更されることになります。このため、送信されたメッセージと異なるメッセージを受信したように見えてしまいます。これは、このメッセージがそれを受信した誰かによって修正され、元の場所に戻されたからです。

Q. キューを管理して特定のメッセージを参照および削除する方法を教えてください。

A. QueueBrowser を使用するプログラムを記述します。次に、以下の例のとおり、セレクタとメッセージ識別子を使用して、特定のメッセージを削除します。

```
String selector = "JMSMessageID = '" + message.getMessageID() +
"'";
```

キュー ブラウザは、キューの実際のビューではないことに注意してください。これはスナップショットです。

Q. キューをクローズして、サーバの次回の起動時にメッセージがリロードされないようにする方法を教えてください。

A. JMS は、キューの削除を定義しません。WLS JMS を使用すると、キューの作成と削除を管理できます。実行時にキューを動的に削除する方法は存在しません。

必要な場合、メッセージを確認応答するか、永続性を使用しないようにできます。また、一時的なキューを使用することもできます。これらは、接続の存続期間中にのみ存在します。さらに、恒久サブスクリプションを使用することもできます。恒久サブスクライバのメッセージは、その恒久サブスクライバがサブスクライブしないにもかかわらず、トピックがそのまま存在する場合に削除されます。

Q. オブジェクト メッセージの受信後にそれが null として出力されるのはなぜですか。

A. オブジェクトは、`ObjectMessage.getObject()` が呼び出されるまでデシリアライズされません。`toString()` は、これが起こるまで NULL を出力します。アプリケーションが `setObject()` を呼び出していない場合、WebLogic Server 6.1 はメッセージを自動的にデシリアライズします。

Q. メッセージはどのような順序でコンシューマに配信されるのですか。

A. プロデューサとコンシューマ間の順序は、配信モードと同じようにソート順序で管理され、ロールバックまたは回復が存在しない場合はセレクタによって管理されます。複数のプロデューサが単独のコンシューマに送信するか、または複

数のコンシューマが複数のプロデューサから受信する場合、順序の保証はまったくありません。

順序は、一般にプロデューサとコンシューマの間で管理されます。ただし、非永続メッセージは、ソート順序がより高い（優先度がより高い、など）永続メッセージより優先し、他より前に移動できます。また、回復またはロールバックは受信済みのメッセージをキュー / トピックに戻します。これにより、順序が影響を受けます。

大部分のメッセージング システム（WLS JMS を含む）は、プロデューサと送り先間、および送り先とコンシューマ間の順序を管理します。このため、いったんメッセージが送り先に到着したら、順序は変更されません。

最後に、WLS JMS でサポートされる非同期パイプラインも順序に影響を与えます。デフォルトでは、サーバから非同期クライアントに送信される未処理メッセージが最大 10 通も存在する可能性があります。非同期コンシューマが「捕まえられた」場合、これらのメッセージはソートされません。パイプラインでは送り先のソートは行われません。送り先が優先度でソートされ、到着する新しいメッセージの優先度がパイプラインに既に存在するメッセージより高い場合、新規メッセージはパイプライン内を飛び越えることはなく、送り先の最初に置かれます。パイプラインのサイズはコンフィグレーション可能です。使用する接続ファクトリの MessagesMaximum 設定を参照してください。真の優先度ソートを行いたい場合、接続ファクトリの最大メッセージ数を 1 に変更してください。

Q. 接続ファクトリを見つけようとしているときに例外が送出されるのはなぜですか。

A. 例外は、一般に `java.io.InvalidClassException` または `java.lang.NoClassDefFoundError` のようなものです。

クライアントの CLASSPATH に `weblogic.jar` が存在することを確認してください。また、適切な Java 実行時 jar ファイルが含まれていることを確認してください（たとえば `rt.jar` が必要な場合もあります）。

Q. メッセージ セレクタの使用を避ける必要があるのはなぜですか。

A. BEA は、セレクタを頻繁に使用するアーキテクチャを避けることをお勧めしています。セレクタを使用する必要がある場合、トピック セレクタを使用するようにしてください。これらは、1 サブスクライバ、1 メッセージ当たり 1 度の負担しか負いません。メッセージがサブスクライバの選択条件に一致しない場合、そのメッセージは無視され、サブスクライバの「サブスクリプション」には置かれません。

キューの場合、セレクタはより重くなります。各受信要求では、レシーバの選択条件に一致するメッセージの検索でキュー全体のスキャンが必要となる場合があります。選択条件が厳しく、キューの深さが大きい場合、これは非常に重い負担になる可能性があります。

トピックの動作は、一般にキューより優れています。ただし、これはメッセージの混在とキューの深さに依存します。キューを使用する場合、検索しているメッセージに一致しないメッセージが数多く存在する場合、自分のメッセージの 1 つを検索するたびに、多数のメッセージとセレクタを何度も繰り返し比較しなければなりません。トピックを使用する場合は、各メッセージとすべてのコンシューマが 1 度だけ比較されます。このため、重複する作業は存在しません。

また、非同期キュー（受信呼び出しのポスト）を使用する場合、アクセスするたびにメッセージを検索する必要があります。つまり、サーバが比較を行っている間は待機する必要があります。トピックを使用する場合、メッセージはあらかじめ比較され、本質的に専用のキューに置かれます。そのポイントを参照する必要はありません。それは独自の専用キューからの FIFO となります。サーバは、順序を処理する間に比較を行うことができます。これに対し、アプリケーションはサーバがメッセージを検索している間はブロックされます。

Q. メッセージ セレクタ（通常相関 ID に基づくフィルタ処理）を使用して実際にメッセージを受信するリスナを決定することによって、複数のキュー レシーバが同じキューをリスンすることは可能ですか。

A. JMS 仕様では、こうした動作は定義されていません。WebLogic JMS 6.1 は、これをサポートしています。メッセージがキューに置かれると、JMS は、そのキューのすべてのコンシューマを、それらが受信を行った順番に検索します。その結果、そのメッセージに一致する最初のコンシューマがそのメッセージを受信します。非同期キュー コンシューマの場合は、リスナを最初に設定しておく必要があり、これによってコンシューマ リストの先頭に置かれます。ただし、非同期キュー コンシューマがメッセージを受信するたびに、コンシューマはリストの最後に移動します。トピックとは異なり、メッセージがセレクタに一致しない場合、誰かがセレクタを持たないか、または一致するセレクタを持つまでそのメッセージはキューに残されます。

Q. 1 つのアプリケーションがあるキューにリスナとして 1 つのオブジェクトを持っており、他のアプリケーションがそのキューのメッセージをリスンできるようにキューを作成する方法はありますか。

A. いいえ。その代わりとして、1 つの恒久サブスクリプションを持つトピックを作成できます。恒久サブスクリプションには、1 つのコンシューマだけを関連付けることができるからです。この欠点は、セレクタがキューのときのようには機能しなくなることです。JMS 1.0.2 仕様によると、恒久サブスクリプションに対するセレクタを変更すると、サブスクリプションが「リセット」されます。こ

の結果、そのサブスクリプションに現在存在するすべてのメッセージが削除されます。

Q. `javax.jms.Message.setJMSPriority`、`DeliveryMode`、`Destination`、`TimeStamp`、または `Expiration` を使用するときに設定値が機能しないのはなぜですか。

A. これらのメソッドは、ベンダ専用です。メッセージの値は、送信 / パブリッシュのたびに上書きされます。これらの値を設定するには、`MessageProducer`、`QueueSender`、または `TopicPublisher` の対応メソッド（`setJMSPriority`、`setDeliveryMode`、`setTimeToLive` など）を使用する必要があります。これらの値がオプションのテンプレート コンフィグレーションのオーバーライド値によってオーバーライドされていないかどうかをチェックしてください。

Q. JMS クライアントをマルチ スレッド化する場合は、どのような注意が必要ですか。

A. JavaSoft JMS 仕様バージョン 1.0.2 では、JMS セッションがシングル スレッドであることが示されています。したがって、複数のスレッドがセッションあるいはコンシューマまたはプロデューサのどれかに同時にアクセスする場合、動作は保証されません。さらに、複数の非同期コンシューマがセッションに存在する場合、メッセージは並列的ではなく連続的にそれらに配信されます。

JMS で複数のスレッドを利用するには、複数のセッションを使用します。たとえば、並列な同期受信要求を可能にするには、セッションごとに 1 つのコンシューマだけがアクティブになるようにアプリケーションを設計し、複数のセッションを使用します。

Q. 複数のトピックをサブスクライブするにはアプリケーションをどのように設定すればよいですか。

A. N 個のトピックをリスンする場合、N 個のサブスクライバと N 個のセッションを使用すると N 個の同時実行スレッドまで同時実行性が提供されます（それだけ多くのスレッドを使用する場合）。N 個のサブスクライバと 1 個のセッションでは、そのセッションを介してすべてのサブスクリプションがシリアライズされます。負荷が重い場合、追加のスレッドがなければ追いつくことができません。また、`CLIENT_ACKNOWLEDGE` を使用する場合、N 個のセッションによって、個々に回復可能な N 個の独立したメッセージ ストリームが与えられます。1 個のセッションだけがそれらのストリームを横切ると、制御が低下します。WebLogic JMS の旧バージョンとは異なり、サーバサイドのバージョン 6.1 は、存在するクライアント セッションの数に関係なく、小さい固定数のスレッドを有効に使用します。

Q. `receive()` 呼び出しのブロックおよび非同期の `receive()` 呼び出しはどのように利用するのですか。

A. 同期 `receive()` メソッドは、メッセージが生成されるまで、タイムアウト値に達するまで（指定されている場合）、またはアプリケーションが閉じるまでブロックされます。サーバサイドでの `receive()` 呼び出しのブロックはできる限り行わないでください。同期 `receive()` 呼び出しは、その呼び出しがブロックされている全期間にわたってリソースを消費するからです。

『WebLogic JMS プログラマーズ ガイド』の「クライアント サーブレットを使用したメッセージの受信」で説明されているように、クライアント サーブレットを使用してメッセージを受信する場合はサーバのデッドロックが発生する可能性があります。

メソッドが非同期で受信される場合、メッセージが生成されたときにのみメッセージ リスナを使用してアプリケーションに通知されます。そのため、メッセージを待つことでリソースが消費されることがありません。

Q. `receive()` 呼び出しをブロックするときに注意することは何ですか。

A. アプリケーションの設計上、メッセージを同期的に受信する必要がある場合、以下のメソッドのいずれかを使用することをお勧めします（望ましい順に挙げてあります）。

- `receive()` に対する引数としてタイムアウト値を渡し、0 より大きい最小値を設定します。これは、サーバからの応答を待つスレッドの浪費を避けるためにアプリケーションが許可する値です。
- `receiveNoWait()` メソッドを使用します。このメソッドは次のメッセージを返すか、現在取得できるメッセージが存在しない場合は NULL を返します。この場合、呼び出しはブロックされません。サーブレットは、`wait()` を呼び出さずに、リクエストに戻るか、リクエストを再スケジューリングする方法を提供します。

注意： ビジー状態のサーバでデッドロックを引き起こすことがあるため、このオプションの使用は最小限にします。

- 同時にブロックされる `receive()` 呼び出しの数よりも多くのスレッドをコンフィグレーションするようにしてください。

Q. `NO_ACKNOWLEDGE` 確認応答モードの目的は何ですか。

A. `NO_ACKNOWLEDGE` 確認応答モードは、受信したメッセージで確認応答の必要がないことを示します。これでパフォーマンスは向上しますが、メッセージが失われる恐れがあります。このモードは、セッションの確認応答で提供されるサービスの質を必要とせず、それに関連するオーバーヘッドを避ける必要があるアプリケーションで使用します。

NO_ACKNOWLEDGE セッションに送信されたメッセージは、サーバから即座に削除されます。このモードで受信されたメッセージは回復されないので、メッセージを配信する最初の試行が失敗した場合はメッセージが失われたり、重複メッセージが配信されたりします。

注意： アプリケーションで、失われたメッセージや重複メッセージを処理できない場合は、このモードは使用しないでください。重複メッセージは、メッセージを配信する最初の試行が失敗した場合に送信されます。

また、この確認応答モードは永続的なメッセージングでは使用しないでください。永続的なメッセージングにとっては、サービスの質が低すぎる可能性があります。

Q. マルチキャスト サブスクライバを使用するのはどのような場合ですか。

A. マルチキャストを使用することによって、後でメッセージをマルチキャスト サブスクライバに転送する、指定したホストのグループにメッセージを配信できます。マルチキャストには、次のような利点があります。

- ホスト グループにメッセージをほとんどリアルタイムに配信できます。
- メッセージをマルチキャスト サブスクライバに配信する場合に JMS サーバで必要になるリソース量が削減されるので、スケーラビリティが向上します。

注意： マルチキャストは、Pub/sub メッセージング モデルのみでサポートされています。

マルチキャストを使用すると便利な例としては、株価表示があります。最新の株価を入手する場合に重要になるのは、信頼性よりもタイムリーな配信です。全部または一部の内容が配信されなくても、リアルタイムの株価情報にアクセスするときに、クライアントは簡単に情報の再送信を要求できます。クライアントでは情報の回復は必要とされません。回復された情報が再配信される頃には、その情報は古くて価値のないものになっているからです。

マルチキャストでは、ホスト グループの全メンバに対するメッセージの配信は保証されません。確実な配信と回復が必要なメッセージについては、マルチキャストを使用しないでください。

Q. サーバ セッション プールと接続コンシューマは、どのような場合に使用するのですか。

A. WebLogic JMS には、サーバ セッションのサーバ管理プールを定義するためのオプションの JMS 機能が実装されています。この機能を使用すると、アプリ

ケーションで複数のメッセージを並行して処理できます。ConnectionConsumer オブジェクトでは、サーバ セッションを使用して受信メッセージを処理します。メッセージ トラフィックが大きい場合は、スレッド コンテキストの切り替えを最小限に抑えるために、接続コンシューマでは複数のメッセージで各サーバ セッションをロードすることができます。複数の接続コンシューマで、サーバ セッション プールのサーバ セッションを共有できます。

アプリケーションで接続コンシューマを使用する方法については、『WebLogic JMS プログラマーズ ガイド』の「メッセージの並行処理」、または javax.jms.ConnectionConsumer javadoc を参照してください。

注意： サーバ セッション プールはメッセージ駆動型 Bean を使用して実装することもできます。サーバ セッション プールを使用する場合、MDB を使用することをお勧めします。「サーバ セッション プールとメッセージ駆動型 Bean を比較したいのですが」を参照してください。メッセージ駆動型 Bean によるサーバ セッション プールの実装については、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』を参照してください。

Q. onMessage() メソッド呼び出し内で close() メソッドを発行するにはどのようにすればよいですか、また、close() メソッドのセマンティクスは何ですか。

A. onMessage() メソッド呼び出し内で close() メソッドを発行する場合、システム管理者は接続ファクトリをコンフィグレーションするときに [メッセージの短縮を許可] チェック ボックスを選択しなければなりません。詳細については、Administration Console オンライン ヘルプの「JMS 接続ファクトリ」を参照してください。このチェックボックスが選択されていない状態で、onMessage() メソッド呼び出し内で close() メソッドを発行すると、その呼び出しはハングします。

close() メソッドは、次の手順を実行して系統的にシャットダウンを行います。

- 保留中のすべてのメッセージの受信を終了させます。アプリケーションは、クローズ時にメッセージを受信できない場合はメッセージまたは NULL を返す場合があります。
- 現在メッセージを処理しているすべてのメッセージ リスナが完了するまで待機します。ただし、close() メソッドを呼び出すメッセージ リスナは除きます。
- 処理中のトランザクションを、そのトランザクション セッション上でロールバックします（こうしたトランザクションが外部 JTA ユーザ トランザクションの一部である場合を除きます）。

- クライアントが確認応答を行うセッションの確認応答は強制しません。確認応答を強制しないことにより、キューおよび信頼性の高い処理が要求される恒久サブスクリプション用のメッセージが失われなくなります。

接続をクローズすると、関連付けられているすべてのオブジェクトがクローズされます。受信メッセージの `acknowledge()` メソッドを除いて、接続で作成または受信されたメッセージ オブジェクトは引き続き使用できます。閉じた接続をクローズしても影響ははありません。

注意： クローズされた接続のセッションから受信したメッセージを確認応答しようとすると、`IllegalStateException` が送出されます。

セッションをクローズすると、関連付けられているすべてのプロデューサとコンシューマもクローズされます。

各オブジェクトについての `close()` メソッドの影響については、適切な `javax.jms` javadoc を参照してください。

Q. XML メッセージをパブリッシュするにはどのようにすればよいですか。

A. 以下の手順を実行します。

1. DOM ドキュメント ツリーから XML を生成します。
2. 生成された DOM ドキュメントを `StringWriter` にシリアライズします。
3. `StringWriter` の `toString` を呼び出し、それを `message.setText` に渡します。
4. メッセージをパブリッシュします。

