Oracle® Application Server 10*g*

Oracle Application Server からの移行

10*g*（9.0.4）

部品番号 :B13561-01

2004 年 2 月

ORACLE®

Oracle Application Server 10*g* Oracle Application Server からの移行 , 10*g*（9.0.4）

部品番号 :B13561-01

原本名 : Oracle Application Server 10*g* Migrating from Oracle Application Server, 10*g* (9.0.4)

原本部品番号 : B10424-01

原本著者 : Priya Darshane

原本協力者 : Kai Li, Song Lin, Stephen Mayer, Baogang Song

目次

2 JWeb アプリケーションの OC4J への移行

3 Oracle Application Server カートリッジの移行

はじめに

このマニュアルでは、ご使用のシステムを Oracle Application Server から Oracle Application Server 10*g* へ移行するプロセスについて説明します。

「はじめに」の項目は次のとおりです。

対象読者

『Oracle Application Server 10*g* Oracle Application Server からの移行』は、システムを Oracle Application Server から Oracle Application Server 10*g* に移行するシステム管理者およびアプリケーション開発者を対象としています。

このマニュアルの読者は、Oracle Application Server の設定、操作と開発、およびその他のシステム管理タスクに習熟していることを前提としています。

このマニュアルの構成

このマニュアルは、次の章から構成されています。

第 1 章「Oracle Application Server 10g の概要」

この章では、Oracle Application Server 10*g* の概要について説明し、Oracle Application Server ユーザーの移行オプションを示します。

第 2 章「JWeb アプリケーションの OC4J への移行」

この章では、Oracle Application Server の JWeb アプリケーションを、Oracle Application Server 10*g* の OC4J に移行する方法について説明します。

第 3 章「Oracle Application Server カートリッジの 移行」

この章では、Oracle Application Server カートリッジの機能と、Oracle Application Server 10*g* での対応する機能とを比較し、カートリッジを Oracle Application Server 10*g* Infrastructure に移行する際の考慮点について説明します。

第 4 章「EJB、ECO/Java および JCORBA アプリケーションの移行」

この章では、Oracle Application Server の EJB、ECO for Java および JCO のアプリケーションを Oracle Application Server 10*g* OC4J に移行する方法について説明します。

関連ドキュメント

リリース・ノート、インストール関連ドキュメント、ホワイト・ペーパーまたはその他の関連ドキュメントは、OTN-J（Oracle Technology Network Japan）から、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。登録は、次の Web サイトから無償で行えます。

`http://otn.oracle.co.jp/membership/`

すでに OTN-J のユーザー名およびパスワードを取得している場合は、次の URL で OTN-J Web サイトのドキュメントのセクションに直接接続できます。

`http://otn.oracle.co.jp/document/`

表記規則

この項では、このマニュアルの本文およびコード例で使用される表記規則について説明します。この項の内容は次のとおりです。

本文の表記規則

本文では、特定の項目が一目でわかるように、次の表記規則を使用します。次の表に、その規則と使用例を示します。

規則	意味	例
太字	太字は、本文中で定義されている用語および用語集に記載されている用語を示します。	この句を指定すると、**索引構成表**が作成されます。
固定幅フォントの大文字	固定幅フォントの大文字は、システム指定の要素を示します。このような要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージおよびメソッドがあります。また、システム指定の列名、データベース・オブジェクト、データベース構造、ユーザー名およびロールも含まれます。	`NUMBER` 列に対してのみ、この句を指定できます。 `BACKUP` コマンドを使用して、データベースのバックアップを作成できます。 `USER_TABLES` データ・ディクショナリ・ビュー内の `TABLE_NAME` 列を問い合せます。 `DBMS_STATS.GENERATE_STATS` プロシージャを使用します。
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびユーザーが指定する要素のサンプルを示します。このような要素には、コンピュータ名およびデータベース名、ネット・サービス名および接続識別子があります。また、ユーザーが指定するデータベース・オブジェクトとデータベース構造、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニットおよびパラメータ値も含まれます。 **注意 :** プログラム要素には、大文字と小文字を組み合わせて使用するものもあります。これらの要素は、記載されているとおりに入力してください。	`sqlplus` と入力して、SQL*Plus をオープンします。 パスワードは、`orapwd` ファイルで指定します。 `/disk1/oracle/dbs` ディレクトリ内のデータ・ファイルおよび制御ファイルのバックアップを作成します。 `hr.departments` 表には、`department_id`、`department_name` および `location_id` 列があります。 `QUERY_REWRITE_ENABLED` 初期化パラメータを `true` に設定します。 `oe` ユーザーとして接続します。 `JRepUtil` クラスが次のメソッドを実装します。

規則	意味	例
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。*old_release* とはアップグレード前にインストールしたリリースを示します。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus または他のコマンドライン文の例です。次のように固定幅フォントで表示され、通常のテキストと区別されます。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表に、コード例で使用される表記規則とその使用例を示します。

規則	意味	例
[ ]	大カッコは、カッコ内の項目を任意に選択することを表します。大カッコは、入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコは、カッコ内の項目のうち、1 つが必須であることを表します。中カッコは、入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択項目の区切りに使用します。項目のうちの 1 つを入力します。縦線は、入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のいずれかを示します。	
	■ 例に直接関連しないコードの一部が省略されている。	CREATE TABLE ... AS *subquery*;
	■ コードの一部を繰り返すことができる。	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関連しない複数の行が省略されていることを示します。	
その他の記号	大カッコ、中カッコ、縦線および省略記号以外の記号は、記載されているとおりに入力する必要があります。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック体	イタリック体は、特定の値を指定する必要があるプレースホルダや変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

規則	意味	例

大文字	大文字は、システム指定の要素を示します。これらの要素は、ユーザー定義の要素と区別するために大文字で示されます。大カッコ内にないかぎり、表示されているとおりの順序および綴りで入力します。ただし、大 / 小文字が区別されないため、小文字でも入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名などです。
	注意 : プログラム要素には、大文字と小文字を組み合わせて使用するものもあります。これらの要素は、記載されているとおりに入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Microsoft Windows オペレーティング・システム環境での表記規則

次の表に、Microsoft Windows オペレーティング・システム環境での表記規則とその使用例を示します。

規則	意味	例
ファイル名およびディレクトリ名	ファイル名およびディレクトリ名は大 / 小文字が区別されません。特殊文字の左山カッコ（<)、右山カッコ（>)、コロン（:)、二重引用符（")、スラッシュ（/)、縦線（|）およびハイフン（-）は使用できません。円記号（¥）は、引用符で囲まれている場合でも、要素のセパレータとして処理されます。Windows では、ファイル名が ¥¥ で始まる場合、汎用命名規則が使用されていると解釈されます。	`c:¥winnt"¥"system32` は `C:¥WINNT¥SYSTEM32` と同じです。
`Windows コマンド・プロンプト`	Windows コマンド・プロンプトには、カレント・ディレクトリが表示されます。コマンド・プロンプトのエスケープ文字はカレット（^）です。このマニュアルでは、コマンド・プロンプトと呼びます。	`C:¥oracle¥oradata`

規則	意味	例

特殊文字	Windows コマンド・プロンプトで二重引用符（"）のエスケープ文字として円記号（¥）が必要な場合があります。丸カッコおよび一重引用符（'）にはエスケープ文字は必要ありません。エスケープ文字および特殊文字の詳細は、Windows オペレーティング・システムのドキュメントを参照してください。	`C:¥>exp scott/tiger TABLES=emp QUERY=¥"WHERE job='SALESMAN' and sal<1600¥"`
		`C:¥>imp SYSTEM/`*`password`*
		`FROMUSER=scott TABLES=(emp, dept)`

HOME_NAME	Oracle ホームの名前を表します。ホーム名には、英数字で 16 文字まで使用できます。ホーム名に使用可能な特殊文字は、アンダースコアのみです。	`C:¥> net start Oracle`*`HOME_NAME`*`TNSListener`
ORACLE_HOME および *ORACLE_BASE*	Oracle8*i* より前のリリースでは、Oracle コンポーネントをインストールすると、すべてのサブディレクトリが最上位の *ORACLE_HOME* の直下に置かれました。*ORACLE_HOME* ディレクトリの名前は、デフォルトでは次のいずれかです。 ■ `C:¥orant`（Windows NT の場合） ■ `C:¥orawin98`（Windows 98 の場合） このリリースは、Optimal Flexible Architecture（OFA）のガイドラインに準拠しています。*ORACLE_HOME* ディレクトリ下に配置されないサブディレクトリもあります。最上位のディレクトリは *ORACLE_BASE* と呼ばれ、デフォルトでは `C:¥oracle` です。他の Oracle ソフトウェアがインストールされていないコンピュータに Oracle9*i* リリース 2（9.2）をインストールした場合、Oracle ホーム・ディレクトリは、デフォルトで `C:¥oracle¥ora90` に設定されます。Oracle ホーム・ディレクトリは、*ORACLE_BASE* の直下に配置されます。 このマニュアルに示すディレクトリ・パスの例は、すべて OFA の表記規則に準拠しています。	*ORACLE_BASE*¥*ORACLE_HOME*¥rdbms¥admin ディレクトリへ移動します。

