BEA WebLogic Server™

BEA WebLogic Server 7.0 へのアップグレード

BEA WebLogic Server バージョン 7.0
マニュアルの日付 :2002 年 6 月
改訂 : 2002 年 6 月 28 日

著作権

限定的権利条項

商標または登録商標

BEA WebLogic Server 7.0 へのアップグレード

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	2002 年 6 月 28 日	BEA WebLogic Server バージョン 7.0

目次

このマニュアルの内容

対象読者 ix
e-docs Web サイト x
このマニュアルの印刷方法 x
サポート情報 x
表記規則 xi

1. WebLogic Server 6.x からバージョン 7.0 へのアップグレード

WebLogic Server コンフィグレーションのアップグレード : 主な手順 1-2
起動スクリプトの修正 1-3
WebLogic Server 7.0 のディレクトリ構造について 1-4
WebLogic Server 6.x から WebLogic Server 7.0 へのアプリケーションの移植 1-5
セキュリティのアップグレード 1-6
互換性セキュリティでの WebLogic Server の起動 1-7
MBean の ACL 1-8
互換性セキュリティから WebLogic Server 7.0 セキュリティへのアップグレード 1-8
セキュリティ レルム 1-10
ゲスト ユーザ 1-12
password.ini ファイル 1-12
SSL プロトコルのアップグレード 1-13
信頼性のある CA キーストアの作成 1-13
互換性セキュリティでの CertAuthenticator の使用 1-15
暗号スイート 1-16
WebLogic Tuxedo Connector のアップグレード 1-16
WebLogic Tuxedo Connector の起動 1-16
WebLogic Tuxedo Connector の XML コンフィグレーション ファイルの変換 1-17
インバウンド RMI-IIOP アプリケーションの更新 1-19

リモート ユーザの認証 1-20
ACL ポリシーが LOCAL の場合 1-20
ACL ポリシーが GLOBAL の場合 1-21
WebLogic Tuxedo Connector のプロパティの設定 1-21
その他のアップグレード手順と情報 1-22
ant.jar 1-23
Apache Xalan XML トランスフォーマ 1-23
Apache Xerces XML パーサ 1-24
applications ディレクトリ 1-24
デプロイメント 1-25
EJB 2.0 1-26
weblogic.management.configuration.EJBComponentMBean の変更点 1-27
max-beans-in-cache パラメータ 1-28
完全修飾パス式 1-28
jCOM 1-28
JDBC 1-29
JMS 1-29
JMX 1-29
Jolt Java Client 1-30
JSP 1-30
管理対象サーバ 1-30
MBean API の変更 1-31
Security 1-31
guest および < 匿名 > ユーザ 1-31
サーブレット 1-32
スレッド プール サイズ 1-32
Web アプリケーション 1-33
Solaris での WebLogic Server クラスタ 1-34
Web サービス 1-34
書き込み可能な config.xml ファイル 1-35
非推奨になった API と機能 1-35
削除された API と機能 1-37

2. WebLogic Server 4.5 および 5.1 からバージョン 7.0 への

アップグレード

WebLogic Server コンフィグレーションのアップグレード : 主な手順.....2-2
WebLogic Server ライセンス ファイルのアップグレード.....2-4
ライセンス・アップグレードに際してのご注意.....2-4
WebLogicLicense.class ライセンスの変換.....2-5
WebLogicLicense.XML ライセンスの変換.....2-5
weblogic.properties ファイルから XML ファイルへの変換.....2-6
WebLogic Server 7.0 でのクラスのロード.....2-9
起動スクリプトの修正.....2-9
WebLogic Server 7.0 の J2EE アプリケーション タイプ.....2-10
既存のアプリケーションの Web アプリケーションへの変換と移植.....2-10
Web アプリケーションのディレクトリ構造.....2-11
XML デプロイメント記述子.....2-12
WAR ファイル.....2-13
Web アプリケーションのデプロイメント.....2-14
セッションの移植.....2-15
JavaServer Pages (JSP) とサーブレット.....2-15
WebLogic Server 5.1 から WebLogic Server 7.0 への単純なサーブレットの移植.....2-16
エンタープライズ JavaBean アプリケーションの移植と変換.....2-18
EJB の移植に関する考慮事項.....2-19
EJB の移植に関する推奨事項.....2-20
1.0 EJB を WebLogic Server 4.5.x から WebLogic Server 7.0 に移植する手順.....2-22
1.1 EJB を WebLogic Server 5.1 から WebLogic Server 7.0 に移植する手順 2-24
EJB 1.1 を EJB 2.0 に変換する手順.....2-25
その他の J2EE アプリケーション サーバからの EJB の移植.....2-26
エンタープライズ アプリケーションの作成.....2-26
J2EE クライアント アプリケーションについて.....2-27
JMS のアップグレード.....2-28
Oracle のアップグレード.....2-29
移植とデプロイメントに関するその他の考慮事項.....2-29
アプリケーションと管理対象サーバ.....2-30
デプロイメント.....2-31

プラグイン 2-31
FileServlet 2-31
インターナショナライゼーション (I18N) 2-32
Java Transaction API (JTA) 2-32
Java Database Connectivity (JDBC) 2-33
JSP 2-33
エラー処理 2-33
null の属性 2-33
JVM 2-34
RMI 2-34
セキュリティ 2-35
新しいセキュリティ アーキテクチャへのアップグレード 2-35
証明書サーブレットによって生成されるデジタル証明書 2-36
プライベート キーとデジタル証明書 2-36
セッションの移植 2-37
スタンドアロンの HTML と JSP 2-37
Web コンポーネント 2-38
WAP アプリケーション 2-39
書き込み可能な config.xml ファイル 2-40
XML 7.0 パーサおよびトランスフォーマ 2-40
非推奨となった API と機能 2-41
削除された API と機能 2-42

A. weblogic.properties のマッピング表

B. Pet Store アプリケーションおよびサンプル サーバのアップグレード

このドキュメントで使用する用語 B-1
Pet Store アプリケーションの WebLogic 6.1 サービス パック 3 から WebLogic Server 7.0 へのアップグレード B-2
WebLogic Server 7.0 をインストールする B-2
6.1 サービス パック 3 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する B-3
JSP 解析エラーの修正 B-4
Pet Store の再構築 B-6
WebLogic Server 6.1 サービス パック 3 の起動に使用する

startPetstore.cmd スクリプト B-7
WebLogic Server 7.0 を起動するために変更された上記の startPetstore.cmd スクリプト B-9
WebLogic Server 6.1 サービス パック 3 で使用する config.xml ファイル B-12
WebLogic Server 7.0 で使用する上記の config.xml ファイル B-15
WebLogic Server 7.0 で Pet Store アプリケーションを起動する B-19
WebLogic 6.0 サービス パック 2 サンプル サーバの WebLogic Server 7.0 へのアップグレード B-19
WebLogic Server 7.0 をインストールする B-20
6.0 サービス パック 2 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する B-20
WebLogic 6.0 サービス パック 2 サンプル サーバの起動に使用する setExamplesEnv.cmd スクリプト B-22
WebLogic 7.0 サンプル サーバを起動するために変更された上記の setExamplesEnv.cmd スクリプト B-23
WebLogic 6.0 サービス パック 2 サンプル サーバの起動に使用する startExamplesServer.cmd スクリプト B-25
WebLogic 7.0 サンプル サーバを起動するために変更された上記の startExamplesServer.cmd スクリプト B-27
WebLogic Server 7.0 でサンプル サーバを起動する B-29
WebLogic 6.1 サービス パック 2 サンプル サーバの WebLogic Server 7.0 へのアップグレード B-30
WebLogic Server 7.0 をインストールする B-30
6.1 サービス パック 2 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する B-30
WebLogic 6.1 サービス パック 2 サンプル サーバの起動に使用する setExamplesEnv.cmd スクリプト B-32
WebLogic Server 7.0 を起動するために変更された上記の setExamplesEnv.cmd スクリプト B-34
WebLogic 6.1 サービス パック 2 サンプル サーバの起動に使用する startExamplesServer.cmd スクリプト B-36
WebLogic Server 7.0 を起動するために変更された上記の startExamplesServer.cmd スクリプト B-38
WebLogic Server 7.0 でサンプル サーバを起動する B-40

このマニュアルの内容

このマニュアルでは、以前のバージョンの BEA WebLogic Server を WebLogic Server 7.0 にアップグレードするために必要な手順およびその他の情報を示します。また、以前のバージョンから WebLogic Server 7.0 へのアプリケーションの移行についての情報も示します。

このマニュアルの構成は次のとおりです。

- 第 1 章「WebLogic Server 6.x からバージョン 7.0 へのアップグレード」では、WebLogic Server 6.x から WebLogic Server 7.0 にアップグレードする方法について説明します。
- 第 2 章「WebLogic Server 4.5 および 5.1 からバージョン 7.0 へのアップグレード」では、WebLogic Server 4.5 または 5.1 から WebLogic Server 7.0 にアップグレードする方法について説明します。
- 付録 A「weblogic.properties のマッピング表」では、前のバージョンの `weblogic.properties` ファイル内のプロパティと、現在の `config.xml`、`web.xml`、または `weblogic.xml` ファイル内の属性との間の対応関係を示します。

対象読者

このマニュアルは、WebLogic Server 7.0 へのアップグレードを行う予定の、WebLogic Server 4.5、5.1、6.0、および 6.1 のすべてのユーザを対象としています。

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を一度に 1 章ずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。PDF 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) で無料で入手できます。

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@beasys.com までお送りください。寄せられた意見については、WebLogic Server の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の名前とバージョン、およびドキュメントのタイトルと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〕	複数のキーを同時に押すことを示す。
斜体	強調または書籍のタイトルを示す。

表記法	適用

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、データ構造体とそのメンバー、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;`
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
{ }	構文の中で複数の選択肢を示す。
[ ]	構文の中で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}`

表記法	適用
. . .	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 WebLogic Server 6.x からバージョン 7.0 へのアップグレード

WebLogic Server 6.x からバージョン 7.0 へのアップグレードは、最も簡単な状況では、WebLogic Server 起動コマンド スクリプトと環境設定を変更するだけで完了します。場合によっては、ドメイン ディレクトリの移動が必要になります。また非常にまれですが、アップグレード対象のサブシステムに固有の変更が必要になることがあります。

以下の節では、システムを WebLogic Server 6.x から WebLogic Server 7.0 にアップグレードするために必要な情報を示します。

- 1-2 ページの「WebLogic Server コンフィグレーションのアップグレード : 主な手順」
- 1-3 ページの「起動スクリプトの修正」
- 1-4 ページの「WebLogic Server 7.0 のディレクトリ構造について」
- 1-5 ページの「WebLogic Server 6.x から WebLogic Server 7.0 へのアプリケーションの移植」
- 1-6 ページの「セキュリティのアップグレード」
- 1-16 ページの「WebLogic Tuxedo Connector のアップグレード」
- 1-22 ページの「その他のアップグレード手順と情報」

Pet Store アプリケーションを WebLogic Server 6.1 から WebLogic Server 7.0 にアップグレードする方法、また、WebLogic 6.0 および 6.1 のサンプル サーバを WebLogic Server 7.0 にアップグレードする方法については、「Pet Store アプリケーションおよびサンプル サーバのアップグレード」を参照してください。

WebLogic Platform 7.0 (7.0.0.1) へアップグレードする方法については、『WebLogic Server FAQ 集』の「アップグレード」を参照してください。

注意： このマニュアル全体を通して、「アップグレード」という用語は WebLogic Server のより新しいバージョンへのアップグレードを指し、「移植」は、アプリケーションを WebLogic Server の以前のバージョンからより新しいバージョンに移動することを指します。

WebLogic Server コンフィグレーションのアップグレード：主な手順

WebLogic Server 6.x から WebLogic Server 7.0 にアップグレードするには、次の手順に従います。

1. アップグレード作業を開始する前に、バージョン 6.x のドメインのバックアップ コピーを作成します。WebLogic Server 7.0 のクラスを使用してサーバを起動した後で、6.x にダウングレードすることはできません。
2. WebLogic Server 7.0 をインストールします。『インストール ガイド』を参照してください。

   注意： インストーラでは、新しいバージョンを古いバージョンの上に直接インストールすることはできません。新しいディレクトリの場所を選択する必要があります。
3. WebLogic Server 7.0 で動作するようにバージョン 6.x の起動スクリプトを修正します。1-3 ページの「起動スクリプトの修正」を参照してください。
4. 必ず、WebLogic Server 7.0 でのディレクトリ構造の変更点について考慮してください。起動の前にファイルの場所を変更しなければならない場合があります。1-4 ページの「WebLogic Server 7.0 のディレクトリ構造について」を参照してください。
5. アプリケーションを WebLogic Server 7.0 に移植します。1-5 ページの「WebLogic Server 6.x から WebLogic Server 7.0 へのアプリケーションの移植」を参照してください。
6. 必要に応じて、1-6 ページの「セキュリティのアップグレード」、1-16 ページの「WebLogic Tuxedo Connector のアップグレード」、および 1-22 ページの「その他のアップグレード手順と情報」で説明する他のアップグレード手順を実行します。

サーバのクラスタをアップグレードするには、それぞれのサーバについて上記の手順に従い、その後、『WebLogic Server クラスタ ユーザーズ ガイド』の「WebLogic クラスタのセットアップ」で説明されている手順に従います。RMI/T3 または RMI/IIOP を使用してアプリケーションを起動する場合、WebLogic Server 6.1 と 7.0 は相互運用が可能です。ただし、1 つのドメインの内部ではすべてのサーバが同じバージョンでなければなりません。

WebLogic Server ライセンス ファイルのアップグレードについては、『インストール ガイド』の「ライセンス アップグレードに際してのご注意」を参照してください。

起動スクリプトの修正

以前のバージョンで WebLogic Server 起動スクリプトを使用していた場合は、WebLogic Server 7.0 で利用できるようにスクリプトを変更する必要があります。

ここでの説明に従って、起動スクリプトを修正します。起動スクリプトの修正方法についてのもう 1 つの例は、「Pet Store アプリケーションおよびサンプル サーバのアップグレード」を参照してください。

ここでの説明に従って、起動スクリプトを修正します。

1. `bea.home` プロパティを変更します。

   WebLogic Server 7.0 の `license.bea` ファイルが置かれた BEA ホーム ディレクトリを指すようにします。次に例を示します。

   ```
   -Dbea.home=C:\bea700
   ```

2. `WL_HOME` を変更します。

   WebLogic Server 7.0 のインストール ディレクトリを指す必要があります。次に例を示します。

   ```
   WL_HOME=c:\bea700\weblogic700
   ```

3. `PATH` を変更します。

   `%WL_HOME%` 7.0 のホームが含まれるようにします。次に例を示します。

   ```
   PATH=%WL_HOME%\bin;%PATH%
   ```

4. `CLASSPATH` を変更します。

WebLogic Server 7.0 のクラスを指すようにします。次に例を示します。

```
CLASSPATH=%WL_HOME%\lib\weblogic_sp.jar;%WL_HOME%\lib\weblogic.jar
```

5. WebLogic Server 6.x の起動スクリプトにディレクトリ構造テストがあれば修正するか、または削除します。たとえば、スクリプトで相対パスの検証を試みる場合、ディレクトリ構造テストを修正するか、または削除します。

WebLogic Server 7.0 は、サーバのインストール時に JDK 1.3.1_02 の JVM をインストールします。サーバに付属する `setenv.sh` スクリプトはすべてこの JVM を指しています。動作確認された JVM についての最新情報は、「動作確認状況」ページで公開されています。

WebLogic Server 7.0 のディレクトリ構造について

WebLogic Server 7.0 のディレクトリ構造は、バージョン 6.x のものとは異なります。新しくなったディレクトリ構造の詳細については、『インストール ガイド』の「インストール後の作業の実行」の章の「WebLogic Server のディレクトリ構造について」を参照してください。

WebLogic Server 7.0 環境を使用して WebLogic Server 6.x ドメインを起動している場合、新しいディレクトリ構造が自動的に作成されます。ただし、WebLogic Server 6.x ドメインのディレクトリ構造を前提としているカスタム ツールまたはスクリプトがある場合、新しいディレクトリ構造に合わせて、それらのツールの相対パス設定を更新する必要があります。また同様に、スクリプトによって WebLogic Server 6.x 環境でドメインを作成するためのツールがある場合、そのスクリプトを変更する必要があります。スクリプトによる制御が可能なコンフィグレーション ウィザードを使用するのが最もよい方法です。

WebLogic Server 6.x から WebLogic Server 7.0 へのアプリケーションの移植

注意： このマニュアル全体を通して、作成される新しい WebLogic Server 7.0 ドメインのディレクトリを `domain` と表記します。

WebLogic 6.x アプリケーションを WebLogic Server 7.0 に移行するには、次の手順に従います。

1. WebLogic Server 7.0 をまだインストールしていない場合、ここでインストールします。詳細については、『インストール ガイド』を参照してください。

   注意： 古いバージョンと同じ場所に新しいバージョンをインストールしようとすると、インストーラにより警告が出されます。

2. バージョン 6.x のドメインと 7.0 のドメインには、それぞれ個別のディレクトリが必要です。同じディレクトリに複数の `config.xml` ファイルを置くことはできません。

   a. WebLogic Server 7.0 に移植する 6.x コンフィグレーション ドメインのそれぞれについて、選択したディレクトリに `/config/`*`domain`* ディレクトリをコピーします。名前がピリオド「`.`」で始まるディレクトリはすべて除外します (これらは WebLogic Server が内部的に使用するために作成したファイルまたはディレクトリです)。

      このディレクトリは新しいドメインの位置であり、そのドメインのすべてのコンフィグレーション情報が格納されます。バージョン 6.x の config ディレクトリが WebLogic Server 6.x 配布キットにない場合、WebLogic Server 7.0 で独自の WebLogic 6.x コンフィグレーションを再利用できます。

   b. デプロイメント記述子ファイル (`web.xml` および `weblogic.xml`) には Java コンパイラや外部ファイルなどの項目へのファイル パスが格納されている場合があるため、これらのファイルを識別します。WebLogic Server のコンフィグレーションは、ファイル システム上に格納できる多数のファイルを使用して行われます。一般に、これらのファイルは永続性リポジトリ (ログ ファイル、ファイルベースのリポジトリなど) またはユーティリティ (Java コンパイラ) です。これらのファイルは絶対パスまたは相対パスを使用してコンフィグレーションできます。

すべての外部ファイルが相対パスを使用して定義されており、ドメイン ディレクトリ内またはそれよりも下の階層に位置している場合、以降の手順はスキップしてください。

ドメイン ディレクトリの外部の位置を指す相対パスを使用して外部ファイルが定義されている場合、新しい config ディレクトリを起点にしたディレクトリ構造を作成し直してから、関連付けられているファイルを新しいディレクトリにコピーします。外部ファイルが絶対パスを使用して定義されている場合、これらのファイルを WebLogic Server 7.0 デプロイメントで再利用するのが適切かどうかを判断します。

たとえば、ログ ファイルや永続性ストアは再利用できますが、Java コンパイラなどのユーティリティについては、最新バージョンを使用するための更新が必要な場合があります。更新が必要なファイルについては、次の手順に進む前に、WebLogic Server 6.x の Administration Console を使用して必要な属性を編集し、新しいファイルまたはユーティリティが使われるようにします。

3. サーバ起動スクリプトをまだ編集していない場合、ここで編集します。手順については、1-3 ページの「起動スクリプトの修正」を参照してください。

注意： WebLogic Server 7.0 では、デプロイメント記述子にエラーのあるアプリケーションはデプロイされません。以前のバージョンの WebLogic Server では、デプロイメント記述子のエラーのあるアプリケーションもデプロイされていました。

セキュリティのアップグレード

WebLogic Server 7.0 は新しいセキュリティ アーキテクチャを備えています。個別の質問とその回答については、『WebLogic Security の紹介』を参照してください。

WebLogic Server 7.0 は、WebLogic Server の以前のバージョンからのアップグレードを行っているのか、それともバージョン 7.0 の新規インストールなのかを検出します。WebLogic Server 6.x からアップグレードしている場合、WebLogic Server 7.0 は互換性セキュリティで動作し、6.x のユーザおよびグループのコンフィグレーションを保持することができます。

ただし、6.x の主なセキュリティ機能には非推奨となったものもあり、WebLogic Server 7.0 ではセキュリティ機能が改良および拡張されているため、セキュリティ コンフィグレーションのアップグレードをお勧めします。以下の問題と手順のリストを参照してください。

- 1-7 ページの「互換性セキュリティでの WebLogic Server の起動」
- 1-8 ページの「MBean の ACL」
- 1-8 ページの「互換性セキュリティから WebLogic Server 7.0 セキュリティへのアップグレード」
- 1-10 ページの「セキュリティ レルム」
- 1-12 ページの「ゲスト ユーザ」
- 1-12 ページの「password.ini ファイル」
- 1-13 ページの「SSL プロトコルのアップグレード」

注意： WebLogic Server 7.0 のサンプルと PetStore は、デフォルトのセキュリティ コンフィグレーションを使用するようにコンフィグレーションされています。WebLogic Server 7.0 のサンプルと PetStore を互換性セキュリティで実行することはできません。

互換性セキュリティでの WebLogic Server の起動

WebLogic Server の以前のリリースでは、ファイル レルムがデフォルトとしてコンフィグレーションされていました。そのため、`config.xml` ファイルにセキュリティ レルムが定義されていなくても、WebLogic Server はファイル レルムを使用して起動できました。ただし、WebLogic Server を互換性セキュリティで実行するには、`config.xml` ファイルでファイル レルムまたは代替セキュリティ レルムを定義している必要があります。定義されていない場合、サーバが起動しないことがあります。

WebLogic Server を互換性セキュリティで起動できない場合は、新しいドメイン フォルダに `SerializedSystemini.dat` をコピーしてから、次のいずれかの作業を行います。

- WebLogic Server バージョン 6.x でコンフィグレーションを起動し、サーバの config.xml ファイルを保存し、保存した config.xml ファイルを WebLogic Server バージョン 7.0 に移植します。
- バージョン 6.x の config.xml ファイルを編集し、次の定義を追加します。

```
<Security Name=mydomain Realm=mysecurity/>
<Realm Name=mysecurity FileRealm=myrealm/>
<FileRealm Name=myrealm/>
```

MBean の ACL

MBean の ACL は WebLogic Server 7.0 ではサポートされていません。WebLogic Server 7.0 で MBean を保護する方法については、『管理者ガイド』の「システム管理操作の保護」を参照してください。

互換性セキュリティから WebLogic Server 7.0 セキュリティへのアップグレード

WebLogic Server 7.0 の新しいセキュリティ機能を活用する場合、既存のセキュリティ レルムを WebLogic Server 7.0 のセキュリティ レルムにアップグレードする必要があります。アップグレードを行うには、WebLogic Server 7.0 のセキュリティ プロバイダに既存のユーザとグループの情報を取り込み、リソースのセキュリティ ポリシーを定義して ACL 反映させます。

WebLogic Server 6.x コンフィグレーションが正常に起動する過程で、デフォルトのセキュリティ レルムとして互換性レルムが作成されます。互換性レルムには 6.x のセキュリティ データがすべて含まれています。また、myrealm という名前の、デフォルトの WebLogic Server 7.0 セキュリティ レルムも作成されます。アップグレードするには、互換性レルムを myrealm に置き換える必要があります。WebLogic Server Administration Console で次の操作を行います。

1. [レルム] ノードをクリックします。

   [レルム] テーブルに、コンフィグレーション済みの 2 つのセキュリティ レルムが表示されます。2 つのセキュリティ レルムは、CompatibilityRealm と

myrealm です。CompatibilityRealm のデフォルト属性は `true` に設定されます。

2. [myrealm] ノードをクリックします。
3. [プロバイダ] タブをクリックし、myrealm に対してコンフィグレーションされているセキュリティ プロバイダを表示します。デフォルトでは、WebLogic セキュリティ プロバイダが myrealm にコンフィグレーションされています。
4. WebLogic Server を起動できるユーザを `Administrators` グループに追加します。このユーザは `system` ユーザに代わるものです。`Administrators` グループにユーザを追加するには、次の手順に従います。
   a. [セキュリティ] ノードをクリックします。
   b. [レルム] ノードをクリックします。
   c. コンフィグレーションするレルムの名前 (myrealm など) をクリックします。
   d. [グループ] をクリックします。

      [グループ] タブが表示されます。このタブには、デフォルトの認証プロバイダで定義されているすべてのグループの名前が表示されます。
   e. [グループ] タブで Administrators グループをクリックします。
   f. [メンバシップ] タブをクリックして、WebLogic Server を起動できるユーザを `Administrators` グループに追加します。
   g. [適用] ボタンをクリックして、変更を保存します。
5. 6.x のセキュリティ レルムでコンフィグレーションしたユーザおよびグループを、認証プロバイダに追加します。
6. オプションで、6.x のユーザとグループに対するロールを定義します。『WebLogic リソースのセキュリティ』を参照してください。
7. 6.x の ACL をセキュリティ ポリシーとして表現します。『WebLogic リソースのセキュリティ』を参照してください。
8. myrealm をデフォルトのセキュリティ レルムとして設定します。『WebLogic Security の管理』の「デフォルト セキュリティ レルムの設定」を参照してください。

9. WebLogic Server を再起動します。

   WebLogic Server 7.0 が起動され、サーバがデプロイされるたびに、ロールとセキュリティ ポリシーが適用されます。これ以降のサーバとそのメソッドへのアクセスは、変更しない限りこれらのロールとセキュリティ ポリシーによって制約されます。

セキュリティ レルム

WebLogic Server 7.0 ではセキュリティ レルムの有効範囲が変更されました。WebLogic Server 6.x では、セキュリティ レルムで認証および認可サービスを提供していました。ファイル レルムか、または Lightweight Data Access Protocol (LDAP)、Windows NT、UNIX、RDBMS レルムなどの代替レルムを選択しました。また、カスタム セキュリティ レルムを記述することもできました。

WebLogic Server 7.0 では、セキュリティ レルムはスコーピング (有効範囲の設定) メカニズムとして機能します。各レルムはコンフィグレーション済みのセキュリティ プロバイダ、ユーザ、グループ、ロール、およびセキュリティ ポリシーのセットから構成されます。セキュリティ レルム内の認証および認可プロバイダが、認証および認可サービスを提供します。

6.x のセキュリティ レルムを WebLogic Server 7.0 にアップグレードするには、以下のような方法があります。

- 互換性セキュリティを使用して、ユーザ、グループ、および、LDAP、Windows NT、UNIX、RDBMS、カスタム セキュリティ レルムでコンフィグレーション済みの ACL にアクセスします。互換性レルムのレルム アダプタ認証プロバイダで、6.x のセキュリティ レルムに格納されているユーザ、グループ、および ACL にアクセスできます。

  互換性セキュリティの使い方については、1-7 ページの「互換性セキュリティでの WebLogic Server の起動」を参照してください。

- レルム アダプタ認証プロバイダを WebLogic Server 7.0 のセキュリティ プロバイダと共に使用します。この方法では、LDAP、Windows NT、UNIX、または RDBMS セキュリティ レルムに格納されているユーザおよびグループにアクセスする代わりに、WebLogic Server 7.0 で使用可能なロールおよびセキュリティ ポリシーを使用できます。複数の認証プロバイダをコンフィグレーションできるので、ユーザおよびグループを WebLogic Server 7.0 の認