Q. JMS をアプレットで使用するにはどのようにすればよいですか。

A. このトピックについては、news://newsgroups.bea.com/3ad321d5@newsgroups.bea.com を参照してください。

Q. スタートアップ クラスを使用して JMS オブジェクトを初期化し、後でそれを参照するにはどのようにすればよいですか。

A. このトピックについては、news://newsgroups.bea.com/3ad0d7f3@newsgroups.bea.com を参照してください。サンプル コードでは、シャットダウン時に正しくクリーンアップが実行されません。以下のような処理を行うシャットダウン クラスを使用できます。

```
JMSobject WLSobject = null;
try {
 WLSobject = JMSStartUp.getJMSobject();
 WLSobject.JMSCleanup();
} catch(Exception e) {}
```

サーブレットは、初期化とクリーンアップの両方を行うための優れたソリューションを提供します。「アプリケーション サーバ内でスレッドの作成や初期化の実行などを行うための標準的な方法を教えてください」を参照してください。

Q. メッセージ リスナ内からのメッセージの送受信は可能ですか。

A. はい。メッセージ リスナ内から任意のキューまたはトピックに対して送受信を行うことができます。

MDB ではない場合、`onMessage()` を含む同じ Connection または Session を使用してこれを行うことができます。メッセージ リスナを作成する場合、コンストラクタにセッションを渡します。次に、`onMessage` メソッドにアクセスし、`onMessage` メソッド内から非同期ではなく同期呼び出しを行うことができます。別の `onMessage()` にサービスを提供する Session を使用しないでください。Session および Sessions はマルチスレッドをサポートしないからです。

トランザクション非対応の処理を行った場合、メッセージの重複または紛失が起こる場合があります（`onMessage()` コードがメッセージを転送しようとした場合）。

1. publish() の後に確認応答を呼び出し、確認応答が何らかの理由（ネットワーク / サーバの障害）で失敗した場合、メッセージが再び表示され、パブリッシュが 2 度行われます（セマンティクスが重複する可能性があります）。連続番号を追跡して重複を削除することもできますが、簡単ではありません。
2. `publish()` の前に確認応答を呼び出した場合、セマンティクスを 1 度だけ取得します。`publish()` が失敗すると、メッセージがサーバに到着する前か後のどちらで障害が発生したのかが分かりません。

`onMessage` を使用してトランザクション セマンティクスが 1 度だけ必要な場合は、トランザクション MDB を使用する必要があります。トランザクション MDB の `onMessage()` は、トランザクションを起動し、そのトランザクション内で受信した WebLogic Server JMS メッセージを組み込みます。`publish()` も同じトランザクションに入ります。次のコードは、受信した各メッセージへの応答を送信します。このコードは、`ejbCreate` メソッドで接続などを作成し、`onMessage` が呼び出されるたびに接続を作成する必要がないようにします。`QueueSender` は匿名（ヌル Queue）です。これは、誰に返信する必要があるの

か分からないからです。ejbRemove メソッドは、接続をクローズすることによってクリーンアップされます。これと同じアプローチを使用して、レシーバ、サブスクライバ、またはパブリッシャを作成できます。

```
import javax.ejb.CreateException;
import javax.ejb.EJBContext;
import javax.naming.*;
import javax.naming.directory.*;
import java.util.Hashtable;
import javax.ejb.MessageDrivenBean;
import javax.ejb.MessageDrivenContext;
import javax.jms.*;

public class MDB
implements MessageDrivenBean, MessageListener {
public static final String WLSqcf =
"javax.jms.QueueConnectionFactory";
public static final String WLSqname =
"jms.queue.TestQueue1";
public static final String WLSurl =
"t3://localhost:7001";
public static final String WLSJNDIfactory =
"weblogic.jndi.WLInitialContextFactory";
private MessageDrivenContext context;
private QueueSession session;
private QueueConnection connection = null;
private QueueConnectionFactory factory;
private InitialContext ctx;
private QueueSender QueueSender;

// 必須 - 引数のないパブリック コンストラクタ
public MDB() {}

// 必須 - ejbActivate
public void ejbActivate() {}
  // 必須 - ejbRemove
  public void ejbRemove() {
  context = null;
  if (connection != null) {
    try {
      connection.close();
    } catch(Exception e) {}
    connection = null;
  }
}

// 必須 - ejbPassivate
public void ejbPassivate() {}

public void setMessageDrivenContext(
 MessageDrivenContext mycontext) {
 context = mycontext;
}

// 必須 - 引数のない ejbCreate()
public void ejbCreate () throws CreateException {
```

```
    try {
  // 初期コンテキストを取得
  Hashtable env = new Hashtable();
  env.put(Context.INITIAL_CONTEXT_FACTORY, WLSJNDIfactory);
  env.put(Context.PROVIDER_URL, WLSurl);
  env.put(Context.REFERRAL, "throw");
  ctx = new InitialContext(env);

    factory = (QueueConnectionFactory)ctx.lookup(WLSqcf);

  // QueueConnection、QueueSession、QueueSender を作成
  connection = factory.createQueueConnection();
  session = connection.createQueueSession(false,
    Session.AUTO_ACKNOWLEDGE);
  queueSender = session.createSender(null);
  connection.start();
  } catch (Exception e) {
  throw(new CreateException(e.toString()));
 }
}

// MessageListener の実装
// 例外の送出なし
public void onMessage(Message msg) {
   try {
   System.out.println("MDB: " +
   ((TextMessage)msg).getText());
   msg.clearBody();
   ((TextMessage)msg).setText("reply message");
   queueSender.send((Queue)msg.getJMSReplyTo(), msg);
  }
  catch(Exception e) { // すべての例外を取得
   e.printStackTrace();
  }
 }
}
```

このアプローチにより、EJB/MDB インスタンスごとに接続が作成されます。このため、EJB インスタンスによって共有されるプロデューサ プールを作成できます。これを行うには、静的プールにプロデューサを挿入するクラスを記述します（サンプル プロデューサ プールについては次の質問を参照）。onMessage 呼び出しは、必要なときにプロデューサを取得します。Sessions は単一スレッドである必要があるので、プロデューサ プール内のセッションごとに 1 つのプロデューサしか存在しません。

Q. プロデューサ プールはどのように作成するのですか。

A. 次に、プロデューサ クラスの疑似コードを示します。

```
class ProducerPool {
  static Hashmap pSets = new Hashtable();
  static Hashmap inUse = new Hashtable();
```

```
QueueSender get(String contextURL,
 String connectionFactoryName,
 String destinationName) {
 String lookup = contextURL+";"+connectionFactName+";"+destName;
 synchronized(pSets) {
  producer set = pSets.get(lookup);
  if (set != null && set not empty)
   qs = set.removeFirst();
 }
 if (producer == null) {
   create ctx
   get connect factory
   create connection
   create session
   look up destination
   qs = create queue sender
 }
 synchronized(inUse) {
  inUse.put(qs, lookup);
 }
 return qs;
}

void put(QueueSender qs) {
  String lookup;
  synchronized(inUse) {
   lookup = inUse.remove(p);
  }
  synchronzied(pSets) {
   producer set = pSets.get(lookup);
   if (set == null) {
    producer set = new producer set
    pSets.put(lookup, producer set);
   }
   producer set.add(qs);
  }
 }
}
```

注意： 静的クラスへの参照が存在しない場合、それらはガベージ コレクションによって削除されます。このため、アプリケーション サーバが何らかの方法でそれらのクラスへの永続的なポインタを持っていることを確認してください。その方法の 1 つは、起動時に初期化されるときにサーブレットまたは EJB 内から永続的にそのクラスを参照することです。

以下は、onMessage メソッド内でプロデューサ プールを使用する例です。

```
onMessage() {
  QueueSender qs = ProducerPool.get(...);
  qs.send(...);
  ProducerPool.put(qs);
}
```

スタートアップ クラスまたは起動時にロードされるサーブレット クラスから呼び出すことによって、このプールにあらかじめプロデューサを入れておくことができます。

Q. コンソール内の保留中のメッセージとは何ですか。

A. 保留中とは、メッセージが以下の状態にあることです。

- トランザクション内で送信されたが、コミットされていない。
- 受信されたが確認応答されていない。
- 受信されたがコミットされていない。
- 再配信遅延の対象となっている（WebLogic Server 6.1 より）。
- 配信時間に制約されている（WebLogic Server 6.1 より）。

ロール バックされたメッセージは、トランザクションが実際にロール バックするまで保留状態に置かれます。ロール バックを何度も行っても、重複カウントは発生せず、例外も発生しません。トランザクションが rollbackOnly として設定され、続いて実際のロールバックが発生します。

Current は、保留中でないメッセージを表します。

Total は、サーバが最後に起動したときからの合計を表します。バイト数は、メッセージのペイロードのみを考慮します。これには、プロパティと本文が含まれますが、ヘッダは含まれません。

Q. ejb-jar.xml のメッセージ選択で「より小さい」または「より大きい」を使用する方法を教えてください。

A. セレクタを CDATA セクションで囲みます。これにより、XML パーサは「より小さい」または「より大きい」をタグとして認識しなくなります。

```
<jms-message-selector>
<![CDATA[ JMSXAppID <> 'user' ]]>
</jms-message-selector>
```

Q. 一定数のサブスクライバに対するセッションを増やした方がよいですか、減らした方がよいですか。

A. N 個のサブスクライバに対して N 個のセッションを使用すると N 個の同時実行スレッドまで同時実行性が提供されます（それだけ多くのスレッドを使用する場合）。各 Session は、十分なスレッドを使用できる限り独自のスレッドを取得

します。それ以外の場合、セッションは使用可能なスレッドを逐次的に再利用します。

N 個のサブスクライバに対する 1 個のセッションでは、その 1 個のセッションを介してすべてのサブスクリプションがシリアライズされます。負荷が重い場合、追加のスレッドがなければ追いつくことができません。

`CLIENT_ACKNOWLEDGE` を使用する場合、N 個のセッションによって、個々に回復可能な N 個の独立したメッセージ ストリームが与えられます。1 個のセッションだけがそれらのストリームを横切ると、制御が低下します。

Q. 外部 JMS メッセージで外部送り先は処理されますか。

A. WebLogic Server JMS は、外部の送り先を処理する方法を知りません。この問題については JavaSoft と協議してきましたが、JavaSoft 仕様での送り先の定義は、ベンダがそのレベルで相互運用できるほど明確ではありません。JavaSoft 側は、受信 / 送信が正常に機能するように外部送り先を処理するには、この仕様は十分でないことを認めています。WebLogic Server JMS の場合、外部送り先を使用して `setJMSdestination` を実行した場合（これを設定するのはプロバイダだけなので、実際には行わないでください）、それは無視されます（NULL に設定される）。同様に、外部送り先用に `setJMSReplyTo` を実行した場合、WebLogic Server JMS はそれを無視します（NULL に設定します）。

Q. アプリケーション サーバ内でスレッドの作成や初期化の実行などを行うための標準的な方法を教えてください。

A. 通常、スレッドを直接作成しないでください。正常に機能しない場合があります。ユーザが作成したスレッドは、WebLogic Server が独自の実行スレッド、関連付けられるトランザクション コンテキスト、または環境（適切なクラスローダなど）の作成時にあらかじめ設定するスレッドローカル変数の一部を備えていません。WebLogic 独自の方法でこれを行うには、スタートアップ クラスまたは WebLogic Time サービスを使用します。これを行うための移植可能な方法は、起動時にロードされるサーブレットを定義し、`init()` メソッドで初期化を行い、`destroy()` メソッドでクリーンアップを行うことです。サーブレット自体は何も行いません。このアプローチを使用すると、サーバを再起動せずにアンデプロイ / 再デプロイを行うことができます（毎回の適切なクリーンアップ / 初期化を含む）。また、サーバを起動せずに依存クラスをより動的に管理できるようになります。

Q. トピック A.B と 2 番目のトピック A.B.C に名前を付けたときに JNDI の問題が発生するのはなぜですか。

A. これは JNDI の実装の問題です。JNDI はドット（.）を使用してディレクトリに似た構造を構築します。ある要素がノードとツリー内のリーフを兼ねること

はできません。この例では、B は A のリーフとして使用されますが、次にリーフ C のノードとして使用されます。

Q. トピック メッセージを処理するためにネットワーク間で送信されるメッセージの数はどのくらいですか。

A. たとえば、あるメッセージのサブスクライバが 3 つ存在し、そのうちの 1 つが一致しないセレクタを持っている場合、どのくらいのメッセージが送信されるでしょうか。

WebLogic JMS 6.1 では、3 つのコンシューマがすべて同じセッションに参加している場合、フロー制御されないすべてのサブスクライバに対してメッセージの 1 つのコピーがネットワーク間で送信されます。多数のメッセージを確認応答しないようコンシューマがフロー制御されると、そのフロー制御が緩和されるまでメッセージは送信されません。したがって、答えは通常は 1 ですが、2 の場合もあります。この選択はサーバ サイドで行われるので、一致しないサブスクライバは何も破棄する必要がありません。

Q. XPATH セレクタとはどのようなものですか。

A. 次に、XPATH セレクタの例を示します。二重引用符と単一引用符の使い方に注意してください。

```
String selector =
"JMS_BEA_SELECT('xpath', '/recipient/transport/text()') =
'email'";
tsubscriber = tsession.createSubscriber(topic, selector, false);
```

JMS_BEA_SELECT は、WebLogic Server JMS SQL 構文の組み込み関数です。この関数は、コンシューマの作成時にセレクタ文字列に組み込みます。xpath、XML タブ、および文字列値が単一引用符で囲まれていることに注意してください。

Q. JMS を使用して要求 / 応答を処理する方法を教えてください。

A. JMS で要求 / 応答を処理する方法はいくつかあります。

- 各要求者に対して一時キューを使用し、応答がそのキューに戻るようにします。
- QueueRequestor クラスを使用します。このクラスは、次のとおり自動的に一時キューを行い、応答を待ちます。

  ```
  // 応答用の一時キューを作成する
  qrequestor = new QueueRequestor(qsession, queue);
  ```

```
TextMessage msg = qsession.createTextMessage();
TextMessage reply = (TextMessage) qrequestor.request(msg);
```

- メッセージ セレクタとともに専用の応答トピックまたはキューを使用します。

Q. メッセージをキューに戻して処理するにはどうすればよいですか。

A. 以下のように、いくつかの方法があります。

- トランザクション セッションを使用し、そのセッションをロールバックしてメッセージをキューに戻します。
- セッションの作成時に `Session.CLIENT_ACKNOWLEDGE` を使用し、そのセッションを回復させてメッセージをキューに戻します。
- JTA トランザクションを使用し、そのトランザクションをロールバックしてメッセージをキューに戻します。

Q. キューまたはトピック接続が開始されてから新しいセッションとサブスクライバをそれらに追加することはできますか。

A. はい。ただし、1 つ注意が必要です。セッションがアクティブな非同期コンシューマを持つ場合、そのセッションに新しいサブスクライバ / コンシューマを追加できません。JMS 仕様により、セッションは単一スレッドでのみアクセスされる必要があります。これを行う必要がある場合、新しい Session を作成し、そのセッションにそれを追加します。

開始された接続にレシーバを追加できます。レシーバ自体は非同期ではありません。非同期にするには、リスナが必要となります。最初のレシーバの作成は、常に安全です。その最初のレシーバに対してリスナを追加する場合、同じセッションに参加する将来のレシーバについて心配する必要があります。新しいセッション、およびそのセッションの最初のレシーバは、何の心配もなく作成できます。

セッションに 2 番目のレシーバを作成する場合、最初のレシーバが `MessageListener` を持っているときには、そのセッションに他の実行スレッドが存在しないことを確認する必要があります。そのためには、接続を停止するか、または最初のレシーバの `onMessage` ルーチンから 2 番目のレシーバを実際に作成します。

Q. プロデューサがコンシューマより高速であるため `java.lang.OutOfMemoryError` を受け取った場合、何を行えばよいですか。

A. 割り当てを使用すると、この状況を解決できます。送信者はセンダは `ResourceAllocationExceptions` を受け取り、サーバは引き続き正常に動作します。WLS 6.X は、メッセージのメモリからのページングをサポートしていません。

WLS 6.1 SP02 以降では、メッセージ ページング機能を使用できます。この機能を使えば、メッセージ負荷が指定のしきい値に達した時点で仮想メモリから永続ストレージにメッセージをスワップ アウトすることで、メッセージ負荷がピークのときに貴重な仮想メモリを解放できます。詳細については、『WebLogic Server 管理者ガイド』の「JMS の管理」を参照してください。