1

Oracle Application Server 10*g* の概要

この章では、Oracle Application Server 10*g* の特長を、Oracle Application Server と比較して説明します。また、Oracle Application Server のコンポーネントと、Oracle Application Server 10*g* における同等の機能のマッピングも示します。

項目は次のとおりです。

Oracle Application Server 10*g* について

Oracle Application Server 10*g* は、Java 2 Platform Enterprise Edition（J2EE）、XML および新興の Web サービス標準を完全にサポートします。Oracle Application Server 10*g* を使用すると、ブラウザやワイヤレス・デバイスからのカスタマイズあるいはアクセスが可能な企業ポータルを配信することにより、顧客および取引パートナへの情報アクセスを簡略化できます。ビジネス・プロセスを再定義したり、使用するアプリケーションおよびデータソースを顧客またはパートナのものと統合できます。リアルタイムのパーソナライズによって各顧客に合せた体験を配信したり、Oracle Application Server 10*g* 統合ビジネス・インテリジェンス・サービスを使用して Web サイトの通信量パターンの評価や相関付けができます。

また、ご使用の分散システムおよび多様なユーザー・コミュニティのすべてを管理および監視するために、集中管理、セキュリティおよびディレクトリ・フレームワークを実装できます。Oracle Application Server 10*g* により、組込み Web キャッシュ、ロード・バランシングおよびクラスタリングを利用した高速でスケーラブルなインターネット・アプリケーションの配置を、Web サイトのインフラストラクチャ上で効率的に行うことができます。

関連項目：『Oracle Application Server 10*g* 概要』を参照してください。

Oracle Application Server のコンポーネントの移行オプション

表 1-1 には、Oracle Application Server コンポーネントとそれらに対応する Oracle Application Server 10*g* の機能、および各コンポーネントについて詳細に説明しているこのマニュアル内の章を示します。移行プロセス中に、これらの Oracle Application Server コンポーネントを、Oracle Application Server 10*g* の最も対応するコンポーネントに移行する必要があります。

表 1-1 Application Server のコンポーネントの比較

Oracle Application Server コンポーネント	Oracle Application Server 10*g* の最も対応する等価コンポーネント	参照先
JWeb アプリケーション	Oracle Application Server Containers for J2EE（OC4J）アプリケーション	第 2 章
JServlet アプリケーション	OC4J アプリケーション	第 2 章
LiveHTML アプリケーション	Apache SSI アプリケーションおよび JavaServer Page（JSP）アプリケーション	第 3 章
Perl アプリケーション	`mod_perl` アプリケーション	第 3 章
CWeb アプリケーション	カスタム Apache モジュール、Common Gateway Interface（CGI）、FastCGI、Java ネイティブ・インタフェース（JNI）	第 3 章
PL/SQL アプリケーション	`mod_plsql` アプリケーション	第 3 章
ECO/Java アプリケーション	OC4J アプリケーション	第 4 章

表 1-1 Application Server のコンポーネントの比較 （続き）

Oracle Application Server コンポーネント	Oracle Application Server 10*g* の最も対応する等価コンポーネント	参照先
EJB アプリケーション	OC4J アプリケーション	第 4 章
JCORBA アプリケーション	OC4J アプリケーション	第 4 章

企業サービスの移行

この項では、企業サービスについて、管理者および開発者に関係する Web サイトの特長を説明します。項目は次のとおりです。

- 概要
- スケーラビリティ
- 可用性とフォルト・トレランス
- ロード・バランシング
- 管理
- セキュリティ
- サード・パーティの Web サーバーのサポート

またこの項では、ご使用の Web サイトを Oracle Application Server から Oracle Application Server 10*g* に移行することが、これらの特長にどのように影響を及ぼすかについても説明します。

概要

Oracle Application Server は、HTTP リスナー層、サーバー層およびアプリケーション層という 3 層で構成されます。HTTP リスナー層は、リスナー、アダプタ・インタフェースおよびディスパッチャで構成されます。サーバー層は、これらのアプリケーションを管理するための共通のコンポーネント・セットを提供します。これらのコンポーネントには、ロード・バランシング、ロギング、自動障害リカバリ、セキュリティ、ディレクトリおよびトランザクション管理コンポーネントが含まれます。アプリケーション層はアプリケーション、カートリッジおよびカートリッジ・サーバーで構成されています。リクエストを受信すると、ディスパッチャはそのリクエストをアプリケーション・サーバー層にルーティングします。使用可能なカートリッジ・インスタンスが存在する場合は、そのインスタンスがリクエストを処理します。存在しない場合は、新しいインスタンスが作成されます。

Oracle Application Server 10*g* では、Oracle HTTP Server によって、ロード・バランシング、`mod_oc4j` によるサーブレット・リクエストの OC4J へのルーティング、`mod_osso` および SSL によるシングル・サインオン認証およびセキュリティ・コンテキストの伝播が処理されます。OC4J は、JSP、サーブレットおよび EJB を実行する Pure J2EE コンテナで構成され、

J2EE コンテナ・サービスを提供します。Oracle HTTP Server および OC4J は、どちらも Oracle Application Server での 3 層と同じ働きをします。

スケーラビリティ

Oracle Application Server は、シングル・ホストの環境やマルチ・ホストの環境にも配置できます。Oracle HTTP Server および OC4J は、シングル・ホスト環境またはクラスタリングされたホスト環境に合せて設定できます。

Oracle HTTP Server

Oracle Application Server では、各リスナーは、同時接続の最大値まで処理できます。この数値は、オペレーティング・システムの制限によって異なります。1 つのサイト上でリクエスト・ロードを分散するには、複数の Web リスナーを作成して、それぞれが別の TCP ポートをリスニングするようにします。

UNIX 上の Oracle Application Server 10*g* の場合、Oracle HTTP Server は起動中にあらかじめ子プロセスのプールを作成し、受信クライアント・リクエストの処理に対応します。リクエスト・ロードが増加するに従って、後続のリクエストのためにサーバーが新しいプロセスを起動します。プールの初期サイズと最大サイズ、および予備のサーバー・プロセスの最小数と最大数は、それぞれ `StartServers`、`MaxClients`、`MinSpareServers` および `MaxSpareServers` ディレクティブで設定されます。

Windows 上の Oracle Application Server 10*g* の場合、Oracle HTTP Server はマルチスレッド・プロセスとして稼働します。同時接続数は、`ThreadsPerChild` ディレクティブで設定されます。これは、UNIX の `StartServers` および `MaxClients` ディレクティブに対応するものです。

Oracle Application Server は、Node Manager を使用して設定できます。Oracle Application Server 10*g* の場合は、Oracle Enterprise Manager 10*g* Application Server Control を使用するか、手動で `httpd.conf` ファイルを編集して、Oracle HTTP Server を設定できます。このファイルは、次の場所にあります。

- UNIX: *`ORACLE_HOME`*`/Apache/Apache/conf/httpd.conf`
- Windows: *`ORACLE_HOME`*`¥Apache¥Apache¥conf¥httpd.conf`

関連項目：

- 『Oracle HTTP Server 管理者ガイド』を参照してください。
- 『Oracle Application Server 10*g* 管理者ガイド』を参照してください。

OC4J

Oracle Application Server では、リクエスト数が増加すると、システム内に新しいカートリッジ・サーバーとインスタンスが作成されます。

Oracle Application Server 10*g* では、Oracle HTTP Server の `mod_oc4j` がサーバーからのリクエストを受信して、OC4J サーブレット・コンテナにルーティングします。

関連項目：

- OC4J の詳細は、『Oracle Application Server Containers for J2EE ユーザーズ・ガイド』を参照してください。
- 第 2 章「JWeb アプリケーションの OC4J への移行」
- 第 4 章「EJB、ECO/Java および JCORBA アプリケーションの移行」

可用性とフォルト・トレランス

リスナーまたはカートリッジ・サーバーなどのコンポーネントで障害が発生した場合、Oracle Application Server は障害を検出してそのコンポーネントを再起動します。可能であれば保存されていた状態情報すべてをリストアします。

Oracle HTTP Server では、HTTP サーバー・ホストまたは OC4J ホストが複数ある場合、1 台のホストが停止しても、1 台の HTTP サーバーおよび 1 台の OC4J が稼働している限りシステムは機能します。ただし、J2EE コンポーネントが OC4J インスタンスのクラスタに配置されている場合に限ります。すべての Oracle HTTP Server インスタンスは、任意の OC4J インスタンスにリクエストをルーティングできます。ルーティング情報を Cookie で保持しておくと、単一の障害箇所が全体に影響を及ぼすことを排除できます。

ロード・バランシング

Oracle Application Server では、優先順位ベースおよび最小 / 最大ベースの 2 種類のロード・バランシング方式に基づいて、システム・リソースの割当て、およびリクエストの優先順位決定が行われます。

優先順位モードでは、アプリケーションおよびカートリッジに設定した優先順位レベルに基づいて、システムによってリソースが自動的に割り当てられます。プロセス数、スレッド数およびインスタンス数は、そのアプリケーションおよびコンポーネントのリクエスト・ロードと優先順位レベルに基づいて自動的に決定されます。

最小 / 最大モードでは、それぞれのカートリッジのインスタンス数、スレッド数およびクライアントのパラメータ数をカートリッジ・レベルで設定します。

Oracle HTTP Server では、ホストの数およびホストの論理セットを設定ファイルで定義します。システムは、受信リクエストを OC4J インスタンスに割り当てます。

