

# Oracle8*i*

Oracle Servlet Engine ユーザーズ・ガイド

リリース 8.1

2000 年 11 月

部品番号 : J02339-01

ORACLE®

---

Oracle8i Oracle Servlet Engine ユーザーズ・ガイド, リリース 8.1

部品番号 : J02339-01

原本名 : Oracle Servlet Engine User's Guide, Release3(8.1.7)

原本部品番号 : A83720-01

原本著者 : Susan Kraft, John Russell

原本協力者 : Ellen Barnes, Jose Fernandez, Hal Hildebrand, Sunil Kunisetty, Angela Long, Jasen Minton, Brian Wright, Ronald Decker, Kannan Muthukkaruppan

Copyright © 1996, 1999, 2000, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されています。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

\* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

---

---

# 目次

<b>はじめに</b> .....	vii
対象読者 .....	vii
Java 情報リソース .....	viii
 <b>1 概要</b>	
<b>Web アプリケーションの使用</b> .....	1-2
セッション・メモリーおよび拡張性 .....	1-2
データベース内の OSE .....	1-4
データベース・セッション .....	1-4
<b>Web サービス</b> .....	1-5
Web ドメイン .....	1-6
シングル・ドメイン .....	1-7
マルチ・ドメイン .....	1-8
サーブレット・コンテキスト .....	1-10
サーブレット .....	1-11
Java Naming and Directory Interface (JNDI) .....	1-12
アクセス権とスキーマ .....	1-12
<b>制御階層の概要</b> .....	1-13
<b>セキュリティ</b> .....	1-13
 <b>2 OSE でのサーブレットの基本</b>	
JNDI .....	2-2
セッション・シェル .....	2-3
オブジェクト・パラメータの取得と設定 .....	2-3
仮想パス .....	2-4

プロパティ・グループ .....	2-4
サーブレットの配置 .....	2-5
サーブレット・コンテキストで公開されるサーブレットのクラス .....	2-5
公開されたサーブレット .....	2-5
HTTP 要求の処理 .....	2-7
Web ドメインの検索 .....	2-7
シングル・ドメインの Web サービス : .....	2-7
マルチ・ドメインの Web サービス : .....	2-7
サーブレット・コンテキストの検索 .....	2-7
サーブレットへの仮想パスのマッピング .....	2-8
デフォルトのサーブレットの試用 - 静的コンテンツ .....	2-8

### 3 JNDI およびセッション・シェル

JNDI について .....	3-2
JNDI アクセス権 .....	3-2
セッション・シェルの起動 .....	3-2
ディレクトリのナビゲーションと管理 .....	3-3
アクセス権および所有権 .....	3-3
OSE でのセッション・シェルのコマンドの概要 .....	3-3
サービスの構成 .....	3-4
Web ドメインの構成 .....	3-4
セキュリティ管理 .....	3-4
サーブレット・コンテキストの管理 .....	3-5
サーブレットの管理 .....	3-5
エクスポート・コマンド .....	3-6

### 4 アーキテクチャ

HTTP 要求 .....	4-2
URL .....	4-2
HTTP 要求のライフ・サイクル .....	4-2
エンド・ポイント .....	4-3
Web ドメインの検索 .....	4-3
サーブレット・コンテキストの検索 .....	4-3
サーブレットの検索 .....	4-4
デフォルトのサーブレットの検索 .....	4-4

データベース・セッション .....	4-5
データベース・セッションのユーザー認証 .....	4-7
データベース・セッション .....	4-7
データベース・セッションの作成、終了、タイムアウト .....	4-7
データベース・セッションのタイムアウト .....	4-7
データベース・セッションの自動終了 .....	4-8
HTTP セッションの作成、終了およびタイムアウト .....	4-8

## 5 Oracle OSE の Apache モジュール

Apache での OSE の要件 .....	5-2
OSE の Apache での mod_ose の概要 .....	5-2
OSE の Apache での mod_ose の構成 .....	5-3
tnsnames.ora の接続記述子と構文 .....	5-4
OSE アプリケーションでの Apache の構成 .....	5-5
Apache のステートフルなアプリケーションとステートレスなアプリケーション .....	5-5
Apache のステートフルなハンドラとステートレスなハンドラ .....	5-6
Apache の構成情報の抽出 .....	5-8
mod_ose を使用したサイトのトポロジ .....	5-10
事前インストールされている Apache システムのための mod_ose のインストール .....	5-11
Apache 構成 .....	5-11
OSE サービスの指定 .....	5-11
動的要求の指定 .....	5-12
URL の転送 .....	5-13

## 6 OSE サーバー構成

OSE の設定 .....	6-2
サービスの作成 .....	6-2
ドメインの作成 .....	6-2
コンテキスト・グループ .....	6-2
MIME グループ .....	6-3
サブレット・コンテキストの作成 .....	6-3
MIME グループ .....	6-3
サブレットの追加 .....	6-3

## 7 開発者用のツールと手順

Web サービス .....	7-2
Web サービスの仮想ホストと IP ドメイン .....	7-2
Web サービスのポートの変更 .....	7-3
サブリット・コンテキストの管理 .....	7-3
サブリット・コンテキストの作成または変更 .....	7-3
サブリット・コンテキストの削除 .....	7-4
サブリット・コンテキストの内容 .....	7-4
config オブジェクト .....	7-5
doc_root オブジェクト .....	7-5
named_servlets サブディレクトリ .....	7-5
defaultServlet オブジェクト .....	7-5
policy ディレクトリ .....	7-5
httpSecurity .....	7-6
サブリット・コンテキスト・グループ・パラメータ .....	7-6
context.params .....	7-7
context.mime .....	7-7
context.servlets .....	7-7
context.error.uris .....	7-7
公開されたサブリットの管理 .....	7-8
サブリット・コンテキストで公開されたサブリット・クラス .....	7-8
サブリットの作成 .....	7-10
サブリットのソースコードの作成 .....	7-10
サブリットのコンパイル .....	7-10
サブリット・コードのロード .....	7-10
サブリットの公開 .....	7-11
サブリットへのアクセス .....	7-11

## 8 セキュリティ HTTP の管理

概要 .....	8-1
Web サーバー・セキュリティの前提条件 .....	8-2
認証と許可 .....	8-2
プリンシパルの宣言 .....	8-3
グループ .....	8-3
ユーザー .....	8-3

レルム .....	8-4
DBUSER の考慮事項 .....	8-4
ツール .....	8-5
コマンド構造 .....	8-5
レルム .....	8-5
プリンシパル .....	8-6
詳細 .....	8-7
リソースの保護 .....	8-7
コマンド構造 .....	8-8
詳細 .....	8-8
アクセス権の宣言 .....	8-9
コマンド構造 .....	8-9
セキュリティ・サブレットの宣言 .....	8-10
コマンド構造 .....	8-10
トラブルシューティング .....	8-10

## A PL/SQL サブレットの作成

PL/SQL サブレットの概要 .....	A-2
PL/SQL サブレット実行のための mod_ose の構成 .....	A-2
ステートフルな PL/SQL ストアド・プロシージャの作成 .....	A-3
アプリケーションからのデータベース・アクセス記述子の構成 .....	A-4
パッケージ DBMS_EPGC .....	A-7
サブプログラムの概要 .....	A-9
CREATE_INSTANCE プロシージャ .....	A-9
DROP_INSTANCE プロシージャ .....	A-9
DROP_ALL_INSTANCES プロシージャ .....	A-9
GRANT_ADMIN プロシージャ .....	A-10
REVOKE_ADMIN プロシージャ .....	A-10
GET_ADMIN_LIST プロシージャ .....	A-10
SET_GLOBAL_ATTRIBUTE プロシージャ .....	A-10
GET_GLOBAL_ATTRIBUTE プロシージャ .....	A-10
DELETE_GLOBAL_ATTRIBUTE プロシージャ .....	A-11
GET_ALL_GLOBAL_ATTRIBUTES プロシージャ .....	A-11
CREATE_DAD プロシージャ .....	A-11
DROP_DAD プロシージャ .....	A-11
SET_DAD_ATTRIBUTE プロシージャ .....	A-11
GET_DAD_ATTRIBUTE プロシージャ .....	A-12

DELETE_DAD_ATTRIBUTE プロシージャ .....	A-12
GET_DAD_LIST プロシージャ .....	A-12
GET_ALL_DAD_ATTRIBUTES プロシージャ .....	A-12
IMPORT プロシージャ .....	A-13
EXPORT プロシージャ .....	A-13

**B 例**

**索引**



---

# はじめに

## 対象読者

このマニュアルは、次の方を対象読者とします。

- 管理者 – データベース内での Java 開発以外の目的で、Oracle8i を購入された方。ただし、管理面について、Oracle8i Java 特性の詳細を知りたい場合は、『Oracle8i Java 開発者ガイド』を参照してください。
- 非 Java 開発者 – Oracle データベースのプログラミングは、PL/SQL およびその他の Java 以外のプログラミングで構成されます。PL/SQL の経験者で、Java に慣れていない開発者のために、『Oracle8i Java 開発者ガイド』の第 1 章では、Java の簡単な概要とオブジェクト指向の概念を説明しています。Java に関する詳細情報については、この章の最後の「[Java 情報リソース](#)」を参照してください。
- Java 開発者 – Java 開発者は、Sun Microsystems 社の仕様に準拠する Java 環境に慣れています。ただし、Java がデータベースに結合されている場合は、Java とデータベースの両方の概念が結合されます。このため、Oracle8i 内の Java 環境は拡張されて、データベースの問題も扱うようになります。このマニュアルの多くの部分で、データベースでの Java の実行を理解するために必要な相違点を扱っています。次に、この概念の結合により生じる 2 つの環境を概説します。
  - \* Java 環境 – Oracle8i は、Java の使用する環境を提供し、すべての 100% Pure Java コードが機能します。Oracle8i JVM では、クラスおよびそのクラスが存在する環境の管理方法に関して、Java 開発に影響を与えます。たとえば、クラスはデータベースにロードする必要があります。また、Oracle8i モデル内のクライアントとサーバーは明確に区別されています。
  - \* データベース環境 – Java オブジェクトを管理するためにデータベース概念を認識する必要があります。このマニュアルでは、明確に定義されている 2 つのレلمム、Oracle8i データベースと Java 環境を適合させる方法を概説します。たとえば、セキュリティ・ポリシーを決定する場合には、全体のセキュリティ・ポリシーのために、データベース・セキュリティと Java セキュリティの両方を考慮する必要があります。

JavaServer Pages についての詳しい説明は、『Oracle8i JavaServer Pages 開発者ガイドおよびリファレンス』を参照してください。

## Java 情報リソース

『Oracle8i Java Tools リファレンス』には、Oracle Servlet Engine の管理に使用するすべてのコマンドのリストと説明が示されています。

次の表に、Java プログラミング・ドキュメント・セットで説明されている現在の情報のソースをリストします。

場所	説明
<code>http://www.oracle.com/java</code>	Oracle8i データベースの Java に関する最新製品、更新情報およびニュース。このサイトには、よくある質問 (FAQ)、更新された JDBC ドライバ、SQLJ リファレンス実装、および Java アプリケーション開発について詳述する白書が含まれています。また、このサイトから評価および購入用の Java ツールをダウンロードできます。
<code>http://java.sun.com/</code>	Java の中心的なソースである Sun Microsystems 社の Web サイト。このサイトには、チュートリアル、推奨文献、Java Developer's Kit (JDK) などの、Java 製品や情報が示されています。JDK は、 <code>http://java.sun.com/products</code> にあります。
<code>http://java.sun.com/docs/books/jls</code> <code>http://java.sun.com/docs/books/vmspec</code>	Oracle8i JVM は、Java 言語仕様 (JLS) および Java 仮想マシン (JVM) 仕様に基づいています。
<code>comp.lang.java.programmer</code> <code>comp.lang.java.databases</code>	インターネット・ニュースグループは、他の Java 開発者からの Java に関する情報を得る貴重なソースです。これらの 2 つのニュースグループをモニターすることをお勧めします。 <b>注意:</b> オラクル社は、これらのニュースグループのいくつかのアクティビティをモニターし、Oracle 固有の問題に対する回答を投稿しています。

お近くの書店やオンラインの書店には、参考用の Java リファレンスが多数用意されています。初心者向けの資料のリストは、『Oracle8i JavaServer Pages 開発者ガイドおよびリファレンス』で参照できます。またこれらの資料は、一般的なりファレンスとしても使用できます。

Oracle Servlet Engine (OSE) は、Oracle8i データベース内部のスケーラブルなサーブレット・コンテナとして設計された、特殊な Web サーバーとして機能します。

サーブレット・クラスは、loadjava コマンドを使用して、Oracle8i にロードされ、データベース内部の名前空間で公開されます。サーブレット・コンテナは、HTTP 要求を処理し、セッション内で公開されたサーブレットをインスタンス化して、サーブレットのメソッドをコールします。

この章では、次の項目について図を使用して説明します。

- [Web アプリケーションの使用](#)
- [Web サービス](#)
- [制御階層の概要](#)
- [セキュリティ](#)

ドメイン、コンテキストおよびサーブレットを定義するには、作成用コマンドの 1 つを使用します。すべてのコンテンツは、サーブレットによって提供されます。サーブレットの位置は、クライアントにより URL 形式で定義されます。

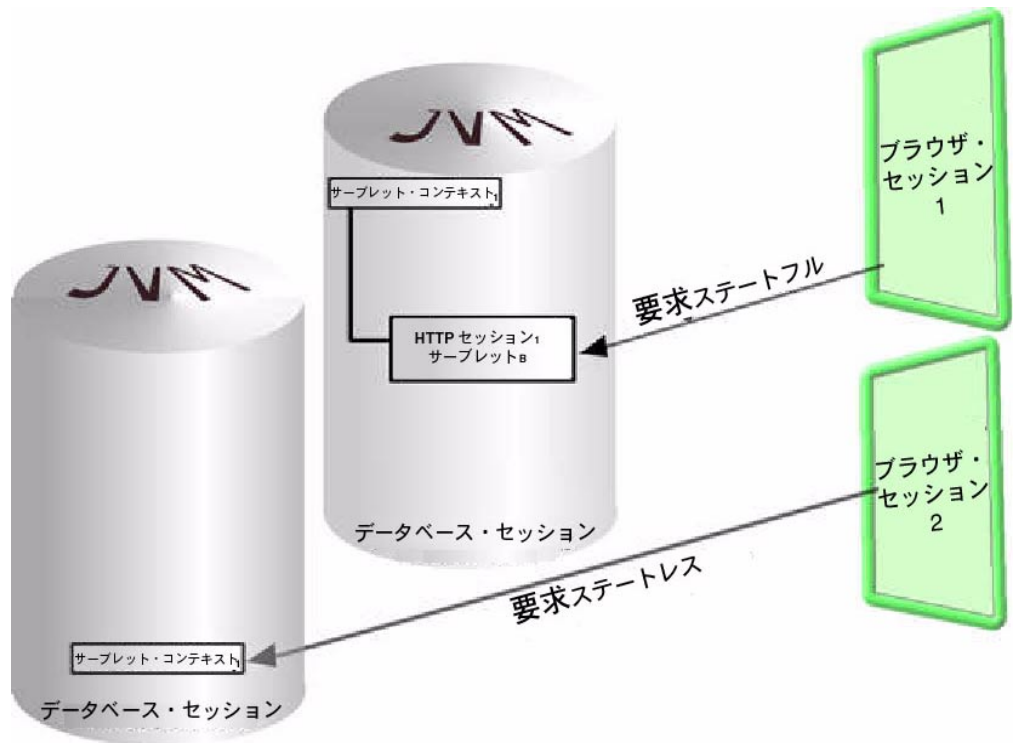
## Web アプリケーションの使用

Web ページは、ハイパーリンク、図形およびフォーマットされたテキスト表示を表す静的な HTML での記述のみを含む HTML ファイルとして定義されます。Web アプリケーションは、静的ページよりもインタラクティブなコンテンツをユーザーに提供します。HTML コンテンツをサーブレットで拡張すると、動的なコンテンツを作成することができ、Web ページが Web アプリケーションとなります。