証プロバイダにアップグレードせずに、既存の 6.x のセキュリティ レルムを使用することもできます。

注意： レルム アダプタ認証プロバイダを、バージョン 7.0 のセキュリティ レルムの認証プロバイダとしてコンフィグレーションしている場合には、RDBMS レルムに格納されている ACL にアクセスできません。ロールおよびセキュリティ ポリシーを使用してアプリケーション リソースを保護する必要があります。

WebLogic Server 7.0 のセキュリティ レルムでレルム アダプタ認証プロバイダを使用するには、以下の手順に従います。

1. 互換性セキュリティを起動します。
2. 互換性レルム内のレルム アダプタ認証プロバイダに、6.x のセキュリティ レルムからのユーザおよびグループが含まれていることを確認します ([ユーザ] および [グループ] テーブルに、既存のユーザおよびグループが表示されることを確認します)。ユーザおよびグループの情報は `filerealm.properties` ファイルにコピーされます。
3. 使用するレルム アダプタ認証プロバイダを含むセキュリティ レルムをクリックします (たとえば myrealm など)。
4. [プロバイダ | 認証プロバイダ] をクリックします。
5. セキュリティ レルムでレルム アダプタ認証プロバイダをコンフィグレーションします。レルム アダプタ認証プロバイダに、互換性レルム内のレルム アダプタ認証プロバイダと同じ名前を与えます。
6. レルム アダプタ認証プロバイダの [制御フラグ] 属性を OPTIONAL に設定します。
7. WebLogic 認証プロバイダ (Administration Console では [Default Authenticator] と呼ばれる) の [制御フラグ] 属性を SUFFICIENT に設定します。
8. WebLogic Server を起動できるユーザを `Administrators` グループに追加します。このユーザは `system` ユーザに代わるものです。詳細については、1-8 ページの「互換性セキュリティから WebLogic Server 7.0 セキュリティへのアップグレード」を参照してください。
9. WebLogic Server を再起動します。
10. [ドメイン | セキュリティ] ノードを展開します。>

11. [一般] タブを選択します。
12. レルム アダプタ認証プロバイダをコンフィグレーションしたセキュリティ レルムを、デフォルト セキュリティ レルムとして設定します。
13. [適用] をクリックします。

注意： ACL は WebLogic Server 7.0 にアップグレードされません。

ゲスト ユーザ

WebLogic Server 7.0 では、guest ユーザはデフォルトでは提供されなくなりました。guest ユーザを使用するには、互換性セキュリティで実行するか、または使用しているセキュリティ レルムのデフォルトの認証プロバイダにユーザとして guest ユーザを定義する必要があります。ユーザを定義する方法については、『WebLogic リソースのセキュリティ』の「ユーザの作成」を参照してください。

WebLogic Server 6.x では、認証を受けていないユーザ (匿名ユーザ) がすべてゲスト ユーザ (guest) として認識され、ゲスト ユーザは WebLogic Server リソースへのアクセスを許可されていました。WebLogic Server 7.0 では、ゲスト ユーザと匿名ユーザが区別されます。WebLogic Server 6.x のときと同じようにゲスト ユーザを使用するには、デフォルトの認証プロバイダにゲスト ユーザを追加し、WebLogic Server の起動時に次のプロパティを設定します。

```
-Dweblogic.security.anonymousUserName=guest
```

このコマンドライン プロパティを指定しない場合、匿名ユーザの名前は <anonymous> になります。

password.ini ファイル

WebLogic Server の以前のリリースでは、password.ini ファイルを使用して、WebLogic Server デプロイメントの管理 ID を特定することができました。password.ini ファイルは、WebLogic Server 7.0 ではサポートされなくなりました。このファイルは、暗号化された形式でユーザ名とパスワードを格納する boot.properties ファイルに置き換えられています。boot.properties ファイルの使用方法については、『管理者ガイド』の「起動 ID ファイルの作成」を参照

してください。古い password.ini ファイルにはクリア テキストのパスワードが格納され、ユーザ名は格納されていないため、このファイルを直接アップグレードすることはありません。

SSL プロトコルのアップグレード

この節では、SSL プロトコルのアップグレード方法についての情報を示します。信頼性のある CA キーストアの作成、プライベート キーのキーストアの作成、および、互換性セキュリティにおける CertAuthenticator の使用の手順を説明します。

信頼性のある CA キーストアの作成

WebLogic Server 7.0 では、デフォルトで、クライアントがサーバの信頼性のある認証局 (CA) のチェックを行います。このチェックは、WebLogic Server がクライアントとして機能している場合だけでなく、SSL プロトコルを介してクライアントおよびサーバを接続する場合には常に行われます。たとえば、クライアントが SSL プロトコルを使用して Apache HTTP サーバに接続している場合、クライアントはサーバから提示された信頼性のある認証局のチェックを実行します。クライアントがサーバの認証局に信頼性がないと判断した場合、認証局は拒否されます。以前のバージョンの WebLogic Server では、この信頼性検証は行われませんでした。

既存の WebLogic 6.x のクライアントで SSL プロトコルを介してサーバと通信するには、以下の変更を行います。

1. クライアントに対して以下のコマンドライン引数を指定します。

   `-Dweblogic.security.SSL.trustedCAKeyStore=`*`absoluteFilename`*

   `absoluteFilename` は信頼性のある認証局を含むキーストアの名前です。

注意： このファイル フォーマットは、証明書ファイルではなく、キーストアです。信頼性のある認証局をキーストアにロードする必要があります。

2. 信頼性のある認証局を、サーバからクライアントのキーストアにロードします。キーストア内の信頼性のある認証局を表示したり、キーストアに新規の信頼性のある認証局をロードするには、JDK の keytool ユーティリティを使用します。

キーストアに信頼性のある認証局を追加するには、コマンド プロンプトで以下のように入力します。

`keytool -import -trustcacerts -alias` <some alias name> `-file` <the file that contains the trusted CA> `-keystore` <the trusted CA keystore> `-storepass` <your trusted CA Keystore password>

WebLogic Server に付属の信頼性のある認証局は、*`WL_HOME`*`/server/lib/cacerts` ディレクトリに置かれています。以下のコマンドを使用して、WebLogic Server に付属の信頼性のある認証局をキーストアに追加します。

`keytool -import -trustcacerts -alias` <some alias name> `-file` <the file that contains the trusted CA> `-keystore WL_HOME/server/lib/cacerts -storepass changeit`

`keytool` の詳細については、Sun の Web サイト (http://java.sun.com/products/jdk/1.2/docs/tooldocs/solaris/keytool.html) を参照してください。

クライアントについては、`trustedCAKeyStore` コマンドライン引数はデフォルトで JDK の `jre/lib/security/cacerts` キーストアになります。独自の CA を JDK の信頼性のある CA キーストアに追加し、コマンドライン引数は指定しないでおくことができます。あるいは、信頼性のある独自の CA キーストアを作成し、そのキーストアを指すように引数を設定することができます。

双方向 SSL または相互認証については、クライアント側で上記の 2 つの手順を実行することに加えて、サーバ側で次のどちらかの手順を実行します。

- サーバのコマンドラインに `-Dweblogic.security.SSL.trustedCAKeyStore=`*`absoluteFilename`* を追加します。

  `absoluteFilename` は、信頼性のある CA キーストアの名前です。

  または

- キーストア プロバイダをコンフィグレーションするときに、`RootCAKeyStoreLocation` 属性を設定します。

信頼性のある CA 証明書を信頼性のある CA キーストアにロードしない場合、セキュア ポートの使用時に問題が生じる場合があります。

互換性セキュリティでの CertAuthenticator の使用

WebLogic Server 7.0 では、ユーザ名とパスワードの認証の前に、最初に CertAuthenticator が呼び出されます。この動作は WebLogic Server 6.x からの変更点です。従って、クライアントが双方向 SSL を使用し、セキュリティ資格としてとユーザ名とパスワードを提供していた場合、WebLogic Server 6.x 用に記述された CertAuthenticator を変更する必要があります。

変更の必要がある CertAuthenticator を使用すると、WebLogic Server 6.x では許可されても、WebLogic Server 7.0 ではアクセスが拒否されます。CertAuthenticator を変更するには、証明書が SSL 接続を作成するためだけに使用される場合、認識されない証明書またはすべての証明書に対して `NULL` を返すようにします。

WebLogic Server 7.0 では、X.509 ID アサーションはデフォルトで無効になっています。WebLogic Server 6.x の `config.xml` ファイルで CertAuthenticator を使用していた場合、互換性セキュリティを使用するときに、X.509 ID アサーションを手動でコンフィグレーションする必要があります。X.509 ID アサーションの有効化は、レルム アダプタ認証プロバイダに関するオプションを設定することによって行います。互換性セキュリティでの実行中に WebLogic Server Administration Console で次の手順を行います。

1. [セキュリティ] ノードをクリックします。
2. [レルム] ノードをクリックします。
3. [CompatibilityRealm] ノードをクリックします。
4. [プロバイダ] ノードをクリックします。
5. [認証プロバイダ] ノードをクリックします。
6. [RealmAdapterAuthenticator] ノードをクリックします。

   [一般] タブが表示されます。
7. [アクティブ タイプ] ボックスに `X.509` と入力します。
8. [適用] をクリックして、変更を保存します。
9. WebLogic Server を再起動します。

暗号スイート

Certicom 暗号スイートに関して問題が発生した場合の対処については、『リリース ノート』の問題 073360 についての情報を参照してください。

WebLogic Tuxedo Connector のアップグレード

WebLogic Server 7.0 で WebLogic Tuxedo Connector を使用するためには、アプリケーションおよびコンフィグレーションに以下の変更を行う必要があります。

- 1-16 ページの「WebLogic Tuxedo Connector の起動」
- 1-17 ページの「WebLogic Tuxedo Connector の XML コンフィグレーション ファイルの変換」
- 1-19 ページの「インバウンド RMI-IIOP アプリケーションの更新」
- 1-20 ページの「リモート ユーザの認証」
- 1-21 ページの「WebLogic Tuxedo Connector のプロパティの設定」

WebLogic Tuxedo Connector の起動

注意： WebLogic Tuxedo Connector のコンフィグレーション方法について、詳しくは「Administration Console を使用した WebLogic Tuxedo Connector のコンフィグレーション」を参照してください。

コネクタの以前のリリースでは、WebLogic Tuxedo Connector セッションの開始に WebLogic Server の起動クラスを、またセッションの終了に WebLogic Server の停止クラスを使用していました。WebLogic Server 7.0 では、WebLogic Tuxedo Connector は起動クラスまたは停止クラスを使用しません。WebLogic Tuxedo Connector セッションは WTCServer MBean を使用して管理されます。

- WebLogic Tuxedo Connector セッションは、コンフィグレーション済みの WTCServer MBean を、選択したサーバに割り当てるときに開始します。

- WebLogic Tuxedo Connector セッションは、WTCServer MBean を WebLogic Server から削除して、または WebLogic Server を停止することによって終了します。

WebLogic Server Tuxedo Connector セッションの開始と終了については、「サーバへの WTCServer の割り当て」を参照してください。

WebLogic Tuxedo Connector の XML コンフィグレーション ファイルの変換

WebLogic Tuxedo Connector はサービスとして実装され、独立した XML コンフィグレーション ファイルを使用しなくなりました。`WTCMigrateCF` ツールは、XML コンフィグレーション ファイルの情報を、アクティブな管理サーバの `config.xml` ファイルに移行します。WebLogic Tuxedo Connector の XML コンフィグレーション ファイルを変換するには、次の手順に従います。

1. 「WebLogic Server 環境の設定」での説明に従って、WebLogic Server 開発シェルをセットアップします。
2. WebLogic Server のインスタンスを起動します。
3. 新しいシェル ウィンドウを開きます。
4. `WTCMigrateCF` ツールを起動します。次のコマンドを入力します。

```
java -Dweblogic.wtc.migrateDebug weblogic.wtc.gwt.WTCMigrateCF
-url URL -username USERNAME -password PASSWORD -infile CONFIGWTC
[-server SERVERNAME] [-domain DOMAIN] [-protocol PROTOCOL]
[-deploy]
```

このコマンドの引数の定義は次のとおりです。

引数	説明
`-Dweblogic.wtc.migrateDebug`	wtc.migrateDebug モードを有効にするために使われる `WebLogic` プロパティ。
`URL`	サーバに渡される URL。 例：`\\myServer:7001`

引数	説明
USERNAME	サーバに渡されるユーザ名。 例 : system
PASSWORD	サーバに渡されるパスワード。 例 : mypasswd
CONFIGWTC	config.xml ファイルに移行される、WebLogic Tuxedo Connector の XML コンフィグレーション ファイルの絶対パスとファイル名。 例 : d:\bea\weblogic700\server\samples\examples\wtc\atmi\simpapp\bdmconfig.xml
SERVERNAME	省略可能。新しい WTCServer MBean を割り当てる管理サーバまたは管理対象サーバの名前。デフォルト値 : その時点でアクティブな管理サーバ
DOMAIN	省略可能。新しい WTCServer MBean を割り当てる WebLogic Server ドメインの名前。デフォルト値 : その時点でアクティブなドメイン
PROTOCOL	省略可能。URL と組み合わせて使用するプロトコル。デフォルト値 : t3:
-deploy	省略可能。選択したサーバに WTCServer MBean を割り当てるために使用する。このフラグが設定されている場合、WebLogic Tuxedo Connector セッションは直ちに開始し、WTCServer MBean によって指定されたサービスは直ちに開始される。

移行が完了すると、移行ユーティリティは次のメッセージを表示します。

```
The WTC configuration file migration is done!

No error found!!!
```

指定された XML コンフィグレーション ファイル内の情報が WTCServer MBean に移行され、手順 2 で指定したサーバの config.xml ファイルに取り込まれます。

インバウンド RMI-IIOP アプリケーションの更新

WebLogic Tuxedo Connector でインバウンド RMI-IIOP を使用する方法については、「RMI/IIOP および CORBA を相互に運用する WebLogic Tuxedo Connector の使用」を参照してください。

インバウンド RMI-IIOP を使用する場合、WebLogic Tuxedo Connector のインスタンスを Tuxedo の CORBA アプリケーションに接続する参照オブジェクトを修正しなければなりません。Tuxedo の CORBA オブジェクトでは現在、サーバ名を使用してリモートの WebLogic Tuxedo Connector オブジェクトを参照しています。以前のリリースでは `DOMAINID` を使用していました。

1. `ior.txt` ファイルで、オブジェクト参照 `corbaloc:tgiop` または `corbaname:tgiop` を修正します。

   例： `corbaloc:tgiop:`*`servername`*`/NameService`

   `servername` はサーバの名前です。

2. 次のコマンドを入力して、WebLogic Server (WLS) ネーミング サービスを登録します。

   `cnsbind -o ior.txt` *`your_bind_name`*

   `your_bind_name` は、Tuxedo アプリケーションでのオブジェクト名です。

3. Tuxedo ドメイン コンフィグレーション ファイルの `*DM_REMOTE_SERVICES` セクションを修正します。以前は `DOMAINID` であった WebLogic Server サービス名を、使用している WebLogic Server の名前で置き換えます。

コード リスト 1-1　ドメイン コンフィグレーション ファイル

```
*DM_RESOURCES
     VERSION=U22

*DM_LOCAL_DOMAINS
     TDOM1 GWGRP=SYS_GRP
     TYPE=TDOMAIN
     DOMAINID="TDOM1"
     BLOCKTIME=20
     MAXDATALEN=56
     MAXRDOM=89

*DM_REMOTE_DOMAINS
     TDOM2 TYPE=TDOMAIN
```

```
     DOMAINID="TDOM2"

*DM_TDOMAIN
     TDOM1 NWADDR="<Tuxedo ドメインのネットワーク アドレス：ポート>"
     TDOM2 NWADDR="<WTC ドメインのネットワーク アドレス：ポート>"

*DM_REMOTE_SERVICES
"//servername"
```

servername は、使用している WebLogic Server の名前です。

4. dmloadcf を使用して、修正したドメイン コンフィグレーション ファイルをロードします。

これで、アプリケーションを起動する準備ができました。

リモート ユーザの認証

詳細については、「ユーザ認証」を参照してください。

WebLogic Tuxedo Connector はアクセス制御リスト (ACL) を使用します。ACL は、サービスを実行できるリモート ドメインを制限することによって、ローカル ドメインの内部にあるローカル サービスへのアクセスを制限します。このパラメータの有効な値は次のとおりです。

- LOCAL
- GLOBAL

ACL ポリシーが LOCAL の場合

WebLogic Tuxedo Connector の ACL ポリシーが LOCAL に設定されている場合、Tuxedo リモート ドメインの DOMAINID がローカル ユーザとして認証される必要があります。WebLogic Tuxedo Connector で DOMAINID がローカル ユーザとして認証されるようにするには、WebLogic Server コンソールを使用して次の手順を実行します。

1. [セキュリティ] ノードをクリックします。
2. [レルム] をクリックします。

3. デフォルトのセキュリティ レルムを選択します。
4. [ユーザ] をクリックします。
5. [新しい User のコンフィグレーション] テキスト リンクをクリックします。
6. [DefaultAuthenticator] をクリックします。
7. [一般] タブで、次のことを行います。
   a. [名前] フィールドに Tuxedo の `DOMAINID` を追加します。
   b. パスワードを入力し、確認用にもう一度入力します。
   c. [適用] をクリックします。

ACL ポリシーが GLOBAL の場合

WebLogic Tuxedo Connector の ACL ポリシーが `GLOBAL` の場合、ユーザのセキュリティ トークンが渡されます。管理上の変更は必要ありません。

WebLogic Tuxedo Connector のプロパティの設定

注意： WebLogic Tuxedo Connector のプロパティの設定方法については、「WebLogic Tuxedo Connector のプロパティの設定方法」を参照してください。

TraceLevel および PasswordKey は、現在では WebLogic Server のプロパティとなっています。

WebLogic Server ログ ファイルを使用して WebLogic Tuxedo Connector をモニタするには、WebLogic Server の TraceLevel プロパティを使用してトレース レベルを設定する必要があります。詳細については、「WebLogic Tuxedo Connector のモニタ」を参照してください。

- `-Dweblogic.wtc.TraceLevel=`*tracelevel*

  tracelevel は、WebLogic Tuxedo Connector のトレース レベルを指定する 10,000 ～ 100,000 の範囲の数値です。

weblogic.wtc.gwt.genpasswd ユーティリティを使用してパスワードを生成するには、WebLogic Server の PasswordKey プロパティを使用してパスワード キーを設定する必要があります。パスワードの生成方法については、「WTCPassword MBean のコンフィグレーション」を参照してください。

- `-Dweblogic.wtc.PasswordKey=`*mykey*

  mykey はキー値です。

その他のアップグレード手順と情報

以下の節では、非推奨となった機能、アップグレード、および WebLogic Server 7.0 での重要な変更点についての有益な補足情報を示します。

注意： WebLogic Server 7.0 ではサンプル データベースとして PointBase 4.2 を使用しており、Cloudscape データベースは同梱されていません。

- 1-23 ページの「ant.jar」
- 1-23 ページの「Apache Xalan XML トランスフォーマ」
- 1-24 ページの「Apache Xerces XML パーサ」
- 1-24 ページの「applications ディレクトリ」
- 1-25 ページの「デプロイメント」
- 1-26 ページの「EJB 2.0」
- 1-28 ページの「jCOM」
- 1-29 ページの「JDBC」
- 1-29 ページの「JMS」
- 1-29 ページの「JMX」
- 1-30 ページの「Jolt Java Client」
- 1-30 ページの「JSP」
- 1-30 ページの「管理対象サーバ」

- 1-31 ページの「MBean API の変更」
- 1-31 ページの「Security」
- 1-32 ページの「サーブレット」
- 1-32 ページの「スレッド プール サイズ」
- 1-33 ページの「Web アプリケーション」
- 1-34 ページの「Solaris での WebLogic Server クラスタ」
- 1-34 ページの「Web サービス」
- 1-35 ページの「書き込み可能な config.xml ファイル」
- 1-35 ページの「非推奨になった API と機能」
- 1-37 ページの「削除された API と機能」

ant.jar

新しい ant タスクを開発している場合、`server/lib/ant.jar` ファイルをクラスパスに手動で追加する必要があります。

Apache Xalan XML トランスフォーマ

WebLogic Server 7.0 の組み込みトランスフォーマは、Apache Xalan 2.2 トランスフォーマをベースとしています。

Xalan API を直接使用することは非推奨になりました。それらの API を引き続き使用していて問題が発生する場合は、Java API for XML Processing (JAXP) を使用して XSLT を使用することをお勧めします。

Apache の Xalan コードは、Xerces と Xalan が連携できるように変更が行われています。その変更が含まれていないので、Apache から Xalan を使用すると問題が発生する場合があります。

一般には、JAXP を使用し、ベンダ固有のコードを SAX、DOM、および XSL 処理用に JAXP などの中立な API に移行するのが最適です。

以前のバージョンの WebLogic Server では、www.apache.org から入手できる Xerces パーサと Xalan トランスフォーマの未修正版が *WL_HOME*\server\ext\xmlx.zip ファイルに収められていました。現在では、ZIP ファイルにこれらのクラスおよびインタフェースは含まれていません。未修正版の Xerces パーサおよび Xalan トランスフォーマは、Apache Web サイトから直接ダウンロードしてください。

Apache Xerces XML パーサ

WebLogic Server 7.0 の組み込み XML パーサは、Apache Xerces 1.4.4 パーサをベースとしています。パーサは SAX および DOM インタフェースのバージョン 2 を実装しています。以前のバージョンに付属していた古いパーサを使用すると、旧式であることを示すメッセージが表示される場合があります。

WebLogic Server 7.0 には、WebLogic FastParser も付属します。これは、WebLogic Web サービスに関連付けられた SOAP および WSDL ファイルなど、中小規模のドキュメントを処理するために特別に設計された高性能の XML パーサです。アプリケーションで中小規模 (要素数が 10,000 個程度まで) の XML ドキュメントを処理することがほとんどの場合、FastParser を使用するように WebLogic Server をコンフィグレーションします。

以前のバージョンの WebLogic Server では、www.apache.org から入手できる Xerces パーサと Xalan トランスフォーマの未修正版が *WL_HOME*\server\ext\xmlx.zip ファイルに収められていました。現在では、ZIP ファイルにこれらのクラスおよびインタフェースは含まれていません。未修正版の Xerces パーサおよび Xalan トランスフォーマは、Apache Web サイトから直接ダウンロードしてください。

applications ディレクトリ

WebLogic Server 6.1 および 7.0 には 2 種類の実行時モードがあります。2 つのモードは「開発」と「プロダクション」です。実行時モードは、WebLogic Server の起動時にコマンドライン パラメータを使用して選択します (`-Dweblogic.ProductionModeEnabled=true | false`)。このパラメータを設定しないと、サーバは開発モードで動作します。開発モードでは、サーバの動作は WebLogic Server 6.0 の場合と同じです。ただし、プロダクション モードの場合

には自動デプロイメント機能が無効になります。applications ディレクトリ内のデプロイメント ユニットのうち、コンフィグレーション リポジトリ (config.xml) においてデプロイ対象として明示的に設定されていないものは自動的にはデプロイされません。WebLogic Server 6.1 および 7.0 では、デフォルト ドメイン (mydomain) および Pet Store の出荷時コンフィグレーションはプロダクション モードである点に注意してください。サンプルの出荷時コンフィグレーションは開発モードです。

デプロイメント

WebLogic Server 7.0 では、新しい 2 フェーズ デプロイメント モデルを備えています。このデプロイメント モデルと、バージョン 7.0 のその他のデプロイメント機能の詳細については、「WebLogic Server デプロイメント」を参照してください。デフォルトでは、静的にコンフィグレーションされるアプリケーションは 6.x デプロイメント モデルを使用します。コンソールまたは新しいコマンドライン ユーティリティの weblogic.Deployer を使用して開始されるデプロイメントでは、新しい 2 フェーズ デプロイメント モデルが使用されます。

WebLogic Server 7.0 では、デプロイメント記述子にエラーのあるアプリケーションはデプロイされません。以前のバージョンの WebLogic Server では、デプロイメント記述子のエラーのあるアプリケーションもデプロイされていました。たとえば、6.x アプリケーションのデプロイメント記述子で参照記述スタンザが欠落していた場合、そのスタンザを追加するまでは 7.0 サーバにアプリケーションがデプロイされません。一般的なスタンザは次のようになります。

```
<ejb-reference-description>
<ejb-ref-name>ejb/acc/Acc</ejb-ref-name>
<jndi-name>estore/account</jndi-name>
</ejb-reference-description>
```

WebLogic Server 7.0 を使用する場合、6.x プロトコルを使用してコンソールからデプロイメントを行うことはできなくなりました。結果として、新しいデプロイメント API を使用しなければなりません。アプリケーションが以前に 6.x でデプロイされており、単にサーバを起動しているだけの場合、アプリケーションは 1 フェーズのデプロイメントによってデプロイされます。コマンドライン ユーティリティの weblogic.deploy および weblogic.refresh と、weblogic.management.tools.WebAppComponentRefreshTool はバージョン 7.0 では非推奨になっています。

非推奨になった MBean の属性および操作については、1-35 ページの「非推奨になった API と機能」を参照してください。

開発モードのとき、管理サーバ上の applications ディレクトリ内のアプリケーションがステージングされるようになりました。バージョン 6.x ではステージングされませんでした。詳細については、『WebLogic Server アプリケーションの開発』の「2 フェーズ デプロイメント」の節の「アプリケーション ステージング」を参照してください。

EJB 2.0

EJB 2.0 仕様は、WebLogic Server 6.0 から WebLogic Server 7.0 までの間に大幅に、また WebLogic Server 6.1 から WebLogic Server 7.0 までの間にも若干変更されています。

ここでは重要な変更点をいくつか示します。WebLogic Server 6.0 から WebLogic Server 7.0 までの間に行われたすべての仕様変更を確認するには、http://java.sun.com/products/ejb/2.0.html から EJB 2.0 最終仕様を参照またはダウンロードしてください。

WebLogic Server 6.0 から WebLogic Server 6.1 までの間に行われた変更については、WebLogic Server バージョン 6.1 マニュアルの「WebLogic Server エンタープライズ JavaBean の概要」の「このリリースの拡張機能」を参照してください。WebLogic Server 6.x で動作した EJB 1.1 Bean は、通常はそのまま WebLogic Server 7.0 でも動作します。

EJB 2.0 Bean については、以下の変更が必要になる場合があります。

- デプロイメント記述子に、バージョン 7.0 のものと名前が異なる 6.0 要素が含まれている場合、デプロイメント記述子を編集してその名前を手動で変更する必要があります。以下、バージョン 7.0 で変更が必要な場合がある要素名の例をいくつか示します。
  - 7.0 では、関係に参加する特定の EJB を識別するために使用される要素の名前は `relationship-role-source` です。6.0 では、要素名は `role-source` でした。
  - 7.0 では、送り先がキューまたはトピックのどちらであるかを指定する要素の名前は `destination-type` です。6.0 では、要素名は `jms-destination-type` でした。

  - 7.0 では、送り先がキューまたはトピックのどちらであるかを指定する要素の名前は `run-as` です。6.0 では、要素名は `run-as-specified-identity` でした。

- 7.0 では、`EJB-QL` クエリに `SELECT` 句が必要です。
- WebLogic Server 7.0 では、すべての EJB 2.0 CMP Bean の `ejb-jar.xml` ファイルで `abstract-schema-name` 要素を指定する必要があります。