管理

Oracle Application Server では、サイト、リスナーおよびアプリケーションの管理と監視用に、GUI ツールや組込みサポートが用意されています。Oracle Application Server Manager ツールの設定データは、様々な設定ファイルに格納されています。

Oracle HTTP Server では、Oracle Enterprise Manager を使用することによって、あるいは設定ファイルを介してサイトの管理およびメンテナンスを実行できます。表 1-2 に、Oracle Application Server HTTP リスナーおよび Oracle HTTP Server の設定ファイルを示します。

関連項目：

- 『Oracle HTTP Server 管理者ガイド』を参照してください。
- 『Oracle Application Server 10*g* 管理者ガイド』を参照してください。

表 1-2 設定ファイルの比較

Oracle Application Server HTTP リスナー	Oracle Application Server 10*g* Oracle HTTP Server
`owl.cfg`: 登録済のリスナーとそれぞれの設定のリスト	`httpd.conf`: 主要な（または唯一の）サーバー全体の設定ファイル （ファイルの場所と変換情報を `srm.conf`、セキュリティ情報を `access.conf` でそれぞれ管理するか、またはすべてのディレクティブを 1 つのファイルで管理するかを選択可能）
`site.app`: サイトの設定ファイル	（対応する設定ファイルなし）
`sv`*`listenerName`*`.cfg`: リスナーの設定ファイル	（対応する設定ファイルなし）
`wrb.app`: プロセスおよびカートリッジの設定ファイル	（対応する設定ファイルなし）
`resources.ora`: ORB の設定ファイル	（対応する設定ファイルなし）

セキュリティ

証明書を Oracle Application Server から Oracle Application Server 10*g* に移行する必要があります。この項では、必要な手順について説明します。

証明書の移行

セキュリティ対策の 1 つに、SSL によるサイトの保護があります。SSL によってサイトを保護しており、SSL 証明書を Oracle Application Server 10*g* に移行する場合、その証明書を移行する必要があります。

Oracle Application Server 10*g* には、`pconvert` と、`ssl2ossl`（Unix の場合）または `osslconvert`（Windows の場合）という 2 つの移行ツールが含まれています。Oracle Application Server 証明書から Oracle Application Server 10*g* 証明書または Wallet に移行するには、次の手順を実行します。

1. 移行ツール `pconvert` を使用して、Oracle Application Server の秘密鍵を Oracle Application Server 10*g* の秘密鍵に移行します。この移行ツールは、次の場所にあります。

   - UNIX: `ORACLE_HOME/Apache/Apache/bin/pconvert`
   - Windows: `ORACLE_HOME¥Apache¥Apache¥bin¥pconvert.exe`

   `pconvert` を実行する構文は次のとおりです。

   ```
   pconvert -s oas_private_key_file -d ias_private_key_file
   ```

   たとえば、次のようになります。

   ```
   prompt> pconvert -s privkey.der -d iaskey.pem
   ```

2. Oracle Application Server の証明書ファイルおよび手順 1 で取得した `ias_private_key_file` を使用して、移行ツール `ssl2ossl` または `osslconvert` で Oracle Application Server 10*g* Wallet ファイルを生成します。ツールへのフルパスは次のとおりです。

   - UNIX: `ORACLE_HOME/Apache/Apache/bin/ssl2ossl`
   - Windows: `ORACLE_HOME¥Apache¥Apache¥bin¥osslconvert.exe`

   Unix で `ssl2ossl` を実行する構文は次のとおりです。

   ```
   ssl2ossl -cert oas_certificate_file
            -key ias_private_key_file
            -wltpass password_for_wallet
            -certpass password_for_oas_certificate_file
            -chain oas_certificate_chain_file
            -capath oas_certificate_authority_path
            -cafile oas_certificate_authority_file
            -wallet wallet_full_path
   ```

```
-ssowallet yes/no
-validate yes/no
```

Windows で osslconvert を実行する構文は次のとおりです。

```
osslconvert.exe -cert oas_certificate_file
                -key ias_private_key_file
                -wltpass password_for_wallet
                -certpass password_for_oas_certificate_file
                -chain oas_certificate_chain_file
                -capath oas_certificate_authority_path
                -cafile oas_certificate_authority_file
                -wallet wallet_full_path
                -ssowallet yes/no
                -validate yes/no
```

表 1-3 は、移行ツール ssl2ossl または osslconvert のパラメータと、それぞれに関連付けられた要件をまとめたものです。

表 1-3 ssl2ossl または osslconvert ツールのパラメータのサマリー

パラメータ	説明	要件
cert	Oracle Application Server 証明書ファイル	必須。
key	手順 1 で取得した ias_private_key_file	必須。
certpass	証明書のパスワード	オプション。
wltpass	Wallet のパスワード	オプション。
chain	Oracle Application Server の証明書チェーン・ファイル	オプションだが、chain、capath または cafile のうち少なくとも 1 つのパラメータが必要。
capath	Oracle Application Server の認証局のパス	オプションだが、chain、capath または cafile のうち少なくとも 1 つのパラメータが必要。
cafile	Oracle Application Server の認証局ファイル。	オプションだが、chain、capath または cafile のうち少なくとも 1 つのパラメータが必要。
wallet	Wallet ファイルのフルパス	オプションだが、デフォルトのパスは *ORACLE_HOME*/Apache/Apache/conf/ssl.wlt/0。
ssowallet	yes または no	オプションだが、デフォルト値は no。
validate	yes または no。yes の場合、ツールは Wallet を生成しない。no の場合、ツールは Wallet を生成する。	オプションだが、デフォルト値は no。

関連項目： `ssowallet` およびその他のセキュリティ情報の詳細は、『Oracle Application Server 10*g* セキュリティ・ガイド』を参照してください。

サード・パーティの Web サーバーのサポート

Oracle Application Server 10*g* は Oracle HTTP Server を、Oracle Application Server は HTTP Server をそれぞれの Web リスナーとして使用します。しかし、多くの企業では会社の標準 Web サーバーとして、Microsoft Internet Information Services（IIS）または Sun ONE のみが使用されています。

Oracle Application Server および Oracle Application Server 10*g* は、どちらも IIS および Sun ONE のようなサード・パーティの Web サーバーをサポートします。

2

JWeb アプリケーションの OC4J への移行

この章では、Oracle Application Server の JWeb アプリケーションを、Oracle Application Server 10*g* の OC4J に移行する方法について説明します。

項目は次のとおりです。

JWeb と OC4J との違い

この項では、JWeb および OC4J の背景について説明します。また、JWeb アプリケーションと OC4J アプリケーションとの違いについても説明します。

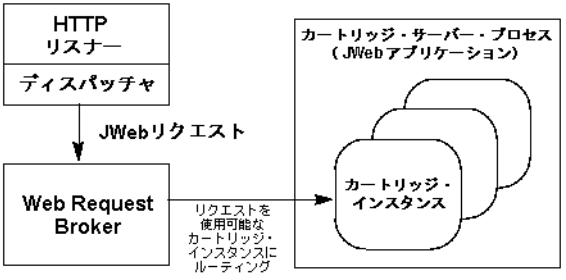
アーキテクチャ

JWeb アプリケーションは、Oracle Application Server のカートリッジ・インフラストラクチャ内で実行されます。一方、OC4J は標準の仮想マシン上で稼働します。

JWeb のアーキテクチャ

Oracle Application Server では、JWeb カートリッジ宛てのリクエストは、HTTP リスナーが受信します。リスナーはリクエストをディスパッチャに渡し、ディスパッチャは認証機能または Web Request Broker（WRB）と通信します。WRB は URL マッピングを使用して、リクエストの送信先カートリッジ・インスタンスを識別します。リクエストされたカートリッジのカートリッジ・インスタンスが存在しない場合、カートリッジ・サーバー・ファクトリがカートリッジ・サーバー・プロセスを作成して、カートリッジのインスタンスを生成します。JWeb では、カートリッジ・サーバー・プロセスは JWeb アプリケーション（Oracle Application Server アプリケーションのパラダイム）を実行する JVM をロードします。図 2-1 に、このプロセスを示します。

図 2-1 Oracle Application Server のカートリッジ・インフラストラクチャ



OC4J アーキテクチャ

OC4J は、サーブレットおよび JSP エンジン、EJB コンテナ、J2EE サービス API（JNDI、JTA、JMS および JAAS）、企業情報システム API（JDBC、SQLJ、J2EE Connector Architecture）を含む Web コンテナで構成されます。

mod_oc4j は、Oracle HTTP Server を通して OC4J プロセスにリクエストをルーティングする目的で、Oracle HTTP Server に動的にロードされるモジュールです。mod_oc4j は、OC4J のセッション情報を考慮して、元の OC4J プロセスにリクエストをルーティングして返します。元の OC4J プロセスに到達できない場合は、同じ OC4J アイランドの別のメンバーに失敗したセッション・リクエストをリルートします。