Q. さまざまな接続ファクトリがあるのはなぜですか。

A. 複数の異なる接続属性セットを取得するためです。異なる動作が必要なクライアントは、異なるファクトリを使用する必要があります。すべてのクライアントが同じ動作を必要とする場合、1 つのファクトリで十分です。

Q. 接続とセッションはどのように割り当てればよいですか。

A. 接続は、単一の物理的な接続（TCP/IP リンク）であると考えることができます。セッションは、順序付けられた一連のメッセージを作成および消費するための手段です。接続の作成は、一般に高くつきます。セッションの作成は、それほど高くありません。一般に、1 つの接続を使用し、すべてのスレッドを共有します。各スレッドは、独自のセッションを持ちます。複数のスレッド グループがあり、特定のスレッド グループのリソースを起動 / 停止 / クローズする必要がある場合は、グループごとに 1 つの接続を使用するのが適切です。グループは、1 つのスレッドを持つことができます。

Q. アプリケーションは、アプリケーション サーバがダウンしているかどうかをどのように知るのですか。

A. 登録できるリスナが 2 種類存在します。JMS 仕様では、接続に問題があるかどうかを通知する `Connection.setExceptionListener` が定義されています。接続に問題がある場合、その接続を使用するすべてのコンシューマにも問題が発生します。接続例外を取得する理由は、接続する相手側の WebLogic サーバがダウンしているか、無応答であるか、または誰かが Mbean インタフェースを介して接続を切断したためです。しかし、WebLogic Server JMS の場合、1 つの接続に複数のセッションが存在し、それらが複数のバックエンド サーバにアクセスする場合があります。このため WebLogic Server には、セッションに問題があるかどうかを通知する `WLSession.setExceptionListener` という拡張が用意されています。詳細については、
http://e-docs.bea.com/wls/docs61/javadocs/weblogic/jms/extensions/WLSession.html
を参照してください。

Q. Visual Cafe 4.1 を使用して WebLogic Server をデバッグする方法を教えてください。

A. VisualCafe Enterprise Edition 4.1 をインストールして、サーバに接続します。3.X と同じように機能します。

次に、VC 4.1 を使用してデバッグを行う手順を示します。必要に応じてディレクトリ名を変更してください。

1. d:\VisualCafeEE にインストールします。特別なオプションは必要ありません。
2. c:\Program Files\Common Files\WebGain Shared にライセンスをインストールします。
3. ［スタート｜プログラム｜ WebGain Studio Professional ｜ Visual Cafe Enterprise Edition 4.1 ｜ Distributed Debugging Services ｜ Start DD Services (Java2 - 1.3)］を選択して、ddservices を起動します。
4. `java.exe` の代わりに `debugvm.exe` を使用して WebLogic Server を起動します。
   cd D:\bea\wlserver6.1\config\mydomain
   setEnv
   startWebLogic.cmd
   change "%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m to
   \visualcafeEE\jdk13\bin\debugvm.exe
5. startWebLogic を実行します。デバッグ情報が出力されます。
6. ［スタート｜プログラム｜ WebGain Studio Professional ｜ Visual Cafe Enterprise Edition 4.1 ｜ Visual Cafe Enterprise Edition 4.1］を選択して、VisualCafe を実行します。
7. ［ファイル］メニューで、［プロセスへのアタッチ］を選択します。すべてが正常に機能する場合、マシン名が表示されます。
8. + 記号をクリックしてツリーを展開し、実行中の WebLogic Server を選択します。

Q. `setMessageSelect(String s)` を使用して、`TopicConsumer` の既存のセレクタを動的に変更する方法はありますか。

A. いいえ。いったんコンシューマをインスタンス化したら、セレクタはコンシューマが作成される時点で固定されます。セレクタを変更することは、現在の

コンシューマを削除し、関連付けられているすべてのメッセージを削除して新しいコンシューマを作成することとほぼ同じです。

Q. 非同期メッセージのデッドロックを回避するにはどうすればよいですか。

A. JMS 1.0.2 仕様の制限により、セッションの `close()` メソッドがユーザ同期ブロックの内部に存在する場合、非同期メッセージがデッドロックされる場合があります。これを解決するには、`close()` メソッドをユーザ同期ブロックの外部に移動する必要があります。次に例を示します。

```
public class CloseTest() {
private void xxx() {
synchronized (this) {
create connection/session/consumer
initialize and set a listener for this consumer;
wait();
connection.close();
}
}

private void onMessage(Message message) {
synchronized (this) {
notify();
}
}
}
```

`connection.close()` メソッドがクローズされる前に、`JMSProvider` によって別のメッセージが `onMessage` ルーチンに配信される場合があります。`main()` メソッド スレッドは、`CloseTest` メソッドのモニタ ロックを保有します。`CloseTest` クラスの `onMessage()` メソッドが実行される前に、JMS は INLISTENER を `JMSSession` のセッションのステートとして設定します（JMS 仕様では、`close()` メソッドは `onMessage` ルーチンを待つ必要があります）。このため、`main()` メソッド スレッドは `onMessage` ルーチンが完了するのを待つことができます。

`onMessage` ルーチンがモニタ ロックを取得しようとすると、このルーチンはブロックして `main()` メソッド スレッドがあきらめるのを待ち、`main()` メソッド スレッドは `onMessage` が完了するのを待ちます。

また、コンシューマの `close()` メソッドが `onMessage` ルーチンから実行され、`config.xml` ファイルの `allowCloseInOnMessage` 属性が false に設定されると、JMS もブロックします。

Q. メッセージ駆動型 Bean の利点は何ですか。

A. メッセージ駆動型 Bean は、JMS のキューやトピックからメッセージを受信した結果として EJB コンテナによって呼び出されるステートレスなコンポーネントです。呼び出されたメッセージ駆動型 Bean は、メッセージの内容に基づいてビジネス ロジックを実行するので、JMS のコンフィグレーションや再接続といった面倒な作業を行う必要がなくなります。

メッセージ駆動型 Bean モデルを使用すると、EJB 開発者は慣れ親しんだフレームワークやツールを利用できると同時に、コンテナの提供する追加サポートへのアクセスを提供することもできます。メッセージ駆動型 Bean モデルの目的は、JMS メッセージを処理するために非同期で呼び出される EJB の開発を、他の JMS `MessageListener` で同じ機能を開発することと同じくらい容易にすることです。

標準の JMS `MessageListener` の代わりにメッセージ駆動型 Bean を使用する主な利点の 1 つは、JTA トランザクションを自動的に開始することができ、受信メッセージがそのトランザクションの一部になることです。この場合、他の処理が同じ JTA トランザクション（データベース処理など）に参加できます。これは、非同期コンシューマからのメッセージと別の JTA 処理を同じトランザクションに参加させる唯一の方法です。

メッセージ駆動型 Bean の詳細については、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』の「メッセージ駆動型 Bean の使い方」を参照してください。

Q. メッセージ駆動型 Bean の同時実行性はどのように機能するのですか。

A. Queue の同時実行性は、プール内の MDB インスタンスごとに 1 つの JMSSession を生成することによって実現されます。`JMSSessions` は JMS によって並列に処理されるので、同時実行性が自然に取得され、JMS はメッセージを 1 つのリスナに配信します。MDB がクラスタ内の複数のサーバにデプロイされる場合、`JMSSessions` は各サーバの MDB インスタンスごとに作成され、それらの間でロード バランシングが行われます。

WebLogic JMS 6.1 の Topics の場合、プール内の Bean インスタンスごとに 1 つの `JMSSession` が存在します。Topics の機能により、セッション、そしてすべての Bean インスタンスはその Topic でパブリッシュされた各メッセージのコピーを受信します。（並行処理の異常を引き起こす問題もありましたが、WLS 6.0 サービス パック 1 で修正されました。）単独のサーバでは、各メッセージのコピーを 1 つだけ作成し、1 つのトピック コンシューマを使用してメッセージを複数のスレッドに渡すことによって同時実行性を取得します。複数の MDB をコンフィグレーションして同じトピックでリスンでき、各 MDB はすべての各メッセージのコピーを受信します。複数のサーバを使用する場合、各サーバは独自の

コンシューマを取得し、したがって各メッセージの 1 つのコピーを取得します。複数のサーバ間でコンシューマを共有することは現時点ではできません。メッセージを 1 つの MDB によって処理するには、キューを使用してください。

ある顧客の例では、同じトピックでリスンしている MDB の複数の実装でトピック MDB が必要になりました。この場合、異なる実装を持つ複数の MDB が同じトピックをサブスクライブする場合があります。クライアントは、同じトピックでリスンしている MDB の種類がいくつあるのかを調べる効率的な方法を持っていません。しかし、キューではなく複数のリスナ、したがってトピックが存在することは可能です。トピックでリスンしている各 MDB に対して、メッセージは正確に 1 度だけ配信されます（たとえば、メッセージはトピックをリスンしている名前付きの MDP プール内のインスタンスに 1 度だけ配信される）。

Q. MDB はメッセージ プロデューサ、またはプロデューサとコンシューマの両方になれますか。

A. はい。MDB の内部に JMS コンテキストは存在しないので、接続、セッション、およびプロデューサを自分で確立する必要があります。そのための 1 つ目のオプションは、MDB の `onMessage` ルーチンに入るたびにこれを行うことです。メッセージ レートが相対的に低い場合、これで十分です。2 つ目のオプションは、`ejbActivate()` に必要なオブジェクトを確立することです。これらのオブジェクトはシリアライズ可能ではないので、ステートフル セッション Bean またはエンティティ Bean に対してパッシベーションを行うことができません。EJB が非アクティブ化した場合、関連付けられているオブジェクトをクローズする必要があります。3 番目のオプションは、スタートアップ クラスに JMS 接続 / セッション プールを構築し、独自の同期化とブロッキングを使用して完成させ、接続を取得することです。この質問の回答例については、「メッセージ リスナ内からのメッセージの送受信は可能ですか」を参照してください。

Q. MDB が恒久サブスクリプションを使用する場合、MDB がデプロイされないときにメッセージは蓄積されますか。

A. その時点では蓄積されません。MDB が最初にデプロイされる前は、MDB のデプロイ時にメッセージは蓄積されません。これはパッシベーションとは異なります。MDB が現在アクティブ化されていないということは、それがデプロイされないということを意味しないからです。コンテナは依然としてサブスクリプションを管理し、メッセージを MDB に送信します。

Q. 非 WebLogic Server JMS プロバイダの送り先を使用して MDB を駆動する方法を教えてください。

A. このトピックに関する資料については、http://developer.bea.com/docs/jmsmdb.jsp を参照してください。

Q. 外部 JMS プロバイダを使用してトランザクション対応 MDB を駆動できますか。

A. いいえ。メッセージはトランザクションの外部で非同期に受信され、メッセージをトランザクションに関連付けるための J2EE API は存在しません。

WebLogic Server JMS でこれが機能する唯一の理由は、メッセージをトランザクションに関連付けるためのメソッドを備えた WebLogic Server 拡張インタフェースが定義されているためです。このインタフェース、`MDBTransaction` は、news://newsgroups.bea.com/3b3a009b$1@newsgroups.bea.com に定義されています。これは `associateTransaction()` というメソッドを持ち、このメソッドは `javax.jms.Message` というパラメータを取ります。このメッセージはトランザクションに関連付ける必要があります。BEA は、WebLogic Server との統合に関心のある他のベンダがこのインタフェースを実装することを期待しています。

もう 1 つのアプローチは、ソース管理トランザクションと呼ばれているものです。このアプローチでは、メッセージを非同期コンシューマに配信する前にユーザに代わってトランザクションを開始するよう JMS プロバイダに通知するための API が存在するはずでした。この API は、J2EE にも存在しません。このような規定が存在したにもかかわらず、このようなトランザクションを独自に開始および駆動できる非 WLS JMS プロバイダはほとんど存在しません。

現在のソリューションは、すべてのメッセージを外部送り先から WebLogic Server JMS 送り先（外部 JMS プロバイダがこれをサポートする場合はトランザクション内の）に移動し、その WebLogic Server JMS 送り先にトランザクション対応 MDB を駆動させることです（WLS JMS 特別インタフェースを使用する）。現在のところ、http://developer.bea.com/docs/jmsproviders.jsp で説明されているのとほぼ同じコードを使用して、メッセージをプロバイダ間で移動できます。このコードは、スレッドを開始し、以下のことを行うスタートアップ クラスに含めることができます。

```
while (true) {
トランザクションを開始
タイムアウトに同期してメッセージを受信
if timed_out { ロールバックして継続 }
処理を実行
トランザクションをコミット }
```

同期受信を行うと、問題が発生して、メッセージの受信前にトランザクションがタイム アウトになる場合があります。指定されたタイムアウトを使用して受信を行うことができます。これにより、トランザクションが処理を完了するための時間を十分に確保できます。受信が失敗した場合、トランザクションをロール

バックして再試行します。また、これは MDB ほど効率的ではありません。非同期受信ではなく同期受信を行う（スレッドを拘束する）からです。また、タイムアウトが増加すると、CPU の負担が増加します。

実際のところ、WLS JMS はこのメッセージ渡しを処理するブリッジを提供します。

これらのアプローチのどれを使用する場合でも、`XAResource` を Transaction Manager に登録する必要があります（これは WebLogic Server JMS では自動的に行われる）。

Q. JTA トランザクションを MDB で使用するにはどのようにすればよいですか。

A. 次の例のとおり、`ejb-jar.xml` ファイルで、トランザクション タイプを Container、trans-attribute を Required として定義します。

```
<ejb-jar>
 <enterprise-beans>
   <message-driven>
     <ejb-name>MDB</ejb-name>
     <ejb-class>MDB</ejb-class>
     <transaction-type>Container</transaction-type>
     <message-driven-destination>
        <destination-type>javax.jms.Queue</destination-type>
     </message-driven-destination>
     </message-driven>
   </enterprise-beans>
 <assembly-descriptor>
  <container-transaction>
    <method>
      <ejb-name>MDB</ejb-name>
      <method-name>*</method-name>
      </method>
      <trans-attribute>
      Required
     </trans-attribute>
    </container-transaction>
   </assembly-descriptor>
  </ejb-jar>
```

トランザクションをロールバックするには、次のコード例のとおり、WebLogic 拡張の `TXHelper` か、または MDB コンテキストを使用します。

```
UserTransaction ut =
   weblogic.transaction.TXHelper.getUserTransaction();
   ut.setRollbackOnly();
```

または

```
private MessageDrivenContext context;
```

```
public void setMessageDrivenContext(
  MessageDrivenContext mycontext) {
    context = mycontext;
}
public void onMessage(Message msg) {
try {    // 何らかのロジック
}
   catch(Exception e) {
   System.out.println("MDB doing rollback");
   context.setRollbackOnly();
}
```

Q. サーバ セッション プールとメッセージ駆動型 Bean を比較したいのですが。

A. MDB は、1 つの送り先だけでリスンします。コンシューマは、1 つの送り先だけでリスンします。ConnectionConsumers は、1 つの送り先だけでリスンします。

ServerSessionPool は、複数の ConnectionConsumers を持つことができます。このため、1 つの MessageListener を複数のコンシューマによって供給できます。

ServerSessionPools は、MessageListener にトランザクションを関連付けません。このため、リスナを実行したメッセージの受信は、トランザクションの一部ではありません。MDB の場合、トランザクションを REQUIRED として指定でき、メッセージの起動はトランザクションの一部です。

ServerSessionPools には、接続コンシューマがリスンしている送り先が同じ JVM によってホストされなければならないという制限があります。つまり、リモート キューまたはトピックをリスンするサーバ セッション プールを持つことができず、外部（非 WebLogic Server JMS）送り先はサポートされません。MDB は配布することができ、外部送り先を持つことができます。

MDB は、現時点では 1 回のデプロイメントにつき 1 つのメッセージのみを受信します（サービス パック 1 の場合）。トピックをリスンする単一のサーバ上の複数のインスタンスを持つ MDB が存在する場合、その MDB は、インスタンスの数に関係なく、パブリッシュされたメッセージのコピーを 1 つだけ受信します。トピックをリスンする複数のマシンに MDB がデプロイされている場合、各デプロイメントはそのトピックのパブリッシュされたメッセージのコピーを受信します。すべての MDB デプロイメントに均等に配布されるメッセージの複数のコピーを取得します。