EJB 2.0 仕様の変更に由来する、その他の主な変更点は以下のとおりです。

- バージョン 6.0 にはなかったローカル インタフェースが 6.1 および 7.0 では存在します。
- リモート リレーションシップはバージョン 6.1 および 7.0 では存在しません。
- WebLogic Server 7.0 における読み込み専用 Bean の新しい実装では、トランザクション属性は `NOT_SUPPORTED` に限定されません。この実装では排他的ロックも行われず、各トランザクションが使用できる、読み込み専用 Bean の個別のインスタンスが提供されます。
- 最終版でない 2.0 仕様に基づく EJB 実装では、`EJB QL` クエリで、修飾形式のパスを使用せずに CMP または CMR フィールドを参照できました。最終仕様では、すべての `EJB QL` クエリで修飾形式パスの使用が必須となっています。
- WebLogic Server 7.0 における読み込み専用 Bean の新しい実装では、トランザクション属性は `NOT_SUPPORTED` に限定されません。新しい実装では排他的ロックも行われず、各トランザクションが使用できる、読み込み専用 Bean の個別のインスタンスが提供されます。

アプリケーション デプロイメントのスコープ内でサーブレットを使用した際の問題への対処については、1-25 ページの「デプロイメント」を参照してください。

weblogic.management.configuration.EJBComponentMBean の変更点

WebLogic Server 6.1 から WebLogic Server 7.0 にかけて、インタフェース `weblogic.management.configuration.EJBComponentMBean` が変更され、`ComponentMBean` および `EJBContainerMBean` の両方を拡張するようになりま

した。EJBComponentMBean を実装するすべてのメソッド (getVerboseEJBDeploymentEnabled など) は、WebLogic Server 6.0 から 7.0 に移行する際、EJBContainerMBean をサポートするように変更しなければなりません。

max-beans-in-cache パラメータ

WebLogic Server 7.0 の max-beans-in-cache パラメータは、データベースの同時実行性を高めるためにキャッシュに格納される Bean の最大数を制御します。以前のバージョンの WebLogic Server では、max-beans-in-cache は無視され、キャッシュのサイズは無制限でした。このパラメータの値をより大きく調整しなければならない場合があります。

完全修飾パス式

WebLogic Server 7.0 で実行される EJB QL クエリでは、すべてのパス式は完全修飾形式でなければなりません。これは WebLogic Server 6.x からの変更点です。WebLogic Server 7.0 上で 6.x EJB を使用していて、ejbc の実行時に ejbc エラーが発生する場合、ejb-jar.xml ファイル内の ejb-ql 要素、または weblogic-cmp-jar.xml ファイル内の weblogic-ql 要素のどちらかを修正する必要があります。次に例を示します。

- WebLogic Server 6.x では、次のクエリは正しくコンパイルされます。

  ```
  SELECT address FROM CustomerBean AS c WHERE zip = ?1
  ```

- WebLogic Server 7.0 で同じクエリをコンパイルするためには、address フィールドと zip フィールドを修飾する必要があります。

  ```
  SELECT c.address FROM CustomerBean AS c WHERE c.zip = ?1
  ```

jCOM

WebLogic jCOM 6.1 から WebLogic jCOM 7.0 へのアップグレードについては、『WebLogic jCOM プログラマーズ ガイド』の「アップグレードの考慮事項」を参照してください。

JDBC

JDBC 接続プールの最小の増加容量は、WebLogic Server 6.1 では 0 でしたが、バージョン 7.0 では 1 に変更されました。『WebLogic Server コンフィグレーション リファレンス』の「JDBCConnectionPool」を参照してください。

JDBCConnectionPool MBean の `PreparedStatementCacheSize` および `XAPreparedStatementCacheSize` のデフォルト値は、WebLogic Server 6.1 では 0 でしたが、WebLogic Server 7.0SP2 以降のリリースでは 5 に増加されました。Prepared Statement キャッシュとその制限の詳細については、『管理者ガイド』の「Prepared Statement キャッシュのパフォーマンスの向上」を参照してください。

JMS

WebLogic Server 7.0 は、JavaSoft の JMS 仕様バージョン 1.0.2 をサポートしています。

すべての WebLogic JMS 6.x アプリケーションは WebLogic JMS 7.0 でサポートされています。ただし、新しい高可用性 JMS の機能をアプリケーションで利用する場合は、既存の物理的な送り先 (キューおよびトピック) を、単一の分散送り先セットの一部としてコンフィグレーションする必要があります。JMS の分散送り先の使用方法については、『WebLogic JMS プログラマーズ ガイド』の「分散送り先の使用」を参照してください。

WebLogic JMS アプリケーションの移植の詳細については、『WebLogic JMS プログラマーズ ガイド』の「WebLogic JMS アプリケーションの移植」を参照してください。

JMX

WebLogic Server 6.x のすべてのパブリックな MBean および属性は、WebLogic Server 7.0 でサポートされています。ただし、内部的な MBean または属性を使用している場合には、移植時に問題が発生することがあります。

非推奨になった MBean の属性および操作については、1-35 ページの「非推奨になった API と機能」を参照してください。

Jolt Java Client

Jolt ユーザは、BEA WebLogic Server 7.0 を使用する Web に BEA Tuxedo サービスを対応させるために、Jolt Java Client 8.0.1 にアップグレードする必要があります。Jolt Java Client 8.0.1 は BEA Product Download Center から入手できます。BEA WebLogic Server 7.0 へのリンクを辿り、[Modules for WebLogic Server] リストから [Jolt Java Client 8.0.1] を選択してください。

JSP

JSP 仕様の変更により、null のリクエスト属性は、空の文字列ではなく文字列「null」を返すようになりました。WebLogic Server バージョン 6.1 からは、`weblogic.xml` に `printNulls` という新しいフラグが導入されています。このフラグはデフォルトで true になっており、デフォルトが「null」であることを意味しています。このフラグを false に設定すると、「null」の式が文字列「null」ではなく空の文字列として出力されます。

`weblogic.xml` の `printNulls` 要素のコンフィグレーション例を次に示します。

```
<weblogic-web-app>
   <jsp-param>
     <param-name>printNulls</param-name>
     <param-value>false</param-value>
   </jsp-param>
</weblogic-web-app>
```

管理対象サーバ

WebLogic Server 6.x を実行している管理対象サーバは、WebLogic Server 7.0 を実行している管理サーバを使用してそのコンフィグレーションおよび起動情報を取得できません。WebLogic Server 6.x のインストール ディレクトリにある `running-managed-servers.xml` ファイルを、WebLogic Server 7.0 のインストール ディレクトリにはコピーしないでください。

MBean API の変更

このドキュメントの以前のバージョンおよび他のサンプルのドキュメントに、Administration Server で MBeanHome インタフェースをルックアップする方法として weblogic.management.Admin.getInstance().getAdminMBeanHome() を使用するという誤った説明がありました。

weblogic.management.Admin クラスはパブリックではありません。非パブリックなこのクラスを使用する代わりに、JNDI を使用して MBeanHome を取得します。『WebLogic JMX Service プログラマーズ ガイド』の「例 : アクティブなドメインとサーバの判別」を参照してください。

Security

guest および < 匿名 > ユーザ

WebLogic Server 6.x では、認証されないユーザ (匿名ユーザ) はすべて guest ユーザとして識別されました。また、guest ユーザは WebLogic リソースにアクセスできました。しかし、この機能はセキュリティ リスクを招くおそれがあったため修正されました。

新しいバージョンの WebLogic Server では、guest ユーザはデフォルトではなくなりました。WebLogic Server 7.0 では、匿名ユーザに <anonymous> という名前が割り当てられ、guest ユーザと匿名ユーザが区別されます。

WebLogic Server 6.x と同じように guest ユーザを使用するには、以下のいずれかの方法で行います。

- 互換性セキュリティを使用する (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)。
- guest ユーザを WebLogic 認証プロバイダでユーザとして定義する (WebLogic 認証プロバイダはすでにデフォルト セキュリティ レルムとしてコンフィグレーションされている)。WebLogic Server のインスタンスを起動するときに、以下の引数を設定して定義します。

  ```
  -Dweblogic.security.anonymousUserName=guest
  ```

警告 : この引数は、既存の WebLogic Server ユーザが効率的にセキュリティ機能をアップグレードできるように追加されたものです。guest ユーザをプロダクション環境で使用する場合には十分な注意が必要です。アップグレードの詳細については、『WebLogic Server 7.0 へのアップグレード』の「セキュリティのアップグレード」を参照してください。

サーブレット

次の新しいクラスを使用するように、web.xml ファイルを更新してください。

`weblogic.servlet.proxy.HttpClusterServlet`

以前のクラス

`weblogic.servlet.internal.HttpClusterServlet`

および

`weblogic.servlet.proxy.HttpProxyServlet`

以前のクラス

`weblogic.t3.srvr.HttpProxyServlet`

アプリケーション デプロイメントのスコープ内でサーブレットを使用した際の問題への対処については、1-25 ページの「デプロイメント」を参照してください。

スレッド プール サイズ

WebLogic Server 6.0 では、ワーカ スレッドの数はサーバ MBean の `ThreadPoolSize` パラメータで指定しました。WebLogic Server 6.1 からは、ワーカ スレッドの数はサーバ MBean の `ExecuteQueue` で定義します。

WebLogic Server 7.0 ではこのパラメータの移植パスが提供されるので、それが `config.xml` で指定される場合、またはコマンドラインでクライアントまたはサーバに渡される場合 (`-Dweblogic.ThreadPoolSize=<xx>`)、`ThreadPoolSize` が自動的に `ThreadCount` 設定に移行されます。

ワーカ スレッド数は次のように設定します。

WebLogic Server 6.x の場合

```
<Server
Name="myserver"
ThreadPoolSize="23"
...
/Server>
```

WebLogic Server 7.0 以降

```
<Server
Name="myserver"
... >
<ExecuteQueue
Name="default"
ThreadCount="23" />
/Server>
```

コンソールからスレッド数の値を変更するには、次の手順を行います。

1. コンソールで [サーバ | myServer | モニタ] を選択します。
2. [すべてのアクティブなキューのモニタ] をクリックします。
3. 「default」キューをクリックします (スレッドとそれらのスレッドの処理内容のリストが表示されます)。
4. [Execute Queue のコンフィグレーション] (ページの最上部) をクリックします。
5. 「default」キューをクリックします。
6. このサーバと関連付けられているスレッド数を入力します。
7. サーバを再起動して変更を有効にします。

Web アプリケーション

`weblogic.management.runtime.ServletRuntimeMBean.getName()` API (WebLogic Server 6.0) は、WebLogic Server 6.1 および 7.0 で `weblogic.management.runtime.ServletRuntimeMBean.getServletName()` に変更されています。このインタフェースを使用する場合は、ソース コードを更新して再コンパイルする必要があります。

Java サーブレット仕様 2.3 では、転送時の認可がデフォルトでは行われなくなりました。セキュアなリソースへの転送時に認可を得るには、`<check-auth-on-forward>` を `weblogic.xml` ファイルに追加します。

サーブレットの Request オブジェクトと Response オブジェクトには新しい API があります。それらのオブジェクトのシリアライズ可能で軽量な一部の実装は、新しい API を実装しないとコンパイルされない可能性があります。新しいサーブレット 2.3 モデルを使用し、サーブレットの Request オブジェクトと Response オブジェクトの実装を置き換えることをお勧めします。WebLogic Server 6.0 でこれを行った場合は、これらのオブジェクトの文書化されていない内部実装に従っていたものと思われます。WebLogic Server 7.0 ではサーブレット 2.3 がサポートされているので、新しい ServletRequest/ResponseWrapper オブジェクトを利用できます。

Solaris での WebLogic Server クラスタ

Solaris 上の WebLogic Server クラスタにデプロイされた一部のアプリケーション (重い EJB アプリケーション) は、サーバ JVM ではなくクライアント JVM を使用することでパフォーマンスが向上します。このことは、負荷の大きな状況で特に顕著になります。

Web サービス

WebLogic Server のバージョン 6.1 から 7.0 の間に、Web サービスの実行時システムとアーキテクチャが変化したことにより、バージョン 6.1 で作成した Web サービスをバージョン 7.0 上で実行するにはアップグレードが必要です。

WebLogic Server 6.1 に組み込まれていた WebLogic Web サービス クライアント API は非推奨になり、この API を使用して 7.0 の Web サービスを起動することはできません。WebLogic Server 7.0 には、Java API for XML-based RPC (JAX-RPC) をベースとした新しいクライアント API が組み込まれています。

バージョン 6.1 の WebLogic Web サービスから 7.0 へのアップグレードについては、「WebLogic Web サービス 6.1 から 7.0 へのアップグレード」を参照してください。

JAX-RPC を使用して WebLogic Web サービスを起動する例については、「Web サービスの呼び出し」を参照してください。

バージョン 6.1 と 7.0 の Web サービスの違いについては、「WebLogic Web サービスの概要」を参照してください。

書き込み可能な config.xml ファイル

WebLogic Server 7.0 では、バージョン 6.x の `config.xml` ファイルから読み取られたコンフィグレーション情報が、WebLogic Server 7.0 の情報を取り込んで自動的に更新されます。サーバの複数回の起動の間にこれらの変更を保持するためには、`config.xml` ファイルを書き込み可能に設定する必要があります。ファイルを書き込み可能にするには、バージョン 6.x のコンフィグレーションから `config.xml` ファイルのバックアップ コピーを作成し、ファイル属性を変更します。

非推奨になった API と機能

- WebLogic Time サービスは非推奨であり、JMX Timer サービスで置き換えることをお勧めします。JMX Timer サービスについては、「Interface TimerMBean」および「Class Timer」を参照してください。
- WebLogic ワークスペース
- Zero Administration Client (ZAC) は、WebLogic Server 7.0 のこのリリースでは非推奨となっています。これは JavaWebStart で置き換えられています。
- WebLogic Enterprise Connectivity (WLEC) は、WebLogic Server 7.0 のこのリリースでは非推奨になっています。WLEC を使用する Tuxedo CORBA アプリケーションは、WebLogic Tuxedo Connector に移行することをお勧めします。詳細については、「WebLogic Tuxedo Connector」を参照してください。
- `weblogic.deploy` は WebLogic Server 7.0 のこのリリースでは非推奨であり、`weblogic.Deployer` に置き換えられています。詳細については、「WebLogic Server デプロイメント」の「デプロイメント ツールおよび手順」を参照してください。

- このバージョンの WebLogic Server では、`application.xml` ファイルのない複数の J2EE モジュールのデプロイは非推奨になりました。デプロイメントとしてディレクトリを選択する場合、ディレクトリには、1 つのスタンドアロン J2EE モジュール (EJB、Web アプリケーション、リソース アダプタ)、または `application.xml` に関連付けられた複数のモジュール (エンタープライズ アプリケーション) が格納されている必要があります。
- このリリースでは、リソース アダプタ デプロイヤが、必要に応じて、指定した認証 / 認可メカニズムをセキュア パスワード資格ストレージを通して組み込むための、標準的な方法が提供されています。リソース アダプタ アーカイブ内の `weblogic-ra.xml` デプロイメント記述子で提供されていたセキュリティ プリンシパル マッピング メカニズムは、この WebLogic Server ストレージ メカニズムに変わっています。結果として、その構成要素が非推奨になっています。
- WebLogic Server の旧リリースで提供されていたパスワード変換ツールが非推奨になっています。
- 新しい接続リーク検出メカニズムに従って、`weblogic-ra.xml` デプロイメント記述子の `<connection-cleanup-frequency>` 要素と `<connection-duration-time>` 要素は非推奨になりました。
- WebLogic Server 7.0 では、WebLogic Server 6.x で使用された単一フェーズのデプロイメント モデルは非推奨となりました。WebLogic 6.x デプロイメント プロトコルのすべての API とツールは非推奨となりました。プロダクション アプリケーションのデプロイには使用しないでください。代わりに、WebLogic Server 7.0 のマニュアルで説明されている、新しい 2 フェーズ デプロイメント プロトコルの API とツールを使用してください。
- WebLogic jDriver for Microsoft SQL Server は非推奨となりました。WebLogic Server の将来のリリースでは削除されます。Microsoft SQL Server データベースに接続するには、Microsoft から入手できる JDBC ドライバを使用することをお勧めします。『WebLogic JDBC プログラマーズ ガイド』の「Microsoft SQL Server 2000 Driver for JDBC のインストールと使い方」を参照してください。
- WebLogic キーストア プロバイダは WebLogic Server 7.0 SP1 で非推奨となりました。

- `weblogic.management.tools.WebAppComponentRefreshTool` および `weblogic.refresh` はともに、WebLogic Server 7.0 のこのリリースでは非推奨となっています。これらは `weblogic.Deployer` に置き換えられています。
- WebLogic JDBC t3 ドライバ。「WebLogic File サービス (非推奨)」(http://edocs.beasys.co.jp/e-docs/wls/docs70/file/filesrvc.html#1028649) を参照してください。
- プログラムによるデプロイメントまたは `weblogic.Admin` コマンドによってアプリケーションおよびコンポーネント MBean を作成し、MBean の属性を設定し、それらの MBean の操作を呼び出して MBean をシステムにデプロイしていた場合、MBean の次の属性および操作が非推奨となっている点に注意してください。

  非推奨になった `ApplicationMBean` の操作

  `deploy`

  `load`

  `undeploy`

  非推奨になった `ApplicationMBean` の属性

  `LastModified`

  `LoadError`

  `isDeployed`

  これらの属性および操作に代わる要素については、WebLogic Server 7.0 javadoc の、`ApplicationMBean` インタフェースに関する説明箇所を参照してください。

削除された API と機能

WebLogic Enterprise Connectivity (WLEC) サンプルは削除されました。

2 WebLogic Server 4.5 および 5.1 からバージョン 7.0 へのアップグレード

WebLogic Server 4.5 および 5.1 からバージョン 7.0 へのアップグレードは、Web アプリケーションを定義し、`weblogic.properties` ファイルを新しい XML ファイル形式に変換する複数の手順からなるプロセスです。また、アップグレード プロセスに影響するいくつかの仕様上の変更も行われています。この章ではアップグレードに関連する問題の大半を扱っていますが、特定の環境に固有の問題については説明を省略している場合があります。

以下の節では、システムを WebLogic Server 4.5 または 5.1 から WebLogic Server 7.0 にアップグレードする場合、またアプリケーションを WebLogic Server 4.5 または 5.1 から WebLogic Server 7.0 に移植してデプロイする場合に必要な手順およびその他の情報を示します。説明する内容は、WebLogic Server 4.5 と 5.1 の両方から WebLogic Server 7.0 へのアップグレードを対象としています。

- 2-2 ページの「WebLogic Server コンフィグレーションのアップグレード : 主な手順」
- 2-4 ページの「WebLogic Server ライセンス ファイルのアップグレード」
- 2-6 ページの「weblogic.properties ファイルから XML ファイルへの変換」
- 2-9 ページの「WebLogic Server 7.0 でのクラスのロード」
- 2-9 ページの「起動スクリプトの修正」
- 2-10 ページの「WebLogic Server 7.0 の J2EE アプリケーション タイプ」
- 2-10 ページの「既存のアプリケーションの Web アプリケーションへの変換と移植」
- 2-18 ページの「エンタープライズ JavaBean アプリケーションの移植と変換」
- 2-26 ページの「エンタープライズ アプリケーションの作成」

- 2-27 ページの「J2EE クライアント アプリケーションについて」
- 2-28 ページの「JMS のアップグレード」
- 2-29 ページの「Oracle のアップグレード」
- 2-29 ページの「移植とデプロイメントに関するその他の考慮事項」

注意： このマニュアル全体を通して、「アップグレード」という用語は、WebLogic Server のより新しいバージョンへのアップグレードを指します。また「移植」という用語は、アプリケーションを WebLogic Server の古いバージョンからより新しいバージョンに移動することを指します。

WebLogic Server コンフィグレーションのアップグレード：主な手順

WebLogic Server 4.5 または 5.1 から WebLogic Server 7.0 にアップグレードするには、次の手順に従います。

1. アップグレードの手順を開始する前に、バージョン 4.5 または 5.1 のドメインのバックアップ コピーを作成します。WebLogic Server 7.0 クラスを使用してサーバを起動した後で、前のバージョンにダウングレードすることはできません。
2. WebLogic Server 7.0 をインストールします。『インストール ガイド』を参照してください。

   WebLogic Server 6.0 よりも前のバージョンでは、56 ビット暗号ではなく 128 ビット暗号を使用する場合に別途のダウンロードが必要でした。WebLogic Server 7.0 では、56 ビット暗号と 128 ビット暗号の両方に対応した単一のダウンロードが用意されています。128 ビット暗号を有効にする方法については、『インストール ガイド』の「128 ビット暗号の有効化」を参照してください。

   注意： 古いバージョンと同じ場所に新しいバージョンをインストールしようとすると、インストーラにより警告が出されます。

3. サーバのライセンス ファイルをアップグレードします。ライセンスを新しい形式に変換する方法については、2-4 ページの「WebLogic Server ライセンス ファイルのアップグレード」を参照してください。

4. `weblogic.properties` ファイルを変換します。`weblogic.properties` ファイルを変換する方法については、2-6 ページの「weblogic.properties ファイルから XML ファイルへの変換」および Administration Console オンライン ヘルプを参照してください。

5. コンソールに表示されたウィンドウで、新しい管理サーバの名前を入力します。この名前は、管理サーバのサーバ名として使われます。指定しない場合、名前はデフォルトの `myserver` になります。

6. 変換結果の出力が生成されるディレクトリを入力します。元のドメインを変換した結果として作成されるすべてのファイルおよびサブディレクトリは、ここで指定するディレクトリ内に配置されます。

7. Java システム `CLASSPATH` にクラスを追加します。詳細については、2-9 ページの「WebLogic Server 7.0 でのクラスのロード」を参照してください。

8. WebLogic Server 7.0 で動作するように既存の起動スクリプトを修正します。2-9 ページの「起動スクリプトの修正」を参照してください。

9. WebLogic サーバサイド ビジネス オブジェクト実装 (WebLogic Server 6.0 以降 Web アプリケーションと呼ばれる) を、WebLogic 7.0 上で実行するためにパッケージ化し、移植します。2-10 ページの「既存のアプリケーションの Web アプリケーションへの変換と移植」を参照してください。

10. EJB の移植が必要な場合は、2-18 ページの「エンタープライズ JavaBean アプリケーションの移植と変換」を参照してください。

11. JMS をアップグレードします。WebLogic Server 4.5 以降、多くの新しいコンフィグレーション属性が JMS に追加されています。詳細については、2-28 ページの「JMS のアップグレード」を参照してください。

サーバのクラスタをアップグレードするには、それぞれのサーバについて上記の手順に従い、その後、『WebLogic Server クラスタ ユーザーズ ガイド』の「WebLogic クラスタのセットアップ」で説明されている手順に従います。バージョン 4.5 と 7.0、または 5.1 と 7.0 の間では相互運用性がないため、クラスタを構成するすべてのサーバでバージョン 7.0 を実行していることが推奨されます。

WebLogic Server 7.0 から WebLogic Server 7.0.0.1 にアップグレードするには、『インストール ガイド』の「WebLogic Server 7.0 GA (7.0.0.0) から 7.0.0.1 へのアップデート」を参照してください。

注意： WebLogic Server 7.0 のディレクトリ構造は、4.5 および 5.1 のものと異なります。新しくなったディレクトリ構造の詳細については、『インストール ガイド』の「WebLogic Server のディレクトリ構造について」を参照してください。

WebLogic Server ライセンス ファイルのアップグレード

Java 形式のライセンス ファイル (`WebLogicLicense.class`) および XML 形式のライセンス ファイル (`WebLogicLicense.XML`) は、現在はサポートされていません。これらのファイルは以前のバージョンの WebLogic Server で使われていたもので、新しい形式に変換する必要があります。新しいライセンス ファイルは `license.bea` です。

ライセンス・アップグレードに際してのご注意

ライセンス・アップグレードは、お客様が製品を購入された販売元にご依頼ください。

お客様が「日本 BEA システムズ販売パートナ」から WebLogic Server をご購入された場合は、販売パートナへお問い合わせ、ご依頼ください。弊社販売パートナがライセンスのアップグレードを行い、新しいライセンスファイルをお届けいたします。

お客様が日本 BEA システムズ (株) から直接 WebLogic Server をご購入された場合は、日本 BEA システムズの営業担当者へご依頼ください。日本 BEA システムズよりアップグレードされたライセンスファイルをお届けいたします。

WebLogicLicense.class ライセンスの変換

既存の WebLogic Server インストールで `WebLogicLicense.class` ファイルを使用していた場合は、WebLogic Server 7.x をインストールする前に以下のタスクを実行します。

1. `licenseConverter` ユーティリティを使用して、`WebLogicLicense.class` ライセンス ファイルを `WebLogicLicense.XML` ファイルに変換します。
2. WebLogicLicense.XML ライセンスの変換で説明するように、`WebLogicLicense.XML` ファイルを変換します。

WebLogicLicense.XML ライセンスの変換

`WebLogicLicense.XML` ファイルを、WebLogic Server 6.x および 7.x と互換性のある `license.bea` ファイルに変換するには、以下の手順を実行します。この手順を実行するマシンで、`WebLogicLicense.XML` ライセンス ファイルが使用できることを確認してください。

1. BEA カスタマ サポートの Web サイト (`http://websupport.beasys.com/custsupp`) にログインします。
2. WebLogic Server ライセンスを更新するためのリンクをクリックします。リンクを表示するために下の方にスクロールした方がよいこともあります。
3. 変換するライセンス ファイルの入ったディレクトリのパス名を参照して選択するか、表示されるボックスにパス名を入力します。[Submit License] をクリックします。
4. 変換済みの `license_wls`*xx*`.bea` ファイルが電子メールで返送されます。システムにある `license.bea` ファイルを更新するには、『インストール ガイド』の「license.bea ファイルの更新」を参照してください。

weblogic.properties ファイルから XML ファイルへの変換

注意： このマニュアル全体を通して、ユーザが作成する WebLogic Server 7.0 のディレクトリ ドメインを `domain` と表記します。

WebLogic Server 6.0 以前のリリースでは、アプリケーションのコンフィグレーションに `weblogic.properties` ファイルを使用していました。WebLogic Server 7.0 では、アプリケーションのコンフィグレーションは XML 記述子ファイルと Administration Console を使用して処理します。`weblogic.properties` ファイルを `config.xml` ファイルに変換すると、アプリケーションの新しいドメインが作成され、アプリケーションのセットアップ内容を定義する XML ファイルが追加されます。変換の間に作成されるデフォルトのドメイン名は、アプリケーションの内容に関連性のある名前に変更することをお勧めします。

`config.xml` ファイルは、WebLogic Server ドメイン全体のコンフィグレーションを記述した XML ドキュメントです。`config.xml` ファイルは XML 要素群で構成されています。`domain` 要素はトップレベルの要素であり、domain 内のすべての要素は domain 要素の子です。`domain` 要素には `server`、`cluster`、`application` などの子要素が含まれます。これらの子要素の中にさらに子要素がある場合もあります。各要素には、コンフィグレーション可能な 1 つ以上の属性があります。

`weblogic.xml` ファイルには、Web アプリケーションで使われる WebLogic 固有の属性が格納されます。このファイルでは HTTP セッション パラメータ、HTTP クッキー パラメータ、JSP パラメータ、リソース参照、セキュリティ ロールの割り当て、文字セット マッピング、およびコンテナ属性の各属性を定義します。

デプロイメント記述子の `web.xml` ファイルは、Sun Microsystems のサーブレット 2.3 仕様で定義されています。`web.xml` ファイルは個々のサーブレットおよび JSP ページを定義し、Web アプリケーションで参照されるエンタープライズ Bean を列挙します。このデプロイメント記述子を使用して、J2EE 準拠のアプリケーション サーバに Web アプリケーションをデプロイできます。