## セッション・メモリーおよび拡張性

OSE は、仮想的かつ無制限な、線形性の拡張性を目的として設計されています。各ブラウザ・セッションを単一の仮想的な Java 仮想マシン (JVM) と組み合わせて、複数の仮想的な JVM を存在させることが可能です (図 1-1「[仮想的な JVM と組み合わせたブラウザ・セッション](#)」を参照)。仮想的な JVM のセッションをサポートするデータベース・プラットフォームでの唯一の制限は、一時点に扱えるハードウェアの容量のみです。詳細は、『Oracle8i Java 開発者ガイド』を参照してください。

図 1-1 仮想的な JVM と組み合わされたブラウザ・セッション



OSE は静的ページを使用することも、CGI スクリプトを実行することもできますが、本来、Apache、Netscape、IIS などの標準の Web サーバーの背後で動作するサーブレット・コンテナとして配置されることが目的です。第 5 章「Oracle OSE の Apache モジュール」を参照してください。

OSE とその他のサーブレット・エンジンとの相違点の 1 つに、使用するデータベース・セッションの設計方法があります。ブラウザ・セッションは、サーブレット・セッションを生成します。データベース・セッションには、特定のブラウザ・セッション用に作成されたすべての HTTP セッションのオブジェクトが格納されます。データベース・セッション・メモリーを使用すると、オブジェクトは要求（接続）が終了した後も削除されません。ただし、データベース・セッションが終了するとオブジェクトは削除されます。

## データベース内の OSE

OSE は、Web アプリケーションをサポートする Oracle8i 内部で実行される組込み Web サーバーです。各セッション（仮想的な Java VM）では、次の項目が実行されます。

- JavaServer Page (JSP)
- サークレット
- Java ストアド・プロシージャ

## データベース・セッション

セッションは、明示的にまたは管理者が設定したタイムアウトで終了します。詳細は、[第 6 章「OSE サーバー構成」](#)を参照してください。

OSE 固有の属性は次のとおりです。

- すべてのサークルレットは、データベース・セッションによりサポートされるブラウザ・セッションでアクティブになります。
- 各データベース・セッションには独自の仮想的な JVM があります。各データベース・セッションには ブラウザ・セッションごとに 1 つずつ、複数の仮想的な JVM があります。
- OSE では、セッションごとに 1 つの仮想的な JVM を使用し、各ブラウザ・セッションからセッションとそれらの静的変数を切り離します（ブラウザ・セッションごとに新しいスレッドは生成されません）。
- データベース・セッションには、特定のブラウザ・セッション用に作成されたすべての HTTP セッション・オブジェクトが格納されます。
- HTTP セッションは、サークルレットの要求に応じて、ステートフルなサークルレット・コンテキストごとにアクティブになります。
- Oracle のモジュール、mod\_ose は、Oracle HTTP Server (Apache) と、Oracle Servlet Engine (OSE) を格納している Oracle8i データベース間のレイヤーを提供します。mod\_ose は、ポート、サーバーおよび Web ドメインを通信変数を設定して定義します。

## Web サービス

OSE は、1 つ以上の Web サービスに対する実行コンテキストとして機能します。Web サービスを作成する場合は、名前空間でサービスのルートを指定する必要があります。サービスのルートは、Web サービス全体のドメイン情報を格納する最上位レベルになります。

OSE で Web サービスを構成する場合は、Web サービス、そのエンド・ポイント、Web ドメインおよびサブレット・コンテキストを作成します。Web サービスは、1 つ以上のネットワーク・エンド・ポイントと対応付けられます。HTTP クライアントは、Web サービスのエンド・ポイントに接続して、対応付けられている Web ドメインの 1 つからコンテンツを取得します。また、名前空間内の情報に、要求を処理および実行するよう定義されたパスとポイントが含まれるように、すべての属性を設定する必要があります。

OSE は、次の 2 つのタイプの Web サービス構成をサポートします。

- シングル・ドメイン
- マルチ・ドメイン

ほとんどの場合、シングル・ドメインの Web サービスで十分対応できます。シングル・ポートでリスニングし（以降、すべての例で 8080 を使用）、固有のドメインにエンド・ポイントで受信したすべての要求が送信されるように、Web サービスを構成します。

HTTPS をサポートするには、追加のエンド・ポイントをシングル・ドメインの Web サービスに対応付けます。このエンド・ポイントは、別のポート（以降、すべての例で 9090 を使用）をリスニングし、SSL 接続用に構成できます。

マルチ・ドメインの Web サービスは、次の場合のような、複雑な構成で使います。

- 仮想ホスト構成で、複数のインターネット・ドメイン名システム（DNS）名を持つ 1 つのサーバーが、複数の Web ドメインに対応する場合。
- 複数の IP アドレス構成で、複数のネットワーク・インタフェース・カード（NIC）を持つサーバーが、複数の Web ドメインに対応する場合。

マルチ・ドメインでは、ネットワークで、IP アドレスまたはホスト名のいずれか、あるいはその両方が使用されて、要求された接続が確立され、適切なドメインに要求がルーティングされます。

## Web ドメイン

Web ドメインにはサーブレット・コンテキストが格納されますが、IP および仮想ホストのマルチ・ドメインの Web サービスの場合は、別の Web ドメインが格納されます。

HTTP 要求に対して Web ドメインは、URL のアドレス部分で識別されます。Web ドメインのルートは、格納されている Web サービスのサービス・ルートになります。

---

---

**注意：** Web ドメインの構造をデフォルトと異なる構成に変更するには、『Oracle8i Java Tools リファレンス』で説明するツールを使用します。

---

---

各ドメインの構成パラメータは、Web ドメインのディレクトリ・ツリーの config オブジェクト内にあります。これらの構成は、[図 1-2 「シングル・ドメインの名前空間モデルの構造」](#) および [図 1-3 「仮想ホストを持つマルチ・ドメイン、マルチホームの例」](#) に示されています。config オブジェクトは構造の最上位レベルにあることに注意してください。

次に、Web ドメインの階層について説明します。

- Web ドメインには、スキーマ名前空間ディレクトリ所有者と一致する所有者がいます。
- シングル・ドメインの場合、サービス・ルートは、Web ドメインと Web サービス両方のルートになります。（1-7 ページの [図 1-2 「シングル・ドメインの名前空間モデルの構造」](#) を参照してください。）
- IP ホスト・ドメインの場合は、Web ドメイン名は、サービス・ルートに位置するホスト名になります。（1-7 ページの [図 1-3](#) を参照してください。）
- 仮想ホスト・ドメインの場合は、Web ドメイン名はサービス・ルートに位置する仮想ホスト名になります。（1-7 ページの [図 1-3](#) を参照してください。）
- ドメイン・ルート内にある /contexts サブディレクトリには、サーブレット・コンテキスト・ディレクトリが格納されます。
- 公開されたサーブレットは、/named\_servlets ディレクトリ下に位置します。
- /named\_servlets ディレクトリは、名前空間のオブジェクト、config、httpSecurity、policy、defaultServlet および doc\_root の構成パラメータと並列関係にあります。
- doc\_root オブジェクトは、クライアントの静的コンテンツに対するポインタ（ソフト・リンク）です。

名前空間を示すディレクトリ構造によって、Web ドメインを構成する要素の検索および操作が可能です。各ファイルおよびディレクトリは、コンテンツを示すファイルではなく、実際は Oracle8i 内のプロパティを持つオブジェクトです。ただし、この説明のほとんどの部分では、ツールを使用して Web ドメインを作成しているのので、ディレクトリとファイルのモデルで説明できます。



## シングル・ドメイン

たとえば、`http://cavist.com:8080/cellar/welcome.html` という URL が、シングル・ドメインに送信されると、要求は、`cavist.com` という名前のホストの、ポート 8080 でリスニングする Web サービスにアクセスします。

### 例 1-1 シングル・ドメイン：URL 内のポート関係とドメイン・ルート

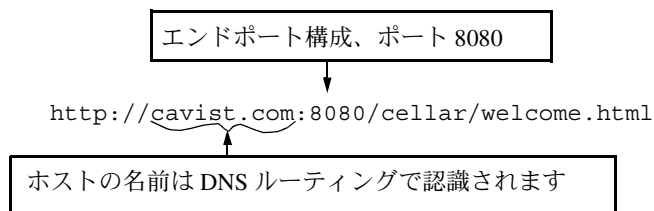
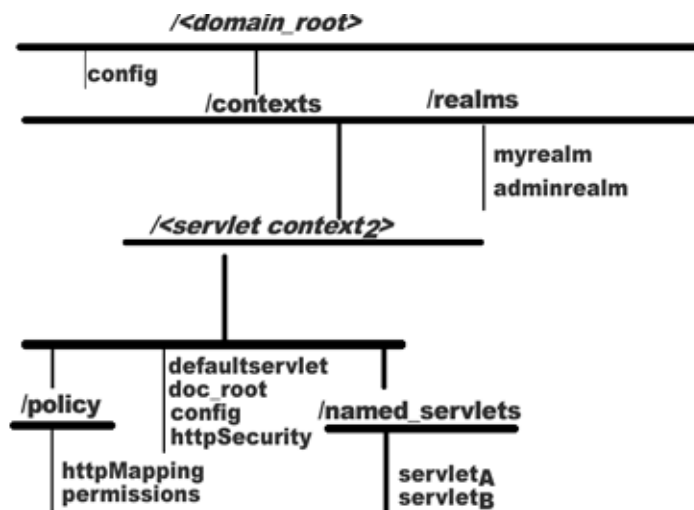


図 1-2 シングル・ドメインの名前空間モデルの構造



## マルチ・ドメイン

マルチ・ドメインの Web サービスを設定する場合、Web ドメイン名はサービス・ルートの設定に依存します。マルチ・ドメインの例として、複数の仮想ホストと結合されているマルチホーム構成（マシン上の複数の IP アドレス）を [図 1-3](#) に示します。

たとえば、IP アドレス、10.1.1.20 で定義されているドメイン内の、仮想ホスト、cavist.com は、別の IP アドレスと仮想ホストの組み合わせ（jones.com をホストする 10.1.1.30 など）として、同一の構造ネーミング・パターンを持つことができます。この場合、同じサービス・ルートを共有します。名前は、下位のディレクトリがドメイン名で一意に定義されるように、1 つのパスで定義します。

### 例 1-2 マルチ・ドメイン：URL 内のポート関係とサービス・ルート

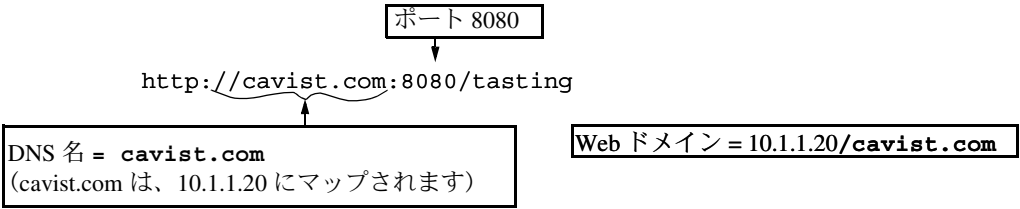
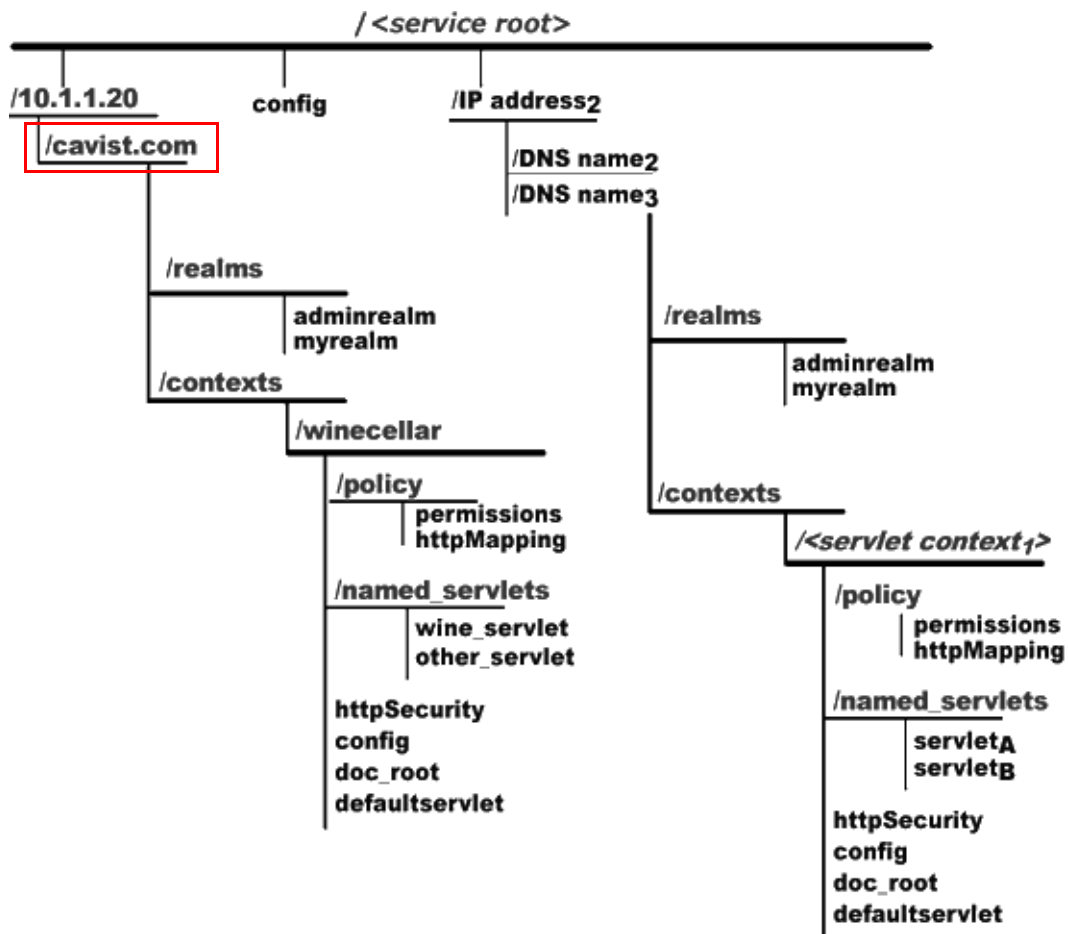