15 FAQ: WebLogic メッセージング ブリッジ

- メッセージング ブリッジがソース ブリッジ送り先に接続できないのはなぜですか。
- 「かならず 1 回」サービス品質を使って 2 フェーズ トランザクションを行うようにメッセージング ブリッジをコンフィグレーションしました。それに対し、「サービスの品質に達することができない」という意味のエラーが発生するのはなぜですか。
- ソースまたは対象の対象ブリッジ送り先で「かならず 1 回」のサービスを利用できない場合、サービス品質を自動的に下げるようにメッセージング ブリッジをコンフィグレーションすることはできますか。
- WebLogic Server 6.1 の送り先からリリース 7.0 以降の送り先にメッセージを転送しようとすると、セキュリティ認証例外が発生するのはなぜですか。
- ソースまたは対象メッセージング ブリッジ送り先を設定するとき、アダプタのクラスパス フィールドの設定は必要ですか。
- WebLogic Server 6.1 ドメインと、それとは異なるリリース 7.0 以降のドメインの間で、メッセージング ブリッジを使って恒久サブスクリプション メッセージを転送することはできますか。
- メッセージング ブリッジのデバッグを有効にする方法を教えてください。
- Adminstration Console の [サーバ | サービス | モニタ | メッセージング ブリッジ] ページで、メッセージング ブリッジのモニタ状態は何を示していますか。
- Adminstration Console を使わないでメッセージング ブリッジをモニタする方法はありますか。

Q. メッセージング ブリッジがソース ブリッジ送り先に接続できないのはなぜですか。

A. ソース ブリッジ送り先パラメータをコンフィグレーションするときにエラーが発生しています。または、実際のソース送り先が動作してないため、メッセージング ブリッジと通信できません。

- ブリッジのソース送り先が正しくコンフィグレーションされているかどうか調べます。そのためには、[JMS ブリッジ送り先 | コンフィグレーション | 一般] コンソール ページで、以下のフィールドが正しく設定されていることを確認します。
  - [接続 URL] — 接続ファクトリおよび実際の送り先をロックアップするために使用する JNDI プロバイダの URL でなければなりません。
  - [送り先 JNDI 名] — ソース ブリッジ送り先にマップされる実際の送り先の JNDI 名でなければなりません。
  - [接続ファクトリ JNDI 名] — ソース ブリッジ送り先にマップされる実際の送り先に対する接続を作成するために使用する接続ファクトリでなければなりません。
  - [ユーザ名]/[ユーザ パスワード] — 実際のソース送り先にアクセスするためのパーミッションを持つユーザ ID でなければなりません。
- ソース ブリッジ送り先にマップされる実際のソース キュー送り先またはソース トピック送り先が動作していて問題がないことを、以下のようにして確認します。
  - ソース送り先をホストしている WebLogic Server インスタンスが稼働していますか。
  - ソース送り先をホストしている JMS サーバが正しくデプロイされていますか。

注意 : ソース ブリッジ送り先の接続障害を解決するためのこのトラブルシューティング手順は、対象ブリッジ送り先に対しても利用できます。

Q. 異なる WebLogic Server ドメイン間または異なるリリース間の 2 フェーズ トランザクションまたはグローバル トランザクションを、メッセージング ブリッジで処理できますか。

A. 通信を行うソースと対象の WebLogic ドメインがどちらもリリース 6.1 SP3 以降を実行していて、かつブリッジが「かならず 1 回」品質のサービスを使用するようにコンフィグレーションされている場合に限り、可能です。

ただし、ブリッジがリリース 7.0 以降のドメインと通信するには、リリース 6.1 ドメインとリリース 7.0 以降のドメインの間に信頼関係を確立する必要があります。15-60 ページの「WebLogic Server 6.1 の送り先からリリース 7.0 以降の送り先にメッセージを転送しようとすると、セキュリティ認証例外が発生するのはなぜですか。」の説明を参照してください。

リリース 6.1 以降のドメイン間の相互運用で「かならず 1 回」の QOS を使用する方法の詳細については、『管理者ガイド』の「リリース 6.1 以降のドメイン内にある送り先へのメッセージング ブリッジを用いたアクセス」を参照してください。

Q. 「かならず 1 回」サービス品質を使って 2 フェーズ トランザクションを行うようにメッセージング ブリッジをコンフィグレーションしました。それに対し、「サービスの品質に達することができない」という意味のエラーが発生するのはなぜですか。

A. WebLogic ドメイン間のトランザクションをメッセージング ブリッジで処理するには、次のような追加設定が必要です。

- サポートされているアダプタは、`WL_HOME\lib` ディレクトリにあります。「かならず 1 回」QOS を使用するには、Console の [デプロイメント | アプリケーション] ノードを選択して、ブリッジが動作しているリリース 6.1 ドメインにトランザクション アダプタ `jms-xa-adp.rar` をデプロイする必要があります。
- ソースと対象の両方のブリッジ送り先に対し、[JMS ブリッジ送り先 | コンフィグレーション] タブの [JNDI アダプタ名] 属性に「`eis.jms.WLSConnectionFactoryJNDIXA`」と設定して、この `jms-xa-adp.rar` アダプタを示す必要があります。
- WebLogic JMS の場合は、ソースと対象の両方のブリッジ送り先にマップされるキューまたはトピックについて、トランザクション対応の XAConnectionFactory を使用していることを確認します。確認するには、Console の [JMS | 接続ファクトリ | コンフィグレーション | トランザクション] タブまたはコンフィグレーション ファイル (`config.xml`) で以下の属性を設定する必要があります。

  ```
  UserTransactionsEnabled=true
  XAConnectionFactory=true
  ```

- サードパーティ ベンダの JMS の場合は、ソースと対象のブリッジ送り先にマップされる送り先に対し、トランザクション対応の接続ファクトリを使用していることを確認します。

リリース 6.1 以降のドメイン間の相互運用で「かならず 1 回」の QOS を使用する方法の詳細については、『管理者ガイド』の「リリース 6.1 以降のドメイン内にある送り先へのメッセージング ブリッジを用いたアクセス」を参照してください。

Q. ソースまたは対象の対象ブリッジ送り先で「かならず 1 回」のサービスを利用できない場合、サービス品質を自動的に下げるようにメッセージング ブリッジをコンフィグレーションすることはできますか。

A. できます。Administration Console の [メッセージング ブリッジ | コンフィグレーション | 一般] ページで、[QOS デグラデーション] チェックボックスを選択してください。

Q. WebLogic Server 6.1 の送り先からリリース 7.0 以降の送り先にメッセージを転送しようとすると、セキュリティ認証例外が発生するのはなぜですか。

```
java.lang.SecurityException: Invalid Subject: principals=[system]
```

A. WebLogic Server 6.x では、2 つの WebLogic Server ドメイン間の信頼関係は、両ドメインのシステム パスワードが同じであるときに確立されました。WebLogic Server 7.0 以降では、あるドメインの資格属性が別のドメインの資格属性と一致したときに信頼関係が確立されます。したがって、WebLogic Server 6.x のドメインとリリース 7.0 以降のドメインを相互運用する必要がある場合は、両方のドメインの資格属性を、リリース 6.x ドメインの「system」ユーザのパスワードに変更する必要があります。

- リリース 6.1 のドメインとリリース 7.0 以降のドメインの間にメッセージング ブリッジのための信頼関係を確立する方法の詳細については、『管理者ガイド』の「WebLogic ドメインに対するセキュリティの相互運用性の有効化」を参照してください。
- リリース 6.1 ドメインの相互運用のセキュリティに関する詳細については、『WebLogic Security の管理』の「互換性セキュリティの使い方」を参照してください。
- WebLogic Server 7.0 ドメイン間のセキュリティの詳細については、『Web Logic Security の管理』の「WebLogic ドメイン間の信頼関係の有効化」を参照してください。

Q. メッセージ ブリッジが実行されている WebLogic 6.1 ドメインにトランザクション対応の `jms-xa-adp.rar` リソース アダプタをデプロイしているのに、「ブリッジ アダプタが見つかりません」というメッセージが出るのはなぜですか。

A. ブリッジが送り先と通信するには、ソースと対象の*両方*のブリッジ送り先を、適切な .rar アダプタに関連付ける必要があります。jms-xa-adp.rar トランザクション アダプタの場合は、ソースと対象の両方のブリッジ送り先について、[JMS ブリッジ送り先 | コンフィグレーション] タブの [JNDI アダプタ名] 属性に「eis.jms.WLSConnectionFactoryJNDIXA 」と設定し、アダプタを示す必要があります。

注意 : 「ブリッジ アダプタが見つかりません」というメッセージは、1 回だけ発生する場合には必ずしも問題を示しているとは限りません。一方、繰り返し発生する場合は、アダプタのデプロイメントと、ソースおよび対象のブリッジ送り先で使用されている JNDI アダプタ名を確認する必要があります。

ブリッジ リソース アダプタの詳細については、『管理者ガイド』の「ブリッジ アダプタの使い方」を参照してください。

Q. ソースまたは対象メッセージング ブリッジ送り先を設定するとき、アダプタのクラスパス フィールドの設定は必要ですか。

A. ソース送り先、対象送り先の両方がリリース 6.1 以降で実行されている場合は、アダプタのクラスパス フィールドはブランクにしておきます。送り先のいずれかが 6.0 以前のリリースで実行されている場合は、アダプタの Classpath フィールドに古い方の WebLogic Server リリースのクラスの場所を設定する必要があります。サードパーティの JMS プロバイダに接続するときは、WebLogic Server の CLASSPATH にプロバイダの CLASSPATH を指定する必要があります。

Q. WebLogic Server 6.1 ドメインと、それとは異なるリリース 7.0 以降のドメインの間で、メッセージング ブリッジを使って恒久サブスクリプション メッセージを転送することはできますか。

A. WebLogic 7.0 ドメインがサービス パック 1 以降を使用している場合に限り可能です。メッセージング ブリッジを介した恒久メッセージを有効にするには、Administration Console の [Messaging Bridge | コンフィグレーション | 一般] タブで [永続性を有効化] 属性を選択します。

Q. メッセージング ブリッジのデバッグを有効にする方法を教えてください。

A. メッセージング ブリッジのデバッグは、次のいずれかの方法を使って有効にできます。

- WebLogic の起動スクリプトに、以下の行を追加します (weblogic.Server の行の前)。

  ```
  -Dweblogic.Debug.DebugMessagingBridgeStartup=true
  -Dweblogic.Debug.DebugMessagingBridgeRuntime=true
  ```

- メッセージング ブリッジを実行するサーバのコンフィグレーション ファイル (config.xml) の ServerDebug エントリに、以下の文を追加します。

```
DebugMessagingBridgeStartup="true"
DebugMessagingBridgeRuntime="true"
```

メッセージング ブリッジのデバッグを有効にすると、デバッグ メッセージはデフォルトでサーバ ログに送られます。Administration Console にデバッグ メッセージを表示する場合は、前記の文に「DumpToConsole」を追加します。次に例を示します。

```
-Dweblogic.Debug.DebugMessagingBridgeStartupDumpToConsole=true
```

Q. Adminstration Console の [サーバ | サービス | モニタ | メッセージング ブリッジ] ページで、メッセージング ブリッジのモニタ状態は何を示していますか。

A. メッセージング ブリッジの状態をモニタするときは、必要に応じて、次の表を使って一連のアクションを決定してください。

状態	アクション
WARN: Failed to find the source adapter	アダプタがデプロイされていること、またはソース JMSBridgeDestination インスタンスの JNDI 名が正しいことを確認する。
WARN: Failed to find the target adapter	アダプタがデプロイされていること、または対象 JMSBridgeDestination インスタンスの JNDI 名が正しいことを確認する。
Found both of the adapters and making connections	なし。
WARN: Stopped by the administrator	なし。
WARN: Failed to look up the source adapter	アダプタがデプロイされていること、またはソース JMSBridgeDestination インスタンスの JNDI 名が正しいことを確認する。
WARN: Failed to look up the target adapter	アダプタがデプロイされていること、または対象 JMSBridgeDestination インスタンスの JNDI 名が正しいことを確認する。

状態	アクション
Found two adapters and about to make connections	なし。
WARN: Failed to connect to the source	ソース ブリッジ送り先に対してコンフィグレーションされているすべてのパラメータを確認する。 ソース サーバが稼働していて、実際の送り先がアクティブであることを確認する 。
Connected to the source	なし。
WARN: Failed to connect to the target	対象ブリッジ送り先に対してコンフィグレーションされているすべてのパラメータを確認する。 対象サーバが稼働していて、実際の送り先がアクティブであることを確認する 。
Connected to the target	なし。
Forwarding messages	なし。
WARN: Failed to connect and will reconnect later	ソースと対象のブリッジ送り先が正常に動作していることを確認する。

Q. Adminstration Console を使わないでメッセージング ブリッジをモニタする方法はありますか。

A. はい、ブリッジインスタンスごとに実行時 MBean (`MessagingBridgeRuntimeMBean`) があります。WebLogic Server の実行時 MBean は、ドメインのリソースに関する情報のスナップショットを提供します。ドメイン内の特定のリソース (メッセージング ブリッジなど) がインスタンス化されると、MBean のインスタンスが作成されて、そのリソースに関する情報を収集します。

現在、`MessagingBridgeRuntimeMBean` の `getState()` メソッドは文字列 (「Active」または「Inactive」) を返し、`getDescription()` メソッドはさらに詳細な情報の文字列を返すようになっています。ブリッジの実行時 MBean の名前は、WebLogic Server インスタンスの名前とブリッジの名前で構成されます。*mybridge* というブリッジが *myserver* という WebLogic Server インスタンスで実行されると、ブリッジの実行時 MBean の名前は *myserver.bridge.mybridge* になります。

実行時 MBean 管理コマンドの使い方の詳細については、『管理者ガイド』の「WebLogic Server コマンドライン インタフェース リファレンス」を参照してください。MBean のモニタ通知のプログラミングに関する詳細については、『WebLogic JMX Service プログラマーズ ガイド』の「WebLogic Server MBean のモニタ」を参照してください。

16 FAQ: JTA

- 分散トランザクションで XA 以外のドライバを使用できますか。
- 分散トランザクションで複数の XA 以外の接続プールを使用できますか。
- 分散トランザクションでの XA ドライバと XA 以外のドライバの違いは何ですか。
- WebLogic jDriver for Oracle/XA に加えてどの XA ドライバを使用できますか。
- 分散トランザクションで、Oracle Thin ドライバを XA ドライバとして使用できますか。
- SQLException「Result set already closed」メッセージが表示されるのはなぜですか。
- JMS と 1 つの XA 以外の JDBC ドライバを使用する場合に 2PC ライセンスは必要ですか。
- JMS と XA 以外のドライバを使用している場合に例外が送出されるのはなぜですか。
- EJB CMP 1.1 を使用している場合に例外が送出されるのはなぜですか。
- EJB CMP 1.1 と XA 接続プールを使用しているときに例外が送出されるのはなぜですか。
- 分散トランザクションを開始する前に JDBC 接続を取得できますか。
- 分散トランザクションがコミットまたはロールバックされた後に JDBC 接続を閉じることができますか。
- `XAResource` にアクセスしたときに、「`Internal error: XAResource '<name>' is unavailable`」という `XAER_RMFAIL XAException` を受け取りました。これは何を意味しているのですか。また、どのように処理すればいいのですか。

Q. 分散トランザクションで XA 以外のドライバを使用できますか。

A. XA 以外の接続プールが、複数のサーバに分散したトランザクションに参加している唯一のリソースである場合は、XA 以外のドライバの TxDataSource をコンフィグレーションする必要があります。このコンフィグレーションは、WLS 5.1 で JTS ドライバを使用する場合と同じです。

ただし、複数のリソースが分散トランザクションに参加する場合は、TxDataSource プロパティ `EnableTwoPhaseCommit=true` も設定する必要があります。詳細については、『管理者ガイド』の「JDBC 接続の管理」を参照してください。どちらの場合でも、常に、非推奨の DriverManager インタフェースではなく DataSource インタフェースを通じて接続を取得します。DriverManager を通じて接続を取得した場合、インタフェースでは TxDataSource の `EnableTwoPhaseCommit` 設定を取得できません。その場合は、分散トランザクションで予期しない動作が発生する場合があります。また、DataSource インタフェースを使用する場合は、URL または特定の WebLogic 多層ドライバ（JTS、RMI、またはプール）を区別する必要がありません。URL および特定のドライバは、config.xml ファイルおよび JNDI ルックアップを通じて取得されます。

Q. 分散トランザクションで複数の XA 以外の接続プールを使用できますか。

A. できません。両方の接続プールの TxDataSource で `EnableTwoPhaseCommit=true` を設定しても、同じ分散トランザクションで 2 つの XA 以外の接続プールを使用しようとすると、2 番目の XA 以外の接続プールから接続を取得しようとしたときに、

"java.sql.SQLException: Connection has already been created in this tx context for pool named <first pool's name>.Illegal attempt to create connection from another pool: <second pool's name>"

という例外になります。

Q. 分散トランザクションでの XA ドライバと XA 以外のドライバの違いは何ですか。

A. XA JDBC ドライバと XA 以外の JDBC ドライバの違いは以下のとおりです。

- **原子性の保証**。XA ドライバは XAResource インタフェースを実装し、WLS トランザクション マネージャによって制御される 2PC プロトコルにフルに参加できます。このため、複数の参加リソースにまたがる更新の原子性が保証されます。