既存の `weblogic.properties` ファイルを適切な XML ファイルに変換するには、次の手順に従います。

1. WebLogic Server 7.0 のサンプル サーバを起動します。WebLogic Server 7.0 のサンプル サーバを起動する方法については、「インストール後の作業の実行」の「サンプル サーバ、Pet Store サーバ、および Workshop サンプル サーバの起動」を参照してください。

   ユーザ名とパスワードの入力を求められます。

2. WebLogic Administration Console のホーム ページ (`http://localhost:7001/console/index.jsp` など) で、[ツール] の見出しの下にある [weblogic.properties をコンバート] リンクをクリックします。

3. コンソールのリンクを使用して、サーバのファイル システム内を移動し、以前のバージョンの WebLogic Server のルート ディレクトリ (`C:\weblogic` など) を見つけます。ルート ディレクトリが見つかったら、その隣にあるアイコンをクリックして選択します。

4. それ以外に、サーバ別の `weblogic.properties` ファイルやクラスタ化 `weblogic.properties` ファイルがほかのディレクトリにある場合は、表示されているウィンドウを使用してそれらのファイルを選択します。**以前のバージョンの WebLogic Server のルート ディレクトリが正しく選択されていれば、選択した追加の properties ファイルに関係なく、グローバルの `weblogic.properties` ファイルが変換されます。**

5. コンソールに表示されたウィンドウで、新しいドメインの名前を入力します。[コンバート] をクリックします。

properties ファイルを変換するとき、デフォルト Web アプリケーション用の `web.xml` および `weblogic.xml` ファイルが自動的に作成され、`domain\applications\DefaultWebApp_myserver\WEB-INF` ディレクトリの内部に配置されます。`weblogic.properties` ファイルを変換する過程では、`domain` ディレクトリに配置される `config.xml` ファイルも作成されます。このファイルには、ドメインに固有のコンフィグレーション情報が格納されます。

注意 : ここまでに説明した変換ユーティリティでは、`weblogic.xml` ファイルで Java ホームの場所を指定します。変換ユーティリティは `System.getProperty(java.home)` を使用してこの場所を読み取ります。つまり、変換のために WebLogic Server が起動された Java ホームの場所が指定されます。

- `config.xml` ファイルは直接編集しないことを強くお勧めします。コンフィグレーションの参照および編集は、Administration Console またはコマンドライン ユーティリティを使用して、あるいはコンフィグレーション API を通

じてプログラムから行うようにします。WebLogic Server Web コンポーネントのコンフィグレーションの詳細については、『管理者ガイド』の「WebLogic Server Web コンポーネントのコンフィグレーション」を参照してください。

- セキュリティ プロパティは *domain* ディレクトリに位置する `fileRealm.properties` ファイルに格納されます。WebLogic Server 7.0 でのセキュリティ レルムの使い方については、「デフォルト セキュリティ コンフィグレーションのカスタマイズ」を参照してください。
- `weblogic.properties` ファイルからプロパティを取得するためのメソッドを提供していた `weblogic.common.ConfigServicesDef` API は、このバージョンから削除されています。

`weblogic.properties` ファイルを変換する手順については、Administration Console オンライン ヘルプを参照してください。

以前に `weblogic.properties` のプロパティによって実行されていた機能を、`config.xml`、`web.xml`、または `weblogic.xml` のどの属性で処理するのかのリストについては、「weblogic.properties のマッピング表」を参照してください。

`weblogic.properties` ファイルの変換時に生成される起動スクリプトの名前は次のとおりです。

- `startdomainName.cmd` (Windows ユーザの場合)
- `startdomainName.sh` (UNIX ユーザの場合)

`domainName` は *`domain`* ディレクトリの名前です。

これらのスクリプトは、WebLogic Server 7.0 配布キットの *`domain`* ディレクトリに収められており、新しいドメインで管理サーバを起動します。

スクリプトとサーバの起動の詳細については、『管理者ガイド』の「WebLogic Servers の起動と停止」を参照してください。

WebLogic Server 7.0 でのクラスのロード

WebLogic Server の以前のバージョンでは、クラスの動的なロードを容易にするために WebLogic クラスパス プロパティ (`weblogic.class.path`) が使用されていました。WebLogic 6.0 以降のバージョンでは、`weblogic.class.path` は必要ありません。現在は、Java システム クラスパスからクラスをロードできます。

以前に `weblogic.class.path` で指定していたクラスを標準の Java システム クラスパスに含めるには、`CLASSPATH` 環境変数を設定するか、または次の例のようにコマンドライン上で `-classpath` オプションを使用します。

```
java -classpath %CLASSPATH%;%MyOldClassspath% weblogic.Server
```

%MyOldClasspath% には、古いアプリケーションを指し示すディレクトリだけを指定します。

起動スクリプトの修正

以前のバージョンで WebLogic Server 起動スクリプトを使用していた場合は、バージョン 7.0 で利用できるようにスクリプトを変更する必要があります。

- 『管理者ガイド』の「クラスパスの設定」の説明のとおりに、起動スクリプトを変更します。WebLogic クラスパスは使用されなくなりました。前の節の 2-9 ページの「WebLogic Server 7.0 でのクラスのロード」で説明したように、Java システム クラスパスを使用してください。
- WebLogic Server 7.0 はドメイン ディレクトリから起動します。起動スクリプトがドメイン ディレクトリからサーバを起動することを確認してください。
- クラスパスにライセンス ファイルを含める必要はなくなりました。
- 新しい管理システムでは、管理サーバと管理対象サーバが区別されます。そのため、サーバを起動するスクリプトは、ユーザが予定するサーバの管理方法に従って、記述し直す必要があります。新しいコマンドと必要な引数については、『管理者ガイド』の「WebLogic Server の起動と停止」を参照してください。

WebLogic Server 7.0 の J2EE アプリケーション タイプ

WebLogic Server 7.0 などの J2EE 準拠のサーバ上で動作するアプリケーションは、Web アプリケーション、エンタープライズ JavaBean、エンタープライズ アーカイブ、およびクライアント アプリケーションの 4 つのタイプのいずれかとして作成およびデプロイされます。既存のコンポーネントを WebLogic Server 7.0 に移植するには、適切な J2EE デプロイメント ユニットを作成します。J2EE デプロイメント ユニットの詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』の「エンタープライズ アプリケーションの一部としての Web アプリケーションのデプロイ」を参照してください。Web アプリケーションは通常はサーブレット、JSP、および HTML ファイルの集合であり、WAR ファイルとしてパッケージ化されます。(JAR ファイルとしてパッケージ化される) エンタープライズ JavaBean は、EJB 仕様に従って記述されるサーバサイド Java コンポーネントです。エンタープライズ アーカイブ (EAR ファイル) には、アプリケーションのすべての JAR および WAR コンポーネント アーカイブ ファイルと、ひとまとめにされるコンポーネント群を記述する XML 記述子が格納されます。クライアント アプリケーションは、Remote Method Invocation (RMI) を使用して WebLogic Server に接続する Java クラスです。前述の J2EE デプロイメント ユニットについては、後の節で詳しく説明します。

既存のアプリケーションの Web アプリケーションへの変換と移植

アプリケーションを Web アプリケーションに変換し、WebLogic Server 7.0 上にデプロイされる Web アプリケーションに移植するためには、特定のパターンに従うディレクトリ構造の内部にアプリケーションの構成ファイル群を配置する必要があります。開発段階では、これらのファイルはどのようなディレクトリ形式で配置しておいてもかまいません。ただし、プロダクション段階では、アプリケーションを WAR ファイルに 1 つの Web アプリケーションとしてまとめることを強くお勧めします。Web アプリケーションの詳細については、『WebLogic

Server アプリケーションの開発』の「WebLogic Server J2EE アプリケーションについて」および『Web アプリケーションのアセンブルとコンフィグレーション』を参照してください。

以下の節では、WebLogic Server 5.1 から WebLogic Server 7.0 に単純なサーブレットを移植する手順など、Web アプリケーションの移植およびデプロイメントについて知っておく必要がある情報を示します。

- 2-11 ページの「Web アプリケーションのディレクトリ構造」
- 2-12 ページの「XML デプロイメント記述子」
- 2-13 ページの「WAR ファイル」
- 2-14 ページの「Web アプリケーションのデプロイメント」
- 2-15 ページの「セッションの移植」
- 2-15 ページの「JavaServer Pages (JSP) とサーブレット」
- 2-16 ページの「WebLogic Server 5.1 から WebLogic Server 7.0 への単純なサーブレットの移植」

Web アプリケーションのディレクトリ構造

Web アプリケーションは、アーカイブ化して WebLogic Server 上にデプロイできるように、あらかじめ定められたディレクトリ構造に整理されます。Web アプリケーションに属するすべてのサーブレット、クラス、静的ファイル、およびその他のリソースはディレクトリ階層の下に整理されます。この階層構造のルートは、Web アプリケーションのドキュメント ルートを定義します。このルート ディレクトリの下に置かれたファイルは、ルート ディレクトリ内の `WEB-INF` および `META-INF` という特別なディレクトリの下にあるファイルを除いて、すべてクライアントに対して何らかの働きをします。ルート ディレクトリの名前は Web アプリケーションと同じにすることが推奨されます。

次の図は、Web アプリケーションのディレクトリ構造を示したものです。

```
WebApplicationRoot\(.jsp、.html、.jpg、.gif などの
                   |  公開されるファイル)
                   |
                   +WEB-INF\-+
                             |
                             + classes\(Web アプリケーションによって
```

```
|          使われるサーブレットなどの
|          Java クラスを格納する
|        格納するディレクトリ )
|
+ lib\(Web アプリケーションによって
|        使われる JAR ファイルを
|        格納するディレクトリ )
|
+ web.xml
|
+ weblogic.xml
```

weblogic.properties ファイルを変換すると、domain\applications\DefaultWebApp_myserver\WEB-INF ディレクトリの下に、適切な weblogic.xml ファイルと weblogic.xml ファイルが自動的に作成されます。上記のディレクトリ構造に従って、ユーザが作成する domain\applications\webAppName\WEB-INF ディレクトリに XML ファイルを配置します。Web アプリケーションのデプロイメントの詳細については、『WebLogic Server アプリケーションの開発』を参照してください。

XML デプロイメント記述子

Web アプリケーションのデプロイメント記述子 (web.xml) ファイルは標準の J2EE 記述子であり、サーブレットの登録、サーブレット初期化パラメータの定義、JSP タグ ライブラリの登録、セキュリティ制約の定義、およびその他の Web アプリケーション パラメータの定義に使用されます。デプロイメント記述子を作成する手順については、『Web アプリケーションのアセンブルとコンフィグレーション』の「web.xml デプロイメント記述子の記述」を参照してください。

WebLogic 固有のデプロイメント記述子 (weblogic.xml) もあります。このファイルでは、JSP プロパティ、JNDI のマッピング、セキュリティ ロールのマッピング、および HTTP セッション パラメータを定義します。WebLogic 固有のデプロイメント記述子では、web.xml ファイルで指定されたリソースが WebLogic Server のどのリソースにどのようにマップされるのかも定義します。WebLogic 固有のデプロイメント記述子を作成する手順については、「WebLogic 固有のデプロイメント記述子の記述」を参照してください。前述のプロパティ、マッピング、またはパラメータが不要な場合には、このファイルが必要でない場合があります。

web.xml ファイルと weblogic.xml ファイルは、アプリケーションをコンフィグレーションするために Administration Console と組み合わせて使用します。XML ファイルの内容は任意のテキスト エディタで表示できます。ファイルの内容を編集するには、必要な変更を行ってから、2-11 ページの「Web アプリケーションのディレクトリ構造」に示したディレクトリ構造によって規定される適切なパスに web.xml または weblogic.xml の名前でファイルを保存します。詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』を参照してください。アプリケーション群を 1 つの Web アプリケーションとしてまとめてデプロイしない場合は、自動的に作成された XML ファイルを分割し、各 Web アプリケーションに固有の適切な XML ファイルを作成する必要があります。各 Web アプリケーションには、アプリケーションで使用するように選択したファイルに加えて、weblogic.xml ファイルと web.xml ファイルが必要です。

WAR ファイル

WAR ファイルは Web アプリケーションのアーカイブです。前に説明した Web アプリケーションのディレクトリ構造に正しく従っており、適切な web.xml ファイルと weblogic.xml ファイルが作成済みである場合、プロダクション環境ではできる限り、アプリケーション群を WAR ファイルとしてデプロイされる 1 つのアプリケーションにまとめるようにしてください。アプリケーションを WAR ファイルにまとめた後は、WebLogic Server 上の各アプリケーションのインスタンスが 1 つだけになるように、以前のディレクトリ構造を削除することが重要です。

WAR ファイルを作成するには、Web アプリケーションのあるルート ディレクトリで次のコマンドラインを使用します。「*webAppName*」の部分は、Web アプリケーションに対して設定した独自の名前に置き換えてください。

```
jar cvf webAppName.war *
```

これにより、Web アプリケーションのすべてのファイルおよびコンフィグレーション情報を格納した WAR ファイルが作成されます。

Web アプリケーションのデプロイメント

WebLogic Server Administration Console を使用して Web アプリケーションをコンフィグレーションおよびデプロイするには、次の操作を行います。

1. WebLogic Server Administration Console を起動します。
2. 作業を行うドメインを選択します。
3. Administration Console の左ペインで、[デプロイメント] をクリックします。
4. Administration Console の左ペインで [アプリケーション] をクリックします。Administration Console の右ペインに、すべてのデプロイメント済みアプリケーションを示すテーブルが表示されます。
5. [新しい Application のコンフィグレーション] オプションを選択します。
6. アプリケーション アーカイブ、または展開されたアプリケーションが格納されたディレクトリの場所を確認します。WebLogic Sever は、指定したディレクトリおよびその下位ディレクトリで見つかった全コンポーネントをデプロイします。
7. ディレクトリまたはファイルの左のアイコンをクリックして選択し、次の操作に進みます。
8. 表示されたフィールドにアプリケーションまたはコンポーネントの名前を入力して、[作成] をクリックします。
9. 以下の情報を入力します。

   [ステージング モード] — ステージング モードを指定します。

   [デプロイ] — 表示されているチェックボックスを使用して、作成時に .ear、.war、.jar、または .rar ファイルをデプロイするかどうかを示します。
10. アプリケーションのコンポーネントをコンフィグレーションするために、[このアプリケーションのコンポーネントをコンフィグレーション] をクリックします。
11. [コンポーネント] テーブルが表示されます。コンフィグレーションするコンポーネントをクリックします。
12. 使用できるタブで、以下の情報を入力します。

[コンフィグレーション] — ステージング モードを編集し、デプロイメント順を入力します。

[対象] — アプリケーションを [選択可] リストから [選択済み] リストに移動することによって、このコンフィグレーション対象のアプリケーションの対象サーバを指定します。

[デプロイ] — 選択したすべての対象にアプリケーションをデプロイするか、またはすべての対象からアプリケーションをアンデプロイします。

[モニタ] — アプリケーションに関連するモニタ情報を表示します。

[メモ] — アプリケーションについての注記事項を入力します。

13. [適用] をクリックします。

コンソールでの属性の設定については、Administration Console オンライン ヘルプの「Web アプリケーション」の節を参照してください。

セッションの移植

WebLogic Server 6.0 でクッキーの形式が変更されたため、WebLogic Server 7.0 では、バージョン 6.0 以前のクッキーが認識されません。古い形式のクッキーは無視され、新しいセッションが作成されます。新しいセッションは自動的に作成される点に注意してください。

クッキーのデフォルト名はバージョン 5.1 から変更されています (以前の名前は `WebLogicSession`)。WebLogic Server 7.0 では、クッキーのデフォルト名は `JSESSIONID` です。

詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』の「weblogic.xml デプロイメント記述子の要素」を参照してください。

JavaServer Pages (JSP) とサーブレット

この節では、アプリケーションで使用できる JSP およびサーブレットに固有の情報を示します。

- サーブレットおよび JSP がデフォルト Web アプリケーション以外の Web アプリケーションにデプロイされるとき、Java コードおよび HTML コードの

中で記述されている URL に多少の変更が必要になる場合があります。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』の「管理とコンフィグレーション」を参照してください。コード内で相対 URL を使用しており、すべてのコンポーネントが同じ Web アプリケーションに格納される場合には、URL の変更は不要です。

- アプリケーションで分散型の構成を想定している場合、シリアライズ可能なオブジェクトだけをセッションに格納できます。
- `weblogic.properties` ファイル内のプロパティを、`web.xml` または `weblogic.xml` ファイル内の `XML` 形式の属性に変換する必要があります。このプロセスの詳細については、Administration Console オンライン ヘルプの変換に関する節を参照してください。
- ACL によるアクセス制御は、Web アプリケーションのデプロイメント記述子に定義されたアクセス制御に基づくセキュリティ制約に置き換えられています。
- サーバサイドインクルードはサポートされていません。この機能は JSP を使用して実現する必要があります。
- WebLogic Server 7.0 は、サーブレット 2.3 仕様に完全に準拠しています。
- 次の新しいクラスを使用するように、web.xml ファイルを更新してください。

  `weblogic.servlet.proxy.HttpClusterServlet`

  以前のクラス

  `weblogic.servlet.internal.HttpClusterServlet`

  および

  `weblogic.servlet.proxy.HttpProxyServlet`

  以前のクラス

  `weblogic.t3.srvr.HttpProxyServlet`

WebLogic Server 5.1 から WebLogic Server 7.0 への単純なサーブレットの移植

次の手順では、WebLogic Server 5.1 で提供されていた単純な Hello World サーブレットを WebLogic Server 7.0 に移植します。

1. WebLogic Server 7.0 で、『WebLogic HTTP サーブレット プログラマーズ ガイド』の「管理とコンフィグレーション」での説明に従って正しいディレクトリ構造を作成します。この作業ではルート アプリケーション ディレクトリ (C:\hello など) を作成し、その下に C:\hello\WEB-INF ディレクトリと C:\hello\WEB-INF\classes ディレクトリを作成します。HelloWorld.Servlet.java ファイルを C:\hello\WEB-INF\classes ディレクトリに配置します。

2. このサーブレットの web.xml ファイルを作成します。
   weblogic.properties ファイルを変換した場合、web.xml ファイルは既に自動的に作成されています。weblogic.properties ファイルを変換する前にこのファイルに HelloWorldServlet を登録した場合、サーブレットは新しい web.xml ファイル内で適切にコンフィグレーションされます。XML ファイルは任意のテキスト エディタを使って作成できます。HelloWorldServlet で使用できる基本的な web.xml ファイルの例を次に示します。

   ```
   <!DOCTYPE web-app ( 完全な DOCTYPE 宣言についてはソースを参照 ...)>
   - <web-app>
   - <servlet>
   <servlet-name>HelloWorldServlet</servlet-name>
   <servlet-class>examples.servlets.HelloWorldServlet</servlet-cla
   ss>
   </servlet>
   - <servlet-mapping>
   <servlet-name>HelloWorldServlet</servlet-name>
   <url-pattern>/hello/*</url-pattern>
   </servlet-mapping>
   </web-app>
   ```

   web.xml ファイルの詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』の「Web アプリケーションのデプロイメント記述子の記述」を参照してください。weblogic.xml ファイルは、HelloWorld のようなスタンドアロンの単純なサーブレットでは必要ありません。

   weblogic.xml ファイルの詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』の「WebLogic 固有のデプロイメント記述子の記述」を参照してください。

3. web.xml ファイルを、domain\applications\DefaultWebApp_myserver\WEB-INF ディレクトリから C:\hello\WEB-INF\ ディレクトリに移動します。

4. 開発環境をセットアップし (『WebLogic Server アプリケーションの開発』の「開発環境の構築」を参照)、次のようなコマンドを使用して HelloWorldServlet をコンパイルします。

   ```
   C:\hello\WEB-INF\classes>javac -d  . HelloWorldServlet.java
   ```

   これにより、ファイルがコンパイルされ、正しいパッケージ構造が作成されます。

5. この時点で、次のコマンドを使用してサーブレットをアーカイブの WAR ファイルにまとめることができます。

   ```
   jar cvf hello.war *
   ```

   これにより、`hello.war` ファイルが作成されて `C:\hello` ディレクトリ内に配置されます。

6. この Web アプリケーションをインストールするには、サーバを起動して Administration Console を開きます。[はじめに] メニューの下にある [アプリケーションのインストール] を選択します。新しく作成した WAR ファイルを選択して [Upload] をクリックします。

   サーブレットがデプロイされ、コンソールの左ペインにある [デプロイメント] の下の [Web アプリケーション] ノードの下に表示されます。

7. サーブレットを呼び出すには、Web ブラウザのアドレス欄に `http://localhost:7001/hello/hello` と入力します。

この場合、`/hello/` はサーブレットのコンテキスト パスです。このパスは WAR ファイルの名前に基づいて決定され、この場合は `hello.war` です。2 番目の `/hello` は、`web.xml` ファイル内のサーブレット マッピング タグにマップされています。

エンタープライズ JavaBean アプリケーションの移植と変換

以降の節では、エンタープライズ JavaBean の移植および変換の手順について説明します。

EJB の移植に関する考慮事項

エンタープライズ JavaBean を WebLogic Server 7.0 に移植するときは、以下の事項を考慮してください。

- WebLogic Server バージョン 7.0 ではエンタープライズ JavaBean 1.1 と 2.0 の仕様がサポートされています。
- WebLogic 7.0 では WebLogic 5.1 に比べて、XML パーサによる XML デプロイメント記述子の解析がより厳格になっています。以前のバージョンでは問題にされなかったいくつかのエラーが許可されなくなっています。これについては、「WebLogic Server エンタープライズ Java Bean の概要」で説明されています。
- EJB 1.1 Bean は WebLogic Server 7.0 でデプロイできます。ただし、新しい Bean を作成する場合は、EJB 2.0 の仕様に従うことをお勧めします。EJB 1.1 Bean は、DDConverter ユーティリティを使用して EJB 2.0 Bean に変換できます。詳細については、『WebLogic エンタープライズ JavaBean プログラマーズ ガイド』の「DDConverter」を参照してください。
- EJB 1.0 のデプロイメント記述子は DDConverter ユーティリティを使用して EJB 2.0 形式にアップグレードできますが、その前にそれらの記述子をまず 1.1 形式にアップグレードする必要があります。WebLogic Server 5.1 のデプロイメント記述子を 7.0 形式にアップグレードすると、WebLogic Server 7.0 の新しい機能を利用できるようになります。DDConverter ユーティリティの詳細については、『WebLogic エンタープライズ JavaBean プログラマーズ ガイド』の「WebLogic Server EJB のユーティリティ」の節を参照してください。
- EJB 1.1 のファインダ式機能は現在はサポートされていません。EJB 1.1 の機能でサポートされていないのはこの機能だけです。
- Bean のデプロイメントについては、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』の「WebLogic Server コンテナ用の EJB のパッケージ化」の節で説明されています。
- EJB に対して `ejbc` コマンドがまだ実行されていない場合、Bean のデプロイ時に WebLogic Server 7.0 によって `ejbc` が自動的に実行されます。デプロイメントの前に `ejbc` コマンドで Bean をコンパイルする必要はありません。必要な場合は、起動時に `ejbc` を実行することもできます。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「WebLogic Server コンテナ用の EJB のパッケージ化」を参照してください。

- EJB のデプロイメントでは、`ejb-jar.xml` ファイルに標準のデプロイメント記述子が含まれます。`ejb-jar.xml` は、EJB 1.1 DTD (文書型定義) または EJB 2.0 DTD に準拠している必要があります。
- EJB のデプロイメントでは、`weblogic-ejb-jar.xml` ファイルが必要です。このファイルは、WebLogic Server EJB コンテナのコンフィグレーション情報が含まれる WebLogic Server 固有のデプロイメント記述子です。このファイルは、WebLogic Server 5.1 DTD または WebLogic Server 7.0 DTD に準拠している必要があります。
- コンテナ管理による永続性を使用するエンティティ Bean には、データベースへのマッピングを指定するために、WebLogic Server 5.1 CMP DTD、WebLogic Server 7.0 EJB 1.1 DTD、または WebLogic Server 7.0 EJB 2.0 DTD のいずれかに準拠した CMP デプロイメント記述子が必要です。
- WebLogic Server 7.0 の `max-beans-in-cache` パラメータは、データベースの同時実行性を高めるためにキャッシュに格納される Bean の最大数を制御します。以前のバージョンの WebLogic Server では、`max-beans-in-cache` は無視され、キャッシュのサイズは無制限でした。このパラメータの値をより大きく調整しなければならない場合があります。

EJB の移植に関する推奨事項

- `TxDataSource` を使用する。

  EJB では、常に `TxDataSource` からデータベース接続を取得することが推奨されます。それにより、EJB コンテナのトランザクション管理は JDBC 接続と連携でき、XA トランザクションをサポートすることもできます。

  WebLogic Server 7.0.x CMP デプロイメント記述子は `TxDataSource` をサポートするので、接続プールだけを指定する WebLogic Server 5.1 CMP デプロイメント記述子の代わりに使用することが推奨されます。

- 高速なコンパイラ `ejbc` を使用する。

  WebLogic Server の EJB コンパイラ (`weblogic.ejbc`) は Java コードを生成し、その後、このコードが Java コンパイラによってコンパイルされます。デフォルトでは、WebLogic Server は付属の JDK に含まれている `javac` コン

パイラを使用します。より高速な Java コンパイラを使用すると、EJB コンパイラの動作がずっと高速になります。`-compiler` オプションを使って、代わりに使用するコンパイラを次のように指定できます。

```
java weblogic.ejbc -compiler sj pre_AccountEJB.jar
AccountEJB.jar
```

- EJB を WebLogic Server 7.0 にデプロイする前にエラーを修正する。

  WebLogic Server 7.0 EJB コンパイラ (`ejbc`) には、以前のリリースの WebLogic Server にはなかった検証機能が追加されています。このため、以前のバージョンの WebLogic Server でエラーなくデプロイされた EJB でも、WebLogic Server 7.0 ではエラーが報告される可能性があります。それらのエラーは、EJB を WebLogic Server 7.0 にデプロイする前に修正する必要があります。

  たとえば、WebLogic Server 7.0 では、あるメソッド名に対してトランザクション属性が設定されている場合に、そのメソッドが存在することを保証します。このことは、存在しないメソッドに対してトランザクション属性が誤って設定されている、などのよくあるエラーを識別するために役立ちます。

- WebLogic Server 7.0 のサンプルを参照する。

  WebLogic Server の配布キットには、デプロイメント記述子を含むいくつかの EJB サンプルが収録されています。EJB のサンプルは、WebLogic Server 7.0 配布キットの `samples\server\src\examples\ejb11` および `samples\server\src\examples\ejb20` ディレクトリにあります。

  次の表は、WebLogic Server 7.0 でサポートされている記述子の組み合わせを示しています。

表 2-1

EJB のバージョン	WebLogic Server のバージョン	CMP のバージョン
既存の WebLogic Server 5.1 デプロイメントは、以下の組み合わせを使用し、記述子またはコードを変更することなく WebLogic Server 7.0 でデプロイできる。		
1.1	5.1	5.1

表 2-1

EJB のバージョン	WebLogic Server のバージョン	CMP のバージョン
以下の組み合わせでは、WebLogic Server 7.0 CMP デプロイメント記述子を利用する。WebLogic Server 7.0 EJB 1.1 CMP デプロイメント記述子を利用すると、複数の EJB を 1 つの EJB JAR ファイルで指定でき、EJB が 2 フェーズ トランザクションまたは XA トランザクションに関与する場合に必要な `TxDataSource` を使用できる。		
1.1	5.1	7.0
1.1	7.0	7.0
EJB 2.0 Bean では常に WebLogic Server 6.x または 7.0 形式のデプロイメント記述子を使用する。		
2.0	6.x	7.0
2.0	7.0	7.0