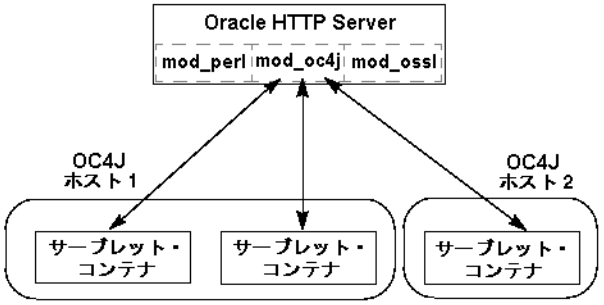
mod_oc4j は、Oracle Process Manager and Notification Server（OPMN）および OC4J の 2 つのコンポーネントと相互作用します。mod_oc4j はリクエストをルーティングすることによって OC4J と相互作用します。OPMN は、mod_oc4j を起動する Oracle HTTP Server およびすべての OC4J プロセスを起動します。OPMN は起動した各プロセスを監視し、各プロセスが到達可能であることを定期的に確認します。プロセスが使用不可能あるいは到達不可能になった場合、OPMN はそのプロセスを再起動します。さらに OPMN は OC4J のプロセス・ステータスを mod_oc4j に伝達するので、mod_oc4j は OC4J プロセスがいつ起動され、いつ停止されたかを把握できます。mod_oc4j はこの情報を使用して、内部 OC4J プロセス表をメンテナンスし、リクエストのルーティングを迅速に行えるようにします。

関連項目：

- 『Oracle Application Server Containers for J2EE ユーザーズ・ガイド』を参照してください。
- 『Oracle HTTP Server 管理者ガイド』を参照してください。

図 2-2 に、1 対多の設定を示します。

図 2-2 OC4J のアーキテクチャ（1 対多の例）



1 対多の構成は、1 つの Oracle HTTP Server リスナーと複数の OC4J インスタンスで成り立ちます。前の図では、1 つの Oracle HTTP Server インスタンスが 2 つの OC4J ホストと通信しています。OC4J ホスト 1 は 2 つのサーブレット・コンテナを実行し、OC4J ホスト 2 は 1 つのサーブレット・コンテナを実行しています。各サーブレット・コンテナと OC4J インスタンス内の 1 つの mod_oc4j との間に、3 つの接続が開かれています。

サーブレット・コンテナは、Servlet 2.3 の Application Programming Interface（API）仕様を実装するサーブレットを実行するためのランタイム環境を提供します。サーブレット・コンテナは、mod_oc4j と同じホストまたは異なるホストの JVM プロセスで実行されます。各 JVM は 1 つずつサーブレット・コンテナを持っており、サーブレット・コンテナの数は Web サーバー（mod_oc4j モジュール）の数に比例しません。1 つの mod_oc4j は複数のサーブレット・コンテナを使用して動作することが可能で、その逆も可能です。また、複数の mod_oc4j モジュールが複数のサーブレット・コンテナと動作することも可能です。

シングル・ホスト構成

サーブレット・コンテナが Web サーバーと同じマシンに存在する場合、サーブレット・コンテナと JVM を Web サーバーに合せて停止または起動するように mod_oc4j モジュールを設定できます。JVM を正常終了するために必要なタスクは、モジュールがすべて実行します。またこの場合、mod_oc4j が定期的に JVM のステータスをチェックし、1 つ目の JVM が使用不可能になったときには別の JVM を起動することにより、フェイルオーバーを実行することも可能です。

ライフ・サイクル

JWeb クラスと OC4J アプリケーションのライフ・サイクルは異なります。

JWeb のライフ・サイクル

JWeb クラスでは、標準の main() エントリ・ポイントを使用して実行ロジックを開始します。JWeb クラスのライフ・サイクルは、標準の Java クラスにおける main() のロード、リンク、初期化および実行と似ています。

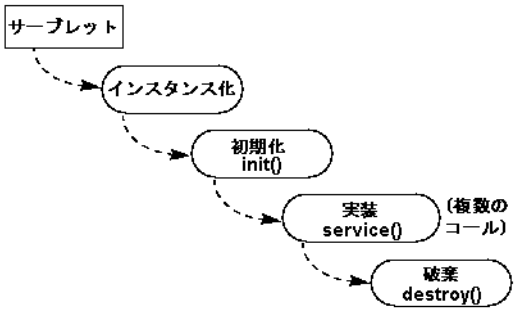
> **関連項目：** Java Virtual Machine（JVM）の詳細は、http://java.sun.com/docs を参照してください。

OC4J のライフ・サイクル

OC4J では、サーブレットのライフ・サイクルは Servlet 2.3 の仕様に準拠しています。ライフ・サイクルは、すべてのサーブレットによって直接または間接的に実装される javax.servlet.servlet インタフェースによって定義されます。このインタフェースには、サーブレットのライフ・サイクル中に、サーブレット・エンジンにより特定の順序で特定のタイミングにコールされるメソッドがあります。init() および destroy() メソッドはサーブレットの存続期間中に 1 回だけ実行されますが、service() メソッドはサーブレットのロジックを実行するために複数回コールされます。

図 2-3 に、サーブレットのライフ・サイクルを示します。

図 2-3 サーブレットのライフ・サイクル



スレッド

JWeb カートリッジと OC4J サーブレット・コンテナでは、シングル・スレッドまたはマルチスレッドの実行をサポートしていますが、スレッドの実装は異なります。

JWeb のスレッド

JWeb カートリッジのスレッドは、Oracle Application Server のカートリッジ設定で `Stateless` パラメータを切り替えることにより定義されます。Stateless パラメータが `true` の場合、カートリッジ・インスタンスは複数のクライアントで共有されます。`false` の場合は共有されず、一度に 1 つのクライアントのみアクセスできます。また、Oracle Application Server が最小 / 最大モードの場合、最小 / 最大カートリッジ・サーバーと最小 / 最大スレッド値を変更して、そのカートリッジのマルチスレッドの実装方法を変更できます。

OC4J スレッド

OC4J サーブレット・コンテナは、デフォルトではマルチスレッドに設定されています。OC4J サーブレット・コンテナが、クライアントのリクエストを処理するスレッドを管理します。サーブレット・クラスの各インスタンスに、複数のスレッドを実行させることが可能です。この場合、サーブレット・インスタンスは複数のクライアントで共有されます。あるいはまた、あるクラスで `javax.servlet.SingleThread` インタフェースを実装することにより、そのクラスが一度に 1 つのスレッドのみ実行するよう指定できます。この場合、このサーブレット・クラスのインスタンスのプールが維持され、各インスタンスは一度に 1 つのクライアントにのみ割り当てられます（インスタンスは共有されません）。

セッション

JWeb カートリッジでは、Oracle Application Server の Node Manager を使用してクライアント・セッションを使用可能にできます。OC4J では、Servlet 2.3 の仕様に従い、プログラムによるセッションのみ使用可能です。このため、コード以外の手段によってセッションが使用可能になっている JWeb アプリケーションを移行する場合は、サーブレット・セッション API を使用して、セッション・メカニズムをプログラムによって実装する必要があります。

関連項目： 2-8 ページの「セッション制御」を参照してください。

HTML ページの動的コンテンツ生成

HTML ページの動的コンテンツを生成するために、JWeb Toolkit 機能を使用できます。JWeb Toolkit は、HTML ページ内に特殊なプレースホルダを埋め込みます。ファイルが `oracle.html.HtmlFile` オブジェクトとして JWeb クラスにインポートされると、`setItemAt()` メソッドにより、プレースホルダの場所にコードから生成されたデータが挿入されます。

これは JWeb 固有の機能であるため、Oracle Application Server 10*g* では使用できません。動的情報を HTML ページに埋め込む（スクリプト）場合は、Oracle Application Server 10*g* で JSP を使用することを検討してください。

関連項目：『Oracle Application Server Containers for J2EE JavaServer Pages 開発者ガイド』を参照してください。

移行方法

OC4J では、J2EE 1.2 完全サポートに加えて、Servlet 2.3 全体、EJB 2.0（Message-Driven Bean）の一部、JAAS および JCA サポート全体など J2EE 1.3 の大部分もサポートします。Oracle Application Server 4.x 上に配置された JWeb アプリケーションまたは JServlet アプリケーションを Oracle Application Server 10*g* に移行するには、OC4J に適する仕様に準拠するように、移行する JWeb アプリケーションまたは JServlet アプリケーションを修正する必要があります。

準拠標準の比較

表 2-1 に、Oracle Application Server の JWeb カートリッジおよび JServlet カートリッジと、Oracle Application Server 10*g* の OC4J との間の準拠標準の比較を示します。JWeb または JServlet を Oracle Application Server から Oracle Application Server 10*g* 内の OC4J サーブレットに移行するときは、Servlet 2.3 の仕様に準拠するようにコードを変更する必要があります。

表 2-1 JWeb、JServlet および OC4J の準拠標準の比較

準拠標準	JWeb	JServlet	OC4J
サーブレットの仕様	なし	2.3	2.3
JSP の仕様	なし	なし	1.2

関連項目： サーブレットの仕様の詳細は、`http://java.sun.com` を参照してください。

JWeb およびサーブレットの主要メソッド

移行を実行するには、次の主要メソッドを理解して使用する必要があります。

JWeb: `main()` メソッドを持つ Java クラスが含まれます。JWeb カートリッジとも呼ばれます。JWeb のインフラストラクチャにより、URL がこのメソッドにマップされます。

サーブレット: `doGet()` メソッドおよび `doPut()` メソッドをいくつか持つ Java クラスが含まれます。これらのメソッドは Sun 社により指定されており、URL にマップされます。