図 1-3 仮想ホストを持つマルチ・ドメイン、マルチホームの例



## サーブレット・コンテキスト

サーブレット・コンテキストは、OSE へロードされるアプリケーションとして考えます。サーブレット・コンテキストは、同じ仮想パスの下でアクセス可能なサーブレット、構成パラメータ、JSP およびファイルシステム上の静的コンテンツに対するポインタのセットです。サーブレット・コンテキストは、通常 URL のパスの最初のセグメントとして識別できます。

サーブレット・コンテキストには、Web サーバーがそのコンテンツを使用する場合の動作（セキュリティ、タイムアウト、MIME タイプ、サーブレットへの仮想パスの拡張のマッピング、ステートフル性）の設定が含まれます。OSE は、その親から構成プロパティを継承できるネストしたサーブレット・コンテキストをサポートします。

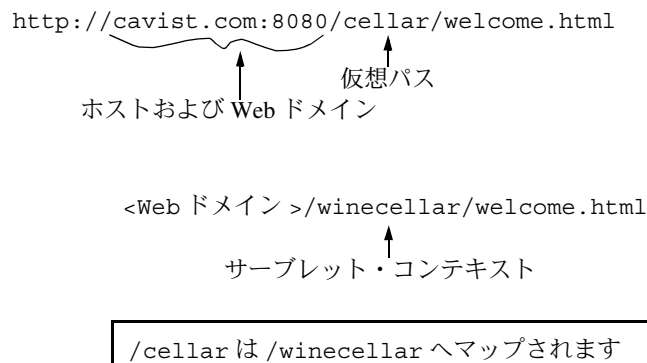
実際には、次のようになります。

- サーブレット・コンテキストの親ディレクトリ、`/contexts` は、そのドメイン・ルート・ディレクトリの名前空間でのサブディレクトリです。
- `winecellar` サーブレット・コンテキストは、`<domain root>/contexts/winecellar` ディレクトリに格納されます。
- サーブレット・コンテキスト・ディレクトリには、その構成パラメータ、静的ドキュメント・ルートへのポインタ、サーブレットを格納するサブディレクトリ、`named_servlets` が格納されます。

サーブレット・コンテキストの設定情報は、`servlet context` ディレクトリ内の名前空間オブジェクト、`config` です。図 1-2「[シングル・ドメインの名前空間モデルの構造](#)」を見ると、`<servlet context1>` ディレクトリの下に `config` オブジェクトがあるのがわかります。

`config` オブジェクトに、`/winecellar` へのマップとしてエントリ `/cellar` が指定されている場合、URL、`http://cavist.com:8080/cellar/welcome.html` では、Web ドメイン `cavist.com` の、サーブレット・コンテキスト `winecellar` にアクセスします。

### 例 1-3 仮想パスのサーブレット・コンテキストへのマッピング



## サーブレット

サーブレットは Java のクラスです。サーブレットのクラスおよび関連するサポートのクラスをデータベースへロードするには、loadjava コマンドを使用します。サーブレットをサーブレット・コンテキストに公開するには、セッション・シェルを使用します。

サーブレットを公開すると、次のことが実行されます。

- サーブレット・コンテキストのディレクトリの named\_servlets サブディレクトリに、指定した名前空間オブジェクトが作成されます。
- 仮想パスがサーブレットと対応付けられ、URI との一致に使用するサーブレット・コンテキストの config オブジェクトに格納されます。

### 例 1-4 HTTP 仮想パスに対応付けられたサーブレット

http://cavist.com:8080/cellar/winefinder でアクセス可能なサーブレットは、サービス・ルート /contexts/winecellar/config 内の仮想パス・マッピング・エントリ /winefinder=winefinderservlet を使用して、サービス・ルート /contexts/winecellar/named\_servlets/winefinderservlet という名前の名前空間オブジェクトとして公開できます。

## Java Naming and Directory Interface (JNDI)

前述のように、サーブレットは、名前空間で公開されてディレクトリ構造がモデルとなります。この名前空間は、Sun Java Naming and Directory Interface (JNDI) です。OSE の JNDI 実装では、SQL 表を使用して、JNDI でアクセス可能な名前空間のコンテンツを格納します。

JNDI にアクセスするには、セッション・シェルのコマンドライン・ツールを使用します。名前空間を検索および操作する場合は、これらのコマンドを使用します。

セッション・シェルを使用すると、ファイル・システムでディレクトリを変更してコンテンツをリストするのと同じように、名前空間を検索できます（ツールの定義および例については、[第3章「JNDI およびセッション・シェル」](#)を参照）。

コンテンツは、セキュリティ、マッピング、サーブレットおよびデフォルトのオブジェクトの階層的に格納された関係に編成されます。サーブレット・コンテキストは、Web サーバーで配置されたアプリケーションに対応します。サーブレット・コンテキストは、URL のアドレス部分にマップされて、Web ドメインでグループ化されます。

### 例 1-5 コンテンツへのクライアント・アクセスを示す URL

```
http://<Web domain:port>/<servlet-context>/<path>
```

この例の URL は、次のように実際の URL および名前に変換できます。

```
http://cavist.com:8080/winecellar/welcome.html
```

Web ドメインおよびサーブレット・コンテキストには、完全な管理権を持つ所有者（Oracle スキーマ）がいます。Web ドメインの所有者は、Web ドメインを管理し、サーブレット・コンテキストを作成し、その他のスキーマへアクセス権を付与できます。サーブレット・コンテキストの所有者は、サーブレット・コンテキスト内でコンテンツを公開し、その構成パラメータを変更できます。

## アクセス権とスキーマ

所有権とアクセス権は、セッション・シェルを使用して読取り、書込みおよび実行権限を定義するという点で、UNIX 環境の構造と似ています。Web ドメイン・スキーマは、JNDI 名前空間で唯一有効なユーザーです。ユーザーは、ドメイン内でのアクセス権および所有権を管理者から付与されます。

## 制御階層の概要

OSE の制御階層は、次のように定義されます。

Web サービス > Web ドメイン > サブレット・コンテキスト > サブレット

## セキュリティ

OSE は、「Servlet 2.2 仕様」で要求される認証およびアクセス制御をサポートします。

セキュリティの定義では、有効なユーザーを定義する必要があります。次のことを実行できます。

- データベース・ユーザーをユーザー数として使用できます。
- 独自のレルムのプリンシパルを定義できます。

プリンシパルの宣言は、Web サービスの realms ディレクトリに格納されます。次に、レルムのプロパティを示します。

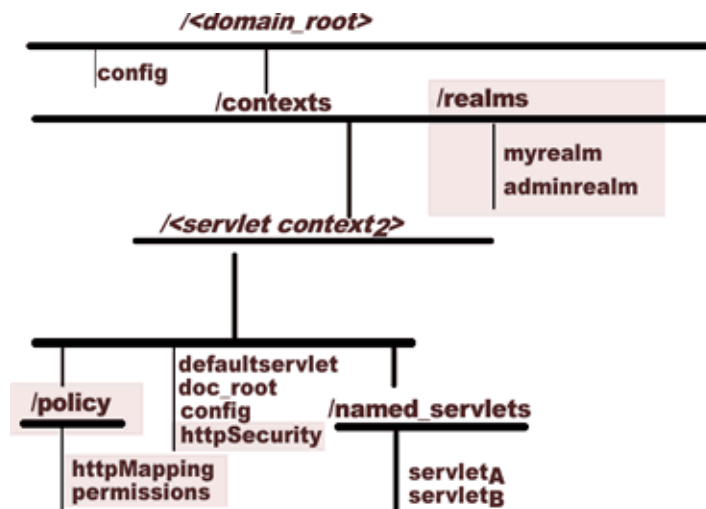
- レルムの定義は、サービスの適用範囲内です。
- レルムは、ユーザーとグループで構成されます。
- サービス内のすべてのサブレット・コンテキストでは、同じレルムの定義を使用できます。

policy ディレクトリは、アクセス権を設定して URL セキュリティ・マッピングを作成すると自動的に作成されます。

サブレット・コンテキストに対する policy ディレクトリの一般的な関係については、[図 1-4 「JNDI 名前空間のセキュリティ・コンポーネント」](#)を参照してください。

[図 1-4](#) に、JNDI 名前空間の残りのドメインに関する realms および policy ディレクトリ関係を示します。realms は、ドメインの最上位に位置するのに対し、policy は、サブレット・コンテキストのサブディレクトリになります。

図 1-4 JNDI 名前空間のセキュリティ・コンポーネント



バージョン OSE は、サーブレット 2.1 および JSP 1.0 をサポートします。



---

# OSE でのサーブレットの基本

この章では、サーブレット・コンテナを初めて使用するときに役立つ一連の操作を説明します。

この章では、次の項目について説明します。

- [JNDI](#)
- [セッション・シェル](#)
- [サーブレットの配置](#)
- [HTTP 要求の処理](#)

# JNDI

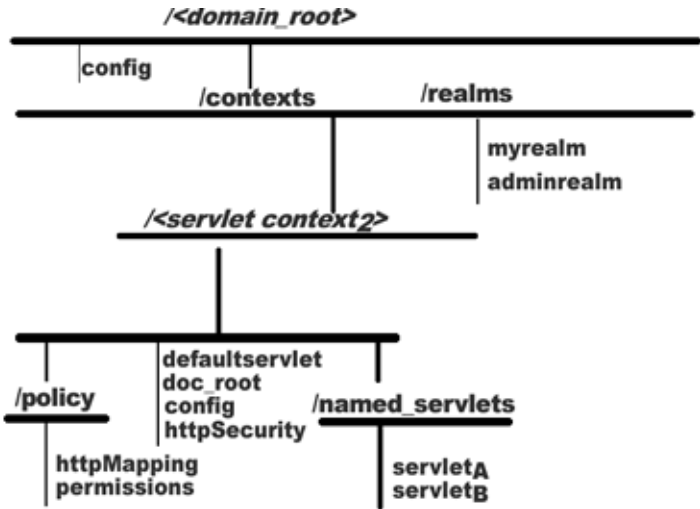
JNDI 名前空間は、サーバーの情報とコンテンツを格納する API です。この JNDI 名前空間は、Java プログラミング言語で作成されたアプリケーションに、ネーミングおよびディレクトリ機能を提供します。JNDI は、特定のネーミングまたはディレクトリ・サービス実装から独立しているため、Java アプリケーションで、共通の API を使用して様々なネーミングおよびディレクトリ・サービスへアクセスできます。この共通の API の背後には様々なネーミングおよびディレクトリ・サービス・プロバイダがシームレスに組み込まれているため、Java アプリケーションを、レガシー・アプリケーションやシステム（ファイル・システムなど）と共存させることができます。

Oracle8i 内の全 JNDI サーバーには、Oracle8i の内部および外部から JNDI API を使用してアクセスできます。Web サーバーを管理している場合に、名前空間のコンテンツとプロパティをインタラクティブに参照および変更するには、セッション・シェルを使用します。セッション・シェルのコマンドおよび構文の詳細は、『Oracle8i Java Tools リファレンス』を参照してください。

名前空間は階層で、2 つのタイプのエントリ、オブジェクトとディレクトリがあります。

- オブジェクトは、Java リファレンスを格納するために使用します。
- ディレクトリには、その他のディレクトリまたはオブジェクトが格納されます。JNDI エントリのパスは、通常の UNIX 表記法 /dir/dir/dir/leaf を使用して指定します。

図 2-1 dir/dir/dir/leaf を示す JNDI OSE 名前空間構造



## セッション・シェル

この項で説明するセッション・シェルのコマンドは、OSE の管理で最も頻繁に実行されるタスクで使用されます。次の項では、JNDI でオブジェクトの作成および処理に使用する簡単なコマンドについて説明します。

---

**注意：** 二重引用符を使用して複数行のエントリをコールすると、セッション・シェルでは、もう 1 つの二重引用符を入力するまで、次のエントリに右の山カッコ (>) を付けるよう求められます。

複数行エントリを示す別の方法は、継続標識としてバックスラッシュ (\) を使用します。

---

名前空間で利用できるセッション・シェルのコマンドがあります。次の特徴を持つ様々なスタイルのコマンドがあります。

- 名前空間を検索および操作するための、UNIX と同じようなコマンド
- JNDI オブジェクトのプロパティを取得および設定するコマンド
- 新しいオブジェクト（サープレット・コンテキストやその構成パラメータ、サープレットなど）を作成するための特殊なコマンド

コマンドの概要については、第 3 章「JNDI およびセッション・シェル」を参照してください。これらのコマンドの詳細は、『Oracle8i Java Tools リファレンス』を参照してください。

## オブジェクト・パラメータの取得と設定

JNDI オブジェクトのすべてのパラメータを表示するには `getproperties` コマンドを使用し、パラメータを変更するには、`setproperties` コマンドを使用します。また、`setgroup` コマンドを使用すると、グループ全体のパラメータとともに、パラメータの変更もできます。

`setproperties` および `getproperties` の構文は、次のようになります。

```
getproperties <object>
setproperties <object> <properties>
```

`getproperties` では、プロパティは、Java プロパティ・ファイルとして出力されます。`setproperties` では、プロパティを同じ構文を使用して渡す必要があります。複数のプロパティがある場合は、最初のプロパティの前に二重引用符を入力して、各エントリの後に改行を入れます。シェルは、最後のエントリが入力されるまで、続きのデータの入力待ちを示す山カッコ (>) のプロンプトを表示します。最後のプロパティを入力した後に二重引用符を使用し、最後のエントリであることを示します (>**foo=bar"** となります)。

---

---

**注意：** `setproperties` コマンドは、指定されたプロパティを設定する前に、オブジェクトのすべてのプロパティを消去します。

---

---

### 例 2-1 doc\_root パスの変更

```
$ getproperties doc_root
FSContextURL=/usr/local/oracle/jis/public_html
$ setproperties doc_root FSContextURL=/usr/local/winecellar
$ getproperties doc_root
FSContextURL=/usr/local/winecellar
```

---

---

**注意：** 新しいパスを入力する場合は、フルパスを指定してください。

---

---

`doc_root` の詳細は、7-5 ページの「[doc\\_root オブジェクト](#)」を参照してください。

## 仮想パス

サーブレット・コンテキストのサブディレクトリ、`named_servlets` にサーブレットを公開している場合は、仮想パスをサーブレットに対応付けることができます。