エンタープライズ JavaBean の詳細については、「エンタープライズ JavaBean コンポーネント」および『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。

1.0 EJB を WebLogic Server 4.5.x から WebLogic Server 7.0 に移植する手順

WebLogic Server 3.1.x、4.0.x、および 4.5.x では、EJB 1.0 仕様がサポートされていました。WebLogic Server 4.5 から WebLogic Server 7.0 へ 1.0 形式の EJB を移植するには、次の手順に従います。

1. EJB 1.0 デプロイメント記述子を EJB 1.1 または EJB 2.0 XML デプロイメント記述子に変換します。この変換は、DDConverter ツールを使用して自動的に行うことができます。
2. 1 つ前の手順でのデプロイメント記述子の出力と、Bean のクラスを格納する JAR ファイルにデプロイメント記述子をパッケージ化します。

3. WebLogic Server EJB コンパイラ (ejbc) を実行して JAR ファイルをコンパイルします。`ejbc` ツールを使用してコンパイルされる EJB は、EJB 1.1 または EJB 2.0 仕様に準拠することが保証されます。
4. EJB コンテナに EJB をデプロイする前に準拠エラーを修正します。

EJB 1.1 または 2.0 に確実に準拠させるために、EJB 1.0 Bean に対して次の変更を行います。

- EJB 1.0 Bean では、`SessionContext` または `EntityContext` が一時的なものとして参照されていました。EJB 1.1 または 2.0 Bean をデプロイする場合、参照は一時的にはなりません。次に例を示します。

  ```
  private transient SessionContext ctx;
  ```

  これは次のように修正する必要があります。

  ```
  private SessionContext ctx;
  ```

- EJB 1.0 CMP エンティティ Bean の `ejbCreate` メソッドには、void 型の戻り値がありました。EJB 1.1 または 2.0 Bean をデプロイするとき、戻り値は主キー クラスでなければなりません。これにより、Bean 管理による永続性を使用する エンティティ Bean を記述し、その Bean を CMP 実装によってサブクラス化することが可能になります。次に例を示します。

  ```
  public void ejbCreate (String name) {
  firstName = name;
  }
  ```

  これは次のように修正する必要があります。

  ```
  public AccountPK ejbCreate (String name) {
  firstName = name;
  return null; // EJB 仕様により必須
  }
  ```

- EJB 1.1 または 2.0 では、エンティティ Bean で Bean 管理のトランザクションを使用できません。代わりに、コンテナ管理のトランザクション属性 (Required、Mandatory など) を指定して Bean を実行する必要があります。

1.1 EJB を WebLogic Server 5.1 から WebLogic Server 7.0 に移植する手順

WebLogic Server 5.1 デプロイメント記述子では、排他的または読み込み専用の同時実行性オプションだけを使用できます。データベース同時実行性オプションは、WebLogic Server 7.0 の `weblogic-ejb-jar.xml` ファイルにアップグレードすると指定できるようになります。このオプションの詳細については、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』の「weblogic-ejb-jar.xml 文書型定義」にあるデータベース同時実行性に関する情報を参照してください。

WebLogic Server 7.0 CMP デプロイメント記述子では複数の EJB を指定でき、接続プールの代わりに `TxDataSource` を使用できます。EJB 1.1 CMP で XA が使われているときは、`TxDataSource` を使用する必要があります。

WebLogic Server 5.1 から WebLogic Server 7.0 へ 1.1 形式の EJB を移植するには、次の手順に従います。

1. Administration Console を起動します。ホーム ページの [はじめに] メニューから [アプリケーションのインストール] をクリックします。
2. [参照] ボタンをクリックして移植する JAR ファイルを選択し、[開く]、[Upload] の順にクリックします。これで、Bean が WebLogic Server 7.0 に自動的にデプロイされます。
3. クライアント ウィンドウで `setEnv` スクリプトを実行し、開発環境を設定します (詳細については、『WebLogic Server アプリケーションの開発』の「開発環境の構築」を参照)。
4. 必要なすべてのクライアント クラスをコンパイルします。たとえば、WebLogic Server 7.0 で提供されているサンプルのステートレス セッション Bean の場合は、次のコマンドを使用します。

   ```
   javac -d %CLIENTCLASSES% Trader.java TraderHome.java
   TradeResult.java Client.java
   ```

5. クライアントを実行するには、次のコマンドを入力します。

   ```
   java -classpath %CLIENTCLASSES%;%CLASSPATH%
   examples.ejb.basic.statelessSession.Client
   ```

このコマンドでは、EJB インタフェースがクライアントのクラスパスで参照されることが保証されます。

EJB 1.1 を EJB 2.0 に変換する手順

EJB 1.1 Bean を EJB 2.0 Bean に変換するために、WebLogic Server の DDConverter ユーティリティを使用できます。

WebLogic Server 7.0 では、EJB 2.0 Bean を開発することをお勧めします。既にプロダクション環境で使用されている EJB 1.1 Bean については、2.0 Bean に変換する必要はありません。EJB 1.1 Bean は、WebLogic Server 7.0 でデプロイできます。EJB 1.1 Bean を 2.0 Bean に変換する場合、『WebLogic エンタープライズ JavaBeans プログラマーズ ガイド』の「DDConverter」を参照して、この変換を行う方法についての情報を確認してください。

単純な CMP 1.1 Bean を 2.0 Bean に変換するための基本手順を以下に示します。

1. Bean クラスを抽象クラスにします。EJB 1.1 Bean を Bean の CMP フィールドで宣言します。CMP 2.0 Bean は、各フィールドに対応する抽象メソッドの `getXXX` および `setXXX` を使用します。たとえば、EJB 1.1 Bean は `public String name` を使用します。EJB 2.0 Bean は、`public abstract String getName()` と `public abstract void setName(String n)` を使用します。この変更を行うと、Bean クラスでのコンテナ管理フィールドの読み込みに `getName` メソッドが、またコンテナ管理フィールドの更新に `setName` メソッドが使用されるようになります。

2. `java.util.Enumeration` を使用していたすべての CMP 1.1 ファインダは、`java.util.Collection` を使用する必要があります。CMP 2.0 ファインダは `java.util.Enumeration` を返すことができません。このことを反映して、コードを変更してください。

   ```
   public Enumeration findAllBeans()
       Throws FinderException, RemoteException;
   ```

   このコードを次のように変更します。

   ```
   public Collection findAllBeans()
       Throws FinderException, RemoteException;
   ```

その他の J2EE アプリケーション サーバからの EJB の移植

WebLogic Server 7.0 EJB コンテナには、EJB 1.1 仕様または EJB 2.0 仕様に準拠したすべての EJB をデプロイできます。各 EJB の JAR ファイルには、`ejb-jar.xml` ファイル、`weblogic-ejb-jar.xml` デプロイメント記述子、および CMP デプロイメント記述子 (CMP エンティティ Bean を使用する場合) が必要です。WebLogic Server 配布キットの `samples\examples\ejb11` および `samples\examples\ejb20` ディレクトリにある WebLogic Server EJB サンプルには、サンプルの weblogic デプロイメント記述子が含まれています。

エンタープライズ アプリケーションの作成

エンタープライズ アプリケーションは、拡張子が EAR の JAR ファイルです。EAR ファイルには、アプリケーションのすべての JAR および WAR コンポーネント アーカイブ ファイルと、ひとまとめにされるコンポーネント群を記述する XML 記述子が格納されます。`META-INF\application.xml` デプロイメント記述子には、個々の Web モジュールおよび EJB モジュールのエントリのほか、セキュリティ ロールや、データベースなどのアプリケーション リソースを定義する補助エントリがあります。

```
EnterpriseApplicationStagingDirectory\
                                      |
                                      + .jar ファイル
                                      |
                                      + .war ファイル
                                      |
                                      +META-INF\-+
                                                 |
                                                 + application.xml
```

EAR ファイルを作成するには、次の手順に従います。

1. アプリケーションを構成するすべての WAR ファイルと JAR ファイルをアセンブルします。

2. WAR ファイルと EJB JAR ファイルをステージング ディレクトリにコピーし、アプリケーションの `META-INF\application.xml` デプロイメント記述子を作成します。このとき、上の図に示したディレクトリ構造に従います。

   `application.xml` ファイルには、Sun Microsystems が提供する DTD を使用して、アプリケーションの各コンポーネント用の記述子が定義されます。application.xml ファイルの詳細については、『WebLogic Server アプリケーションの開発』の「アプリケーション デプロイメント記述子の要素」を参照してください。JSP を使用していて、JSP を実行時にコンパイルする場合、WAR ファイルの classes ディレクトリにある Bean のホーム インタフェースとリモート インタフェースが必要です。

3. ステージング ディレクトリで次のような jar コマンドを実行して、エンタープライズ アーカイブを作成します。

   ```
   jar cvf myApp.ear *
   ```

4. コンソールのホーム ページの [はじめに] という見出しの下にある [アプリケーションのインストール] リンクをクリックし、EAR ファイルを *`domain`*`\applications` ディレクトリに配置します。エンタープライズ アプリケーションの詳細については、『WebLogic Server アプリケーションの開発』の「エンタープライズ アプリケーションのパッケージ化」を参照してください。

J2EE クライアント アプリケーションについて

WebLogic Server では、標準の XML デプロイメント記述子を使用して JAR ファイルにパッケージ化された J2EE クライアント アプリケーションがサポートされています。このコンテキストのクライアント アプリケーションは、Web ブラウザではないクライアントです。それらのクライアント アプリケーションは、Remote Method Invocation (RMI) を使用して WebLogic Server に接続する Java クラスです。Java クライアントからは、エンタープライズ JavaBean、JDBC 接続、メッセージングなどのサービスに RMI を使用してアクセスできます。クライア

ント アプリケーションの種類は、標準入出力を使用する単純なコマンドライン ユーティリティから、Java Swing/AWT クラスを使用して構築される、対話型の高度な GUI アプリケーションまでさまざまです。

WebLogic Server Java クライアントを実行するには、クライアント コンピュータの側にクライアント アプリケーション クラスだけでなく、`weblogic_sp.jar` ファイル、`weblogic.jar` ファイル、および WebLogic Server 上のすべての RMI クラスとエンタープライズ Bean に対するリモート インタフェースが必要になります。保守とデプロイメントを簡略化するために、クライアントサイド アプリケーションは、クライアントのクラスパスに追加できる JAR ファイルに `weblogic.jar` および `weblogic_sp.jar` ファイルとともにパッケージ化したほうがよいでしょう。`weblogic.ClientDeployer` コマンドライン ユーティリティは、この仕様に基づいてパッケージ化されたクライアント アプリケーションを実行するためにクライアント コンピュータ上で実行されます。J2EE クライアント アプリケーションの詳細については、『WebLogic Server アプリケーションの開発』の「クライアント アプリケーションのパッケージ化」を参照してください。

JMS のアップグレード

WebLogic Server 7.0 は、JavaSoft の JMS 仕様バージョン 1.0.2 をサポートしています。

- WebLogic Server 4.5.1 — 移植は SP15 だけを対象にサポートされています。すべてのサービス パックを適用している場合は、BEA カスタマ サポートまでお問い合わせください。
- WebLogic Server 5.1 — SP07 または SP08 を適用している場合、既存の JDBC ストアをバージョン 7.0 に移植する前に BEA カスタマ サポートに連絡する必要があります。
  - オブジェクト メッセージを移植するには、オブジェクト クラスが WebLogic Server 7.0 のサーバ クラスパスに登録されている必要があります。
  - WebLogic Server 7.0 でコンフィグレーションされていない送り先については、移植されたメッセージは削除され、イベントがログに記録されます。

WebLogic JMS アプリケーションの移植の詳細については、『WebLogic JMS プログラマーズ ガイド』の「WebLogic JMS アプリケーションの移植」を参照してください。WebLogic Event は非推奨となっており、NO_ACKNOWLEDGE または MULTICAST_NO_ACKNOWLEDGE 配信モードの JMS メッセージで置き換えられています。それぞれの配信モードについては、『WebLogic JMS プログラマーズ ガイド』の「WebLogic JMS の基礎」で説明されています。

Oracle のアップグレード

BEA Systems では Oracle のサポート方針に従い、「動作確認状況」ページの「WebLogic jDriver JDBC ドライバのプラットフォーム サポート」に示されている Oracle のリリースをサポートしています。BEA では現在、バージョン 7.3.4、8.0.4、8.0.5、および 8.1.5 の Oracle Client をサポートしていません。

バージョン 7.3.4 の Oracle Client を使用するには、後方互換性のある oci816_7 共有ライブラリを使用します。既に述べているように、BEA では現在このコンフィグレーションをサポートしていません。

Oracle Client バージョン 9i にアップグレードする方法については、WebLogic jDriver および Oracle データベースのマニュアル、または『WebLogic jDriver for Oracle のインストールと使い方』の「WebLogic jDriver for Oracle のコンフィグレーション」を参照してください。

サポート対象のプラットフォーム、DBMS、およびクライアント ライブラリについては、BEA の「動作確認状況」ページを参照してください。「動作確認状況」ページでは、常に最新の動作確認情報を公開しています。

移植とデプロイメントに関するその他の考慮事項

以下の節では、WebLogic Server 7.0 でアプリケーションをデプロイする際に役立つ補足情報を示します。WebLogic Server 7.0 で非推奨となった機能、アップグレード、および重要な変更点を示しています。

注意： WebLogic Server 7.0 ではサンプル データベースとして PointBase 4.2 を使用しており、Cloudscape データベースは同梱されていません。

- 2-30 ページの「アプリケーションと管理対象サーバ」
- 2-31 ページの「デプロイメント」
- 2-31 ページの「プラグイン」
- 2-32 ページの「インターナショナライゼーション (I18N)」
- 2-32 ページの「Java Transaction API (JTA)」
- 2-33 ページの「Java Database Connectivity (JDBC)」
- 2-33 ページの「JSP」
- 2-34 ページの「JVM」
- 2-34 ページの「RMI」
- 2-35 ページの「セキュリティ」
- 2-37 ページの「セッションの移植」
- 2-37 ページの「スタンドアロンの HTML と JSP」
- 2-38 ページの「Web コンポーネント」
- 2-39 ページの「WAP アプリケーション」
- 2-40 ページの「書き込み可能な config.xml ファイル」
- 2-40 ページの「XML 7.0 パーサおよびトランスフォーマ」
- 2-41 ページの「非推奨となった API と機能」
- 2-42 ページの「削除された API と機能」

アプリケーションと管理対象サーバ

デフォルトでは、アプリケーションは管理サーバにデプロイされます。ただし、これはほとんどの場合適切な形態ではありません。管理サーバは管理目的にのみ使用することをお勧めします。Administration Console を使用して新しい管理対

象サーバを定義し、それらのサーバにアプリケーションを関連付けてください。詳細については、『WebLogic Server クラスタ ユーザーズ ガイド』および『管理者ガイド』の「WebLogic システム管理の概要」を参照してください。

デプロイメント

デフォルトでは、WebLogic Server バージョン 7.0 は 2 フェーズ デプロイメント モデルを使用します。このデプロイメント モデル、およびバージョン 7.0 のその他のデプロイメント機能の詳細については、『WebLogic Server アプリケーションの開発』の「WebLogic Server デプロイメント」を参照してください。7.0 のサーバに 4.5 または 5.1 のアプリケーションをデプロイする場合、デプロイメント モデルが指定されていないため、2 フェーズ デプロイメントを使用します。詳細については、『リリース ノート』を参照してください。

プラグイン

プラグインと WebLogic Server 4.5 および 5.1 の間の通信はクリア テキストです。WebLogic Server 7.0 のプラグインは、プラグインとバックエンドの WebLogic Server の間での SSL 通信をサポートしています。

プラグインをアップグレードするには、新しいプラグインを古いプラグインの上に上書きコピーし、IIS、Apache、または iPlanet Web サーバを再起動します。

FileServlet

WebLogic Server 6.1 のサービス パック 2 以降では、FileServlet (Web アプリケーションのデフォルト サーブレット) の動作が変更されています。現在の FileServlet では、ソース ファイル名を指定する際に SERVLET_PATH も指定できます。この設定によって、FileServlet を `/dir/*` などにマップすることで、特定のディレクトリからのファイルのみを明示的に提供できるようになりました。

「デフォルト サーブレットの設定」を参照してください。

インターナショナライゼーション (I18N)

このバージョンでは、インターナショナライゼーションとローカライゼーションに関連するいくつかの変更が行われています。

- ログ ファイル形式の変更は、メッセージのローカライズ方法に影響します。また新しいメッセージ形式では、最初の行に *begin marker*、*machine name*、*server name*、*thread id*、*user id*、*tran id*、および *message id* の各エントリが追加されています。
- インターナショナライズされた新しいログ API により、サーバおよびクライアントでメッセージをログに記録できるようになりました。
- クライアントはクライアントのログ ファイルにログを記録します。このファイルは、*servername* フィールドと *threadid* フィールドを除いて、サーバのログ ファイルと同じ形式です。
- `LogServicesDef` は非推奨になりました。代わりに、インターナショナライズされた API または `weblogic.logging.NonCatalogLogger` (インターナショナライゼーションが不要な場合) を使用してください。

このバージョンでのインターナショナライゼーションの詳細については、『インターナショナライゼーション ガイド』を参照してください。

Java Transaction API (JTA)

JTA の変更点は以下のとおりです。

- WebLogic Server 7.0 では、JTA 1.0.1 仕様がサポートされています。『WebLogic JTA プログラマーズ ガイド』で、JTA に関連する記述が更新されています。
- JTA のサポートの追加のため、JTS JDBC ドライバ (プロパティは `weblogic.jts.*` 内、URL は `jdbc:weblogic:jts:..`) が JTA JDBC/XA ドライバに置き換えられています。下位互換性のために既存のプロパティを使用できますが、JTS から JTA に名前が変更されたことに合わせてクラス名とプロパティを変更する必要があります。

Java Database Connectivity (JDBC)

JDBC の変更点は以下のとおりです。

- WebLogic T3 API は WebLogic Server 6.1 で非推奨となっています。代わりに RMI JDBC ドライバを使用します。この注意は、WebLogic Server 4.5.x からの移植の場合にも当てはまります。
- `weblogic.jdbc20.*` パッケージは、`weblogic.jdbc.*` パッケージへの置き換えが進んでいます。すべての WebLogic JDBC ドライバが JDBC 2.0 準拠になりました。
- アクティブな接続上で `preparedStatement` を使用していて、ストアド プロシージャが DBMS で削除された場合、ストアド プロシージャを作成するには新しい名前を使用します。同じ名前でストアド プロシージャを作成し直すと、`preparedStatement` は新しく作成されたストアド プロシージャにアクセスできません。新しく作成されたプロシージャは基本的に、同じ名前を持つ別のオブジェクトであるためです。

JSP

エラー処理

WebLogic Server 5.1 から現在のバージョンに至るまでの間に、JSP include ディレクティブの動作が変更されています。WebLogic Server 5.1 までのバージョンでは JSP include ディレクティブに存在しないページが含まれる場合、警告レベルのメッセージがログに記録されました。WebLogic Server 6.0 以降のバージョンでは、そうした場合に「500 Internal Server Error」が報告され、参照先となる場所に空のファイルを置くことでエラーを回避できます。

null の属性

JSP 仕様の変更により、null のリクエスト属性は空の文字列の代わりに文字列「null」を返すようになりました。バージョン 6.1 以降の WebLogic Server では、`weblogic.xml` に `printNulls` と呼ばれる新しいフラグがあります。デフォルト

ではこのフラグは true、すなわち「null」を返す設定になっています。printNulls を false に設定すると、式の結果が「null」になる場合に文字列「null」ではなく空の文字列が出力されます。

weblogic.xml における printNulls 要素のコンフィグレーションの例を次に示します。

```
<weblogic-web-app>
<jsp-param>
<param-name>printNulls</param-name>
<param-value>false</param-value>
</jsp-param>
</weblogic-web-app>
```

JVM

WebLogic Server 7.0 では、サーバのインストール時に JDK 1.3.1_02 の Java 仮想マシン (JVM) がインストールされます。サーバに付属する setenv.sh スクリプトはすべてこの JVM を指しています。動作確認された JVM についての最新情報は、「動作確認状況」ページで公開されています。

RMI

以下のヒントは、以前のバージョンの WebLogic Server で RMI を使用していたユーザが WebLogic Server 7.0 に移植する際のものです。

- 既存のコードで WebLogic RMI コンパイラ (weblogic.rmic) を再実行して、WebLogic Server 7.0 と互換性を持つようにラッパー クラスを再生成してください。
- java.rmi.Remote を使用して、インタフェースをリモートとしてタグ付けします。weblogic.rmi.Remote は使用しないでください。
- java.rmi.*Exception (import java.rmi.RemoteException; など) を使用します。weblogic.rmi.*Exception は使用しないでください。
- *.rmi.Naming の代わりに JNDI を使用します。

- `weblogic.rmic` を使用して、動的なプロキシとバイトコードを生成します。RMI IIOP の場合を除き、スタブ クラスとスケルトン クラスは生成されなくなりました。

注意： 詳細については、『WebLogic RMI プログラマーズ ガイド』の「WebLogic RMI の機能とガイドライン」を参照してください。

- `weblogic.rmi.server.UnicastRemoteObject.exportObject()` を使用してスタブのインスタンスを取得します。
- 現時点では、RMI のサンプルは、java.rmi.* と JNDI を使用するように更新されていません。将来のリリースで `java.rmi.*` と JNDI を反映するように見直される予定です。

セキュリティ

新しいセキュリティ アーキテクチャへのアップグレード

WebLogic Server 7.0 は新しいセキュリティ アーキテクチャを備えています。WebLogic Server 4.5 または 5.1 から WebLogic Server 7.0 のセキュリティ機能へのアップグレードは、以下の 2 つの手順で行います。

1. セキュリティ コンフィグレーションを WebLogic Server 6.x にアップグレードします。

   セキュリティ コンフィグレーションを WebLogic Server 4.5 または 5.1 から WebLogic Server 6.x にアップグレードする方法については、「WebLogic Server 4.5 および 5.1 アプリケーションのバージョン 6.x への移行」の「移行およびデプロイメントの補足事項」の「セキュリティ」を参照してください。

2. 6.x セキュリティ コンフィグレーションを WebLogic Server 7.0 にアップグレードします。

   詳細については、「WebLogic Server 6.x からバージョン 7.0 へのアップグレード」の「セキュリティのアップグレード」を参照してください。

WebLogic Server 7.0 の新しいセキュリティ アーキテクチャの詳細については、WebLogic Server 7.0 ドキュメントの「セキュリティ」を参照してください。

証明書サーブレットによって生成されるデジタル証明書

WebLogic Server 5.1 の Certificate Request Generator によって作成される CSR を通じて取得したデジタル証明書は、このリリースの WebLogic Server では使用できません。

WebLogic Server 5.1 の Certificate Request Generator を使用して CSR を作成した場合、サーブレットはプライベート キーのパスワードの指定を要求しません。プライベート キーおよび関連のデジタル証明書をこのリリースの WebLogic Server で使用するには、パスワードが必要です。

デジタル証明書のプライベート キー用パスワードを定義するには、JDK keytool を使用します。これで、このリリースの WebLogic Server でデジタル証明書を使えるようになります。keytool を使用してプライベート キーのパスワードを定義する前に、プライベート キーの各行末にある余分な文字を削除することが必要な場合があります。

プライベート キーとデジタル証明書

このリリースの WebLogic Server では、プライベート キーとデジタル証明書でより厳密なチェックが行われます。既存のプライベート キーとデジタル証明書を使用するためには、次のアップグレード手順を行う必要があります。

1. プライベート キーが暗号化されている場合は、`java utils der2pem` コマンドを使用して PEM フォーマットに変換し、ヘッダを次のように修正します。

   ```
   ----------BEGIN ENCRYPTED PRIVATE KEY----------
   ...
   -----------END RSA PRIVATE KEY---------------------
   ```

   プライベート キーが PEM フォーマットでない場合は、次の例外が送出されます。

   ```
   java.lang.Exception:Cannot read private key from file
   C:\bea700sp5\user_proects\mydomain\privatkey.der
   Make sure password specified in environnment property
   weblogic.management.pkpassword is valid.
   ```

   プライベート キーが暗号化されていない場合は、`java utils der.2pem` コマンドを使用し、ヘッダを次のように修正します。

   ```
   ----------BEGIN RSA PRIVATE KEY----------
   ...
   ----------END RSA PRIVATE KEY----------
   ```

2. デジタル証明書のファイルの最後に余分な行があるか確認します。証明書ファイルの最後の行は、次のようでなければなりません。

   ```
   ----------END CERTIFICATE----------
   ```

   余分な行はすべて削除します。

既存のプライベート キーがパスワードで保護されていない場合は、サーバの起動時に `weblogic.management.pkpassword` 引数を指定する必要はありません。

WebLogic Server Administration Console で SSL プロトコルをコンフィグレーションする際には、[暗号化キーを使用] 属性を使用してプライベート キーがパスワードで暗号化されるかどうかを指定しないことに注意してください。パスワードがプライベート キーのパスフレーズとして使用されない場合は、この属性は関係ありません。

変換したプライベート キーとデジタル証明書をキー ストアにインポートする場合は、`java utils.ImportPrivateKey` を使用します。

セッションの移植

バージョン 6.0 でクッキーの形式が変更されたため、WebLogic Server 6.0 以降では、以前のバージョンのクッキーが認識されません。WebLogic Server では、古い形式のクッキーは無視され、新しいセッションが作成されます。

クッキーのデフォルト名はバージョン 5.1 から変更されています (以前の名前は `WebLogicSession`)。WebLogic 6.0 以降は、クッキーのデフォルト名は `JSESSIONID` です。

詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』の「weblogic.xml デプロイメント記述子の要素」を参照してください。

スタンドアロンの HTML と JSP

WebLogic Server 7.0 で用意されているオリジナルのドメインと、`weblogic.properties` ファイル コンバータを使用して作成されたすべてのドメインでは、`domain\applications\DefaultWebApp_myserver` ディレクトリが作成されます。このディレクトリには、Web サーバが公開するファイルが格納されます。HTML ファイルと JSP ファイルは、インストールしたアプリケー

ションとは別にこの場所に配置して公開できます。必要な場合は、画像ファイルなどの相対リンクを処理するために、DefaultWebApp_myserver ディレクトリの内部にサブディレクトリを作成できます。

Web コンポーネント

以下のヒントは、以前のバージョンの WebLogic Server で Web コンポーネントを使用していて、WebLogic Server 7.0 に移植するユーザを対象としています。