移行のアプローチ

主要な移行のアプローチとして、JWeb カートリッジの `main()` メソッドを、対応するサーブレットの `doGet()` メソッドの中でコールできます。

特に次の点に注意する必要があります。

- **ロギング API**: Oracle Application Server 10*g* では、Oracle Application Server のロギング API はサポートされません。かわりにサーブレット・ロギング API が使用されます。そのため、ロギング API の変更を反映するように、JWeb カートリッジのコードを変更する必要があります。
- **ユーティリティ API**: ユーティリティ API を書き込むときは JSP を使用してください。現在、`oracle.html.*.` パッケージは使用できません。

  関連項目： `html.*.` パッケージの詳細は、`http://jakarta.apache.org/ecs/index.html` を参照してください。

- **WRB コール**: Oracle Application Server 10*g* ではほとんどの WRB API がサポートされないので、セキュリティ・コードを書き込むときは標準のサーブレット API を使用する必要があります。たとえば、`getClientCertificate()` および `getLogin()` などのメソッドがあります。
- **セッション**: 2-8 ページの「セッション制御」を参照してください。

- **アプリケーション・スレッド**: 2-9 ページの「アプリケーション・スレッド」を参照してください。
- **ロギング**: 2-9 ページの「ロギング」を参照してください。

JWeb アプリケーションのコードの変更

JWeb アプリケーションを OC4J に移行するには、次の分野のコードを変更する必要があります。

- セッション制御
- アプリケーション・スレッド
- ロギング

セッション制御

JWeb アプリケーションでセッションを使用可能にするには、Node Manager の「Web パラメータ」フォーム内のカートリッジの「クライアント・セッション」パラメータを使用します。これにより、実行されたクラスの静的パラメータが、複数のコールにまたがってクライアント・データを保有することが可能になります。OC4J では、Servlet 2.3 の API により、セッションの状態はサーブレット・クラスの静的変数に格納されません。かわりにセッション・オブジェクトが明示的に取得され、名前付きの属性を使用してセッションの状態を格納します。

OC4J では、セッションを使用可能にする設定をサポートしていません。そのため、次のように `javax.servlet.http.HttpServletRequest` の `getSession()` メソッドを使用して、コード内でセッションを使用可能にする必要があります。

```
HttpSession session = request.getSession(true);
```

セッションの状態情報は、後で格納および取得できます。たとえば、`javax.servlet.http.HttpSession` の `setAttribute(java.lang.String, java.lang.Object)` メソッドおよび `getAttribute(java.lang.String)` メソッドをそれぞれ使用します。

```
session.setAttribute("List", new Vector());
Vector list = (Vector) session.getAttribute("List");
```

注意： OC4J では、セッションの状態の保持に静的データ・メンバーを使用しないでください（ただし、これは JWeb では一般的な手法です）。かわりにサーブレット・セッション API を使用します。後者のほうが、サーブレット・コンテナはメモリーを効率的に使用します。

セッションのタイムアウト

OC4J コンテナに対するデフォルトのセッション・タイムアウトは、XML デプロイメント・ディスクリプタの session.config 要素内で指定できます。HTTPSession インタフェースで getMaxInactiveInterval メソッドを使用できます。コンテナのタイムアウト値を設定するには、setMaxInactiveInterval メソッドを使用してください。

JWeb のセッション・タイムアウト・コールバックは、OC4J では使用できません。

アプリケーション・スレッド

JWeb では、アプリケーションは oracle.owas.wrb.WRBRunnable クラスを使用してスレッドを管理できます。このクラスにより、アプリケーション・スレッドはリクエスト情報およびレスポンス情報にアクセスできます。OC4J では、アプリケーション・スレッドの管理には、標準の Java スレッド管理のみ必要です（java.lang.Runnable インタフェースが使用されます）。JWeb でも OC4J でも、マルチスレッド・アプリケーションはロード・バランシングの効果を制限するため、アプリケーション・スレッドの使用はお薦めしません。

ロギング

Oracle Application Server では、JWeb アプリケーションは、WRB により提供される Logger サービスを使用して、メッセージのログを出力します。このサービスにより、ファイル・システムまたはデータベースなどの中央リポジトリにメッセージを書き込むことができます。oracle.owas.wrb.services.logger.OutputLogStream クラスは、Logger サービスとの間のインタフェースとして機能します。

Oracle Application Server 10*g* では、OC4J によって、サーブレット・ロギング API に関連付けられた診断メッセージが生成されます。このロギング・ファイルは、*ORACLE_HOME*/j2ee/home/log/*digit digit_island-name*/server.log にあります。

注意： OC4J にログ・レベルはありません。

JWeb Toolkit パッケージ（JWeb API）

Oracle Application Server には、独自の Java パッケージを有する JWeb Toolkit が含まれています。Oracle Application Server に移行する JWeb アプリケーションの中でこれらのパッケージを使用している場合は、コードを変更して、対応する Servlet 2.3 のクラスおよびメソッドを使用する必要があります。対応する機能がない場合は、JWeb パッケージで提供されていた機能を実装するためにコードを再作成する必要があります。

一部の JWeb Toolkit パッケージは、WRB などの Oracle Application Server コンポーネントと対話するように特別に設計されているため、これらのパッケージの機能は標準のサーブレット API では再現できません。このため、移行プロセスには、一部のアプリケーションの再設計も含まれます。

表 2-2 ～表 2-8 に、JWeb のメソッドと、それぞれの機能と同等のサーブレット API クラスのメソッドを示します。

表 2-2 に、javax.servlet.http.HttpServletRequest クラスのメソッドと同等の JWeb のメソッドを示します。

表 2-2 javax.servlet.http.HttpServletRequest クラスのメソッドと同等の JWeb のメソッド

JWeb のメソッド	サーブレットのメソッド
oracle.owas.wrb.services.http.HTTP.getHeader(String)	getHeader(string)
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("AUTH_TYPE")	getAuthType()
oracle.owas.wrb.services.http.HTTP.getHeaders()[1]	getHeaderNames()[2]
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("PATH_INFO")	getPathInfo()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("PATH_TRANSLATED")	getPathTranslated()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("QUERY_STRING")	getQueryString()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("REQUEST_METHOD")	getMethod()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("REMOTE_USER")	getRemoteUser()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("SCRIPT_NAME")	getServletPath()

[1] ヘッダー名と値のハッシュテーブルが返されます。
[2] ヘッダー名の列挙が返されます。

表 2-3 に、javax.servlet.ServletRequest クラスのメソッドと同等の JWeb のメソッドを示します。

表 2-3 javax.servlet.ServletRequest クラスのメソッドと同等の JWeb のメソッド

JWeb のメソッド	サーブレットのメソッド
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("CONTENT_TYPE")	getContentType()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("CONTENT_LENGTH")	getContentLength()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("SERVER_PROTOCOL")	getProtocol()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("REMOTE_ADDR")	getRemoteAddr()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("REMOTE_HOST")	getRemoteHost()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("SERVER_NAME")	getServerName()
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("SERVER_PORT")	getServerPort()
oracle.owas.wrb.services.http.HTTP.getPreferredAcceptCharset()	getCharacterEncoding()
oracle.owas.wrb.services.http.HTTP.getURLParameter(name)	getParameter(string)

表 2-3 javax.servlet.ServletRequest クラスのメソッドと同等の JWeb のメソッド（続き）

JWeb のメソッド	サーブレットのメソッド
oracle.owas.wrb.services.http.HTTP.getURLParameters(name)	getParameterValues(string)

表 2-4 に、javax.servlet.ServletResponse クラスのメソッドと同等の JWeb のメソッドを示します。

表 2-4 javax.servlet.ServletResponse クラスのメソッドと同等の JWeb のメソッド

JWeb のメソッド	サーブレットのメソッド
oracle.owas.wrb.WRBWriter	getWriter()

表 2-5 に、javax.servlet.ServletContext クラスのメソッドと同等の JWeb のメソッドを示します。

表 2-5 javax.servlet.ServletContext クラスのメソッドと同等の JWeb のメソッド

JWeb のメソッド	サーブレットのメソッド
oracle.owas.wrb.services.http.HTTP.getCGIEnvironment("SERVER_SOFTWARE")	getServerInfo()
oracle.OAS.Services.Logger package	log(Exception, String) log(String)

表 2-6 に、javax.servlet.http.HttpUtils クラスのメソッドと同等の JWeb のメソッドを示します。

表 2-6 javax.servlet.http.HttpUtils クラスのメソッドと同等の JWeb のメソッド

JWeb のメソッド	サーブレットのメソッド
oracle.owas.wrb.services.http.HTTP.getURLParameters(Hashtable)	parsePostData(int, ServletInputStream)
oracle.owas.wrb.services.http.HTTP.getURLParameters(Hashtable)	parseQueryString(String)

表 2-7 に、Javax.servlet.ServletOutputStream クラスのメソッドと同等の JWeb のメソッドを示します。

表 2-7 Javax.servlet.ServletOutputStream クラスのメソッドと同等の JWeb のメソッド

JWeb のメソッド	サーブレットのメソッド
oracle.html.HtmlStream.print()	javax.servlet.ServletOutputStream.print()
oracle.html.HtmlStream.println()	javax.servlet.ServletOutputStream.println()