  しかし、XA 以外のドライバは XAResource インタフェースを実装せず、2PC プロトコルにはフルに参加できません。分散トランザクションで XA 以

外のドライバを使用する場合は、WLS がドライバの代わりに XAResource ラッパーを実装します。データ ソース プロパティ `enableTwoPhaseCommit` が `true` に設定されている場合、WLS XAResource ラッパーはトランザクション マネージャが `prepare` メソッドを呼び出したときに XA_OK を返します。トランザクション マネージャが第 2 フェーズで `commit()` または `rollback()` を呼び出すと、WLS XAResource ラッパーは `commit()` または `rollback()` の呼び出しを XA 以外の JDBC 接続に委託します。`commit` または `rollback()` の途中で障害が発生すると、ヒューリスティックな例外が発生します。ヒューリスティック エラーの結果、アプリケーション データは矛盾した状態のまま残される場合があります。

- **接続のリダイレクト。**「分散トランザクションで XA 以外のドライバを使用できますか。」で説明されているように、WLS 5.1 の場合と同じで、XA 以外のドライバは同じ分散トランザクションで複数のプロセスからの更新を実行するようにコンフィグレーションできます。WLS の内部では、別々のプロセスからの JDBC 呼び出しが 1 つのプロセスの同じ物理 JDBC 接続にリダイレクトされます。ただし、XA ドライバを使用する場合は、このようなリダイレクトは行われません。各プロセスは独自のローカル XA データベース接続を使用し、データベースによって、同じ分散トランザクションで行われる別々のプロセスからのすべての分散更新が確実に原子性を維持してコミットされます。

- **接続の管理。**分散トランザクションで XA 以外のドライバを使用するのか、それとも XA ドライバを使用するのかに関係なく、WLS は、すべての JDBC 呼び出しをインターセプトし、必要に応じて接続プールから物理 JDBC 接続を取得する JDBC ラッパーを実装します。

  - 分散トランザクションで XA 以外のドライバを使用する場合、別々のプロセスから行われる更新が原子性を維持してコミットされるようにするために、WLS はコミットまたはロールバックされるまで同じ物理 JDBC 接続を分散トランザクションと関連付けます。その結果、XA 以外の接続プールを使用するアクティブな分散トランザクションの数は、JDBC 接続プールの最大容量によって制限されます。

  - XA ドライバを使用する場合、接続の管理はもっとスケーラブルです。トランザクションがコミットまたはロールバックされるまで、WLS が同じ物理 XA 接続を保持するということはありません。実際、ほとんどの場合では、XA 接続はメソッド呼び出しの間だけ保持されます。WLS JDBC ラッパーは、すべての JDBC 呼び出しをインターセプトし、必要に応じて XA 接続と関連付けられた XAResource を取得します。メソッド呼び

出しが呼び出し側に戻るか、またはさらに別のサーバを呼び出すと、WLS は XA 接続と関連付けられた XAResource を解放します。

- さらに、WLS は開いている結果セットがなければ XA 接続を接続プールに返します。また、コミット処理時に、XAResource オブジェクトは任意数の分散トランザクションを平行してコミットするために使用することもできます。結果として、XA 接続プールを使用するアクティブな分散トランザクションの数と同時コミットまたはロールバックの数の両方とも、接続プールの最大容量によって制限されることはありません。接続プールの最大容量によって制限されるのは、同時データベース アクセスの数だけです。

Q. WebLogic jDriver for Oracle/XA に加えてどの XA ドライバを使用できますか。

A. 理論的には、JDBC 2.0 規格の拡張仕様に準拠していればどのサードパーティ XA ドライバでも使用できます。ただし、個々のベンダの XA ドライバには、正しく機能することを妨げるバグがある場合もあります。

コンフィグレーションの詳細については、http://edocs.beasys.co.jp/e-docs/wls61/adminguide/managetx.html で JDBC コンフィグレーションのガイドラインを参照してください。

Q. 分散トランザクションで、Oracle Thin ドライバを XA ドライバとして使用できますか。

A. Oracle 8.1.6 Thin ドライバには外部 Xid を受け付けないバグがあるので、WLS を含む他のすべてのベンダのトランザクション マネージャでまったく機能しません。

Oracle 8.1.7 Thin ドライバにはスレッド処理に関する問題があるため、次の解決策を考えました。この解決策では、準備、コミット、およびロールバックの処理期間に専用の XA 接続を使用します。これは、XAResource オブジェクトが任意数のトランザクションを平行してコミットするために使用されるという点で、デフォルトの XA 接続管理モデル（FAQ 3 を参照）とは異なります。このため、同時コミットの数が XA 接続プールの最大容量に制限されます。この次善策は Oracle 専用であるため、他の XA ドライバには影響しません。

次善策を利用せずに SP1 で試してみる場合は、XA 接続プールをコンフィグレーションできます。詳細については、『管理者ガイド』の「JDBC 接続の管理」を参照してください。

Q. SQLException「Result set already closed」メッセージが表示されるのはなぜですか。

問題 : クライアント サイドから Weblogic jDriver for Oracle/XA（トランザクション モード）を使用しています。分散トランザクションでの更新は正常に機能します。しかし、クエリを実行しようとすると、「`SQLException Result set already closed`」というメッセージが表示されます。どうしたら回避できますか。

A. Weblogic jDriver for Oracle には、メソッドが呼び出し側に戻ったときに開いているすべての結果セットを閉じるという制限があります。詳細については、「分散トランザクションでの WebLogic jDriver for Oracle/XA の使い方」の「WebLogic jDriver for Oracle XA の制限」を参照してください。

Bean など、サーバサイドからドライバを使用する場合、このような制限はありません。サーバサイドからドライバを使用することは、アプリケーション アーキテクチャやパフォーマンスの観点からも推奨されます。クライアントサイドからドライバを使用すると、すべての JCBC 呼び出しで往復のコストがかかります。

この制限があるのは、Weblogic jDriver for Oracle XA が Oracle の OCI API と C XA スイッチを使用して実装され、マルチスレッド モードで OCI と XA を使用する場合に Oracle に問題があるためです。開かれたものとは異なる OCI カーソルをスレッドで閉じると、サーバのクラッシュや予期しない動作が発生する場合があります。結果として、Weblogic ドライバは呼び出しが呼び出し側に戻るときに開いているすべての結果セットを暗黙的に閉じます。

Q. JMS と 1 つの XA 以外の JDBC ドライバを使用する場合に 2PC ライセンスは必要ですか。

A. 必要です。JMS も、分散トランザクションに参加する XAResource です。したがって、分散トランザクションには 2 つのリソースが参加しているので、2PC ライセンスが必要です。

Q. JMS と XA 以外のドライバを使用している場合に例外が送出されるのはなぜですか。

問題 : JMS と 1 つの XA 以外の JDBC ドライバを使用しています。「`javax.transaction.xa.XAException: JDBC driver does not support XA, hence cannot be a participant in two-phase commit`」という例外で、トランザクションのコミットが失敗します。

A. 「JMS と 1 つの XA 以外の JDBC ドライバを使用する場合に 2PC ライセンスは必要ですか。」の質問で言及したように、JMS も分散トランザクションに参

加する XAResource です。分散トランザクションに複数のリソースが参加している場合は、「分散トランザクションで XA 以外のドライバを使用できますか。」で説明されているようにデータ ソース プロパティ `EnableTwoPhaseCommit=true` を設定する必要があります。

Q. EJB CMP 1.1 を使用している場合に例外が送出されるのはなぜですか。

問題 : EJB CMP 1.1 および XA 以外の接続プールとともに分散トランザクションを使用しています。JDBC 接続プールと TxDataSource は、「分散トランザクションで XA 以外のドライバを使用できますか。」の指示どおりにコンフィグレーションしました。それでも、コミット時には、「`javax.transaction.xa.XAException: JDBC driver does not support XA, hence cannot be a participant in two-phase commit`」という例外が送出されます。なぜでしょうか。

A. 旧式の CMP 1.1 DTD では、データ ソース名を指定できません。接続プール名だけが指定されている場合、TxDataSource の `EnableTwoPhaseCommit` 設定は無視されます。新しい CMP 1.1 DTD を使用し、プール名の代わりにデータ ソース名を指定する必要があります。http://www.bea.com/servers/wls600/dtd/weblogic-rdbms11-persistence-600.dtd を参照してください。

Q. EJB CMP 1.1 と XA 接続プールを使用しているときに例外が送出されるのはなぜですか。

問題 : EJB CMP 1.1 および 2 つの XA 接続プールとともに分散トランザクションを使用しています。XA JDBC 接続プールと TxDataSource は、『管理者ガイド』の「JDBC 接続の管理」の指示どおりにコンフィグレーションしました。しかし、次の例外を受け取りました。「Couldn't get connection:java.sql.SQLException:Connection has already been created in this tx context for pool named <pool name> Pool.Illegal attempt to create connection from another pool:<pool name> Pool」。なぜでしょうか。

A. 回答 : 旧式の CMP 1.1 DTD では、データ ソース名を指定できません。接続プール名しか指定されない場合、非 XA 接続プールのコード パスが誤って代わりに実行されてしまいます。新しい CMP 1.1 DTD を使用し、プール名の代わりにデータ ソース名を指定する必要があります。

記述子が最新の DTD ファイルを使用するようにするには、WebLogic CMP 1.1 記述ファイルの DOCTYPE ヘッダが次のとおりであることを確認してください。

```
<!DOCTYPE weblogic-rdbms-jar PUBLIC
'-//BEA Systems, Inc.//DTD WebLogic 6.0.0 EJB 1.1 RDBMS
Persistence//EN'
```

```
http://www.bea.com/servers/wls600/dtd/weblogic-rdbms11-persistenc
e-600.dtd'>
```

DTD ファイルを見るには、
http://www.bea.com/servers/wls600/dtd/weblogic-rdbms11-persistence-600.dtd にアクセスしてください。

Q. 分散トランザクションを開始する前に JDBC 接続を取得できますか。

A. ドライバが XA であるかどうかによって異なります。

- 分散トランザクションで XA 以外のドライバを使用する場合は、常に、分散トランザクションの開始後に JDBC 接続を取得します。
- XA ドライバを使用する場合は、分散トランザクションが開始される前または後に接続を取得できます。

Q. 分散トランザクションがコミットまたはロールバックされた後に JDBC 接続を閉じることができますか。

A. XA 以外のドライバと XA ドライバのどちらの場合でも、分散トランザクションが完了した後に接続を閉じることができます。

Q. `XAResource` にアクセスしたときに、「`Internal error: XAResource '<name>' is unavailable`」という `XAER_RMFAIL XAException` を受け取りました。これは何を意味しているのですか。また、どのように処理すればいいのですか。

A. JTA には、次の機能を備えた独自のリソース モニタが用意されています。

リソースがアクティブであると見なされるのは、保留中の要求が存在しない場合、または `XAResource` の保留中の要求から `XAER_RMFAIL` ではない結果を取得した場合です。`XAResource` が 2 分以内にアクティブにならない場合、XAResource は応答なしと宣言されます。`XAResource` に対する新たな要求は拒否され、上記のような `XAER_RMFAIL XAException` が送出されます。この目的は、`RM` が応答なし状態の場合にスレッドのさらなる損失を防ぐことです。

リソースが再びアクティブであると宣言されるのは、`weblogic.transaction.TransactionManager.unregisterResource` に続いて `registerStaticResource` または `registerDynamicResource` を呼び出して `XAResource` を WebLogic Server Transaction Manager に再登録した場合か、または 30 分のタイムアウト期間の経過後です。WLS JDBC 接続プールを使用する場合、JDBC 接続プールのリフレッシュ機能を有効にする（接続プールの「RefreshMinutes」プロパティを指定する）だけで済みます。接続プールのリフ

レッシュに成功すると、対応する XAResource が自動的に再登録されます。`weblogic.transaction.TransactionManager.registerStaticResource` または `registerDynamicResource` API のいずれかを介して独自の XAResource を登録する場合、`weblogic.transaction.TransactionManager.unregisterResource` に続いて `registerStaticResource` または `registerDynamicResource` を呼び出して XAResource を再登録する必要があります。

一般に、起こりうる RM 問題をデバッグする方法は、WLS の起動時に JVM パラメータとして `-Dweblogic.Debug=weblogic.JTAXA` を指定することによって JTA XA デバッグを有効にすることです。

17 FAQ: プラグイン

- プラグインはどのように機能するのですか。
- プラグインはどのようにして維持型のセッションのリクエストをルーティングするのですか。
- プラグインのデバッグに関して、WebLogic Server 6.0 で新しくなった点は何ですか。
- `wlproxy.log` の内容はどのようになりますか。
- 6.1 プラグインにおける変更点は何ですか。
- 静的リスト、動的リスト、および一般リストとは何ですか。
- 6.1 のプラグインでは、相互 SSL をサポートしていますか。
- WebLogic Server からプラグインへの応答に `Set-Cookie` ヘッダが含まれていることがあります。これは正常ですか。
- どうすれば、WL-Proxy-Client-Cert ヘッダを使用する偽装クライアントによるセキュリティ攻撃から WebLogic Server を守ることができますか。

Q. プラグインはどのように機能するのですか。

A. プラグインの仕組みの詳細については、『管理者ガイド』の「Apache HTTP Server プラグインのインストールとコンフィグレーション」を参照してください。

Q. プラグインはどのようにして維持型のセッションのリクエストをルーティングするのですか。

A. ブラウザからクッキーが送信されると、「Cookie: 」ヘッダ内で「JSESSIONID」（「CookieName」というパラメータでコンフィグレーション可能）が検索されます。

クッキーが無効になっており、URL 書き換えが利用される場合、セッション ID は URL 内にエンコードされます。WebLogic Server 5.1 以前のバージョンでは、この ID はクエリ文字列内にエンコードされていました。

```
?WebLogicSession=my_dumy_session
```

WebLogic Server 6.0 以降のバージョンでは、この ID はパラメータ内にエンコードされていました。

```
;JSESSIONID=my_dummy_session
```

セッションがクエリ文字列またはパラメータ内に見つからず、またメモリに読み込めるほど小さい場合、WebLogic Server は POST データ内でセッションを検索します。

Q. プラグインのデバッグに関して、WebLogic Server 6.0 で新しくなった点は何ですか。

```
"Debug = ON" logs only informational and error messages
HFC : headers from the client, informational, and error messages
HTW : headers sent to wls, informational and error messages
HFW : headers sent from wls, informational and error messages
HTC : headers sent to the client, informational and error messages
ALL : everything
OFF : nothing -- default(should be used in production)
```

ログ ファイルは 6.1 用にコンフィグレーションできます。後のバージョンの WebLogic Server 用に、デバッグ ファイルの名前および場所をコンフィグレーションするための `WLLogFile` が導入されていました。

Q. `wlproxy.log` の内容はどのようになりますか。

それぞれのリクエストは、次のように表示されます。

```
"================New Request: [GET / HTTP/1.1] ================="

PathTrim, DefaultFileName, and PathPrepend will be performed in
order. The final request will be logged as the following: "Fri Jun
22 14:24:40 2001 The request string is '/index.jsp'"
```

セッション情報を探してプライマリを決定しています。

```
"Fri Jun 22 14:24:40 2001 Initializing lastIndex=0 for a list of
length=1 Fri Jun 22 14:24:40 2001 create a new server node:
id='qa89:443' server_name='mint.beasys.com', port='18080'"
```

SecureProxy が ON に設定されている場合は、SSL を初期化します。

```
"Fri Jun 22 14:24:40 2001 INFO: SSL is configured Fri Jun 22
14:24:40 2001 INFO: Initializing SSL library Fri Jun 22 14:24:40
2001 Loaded 1 trusted CA's Fri Jun 22 14:24:40 2001 INFO:
Successfully initialized SSL Fri Jun 22 14:24:40 2001 INFO: SSL
configured successfully"
```

初期接続を行っています。

```
"Fri Jun 22 14:24:40 2001 general list: trying connect to
'172.17.9.180'/443 Fri Jun 22 14:24:40 2001 Connected to
172.17.9.180:443"
```

クライアント ヘッダおよび Post データ（存在する場合）を読み取っています。

```
"Fri Jun 22 14:24:40 2001 Hdrs from clnt:[Accept]=[image/gif,
image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/msword,
application/vnd.ms-powerpoint, */*]"
```

クライアント ヘッダおよび Post データ（存在する場合）を送信しています。

```
"Fri Jun 22 14:24:40 2001 Hdrs to WLS:[Accept]=[image/gif,
image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/msword,
application/vnd.ms-powerpoint, */*]"
```