- WebLogic Server のすべての Web コンポーネントで、WebLogic Server による JSP、サーブレット、および静的 HTML ページの公開形態を定義するメカニズムとして Web アプリケーションが使用されるようになりました。WebLogic Server の新規インストールでは、サーバでデフォルトの Web アプリケーションがコンフィグレーションされます。WebLogic Server 7.0 にアップグレードする場合、以前のバージョンで weblogic.properties ファイルを使用して行っていたドキュメント ルート、JSPServlet、およびサーブレットの登録に相当する処理をこのデフォルトの Web アプリケーションが行うため、改めて登録を行う必要はありません。
- Administration Console を使用して、既存の weblogic.properties ファイルを XML ファイルに変換します。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。
- SSI は現在ではサポートされていません。
- URL ACL は非推奨です。代わりにサーブレット 2.3 の機能を使用してください。
- 一部の情報が web.xml から weblogic.xml に移動されました。この再編成により、サーブレット 2.3 仕様に厳密に準拠するサード パーティの Web アプリケーションを、そのアプリケーションの J2EE 標準デプロイメント記述子 (web.xml) を修正することなくデプロイできるようになりました。下位互換性のために、<context-param> 要素を使用して web.xml ファイルで行われる WebLogic Server 5.1 スタイルの設定がサポートされていますが、デプロイメントは新しい方式で行うことが推奨されます。以前に web.xml で定義されていた以下のパラメータは、weblogic.xml で定義されるようになりました。

  JSP Parameters (keepgenerated、precompile compileCommand、

verbose、packagePrefix、pageCheckSeconds、encoding)

HTTP sessionParameters (CookieDomain、CookieComment、CookieMaxAgeSecs、CookieName、CookiePath、CookiesEnabled、InvalidationIntervalSecs、PersistentStoreDir、PersistentStorePool、PersistentStoreType、SwapIntervalSecs、IDLength、CacheSize、TimeoutSecs、JDBConnectionTimeoutSecs、URLRewritingEnabled)

- 詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』の「Web アプリケーションのデプロイメント記述子の記述」を参照してください。

WAP アプリケーション

WebLogic Server 7.0 上で WAP (Wireless Application Protocol) アプリケーションを実行するには、Web アプリケーションの web.xml ファイルで、WAP と関連付けられている MIME タイプを指定しなければならなくなりました。WebLogic Server 5.1 では、デフォルトの MIME タイプは weblogic.properties の weblogic.httpd.defaultMimeType を使用して設定できます (デフォルト値は「text/plain」)。WebLogic Server 6.0、WebLogic Server 6.1、および WebLogic Server 7.0 には、デフォルトの mime-type はありません。web.xml ファイルで拡張子ごとに明示的に mime-type を指定する必要があります。必要な MIME タイプの詳細については、『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。web.xml ファイルの作成および編集については、『Web アプリケーションのアセンブルとコンフィグレーション』の「Web アプリケーションのデプロイメント記述子の記述」を参照してください。

web.xml ファイルでの mime-types のコンフィグレーション例を以下に示します。

```
<web-app>
  <mime-mapping>
    <extension>tiff</extension>
    <mime-type>image/tiff</extension>
  </mime-mapping>
  <mime-mapping>
```

```
    <extension>tif</extension>
    <mime-type>image/tiff</extension>
  </mime-mapping>
</web-app>
```

書き込み可能な config.xml ファイル

WebLogic Server 7.0 では、バージョン 6.x の `config.xml` ファイルから読み取られたコンフィグレーション情報が、バージョン 7.0 の情報を取り込んで自動的に更新されます。サーバの複数回の起動の間にこれらの変更を保持するためには、`config.xml` ファイルを書き込み可能に設定する必要があります。ファイルを書き込み可能にするには、バージョン 6.0 のコンフィグレーションから `config.xml` ファイルのバックアップ コピーを作成し、ファイル属性を変更します。

XML 7.0 パーサおよびトランスフォーマ

WebLogic Server 7.0 の組み込みのパーサおよびトランスフォーマは、それぞれ Xerces 1.4.4 および Xalan 2.2 に更新されています。以前のバージョンの WebLogic Server に付属していた古いパーサおよびトランスフォーマに対応する API を使用していた場合や、非推奨となったクラス、インタフェース、またはメソッドを使用していた場合、非推奨であることを示すメッセージがアプリケーションで表示されることがあります。

WebLogic Server 7.0 には、WebLogic FastParser も付属します。これは、WebLogic Web サービスに関連付けられた SOAP および WSDL ファイルなど、中小規模のドキュメントを処理するために特別に設計された高性能の XML パーサです。アプリケーションで中小規模 (要素数が 10,000 個程度まで) の XML ドキュメントを処理することがほとんどの場合、FastParser を使用するように WebLogic Server をコンフィグレーションします。

WebLogic Server 7.0 の配布キットで、未修正版の Xerces パーサおよび Xalan トランスフォーマが *`WL_HOME`*`\server\ext\xmlx.zip` ファイルに含まれなくなりました。

非推奨となった API と機能

以下の API と機能は、将来的に製品から削除される予定なので使用しないでください。

- WebLogic Event

  WebLogic Event は非推奨であり、NO_ACKNOWLEDGE または MULTICAST_NO_ACKNOWLEDGE 配信モードの JMS メッセージで置き換えることをお勧めします。詳細については、『WebLogic JMS プログラマーズ ガイド』の「非トランザクション セッション」を参照してください。

- WebLogic HTMLKona
- WebLogic JDBC t3 ドライバ。「WebLogic File サービス (非推奨)」を参照してください。
- WebLogic Enterprise Connectivity
- WebLogic Time サービスは非推奨であり、JMX Timer サービスで置き換えることをお勧めします。JMX Timer サービスについては、「Interface TimerMBean」および「Class Timer」を参照してください。
- WebLogic ワークスペース
- Zero Administration Client (ZAC) は非推奨であり、JavaWebStart で置き換えることをお勧めします。
- `-Dweblogic.management.host`
- `weblogic.deploy` は WebLogic Server 7.0 のこのリリースでは非推奨であり、`weblogic.Deployer` に置き換えられています。詳細については、『WebLogic Server アプリケーションの開発』の「デプロイメント ツールおよび手順」を参照してください。
- `weblogic.management.tools.WebAppComponentRefreshTool` および `weblogic.refresh` はともに、WebLogic Server 7.0 のこのリリースでは非推奨となっています。これらは `weblogic.Deployer` に置き換えられています。

削除された API と機能

以下の API と機能は削除されています。

- 旧式の管理コンソール GUI
- デプロイヤ ツール
- WebLogic Bean
- WebLogic jHTML
- WebLogic Remote
- ワークスペース
- WebLogic Server Tour
- T3Client
- Jview サポート
- SSI
- Weblogic Bean バー
- RemoteT3
- Jview サポート
- Weblogic COM

  この機能では、現在はサポートされていない Microsoft JVM (Jview) を使用していました。

A weblogic.properties のマッピング表

weblogic.properties のマッピング表では、以前は weblogic.properties ファイルのプロパティで処理されていた機能を、config.xml、web.xml、または weblogic.xml ファイルのどの属性で処理するのかを示します。

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.administrator.email	config.xml: EmailAddress (Administrator 要素)	
weblogic.administrator.location	config.xml: Notes (自由形式、省略可能) (Administrator 要素)	
weblogic.administrator.name	config.xml: Name (Administrator 要素)	
weblogic.administrator.phone	config.xml: PhoneNumber (Administrator 要素)	
weblogic.cluster.defaultLoadAlgorithm	config.xml: DefaultLoadAlgorithm (Cluster 要素)	[クラスタ | *clustername* | コンフィグレーション | 一般 | デフォルトのロード バランス アルゴリズム]

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.cluster.multicastAddress	config.xml: MulticastAddress (Cluster 要素)	[クラスタ | *clustername* | コンフィグレーション | マルチキャスト | マルチキャスト アドレス]
weblogic.cluster.multicastTTL	config.xml: MulticastTTL (Cluster 要素)	[クラスタ | *clustername* | コンフィグレーション | マルチキャスト | マルチキャスト 生存時間]
weblogic.cluster.name	config.xml Cluster Address (Cluster 要素)	[クラスタ | *clustername* | コンフィグレーション | 一般 | クラスタ アドレス]
weblogic.httpd.authRealmName	config.xml: AuthRealmName (WebAppComponent 要素)	[デプロイメント | Web アプリケーション | *applicationname* | コンフィグレーション | その他 | 認証レルム名]
weblogic.httpd.charsets	config.xml: Charsets (WebServer 要素)	
weblogic.httpd.clustering.enable	config.xml: ClusteringEnabled (WebServer 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.httpd.defaultServerName	config.xml DefaultServerName (WebServer 要素)	[サーバ	*servername*	コンフィグレーション	HTTP	デフォルトサーバ名]
weblogic.httpd.defaultServlet	web.xml: 次の要素の URL パターンで servlet-mapping を定義する <servlet-mapping> 要素	
weblogic.httpd.defaultWebApp	config.xml: DefaultWebApp (WebServer 要素)	
weblogic.httpd.enable	config.xml: HttpdEnabled (Server 要素)	
weblogic.httpd.enableLogFile	config.xml: LoggingEnabled (WebServer 要素)	
weblogic.httpd.http.keepAliveSecs	config.xml: KeepAliveSecs (WebServer 要素)	
weblogic.httpd.https.keepAliveSecs	config.xml: HttpsKeepAliveSecs (WebServer 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.httpd.indexDirectories	config.xml: IndexDirectoryEnabled (WebAppComponent 要素)	[デプロイメント	Web アプリケーション	*applicationname*	コンフィグレーション	ファイル	インデックス ディレクトリ]
weblogic.httpd.keepAlive.enable	config.xml: KeepAliveEnabled (WebServer 要素)	[サーバ	*servername*	コンフィグレーション	HTTP	Keep Alive を有効化]
weblogic.httpd.logFileBufferKBytes	config.xml: LogFileBufferKBytes (WebServer 要素)	
weblogic.httpd.logFileFlushSecs	config.xml: LogFileFlushSecs (WebServer 要素)	
weblogic.httpd.logFileFormat	config.xml: LogFileFormat (WebServer 要素)	[サービス	仮想ホスト	ログファイル フォーマット]
weblogic.httpd.logFileName	config.xml: LogFileName (WebServer 要素)	[サービス	仮想ホスト	ログファイル名]
weblogic.httpd.logRotationPeriodMins	config.xml: LogRotationTimeBegin (WebServer 要素)	
weblogic.httpd.logRotationPeriodMins	config.xml: LogRotationPeriodMins (WebServer 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.httpd.logRotationType	config.xml: LogRotationType (WebServer 要素)	[サーバ	*servername*	ログ	HTTP	ローテーション タイプ]
weblogic.httpd.maxLogFileSizeKBytes	config.xml: MaxLogFileSizeKBytes (WebServer 要素)	[サーバ	*servername*	ログ	HTTP	最大ログ ファイル サイズ]
weblogic.httpd.mimeType	web.xml: mime-type (<mime-mapping> 要素)	
weblogic.httpd.postTimeoutSecs	config.xml: PostTimeoutSecs (WebServer 要素)	[サーバ	*servername*	コンフィグレーション	HTTP	POST タイムアウト秒]
weblogic.httpd.servlet.extensionCaseSensitive	config.xml: ServletExtensionCaseSensitive (WebAppComponent 要素)	[デプロイメント	Web アプリケーション	*applicationname*	コンフィグレーション	ファイル	大文字 / 小文字を区別する]
weblogic.httpd.servlet.reloadCheckSecs	config.xml: ServletReloadCheckSecs (WebAppComponent 要素)	[デプロイメント	Web アプリケーション	*applicationname*	コンフィグレーション	ファイル	再ロード間隔 (秒)]

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.httpd.servlet.SingleThreadedModelPoolSize	config.xml: SingleThreadedServletPoolSize (WebAppComponent 要素)	[デプロイメント | Web アプリケーション | *applicationname* | コンフィグレーション | ファイル | シングル スレッド サーブレットのプール サイズ]
weblogic.httpd.session.cacheEntries	weblogic.xml: CacheSize <param-name>/<param-value> 要素の組み合わせ	[サーバ | *servername* | コンフィグレーション | SSL | 証明書キャッシュ サイズ]
weblogic.httpd.session.cookie.comment	weblogic.xml: CookieComment <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.cookie.domain	weblogic.xml: CookieDomain <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.cookie.maxAgeSecs	weblogic.xml: CookieMaxAgeSecs <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.cookie.name	weblogic.xml: CookieName <param-name>/<param-value> 要素の組み合わせ	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.httpd.session.cookie.path	weblogic.xml: CookiePath <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.cookies.enable	weblogic.xml: CookiesEnabled <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.debug	weblogic.xml: SessionDebuggable <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.enable	weblogic.xml: SessionTrackingEnabled <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.invalidationintervalSecs	weblogic.xml: InvalidationIntervalSecs <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.jdbc.connTimeoutSecs	weblogic.xml: JDBCConnectionTimeoutSecs <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.persistentStoreDir	weblogic.xml: PersistentStoreDir <param-name>/<param-value> 要素の組み合わせ	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.httpd.session.persistentStorePool	weblogic.xml: PersistentStorePool <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.persistentStoreShared	weblogic.xml: SessionPersistentStoreShared <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.persistentStoreType	weblogic.xml: PersistentStoreType <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.sessionIDLength	weblogic.xml: IDLength <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.swapintervalSecs	weblogic.xml: SwapIntervalSecs <param-name>/<param-value> 要素の組み合わせ	
weblogic.httpd.session.timeoutSecs	weblogic.xml: TimeoutSecs <param-name>/<param-value> 要素の組み合わせ	[サーバ | *servername* | コンフィグレーション | HTTP | POST タイムアウト秒]
weblogic.httpd.session.URLRewriting.enable	weblogic.xml: URLRewritingEnabled <param-name>/<param-value> 要素の組み合わせ	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.httpd.tunneling.clientPingSecs	config.xml: TunnelingClientPingSecs (Server 要素)	[サーバ	*servername*	コンフィグレーション	チューニング	トンネリング クライアント Ping]
weblogic.httpd.tunneling.clientTimeoutSecs	config.xml: TunnelingClientTimeoutSecs (Server 要素)	[サーバ	*servername*	コンフィグレーション	チューニング	トンネリング クライアント タイムアウト]
weblogic.httpd.tunnelingenabled	config.xml TunnelingEnabled (Server 要素)	[サーバ	*servername*	コンフィグレーション	チューニング	トンネリングを有効化]
weblogic.httpd.URLResource	config.xml: URLResource (WebServer 要素)	
weblogic.iiop.password	config.xml: DefaultIIOPPassword (Server 要素)	[サーバ	*servername*	コンフィグレーション	プロトコル	デフォルト IIOP パスワード]
weblogic.iiop.user	config.xml: DefaultIIOPUser (Server 要素)	[サーバ	*servername*	コンフィグレーション	プロトコル	デフォルト IIOP ユーザ]

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.jdbc.connectionPool ■ url=JDBC ドライバの URL ■ driver=JDBC ドライバの完全パッケージ名 ■ loginDelaySecs= 接続間の秒数 ■ initialCapacity=JDBC 接続の初期数 ■ maxCapacity=JDBC 接続の最大数 ■ capacityIncrement= インクリメント間隔 ■ allowShrinking=true に設定すると縮小が許可される ■ shrinkPeriodMins= 縮小前の間隔 ■ testTable= 自動リフレッシュ テスト用のテーブルの名前 ■ refreshTestMinutes= 自動リフレッシュ テストの間隔 ■ testConnsOnReserve=true に設定すると予約時に接続をテストする ■ testConnsOnRelease=true に設定すると解放時に接続をテストする ■ props=JDBC 接続のプロパティ	URL DriveName LoginDelaySeconds InitialCapacity MaxCapacity CapacityIncrement AllowShrinking ShrinkPeriodMinutes TestTableName RefreshMinutes TestConnectionsOnReserve TestConnectionsOnRelease Properties JDBCConnectionPool Element ConnLeakProfilingEnabled ACLName CapacityEnabled SupportsLocalTransaction KeepLogicalConnOpenOnRelease Password	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.jdbc.enableLogFile	config.xml: JDBCLoggingEnabled (Server 要素)	
weblogic.jdbc.logFileName	config.xml: JDBCLogFileName (Server 要素)	
weblogic.jms.ConnectionConsumer	config.xml JMSConnectionConsumer 要素 MessagesMaximum Selector Destination	
weblogic.jms.connectionFactoryArgs.<<factoryName>> ■ ClientID ■ DeliveryMode ■ TransactionTimeout	config.xml: JMSConnectionFactory 要素 ClientID DefaultDeliveryMode TransactionTimeout UserTransactionsEnabled AllowCloseInOnMessage	
weblogic.jms.connectionFactoryName	config.xml: JMSConnectionFactory 要素 JNDIName	
weblogic.jms.connectionPool	ConnectionPool (JMSJDBCStore 要素)	
weblogic.jms.queue	config.xml: JNDIName StoreEnabled (JMSDestination 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.jms.queueSessionPool	config.xml: ConnectionConsumer ConnectionFactory ListenerClass AcknowledgeMode SessionsMaximum Transacted (JMSSessionPool 要素)	
weblogic.jms.tableNamePrefix	config.xml: PrefixName	
weblogic.jms.topic	config.xml JNDIName StoreEnabled (JMSDestination 要素)	[サービス	JMS	接続ファクトリ	JNDI 名]
weblogic.jms.topicSessionPool	config.xml: ConnectionConsumer ConnectionFactory ListenerClass AcknowledgeMode SessionsMaximum Transacted (JMSSessionPool 要素)	
weblogic.jndi.transportableObjectFactories	config.xml: JNDITransportableObjectFactoryList (Server 要素)	[サーバ	*servername*]
weblogic.login.readTimeoutMillisSSL	config.xml LoginTimeoutMillis (SSL 要素)	[サーバ	*servername*]
weblogic.security.audit.provider	config.xml AuditProviderClassName (Security 要素)	[セキュリティ	一般	監査プロバイダ クラス]

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.security.certificate.authority	config.xml ServerCertificateChainFileName (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	サーバ証明書チェーン ファイル名]
weblogic.security.certificate.server	config.xml: ServerCertificateFileName (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	サーバ証明書ファイル名]
weblogic.security.certificateCacheSize	config.xml: CertificateCacheSize (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	証明書キャッシュ サイズ]
weblogic.security.clientRootCA	config.xml: TrustedCAFileName (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	信頼性のある CA ファイル名]
weblogic.security.disableGuest	config.xml: GuestDisabled (Security 要素)	[セキュリティ	一般	ゲスト不可]
weblogic.security.enforceClientCert	config.xml: ClientCertificateEnforced (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	クライアント証明書を強制]

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.security.key.export.lifespan	config.xml: ExportKeyLifespan (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	エクスポートキーの有効期間]
weblogic.security.key.server	config.xml: ServerKeyFileName (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	サーバ キーファイル名]
weblogic.security.ldaprealm.authentication	config.xml: AuthProtocol (LDAPRealm 要素)	
weblogic.security.ldaprealm.credential	config.xml: Credential (LDAPRealm 要素)	
weblogic.security.ldaprealm.factory	config.xml LdapProvider (LDAPRealm 要素)	
weblogic.security.ldaprealm.groupDN	config.xml: GroupDN (LDAPRealm 要素)	
weblogic.security.ldaprealm.groupIsContext	config.xml: GroupIsContext (LDAPRealm 要素)	
weblogic.security.ldaprealm.groupNameAttribute	config.xml: GroupNameAttribute (LDAPRealm 要素)	
weblogic.security.ldaprealm.groupUsernameAttribute	config.xml: GroupUsernameAttribute (LDAPRealm 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.security.ldaprealm.principal	config.xml: Principal (LDAPRealm 要素)	
weblogic.security.ldaprealm.ssl	config.xml: SSLEnable (LDAPRealm 要素)	
weblogic.security.ldaprealm.url	config.xml: LDAPURL (LDAPRealm 要素)	
weblogic.security.ldaprealm.userAuthentication	config.xml: UserAuthentication (LDAPRealm 要素)	
weblogic.security.ldaprealm.userDN	config.xml: UserDN (LDAPRealm 要素)	
weblogic.security.ldaprealm.userNameAttribute	config.xml: UserNameAttribute (LDAPRealm 要素)	
weblogic.security.ldaprealm.userPasswordAttribute	config.xml: UserPasswordAttribute (LDAPRealm 要素)	
weblogic.security.net.connectionFilter	config.xml: ConnectionFilter (Security 要素)	
weblogic.security.ntrealm.domain	config.xml: PrimaryDomain (NTRealm 要素)	
weblogic.security.realm.cache.acl.enable	config.xml: ACLCacheEnable (CachingRealm 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.security.realm.cache.acl.size	config.xml: ACLCacheSize (CachingRealm 要素)	
weblogic.security.realm.cache.acl.ttl.negative	config.xml: ACLCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.acl.ttl.positive	config.xml: ACLCacheTTLPositive (CachingRealm 要素)	
weblogic.security.realm.cache.auth.enable	config.xml: AuthenticationCacheEnable (CachingRealm 要素)	
weblogic.security.realm.cache.auth.size	config.xml: AuthenticationCacheSize (CachingRealm 要素)	
weblogic.security.realm.cache.auth.ttl.negative	config.xml: AuthenticationCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.auth.ttl.positive	config.xml: AuthenticationCacheTTLPositive (CachingRealm 要素)	
weblogic.security.realm.cache.caseSensitive	config.xml: CacheCaseSensitive (CachingRealm 要素)	
weblogic.security.realm.cache.group.enable	config.xml: GroupCacheEnable (CachingRealm 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.security.realm.cache.group.size	config.xml: GroupCacheSize (CachingRealm 要素)	
weblogic.security.realm.cache.group.ttl.negative	config.xml: GroupCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.group.ttl.positive	config.xml: GroupCacheTTLPositive (CachingRealm 要素)	
weblogic.security.realm.cache.perm.enable	config.xml: PermissionCacheEnable (CachingRealm 要素)	
weblogic.security.realm.cache.perm.size	config.xml: PermissionCacheSize (CachingRealm 要素)	
weblogic.security.realm.cache.perm.ttl.negative	config.xml: PermissionCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.perm.ttl.positive	config.xml: PermissionCacheTTLPositive (CachingRealm 要素)	
weblogic.security.realm.cache.user.enable	config.xml: UserCacheEnable (CachingRealm 要素)	
weblogic.security.realm.cache.user.size	config.xml: UserCacheSize (CachingRealm 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.security.realm.cache.user.ttl.negative	config.xml: UserCacheTTLNegative (CachingRealm 要素)	
weblogic.security.realm.cache.user.ttl.positive	config.xml: UserCacheTTLPositive (CachingRealm 要素)	
weblogic.security.realm.certAuthenticator	config.xml: CertAuthenticator (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	証明書認証プロバイダ]
weblogic.security.SSL.ciphersuite	config.xml Ciphersuites (SSL 要素)	
weblogic.security.ssl.enable	config.xml: Enabled (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	有効化]
weblogic.security.SSL.hostnameVerifier	config.xml HostnameVerifier (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	ホスト名の検証]
weblogic.security.SSL.ignoreHostnameVerification	config.xml HostNameVerificationIgnored (SSL 要素)	
weblogic.security.SSLHandler.enable	config.xml: HandlerEnabled (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	ハンドラを有効化]

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.security.unixrealm.authProgram	config.xml: AuthProgram (UnixRealm 要素)	
weblogic.system.AdministrationPort	config.xml AdministrationPort (Server 要素)	[サーバ	*servername*	コンフィグレーション	一般	管理ポート]
weblogic.system.bindAddr	config.xml: ListenAddress (Server 要素)	
weblogic.system.defaultProtocol	config.xml: DefaultProtocol (Server 要素)	[サーバ	*servername*	コンフィグレーション	プロトコル	デフォルト プロトコル]
weblogic.system.defaultSecureProtocol	config.xml: DefaultSecureProtocol (Server 要素)	[サーバ	*servername*	コンフィグレーション	プロトコル	デフォルト セキュア プロトコル]
weblogic.system.enableConsole	config.xml: StdoutEnabled (Kernel 要素)	[サーバ	*servername*	ログ	一般	Stdout へログ出力]
weblogic.system.enableIIOP	config.xml: IIOPEnabled (Server 要素)	
weblogic.system.enableReverseDNSLookups	config.xml: ReverseDNSAllowed (Server 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.system.enableSetGID,	config.xml: PostBindGID	
weblogic.system.enableSetUID,	config.xml: PostBindUIDEnabled	
weblogic.system.enableTGIOP	config.xml TGIOPEnabled (Server 要素)	[サーバ	*servername*]
weblogic.system.helpPageURL	config.xml HelpPageURL (Server 要素)	[サーバ	*servername*]
weblogic.system.home	config.xml: RootDirectory (Server 要素)	
weblogic.system.ListenPort	config.xml ListenPort (Server 要素)	[サーバ	*servername*	コンフィグレーション	SSL	リスン ポート]
weblogic.system.logFile	config.xml: FileName (Log 要素)	
weblogic.system.MagicThreadBackToSocket	config.xml: MagicThreadDumpBackToSocket (ServerDebug 要素)	
weblogic.system.MagicThreadDumpFile	config.xml: MagicThreadDumpFile (ServerDebug 要素)	
weblogic.system.MagicThreadDumpHost	config.xml: MagicThreadDumpHost (ServerDebug 要素)	

weblogic.properties ファイルのプロパティ	.xml コンフィグレーション属性	コンソールの表示
weblogic.system.magicThreadDumps	config.xml: MagicThreadDumpEnabled (ServerDebug 要素)	
weblogic.system.maxLogFileSize	config.xml: FileMinxSize (Log 要素)	
weblogic.system.nativeIO.enable	config.xml: NativeIOEnabled (Server 要素)	[サーバ	*servername*	コンフィグレーション	チューニング	ネイティブ IO を有効化]
weblogic.system.nonPrivGroup	config.xml PostBindGID (UnixMachine 要素)	
weblogic.system.nonPrivUser	config.xml PostBindUID (UnixMachine 要素)	
weblogic.system.percentSocketReaders	config.xml: ThreadPoolPercentSocketReaders (Kernel 要素)	[サーバ	*servername*	コンフィグレーション	チューニング	ソケット リーダー]
weblogic.system.readTimeoutMillis	config.xml: LoginTimeoutMillis (Server 要素)	[サーバ	*servername*]
weblogic.system.SSL.useJava	config.xml: UseJava (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	Java を使用]

weblogic.properties ファイルのプロパティ	**.xml コンフィグレーション属性**	**コンソールの表示**
weblogic.system.SSLListenPort	config.xml: ListenPort (SSL 要素)	[サーバ	*servername*	コンフィグレーション	SSL	リスン ポート]
weblogic.system.startupFailureIsFatal	config.xml FailureIsFatal (StartupClass 要素)	
weblogic.system.user	config.xml: SystemUser (Security 要素)	
weblogic.system.weight	config.xml ClusterWeight (Server 要素)	[サーバ	*servername*	コンフィグレーション	クラスタ	クラスタの重み]

B Pet Store アプリケーションおよびサンプル サーバのアップグレード

この付録では、以下のプロセスの例を示します。

- Pet Store アプリケーションの WebLogic 6.1 サービス パック 3 から WebLogic Server 7.0 へのアップグレード
- WebLogic 6.0 サービス パック 2 サンプル サーバの WebLogic Server 7.0 へのアップグレード
- WebLogic 6.1 サービス パック 2 サンプル サーバの WebLogic Server 7.0 へのアップグレード

注意： WebLogic Server 7.0 のサンプルと Pet Store は、デフォルトのセキュリティ コンフィグレーションを使用するようにコンフィグレーションされています。WebLogic Server 7.0 のサンプルと Pet Store を互換性セキュリティで実行することはできません。

このドキュメントで使用する用語

説明に 3 つのバージョンがすべて登場する場合は、WebLogic ホーム ディレクトリに対してバージョンごとに個別の用語を使用します。この規則は、ドメイン コンフィグレーションの移植方法の説明をわかりやすくするために、このドキュメントで使用します。

このドキュメントでは、`WL_HOME` を WebLogic Server 6.x および 7.0 のホームとして定義します。

6.0 の場合 `WL_HOME=D:\WLS_6.0\wlserver6.0`

6.1 の場合 `WL_HOME=D:\WLS_6.1\wlserver6.1`

7.0 の場合 `WL_HOME=D:\WLS_7.0\weblogic700`

Pet Store アプリケーションの WebLogic 6.1 サービス パック 3 から WebLogic Server 7.0 へのアップグレード

WebLogic 6.1 サービス パック 3 から Pet Store アプリケーションを WebLogic Server 7.0 にアップグレードする必要はありません。この節では、アプリケーションを 6.1 から 7.0 へアップグレードする方法の例として、その手順を説明します。WebLogic 6.1 サービス パック 3 Pet Store アプリケーションを WebLogic Server 7.0 で使用するためにアップグレードするには、以下の作業を行います。

1. WebLogic Server 7.0 をインストールする .
2. 6.1 サービス パック 3 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する .
3. WebLogic Server 7.0 で Pet Store アプリケーションを起動する .