URL の定義を、

プロトコル + サーバー名 + URI と指定すると、構造は、`http://<servername>/<URI>` となります。

URL の一部として示される `URI` は、次で構成される要求 URI で定義されます。

**ContextPath + ServletPath + PathInfo**

サーブレット・コンテキストにマッピングされる仮想パスの最初の部分は、サーブレット・コンテキストの仮想パスと呼ばれ、特定のサーブレットにマッピングされる 2 つ目の部分は、サーブレットの仮想パスと呼ばれます。URI 内の残りの部分は、サーブレットに対する `PATH_INFO` として指定されます。

## プロパティ・グループ

JNDI エントリ内のパラメータのグループを定義します。

## サーブレットの配置

前の項では、JNDI オブジェクトの例について説明し、次の手順でテストを実行しました。

1. JNDI オブジェクトを作成します。
2. `getproperties` コマンドを使用して、`doc_root` オブジェクトのプロパティを表示します。

この項では、次のコマンドを使用してサーブレットを公開および実行する方法を説明します。

- `loadjava`
- `publishservlet`
- `setproperties`

## サーブレット・コンテキストで公開されるサーブレットのクラス

サーブレットのクラスを Oracle8i ヘロードするには、`loadjava` を使用し、サーブレット・コンテキストで公開するには、セッション・シェルのコマンドを使用します。サーブレットを公開すると、サーブレット・コンテキストの `named_servlets` サブディレクトリに JNDI オブジェクトが作成されます。

この JNDI オブジェクトには、サーブレットのクラス名とその初期パラメータがリストされます。サーブレットは、その JNDI のサーブレット・コンテキストの `config` オブジェクトの記述どおりに、HTTP 仮想パスと対応付けられます。

サーブレットクラスとすべてのサポートのクラスをデータベースヘロードするには、`loadjava` を使用します。（このツールの詳細は、『Oracle8i Java Tools リファレンス』を参照してください。）

### 公開されたサーブレット

公開されたサーブレットは、サーブレット・コンテキストの `named_servlets` ディレクトリ下にあるクラス `SYS:oracle.aurora.mts.ServletActivation` の JNDI オブジェクトになります。HTTP クライアントからアクセスするには、サーブレットを、そのサーブレット・コンテキストの `config` オブジェクトに、仮想パスまたはワイルドカード名と対応付ける必要があります。

```
publishservlet [-virtualpath <path>] [-stateless] [-reuse] [-properties props] \  
contextName <servletName> [className]
```

`-virtualpath` オプションおよび `path` (オプション) : 起動のためにこのサーブレットと対応付けられる仮想パス。

`-stateless` フラグ (オプション) : OSE に、サーブレットがステートレスであることを通知します。このオプションを設定すると、サーブレットは、HTTP セッションへのアクセス権がなくなります。

`contextName`: このサーブレットを含むコンテキストのディレクトリの名前です。

`servletName`: `named_servlets` ディレクトリ内のこのサーブレットに割り当てられる名前です。

`className`: `HttpServlet` インタフェースを実装するクラスの名前です。

このコマンドは、コンテキスト内の名前でサーブレットを公開し、仮想パスを指定したサーブレットと対応付けます。

### 例 2-2 `publishservlet` コマンドを使用したサーブレットの公開

The diagram illustrates the mapping of arguments in the `publishservlet` command to their respective parameters. Arrows point from labels to specific parts of the command:

- `servletName` points to `tastingServlet`.
- `className` points to `SCOTT:winemasters.tasting.Tasting`.
- `サーブレット・パス` (Servlet Path) points to `/tastings`.
- `contextname` points to `/webdomains/contexts/winecellar \`.

```
$ publishservlet -virtualpath /tastings /webdomains/contexts/winecellar \  
tastingServlet SCOTT:winemasters.tasting.Tasting
```

サーブレットが公開されたことを確認するには、次を入力します。

```
$ ls /webdomains/contexts/winecellar/named_servlets  
tastingServlet
```

仮想パス・マッピングを確認します。

```
$ getgroup /webdomains/contexts/winecellar/config context.servlets  
/errors/internal=internalError  
/tastings=tastingServlet
```

### 例 2-3 `setproperties` コマンドを使用した、公開されたサーブレットのプロパティの変更

プロパティを追加するにはセッション・シェルのコマンドの `setproperties` を使用し、サーブレットから追加した新しいプロパティにアクセスすることができます。

```
$ getproperties invoker  
servlet.class=SCOTT:winemasters.tasting.Tasting  
  
$ setproperties invoker "servlet.class=SCOTT:winemasters.tasting.Tasting  
>details=high  
>style=parker"  
  
$ getproperties invoker  
servlet.class=SCOTT:winemasters.tasting.Tasting  
details=high  
style=parker
```

## HTTP 要求の処理

この項では、適切なサブレットを検索するために OSE で使用するアルゴリズムを説明します。

サブレットは、すべての HTTP 要求を処理します。要求の配置は、クライアントが HTTP 要求をサーバーに送信すると開始されます。OSE では、この要求から仮想パスが取得されます。

サービスを起動すると、config オブジェクトでは仮想パスを使用して、サブレットがその要求にマップされます。

次の 3 つの項では、HTTP 要求の処理方法を説明します。

## Web ドメインの検索

要求を処理する Web ドメインは、OSE で構成される Web サービスに依存します。

### シングル・ドメインの Web サービス：

要求を処理する Web ドメインは、サービスと対応付けられている単一の Web ドメインです。

### マルチ・ドメインの Web サービス：

より複雑な Web サービスを構成している場合、使用されるドメインは、OSE にアクセスするためのサービスと URL に依存します。詳細は、[第 1 章「概要」](#)のマルチ・ドメインの説明を参照してください。

## サブレット・コンテキストの検索

OSE は、Web ドメインの config オブジェクトのコンテキスト・プロパティのグループで検索します。このプロパティのセットは、コンテキスト名に対する仮想パスのマッピングのリストです。

OSE では、コンテキスト・プロパティ内の各エントリに、要求の仮想パスを一致させます。一致した桁数が最も多いサブレット・コンテキストが要求を処理します。

### 例 2-4 サブレット・コンテキストを検索する HTTP 要求

次の URL の場合、

```
http://cavist.com:8080/cellar/bordeaux.wine
```

Web ドメイン config オブジェクトからの仮想パスが、サブレット・コンテキストにマップされる場合、

```
/cellar=/winecellar
```

サブレット・コンテキスト、`cavist/contexts/winecellar` が使用されます。

また一致するサブレット・コンテキストが見つからない場合は、要求は、次のコンテキストで処理されます。

`/cavist/contexts/default`

## サブレットへの仮想パスのマッピング

サブレットが URI へマップされると、サブレット・コンテキストへマップされた URI パスの部分が無視されます。残りの部分は、サブレット・コンテキストの `config` オブジェクトの `context.servlets` プロパティ内のエントリと一致します。一致した部分が最も長いサブレットが要求を処理します。

`context.servlets` プロパティのエントリには、「Servlet 2.2 仕様」に基づいて、パスまたはワイルドカード名を指定できます。部分パスは、ワイルドカード名より優先されます。完全一致した場合は、部分パスとワイルドカード名の両方より優先されます。

### 例 2-5 サブレットを検索する HTTP 要求

仮想パス `/cellar/bordeaux.wine`

は、コンテキスト `/cavist/contexts/winecellar` で処理されます。

サブレットの名前を検索するには、`bordeaux.wine` を、サブレット・コンテキストの `config` オブジェクトで定義された、仮想パス・マッピングに一致させます。

サブレット・コンテキスト内で、一致するサブレットが見つからない場合は、要求はデフォルトのサブレットで処理されます。

## デフォルトのサブレットの試用 - 静的コンテンツ

要求した URL に一致するものが見つからない場合は、OSE は、最初にコンテキスト・ディレクトリ、次に Web ドメイン・ディレクトリ、最後に Web サービスで `defaultServlet` という名前のサブレットを検索します。デフォルトのサブレットが見つかった場合は、このサブレットによって要求が処理されます。

### 例 2-6 `defaultServlet` で処理される要求

`winecellar/doc_root` オブジェクトは、ファイル・システムのディレクトリ、`/usr/local/coolplace/html` を示していると仮定します。

このため、`doc_root` オブジェクト・パラメータのコンテンツは、次のようになります。  
`FSContextURL=/usr/local/coolplace/html`

URL は、`http://cavist.com:8080/cellar/labels/lafitte.gif` です。



仮想パス `/cellar` は、サブリット・コンテキスト `/winecellar` へマップされます。サブリット・コンテキストに、`/labels/lafitte.gif` に一致するものがない場合（または、`/labels/*` などの部分的な一致の場合）は、デフォルトのサブリットが使用されます。

デフォルトのサブリットでは、次が実行されます。

1. 開始して `/labels/lafitte.gif` という `pathInfo` を受信します。
2. `doc_root` 下の `pathInfo` と一致させます。
3. `/usr/local/coolplace/html/labels/lafitte.gif` を検索します。

この一致は、完全に一致する必要があります。デフォルトのサブリットが見つからない場合は、OSE で、内部エラー・コードが生成されます。



---

## JNDI およびセッション・シェル

この章では、ツール定義と例を示します。

この章では、次の項目について説明します。

- [JNDI について](#)
- [JNDI アクセス権](#)
- [セッション・シェルの起動](#)
- [OSE でのセッション・シェルのコマンドの概要](#)

Oracle8i 内部の全 JNDI サーバーへは、JNDI セッション・シェルのコマンドを使用して、Oracle8i の内部および外部からアクセスできます。名前空間のコンテンツをインタラクティブに操作するには、セッション・シェルを使用します。

## JNDI について

JNDI には、Oracle8i で実行される様々な Java サーバー（OSE、CORBA および EJB）の情報とコンテンツが格納されます。

名前空間は階層構造で、2つのタイプのエントリ、ディレクトリおよびオブジェクトがあります。

**ディレクトリ**には、他のディレクトリおよびオブジェクトが格納されます。JNDI エントリへのパスは、通常の UNIX 表記法 `/dir/dir/dir/leaf` を使用して指定します。

**オブジェクト**は、Java リファレンスの格納に使用します。リファレンスは、`javax.naming.Reference` クラスのインスタンスです。JNDI サーバーは、リファレンス・コンポーネント（クラス名、クラス・ファクトリ名およびパラメータ）を取得し、この情報を使用して Java オブジェクトをインスタンス化します。セッション・シェルには、JNDI 名前空間内に新しいリファレンスを格納し、そのパラメータを操作するコマンドが用意されています。また、名前空間を検索し、エントリを削除し、それらのアクセス権を変更するコマンドのセットも用意されています。

JNDI 構造を示す、[第1章「概要」](#)の図を参照してください。

## JNDI アクセス権

JNDI サーバー内のすべてのコンテンツは保護されています。JNDI サーバーにユーザーとしてアクセスする場合は、コンテンツを参照および変更するための適切なアクセス権が必要です。UNIX ファイル・システムと同じように、JNDI は、3つのタイプのアクセス権、読取り権限、書き込み権限および実行権限をサポートします。アクセス権は、データベース・ユーザー（スキーマ）またはデータベース・ロールに付与できます。アクセス権の付与は、シェルの1つを使用するか、プログラムで実行できます。JNDI アクセス権（読取り、書き込み、実行権）の設定は、UNIX のファイルおよびディレクトリ・アクセス権の設定と似ています。

JNDI サーバーでは、所有者概念も使用されます。つまり、各エントリは、データベース・スキーマによって所有されます。これは、UNIX ファイル・システムと似ています。

## セッション・シェルの起動

すべてのコマンドは、リモート・アクセスによってサーバー上で実行されるため、サーバーとの接続に使用する通信トランスポートを指定する必要があります。通信オプションとして、JDBC、HTTP および IIOP（`SESS_IIOP`）が使用できます。

クライアントまたはサーバー上でセッション・シェルのコマンドにアクセスするには、次を入力します。

```
sess_sh -s transportURL -u[ser] username[/passwd] [-p[assword] passwd] [otherargs... ]
```

-s **service** transportURL: サーバーとの通信に使用するトランスポートを、URL 記述子の形式で指定します。

次の URL がサポートされます。

jdbc:oracle: <b>type:spec</b>	JDBC を使用したデータベースへの接続方法を指定する JDBC URL
http:// <b>host:port</b>	データベースに事前にインストールされている管理 Web サーバーへの接続に使用するホストとポートを示す HTTP URL
sess_iiop:// <b>host:port[:sid]</b>	サーバー上の GIOP リスナーのホスト、ポートおよび SID を示す SESS_IIOP URL

-u **username**: データベース・セッションへのログイン名

-p **passwd**: ログイン時に使用するパスワード

-command "**cmd...**": サーバーで実行されるコマンド

シェル・ツールと環境に関する詳細は、『Oracle8i Java Tools リファレンス』を参照してください。

## ディレクトリのナビゲーションと管理

セッション・シェルは、cd、pwd、ls、mkdir および rm などの、ディレクトリを検索および管理するための標準の UNIX コマンドをサポートします。

## アクセス権および所有権

各 JNDI エントリには、アクセス権が設定されます。JNDI サーバーは、3 つのタイプのアクセス権（読取り、書込みおよび実行）をサポートします。アクセス権をデータベース・ユーザーやグループへ付与またはそれらから削除するには、シェルの chmod コマンドを使用します。

## OSE でのセッション・シェルのコマンドの概要

セッション・シェルには、Web サーバーを管理し、サーブレットを公開するための特殊なコマンドのセットが用意されています。構文の要件については、『Oracle8i Java Tools リファレンス』を参照してください。OSE の JNDI 名前空間を操作するための各コマンドの使用方法について、この項で簡単に説明します。

## サービスの構成

次のコマンドのセットで、新しいサービスを作成します。

**createservice** — サービスのエンド・ポイント（TCP ポート）を定義せずに、ServicePresentation コードで必要なパラメータを操作します。

**addendpoint** — 動的な登録表内にエンド・ポイントを格納するリスナーで、新しいエンド・ポイントを追加し、エンド・ポイントの動的登録を実行します。

**rmendpoint** — サービスから特定のエンド・ポイントを削除し、リスナーから動的に登録されたポートを削除します。

**destroyservice** — サービスおよび動的登録を含むすべてのそのエンド・ポイントを削除します。-all フラグは、JNDI ツリー全体を（サービス・ルートのレベルから）削除します。

**createwebservice** — ServicePresentation コードで必要なパラメータを操作して、Web 固有の構成を初期化します。

## Web ドメインの構成

次のコマンドのセットで、サーブレット・コンテキストの位置を設定します。すべての JNDI エントリで、各位置に管理者または所有者が必要です。

**createweбdomain** — 現行のスキーマで管理される Web ドメインを作成します。この場合、サーブレットはそのスキーマとして実行されます。また、このコマンドは、初期サーブレット・コンテキスト、デフォルトおよび doc\_root を定義します。