WebLogic Server から応答ヘッダを取得しています。

```
"Fri Jun 22 14:24:46 2001 Hdrs from
WLS:[Set-Cookie]=[JSESSIONID=OzI19WqYmFnRviHEu5gKLvot42ABeD8NPWnF
0jW6cawSGcrp2mru!4038528127411848936!-1408169548!80!443; path=/]
Fri Jun 22 14:24:46 2001 parsed all headers OK"
```

応答ヘッダが WebLogic Server へ送信され、接続がクローズであるかキープアライブであるかを示しています。

```
"Fri Jun 22 14:24:46 2001 Hdrs to client:[Date]=[Fri, 22 Jun 2001
21:24:48 GMT] Fri Jun 22 14:24:46 2001 Hdrs to
client:[Server]=[WebLogic WebLogic Server 6.1 beta 06/21/2001
10:44:44 #122398 - internal build by jlee on client jlee.qa89] Fri
Jun 22 14:24:46 2001 canRecycle: conn=1 status=200 isKA=0 clen=2705
isCTE=0 Fri Jun 22 14:24:46 2001 closeConnection in load_utils:
deleting URL* Fri Jun 22 14:24:46 2001 INFO: Closing SSL context
Fri Jun 22 14:24:46 2001 INFO: sysSend 23 Fri Jun 22 14:24:46 2001
INFO: Error after SSLClose, socket may already have been closed by
peer Fri Jun 22 14:24:46 2001 r->status=200 returning 0"
```

Q. 6.1 プラグインにおける変更点は何ですか。

A. 変更点は次のとおりです。

- HTTP 1.1 のサポート -- チャンク転送およびキープアライブ（Apache 1.3.x 以外）
- セッション解析（これにより下位互換性がなくなります）
- プラグインから WebLogic Server への SSL のサポート

Q. 静的リスト、動的リスト、および一般リストとは何ですか。

A. 定義は以下のとおりです。

- 静的リスト：コンフィグレーション ファイルで定義される初期サーバ リスト
- 動的リスト：リクエストが正常に行われると WLS によって送信される、現在のサーバ リスト
- 一般リスト：現在のリクエストと関連付けられた、プライマリ サーバおよびセカンダリ サーバ以外の（静的または動的な）現在のサーバ リスト

Q. 6.1 のプラグインでは、相互 SSL をサポートしていますか。

A. していません。ただし、クライアント証明書を要求して、それを WebLogic Server に渡すように、プラグインを設定することはできます。次に例を示します。

```
apache ssl

SSLVerifyClient require

SSLVerifyDepth 10

SSLOptions +FakeBasicAuth +ExportCertData +CompatEnvVars
+StrictRequire
```

Q. WebLogic Server からプラグインへの応答に `Set-Cookie` ヘッダが含まれていることがあります。これは正常ですか。

A. はい。リクエストに `Cookie` ヘッダが入っていない場合、または `X-WebLogic-Force-Cookie` が検出された場合に、WebLogic Server は応答に `Set-Cookie` ヘッダを含めて送信します。プラグインは、障害の発生したサーバに接続できなかった場合、ヘッダ `X-WebLogic-Force-Cookie: true` を次に利用可能なサーバに送信し、正しいセッション情報が含まれる該当クッキーをクライアントに強制的に更新させます。

Q. Apache 1.3.19 に `mod_wl_ssl.so` を `mod_perl` と共にインストールして、プラグイン経由で WebLogic にアクセスすると、`mod_wl_ssl.so` において「Segmentation Fault (11)」が発生するのはなぜですか。

A. サーバとして任意の 6.x を、オペレーティング システムとして任意のバージョンの Solaris を使用できます。この環境は WebLogic Server 6.x（`mod_wl_ssl.so` を使用）、Solaris 2.x です。

「Segmentation Fault (11)」の発生を回避するには、次の例のように、HTTP 用の `VirtualHost` ブロックを追加します。

```
#   仮想ホストの一般的な設定
DocumentRoot "/export/home/tgoto/local/apache_1.3.19_dxu/htdocs"
ServerName tgoto1
ServerAdmin tgoto@toyota
ErrorLog /export/home/tgoto/local/apache_1.3.19_dxu/logs/error_log
TransferLog
/export/home/tgoto/local/apache_1.3.19_dxu/logs/access_log
</VirtualHost>
```

SSL ポートについては、以下の IP アドレスも使用してください。

```
<VirtualHost 206.189.223.111:443>
```

ServerName には任意の有効な DNS 名を使用します。

Q. どうすれば、WL-Proxy-Client-Cert ヘッダを使用する偽装クライアントによるセキュリティ攻撃から WebLogic Server を守ることができますか。

A. WebLogic Server に直接アクセスできるクライアントはすべて、WL-Proxy-Client-Cert ヘッダを偽装（使用）できます。WebLogic Server は、このヘッダから証明書情報を取得し、それが安全な送信元（プラグイン）からのものであることを信頼して、ユーザを認証するためにその情報を使用します。WebLogic Server の以前のリリースでは、このヘッダを常に信頼するのが、デフォルトの動作でした。現在では、WL-Proxy-Client-Cert ヘッダの信頼を明示的に定義する必要があります。新しいパラメータである `clientCertProxy` を使用すると、証明書ヘッダの暗黙の信頼を有効にすることができます。セキュリティのレベルを追加する必要がある場合は、接続フィルタを使って、WebLogic Server に対するすべての接続を制限します（つまり、プラグインが動作しているマシンからの接続だけを WebLogic Server が受け付けるようにします）。

`clientCertProxy` パラメータは、HTTPClusterServlet と Web アプリケーションに追加されています。

HTTPClusterServlet の場合は、次のようにして `web.xml` ファイルにパラメータを追加します。

```
<context-param>
    <param-name>clientCertProxy</param-name>
    <param-value>true</param-value>
</context-param>
```

Web アプリケーションの場合は、次のようにして `web.xml` ファイルにパラメータを追加します。

```
ServletRequestImpl context-param
<context-param>
    <param-name>weblogic.httpd.clientCertProxy</param-name>
    <param-value>true</param-value>
</context-param>
```

次のようにすれば、クラスタでこのパラメータを使用することもできます。

```
<Cluster ClusterAddress=”127.0.0.1” Name=”MyCluster”
    ClientCertProxyHeader=”true”/>
```

18 FAQ: サーバ関連の質問

- サーバが「ハング」または「フリーズ」してしまった場合は、どうすればよいでしょうか。
- SOCKS プロキシを使用するように WebLogic をコンフィグレーションする方法を教えてください。
- 接続の応答はどのようにしたら速くできますか。
- IIOP を介した CORBA とクライアントの通信を WebLogic はどのようにサポートしているのですか。
- HTTP トンネリングはどのようにしたら速くできますか。
- WebLogic Server は UNIX の起動と同時に起動できますか。
- クライアントとトラフィック以外に、サーブレットのパフォーマンスには何が影響しますか。
- Solaris で「NoClassDefFound」または「Too Many Open files」というメッセージが表示されるのはなぜですか。
- WebLogic Server のメモリを増やすにはどのようにしますか。
- ログ ファイルで Java.io 例外を引き起こす原因は何ですか。
- Java と CORBA の統合 : RMI-IIOP または Java IDL
- RMI-IIOP アプリケーションと既存の CORBA オブジェクトはどのように相互運用されるのですか。
- WebLogic Server で T3 はどのように機能するのですか。
- WebLogic Server で実行されている Java コードをデバッグする方法を教えてください。

Q. サーバが「ハング」または「フリーズ」してしまった場合は、どうすればよいでしょうか。

A. WebLogic Server が「フリーズ」した場合は、BEA テクニカル サポートに連絡する前に、スレッド ダンプや Java ガベージ コレクション メトリックなどの診断情報を収集する必要があります。詳細については、「診断情報の収集」を参照してください。

Q. SOCKS プロキシを使用するように WebLogic をコンフィグレーションする方法を教えてください。

A. SOCKS を使用するように java.net ソケットをコンフィグレーションするには、Java システム プロパティを設定します。詳細については、「How do I make Java work with a proxy server?」を参照してください。. プロパティを設定すると、WebLogic ソケット接続で SOCKS プロキシが使用されます。

Q. 接続の応答はどのようにしたら速くできますか。

A. 接続の遅延は、たいていは DNS の問題によって生じます。WebLogic では、新しい接続が行われるホスト名の逆引き参照が実行されます。プロキシ サーバからの接続であるために DNS の逆引き参照が正しく機能していない場合は、それが遅延の原因として考えられます。システム管理者との共同作業で、DNS およびサードパーティのネットワーキング ソフトウェアが正しく機能しているかどうかを確認する必要があります。あらゆる接続で逆引き参照を実行する単純なサーバ プログラムを記述してみてください。その参照が遅延する場合は、プロキシ サーバが問題の発生源となっています。

Q. IIOP を介した CORBA とクライアントの通信を WebLogic はどのようにサポートしているのですか。

A. 「CORBA」のサポートとは、さまざまな意味を含んでいます。たいていの場合、それは単純に IIOP/ORB のサポートを意味し、CORBA サービスはあまり重要ではありません。WebLogic では、多様な方法で CORBA をサポートしています。

まず、Java クライアントは CORBA 環境をトンネリングして WebLogic Server にアクセスできます。これは「IIOP トンネリング」と呼ばれる機能で、IONA Wonderwall などの IIOP ファイアウォールを経由するアプレットで使用します。これは、IIOP 通信フレームワーク上で機能する Java-to-Java モデルです。

WebLogic RMI over IIOP は、IIOP を介して多くのクライアント（CORBA クライアントを含む）に RMI サービスを提供します。詳細については、「WebLogic RMI over IIOP の使い方」を参照してください。

WebLogic Enterprise Connectivity を使用すると、BEA WebLogic Enterprise System への IIOP 接続プールを作成でき、WebLogic Server サーブレットおよびエンタープライズ JavaBean から WebLogic Enterprise CORBA オブジェクトを実行できるようになります。

Q. HTTP トンネリングはどのようにしたら速くできますか。

A. 残念ながら、HTTP トンネリングを使用するときには著しくパフォーマンスが低下します。ある程度は最適化していますが、すべてが HTTP でカプセル化されるので、HTTP トンネリングは Java-to-Java のダイレクトな TCP/IP 接続より低速になります。

必ず、本当に HTTP トンネリングが必要なのかを確認してください。たとえば、IP パケットがポート 80 を通じてファイアウォールを通過できる場合は、ポート 80 で高速な t3 プロトコルを使用できます。

HTTP トンネリングを使用してファイアウォールを通過する必要がある場合、e-Border は HTTP プロキシより性能が良い製品を提供しています。

Q. WebLogic Server は UNIX の起動と同時に起動できますか。

A. UNIX rc スクリプトに起動スクリプトを追加すると、UNIX の起動時に WebLogic Server を実行できます。次に、JDK 1.1 で動作する、HP-UX 11 からの例を示します。WebLogic Server の `URL` とシステム パスワードを指定する必要があります。このファイル `wlstart` は `/sbin/init.d` ディレクトリに配置し、`/sbin/rc2.d` ディレクトリにはこのファイルへのリンクがあります。

```
export SHLIB_PATH=\
/home/user1/weblogic/lib/hpux11:/oracle/8.0.4/lib
export CLASSPATH=/home/user1/weblogic/classes:\
/home/user1/weblogic/lib/weblogicaux.jar
export ORACLE_HOME=/oracle/8.0.4
export ORACLE_SID=DEMO
export ORACLE_TERM=vt100
export QAT=/home/user1/weblogic
cd $QAT
PATH=/sbin:/usr/sbin:/usr/bin:/opt/java/bin
export PATH
case $1 in
'start')
    java -ms64m -mx64m -verbosegc weblogic.Server > \
       /home/user1/weblogic/server.out 2>&1
    ;;
'stop')
    java weblogic.Admin URL shutdown system password
    ;;
*)
    echo "usage: $0 {start|stop}"
```

```
    ;;
esac
```

このようなスクリプトを設定する場合には、UNIX のシステム管理者と共同で作業する必要があります。

Q. クライアントとトラフィック以外に、サーブレットのパフォーマンスには何が影響しますか。

A. マシンでスクリーン セーバー（特に OpenGL スクリーン セーバー）を実行している場合は、サーブレットの応答時間が約 5 倍遅くなります。スクリーン セーバーをオフにして、速度が向上するか確認してください。

Q. Solaris で「NoClassDefFound」または「Too Many Open files」というメッセージが表示されるのはなぜですか。

問題 : Solaris で WebLogic Server を使用している状況でアプリケーションを実行しようとすると、「NoClassDefFound」エラーが発生します。ただし、エラーを生じさせたクラスは存在しており、適切なディレクトリに配置されています。同じディレクトリにはロードされる他のクラスが存在し、「Too many open files」エラーも発生します。

A. この状況は、ユーザ アカウントでファイル記述子が足りないときに見受けられます。Solaris では、各ユーザ アカウントに特定の数のファイル記述子があります。ファイル記述子の数は、csh で `limit` コマンドを使用して確認できます。

十分な権限があれば、csh で `ulimit` コマンドを使用してファイル記述子を増やすことができます。権限がない場合は、システム管理者に依頼して、プロセスで利用可能なファイル記述子を増やしてもらいます。

Q. WebLogic Server のメモリを増やすにはどのようにしますか。

A. WebLogic Server に対する Java ヒープ メモリの割り当てを増やします。最小値と最大値は両方とも同じサイズに設定する必要があります。次の例のように、**-ms32m** オプションを使用して WebLogic Server を起動すると、割り当てを増やすことができます。

```
$ java ...-ms32m -mx32m ...
```

これで 32 MB の Java ヒープ メモリが WebLogic Server に割り当てられるので、パフォーマンスが向上し、WebLogic Server ではより多くの同時接続を処理できます。この値は必要に応じて増やすことができます。

Q. ログ ファイルで Java.io 例外を引き起こす原因は何ですか。

A. ログ ファイルで以下のようなメッセージが示される場合があります。

（Windows NT）

```
java.io.IOException Connection Reset by Peer
java.io.EOFException Connection Reset by Peer
```

（Solaris）

```
java.io.Exception: Broken pipe
```

これらのメッセージは、サーブレットの使用時に発生します。クライアントは HTTP リクエストを開始し、ブラウザで以下の一連のアクションを実行します。

1. [Stop] をクリックするか、同等のコマンドを入力するか、または同等のキーを押します。
2. [Refresh] をクリックするか、同等のコマンドを入力するか、または同等のキーを押します。
3. 新しい HTTP リクエストを送信します。

上記のメッセージは、WebLogic Server が割り込まれた HTTP リクエストを検出して回復したことを示します。

Q. Java と CORBA の統合 : RMI-IIOP または Java IDL

A. Java と CORBA を統合するこれら 2 つの方法の違いを理解する必要があります。

RMI-IIOP は、RMI インタフェースをプログラミングするが、基盤の転送には IIOP を使用する Java プログラマが使用します。RMI-IIOP はさまざまな言語で実装された他の CORBA オブジェクトとの相互運用を実現しますが、それはすべてのリモート インタフェースが元々 Java RMI インタフェースとして定義されている場合に限られます。この方法は、エンタープライズ JavaBean（EJB）を利用するプログラマにとって特に有効です。なぜなら、EJB のリモート オブジェクト モデルが RMI に基づいているからです。

Java IDL は、CORBA IDL で定義されたインタフェースに基づく Java でプログラミングを行う CORBA プログラマが使用します。これは「通常どおり」の CORBA プログラミングであり、C++ や COBOL といった他の言語とまったく同じように Java がサポートされます。

Q. RMI-IIOP アプリケーションと既存の CORBA オブジェクトはどのように相互運用されるのですか。

A. 既存の CORBA オブジェクトのリモート インタフェースが元々 CORBA IDL で定義されている場合は、相互運用はできません。RMI-IIOP アプリケーションが他の CORBA オブジェクトと相互運用できるのは、それらのリモート インタフェースが元々 Java RMI インタフェースとして定義されている場合に限られます。

たとえば、RMI-IIOP クライアントと C++ オブジェクトの相互運用を実現するには、次のようにする必要があります。

1. オブジェクトのリモート インタフェースを Java で RMI インタフェースとして定義します。
2. インタフェースに対して rmic -iiop を実行し、RMI-IIOP クライアントのスタブを生成します。
3. インタフェースに対して rmic -idl を実行し、RMI インタフェースと互換の IDL を生成します。
4. IDL ファイルに対して C++ スタブ コンパイラを実行し、C++ サーバ オブジェクトの C++ スケルトンを生成します。

Q. WebLogic Server で T3 はどのように機能するのですか。

A. T3 は、省略形、およびオブジェクトの置換（WebLogic Server クラスタと HTTP トネリングおよび他の製品のトネリングのコンテキストで有効）のような機能などの、パフォーマンスの向上をサポートするメッセージのフレームワークを WebLogic Server で提供します。