表 2-8 に、javax.servlet.ServletInputStream クラスのメソッドと同等の JWeb のメソッドを示します。

表 2-8 javax.servlet.ServletInputStream クラスのメソッドと同等の JWeb のメソッド

JWeb のメソッド	サーブレットのメソッド
oracle.owas.wrb.services.http.MultipartElement()	javax.servlet.ServletInputStream.readLine()

3

Oracle Application Server カートリッジの移行

この章では、Oracle Application Server カートリッジの機能と、Oracle Application Server 10*g* での対応する機能とを比較し、カートリッジを Oracle Application Server 10*g* に移行する際の考慮点について説明します。

項目は次のとおりです。

カートリッジ・タイプと対応する Oracle Application Server 10*g* モジュール

表 3-1 に、対応する Oracle Application Server のカートリッジ・タイプと、それぞれの Oracle Application Server 10*g* コンポーネントを示します。

表 3-1 カートリッジ・タイプ

Oracle Application Server のカートリッジ・タイプ	Oracle Application Server 10*g* で対応するもの
PL/SQL	`mod_plsql`
Perl	`mod_perl`
LiveHTML	Apache SSI および JSP
CWeb	カスタム Apache モジュール、FastCGI、CGI、Java JNI および PL/SQL コールアウト

各アプリケーション・カートリッジの移行方法について、次の各項で説明します。

注意： Oracle Application Server では Perl バージョン 5.004_01 を使用しますが、Oracle Application Server 10*g* では Perl バージョン 5.6.1 を使用します。コードの変更が必要な場合は、Perl の適切なバージョンを使用してください。

PL/SQL の移行

Oracle Application Server PL/SQL カートリッジ・アプリケーションを Oracle Application Server 10*g* `mod_plsql` に移行できます。`mod_plsql` および PL/SQL カートリッジ・アプリケーションのどちらも、Web 上で PL/SQL ベースのアプリケーションを構築および配置するための同様のサポートを提供します。

`mod_plsql` は、Oracle HTTP Server モジュールとして実行されます。mod_plsql は、Oracle データベース内で PL/SQL ロジックを実行する PL/SQL プログラムに、HTTP リクエストの処理を委任します。

関連項目： Oracle Application Server から Oracle Application Server 10*g* に PL/SQL アプリケーションを移行する場合は、『Oracle Application Server 10*g* mod_plsql ユーザーズ・ガイド』を参照してください。

Oracle Application Server のいくつかの PL/SQL カートリッジの機能に対するサポートは、Oracle Application Server 10*g* の PL/SQL では変更されています。この項の続きの部分で、これらの機能を使用する Oracle Application Server アプリケーションの移行方法について説明します。

ファイルのアップロードおよびダウンロード

表 3-2 に、Oracle Application Server と Oracle Application Server 10*g* でサポートされているファイルのアップロードおよびダウンロード機能を示します。

表 3-2 ファイルのアップロードおよびダウンロード機能の比較

ファイルのアップロードおよびダウンロード機能	Oracle Application Server でのサポート	Oracle Application Server 10*g* でのサポート
文字変換なしでロー・バイト・ストリームとしてのファイルのアップロード / ダウンロード	はい	はい
LONG RAW 列型へのファイルのアップロード	はい	はい
BLOB 列型へのファイルのアップロード	いいえ	はい
CLOB 列型、NCLOB 列型へのファイルのアップロード	いいえ	はい
ファイルのアップロード時にデータベース・アクセス記述子（DAD）ごとに表を指定	いいえ - WEBSYS スキーマにのみアップロード	はい
アップロード / ダウンロード時のファイルの圧縮 / 解凍	はい	いいえ
1 回のフォーム送信につき複数のファイルのアップロード	はい	はい

関連項目：『Oracle Application Server 10*g* mod_plsql ユーザーズ・ガイド』を参照してください。

アップロード・ファイルのドキュメント形式

Oracle Application Server の PL/SQL カートリッジと Oracle Application Server 10*g* の mod_plsql は、どちらもアップロード・ファイルをサポートしています。ただし、それぞれ異なるドキュメント表スキーマを使用します。表 3-3 に、Oracle Application Server 10*g* のドキュメント表の列が Oracle Application Server から値を導出する方法を示します。

表 3-3 導出された列値

Oracle Application Server 10*g* のドキュメント表の列	Oracle Application Server の table.column の値
NAME	ows_object.name
MIME_TYPE	ows_fixed_attrib.content_type
DOC_SIZE	ows_content.length
DAD_CHARSET	ows_fixed_attrib.character_set
LAST_UPDATED	ows_object.last_modified
CONTENT_TYPE	"BLOB"
CONTENT	NULL
BLOB_CONTENT	OWS_CONTENT.content

Oracle Application Server から取得した内容は、必ず Oracle Application Server 10*g* のドキュメント表の BLOB_CONTENT 列に格納されます。またこのツールは、Oracle Application Server 10*g* の document 表にロードされるデータが必ず非圧縮データであるようにします。つまり、データが圧縮されている場合は（OWS_ATTRIBUTES 表のエントリのチェックによって検証）、zlib ライブラリを使用してデータを解凍してから Oracle Application Server 10*g* のドキュメント表にロードします。

カスタム認証

カスタム認証は、Oracle Application Server で、自ら（つまりアプリケーション内で）アクセスを制御するアプリケーションで使用されます。アプリケーションは、データベース・レベルではなく、アプリケーション・レベルでユーザーを認証します。

mod_plsql ではカスタム認証もサポートされます。

> **関連項目：**『Oracle Application Server 10*g* mod_plsql ユーザーズ・ガイド』を参照してください。

柔軟なパラメータの受渡し

柔軟なパラメータの受渡し方式では、PL/SQL プロシージャのオーバーロードが可能です。これにより、同じプロシージャ名を再利用して、フォームからプロシージャに渡すパラメータ数によってプロシージャの動作を変更することが可能です。

Oracle Application Server および Oracle Application Server 10*g* のいずれも柔軟なパラメータの受渡しをサポートしています。mod_plsql で柔軟なパラメータの受渡しを使用するには、実行する URL 内でプロシージャ名の先頭に感嘆符（!）を付加します。

たとえば、次の URL により Oracle Application Server のプロシージャが実行されるとします。

```
http://host/virtual_path/procedure?x=1&y=2
```

mod_plsql のプロシージャを実行する URL は、次のようになります。

```
http://host/virtual_path/!procedure?x=1&y=2
```

関連項目：『Oracle Application Server 10*g* mod_plsql ユーザーズ・ガイド』を参照してください。

位置パラメータの受渡し

Oracle Application Server の PL/SQL カートリッジでは、位置パラメータの受渡し方式をサポートしています。この機能は Oracle Application Server 10*g* ではサポートされていないため、使用できません。

関連項目：『Oracle Application Server 10*g* mod_plsql ユーザーズ・ガイド』を参照してください。

SQL ファイルの実行

データベースに格納されている PL/SQL プロシージャの実行に加え、Oracle Application Server の PL/SQL カートリッジでは、ファイル・システムの PL/SQL ソース・ファイルも実行できます。ソース・ファイルには、ファンクションまたはプロシージャを定義しない無名 PL/SQL ブロックが含まれています。この機能により、ユーザーは PL/SQL 文をデータベースに格納しなくても実行できます。この機能は、編集するたびにプロシージャをデータベースにリロードする必要がないため、PL/SQL コードのプロトタイプの作成時に役立ちます。

Oracle Application Server 10*g* ではこの機能をサポートしません。ユーザーは、無名ブロックに名前を割り当て、データベースでストアド・プロシージャとしてコンパイルする必要があります。

Perl の移行

この項では、Oracle Application Server における Perl カートリッジ・アプリケーションの実装方法と、Oracle Application Server 10*g* への移行方法について説明します。項目は次のとおりです。

Oracle Application Server の Perl アプリケーション

Oracle Application Server で実行可能な Perl アプリケーションには 2 種類あります。

- CGI スクリプトとして実行される Perl スクリプト
- Perl カートリッジを使用して実行される Perl スクリプト

CGI スクリプトとして Oracle Application Server で実行される Perl スクリプトは、標準の Perl インタプリタを使用します。これは、Oracle Application Server のインストールとは別に、Perl の実行モジュールとしてシステムにインストールする必要があります。

Perl カートリッジを使用して Oracle Application Server で実行される Perl スクリプトは、カートリッジ内に含まれている、Perl バージョン 5.004_01 に基づいた Perl インタプリタを使用します。 このインタプリタは、次のように構築されています。

- UNIX: `libperlctx.so`: 共有オブジェクト
- Windows: `perlnt40.dll`: 共有ライブラリ

Perl カートリッジは、実行時に共有オブジェクトまたはライブラリとリンクします。

カートリッジ・スクリプトと CGI スクリプトの違い

カートリッジによるインタプリタの実行方法が違うため、Perl カートリッジ用に作成されたスクリプトと CGI 環境用のスクリプトは異なります。Perl カートリッジは、次のことを行います。

- 永続的なインタプリタを維持し、Perl スクリプトをプリコンパイルし、キャッシュします（これによりパフォーマンスが向上します）。
- `stdin` および `stdout` を WRB クライアントの入出力（ブラウザなど）にリダイレクトします。
- `stderr` を WRB Logger にリダイレクトします。
- Perl インタプリタがシステム環境変数をコールすると、追加の CGI 環境変数を Perl インタプリタに返します。