**destroywebdomain** — Web ドメインおよび対応付けられているすべてのサーブレット・コンテキストを削除します。

## セキュリティ管理

次のコマンドのセットで、セキュリティ・クラスにより使用されるレルムおよび認証方式を指定するドメインとサーブレットに、セキュリティを設定します。

**realm** — すべての領域コマンドをリストします。

**realm list -w <Web サービス・ルート>** — サービスで宣言されたすべての領域をリストします。

**realm map -s <servletContextPath> [-(add|remove) <path> -scheme <auth>:<realm>]** — サーブレット・コンテキスト内の保護付きパスを定義およびリストします。また、保護しないように宣言します。

**realm echo [0|1]** — エコーのオンとオフを切り換えます。

**realm publish -w <Web サービス・ルート> [-(add|remove) <realmName> [-type (RDBMS | DBUSER | JNDI)]]** レルムを作成、公開および削除します。

`realm user -d <domainContextPath> -realm <realmName> [-(add | remove) <userName> [-p <user password>]]` ユーザーを作成、削除します。

`realm group -d <domainContextPath> -realm <realmName> [-(add | remove) <groupName> [-p <group password>]]` グループを作成、削除します。

`realm parent -d <domainContextPath> -realm <realmName> [-group <groupName> [-(add | remove) <principalName>]] [-query <principalName>]` グループでプリンシパルを追加、リストまたは削除します。

`realm perm -d <domainContextPath> -realm <realmName> -s <servletContextPath> -name <principalName> [-path <path> (+|-) <permList>]` 有効な HTTP メソッドに対して、ユーザーの指定されたパスでのアクセス付与またはアクセス制限を宣言、消去およびリストします（第 8 章「[セキュリティ HTTP の管理](#)」の[アクセス権の宣言](#)を参照）。

## サーブレット・コンテキストの管理

次のコマンドのセットで、コンテキストを操作します。

`createcontext` - ドメインの対応する仮想パスでコンテキストを作成します。

`destroycontext` - サーブレット・コンテキスト情報およびすべてのサーブレットをそのドメインから削除します。

`adderrorpage` - エラー・コードに対して、このコンテキストのエラーをレポートする URL を定義します。

`rmerrorpage` - 該当するエラー・コードに対応付けられているエラー・ページを削除します。

## サーブレットの管理

このコマンドのセットで、サーブレットの公開と非公開を処理します。

`publishservlet` - コンテキスト内の名前でサーブレットを公開します。また、このコマンドで、仮想パスを指定したサーブレットに対応付けることもできます。ステートレスとして宣言されたサーブレット・コンテキストで公開されたサーブレットは、HTTP セッション・オブジェクトにアクセスできません。

`unpublishservlet` - マッピング・テーブル内のサーブレットの既存の仮想パスのみでなく、サーブレット・コンテキストからサーブレットを削除します。

## エクスポート・コマンド

このコマンドのセットで、Web ドメインの構造を抽出し、該当する構成ファイルを生成できます。mod\_ose またはその他のプロキシでは、このコマンドで生成したファイルを使用します。

exportwebdomain – エクスポート・ユーティリティは、構成ファイルを生成する場合に次の 1 つまたは 2 つの段階で使用できます。

1. XML 形式で、Web ドメインまたはドメイン内のコンテキストの構造を生成する場合。
2. (オプション) XML 構造への変換を適用して、特定の Web サーバー (apache、iis など) の構成ファイルを生成する場合。

Web ドメインのエクスポートの詳細は、[第 5 章「Oracle OSE の Apache モジュール」](#)を参照してください。



---

## アーキテクチャ

この章では、OSE で、シングル・ドメインおよびマルチ・ドメインの Web サービスに対する要求を処理する方法を説明します。OSE で使用される論理に従って、適切なサーブレットを検索できます。

この章では、次の項目について説明します。

- [HTTP 要求](#)
- [HTTP 要求のライフ・サイクル](#)
- [データベース・セッション](#)

## HTTP 要求

最初の HTTP 要求がブラウザから受信されると、クライアント用にセッションが生成されます。次に続くすべての要求は、作成されたセッションに送られます。セッション追跡のメカニズムは、cookie または URL の再書込みです。

## URL

URL は、次に示すように、プロトコル、サーバーおよび Uniform Resource Identifier (URI) で構成されます。

```
<protocol>://<server>/<URI>
```

URI は、クライアントによって URL 要求内に指定されます。URI には、宣言された仮想パスとの一致のために OSE で使用される情報が含まれます。config オブジェクトでのマッピングは、特定の要求に使用するサーブレットの判別に使用されます。サーブレットの位置を特定できない場合は、デフォルトのサーブレットが使用されます。

## HTTP 要求のライフ・サイクル

HTTP 要求のライフ・サイクルの例として、[図 1-3](#) に示すマルチ・ドメインの、URL `http://cavist.com:8080/cellar/bordeaux.wine` を使用します。

次のステップは、このサイクルの進行過程を示しています。

1. ブラウザで、HTTP 要求が送信されます。
2. 要求は、ポート 8080 で受信されます。
3. データベース・セッションが生成されて、OSE に要求が割り当てられます。
4. OSE では、サービスを使用して 8080 ポートでリスニングします。(OSE の設定方法については、[第 6 章「OSE サーバー構成」](#)を参照してください。)
5. 要求は、Web サービスで確認されて、IP アドレスを参照し、IP ドメインにマッピングされます。
6. DNS 名 (cavist.com) は、JNDI 名前空間のディレクトリ (cavist.com) にマップされます。
7. `10.1.1.20/cavist.com/config` オブジェクトは、URI をサーブレット・コンテキストに一致させるために使用されます (URI 仮想パス `cellar` は、JNDI 名前空間のエントリ、`winecellar` にマップされます)。4-3 ページの「[サーブレット・コンテキストの検索](#)」を参照してください。
8. サーブレット・コンテキストの config オブジェクトは、URI の残りの部分 (`bordeaux.wine`) を適切なサーブレット `cavist.com/contexts/winecellar/named_servlets/wine_servlet` に一致させるために使用されます。

9. サブレット `wine_servlet` は、要求 `http://cavist.com:8080/cellar/bordeaux.wine` を処理します。
10. セッションが終了します。(4-7 ページの「[データベース・セッションのタイムアウト](#)」を参照してください。)

## エンド・ポイント

エンド・ポイントは、情報を受信するポートです。Net8 は、ロード・バランシングおよび接続性を制御します。送信された要求を受信するためには、リスナーに割り当てるホストの IP アドレスと有効なポート番号を使用して、リスナーをタイプごとにエンド・ポイントに設定する必要があります。

## Web ドメインの検索

要求を直接処理する Web ドメインは、OSE に格納されている Web サービスの構成によって異なります。

Web サービスでは、次のように要求を受けます。

- シングル・ドメインの場合は、1 つの Web ドメインのみがサービスに対応付けられます。デフォルトで、このドメインは、サービス・ルートに格納されています。
- マルチ・ドメインの場合は、Web ドメインは、URL の DNS 名と対応付けられている IP アドレスに結合されます。

---

**ヒント：** マルチ・ドメインのセットアップを参照する場合は、マルチ・ドメインのディレクトリ・ツリー図の前の [例 1-2](#) を参照してください。

---

## サブレット・コンテキストの検索

OSE は、ドメインの `config` オブジェクトで、サブレット・コンテキスト名に仮想パスをマップするプロパティを検索します。次に、OSE は、`config` オブジェクトのマッピングのプロパティの各エントリと要求の URI を一致させます。一致した部分が最も長いサブレット・コンテキストが要求を処理します。要求されたサブレット・コンテキストが見つからない場合は、デフォルトのサブレット・コンテキストが使用されます。デフォルトのサブレット・コンテキストは、`/<domain root>/contexts/default` に格納されています。

### 例 4-1 HTTP 要求

URI が `/cellar/bordeaux.wine` で、

`config` オブジェクトからのコンテキスト・グループのコンテンツが次のような場合は、

```
/cellar=/winecellar
/test=/test
```

`/cellar` という部分的な URI の一致のため、コンテキスト `/contexts/winecellar` で処理されます。

## サーブレットの検索

サービスのコンテキストが識別されると、URI の不一致部分は、サービス・コンテキストの `config` オブジェクトの仮想パスに対して一致します。OSE では、サーブレット・コンテキストと一致する仮想パスの部分は無視されます。残りの部分は、サーブレット・コンテキストの `context.servlets` プロパティ内のエントリに一致します。一致の部分が最も適切なサーブレットが要求を処理します。

---

---

**注意：** `context.servlets` プロパティ内のエントリには、パスまたはワイルドカード名を指定できます（「Servlet 2.2 仕様」を参照）。部分パスは、ワイルドカード名より優先されます。完全に一致する場合は、部分パスおよびワイルドカード名の両方より優先されます。

---

---

### 例 4-2 サーブレットにより処理される要求

前の例 4-1 のシナリオの続きとして、指定されたサーブレット、`tastingServlet` は、`winecellar/named_servlets` で公開されます。

この場合、`bordeaux.wine` は、サーブレット・コンテキストにより、`tastingServlet` という名前のサーブレットのエントリにマップされます。

## デフォルトのサーブレットの検索

要求された URL に一致するものが見つからない場合、OSE では、`defaultServlet` という名前のサーブレットがサーブレット・コンテキストのディレクトリで検索されます。デフォルトのサーブレットが見つかると、要求が処理されます。

Web ドメインが作成されると、OSE は、ドメインのサーブレット・コンテキストのディレクトリ内にデフォルトのサーブレットをインストールします。Web サービスには常にデフォルトのサーブレットがありますが、新しいデフォルトのサーブレットとして機能する、`/` の仮想マッピングを指定できます。

URI の未使用の部分が、サーブレット・コンテキストで公開された仮想パスとまったく一致しないあるいは部分的にしか一致しない場合、この同じ URI 部分はデフォルトのサーブレットで使用されます。デフォルトのサーブレットは、`doc_root` で完全一致を検索します。一

致するものが存在する場合は、このページが処理されます。完全に一致するものが見つからない場合は、OSE によりエラー・コードが生成されます。

サービス・コンテキストの `config` オブジェクトでのマッピングにより、エラー・コードがエラー・ページへマップされます。たとえば、エラー・コード 404 は、`/system/errors/404.htm` という名前の URI へマップされます。これで、`doc_root` 内の名前と位置から該当のエラー・ファイルがデフォルトのサーブレットにより処理されます。

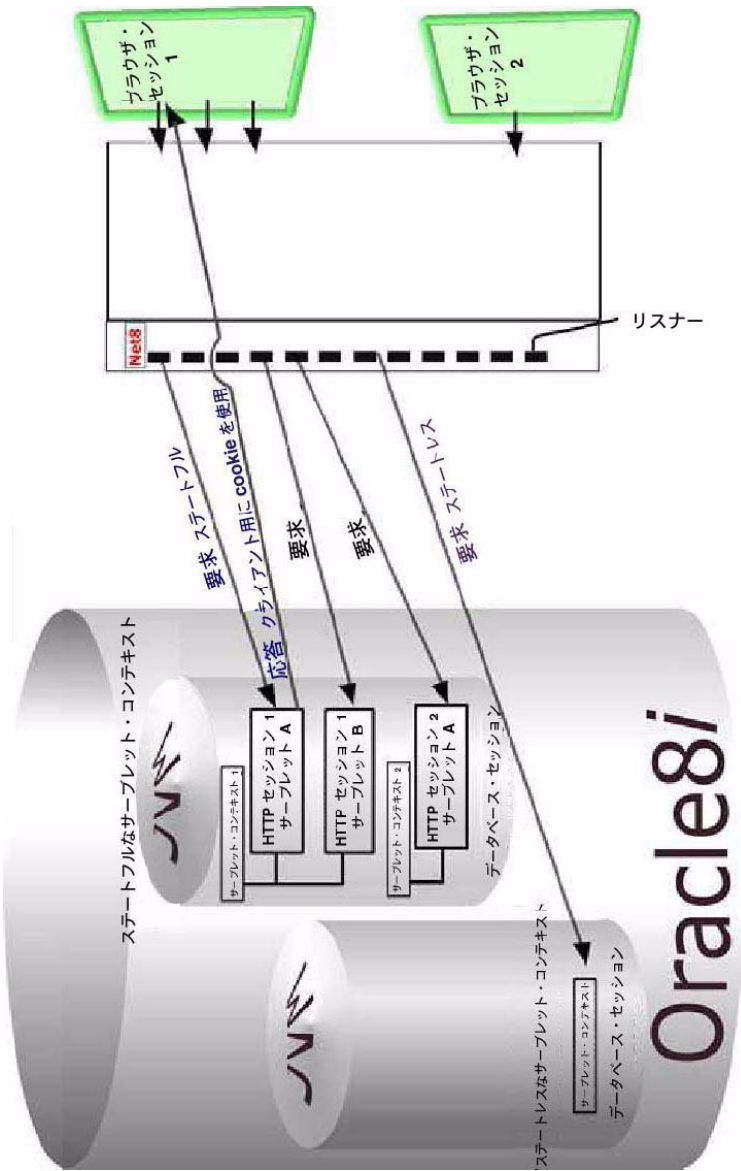
#### 例 4-3 デフォルトのサーブレットにより処理される要求

前の例 4-2 の続きとして、サーブレットに一致するものが見つからなかった場合、要求は、サービス・ルート `/contexts/winecellar/defaultServlet` で処理されます。デフォルトのサーブレットが使用され、その結果はクライアントに戻されます。

## データベース・セッション

データベース・セッションの特徴は、OSE とその他のサーブレット・エンジンの相違点の 1 つです。

図 4-1 1つのブラウザ・セッションに対応する単一の仮想的なJVM、データベース・セッションの複数のHTTPセッション



## データベース・セッションのユーザー認証

デフォルトでは、データベース・セッションは Web ドメインの所有者として認証されます。JDBC または SQLJ を介したすべての SQL アクセスは、Web ドメイン所有者のスキーマで実行されます。

SCOTT が所有するドメインでサブリットの設定を配置する場合は、別のスキーマ名が表名に指定されないかぎり、サブリットは SCOTT のスキーマの表にアクセスします。たとえば、SYS:HTTP\$LOG とします。この例では、スキーマ SCOTT には、他のスキーマの表にアクセスするためのアクセス権が必要です。

この動作を変更するには、定義者権限コードを使用します。定義者権限が付与されるクラスで実行されるコードは、SYS として実行されます。

## データベース・セッション

HTTP クライアントでは、単一のデータベース・セッションにいくつかの HTTP セッション・オブジェクトを持つことができます。HTTP セッションは、サブリットごとに設定されます。クライアントが、2つの異なるサブリット・コンテキストからサブリットをアクティブにし、各サブリットが `getSession(true)` をコールすると、図 4-1 に示すように、2つの異なる HTTP セッションのインスタンスがデータベース・セッション内に存在します。