T3 は Java オブジェクトの直列化および RMI より先に開発されたものですが、これらの仕様をしっかり学習して利用しています。T3 は Java Object Serialization または RMI のスーパーセットです。Java Object Serialization および RMI で実行できることは T3 でもすべて可能です。

T3 は WebLogic Server 間、およびプログラムに基づくクライアントと WebLogic Server クラスタ間では必須です。HTTP と IIOP はオプションのプロトコルであり、他のプロセスと WebLogic Server 間の通信に使用できます。これは何をするのかに依存し、たとえば以下のように使い分けます。

- ブラウザと WebLogic Server の通信には HTTP を使用します。
- ORB と WebLogic Server の通信には IIOP を使用します。

Q. WebLogic Server で実行されている Java コードをデバッグする方法を教えてください。

A. WebGain、JBuilder、NetBeans、JDB などのツールを使用できます。これらのツールは、Java Platform Debugger Architecture（JPDA）に基づいて WebLogic Server で実行されている Java コードをデバッグします。

JPDA は、すべてのプラットフォーム用の Java 2 Platform, Standard Edition（J2SE）SDK 1.3 と、Linux 用の SDK 1.2.2 に統合されています。Sun のサイトでは、Solaris と Microsoft Window プラットフォーム版の J2SE SDK 1.2.2 に JPDA サポートを追加するためのダウンロードを利用できます。これらのプラットフォームで J2SE SDK 1.2.2 を使用している場合、まずこのダウンロードを取得する必要があります。

WebLogic が動作する仮想マシンにデバッガを接続するには、WebLogic をデバッグ モードで起動する必要があります。Sun 仮想マシンを使用して WebLogic をデバッグ モードで起動するには、次の手順に従います（Solar プラットフォームを使用している場合は手順 1 から始めてください）。

1. Solaris プラットフォームを使用している場合は、`LD_LIBRARY_PATH` 環境変数の前に `$JAVA_HOME/lib/sparc` を付加します。

   ```
   export LD_LIBRARY_PATH=$JAVA_HOME/lib/sparc:$LD_LIBRARY_PATH
   ```

2. WebLogic サーバを起動する java コマンド ラインに次のパラメータを追加します（「weblogic.Server」という文字列の前）。

   ```
   -Xdebug
   -Xnoagent
   -Xrunjdwp:transport=dt_socket
   server=y
   address=<port_for_debugger_to_connect>
   suspend=n
   -Djava.compiler=NONE
   ```

Hotspot Performance エンジンを使用する場合、`-Xnoagent` および `-Djava.compiler=NONE` オプションは必要ありませんが、互換性を保持するため、正常に受け付けられて無視されます。

`server=y` パラメータが追加され、`address` パラメータが追加されない場合、WebLogic Server は転送アドレスを選択し、それを標準出力ストリームに出力します。たとえば、次のような行があるとします。

```
Listening for transport dt_socket at address: 46666
```

上の行では、サーバの起動時に標準出力ストリームに出力されます。46666 はポート番号です。このポート番号は、WebLogic の仮想マシンに接続するリモート デバッガに提供されます。

19 FAQ: サーバサイド Java（サーブレット）

- URL でパラメータを指定してサーブレットを呼び出すには、どうすれば良いですか。
- 同じ WebLogic Server インスタンスで同じサーブレット クラスの複数のインスタンスを実行する方法を教えてください。
- http セッションをデシリアライズするにはどうすればよいですか。

Q. URL でパラメータを指定してサーブレットを呼び出すには、どうすれば良いですか。

A. サーブレット パラメータの通常の書式は *name=value* で、URL の最後の疑問符（?）の後に記述します。これらのパラメータにアクセスするには、`HttpServletRequest` オブジェクトで `getParameter()` メソッドを呼び出し、文字列をテストするコードを記述します。たとえば、URL パラメータが「`func=topic`」の場合、URL は次のようになります。

```
http://www.myserver.com/myservlet?func=topic
```

この場合は、次のようにしてパラメータを解析できます。ここで、「`req`」は `HttpServletRequest` オブジェクトです。

```
String func = req.getParameter("func");
if (func.equalsIgnoreCase("topic")) {
  . . . do some work
}
```

Q. 同じ WebLogic Server インスタンスで同じサーブレット クラスの複数のインスタンスを実行する方法を教えてください。

A. 複数のインスタンスを実行する場合、サーブレットは `SingleThreadModel` インタフェースを実装する必要があります。`SingleThreadModel` インタフェースを実装したクラスのインスタンスは、同時に複数のスレッドで呼び出されないことが保証されています。`SingleThreadModel` インタフェースの複数のインスタンスを使用して、それぞれをシングル スレッドで実行しながら、同時に発生するリクエストを処理します。

サーブレットの設計時に、ファイルやデータベースへのアクセスのようなサーブレット クラスの外部の共有リソースの使い方に注意を払う必要があります。同一のサーブレット インスタンスが複数あり、まったく同じリソースを使用する可能性があるため、`SingleThreadModel` インタフェースを実装した場合でも、解決の必要がある同期と共有の問題が発生します。

Q. http セッションをデシリアライズするにはどうすればよいですか。

A. hppt セッションをデシリアライズするには、現在のスレッドの contextclassloader を使用してアプリケーション コンテキスト内のユーザ定義オブジェクトをロードするユーティリティ クラスを作成します。そして、このユーティリティ クラスを、システムの CLASSPATH に追加します。

20 FAQ: セキュリティ

- デモ用デジタル証明書および信頼性のある CA はどうしたら更新できますか。
- サーブレットが「no certificate」というメッセージを返すのはなぜですか。
- WebLogic では、Diffie-Hellman または DSS/DSA のデジタル証明書がサポートされていますか。
- サーバで、RSA 証明書と非 RSA 証明書を同時に使用することはできますか。
- 非 RSA クライアント コードで RSA ライセンス コストを支払う必要がありますか。
- WebLogic Server で Netscape セキュリティ証明書を使用するにはどうすれば良いですか。
- サーブレットおよび JSP へのアクセスを制限する方法を教えてください。
- RSA 暗号化アルゴリズムと javax.crypto.* API を使用してアプリケーションを構築できますか。
- JNDI 初期コンテキストを使用して、WebLogic Server ユーザのセキュリティ資格を渡すことができますか。
- LDAP セキュリティ レルムを使用するときに WebLogic Server を起動できないのはなぜですか。
- WebLogic Server パスワードは安全ですか。
- サーバを起動するときに「`<Alert> <WebLogicServer> <Security> configuration problem with certificate file`」エラーを受け取るのはなぜですか。
- デモ用証明書を使用しているときに発信 SSL 接続を確立できないのはなぜですか。
- WebLogic Server への SSL 接続を確立するときに、「`<WebLogic Server> <SSLListenThread listening on port 8802> Failed to`

`connect to t3s://localhost:8802`」というエラーを受け取るのはなぜですか。

- どうすれば、WL-Proxy-Client-Cert ヘッダを使用する偽装クライアントによるセキュリティ攻撃から WebLogic Server を守ることができますか。

Q. デモ用デジタル証明書および信頼性のある CA はどうしたら更新できますか。

A. デモ用証明書の有効期限が切れた場合、またはプロダクション環境に移行する場合は、Verisign や Entrust などの信頼できるベンダから独自の証明書と信頼性のある CA を購入することをお勧めします。独自の証明書を購入するには、「セキュリティの管理」で説明されている Certificate Request サーブレットを使用して証明書署名リクエスト (CSR) を生成します。新しい証明書と信頼性のある CA を入手したなら、デモ用の証明書と信頼性のある CA のファイルを削除し、新しいものと置き換える必要があります。また、Administration Console で SSL の属性をリセットする必要もあります。

Q. サーブレットが「no certificate」というメッセージを返すのはなぜですか。

A. このメッセージは、証明書キャプチャ機能を使用して 2 方向の認証を提供しようとした場合に発生します。トラブルシューティングでは、以下の事項を調べてください。

1. Web ブラウザにデジタル証明書が組み込まれていますか。
2. Administration Console の [サーバ] ウィンドウの [コンフィグレーション] タブにある [SSL] タブで [クライアント認証を強制する] オプションをチェックすることで、クライアント認証を要求するように WebLogic Server がコンフィグレーションされていますか。
3. \wlserver6.0\config\mydomain ディレクトリに、デジタル証明書がインストールされていますか。
4. 「SSL プロトコルのコンフィグレーション」で説明されているとおりに SSL 通信のポートがコンフィグレーションされていますか。
5. HTTP ではなく HTTPS を使用していますか。

Q. WebLogic では、Diffie-Hellman または DSS/DSA のデジタル証明書がサポートされていますか。

A. いいえ。WebLogic の輸出可能バージョンでは、40 ビット RC4 を使用した 512 ビット RSA のみがサポートされています。また、ブラウザではそのような証明書はサポートされていませんし、DSA 証明書を商用ベースで発行している会社もありません。

Q. サーバで、RSA 証明書と非 RSA 証明書を同時に使用することはできますか。

A. できません。

Q. 非 RSA クライアント コードで RSA ライセンス コストを支払う必要がありますか。

A. WebLogic では、WebLogic Server とクライアント間の SSL に対して RSA の使用を許可しています。WebLogic を使用する場合、RSA に関して他のライセンスは必要ありません。ただし、付加価値再販業者によって規約は異なります。

Q. WebLogic Server で Netscape セキュリティ証明書を使用するにはどうすれば良いですか。

A. Netscape では、プライベート キーと公開鍵が 1 つのファイルに格納され、公開鍵とプライベート キーの分離が防止されます。config.xml の `ServerKeyFileName` 属性は、プライベート キー ファイルのみを参照します。したがって、Netscape ユーティリティを使用しないで別の証明書要求を生成する必要があります。Certificate Request サーブレットを使用すると、新しい証明書に対する要求を生成できます。サーブレットの使い方については、「WebLogic SSL の使い方」を参照してください。

Q. サーブレットおよび JSP へのアクセスを制限する方法を教えてください。

A. Java Servlet API 仕様 v2.2 では、Web アプリケーションのデプロイメント記述子を使用して特定のサーブレットおよび JSP へのアクセスを制限できます。この仕様のセクション 13.3.2 には、宣言型のセキュリティを使用するサンプルのデプロイメント記述子があります。詳細については、『WebLogic HTTP サーブレット プログラマーズ ガイド』を参照してください。

Q. RSA 暗号化アルゴリズムと javax.crypto.* API を使用してアプリケーションを構築できますか。

A. できません。WebLogic の RSA ライセンスは、エンド ユーザが RSA クラスを直接使用することを許可していません。RSA から暗号化ライブラリのライセンスを独自に取得する必要があります。

Q. JNDI 初期コンテキストを使用して、WebLogic Server ユーザのセキュリティ資格を渡すことができますか。

A. できます。ただし、6.1 より後のリリースではできません。6.1 より後のリリースでは、ユーザをセキュリティ コンテキストに関連付ける場合、JNDI ではなく Java Authentication and Authorization Service（JAAS）を使用する必要があります。詳細については、『WebLogic Security プログラマーズ ガイド』を参照してください。

Q. LDAP セキュリティ レルムを使用するときに WebLogic Server を起動できないのはなぜですか。

A. WebLogic Server で代替セキュリティ レルムまたはカスタム セキュリティ レルムを使用する場合、キャッシング レルムをコンフィグレーションおよび有効化する必要があります。

Microsoft Site Server の LDAP サーバがインストールされ、LDAP ディレクトリのルートが作成されると、デフォルトによっていくつかの組織単位が作成されます。Groups の下には、NTGroups というデフォルトの組織があり、この組織には Administrators という空のデフォルト グループがあります。デフォルトでは、WebLogic Server も Administrators というグループを提供します。このグループには、WebLogic Server を起動するユーザである System というメンバーが含まれています。Microsoft Site Server のデフォルトを使用し、デフォルト組織単位の下に独自のグループを作成した場合、WebLogic Server は起動しません。独自の組織単位を LDAP ディレクトリに作成し、その組織単位の下に独自のグループを作成する必要があります。

LDAP ディレクトリに同じ名前のグループが 2 つある場合、WebLogic Server はそのグループのユーザを適切に認証できません。LDAP セキュリティ レルムは、グループの DN（識別名）を使用して LDAP ディレクトリのグループを検索します。複数のグループを同じ名前で作成する場合、WebLogic Server は最初に発見したグループのユーザだけを認証します。LDAP セキュリティ レルムを使用するときは、ユニークなグループ名を使用する必要があります。

LDAP レルム V2 は、LDAP レルム V1 で提供されていた以下の機能を提供しません。

- すべてのユーザのリスン
- グループのメンバーのリスン
- LDAP サーバを認証するための AuthProtocol とユーザ認証メカニズム

LDAP セキュリティ レルムは、そのセキュリティ レルムで使用される LDAP ディレクトリのどこにユーザとグループが格納されているのかを知る必要があります。そのためには、ユーザとグループが存在する LDAP ディレクトリの識別名（DN）を指定します。

LDAP では、DN はリーフ ノードから始まり、ルート ノードに向かいます。次の図に、LDAP ディレクトリのブランチを示します。

```
o=acme.com
   |
   |
ou=Groups
```

このブランチの DN は、`ou=Groups, o=acme.com` として指定されます。

LDAP レルム V1 では、DN は LDAPRealm MBean の GroupDN 属性と UserDN 属性を介して、または Administration Console を使用して指定します。ただし、DN を反対方向に指定する必要があります。したがって、上の例の DN は次のように指定します。

```
groupDN="o=acme.com, ou=Groups"
```

LDAP レルム V2 では、DN の指定は、`user.dn` プロパティと `group.dn` プロパティを CustomRealm MBean の Configuration 属性に追加するか、Administration Console を使用して行います。LDAP レルム V1 とは異なり、DN を反対方向に指定する必要はありません。たとえば、LDAP レルム V2 の `user.dn` プロパティと `group.dn` プロパティは次のようになります。

```
ConfigurationData="..., group.dn=ou=Groups, o=acme.com,..."
```

つまり、LDAP レルム V1 では反対方向の DN が必要で、LDAP レルム V2 では通常の DN が必要となります。

LDAP レルム V1 と LDAP レルム V2 を切り替えるときによく犯すエラーは、反対方向の DN をコピーしてしまい、LDAP レルムが動作を停止してしまうことです。LDAP レルム V1 から LDAP レルム V2 に移行するときには、DN の仕様をチェックしてください。

Q. WebLogic Server パスワードは安全ですか。

A. `config.xml` ファイルには、クリア テキスト形式のパスワードが存在しなくなりました。クリア テキスト形式のパスワードに代わって、`config.xml` ファイルには暗号化およびハッシュ化されたパスワードが格納されます。暗号化パスワードは、別のドメインにコピーできません。代わりに、`config.xml` ファイルを編集して、既存の暗号化、ハッシュ化パスワードをクリア テキスト パスワードに置き換えてから、そのファイルを新しいドメインにコピーします。Administration Console は、次にそのファイルに書き込むときにパスワードを暗号化およびハッシュ化します。

レコーディング セキュリティ マネージャ ユーティリティを使用すると、WebLogic Server の起動時または動作中に発生するパーミッションの問題を検出できます。このユーティリティで出力されるパーミッションを `config.xml` また

は Java セキュリティ ポリシー ファイルに追加して、発見されたパーミッションの問題を解決できます。レコーディング セキュリティ マネージャは、dev2dev オンラインで入手できます。

Q. サーバを起動するときに「`<Alert> <WebLogicServer> <Security> configuration problem with certificate file`」エラーを受け取るのはなぜですか。

A. SSL コンフィグレーション ファイルで「WL_HOME」相対ファイル名を指定しなかった可能性があります。WL_HOME の詳細については、『管理者ガイド』の「セキュリティの管理」の「SSL プロトコルのコンフィグレーション」を参照してください。

Q. デモ用証明書を使用しているときに発信 SSL 接続を確立できないのはなぜですか。

A. SSL 接続を確立するときには、デジタル証明書の主体の DN が、SSL 接続を開始するサーバのホスト名と一致している必要があります。一致していないと、SSL 接続が中断されます。デモ用証明書を使用していると、このホスト名が一致しません。この状況を回避するには、WebLogic Server 起動時に `-Dweblogic.security.SSL.ignoreHostnameVerification=true` フラグを使用します。このフラグにより、主体の DN とホスト名を比較するホスト名検証が無効化されます。この解決策が推奨されるのは、開発環境においてのみです。さらにセキュアな解決策は、発信 SSL 接続を行うサーバ用に新しいデジタル証明書を取得することです。

Q. WebLogic Server への SSL 接続を確立するときに、「`<WebLogic Server> <SSLListenThread listening on port 8802> Failed to connect to t3s://localhost:8802`」というエラーを受け取るのはなぜですか。