- fork コールのかわりにシステム・コールをサポートします。システム・コールは、Perl インタプリタの実装を変更し、子プロセスの出力を WRB クライアントの入出力にリダイレクトします。
- エラーのロギングをサポートします。
- パフォーマンスの計測をサポートします。

Oracle Application Server 10*g* CGI 環境の CGI 環境下にある Oracle Application Server 向けに開発された Perl スクリプトを、Perl スクリプトのインタプリタ行を変更後も実行できます。Oracle Application Server の Perl カートリッジ向けの Perl スクリプトを変更して、Oracle Application Server 10*g* で実行することもできます。

Perl カートリッジ・スクリプトの移行

この項では、Oracle Application Server と Oracle Application Server 10*g* の Perl 実装、および Perl スクリプトを Oracle Application Server 10*g* に移行するためのコードの変更について説明します。

Oracle Application Server 10*g* の Perl 環境

Oracle Application Server 10*g* の Perl 環境は、`mod_perl` に基づいています。Oracle Application Server の実装のように、`mod_perl` は、サーバーに埋め込まれた永続的な Perl インタプリタと、モジュールおよびスクリプトを 1 回だけロードおよびコンパイルしてキャッシュから処理するコード・キャッシング機能を備えています。Oracle Application Server の Perl カートリッジのように、`mod_perl` は `stdout` をリスナーにリダイレクトします。

Perl モジュール

表 3-4 に、Oracle Application Server および Oracle Application Server 10*g* の両方に関連付けられたサード・パーティの Perl モジュールの比較を示します。そのため、これらのモジュールを使用するアプリケーションを Oracle Application Server から Oracle Application Server 10*g* に移行するには、これらのモジュールを入手し、インストールする必要があります。ファイルは、次の URL から入手可能です。

`http://www.cpan.org`

表 3-4 サード・パーティの Perl モジュールの比較

Perl モジュール	Oracle Application Server でのバージョン	Oracle Application Server 10*g* でのバージョン
`DBI`	0.79	1.20
`DBD::Oracle`	0.44	1.12
`LWP` または `libwww-perl`	5.08	5.53_94

表 3-4 サード・パーティの Perl モジュールの比較 （続き）

Perl モジュール	Oracle Application Server でのバージョン	Oracle Application Server 10*g* でのバージョン
CGI	2.36	2.752
MD5	1.7	2.14
IO	1.15	1.20
NET	1.0502	1.0703
Data-Dumper	2.07	なし
Apache DBI	なし	0.88
Devel::Symdump	なし	2.01
Digest::HMAC	なし	1.01
Digest::MD2	なし	2.00
Digest::SHA1	なし	2.00
HTML::Parser	なし	3.25
MIME::Base64	なし	2.12
PlRPC	なし	0.2015
Storable	なし	1.0.12
Net::Daemon	なし	0.35
Time::HiRes	なし	1.20
URI	なし	1.15

Oracle Application Server の Perl カートリッジのバリエーション

Oracle Application Server の Perl カートリッジと、Oracle Application Server 10*g* の mod_perl とでは、次の点に注意する必要があります。

名前空間の競合

Oracle Application Server および Oracle Application Server 10*g* の両方で、コンパイル済の Perl スクリプトはキャッシュされます。複数の Perl スクリプトをキャッシングした場合、正しく処理しないと名前空間の競合が起こる可能性があります。この競合を回避するために、Oracle Application Server と Oracle Application Server 10*g* はいずれも Perl スクリプトのファイル名を一意のパッケージ名に変換し、その後 eval を使用してコードをパッケージにコンパイルします。その後、そのスクリプトは一意のパッケージ名のサブルーチンとしてコンパイル済の形式となり、Perl アプリケーションから使用可能になります。

Oracle Application Server と Oracle Application Server 10*g* では、パッケージ名の作成方法が異なります。Oracle Application Server では、同じ名前のサブルーチンをキャッシュできません。Oracle Application Server 10*g* では、`Apache::ROOT::` と URL のパスを（「`::`」を「`/`」に置換して）付加することにより、パッケージ名を作成します。

cgi-lib.pl の使用

`cgi-lib.pl` を使用する Oracle Application Server の Perl スクリプトは、Perl カートリッジ用にカスタマイズされたバージョンのライブラリを使用するよう、変更する必要があります。Oracle Application Server 10*g* の場合、この必要はありません。

モジュールのプリロード

Oracle Application Server の Perl スクリプトには、スクリプトの要求のたびに繰り返し実行する必要がない命令が含まれていることがあります。このような命令を 1 回だけ実行し、各 Perl スクリプトの要求時には必要な部分のみ実行するようにすると、パフォーマンスが向上します。

Oracle Application Server では、`perlinit.pl` により、モジュールのプリロードと初期タスクの実行が行われます。このファイルは、カートリッジ・インスタンスの起動時に一度だけ実行されます。デフォルトでは、このファイルには実行可能な文は含まれていません。このファイルは、Perl アプリケーションの設定フォームの初期化スクリプト・パラメータで指定します。

Oracle Application Server 10*g* で対応するプリロード・スクリプトは `startup.pl` です。

> **関連項目：** `mod_starup.pl` の詳細は、`http://perl.apache.org` を参照してください。

LiveHTML の移行

Oracle Application Server では、HTML ページにサーバー側インクルード（SSI）およびスクリプトを埋め込むか、あるいは Perl をスクリプトに使用することにより、LiveHTML カートリッジを使用して動的コンテンツを生成できます。LiveHTML アプリケーションを Oracle Application Server 10*g* に移行する場合は、LiveHTML の SSI を Apache の SSI に移行する必要があります。現在 Oracle Application Server 10*g* で LiveHTML の埋込みスクリプトに対応するのは、JSP のみです。

SSI

表 3-5 に、Apache および LiveHTML で使用可能な SSI を示します。

表 3-5 Apache および LiveHTML の SSI のリスト

Apache の SSI	LiveHTML の SSI
config	config
echo	echo
exec	exec
fsize	fsize
flastmod	flastmod
include	include
printenv	使用不可
set	使用不可
使用不可	request

Apache または LiveHTML で SSI を指定する場合の構文は同じです。たとえば、次のようになります。

```
<!--#config sizefmt="bytes" -->
```

注意： 最後の終了記号 "-->" の前には空白が必要です。

Apache の SSI は、mod_include モジュールによって実装されます。このモジュールは、デフォルトで Oracle HTTP Server にコンパイルされます。

前の表で示した要素に加え、Apache の SSI には変数置換およびフロー制御用の要素が含まれています。

スクリプト

Oracle Application Server では、LiveHTML カートリッジを使用して、Perl スクリプトを HTML ファイルに埋め込むことが可能です。Oracle Application Server 10*g* には、これに対応する機能はありません。しかし次の方法があります。

1. ロジックは Perl のままにして、`mod_perl` を使用します。たとえば、HTML ピースを `printf()` に変更できます。
2. HTML はそのままにして、プログラミング言語を PL/SQL に変更します。
3. Perl をスクリプト言語として HTML とともに使用できるツールを Web からダウンロードします。たとえば `http://perl.apache.org` などを利用します。

   注意： このツールは、`mod_perl` の最上部で実行されます。そのため移行アプローチは、このページに記載された他の 3 つのアプローチと比較して、最も簡単です。

4. HTML をそのままにして、プログラミング言語を JSP などの Java に変更します。Oracle Application Server 10*g* は JSP 1.2 仕様に準拠しています。LiveHTML アプリケーションを Oracle Application Server 10*g* に移行するには、次のことを行う必要があります。
   a. LiveHTML アプリケーション・モデルを JSP アプリケーション・モデルに移行する。
   b. LiveHTML タグを JSP タグに移行する。
   c. Perl コードを Java コードとして再作成する。

   注意： ご使用の LiveHTML アプリケーションが Oracle Application Server で Web アプリケーション・オブジェクトを使用している場合、この機能を埋込み Java コードまたは JavaBeans クラスとして実装し、JSP の `<jsp:useBean>` タグで宣言する必要があります。

   関連項目：『Oracle Application Server Containers for J2EE JavaServer Pages 開発者ガイド』を参照してください。

   注意： Oracle Application Server 10*g* では、WRB API は提供されません。

CWeb の移行

Oracle Application Server では、CWeb カートリッジを使用して次のことができます。

- カスタム・カートリッジの作成
- 他のカートリッジから起動されるアプリケーションの開発

Oracle Application Server CWeb カートリッジから Oracle Application Server 10*g* への移行パスには、次の項目が含まれます。

- FastCGI の使用
- カスタム Oracle Application Server 10*g* モジュールの作成

FastCGI の使用

CWeb カートリッジは、本質的には .dll ライブラリまたは .so ライブラリです。このライブラリのエントリ・ポイントを管理ページで指定することによって Oracle Application Server 環境に統合し、Web の URL にマップすることができます。Oracle Application Server Infrastructure により、該当する URL がブラウザからリクエストされると、ライブラリ（CWeb カートリッジ）のエントリ・ポイントが起動されます。さらに、CWeb カートリッジでは、WRB インフラストラクチャからの複数の API を、クライアント情報およびその他の環境情報へのアクセスに使用できます。