Oracle8i 内の Web サーバー・セッションには、次の 2 つのタイムアウトが対応付けられています。

- データベース・セッション自体のタイムアウト
- データベース・セッションで作成された各 HTTP セッションのタイムアウト

### データベース・セッションの作成、終了、タイムアウト

HTTP クライアントを、OSE を実行する Oracle8i のインスタンスに最初に接続するときに、データベース内にセッションが生成されます。このセッションは、独自の仮想的な JVM を実行する通常のデータベース・セッションです。

### データベース・セッションのタイムアウト

データベース・セッションのタイムアウトにより、格納されているすべての HTTP セッションが終了します。HTTP セッションが終了すると、その Java のステータス情報はすべて破棄され、コミットされていない SQL 状態はすべてロールバックされます。

これは、サブリットを実行する仮想的な JVM の終了とみなされます。

データベース・セッションのタイムアウトは、Web サービス・パラメータ、`service.globalTimeout` で設定します。Web サービス・オブジェクトは接続を制御します。このパラメータは、サービスを作成するときに設定するか、後から設定できます（例 4-4 を参照）。

### 例 4-4 サービスの作成とグローバル・タイムアウトの設定

```
createservice [ -http | -iiop ] [-service className ] [-properties propGroups ]  
-root location [ -globalTimeout secs ] service
```

タイムアウトでは、HTTP 要求が戻された後にカウントが開始されます。タイムアウトが時間切れになる前に別の要求が届くと、最初の要求が終了した後に、同じ値を使用して再度カウントが開始されます。

データベース・セッションがタイムアウトになると、ブラウザは、セッション cookie を維持している場合でも、そのデータベース・セッションに接続できなくなります。かわりに、ブラウザは新しいセッションにルーティングされます。

### データベース・セッションの自動終了

HTTP セッションのオブジェクトがなくなると、データベース・セッションは自動的に終了します。HTTP セッションのタイムアウトにより、データベース・セッションが終了する場合もあります。

## HTTP セッションの作成、終了およびタイムアウト

HTTP セッションは、ステートフルなサーブレット・コンテキスト内のサーブレットがメソッド `getSession(true)` をコールすると作成されます。

タイムアウトは、現在の値が接続に関する他のアクティビティに一致しない場合に発生します。このタイムアウト値は、秒単位で定義され、`config` オブジェクトに格納されます。データベース・セッションを終了する場合は、HTTP セッションも終了する必要があります。

複数の HTTP セッション・オブジェクトを、データベース・セッションごとに置くことができます。[図 4-1](#) に示すように、各クライアントには 1 つのデータベース・セッションのみが存在します。

ステートフルなセッションは、クライアントとサーブレット間の対話状況に役立ちます。サーバーでクライアント情報をいつでも利用できるように、cookie はクライアントにメソッドとして送信されます。ステートフルなセッションは、セッション中にクライアントが情報の交換を段階的に実行している場合、次のステップの情報が、前のステップに送信された情報で予測されるため便利です。クライアントが cookie をサポートしない場合、OSE では URL の再書込みが使用されます。



---

## Oracle OSE の Apache モジュール

この章では、Apache (HTTP Web サーバー) を使用して、OSE サブレット・コンテナを構成および使用する方法を説明します。この配置によって、`mod_ose` から `Oracle8i` で実行される OSE への、動的コンテンツの移譲方法を確認できます。

この章では、次の項目について説明します。

- [Apache での OSE の要件](#)
- [OSE の Apache での `mod\_ose` の概要](#)
- [OSE の Apache での `mod\_ose` の構成](#)
- [OSE アプリケーションでの Apache の構成](#)
- [`mod\_ose` を使用したサイトのトポロジ](#)
- [事前インストールされている Apache システムのための `mod\_ose` のインストール](#)
- [URL の転送](#)

## Apache での OSE の要件

`mod_ose` には、次の Apache 機能が必要です。

- （Apache で稼動される）Oracle HTTP server
- `mod_so` を Apache（または、Dynamic Shared Object（DSO）に対するその他のサポート）にインストールする必要があります。
- `mod_mime` を Apache にインストールする必要があります。
- Apache を実行するマシンは、Oracle クライアント側の構成として構成する必要があります。
- Oracle クライアント側のライブラリは、`LD_LIBRARY_PATH` または等価で使用可能である必要があります。

## OSE の Apache での `mod_ose` の概要

`mod_ose` は、Oracle HTTP server（Apache）のモジュールで、Apache から OSE へのパイプとして機能します。`mod_ose` と連携すると、OSE は、Apache のサーブレット・エンジンとなります。この場合、静的ページのみが Apache で処理され、動的ページは OSE で処理されます。

マルチノードの連携構成を作成できます。これらの構成を使用すると、単純なスタンドアロンの OSE または Apache コンポーネントよりも、よりスケーラブルなサービスを実行できます。

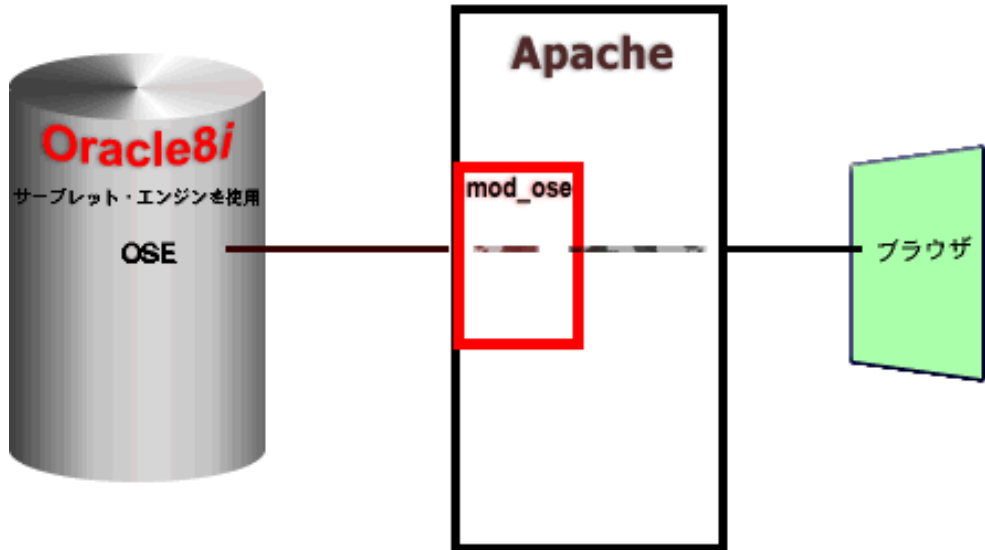
特に、マルチノードの配列では、次の特徴があります。

- 単一障害点がない
- フェイルオーバー構成
- ロード・バランシング機能

`mod_ose` は、Net8 と連携してネットワーク内で機能します。ロード・バランシング、Oracle リスナーの使用方法、フェイルオーバー構成の詳細は、『Oracle8i Net8 開発者ガイド』で説明しています。

ネットワークに必要なサーバーの数とタイプを決定する前に、2つのサーバー（Web およびサーブレット）の基本構成を理解しておく必要があります。[図 5-1](#) に、基本的なネットワーク配列の単純な図を、ブラウザ、Apache、Apache 内の `mod_ose`、Net8（`mod_ose` とデータベースの間の接続として表される）、データベースおよび OSE の順序で示します。

図 5-1 mod\_ose を使用した Apache Web サーバーと Oracle のサーブレット・エンジン (OSE) の単純なトポロジ



## OSE の Apache での mod\_ose の構成

この項では、Apache を HTTP サーバー、OSE をサーブレット・コンテナとして使用するために必要な mod\_ose 構成の手順を説明します。

次に示す手順では、mod\_ose で使用するデータベース接続記述子を定義し、要求が適切なデータベースへ送信されるように Apache 構成を設定します。

1. tnsnames.ora の Oracle Net8 接続記述子を、OSE サービスへのポインタとして指定します (例 5-1 を参照)。
2. セッション・シェルのコマンド、exportwebdomain を使用して、Web ドメイン構成ファイルを生成します (例 5-2 を参照)。
3. 生成した構成ファイルを Apache 構成ファイル、httpd.conf に追加します (例 5-3 を参照)。
4. Apache がすでにインストールされている場合は、使用する接続を定義します (例 5-4 を参照)。

tnsnames.ora の接続記述子と構文

mod\_ose では、他の Oracle クライアント（OCI クライアントなど）と同じ構成メカニズムを使用して、接続記述子を検索します。

sqlnet.ora 構成によって、接続は次のいずれかで定義されます。

- 環境変数 TNS\_ADMIN により示されるディレクトリにある tnsnames.ora ファイル  
または
- 接続アドレスを検索するための別の Oracle ネーム・サービス（LDAP サービスなど）

Apache で使用される接続用のエントリは、tnsnames.ora 内で、次の行で定義する必要があります。

```
<your_defined_http_connection_name> = (DESCRIPTION=
    listenerAddress
    (CONNECT_DATA=
        serviceSpec
        presentationSpec
    )
)
```

構文は次のように定義されます。

```
listenerAddress := (PORT=...) (ADDRESS= (HOST=...) (PROTOCOL=...) ...)
serviceSpec := (SERVICE_NAME=...) [ (INSTANCE_NAME=...) ]
presentationSpec := (PRESENTATION=http://<JndiServiceName>)
```

表 5-1 tnsnames.ora 接続情報キーワード

キーワード	定義
listenerAddress	バックエンドに複数の Oracle リスナーがある（複数ノード）場合の ADDRESS_LIST（ホスト、ポート、プロトコル）。接続集中化のための CMAN の使用のみでなく、ロード・バランシングとフェイルオーバー構成も考慮されます。  パラメータの設定方法の詳細は、『Oracle8i Net8 開発者ガイド』を参照してください。
serviceSpec	データベース・インスタンスのグループ: SERVICE_NAME を指定します。 相互に交換して使用できるインスタンスのグループを定義します。 複数のデータベース・インスタンスがある場合は、mod_ose で、各インスタンス間の接続のロード・バランスが実行されます。  mod_ose では、クライアントからのステートフルな要求は、必ず同じデータベース・インスタンスに送信され、同じデータベース・セッションと対応付けられます。

表 5-1 tnsnames.ora 接続情報キーワード（続き）

キーワード	定義
	単一のデータベース・インスタンス：INSTANCE_NAME を指定します。 サーバー・グループ内の特定のインスタンスを使用することを示します。
presentationSpec	この接続に使用する HTTP サーバーを示します。  <JndiServiceName> は、HTTP を認識し、HTTP と対応付けられた Net8 のエンド・ポイントがある、JNDI 名前空間内のサービスの名前（/service/ServiceName など）に対するプレースホルダです。 セッション・シェルのコマンド、createwebservice を参照してください。

次の例に示されるように、接続は tnsnames.ora ファイルで指定されます。

例 5-1 このシナリオの Apache 接続として、エントリ inst1\_http を定義する tnsnames.ora

```
inst1_http = (DESCRIPTION=
  (ADDRESS= (PORT=5521) (host=dlsun1609) (PROTOCOL=tcp))
  (CONNECT_DATA=
    (SERVICE_NAME=<WebServerName>)
    (PRESENTATION=http://admin)
  )
)
```

**注意：** <WebServerName> は、Web サーバー名（foo.us.oracle.com など）に対するプレースホルダです。

# OSE アプリケーションでの Apache の構成

OSE は、ステートフルな Web アプリケーション用に設計されています。このタイプのアプリケーションの場合、各ブラウザ・クライアントには、ユーザーがドメイン内で実行しているすべてのステートフルなアプリケーションを追跡するデータベース・セッションが割り当てられます。デフォルトでは、すべてのアプリケーションがステートフルとみなされます。

## Apache のステートフルなアプリケーションとステートレスなアプリケーション

OSE へのステートレスな接続を使用する Apache を構成できます。クライアントでは、特定のアプリケーションのデータベース・セッションは、ステータス情報を保持しません。これ

らの要求の場合、各 Apache プロセスに、任意のクライアントに対するステートレスな接続を処理する場合に使用されるセッションへ半永続的な接続があります。各 Apache プロセスは一度に 1 つの要求しか処理しないため、ステートレスな接続は、別の受信したクライアントにより順番に使用されます。ステートレスなサーブレット・コンテキスト およびステートレスなサーブレットは、JNDI 名前空間に公開された場合などに指定する必要があります。

セッション・シェルのコマンドで使用する ServletContext エントリおよび Servlet（各エントリについては、『Oracle8i Java Tools リファレンス』を参照）には、アプリケーション・コンテキストまたは特定のサーブレットがステートレスであることを宣言するために使用できる **-stateless** オプションがあります。この情報は、Export コマンドとともに使用し、構成の管理を簡略化します。

いくつかの Oracle APPS では、ステートレスなモデルが使用されます。このモデルは、OSE へのインストール時にこれらをステートレスと宣言する場合などに便利です。

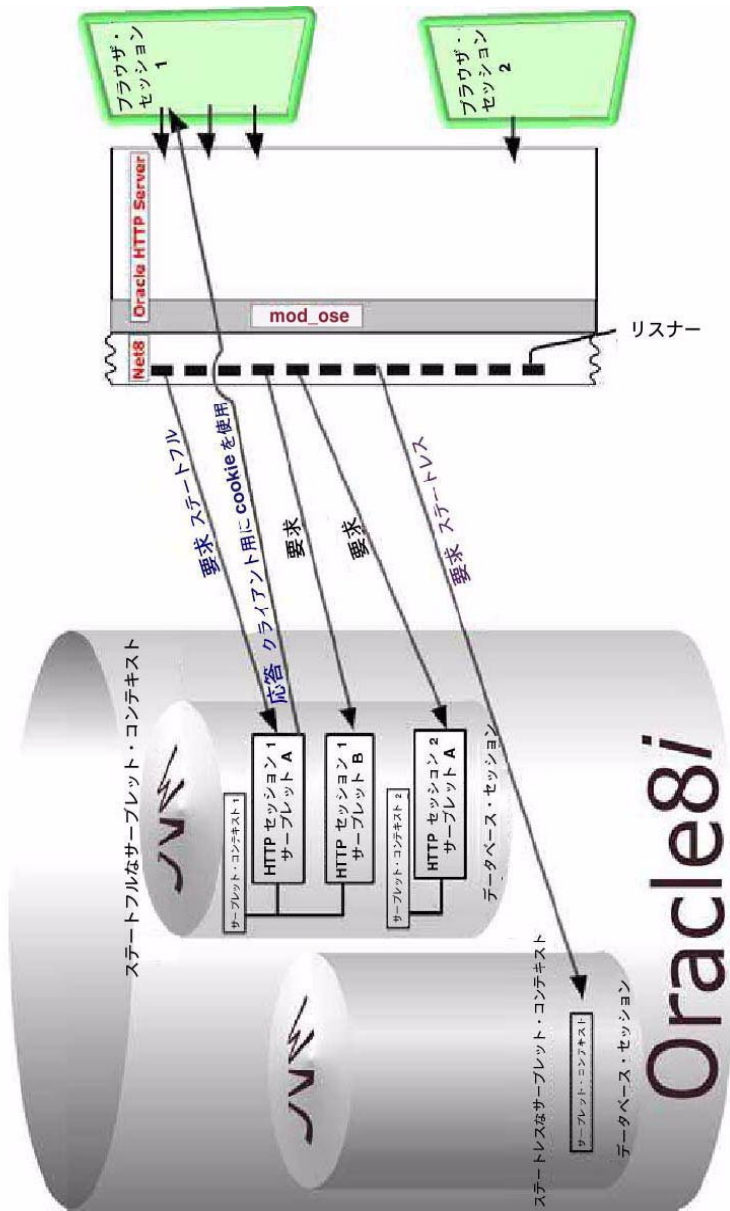
## Apache のステートフルなハンドラとステートレスなハンドラ

ステートフルな接続またはステートレスな接続のどちらを使用するかを指定する場合は、次のように、使用するハンドラの特定の可変を指定します。

```
SetHandler aurora-stateless-server
SetHandler aurora-statefull-server
SetHandler aurora-server
```