A. デフォルトでは、WebLogic Server にはデジタル証明書の主体の DN とホスト名を比較するホスト名検証が含まれています。SSL 接続を確立するときには、デジタル証明書の主体の DN が、SSL 接続を開始するサーバのホスト名と一致している必要があります。デモ用証明書を使用していると、このホスト名が一致しません。この状況を回避するには、WebLogic Server 起動時に `-Dweblogic.security.SSL.ignoreHostnameVerification=true` フラグを使用します。このフラグにより、ホスト名検証が無効化されます。この解決策が推奨されるのは、開発環境においてのみです。さらにセキュアな解決策は、WebLogic クライアント用に新しいデジタル証明書を取得することです。

Q. どうすれば、WL-Proxy-Client-Cert ヘッダを使用する偽装クライアントによるセキュリティ攻撃から WebLogic Server を守ることができますか。

A. WebLogic Server に直接アクセスできるクライアントはすべて、WL-Proxy-Client-Cert ヘッダを偽装（使用）できます。WebLogic Server は、こ

のヘッダから証明書情報を取得し、それが安全な送信元（プラグイン）からのものであることを信頼して、ユーザを認証するためにその情報を使用します。WebLogic Server の以前のリリースでは、このヘッダを常に信頼するのが、デフォルトの動作でした。現在では、WL-Proxy-Client-Cert ヘッダの信頼を明示的に定義する必要があります。新しいパラメータである `clientCertProxy` を使用すると、証明書ヘッダの暗黙の信頼を有効にすることができます。セキュリティのレベルを追加する必要がある場合は、接続フィルタを使って、WebLogic Server に対するすべての接続を制限します（つまり、プラグインが動作しているマシンからの接続だけを WebLogic Server が受け付けるようにします）。

`clientCertProxy` パラメータは、HTTPClusterServlet と Web アプリケーションに追加されています。

HTTPClusterServlet の場合は、次のようにして `web.xml` ファイルにパラメータを追加します。

```
<context-param>
    <param-name>clientCertProxy</param-name>
    <param-value>true</param-value>
</context-param>
```

Web アプリケーションの場合は、次のようにして `web.xml` ファイルにパラメータを追加します。

```
ServletRequestImpl context-param
<context-param>
    <param-name>weblogic.http.clientCertProxy</param-name>
    <param-value>true</param-value>
</context-param>
```

次のようにすれば、クラスタでこのパラメータを使用することもできます。

```
<Cluster ClusterAddress="127.0.0.1" Name="MyCluster"
    ClientCertProxyHeader="true"/>
```

21 FAQ: Web サービス

- WebLogic Server 6.1 で 添付ファイル付き SOAP メッセージはサポートされていますか。
- WebLogic Server 6.1 で SOAP はサポートされていますか。
- WebLogic Server 6.1 では Microsoft SOAP ツールキットおよび .NET はサポートされていますか。
- WebLogic Server 6.0 コンポーネントで、WebLogic Web サービス クライアント API のバージョン 6.1 を使用できますか。

Q. WebLogic Server 6.1 で 添付ファイル付き SOAP メッセージはサポートされていますか。

A. WebLogic Server 6.1 は添付ファイル付き SOAP メッセージを受け取ることができますが、現在の実装では添付ファイルの処理を行いません。つまり、WebLogic Server は添付ファイル付き SOAP メッセージを受け取り、そのメッセージは解析され処理されますが、添付ファイルは無視されるということです。将来のリリースでは、添付ファイルを完全にサポートする予定です。

Q. WebLogic Server 6.1 で SOAP はサポートされていますか。

A. はい。WebLogic Server の SOAP の実装は、Web サービス サブシステムの一部として組み込まれています。WebLogic Server を使用した Web サービスの作成については、『WebLogic Web サービス プログラマーズ ガイド』を参照してください。

Q. WebLogic Server 6.1 では Microsoft SOAP ツールキットおよび .NET はサポートされていますか。

A. WLS 6.1 の開発時には、Microsoft SOAP ツールキットと .NET ベータ版の間に互換性がありませんでした。Microsoft によって正式にサポートされている SOAP 実装が MS SOAP ツールキットのみだったため、WLS 6.1 では MS SOAP ツールキットのみをサポートしています。.NET は将来のリリースでサポート予定です。

Q. WebLogic Server 6.0 コンポーネントで、WebLogic Web サービス クライアント API のバージョン 6.1 を使用できますか。

A. EJB などの WebLogic Server 6.0 コンポーネントで WLS Web サービス クライアント API（バージョン 6.1）を使用すると、`java.rmi.ConnectException: No available router to destination` というエラーが発生する。この原因は、クライアント コードで指定している初期コンテキスト ファクトリではなく、6.0 JNDI URL ファクトリが呼び出されることにあります。

この問題を避けるには、初期コンテキストの作成時に次のコードを追加することによって 6.0 JNDI URL ファクトリを削除します。

```
h.put(Context.URL_PKG_PREFIXES, "");
```

Q. 相互運用性の問題として、6.1 WebLogic Web サービスで注意すべきことは何ですか。

A. 6.1 WebLogic Web サービスは、Round 2 SOAP 相互運用テストのほとんどに合格しています。相互運用上の特定の問題についての詳細は、「相互運用性」を参照してください。

22 FAQ: 無線関連の質問

- 無線（モバイル）デバイスとは何ですか。
- WebLogic Server は無線デバイスをサポートしていますか。
- 無線デバイス用のアプリケーションを記述するときに考慮すべきことは何ですか。
- WAP とは何ですか、また i-Mode とは何ですか。
- 自分のソリューションは、CDMA、GPRS、TDMA、PDC-P などの異なるネットワークで機能しますか。
- 3G 無線ネットワークに対応するためには何を変更する必要がありますか。

Q. 無線（モバイル）デバイスとは何ですか。

A. この文脈での無線デバイスとは、回線でネットワークに物理的に接続することなくインターネットへの接続を提供するデバイスのことです。これらのデバイスの最も一般的な例は、インターネット対応の携帯電話（WAP Phone や i-Mode 電話など）、個人用携帯型情報端末（PDA）（Palm VII など）、Pocket PC（Wireless iPaq など）、ページャ（RIM Blackberry など）です。

Q. WebLogic Server は無線デバイスをサポートしていますか。

A. はい。無線サポートについては、『WebLogic Server Wireless Application 開発プログラマーズ ガイド』を参照してください。無線の例は、WebLogicServer の \samples\examples ディレクトリに格納されており（インストールされている場合）、[スタート] メニューからアクセスできます。

Q. 無線デバイス用のアプリケーションを記述するときに考慮すべきことは何ですか。

A. 無線クライアント用のアプリケーションを記述するときに考慮すべき要素は以下のとおりです。

- デバイスの位置と種類に基づいてコンテンツをパーソナライズする必要があります。

- デバイス マイクロブラウザは、通常 HTML ベースではありません。デバイスによって、WML ベース、cHTML ベース、HDML ベース、Web Clipping ベースとさまざまです。
- デバイスの中には、Bluetooth、電源キー、SMS メッセージングなど、アプリケーションの拡張に使用できる追加機能が搭載されているものもあります。
- 多くの場合、こうしたデバイスは、音声認識およびテキスト スピーチ機能を使用する音声ポータルで使用できます。
- これらのデバイスの大部分は画面が小さく、色またはグラフィック イメージが表示されないものもあります。
- これらの画面の形式は、縦長の長方形から、正方形、横長の長方形まで多岐にわたります。
- これらのデバイスの多くでは、数値キーバッドまたはペンでデータの入力と選択を行いますが、操作が困難です。
- デバイスがモバイルのときには、多くの場合、接続性が失われます。
- 一般に PKI セキュリティ機能が搭載されていません。
- いくつかのデバイスでは、呼び出しのためにそれらに転送可能なデータの量が制限されます。

詳細については、「Wireless, Internet and Email」を参照してください。

Q. WAP とは何ですか、また i-Mode とは何ですか。

A. WAP と i-Mode は、携帯電話やその他のデバイスとの無線インターネット通信向けの 2 種類の主要な無線（OTA）プロトコルです。WAP は Wireless Application Protocol の略語で、ヨーロッパおよび北アメリカで主流になっています。i-Mode は NTT DoCoMo によって開発され、日本で使用されています。現在、WAP-NG（WAP Next Generation）と呼ばれる新しいプロトコルが、これらのプロトコルに取って代わるものと見なされています。

WAP と i-Mode は、どちらもそれぞれのマイクロブラウザで認識可能な OTA プロトコルとマークアップ言語で構成されています。WAP マークアップ言語は WML（Wireless Markup Language）で、cHTML（Compact HTML）は i-Mode で指定されているマークアップ言語です。WML と cHTML は、将来 XHTML（Basic）に取って代わられる可能性があります。

他の無線キャリアおよびデバイスは、WAP と i-Mode 以外のプロトコルとマークアップ言語を使用していることに注意してください。たとえば、Palm VII は独自のプロトコルの上に Web クリッピングを採用してます。

WAP の詳細については、http://www.wapforum.org/faqs を参照してください。i-Mode の詳細については、http://www.nttdocomo.com/i/qa.html を参照してください。

Q. 自分のソリューションは、CDMA、GPRS、TDMA、PDC-P などの異なるネットワークで機能しますか。

A. はい。WAP と i-Mode は、アプリケーション開発者からネットワークの詳細を隠すように設計されています。これらは、基盤ネットワーク上で同じように機能します。このため、キャリアが自社のネットワークをアップグレードする場合でも、WAP または i-Mode のいずれかに対応して記述されたアプリケーションは修正を加えなくても引き続き正常に機能します。ネットワークを高速化した場合、WML または cHTML のいずれかに対応して記述されたアプリケーションのパフォーマンスも同様に向上します。

Q. 3G 無線ネットワークに対応するためには何を変更する必要がありますか。

A. 何もありません。前の回答で説明したとおり、WAP と i-Mode はどちらも基盤ネットワークに依存していません。ただし、開発者は、より広いバンド幅を必要とするストリーミング メディアなどのコンテンツに対応するようアプリケーションを強化することを検討する必要があります。

23 FAQ: XML

- WebLogic Server 6.1 にはどの XML パーサが付属していますか。
- WebLogic Server で XSLT プロセッサは用意されているのですか。
- WebLogic Server 6.1 に実装されている JAXP API 仕様のバージョンは何ですか。
- XML ドキュメントの解析に、バージョン 2.2 の Java Servlet API の `getAttribute()` メソッドと `setAttribute()` メソッドを使用できますか。
- WebLogic Serve 6.1 の組み込みパーサ（Xerces 1.3.1）とは異なる Apache の Xerces XML パーサのバージョンをプラグインできますか。
- Apache Web サイトから Apache Xalan のバージョンをダウンロードしてプラグインしましたが、ドキュメントを変換しようとするとエラーが発生します。何が問題なのでしょうか。
- XML ドキュメントの文書型を識別するにはどのようにすればよいですか。

Q. WebLogic Server 6.1 にはどの XML パーサが付属していますか。

A. WebLogic Server 6.1 には、Apache の Xerces 1.3.1 パーサをベースとしたパーサが付属しています。さらに、小中規模の XML ドキュメントに使用できる WebLogic 独自のハイパフォーマンス、非検証パーサも付属しています。WebLogic XML Registry を使用すると、特定の文書型に使用するパーサをコンフィグレーションできます。

Q. WebLogic Server で XSLT プロセッサは用意されているのですか。

A. はい。WebLogic Server 6.1 には、Apache の Xalan 2.0.1 プロセッサをベースとした XSLT プロセッサが含まれています。

Q. WebLogic Server 6.1 に実装されている JAXP API 仕様のバージョンは何ですか。

A. バージョン 1.1 です。このバージョンには、プラグイン可能な XML 変換とプラグイン可能な XML 解析が組み込まれています。

Q. XML ドキュメントの解析に、バージョン 2.2 の Java Servlet API の `getAttribute()` メソッドと `setAttribute()` メソッドを使用できますか。

A. はい。SAX モード解析には `setAttribute()` メソッドを使用し、DOM モード解析には `getAttribute()` メソッドを使用します。ただし、サーブレットでこれらのメソッドを使用するのは、WebLogic 固有の機能です。つまり、そのサーブレットは他のサーブレット エンジンには移植できません。したがって、この機能は慎重に使用してください。

Q. WebLogic Serve 6.1 の組み込みパーサ（Xerces 1.3.1）とは異なる Apache の Xerces XML パーサのバージョンをプラグインできますか。

A. できます。プラグインできる Xerces のバージョンは、WebLogic Server 6.1 のどのサービス パックがインストールされているかによって異なります。

WebLogic Server 6.1 サービス パック 2 またはそれ以前がコンピュータにインストールされている場合は、次の手順に従ってください。

次のバージョンの Xerces をプラグインできます。

- 1.2.2
- 1.2.3
- 1.3.0
- 1.3.1
- 1.4.0
- 2.0.0
- 2.0.1

警告： バージョン 2.0.0 またはそれ以降の Xerces をプラグインする場合、Xalan 互換性クラスは使用できません。

Apache Xerces パーサの異なるバージョンをプラグインするには、次の手順に従って *`WL_HOME`*`/lib/xmlx.jar` ファイルの内容を置き換える必要があります。

a. `org/apache` ディレクトリ以下のすべてのファイルを `JAR` ファイル（*`WL_HOME`*`/lib/xmlx.jar`）から削除します。*`WL_HOME`* は WebLogic Server のインストール ディレクトリです。

b. （Apache Web サイトからダウンロードした）`xerces.jar` ファイルの `org/apache` ディレクトリ以下のすべてのファイルを `xmlx.jar` ファイルにコピーします。

別の方法 : WebLogic Server の CLASSPATH の末尾に `xerces.jar` ファイルを追加します。

別の方法 : アプリケーションの `META-INF/lib` ディレクトリに `xerces.jar` ファイルを配置します。この場合、Web アプリケーションの `PreferWebInf Classes` フラグは必ず無効にしてください。

c. Administration Console を使用して、パーサおよび `org.apache.xerces.jaxp.*` 内のビルダ ファクトリを使用するように XML レジストリをコンフィグレーションします。

WebLogic Server 6.1 サービス パック 3 がコンピュータにインストールされている場合、次の手順に従います。

すべてのバージョンの Xerces XML パーサをプラグインできます。

Xerces パーサの異なるバージョンをプラグインするには、次の手順に従います。

a. アプリケーションの `WEB-INF/lib` ディレクトリに `xerces.jar` ファイルを配置します。Web アプリケーションの `PreferWebInf Classes` フラグは必ず有効にしてください。

b. xerces.jar ファイル内の META-INF/services ディレクトリ配下にパーサおよびビルダファクトリの指定ファイルが存在していない場合、WEB-INF/lib 内のいずれかのアーカイブ内の META-INF/services ディレクトリ配下に設定します。

警告 : この手順を使用して新しいバージョンの Xerces をプラグインする場合、パーサまたはエンティティの解決をコンフィグレーションする目的で、Administration Console を使用して XML レジストリをコンフィグレーションすることはできません。XML レジストリをコンフィグレーションする必要がある場合、前に説明した、6.1 サービス パック 2 またはそれ以前のバージョンに新しいパーサをプラグインするための手順に従ってください。

Q. Apache Web サイトから Apache Xalan のバージョンをダウンロードしてプラグインしましたが、ドキュメントを変換しようとするとエラーが発生します。何が問題なのでしょうか。

A. プラグインした Apache Xalan のバージョンが、使用中の Apache Xerces のバージョン（組み込みパーサを使用している場合はバージョン 1.3.1）と互換性を備えていることを確認する必要があります。

Q. XML ドキュメントの文書型を識別するにはどのようにすればよいですか。

A. XML ドキュメントにパブリック ID が割り当てられている場合、これがその文書型です。たとえば、XML ドキュメントに次の DOCTYPE 宣言が記述されているとします。

```
<!DOCTYPE mydoc PUBLIC "My public ID String"
                "http://foo.com/url/to/my/dtd">
```

この場合、その文書型は `My public ID String` です。

DOCTYPE 宣言にパブリック ID が設定されていないが、システム ID が指定されている場合、文書型はそのシステム ID です。たとえば、次の DOCTYPE 宣言があるとします。

```
<!DOCTYPE mydoc SYSTEM "http://foo.com/url/to/my/dtd">
```

この場合、文書型は `http://foo.com/url/to/my/dtd` です。

注意： システム ID は DTD のものであり、XML ドキュメント自体のものではありません。しかし、これは XML ドキュメントの識別手段として使用されます。

XML ドキュメントが DOCTYPE 宣言を指定しない場合、文書型はルート要素名またはネームスペース URL（XML ドキュメントにそれが設定されている場合）のいずれかです。