WebLogic Server 7.0 をインストールする

WebLogic Server 7.0 をインストールします。『インストール ガイド』を参照してください。

注意 : 古いバージョンと同じ場所に新しいバージョンをインストールしようとすると、インストーラにより警告が出されます。

6.1 サービス パック 3 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する

WebLogic 6.1 サービス パック 3 の Pet Store アプリケーションを WebLogic Server 7.0 にアップグレードするには、以下の作業を行います。

1. WebLogic Server 6.x では受け入れられていたが WebLogic Server 7.0 のパーサでは拒否される Pet Store のタグ ライブラリ エラーを修正します。
2. `WL_HOME/config/petstore` ディレクトリを、WebLogic Server 6.1 から、インストールされている WebLogic Server 7.0 内の場所にコピーします。次回のアップグレードの際にドメインを移動しなくても済むように、ディレクトリを `WL_HOME` にコピーしないことをお勧めします。

注意： Pet Store をコピーする WebLogic Server 7.0 のディレクトリが、Pet Store の WebLogic Server 6.1 でのディレクトリの位置と同じ場合は、新しいディレクトリ位置を反映するために `config.xml` ファイルを編集する必要はありません。異なるディレクトリ位置に Pet Store をコピーする場合は、`config.xml` ファイルですべてのファイルおよびディレクトリの絶対パスを見つけて、相対パスに変更する必要があります。

3. `startPetstore.cmd` スクリプトを編集して、新しい WebLogic Server 7.0 のインストールと、新しいディレクトリ位置 (存在する場合) を反映します。
4. `config.xml` ファイルを編集して、新しい WebLogic Server 7.0 のインストールと、新しいディレクトリ位置 (存在する場合) を反映します。

この節には以下のものが含まれています。

- Pet Store エラーを修正する手順 : B-4 ページの「JSP 解析エラーの修正」
- B-7 ページの「WebLogic Server 6.1 サービス パック 3 の起動に使用する startPetstore.cmd スクリプト」のサンプル。
- B-9 ページの「WebLogic Server 7.0 を起動するために変更された上記の startPetstore.cmd スクリプト」のサンプル。
- B-12 ページの「WebLogic Server 6.1 サービス パック 3 で使用する config.xml ファイル」のサンプル。

- WebLogic Server 7.0 で使用する上記の config.xml ファイルのサンプル。サンプル サーバを WebLogic Server 7.0 にアップグレードするために、このスクリプトを変更する方法の説明が含まれています。

注意： Pet Store アプリケーションを WebLogic Server 7.0 にアップグレードするために、weblogic.xml および web.xml ファイルの DTD を更新する必要はありません。WebLogic Server 7.0 の DTD 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「weblogic-ejb-jar.xml 文書型定義」を参照してください。

JSP 解析エラーの修正

以前のバージョンの WebLogic Server で解析可能だった軽微なエラーは、JDK 1.4 では受け入れられないため、WebLogic Server 8.1 においてはエラーの原因となります。この節で修正されるエラーは、メソッドとセッターのプロパティが一致しないプロパティ設定です。

エラーの修正には、次のソース ファイルに対する変更が必要となります。

```
ListTag.java
CartListTag.java
MyListTag.java
ProductItemListTag.java
ProductListTag.java
SearchListTag.java
```

これらのファイルはすべて、`WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\petstore\taglib\list` ディレクトリ (`WL_HOME` は WebLogic Server のインストール ディレクトリ) に置かれています。

`ListTag.java` で置換を行うには、次の手順を使用します。

1. コマンド コンソールで、`WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\petstore\taglib\list` に移動します。次に例を示します。

   ```
   C:\> cd
   WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\petstore\taglib\list
   ```

2. テキスト エディタで ListTag.java を開きます。次に例を示します。

```
WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprin
ts\petstore\taglib\list>notepad ListTag.java.
```

3. 次の箇所を変更します。

```
public void setNumItems(String numItemsStr) {
   numItems = Integer.parseInt(numItemsStr);
   }
```

この行を次のように変更します。

```
public void setNumItems(int numItemsIn) {
   numItems = numItemsIn;
   }
```

4. 次の箇所を変更します。

```
public void setStartIndex(String startIndexStr) {
   startIndex = Integer.parseInt(startIndexStr);
   }
```

この行を次のように変更します。

```
public void setStartIndex(int startIndexIn) {
   startIndex = startIndexIn;
   }
```

5. ListTag.java を保存して閉じます。

その他のファイルでの置換を、次のように行います。

1. コマンド コンソールで、WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprints\petstore\taglib\list に移動します。次に例を示します。

```
C:\>cd
WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprin
ts\petstore\taglib\list
```

2. テキスト エディタで CartListTag.java を開きます。次に例を示します。

```
WL_HOME\samples\petStore\src\petstore\src\com\sun\j2ee\blueprin
ts\petstore\taglib\list>notepad CartListTag.java.
```

3. CartListTag.java から、次の行を削除します。

```
public void setNumItems(String numItemsStr) {
   super.setNumItems(numItemsStr);
   }
  public void setStartIndex(String startIndexStr) {
   super.setNumItems(startIndexStr);
   }
```

4. 次の行と置き換えます。

```
public void setNumItems(int numItems) {
   super.setNumItems(numItems);
   }

public void setStartIndex(int startIndex) {
   super.setNumItems(startIndex);
   }
```

5. ProductListTag.java を保存して閉じます。
6. 残りのファイルについても、手順 1 から 5 を繰り返します。

   MyListTag.java
   ProductItemListTag.java
   ProductListTag.java
   SearchListTag.java

Pet Store の再構築

Pet Store に対する修正を行った後、アプリケーションを再構築します。

1. コマンド コンソールで、WebLogic Server 6.x の WL_HOME\config\examples ディレクトリに移動し、環境を設定します。

   ```
   WL_HOME\config\examples> setexamplesenv.cmd (or .sh)
   ```

2. 同じコンソール内で、WL_HOME\samples\petStore\src\petstore\src ディレクトリに移動して、次のように再構築を行います。

   ```
   WL_HOME\samples\petStore\src\petstore\src> build
   ```

スクリプトにより、petstore.ear が
WL_HOME\samples\petStore\src\petstore\build に構築されます。

WebLogic Server 6.1 サービス パック 3 の起動に使用する startPetstore.cmd スクリプト

```
@echo off

@rem This script can be used to start WebLogic Server for the
purpose
@rem of running the PetStore application. This script ensures
that the server is started
@rem using the config.xml file found in this directory and that
the CLASSPATH
@rem is set appropriately. This script contains the following
variables:
@rem
@rem JAVA_HOME     - Determines the version of Java used to start
@rem                   WebLogic Server. This variable must point
to the
@rem                   root directory of a JDK installation. and
will be set
@rem                   for you by the WebLogic Server installer.
Note that
@rem                    this script uses the hotspot VM to run
WebLogic Server.
@rem                 If you choose to use a JDK other than the one
@rem                  included in the disribution, make sure that
the JDK
@rem                   includes the hotspot VM. See the WebLogic
platform
@rem                    support page
(http://e-docs.bea.com/wls/platforms/index.html)
@rem                    for an up-to-date list of supported JVMs
on Windows NT.
@rem
@rem When setting these variables below, please use short file
names (8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
```

```
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server
(http://e-docs.bea.com/wls/docs61/install/index.html).

SETLOCAL

cd ..\..

@rem Set user-defined variables.
set JAVA_HOME=d:\610sp2\jdk131

@rem Check that script is being run from the appropriate
directory
if not exist lib\weblogic.jar goto wrongplace
goto checkJDK

@rem :wrongplace
@rem echo startPetStore.cmd must be run from the config\petStore
directory. 1>&2
@rem goto finish

:checkJDK
if exist "%JAVA_HOME%/bin/javac.exe" goto runWebLogic
echo.
echo Javac wasn't found in directory %JAVA_HOME%/bin.
echo Please edit the startPetStoreServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your JDK
installation.
goto finish

:runWebLogic
echo on
set PATH=.\bin;"%JAVA_HOME%\bin";%PATH%

set CLASSPATH=.;.\lib\weblogic_sp.jar;.\lib\weblogic.jar;.\
samples\eval\cloudscape\lib\cloudscape.jar;.\config\petStore\se
rverclasses
echo off

echo.
echo ************************************************
echo *  To start WebLogic Server, use the password     *
echo *  assigned to the system user.  The system       *
echo *  username and password must also be used to     *
echo *  access the WebLogic Server console from a web  *
echo *  browser.                                       *
echo ************************************************
```

```
@rem Set WLS_PW equal to your system password for no password
prompt server startup.
set WLS_PW=

@rem Set Production Mode.  When set to true, the server starts
up in production mode.
@rem When set to false, the server starts up in development mode.
The default is false.
set STARTMODE=true

echo on
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath
"%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="d:\610sp2" -Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="d:\610sp2\wlserver6.1/lib/weblogic.pol
icy" weblogic.Server
goto finish


:finish
cd config\petStore
ENDLOCAL
```

WebLogic Server 7.0 を起動するために変更された上記の startPetstore.cmd スクリプト

```
@echo off

@rem This script can be used to start WebLogic Server for the
purpose
@rem of running the PetStore application. This script ensures
that
@rem the server is started using the config.xml file found in
@rem this directory and that the CLASSPATH is set appropriately.
@rem This script contains the following variables:
@rem
@rem JAVA_HOME     - Determines the version of Java used to start
@rem                   WebLogic Server. This variable must point
to the
@rem                   root directory of a JDK installation. and
will be set
@rem                   for you by the WebLogic Server installer.
Note that
@rem                    this script uses the hotspot VM to run
```

```
WebLogic Server.
@rem                If you choose to use a JDK other than the one
@rem                included in the disribution, make sure that
the JDK
@rem                  includes the hotspot VM. See the WebLogic
platform
@rem                   support page
(http://e-docs.bea.com/wls/platforms/index.html)
@rem                   for an up-to-date list of supported JVMs
on Windows NT.
@rem
@rem When setting these variables below, please use short file
names (8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server
(http://e-docs.bea.com/wls/docs61/install/index.html).

SETLOCAL

cd ..\..

@rem Set user-defined variables.
@rem 1.  SET THE NEW JAVA HOME APPROPRIATELY
set JAVA_HOME=D:\70bea\jdk131

@rem 2.  FOR SIMPLICITY, CREATE AND SET BEA_HOME AND WL_HOME70
set BEA_HOME=d:\wls70
set WL_HOME70=%BEA_HOME%\weblogic700

@rem 3.  REMOVE THIS ENTIRE CHECK AND ITS TAG SINCE
@rem NEITHER IS RELEVANT ANY LONGER
@rem Check that script is being run from the appropriate
directory
@rem if not exist lib\weblogic.jar goto wrongplace
@rem goto checkJDK

@rem :wrongplace
@rem echo startPetStore.cmd must be run from the config\petStore
directory. 1>&2
@rem goto finish
```

```
:checkJDK
if exist "%JAVA_HOME%/bin/javac.exe" goto runWebLogic echo.
echo Javac wasn't found in directory %JAVA_HOME%/bin.
echo Please edit the startPetStoreServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your JDK
installation.
goto finish

@rem 4.  SET THE PATH VARIABLE APPROPRIATELY USING WL_HOME70 YOU
DEFINED IN STEP 2 ABOVE
:runWebLogic
echo on
set PATH=%WL_HOME70%\server\bin;"%JAVA_HOME%\bin";%PATH%

@rem 5.  SET YOUR CLASSPATH SO THE NEW WLS70 CLASSES ARE USED
WHILE RETAINING ALL CLASS LOCATIONS
@rem RELEVANT TO YOUR APPLICATION.  TO DO THIS, USE WL_HOME70 YOU
SET IN STEP 2.
set CLASSPATH=.;%WL_HOME70%\server\lib\weblogic.jar;.\samples\
eval\cloudscape\lib\cloudscape.jar;.\config\petStore\serverclas
ses
echo off

echo.
echo ************************************************
echo *  To start WebLogic Server, use the password     *
echo *  assigned to the system user.  The system       *
echo *  username and password must also be used to     *
echo *  access the WebLogic Server console from a web  *
echo *  browser.                                       *
echo ************************************************

@rem Set WLS_PW equal to your system password for no password
prompt server startup.
set WLS_PW=

@rem Set Production Mode.  When set to true, the server starts
up in production mode.
@rem When set to false, the server starts up in development mode.
The default is false.
set STARTMODE=true

@rem 6.  SET THE -Dbea.home COMMAND LINE OPTION USING THE
BEA_HOME VARIABLE YOU SET IN STEP 2.
echo on
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath
"%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="%BEA_HOME%"
-Dweblogic.management.password=%WLS_PW%
```

```
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="d:\610sp2\wlserver6.1/lib/weblogic.pol
icy" weblogic.Server
goto finish


:finish
cd config\petStore
ENDLOCAL
```

WebLogic Server 6.1 サービス パック 3 で使用する config.xml ファイル

```
<Domain
Name="petstore">
<JDBCTxDataSource
JNDIName="jdbc.EstoreDB"
Name="EstoreDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<JDBCTxDataSource
JNDIName="jdbc.InventoryDB"
Name="InventoryDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<JDBCTxDataSource
JNDIName="jdbc.SignOnDB"
Name="SignOnDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<Application
Deployed="true"
Name="tour"
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/tour.war">
<WebAppComponent
Name="tour"
Targets="petstoreServer"
URI="tour.war"/>
</Application>

<Application
Deployed="true"
Name="petstore"
```

```
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/petstore.ear">
<EJBComponent
Name="customerEjb"
Targets="petstoreServer"
URI="customerEjb.jar"/>

<EJBComponent
Name="inventoryEjb"
Targets="petstoreServer"
URI="inventoryEjb.jar"/>

<EJBComponent
Name="mailerEjb"
Targets="petstoreServer"
URI="mailerEjb.jar"/>
<EJBComponent
Name="personalizationEjb"
Targets="petstoreServer"
URI="personalizationEjb.jar"/>

<EJBComponent
Name="signonEjb"
Targets="petstoreServer"
URI="signonEjb.jar"/>

<EJBComponent
Name="shoppingcartEjb"
Targets="petstoreServer"
URI="shoppingcartEjb.jar"/>

<EJBComponent
Name="petstoreEjb"
Targets="petstoreServer"
URI="petstoreEjb.jar"/>

<WebAppComponent
Name="petstore"
Targets="petstoreServer"
URI="petstore.war"/>

</Application>

<Application
Deployed="true"
Name="petstoreAdmin"
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/petstoreAdmin.ear>

<EJBComponent
Name="petstoreAdminEjb"
```

```
Targets="petstoreServer"
URI="petstoreadminEjb.jar"/>

<WebAppComponent
Name="petstoreadmin"
Targets="petstoreServer"
URI="petstoreadmin.war"/>

</Application>

<Server
JavaCompiler="D:\WLS 6.1\jdk131/bin/javac"
ListenPort="7001"
Name="petstoreServer"
RootDirectory="D:\WLS 6.1\wlserver6.1"
ThreadPoolSize="15"
TransactionLogFilePrefix="config/petstore/logs/"
IIOPEnabled="false">

<WebServer
DefaultWebApp="tour"
LogFileName="./config/petstore/logs/access.log"
LoggingEnabled="true"
Name="petstoreServer"/>

<SSL
CertificateCacheSize="3"
Enabled="true"
ListenPort="7002"
ServerCertificateChainFileName="./config/petstore/ca.pem"
ServerCertificateFileName="./config/petstore/democert.pem"
ServerKeyFileName="./config/petstore/demokey.pem"
TrustedCAFileName="./config/petstore/ca.pem"

Ciphersuites="SSL_RSA_EXPORT_WITH_RC4_40_MD5,SSL_RSA_WITH_DES_C
BC_SHA,
SSL_RSA_EXPORT_WITH_DES_40_CBC_SHA,SSL_NULL_WITH_NULL_NULL"/>

<Log FileName="./config/petstore/logs/weblogic.log"/>

</Server>

<Log FileName="./config/petstore/logs/wl-domain.log"/>

<JDBCConnectionPool
CapacityIncrement="1"
DriverName="COM.cloudscape.core.JDBCDriver"
InitialCapacity="1"
MaxCapacity="1"
Name="petstorePool"
Properties="user=none;password=none;server=none"
Targets="petstoreServer"
URL="jdbc:cloudscape:petStore"/>
```

```
<FileRealm
Name="myFileRealm"/>

<Security
Realm="myRealm"/>

<Realm
FileRealm="myFileRealm"
Name="myRealm"/>

<MailSession
Name="mailSession"
Targets="petstoreServer"
JNDIName="mail.Session"

Properties="mail.from=orders@javapetstoredemo.com;mail.host=san
-francisco.beasys.com"/>

<StartupClass
Arguments="port=7001"
ClassName="com.bea.estore.startup.StartBrowser"
FailureIsFatal="false"
Name="StartBrowser"
Targets="petstoreServer"

Notes="On Windows, this class automatically starts a browser
after the server has finished booting."/>

</Domain>
```

WebLogic Server 7.0 で使用する上記の config.xml ファイル

1. 7.0 の petstore.ear ファイルを指すようにパスを変更します。

```
<Domain ConfigurationVersion="7.0.0.0"
Name="petstore"
Path="D:\700sp0\weblogic700\samples\server\config\petstore\appl
ications\
petstore.ear" StagedTargets="" TwoPhase="false">

<JDBCTxDataSource
JNDIName="jdbc.EstoreDB"
Name="EstoreDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<JDBCTxDataSource
JNDIName="jdbc.InventoryDB"
Name="InventoryDB"
PoolName="petstorePool"
Targets="petstoreServer"/>
```

```
<JDBCTxDataSource
JNDIName="jdbc.SignOnDB"
Name="SignOnDB"
PoolName="petstorePool"
Targets="petstoreServer"/>

<Application
Deployed="true"
Name="tour"


Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/tour.war">
<WebAppComponent
Name="tour"
Targets="petstoreServer"
URI="tour.war"/>
</Application>

<Application
Deployed="true"
Name="petstore"
Path="D:\WLS
6.1\wlserver6.1/config/petstore/applications/petstore.ear">
<EJBComponent
Name="customerEjb"
Targets="petstoreServer"
URI="customerEjb.jar"/>

<EJBComponent
Name="inventoryEjb"
Targets="petstoreServer"
URI="inventoryEjb.jar"/>

<EJBComponent
Name="mailerEjb"
Targets="petstoreServer"
URI="mailerEjb.jar"/>
<EJBComponent
Name="personalizationEjb"
Targets="petstoreServer"
URI="personalizationEjb.jar"/>

<EJBComponent
Name="signonEjb"
Targets="petstoreServer"
URI="signonEjb.jar"/>

<EJBComponent
Name="shoppingcartEjb"
Targets="petstoreServer"
URI="shoppingcartEjb.jar"/>
```

```
<EJBComponent
Name="petstoreEjb"
Targets="petstoreServer"
URI="petstoreEjb.jar"/>

<WebAppComponent
Name="petstore"
Targets="petstoreServer"
URI="petstore.war"/>

</Application>
```

2. petstoreAdmin.ear ファイルを指すようにパスを変更します。

```
<Application
Deployed="true"
Name="petstoreAdmin"
Path="D:\700sp0\weblogic700\samples\server\config\petstore\appl
ications\
petstoreAdmin.ear" StagedTargets="" TwoPhase="false">
<EJBComponent
Name="petstoreAdminEjb"
Targets="petstoreServer"
URI="petstoreadminEjb.jar"/>

<WebAppComponent
Name="petstoreadmin"
Targets="petstoreServer"
URI="petstoreadmin.war"/>

</Application>
```

3. WebLogic Server 7.0 Java コンパイラの場所を指すようにパスを変更します。

```
<Server
JavaCompiler="D:\700sp0\jdk131_02/bin/javac"
ListenPort="7001"
Name="petstoreServer"
RootDirectory="D:\700sp0"
ThreadPoolSize="15"
TransactionLogFilePrefix="config/petstore/logs/"
IIOPEnabled="false">

<WebServer
DefaultWebApp="tour"
LogFileName="./config/petstore/logs/access.log"
LoggingEnabled="true"
Name="petstoreServer"/>

<SSL
CertificateCacheSize="3"
Enabled="true"
```

```
ListenPort="7002"
ServerCertificateChainFileName="./config/petstore/ca.pem"
ServerCertificateFileName="./config/petstore/democert.pem"
ServerKeyFileName="./config/petstore/demokey.pem"
TrustedCAFileName="./config/petstore/ca.pem"

Ciphersuites="SSL_RSA_EXPORT_WITH_RC4_40_MD5,SSL_RSA_WITH_DES_C
BC_SHA,
SSL_RSA_EXPORT_WITH_DES_40_CBC_SHA,SSL_NULL_WITH_NULL_NULL"/>

<Log FileName="./config/petstore/logs/weblogic.log"/>

</Server>

<Log FileName="./config/petstore/logs/wl-domain.log"/>

<JDBCConnectionPool
CapacityIncrement="1"
DriverName="COM.cloudscape.core.JDBCDriver"
InitialCapacity="1"
MaxCapacity="1"
Name="petstorePool"
Properties="user=none;password=none;server=none"
Targets="petstoreServer"
URL="jdbc:cloudscape:petStore"/>

<FileRealm
Name="myFileRealm"/>

<Security
Realm="myRealm"/>

<Realm
FileRealm="myFileRealm"
Name="myRealm"/>

<MailSession
Name="mailSession"
Targets="petstoreServer"
JNDIName="mail.Session"

Properties="mail.from=orders@javapetstoredemo.com;mail.host=san
-francisco.beasys.com"/>

<StartupClass
Arguments="port=7001"
ClassName="com.bea.estore.startup.StartBrowser"
FailureIsFatal="false"
Name="StartBrowser"
Targets="petstoreServer"

Notes="On Windows, this class automatically starts a browser
after the server has finished booting."/>
```

```
</Domain>
```

WebLogic Server 7.0 で Pet Store アプリケーションを起動する

WebLogic Server 7.0 で Pet Store アプリケーションを起動するには、次の手順に従います。

1. 新しい Web ブラウザ ウィンドウを開きます。
2. http://localhost:7001/estore/index.html に移動します。
3. [**Enter the Store**] をクリックします。

WebLogic 6.0 サービス パック 2 サンプル サーバの WebLogic Server 7.0 へのアップグレード

WebLogic 6.0 サンプル サーバを WebLogic Server 7.0 にアップグレードする必要はありません。この節では、サーバを 6.0 から 7.0 へアップグレードする方法の例として、その手順を説明します。WebLogic 6.0 サンプル サーバのドメイン コンフィグレーションを WebLogic Server 7.0 で使用するためにアップグレードするには、以下の作業を行います。

- WebLogic Server 7.0 をインストールする
- 6.0 サービス パック 2 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する
- WebLogic Server 7.0 でサンプル サーバを起動する

WebLogic Server 7.0 をインストールする

WebLogic Server 7.0 をインストールします。『インストール ガイド』を参照してください。

注意： 古いバージョンと同じ場所に新しいバージョンをインストールしようとすると、インストーラにより警告が出されます。

6.0 サービス パック 2 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する

6.0 の `WL_HOME/config/examples` ディレクトリを 7.0 のディレクトリにコピーするときは、引き続き、config ディレクトリ内に examples ドメイン ディレクトリが含まれるようにすることが大切です。たとえば、次のようなディレクトリ構造を使用できます。

```
c:\my_application_domains\config\examples
```

WebLogic 6.0 のサンプル サーバを WebLogic Server 7.0 にアップグレードするには、以下の 2 つのスクリプトを編集する必要があります。

```
setExamplesEnv.cmd
startExamplesServer.cmd
```

これらのスクリプトは DOS 版 (`.cmd`) と UNIX 版 (`.sh`) の両方が用意されています。

`SAMPLES_HOME\server\config\examples` にある `setExamplesEnv` スクリプトでは、開発用のシェル (サンプルをビルドおよび実行するコマンド ウィンドウ) で、一部の環境変数を設定します。

`setExamplesEnv` では以下の変数を設定します。

- `CLASSPATH`

  サンプルのビルドと実行に必要なすべてのクラスが含まれます。

- `CLIENT_CLASSES`

  クライアント クラスを格納するディレクトリを指します。

- `SERVER_CLASSES`

サーバサイド クラスを格納するディレクトリを指します。

- EX_WEBAPP_CLASSES

  サンプル Web アプリケーションで使用されるクラスを格納するディレクトリを指します。

- PATH

  システム パスが含まれます。JDK と WebLogic Server の bin ディレクトリを追加します。

java および javac コマンドは、CLASSPATH 変数を使用して、ソース ファイルのコンパイルとサンプルの実行に必要な Java クラスを見つけます。CLASSPATH には、サンプルをコンパイルおよび実行するための適切なクラスが含まれなければなりません。

サンプル サーバを WebLogic Server 6.0 から WebLogic Server 7.0 へアップグレードするには、WebLogic Server 6.0 クラス、WebLogic Server 7.0 クラス、および WebLogic Server 7.0 で使用するネイティブ ライブラリにアクセスできるように、WebLogic Server 7.0 の setExamplesEnv.cmd スクリプトを編集する必要があります。

WebLogic Server 7.0 を起動するための手順は、WebLogic Server 6.0 を起動するための手順と同じです。

この節には以下のものが含まれています。

- WebLogic 6.0 サービス パック 2 サンプル サーバの起動に使用する setExamplesEnv.cmd スクリプトのサンプル。
- WebLogic 7.0 サンプル サーバを起動するために変更された上記の setExamplesEnv.cmd スクリプトのサンプル。サンプル サーバを WebLogic Server 7.0 にアップグレードするために、このスクリプトを変更する方法の説明が含まれています。
- WebLogic 6.0 サービス パック 2 サンプル サーバの起動に使用する startExamplesServer.cmd スクリプトのサンプル。
- WebLogic 7.0 サンプル サーバを起動するために変更された上記の startExamplesServer.cmd スクリプトのサンプル。サンプル サーバを WebLogic Server 7.0 にアップグレードするために、このスクリプトを変更する方法の説明が含まれています。