- 最初のハンドラ（ステートレスな接続を指定）は、要求が HTTP セッションをエラーとして開始するこのチャンネルを介して処理されたとみなします。
- 2 つ目のハンドラ（ステートフルな接続を指定）では、HTTP セッションを許可し、Apache で、要求を処理する個別のセッションを定義する必要があります。
- 3 つ目のハンドラでは、デフォルト・モード（ステートフル）が使用されます。

図 5-2 ステートフルなセッションとステートレスなセッション



## Apache の構成情報の抽出

OSE で実行している Web アプリケーションの構成を指定する一意の場所は、Oracle8i JVM 内の JNDI 名前空間です。ツール、`exportwebdomain` は、Web ドメインにインストールされているアプリケーションに関する情報を取得し、対応する `Apache.conf` ファイルを生成します。このファイルを Apache のメインの構成ファイルに追加します（例 5-2 「Web ドメイン構造の Apache CFG ファイルへのエクスポート」を参照）。

1. `mod_ose` の構成ファイル内の `webdomain` の構造を生成するには、`exportwebdomain` コマンドを使用します。エクスポート・ユーティリティは、次の 2 つの段階で機能します。
  - ドメイン内の `webdomain` または `contexts` の構造情報を XML 形式で生成する場合。
  - XML への変換を適用して、`mod_ose` に固有の構成ファイルを生成する場合。構成ファイルをシェルで生成します。

```
sess_sh -s <ose-url> -u <user> -p <passwd>
```

```
exportwebdomain -format Apache -netSERVICE <name in tnsname.ora> <Web domain> & > <file.conf>
```

このコマンドで、サーバーへ接続し、`webdomain` の情報を Apache に必要な形式で生成して、`<file.conf>` に保存します。

パラメータは次のように定義されます。

`<Web domain>` - エクスポートする Web ドメインの JNDI の位置（シングル・ドメインの場合はサービス・ルートがデフォルトになる）。

`<file.conf>` - 生成された構成を格納するファイルの名前。この例では、`Apache.conf`。

オプションは次のように定義されます。

**format type:** 定義された形式で出力を生成します。`mod_ose` の構成ファイルを生成するには、テキスト文字列 `Apache` または `apache` を使用します。

使用可能なすべてのオプションの詳細は、『Oracle8i Java Tools リファレンス』の「export コマンド」を参照してください。特に、特定のコンテキストのエクスポート、つまり `doc_root`（静的ページ）を OSE に要求するか、あるいはローカルで処理するかについて選択できます。

### 例 5-2 Web ドメイン構造の Apache CFG ファイルへのエクスポート

1. 次を入力します。

```
exportwebdomain -format apache -netSERVICE inst1_http /webdomains & > /apache/config/webdomains.cfg
```

出力ファイル `webdomains.cfg` には、Web ドメイン構成が格納されます。



2. 次の例に示される、生成されたファイルは、Apache 構成ファイル、httpd.conf の最後尾に追加されます。

**例 5-3 exportwebdomain コマンドの出力結果 : webdomains.cfg**

```
# Apache configuration
# Domain: /webdomains
#
<IfModule mod_ose.c>

AuroraService inst1_http
#
# Context for VPATH /context-test/
#

<Location /context-test/ >
AddHandler aurora-server snoop
</Location>

<Location /context-test/admin/shell >
SetHandler aurora-server
</Location>

<Location /context-test/errors/internal >
SetHandler aurora-server
</Location>

<Location /context-test/examples/counter >
SetHandler aurora-server
</Location>

<Location /context-test/examples/snoop >
SetHandler aurora-server
</Location>

<Location /context-test/servlet/* >
SetHandler aurora-server
</Location>

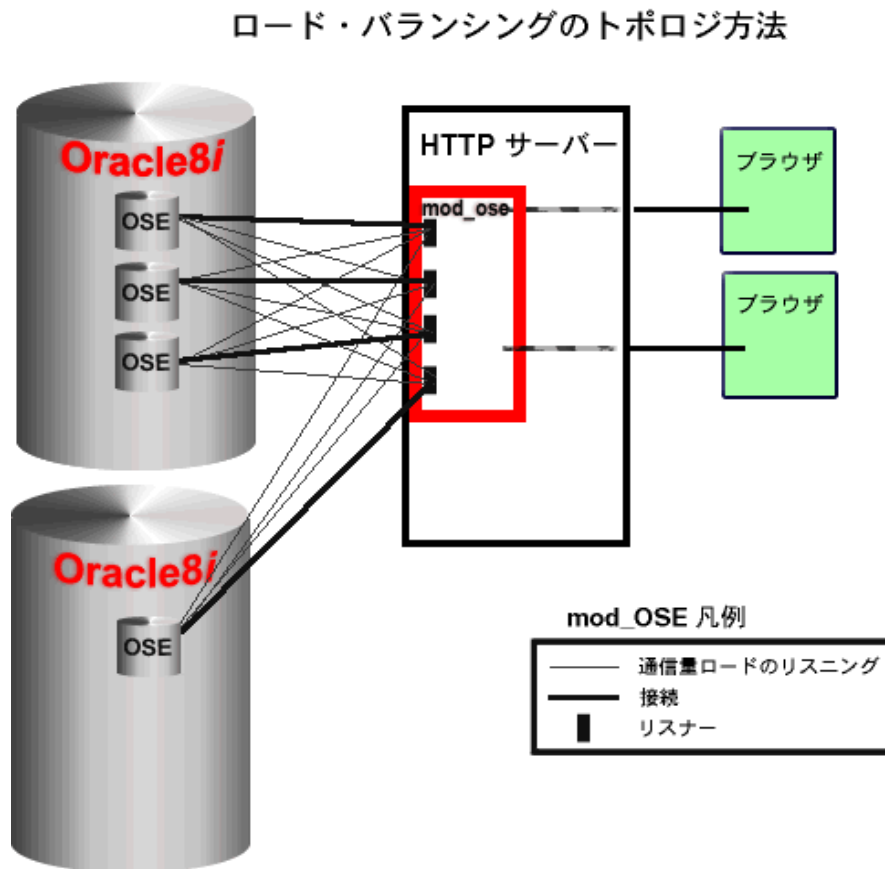
</IfModule>

#
# End of configuration
```

## mod\_ose を使用したサイトのトポロジ

mod\_ose を使用すると、Web 環境のシステムに様々なネットワーク・トポロジを構成できます。特に、単一障害点のない構成を定義できます。このような構成では、ノードの障害が発生すると、使用されているデータベース・インスタンスの 1 つに障害がある場合、利用可能な Oracle リスナーにより他のデータベース・インスタンスに要求をリダイレクトできます。この機能を使用するためには、アプリケーションをすべてのノードで複製し、アプリケーションで、期限切れのデータベース・セッションからのリカバリを処理できるようにする必要があります。

図 5-3 Net8 を使用した、mod\_ose でのリスナーとロード・バランシング



## 事前インストールされている Apache システムのための mod\_ose のインストール

Apache がすでにシステムにインストールされている場合は、OSE に必要な構成を設定する必要があります。Apache と OSE が Oracle からインストールされている場合は、インストールはできています。

Apache インストールの bin ディレクトリから、次のコマンドを実行します。

```
apxs -i -a -n mod_ose $APACHE_HOME/Apache/libexec/libjipa8i.so
```

このコマンドで、モジュールを Apache インストールにコピーして、該当する LoadModule ディレクティブを httpd.config に追加すると、そのモジュールが使用できます。次に Apache を再起動すると、新しいモジュールを使用できます。

---

---

**注意：** インストーラは、ライブラリを \$APACHE\_HOME/Apache/libexec で使用できるようにする必要があります。

libjipa8i.so が依存する libjip.so は、インストーラによって \$ORACLE\_HOME/lib にコピーされます。

---

---

## Apache 構成

Apache 構成は、次の 2 つの部分に分かれています。

- [OSE サービスの指定](#)（要求用）
- [動的要求の指定](#)（URL）

### OSE サービスの指定

このステップでは、データベース内で OSE と通信する場合に使用する接続を構成します。Apache の OSE サービスは、仮想ホスト・レベルでのみ構成できます。仮想ホストのパスは、1 つの OSE サーバーにのみルートできます。

OSE への接続に使用する接続記述子を指定するには、mod\_ose コマンド AuroraService を使用します。構文は次のようになります。

```
<IfModule mod_ose.c>
    AuroraService connection-name
</IfModule>
```

次の構成例では、使用する接続を定義し、[例 5-1](#) で定義した、inst1\_http 接続を示します。

#### 例 5-4 OSE へ接続する接続記述子の定義

```
<IfModule mod_ose.c>
    AuroraService inst1_http
</IfModule>
```

仮想ホストで、独自の AuroraService 構成を定義しない場合は、デフォルトのホスト構成が使用されます。

#### 動的要求の指定

mod\_mime の一部として定義される、AddHandler および SetHandler コマンドは、Aurora へ送信される要求を指定します。(mod\_mime については 5-2 ページの「[Apache で OSE の要件](#)」を参照してください。)

- AddHandler は、ファイル拡張子を指定します。
- SetHandler は、OSE へ送信されるアドレス・セグメントを指定します。

#### 例 5-5 OSE により処理されるファイル拡張子を指定する AddHandler

```
AddHandler aurora-server .jsp .snoop
```

これで、拡張子が .jsp または .snoop のファイルのステートフルな要求は、OSE へ送信されます。

#### 例 5-6 すべての接続要求を指定し、OSE へ送信されるアドレス・セグメントを解析する SetHandler

```
SetHandler aurora-server
```

このコマンドは、Location ディレクティブとともに使用する必要があり、OSE で処理される仮想パスのコンポーネントを次のように解析します。

```
<Location /examples/>
    SetHandler aurora-server
</Location>
```

Apache で使用される要求の仮想パスは、OSE へそのまま転送されます。OSE では、JNDI 名前空間情報から渡された Apache 構成文字列のセグメントが生成されます。

## URL の転送

HTTP 要求が Apache から `mod_ose` へ送信されると、初期のパス情報、`http://<web-domain:port>` は破棄されます。残りのパスは、`OSE` へ送信されます。サブレットが実行されると、その結果はクライアントへ戻されます。通常、要求パスは要求を `mod_ose` からサブレット・サーバーヘルディングする Apache を起動します。

### 例 5-7 要求を `mod_ose` から `OSE` ヘルディングする Apache

Apache がリスニングしているポートが 3000 で、次のディレクティブが `httpd.conf` ファイルに挿入されている場合は、次のようになります。

```
<Location /MyContexts>
    SetHandler aurora-server
</Location>
```

クライアントは、`http://apacheserver:3000/MyContexts/Counter` を要求します。`mod_ose` は、要求を `/MyContexts/Counter` として `OSE` へ転送します。`/Counter` が、`/MyContexts` サブレット・コンテキストのディレクトリ内の公開されたサブレットとして検出された場合は、`OSE` で応答が送信されます。見つからない場合は、デフォルトのサブレットによって、デフォルトの `Error 404 (Not Found)` が戻されます。この動作を変更するには、サブレット管理ツールを使用します。

静的ファイル进行处理する場合も同じ論理が適用されます。

### 例 5-8 静的ファイル进行处理する Apache と `mod_ose`

クライアントが `http://apacheserver:3000/MyContexts/index.html` を要求したとします。

この場合、`/MyContexts/index.html` ファイルは、ルート・ディレクトリ下に、URL で指定されたとおりに存在する必要があります。このファイルは、パスとファイル名を送信する方法と、パス全体が読み取られる方法を示しています。

---

---

**注意：** 例 5-8 では、`mod_jserv` を使用するユーザーために、特定の要求操作の詳細を解説しています。

---

---



---

## OSE サーバー構成

この章では、例と説明を使用して、OSE の管理方法とツールを説明します。

この章では、次の項目について説明します。

- OSE の設定
- サービスの作成
- ドメインの作成
- サブレット・コンテキストの作成
- サブレットの追加

## OSE の設定

OSE は、エンド・ポイントをリスニングするように構成されたサービスを介して機能します。サービスには、1 つ以上のドメインがあります。ドメインには、1 つ以上のサーブレット・コンテキストが格納されます。サーブレット・コンテキストは、Web アプリケーション層を表し、サーブレットおよび様々なサポート・エントリが格納されます。

ドメインには、仮想パスがサーブレット・コンテキストにマップされるように構成されています。サーブレット・コンテキストは、仮想パスがサーブレットにマップされるように構成されます。

---

---

**注意：** 次の項で使用されるコマンドの詳細は、『Oracle8i Java Tools リファレンス』を参照してください。

---

---

## サービスの作成

サービスの基本レベル、サービス名、プロパティ・グループおよびルートの位置を定義するには、`createwebsevice` コマンドを使用します。マルチ・ドメイン・サービスを作成する場合は、このコマンドで IP アドレスおよび仮想ホスト名も定義します。構成したサービス構造を参照するには、次の `getproperties` コマンドを使用します。

```
getproperties /service/<service_name>
```

既存のデータベース・リスナーを使用して新しいエンド・ポイントを動的に追加するには、`addendpoint` コマンドを使用します。新しいエンド・ポイントは、静的に作成することもできます。

## ドメインの作成

Web ドメインを作成するには、`createwebdomain` コマンドを使用します。新しい Web ドメインは、コマンドを実行したスキーマにより所有されます。このドメインに格納されているサーブレットは、ドメインの所有者として実行されます。各 Web ドメインは、サーブレット・コンテキスト、`/default` が生成されます。