CGI は、Oracle Application Server 10*g* をはじめとするすべての Web サーバーでサポートされている標準です。CGI プログラムにマップする URL にアクセスすると、Web サーバーによってプログラムが起動され、その結果がブラウザに返されます。

そのため、CWeb を移行する簡単な方法の 1 つは、起動中に CWeb カートリッジのエントリ・ポイントを起動する簡単な C プログラムを書くことです。

WRB API および他の Oracle Application Server Infrastructure に依存する機能は、新しい Oracle Application Server 10*g* 環境では使用できません。このような WRB API や機能が使用されていた場合は、代替の API を使用するために CWeb カートリッジを変更する必要があります。

Infrastructure の観点から見ると、CWeb カートリッジはロード・バランスが取れていました。新規のインスタンスは各リクエストで起動しませんでした。

FastCGI は、仕様、プロトコル、API、さらに実装までを規格化したもので、CGI にかわるオープンな規格です。要約すると、FastCGI は別個のプロセスを起動し、プロセスが使用可能でありリクエストのライフスタイルに依存しない状態を保ちます。FastCGI プログラムは、リスニングする開始ポイントおよびイベントの特定の標準に準拠する必要があり、その点では Java サーブレット仕様と同様です。そのため、そのライフ・サイクルはインフラストラクチャによって制御されます。

CWeb カートリッジの移行は FastCGI プログラムを書くことに類似しています。仕様に準拠させてから、CWeb カートリッジのエントリ・ポイントをコールします。

関連項目： FastCGI の例は、`http://www.fastcgi.com` を参照してください。

カスタム Oracle Application Server 10*g* モジュールの作成

CWeb を使用して作成されたカスタム・カートリッジが存在する場合、カスタム Oracle Application Server 10*g* モジュールの作成を検討することもできます。

C プログラムの実行に CWeb を使用している場合、次のオプションがあります。

- **CGI スクリプト**: `println` 文を使用して Web コンテンツを生成するスタンドアロンの C プログラム
- **Java JNI**: CWeb ルーチンを OC4J からコールする Java サーブレットまたは JSP
- **PL/SQL コールアウト**: Oracle 10*g* から CWeb ルーチンをコールする PL/SQL アプリケーション

注意： Oracle Application Server 10*g* では、WRB および CWeb の API は提供されません。

4

EJB、ECO/Java および JCORBA アプリケーションの移行

この章では、Oracle Application Server の EJB、ECO for Java および JCO のアプリケーションを Oracle Application Server 10*g* の OC4J に移行する方法について説明します。

項目は次のとおりです。

EJB の OC4J への移行

EJB を Oracle Application Server 4.x から OC4J に移行するには、次の分野のコードを変更する必要があります。

- デプロイメント・ディスクリプタ
- クライアント・コード
- ロギング（サーバー・コード）（該当する場合）

> **注意：** Oracle Application Server EJB は EJB 標準に準拠していませんが、Oracle Application Server 10*g* EJB は EJB 2.0 仕様全体に準拠しています。移行中はご使用のコードを適宜変更してください。

これらの変更について、この後の項で説明します。

デプロイメント・ディスクリプタ

OC4J は、J2EE 1.2 仕様に準拠する XML ファイル構成に準拠します。

> **関連項目：**『Oracle Application Server Containers for J2EE JavaServer Pages 開発者ガイド』を参照してください。

クライアント・コード

クライアント・コードの変更は、JNDI を使用して初期コンテキスト・コールで行われます。初期コンテキスト・コールに渡されるハッシュテーブルには、次のプロパティすべてを含める必要があります。

- `javax.naming.Context.URL_PKG_PREFIXES`
- `javax.naming.Context.SECURITY_AUTHORIZATION`
- `javax.naming.Context.SECURITY_PRINCIPAL`
- `javax.naming.Context.SECURITY_CREDENTIALS`

> **関連項目：**『Oracle Application Server Containers for J2EE ユーザーズ・ガイド』を参照してください。

また、ご使用の EJB ホームにアクセスする URL を、OC4J の次のような URL に変更する必要があります。

`ORMI://<host>:<port>/<path>/<bean>`

たとえば、次のようになります。

```
ORMI://myhost:2481/test/myBean
ORMI://host/port/est/bean
```

ロギング（サーバー・コード）

Oracle Application Server でアプリケーションのロギングを行っている場合は、EJB コードから oracle.oas.ejb.Logger のリファレンスをすべて削除してください。

ECO/Java の OC4J への移行

Oracle Application Server の ECO for Java（ECO/Java）を Oracle Application Server 10*g* の OC4J に移行するときは、前項で EJB の移行に関して説明したデプロイメント・ディスクリプタおよびクライアント・コードの変更だけでなく、この項で説明するサーバー・コードも変更する必要があります。

ご使用の ECO/Java コンポーネントに OC4J との互換性を持たせるには、実装ファイル、リモート・インタフェース・ファイル、ホーム・インタフェース・ファイルおよびデプロイメント・ディスクリプタを変更する必要があります。

リモート・インタフェース

oracle.oas.eco.ECOObject ではなく、javax.ejb.EJBObject を継承するようにリモート・インタフェースを変更します。各メソッドで java.rmi.RemoteException をスローする必要があります。

ホーム・インタフェース

oracle.oas.eco.ECOHome ではなく、javax.ejb.EJBHome を継承するようにホーム・インタフェースを変更します。

作成されたメソッドでは、oracle.oas.eco.CreateException ではなく、javax.ejb.CreateException および java.rmi.RemoteException をスローする必要があります。

実装クラス

実装クラスを次のように変更します。

1. oracle.oas.eco.Logger およびそのリファレンスをすべて削除する。
2. oracle.oas.eco.* をすべて javax.ejb.* に変更する。
3. ECOCreate メソッドを ejbCreate メソッドに変更する。

4. `ECORemove` メソッドを `ejbRemove` メソッドに変更する。
5. `ECOActivate` メソッドを `ejbActivate` メソッドに変更する。
6. `ECOPassivate` メソッドを `ejbPassivate` メソッドに変更する。
7. OC4J では XML ファイルを配置に使用するので、適切な配置ファイルを作成する必要があります。

JCORBA の OC4J への移行

Oracle Application Server リリース 4.0.6 および 4.0.7 では、Java CORBA Object（JCO）と呼ばれる ECO/Java モデルの先行コンポーネント・モデルが提供されていました。Oracle Application Server 10*g* では、CORBA オブジェクトをサポートしません。CORBA オブジェクトを EJB としてコードを再作成する必要があります。この項では、Oracle Application Server の JCO から Oracle Application Server 10*g* の OC4J への移行について説明します。

OC4J へ移行するには、この項で説明するように、サーバー・コードおよびクライアント・コードを変更する必要があります。サーバー・コードを変更するには、リモート・インタフェースの変更、ホーム・インタフェースの作成、JCORBA オブジェクト実装の変更およびパラメータのシリアライズ化が必要です。

> **関連項目：** さらに、4-2 ページの「デプロイメント・ディスクリプタ」で説明したように、デプロイメント・ディスクリプタを変更する必要があります。

リモート・インタフェース

リモート・インタフェースを次のように変更します。

1. `org.omg.CORBA.Object` または `oracle.oas.jco.JCORemote` をすべて `javax.ejb.EJBObject` に変換します。
2. インタフェースのすべてのメソッドに対して、`java.rmi.RemoteException` をスローします。

ホーム・インタフェース

EJB 仕様で定義されているようにホーム・インタフェースを作成する必要があります。次に例を示します。

```
import javax.ejb.*;
import java.rmi.RemoteException;
public interface ServerStackHome extends EJBHome
{
   public ServerStackRemote create() throws CreateException, RemoteException;
}
```

オブジェクトの実装

実装クラスを移行するために次の手順を行います。

1. `import oracle.oas.jco.*` を `import javax.ejb.*` に変更します。
2. そのクラスで `javax.ejb.SessionBean` あるいは `javax.ejb.EntityBean` が実装されることを確認します。

> **注意：** JCORBA のライフ・サイクルは、OC4J 内でサポートされていません。JCORBA オブジェクトによって `oracle.oas.jco.Lifecycle` が実装された場合は、削除する必要があります。

3. Logger のリファレンスが存在する場合はすべて削除します。
4. 初期化操作が存在する場合は、すべて `ejbCreate()` メソッドに移動します。
5. `setSessionContext()` メソッドに渡されたセッション・コンテキストをインスタンス変数に保存します。
6. クラス内のパブリック・メソッドがすべて `java.rmi.RemoteException` をスローすることを確認します。
7. `ObjectManager` タイプが存在する場合は、すべて `SessionContext` タイプに変更します。表 4-1 に、`ObjectManager` クラスのメソッドから `SessionContext` クラスのメソッドへのマッピングを示します。

表 4-1 ObjectManager メソッドおよび SessionContext メソッド

SessionContext メソッド	ObjectManager メソッド
getEnvironment()	getEnvironment()
setSessionContext() に渡されるパラメータ	getObjectManager()
getEJBObject()	getSelf()
getEJBObject().remove()	revokeSelf()
getUserTransaction()	getCurrentTransaction()

パラメータのシリアライズ化

リモート・インタフェースでユーザー定義のパラメータが渡されている場合、そのクラスで `java.io.Serializable` が実装されることを確認してください。

索引