WebLogic 6.0 サービス パック 2 サンプル サーバの起動に使用する setExamplesEnv.cmd スクリプト

```
@echo on
@rem This script should be used to set up your environment for
@rem compiling and running the examples included with WebLogic
@rem Server. It contains the following variables:
@rem
@rem WL_HOME   - This must point to the root directory of your
WebLogic
@rem             installation.
@rem JAVA_HOME - Determines the version of Java used to compile
@rem           and run examples. This variable must point to the
@rem          root directory of a complete JDK installation. See
@rem             the WebLogic platform support page
@rem            (http://e-docs.bea.com/wls/platforms/index.html)
@rem             for an up-to-date list of supported JVMs on
@rem  Windows NT.
@rem
@rem When setting these variables below, please use short file
names(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server (/install/index.html in your local documentation set
or on the
@rem Internet at
@rem http://e-docs.bea.com/wls/docs60/install/index.html).

@rem Set user-defined variables.
set WL_HOME=D:\WLS_6.0\wlserver6.0
set JAVA_HOME=D:\WLS_6.0\jdk130

@if exist %WL_HOME%\lib\weblogic.jar goto checkJava
@echo.
@echo The WebLogic Server wasn't found in directory %WL_HOME%.
@echo Please edit the setExamplesEnv.cmd script so that the
@echo WL_HOME
@echo variable points to the WebLogic Server installation
```

```
@echo directory.
@echo Your environment has not been set.
@goto finish

:checkJava

@if exist %JAVA_HOME%\bin\java.exe goto setEnv
@echo.
@echo The JDK wasn't found in directory %JAVA_HOME%.
@echo Please edit the setEnv.cmd script so that the JAVA_HOME
@echo variable points to the location of your JDK.
@echo Your environment has not been set.
@goto finish

:setEnv
set APPLICATIONS=%WL_HOME%\config\examples\applications
set CLIENT_CLASSES=%WL_HOME%\config\examples\clientclasses
set SERVER_CLASSES=%WL_HOME%\config\examples\serverclasses
set
EX_WEBAPP_CLASSES=%WL_HOME%\config\examples\applications\exampl
esWebApp\WEB-INF\classes

set
CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\lib\weblogic_sp.
jar;%WL_HOME%\lib\weblogic.jar;%WL_HOME%\lib\xmlx.jar;%WL_HOME%
\samples\eval\cloudscape\lib\cloudscape.jar;%CLIENT_CLASSES%;
%SERVER_CLASSES%;%EX_WEBAPP_CLASSES%;D:\WLS 6.0

set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%
@echo.
@echo Your environment has been set.

:finish
```

WebLogic 7.0 サンプル サーバを起動するために変更された上記の setExamplesEnv.cmd スクリプト

```
@echo on
@rem This script should be used to set up your environment for
@rem compiling and running the examples included with WebLogic
@rem Server. It contains the following variables:
@rem
@rem WL_HOME   - This must point to the root directory of your
WebLogic
@rem             installation.
@rem JAVA_HOME - Determines the version of Java used to compile
@rem           and run examples. This variable must point to the
@rem          root directory of a complete JDK installation. See
@rem             the WebLogic platform support page
```

```
@rem           (http://e-docs.bea.com/wls/platforms/index.html)
@rem             for an up-to-date list of supported JVMs on
Windows NT.
@rem
@rem When setting these variables below, please use short file
@rem names(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server (/install/index.html in your local documentation set
or on the
@rem Internet at
@rem http://e-docs.bea.com/wls/docs60/install/index.html).

@rem Set user-defined variables.
@rem changed: set WL_HOME=C:\bea60sp2\wlserver6.0
```

1. WebLogic Server 6.0 クラスにアクセスできるように WL60_HOME 変数を設定します。

   ```
   set WL60_HOME=C:\bea60sp2\wlserver6.0
   ```

2. WebLogic Server 7.0 クラスにアクセスできるように WL_HOME 変数を設定します。

   ```
   set WL_HOME=C:\bea700\weblogic700
   ```

   ```
   @rem changed: set JAVA_HOME=C:\bea60sp2\jdk130
   ```

3. WebLogic Server 7.0 の JDK を指すようにします。

   ```
   set JAVA_HOME=c:\bea700\jdk131
   ```

4. サンプルをビルドするときに作成されるアプリケーション アーカイブを指すようにします。

   ```
   set APPLICATIONS=%WL60_HOME%\config\examples\applications
   ```

5. クライアント クラスの格納に使用するディレクトリを指すようにします。

   ```
   set CLIENT_CLASSES=%WL60_HOME%\config\examples\clientclasses
   ```

6. サーバサイド クラスの格納に使用するディレクトリを指すようにします。

```
set SERVER_CLASSES=%WL60_HOME%\config\examples\serverclasses
```

7. サンプル Web アプリケーションで使用するクラスを格納するためのディレクトリを指すようにします。

```
set
EX_WEBAPP_CLASSES=%WL60_HOME%\config\examples\applications\exam
plesWebApp\WEB-INF\classes
```

8. WebLogic Server 6.0 および WebLogic Server 7.0 のクラスを指すようにします。

```
set
CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\lib\weblogic_sp.
jar;%WL_HOME%\lib\weblogic.jar;%WL_HOME%\lib\xmlx.jar;%WL60_
HOME%\samples\eval\cloudscape\lib\cloudscape.jar;%CLIENT_
CLASSES%;%SERVER_CLASSES%;%EX_WEBAPP_CLASSES%
```

9. %WL_HOME% 7.0 のホームを指すようにします。

```
set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%


@echo.
@echo Your environment has been set.
```

WebLogic 6.0 サービス パック 2 サンプル サーバの起動に使用する startExamplesServer.cmd スクリプト

```
@echo off

@rem This script can be used to start WebLogic Server for the
@rem purpose
@rem of running the examples. This script ensures that the server
is started
@rem using the config.xml file found in this directory and that
the CLASSPATH
@rem is set appropriately. This script contains the following
@rem variable:
@rem
@rem JAVA_HOME     - Determines the version of Java used to start
@rem                   WebLogic Server. This variable must point
to the
@rem                    root directory of a JDK installation and
will be set
@rem                   for you by the WebLogic Server installer.
```

```
Note that
@rem                  this script uses the hotspot VM to run
WebLogic Server.
@rem               If you choose to use a JDK other than the one
@rem                included in the disribution, make sure that
the JDK
@rem                  includes the hotspot VM. See the WebLogic
platform
@rem                  support page
@rem (http://e-docs.bea.com/wls/platforms/index.html)
@rem                  for an up-to-date list of supported JVMs
@rem on Windows NT.
@rem
@rem When setting the variable below, please use short file names
(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
@rem libraries
@rem required for jDriver for Oracle have been installed in the
@rem proper
@rem location and that your system PATH variable has been set
@rem appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server
@rem (http://e-docs.bea.com/wls/docs60/install/index.html).

SETLOCAL

cd ..\..

@rem Set user-defined variables.
set JAVA_HOME=D:\WLS 6.0\jdk130

if exist %JAVA_HOME%\lib\nul goto runWebLogic
echo.
echo The JRE wasn't found in directory %JAVA_HOME%.
echo Please edit the startExamplesServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your Java
installation.
goto finish

:runWebLogic
echo on
set PATH=.\bin;%PATH%
```

```
set
CLASSPATH=.;.\lib\weblogic_sp.jar;.\lib\weblogic.jar;.\samples
\eval\cloudscape\lib\cloudscape.jar;.\config\examples\server
classes

%JAVA_HOME%\bin\java -hotspot -ms64m -mx64m -classpath
%CLASSPATH% -Dweblogic.Domain=examples -Dweblogic.
Name=examplesServer -Dbea.home=D:\WLS 6.0
-Dcloudscape.system.home=./samples/eval/cloudscape/
data -Djava.security.policy==D:\WLS
6.0\wlserver6.0/lib/weblogic.policy weblogic.Server

goto finish

:finish
cd config\examples
ENDLOCAL
```

WebLogic 7.0 サンプル サーバを起動するために変更された上記の startExamplesServer.cmd スクリプト

```
@echo off

SETLOCAL


@rem Set user-defined variables.
@rem original:set JAVA_HOME=C:\bea60sp2\jdk130
```

1. JAVA_HOME を WebLogic Server 7.0 の新しい JDK に設定します。

   ```
   set JAVA_HOME=C:\bea700\jdk131

   @rem added:
   ```

2. WebLogic Server 6.0 クラスにアクセスできるようにするため、WL60_HOME を設定します。

   ```
   set WL60_HOME=c:\bea60sp2\wlserver6.0
   ```

3. WebLogic Server 7.0 クラスにアクセスできるようにするため、以下のように設定します。

   ```
   set WL_HOME=c:\bea700\weblogic700
   ```

```
:checkJRE
if exist %JAVA_HOME%\lib\nul goto runWebLogic
echo.
echo The JRE wasn't found in directory %JAVA_HOME%.
echo Please edit the startExamplesServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your Java
installation.
goto finish

:runWebLogic
echo on
@rem original: set PATH=.\bin;%PATH%
```

4. 使用している %WL_HOME% 7.0 ホームに合わせて PATH を設定します。これを設定していないと、サーバは起動しません。

```
set PATH=%WL_HOME%\bin;%PATH%

@rem original: set
@rem CLASSPATH=.;.\lib\weblogic_sp.jar;
@rem .\lib\weblogic.jar;.\samples\eval\cloudscape\
@rem lib\cloudscape.
@rem jar;.\config\examples\serverclasses
```

5. 必要な古いクラスと新しいクラスを指すように CLASSPATH を設定します。

```
set
CLASSPATH=%WL_HOME%\lib\weblogic_sp.jar;%WL_HOME%\lib\weblogic.
jar;
%WL60_HOME%\samples\eval\cloudscape\lib\
cloudscape.jar;%WL60_HOME%\config\examples\serverclasses

echo CLASSPATH=%CLASSPATH%

@rem original: %JAVA_HOME%\bin\java -hotspot -ms64m -mx64m
@rem -classpath %CLASSPATH% -Dweblogic.Domain=examples
@rem -Dweblogic.Name=examplesServer -Dbea.home=C:\bea60sp2
@rem -Dcloudscape.system.home
@rem =./samples/eval/cloudscape/data
@rem -Djava.security.policy==C:\bea60sp2\wlserver6.0
@rem /lib/weblogic.policy weblogic.Server

%JAVA_HOME%\bin\java -hotspot -ms64m -mx64m -classpath
%CLASSPATH% -Dweblogic.Name=examplesServer
-Dweblogic.ProductionModeEnabled=true -Dbea.home=C:\bea700
-Dcloudscape.system.home=%WL60_HOME%/
samples/eval/cloudscape/data
-Djava.security.policy==%WL60_HOME%/lib/weblogic.policy
weblogic.Server

goto finish
```

```
:finish
ENDLOCAL
```

WebLogic Server 7.0 でサンプル サーバを起動する

WebLogic Server 7.0 でサンプル サーバを起動するには、次の手順に従います。

Windows の場合

1. タスクバーで、[**スタート**] をクリックします。
2. [**プログラム**] を選択します。
3. [**BEA WebLogic E-Business Platform**] を選択します。
4. [**WebLogic Server 7.0**] を選択します。
5. [**Examples**] を選択します。
6. [**Start Examples Server**] を選択します。
7. [**Out-of-the-Box Examples Index Page**] が表示されます。

または

1. Windows エクスプローラで、SAMPLES_HOME\server\config\examples ディレクトリに移動します。
2. startExamplesServer のアイコンをダブルクリックします。
3. [**Out-of-the-Box Examples Index Page**] が表示されます。

UNIX Bourne シェルの場合

1. cd `$SAMPLES_HOME/server/config/examples`
2. sh `startExamplesServer.sh`

WebLogic 6.1 サービス パック 2 サンプル サーバの WebLogic Server 7.0 へのアップグレード

WebLogic 6.1 サンプル サーバを WebLogic Server 7.0 にアップグレードする必要はありません。この節では、サーバを 6.1 から 7.0 へアップグレードする方法の例として、その手順を説明します。WebLogic 6.1 サンプル サーバのドメイン コンフィグレーションを WebLogic Server 7.0 で使用するためにアップグレードするには、以下の作業を行います。

- WebLogic Server 7.0 をインストールする
- 6.1 サービス パック 2 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する
- WebLogic Server 7.0 でサンプル サーバを起動する

WebLogic Server 7.0 をインストールする

WebLogic Server 7.0 をインストールします。『インストール ガイド』を参照してください。

注意： 古いバージョンと同じ場所に新しいバージョンをインストールしようとすると、インストーラにより警告が出されます。

6.1 サービス パック 2 のドメイン コンフィグレーションで WebLogic Server 7.0 の環境を設定する

6.1 の `WL_HOME/config/examples` ディレクトリを新しい場所にコピーするときは、引き続き、config ディレクトリ内に examples ドメイン ディレクトリが含まれるようにすることが大切です。たとえば、次のようなディレクトリ構造を使用できます。

```
c:\my_application_domains\config\examples
```

WebLogic 6.1 のサンプル サーバを WebLogic Server 7.0 にアップグレードするには、以下の 2 つのスクリプトを編集する必要があります。

```
setExamplesEnv.cmd
startExamplesServer.cmd
```

これらのスクリプトは DOS 版 (.cmd) と UNIX 版 (.sh) の両方が用意されています。

SAMPLES_HOME\server\config\examples にある setExamplesEnv スクリプトでは、開発用のシェル (サンプルをビルドおよび実行するコマンド ウィンドウ) で、一部の環境変数を設定します。

setExamplesEnv では以下の変数を設定します。

- CLASSPATH

  サンプルのビルドと実行に必要なすべてのクラスが含まれます。

- CLIENT_CLASSES

  クライアント クラスを格納するディレクトリを指します。

- SERVER_CLASSES

  サーバサイド クラスを格納するディレクトリを指します。

- EX_WEBAPP_CLASSES

  サンプル Web アプリケーションで使用されるクラスを格納するディレクトリを指します。

- PATH

  システム パスが含まれます。JDK と WebLogic Server の bin ディレクトリを追加します。

java および javac コマンドは、CLASSPATH 変数を使用して、ソース ファイルのコンパイルとサンプルの実行に必要な Java クラスを見つけます。CLASSPATH には、サンプルをコンパイルおよび実行するための適切なクラスが含まれなければなりません。

サンプル サーバを WebLogic Server 6.1 から WebLogic Server 7.0 へアップグレードするには、WebLogic Server 6.1 クラス、WebLogic Server 7.0 クラス、および WebLogic Server 7.0 で使用するネイティブ ライブラリにアクセスできるように、WebLogic Server 7.0 の `setExamplesEnv.cmd` スクリプトを編集する必要があります。

WebLogic Server 7.0 を起動するための手順は、WebLogic Server 6.1 を起動するための手順と同じです。

この節には以下のものが含まれています。

- WebLogic 6.1 サービス パック 2 サンプル サーバの起動に使用する setExamplesEnv.cmd スクリプトのサンプル。
- WebLogic Server 7.0 を起動するために変更された上記の setExamplesEnv.cmd スクリプトのサンプル。サンプル サーバを WebLogic Server 7.0 にアップグレードするために、このスクリプトを変更する方法の説明が含まれています。
- WebLogic 6.1 サービス パック 2 サンプル サーバの起動に使用する startExamplesServer.cmd スクリプトのサンプル。
- WebLogic Server 7.0 を起動するために変更された上記の startExamplesServer.cmd スクリプトのサンプル。サンプル サーバを WebLogic Server 7.0 にアップグレードするために、このスクリプトを変更する方法の説明が含まれています。

WebLogic 6.1 サービス パック 2 サンプル サーバの起動に使用する setExamplesEnv.cmd スクリプト

```
@echo on
@rem This script should be used to set up your environment for
@rem compiling and running the examples included with WebLogic
@rem Server. It contains the following variables:
@rem
@rem WL_HOME   - This must point to the root directory of your
WebLogic
@rem             installation.
@rem JAVA_HOME - Determines the version of Java used to compile
@rem           and run examples. This variable must point to the
@rem          root directory of a complete JDK installation. See
@rem            the WebLogic platform support page
@rem           (http://e-docs.bea.com/wls/platforms/index.html)
```

```
@rem             for an up-to-date list of supported JVMs on
Windows NT.
@rem When setting these variables below, please use short file
names(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
@rem proper location and that your system PATH variable
@rem has been set appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server (/install/index.html in your local documentation set
or on the Internet at
@rem http://e-docs.bea.com/wls/docs61/install/index.html).

@rem Set user-defined variables.
set WL_HOME=D:\WLS 6.1\wlserver6.1
set JAVA_HOME=D:\WLS 6.1\jdk131

@dir %WL_HOME%\lib > nul
if errorlevel 0 goto checkJava
@echo.
@echo The WebLogic Server wasn't found in directory %WL_HOME%.
@echo Please edit the setExamplesEnv.cmd script so that the
@echo WL_HOME variable points to the WebLogic Server installation
@echo directory.
@echo Your environment has not been set.
@goto finish

:checkJava
@dir %JAVA_HOME%\jre\bin\java.exe > nul
if errorlevel 0 goto setEnv
@echo.
@echo The JDK wasn't found in directory %JAVA_HOME%.
@echo Please edit the setEnv.cmd script so that the JAVA_HOME
@echo variable points to the location of your JDK.
@echo Your environment has not been set.
@goto finish

:setEnv
set APPLICATIONS=%WL_HOME%\config\examples\applications
set CLIENT_CLASSES=%WL_HOME%\config\examples\clientclasses
set SERVER_CLASSES=%WL_HOME%\config\examples\serverclasses
set EX_WEBAPP_CLASSES=%WL_HOME%\config\examples\applications\
examplesWebApp\WEB-INF\classes

set CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\lib\
weblogic_sp.jar;%WL_HOME%\lib\weblogic.jar;%WL_HOME%\lib\
```

```
xmlx.jar;%WL_HOME%\samples\eval\cloudscape\lib\cloudscape.jar;
%CLIENT_CLASSES%;%SERVER_CLASSES%;%EX_WEBAPP_CLASSES%;
D:\WLS 6.1

set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%
@echo.
@echo Your environment has been set.

:finish
```

WebLogic Server 7.0 を起動するために変更された上記の setExamplesEnv.cmd スクリプト

```
@echo on
@rem This script should be used to set up your environment for
@rem compiling and running the examples included with WebLogic
@rem Server. It contains the following variables:
@rem
@rem WL_HOME   - This must point to the root directory of your
WebLogic
@rem             installation.
@rem JAVA_HOME - Determines the version of Java used to compile
@rem           and run examples. This variable must point to the
@rem          root directory of a complete JDK installation. See
@rem             the WebLogic platform support page
@rem           (http://e-docs.bea.com/wls/platforms/index.html)
@rem             for an up-to-date list of supported JVMs on
Windows NT.
@rem
@rem When setting these variables below, please use short file
@rem names(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
@rem names with spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server (/install/index.html in your local documentation set
or on the
@rem Internet at
http://e-docs.bea.com/wls/docs61/install/index.html).
```

```
@rem Set user-defined variables.
@rem changed: set WL_HOME=C:\bea61sp2\wlserver6.1
```

1. WebLogic Server 6.1 クラスにアクセスできるように WL61_HOME 変数を設定します。

   ```
   set WL61_HOME=C:\bea61sp2\wlserver6.1
   ```

2. WebLogic Server 7.0 クラスにアクセスできるように WL_HOME 変数を設定します。

   ```
   set WL_HOME=C:\bea700\weblogic700
   ```

3. WebLogic Server 7.0 の JDK を指すようにします。

   ```
   @rem changed: set JAVA_HOME=C:\bea61sp2\jdk130
   set JAVA_HOME=c:\bea700\jdk131
   ```

4. サンプルをビルドするときに作成されるアプリケーション アーカイブを指すようにします。

   ```
   set APPLICATIONS=%WL61_HOME%\config\examples\applications
   ```

5. クライアント クラスの格納に使用するディレクトリを指すようにします。

   ```
   set CLIENT_CLASSES=%WL61_HOME%\config\examples\clientclasses
   ```

6. サーバサイド クラスの格納に使用するディレクトリを指すようにします。

   ```
   set SERVER_CLASSES=%WL61_HOME%\config\examples\serverclasses
   ```

7. サンプル Web アプリケーションで使用するクラスを格納するためのディレクトリを指すようにします。

   ```
   set
   EX_WEBAPP_CLASSES=%WL61_HOME%\config\examples\applications\exam
   plesWebApp\WEB-INF\classes
   ```

8. WebLogic Server 6.1 および WebLogic Server 7.0 のクラスを指すようにします。

   ```
   set
   CLASSPATH=%JAVA_HOME%\lib\tools.jar;%WL_HOME%\lib\weblogic_sp.
   jar;%WL_HOME%\lib\weblogic.jar;%WL_HOME%\lib\xmlx.jar;%WL61_
   HOME%\samples\eval\cloudscape\lib\cloudscape.jar;%CLIENT_
   CLASSES%;%SERVER_CLASSES%;%EX_WEBAPP_CLASSES%
   ```

9. %WL_HOME% 7.0 のホームを指すようにします。

   ```
   set PATH=%WL_HOME%\bin;%JAVA_HOME%\bin;%PATH%
   ```

```
@echo.
@echo Your environment has been set.
```

WebLogic 6.1 サービス パック 2 サンプル サーバの起動に使用する startExamplesServer.cmd スクリプト

```
@echo off

@rem This script can be used to start WebLogic Server for the
purpose
@rem of running the examples. This script ensures that the server
is started
@rem using the config.xml file found in this directory and that
the CLASSPATH
@rem is set appropriately. This script contains the following
variable:
@rem
@rem JAVA_HOME     - Determines the version of Java used to start
@rem                  WebLogic Server. This variable must point
to the
@rem                  root directory of a JDK installation and
will be set
@rem                  for you by the WebLogic Server installer.
Note that
@rem                  this script uses the hotspot VM to run
WebLogic Server.
@rem                If you choose to use a JDK other than the one
@rem                included in the disribution, make sure that
the JDK
@rem                  includes the hotspot VM. See the WebLogic
platform
@rem                  support page
(http://e-docs.bea.com/wls/platforms/index.html)
@rem                  for an up-to-date list of supported JVMs
on Windows NT.
@rem

@rem When setting the variable below, please use short file names
(8.3).
@rem To display short (MS-DOS) filenames, use "dir /x". File
names with
@rem spaces will break this script.
@rem
@rem jDriver for Oracle users: This script assumes that native
libraries
@rem required for jDriver for Oracle have been installed in the
```

```
proper
@rem location and that your system PATH variable has been set
appropriately.
@rem For additional information, refer to Installing and Setting
up WebLogic
@rem Server
(http://e-docs.bea.com/wls/docs61/install/index.html).

SETLOCAL

cd ..\..

@rem Set user-defined variables.
set JAVA_HOME=D:\WLS 6.1\jdk131

@rem Check that script is being run from the appropriate
directory
if not exist lib\weblogic.jar goto wrongplace
goto checkJDK

:wrongplace

echo startExamplesServer.cmd must be run from the
config\examples directory. 1>&2
goto finish

:checkJDK
if exist %JAVA_HOME%/bin/javac.exe goto runWebLogic
echo.
echo Javac wasn't found in directory %JAVA_HOME%/bin.
echo Please edit the startExamplesServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your JDK
installation.
goto finish

:runWebLogic
echo on
set PATH=.\bin;%PATH%

set
CLASSPATH=.;.\lib\weblogic_sp.jar;.\lib\weblogic.jar;.\samples
eval\cloudscape\lib\cloudscape.jar;.\config\examples\serverclas
ses
echo off

echo.
echo ************************************************
echo *  To start WebLogic Server, use the password     *
echo *  assigned to the system user.  The system       *
echo *  username and password must also be used to     *
echo *  access the WebLogic Server console from a web  *
```

```
echo *  browser.                                              *
echo ************************************************

@rem Set WLS_PW equal to your system password for no password
prompt server startup.
set WLS_PW=

echo on
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath
"%CLASSPATH%" -Dweblogic.Domain=examples
-Dweblogic.Name=examplesServer
-Dweblogic.management.password=%WLS_PW% -Dbea.home="D:\WLS 6.1"
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="D:\WLS
6.1\wlserver6.1/lib/weblogic.policy" weblogic.Server
goto finish

:finish
cd config\examples
ENDLOCAL
```

WebLogic Server 7.0 を起動するために変更された上記の startExamplesServer.cmd スクリプト

```
@echo off

SETLOCAL


@rem Set user-defined variables.
@rem original:set JAVA_HOME=C:\bea61sp2\jdk130
```

1. JAVA_HOME を WebLogic Server 7.0 の新しい JDK に設定します。

   ```
   set JAVA_HOME=C:\bea700\jdk131

   @rem added:
   ```

2. WebLogic Server 6.1 クラスにアクセスできるようにするため、以下のように設定します。

   ```
   set WL61_HOME=c:\bea61sp2\wlserver6.1
   ```

3. WebLogic Server 7.0 クラスにアクセスできるようにするため、以下のように設定します。

   ```
   set WL_HOME=c:\bea700\weblogic700
   ```

```
:checkJRE
if exist %JAVA_HOME%\lib\nul goto runWebLogic
echo.
echo The JRE wasn't found in directory %JAVA_HOME%.
echo Please edit the startExamplesServer.cmd script so that the
JAVA_HOME
echo variable points to the root directory of your Java
installation.
goto finish

:runWebLogic
echo on
@rem original: set PATH=.\bin;%PATH%
```

4. 使用している %WL_HOME% 7.0 ホームに合わせて PATH を設定します。これを設定していないと、サーバは起動しません。

```
set PATH=%WL_HOME%\bin;%PATH%

@rem original: set
@rem CLASSPATH=.;.\lib\weblogic_sp.jar;
@rem .\lib\weblogic.jar;.\samples\eval\cloudscape\
@rem lib\cloudscape.
@rem jar;.\config\examples\serverclasses


set
CLASSPATH=%WL_HOME%\lib\weblogic_sp.jar;%WL_HOME%\lib\weblogic.
jar;
%WL61_HOME%\samples\eval\cloudscape\lib\cloudscape.jar;
%WL61_HOME%\config\examples\serverclasses
```

5. 必要な古いクラスと新しいクラスを指すように CLASSPATH を設定します。

```
echo CLASSPATH=%CLASSPATH%

@rem original: %JAVA_HOME%\bin\java -hotspot -ms64m -mx64m
@rem -classpath %CLASSPATH% -Dweblogic.Domain=examples
@rem -Dweblogic.Name=examplesServer -Dbea.home=C:\bea61sp2
@rem -Dcloudscape.system.home
@rem =./samples/eval/cloudscape/data
@rem -Djava.security.policy==C:\bea61sp2\wlserver6.1
@rem /lib/weblogic.policy weblogic.Server

%JAVA_HOME%\bin\java -hotspot -ms64m -mx64m -classpath
%CLASSPATH% -Dweblogic.Name=examplesServer
-Dweblogic.ProductionModeEnabled=true -Dbea.home=C:\bea700
-Dcloudscape.system.home=%WL61_HOME%/
samples/eval/cloudscape/data
```

```
-Djava.security.policy==%WL61_HOME%/lib/weblogic.policy
weblogic.Server

goto finish


:finish
ENDLOCAL
```

WebLogic Server 7.0 でサンプル サーバを起動する

WebLogic Server 7.0 でサンプル サーバを起動するには、次の手順に従います。

Windows の場合

1. タスクバーで、[**スタート**] をクリックします。
2. [**プログラム**] を選択します。
3. [**BEA WebLogic E-Business Platform**] を選択します。
4. [**WebLogic Server 7.0**] を選択します。
5. [**Examples**] を選択します。
6. [**Start Examples Server**] を選択します。
7. [**Out-of-the-Box Examples Index Page**] が表示されます。

または

1. Windows エクスプローラで、SAMPLES_HOME\server\config\examples ディレクトリに移動します。
2. startExamplesServer のアイコンをダブルクリックします。
3. [**Out-of-the-Box Examples Index Page**] が表示されます。

UNIX Bourne シェルの場合

1. cd `$SAMPLES_HOME/server/config/examples`
2. sh `startExamplesServer.sh`