## コンテキスト・グループ

構成したドメイン構造の設定を参照するには、次を入力します。

```
getproperties <domainroot>/config
```



コンテキスト・グループには、HTTP 要求を処理する仮想パスとコンテキスト間のマッピングのリストを持ちます。これは、名前と値の組合せのリストになります。

- 名前の部分は、仮想パスになります。
- 値の部分は、Web ドメインのコンテキストのサブディレクトリに対応する、コンテキストの名前になります。

このリストは、HTTP 要求を処理する適切なサーブレット・コンテキストにマップされる仮想パスのリストです。

## MIME グループ

MIME グループは、Web ドメインでサポートされる MIME タイプのマッピングに対する拡張子をリストします。

## サーブレット・コンテキストの作成

サーブレット・コンテキストを作成するには、`createcontext` コマンドを使用します。サーブレット・コンテキストは、ドメインのルート内の `contexts` ディレクトリに格納されます。ステートレスなサーブレットまたはステートフルなサーブレットのどちらかをサポートするように、サーブレット・コンテキストを設定できます。

サーブレット・コンテキストの設定を参照するには、次を入力します。

```
getproperties <domainroot>/contexts/<servlet_context>/config
```

サーブレット・コンテキストの `config` オブジェクトの仮想マッピングを参照するには、次のコマンドを使用します。

```
getproperties <domainroot>/config
```

## MIME グループ

MIME グループは、Web サーバーでサポートされる MIME タイプのマッピングに対する拡張子をリストします。

## サーブレットの追加

サーブレット・コンテキストでサーブレットを公開するには、`publishservlet` コマンドを使用します。このコマンドでは、仮想パスを指定したサーブレットに対応付けることもできます。構成したサービス構造の設定を参照するには、次のように入力します。

```
getproperties <domainroot>/contexts/<servlet_context>/named_servlets/<servletA>
```

サーブレット・コンテキストの `config` オブジェクトの仮想マッピングを参照するには、次のコマンドを使用します。

```
getproperties <domainroot>/contexts/<servlet_context>/config
```

---

## 開発者用のツールと手順

この章では、例と説明使用して、OSE での開発方法とツールを説明します。

この章では、次の項目について説明します。

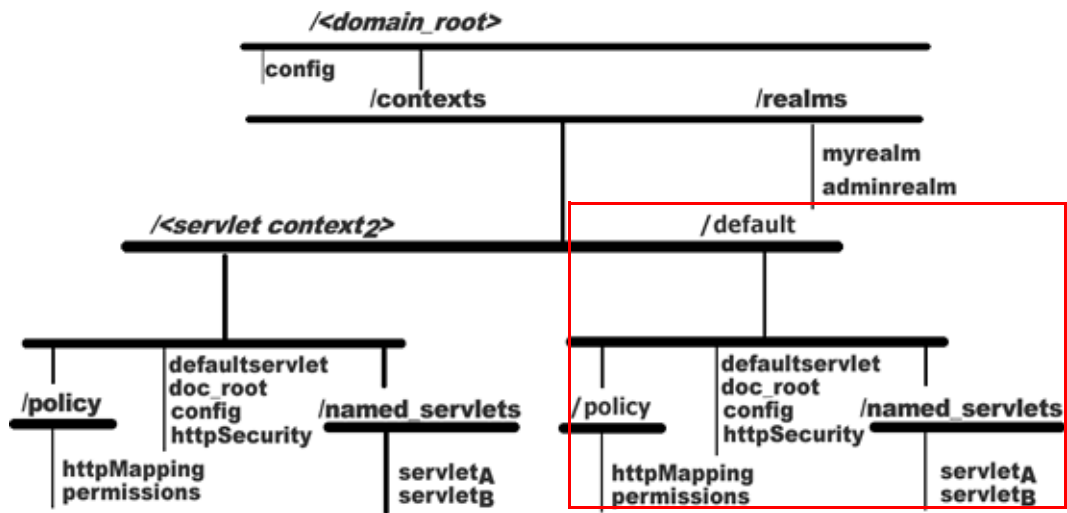
- [Web サービス](#)
- [サーブレット・コンテキストの管理](#)
- [サーブレット・コンテキストの内容](#)
- [公開されたサーブレットの管理](#)
- [サーブレットの作成](#)

# Web サービス

Web サービスには、サービスの構成に応じて、1つ以上のドメインが格納されます。新しく作成された Web ドメインには、1つのサーブレット・コンテキスト、default が格納されます。このサーブレット・コンテキストは、別のサーブレット・コンテキストにマップされないすべての要求のコンテンツを処理します。独自のアプリケーションに固有のサーブレットおよびサーブレット・コンテキストを作成できます（図 7-1 「JNDI 名前空間の JNDI デフォルト・サーブレット・コンテキスト」を参照）。

JNDI 名前空間のディレクトリ構造は、別のサーブレット・コンテキストに対応しています。JNDI 名前空間ツリーの両方のブランチに、同じ構造要件が適用されます。シングル・ドメインと複数ドメインの図示による例は、図 1-2 および図 1-3 を参照してください。

図 7-1 JNDI 名前空間の JNDI デフォルト・サーブレット・コンテキスト



サービスのプロパティを定義するには、createwebsevice コマンドを使用します。このコマンドの詳細は、『Oracle8i Java Tools リファレンス』に記載されています。

## Web サービスの仮想ホストと IP ドメイン

シングル・ドメインには、IP アドレス、仮想ホスト・ドメインのどちらもありません。ドメインのルートは、サービスのルートです。サービスを仮想ホストまたは IP アドレスのどちらかとして、マルチ・ドメイン・サービスとして構成している場合は、ドメインはサービス・ルート内でルーティングされます。サービスに、IP アドレスおよび仮想ホストの両方のタイプのドメインがある場合は、IP アドレスはサービス・ルート内でルーティングされ、仮想ホ

ストは IP アドレス・ドメインに格納されます（[図 1-3「仮想ホストを持つマルチ・ドメイン、マルチホームの例」](#)を参照）。

これらのドメインは、`createwebdomain` コマンドで作成します。構文および引数の詳細は、『Oracle8i Java Tools リファレンス』を参照してください。

## Web サービスのポートの変更

ポートは、静的に、または動的な登録コマンドを使用して、該当する表示に対応付けられます。HTTPS をサポートするには、追加の SSL エンド・ポイントを Web サービスに対応付けます。

サービスに対応付けるポートを構成するには、`addendpoint` コマンドを使用します（構文と引数については、『Oracle8i Java Tools リファレンス』を参照）。

## サブレット・コンテキストの管理

この項では、サブレット・コンテキストの作成、変更および削除の方法を説明します。

### サブレット・コンテキストの作成または変更

新しいサブレット・コンテキストを作成するには、セッション・シェルで `createcontext` コマンドを使用します。

- サブレット・コンテキストを作成するには、格納しているドメインの `contexts` ディレクトリで、書き込み権限を持つユーザー名を設定する必要があります。
- 格納しているドメインの `config` オブジェクトで、書き込み権限を持つユーザー名を設定する必要があります。
- 仮想パスとして渡される引数がドメインの `config` オブジェクトに登録されると、新しいサブレット・コンテキストは HTTP クライアントにアクセスできるようになります。
- サブレットは、コンテキストの作成者の識別情報を使用して実行されます。

`createcontext` は、次のプロパティのすべてまたは一部を定義および設定します。

- このコンテキストに対応付けられている仮想パスを定義します。
- プロパティのリストを使用してコンテキストを初期化します。
- （オプション）静的ページの位置を定義します。
- （オプション）以前指定した既存のコンテキストの破棄および置換をします。
- （オプション）ステートレスのフラグがついている場合、コンテキストはステートレスで、サブレットでは HTTP セッション・オブジェクトを取得できません。

例 7-1「**winecellar** という名前のサブレット・コンテキストの作成」では、サブレット・コンテキストは `http://<host-name>:8080/cellar` で始まるすべての HTTP 要求を処理します。

### 例 7-1 **winecellar** という名前のサブレット・コンテキストの作成

1. セッション・シェルを起動します。
2. 新しいサブレット・コンテキストを作成します。

次のように入力します。

```
$createcontext -virtualpath /cellar /webdomains winecellar
```

	仮想パス	ドメイン名	サブレット・コンテキスト
--	------	-------	--------------

3. コンテキストをリストします。

次のように入力します。

```
$ ls /webdomains/contexts
./                ../                default/          winecellar/
```

4. コンテキストは、Web サービス仮想パスのマッピングでもリストされます。次を入力します。

```
$ getgroup /webdomains/config contexts
/cellar=winecellar
```

## サブレット・コンテキストの削除

サブレット・コンテキストとコンテキスト内のすべてのサブレットを削除するには、セッション・シェルの `destroycontext` コマンドを使用します。ドメインの `config` オブジェクトのマッピングも削除されます。

## サブレット・コンテキストの内容

サブレット・コンテキストは、Web アプリケーションの概念をカプセル化しています。サブレット・コンテキストには、その構成とコンテンツを表すいくつかのエントリが格納されます。エントリは、セッション・シェルを使用して参照できます。

### 例 7-2 構成とコンテンツを表す **winecellar** のサブレット・コンテキストのエントリ

1. セッション・シェルを起動します。
2. `$ cd /webdomains/contexts/winecellar`

### 3. \$ ls

```
./  
../  
config  
doc_root  
named_servlets/  
defaultServlet  
policy  
httpSecurity
```

## config オブジェクト

config エントリには、サブレット・コンテキストの動作を制御するプロパティが含まれています。config オブジェクトを参照および変更するには、オブジェクト・プロパティを管理するセッション・シェルのコマンドを使用します。詳細は、『Oracle8i Java Tools リファレンス』を参照してください。

## doc\_root オブジェクト

doc\_root オブジェクトは、ファイル・システムのリンクとして機能するクラス `SYS:oracle.aurora.namespace.filesystem.FSContextImpl` のインスタンスを表す JNDI オブジェクトです。doc\_root オブジェクトは、静的コンテンツの位置が特定される場所です。このオブジェクトには、`FSContextURL` と呼ばれる 1 つのプロパティがあります。サブレット・コンテキストの静的コンテンツのファイル・パスは、`FSContextURL` プロパティに格納されます。

## named\_servlets サブディレクトリ

named\_servlets サブディレクトリには、サブレット・コンテキストで公開されたサブレットが格納されます。公開された各サブレットは、クラス `SYS:oracle.aurora.mts.ServletActivation` のインスタンスを表す JNDI オブジェクトになります。

## defaultServlet オブジェクト

サブレット・コンテキストに `defaultServlet` という名前のエントリがある場合は、要求を満たすサブレットが他にない場合にそのサブレットが使用されます。サブレット・コンテキストにデフォルトのサブレット用のエントリがない場合は、格納されているサービスによってデフォルトのサブレットが指定されます。

## policy ディレクトリ

policy ディレクトリには、セキュリティ構成のエントリが格納されます。詳細は、[第 8 章「セキュリティ HTTP の管理」](#)を参照してください。

## httpSecurity

セキュリティ・サーブレット、httpSecurity は、すべての要求に対する最初のフィルタとして機能します。このサーブレットでは、例外を呼び出すか、例外を呼び出さずに要求の処理を継続して、セキュリティが処理されます。詳細は、[第 8 章「セキュリティ HTTP の管理」](#)を参照してください。

## サーブレット・コンテキスト・グループ・パラメータ

この項では、サーブレット・コンテキストの config オブジェクトでサポートされるグループとプロパティの完全なリストを示します。

グループ、context.properties には、サーブレット・コンテキストを制御するプロパティがリストされます。

### 例 7-3 config プロパティの参照と変更

```
$ getproperties config
--group--=context.properties
context.browse.dirs=true
context.welcome.names=index.html
context.accept.charset=ISO-8859-1
context.accept.language=en
context.default.languages=*
context.default.charsets=*
--group--=context.params
...
--group--=context.mime
java=text/plain
html=text/html
...

$ addgroupentry config context.properties context.browse.dirs false
$ getgroup config context.properties
context.browse.dirs=false
context.welcome.names=index.html
context.accept.charset=ISO-8859-1
context.accept.language=en
context.default.languages=*
context.default.charsets=*
```



## context.params

context.params グループは、サーブレット・コンテキストの `getAttribute` メソッドで、実行時にアクセスできる任意の名前と値の組合せのリストです。

### 例 7-4 サーブレット・コンテキスト `getAttribute` メソッド

次のパラメータを指定すると、

```
$ getgroup config context.properties
details=high
```

サーブレット内の `details` パラメータに次のようにアクセスできます。

```
public void doGet (HttpServletRequest request, HttpServletResponse response)
    ... {
    ...
    String details = getServletContext().getAttribute ("details");
    if (details.equals ("high")) {
        ...
    }
}
```

## context.mime

グループ `context.mime` は、サーブレット・コンテキストでサポートされる MIME タイプをリストします。このグループの形式は、Web ドメイン構成の MIME タイプ・グループと同じ形式です。

## context.servlets

サーブレットを公開するときに、それらのサーブレットを仮想パスと対応付けて、セッション・シェルの `getgroup` コマンドでこれらの仮想パスを参照できます。詳細は、項「[JNDI 名前空間の JNDI デフォルト・サーブレット・コンテキスト](#)」を参照してください。

グループ `context.servlets` は、公開されたサーブレットの仮想パスのマッピングをリストします。このグループには、名前と値の組合せのリストが格納されます。この場合、名前の部分は仮想パスになり、値の部分は、仮想パスを処理するサーブレットの名前になります。サーブレット名は、サーブレット・コンテキストの `named_servlets` ディレクトリ内のオブジェクトに対応します。このペアの名前の部分には、仮想パスまたはワイルドカード名が含まれます。

## context.error.uris

グループ `context.error.uris` は、HTTP エラー・コードを、HTTP クライアントに対してエラーをレポートするために使用される、サーブレット・コンテキスト内の URI と対応付けます。これらの URI は、デフォルトのサーブレットで頻繁に処理され、サーブレット・コンテキストの `doc_root` パラメータに関連します。

```
$ getgroup config context.error.uris
```

```
401=/system/errors/401.html  
403=/system/errors/403.html  
404=/system/errors/404.html  
406=/system/errors/406.html  
500=/errors/internal
```

## 公開されたサーブレットの管理

サーブレットは、`named_servlets` ディレクトリでアクティブなサーブレット・エントリと仮想パスが、サーブレット・コンテキストの `config` オブジェクトでマップされると公開されます。

この項では、OSE で公開されたサーブレットの管理方法を説明します。

## サーブレット・コンテキストで公開されたサーブレット・クラス

セッション・シェルの `publishservlet` コマンドと `unpublishservlet` コマンドで、`named_servlets` ディレクトリとサーブレット・コンテキストの `config` オブジェクトを変更します。サーブレットは、対応付けられたパスと名前を使用して作成（公開）または削除されます。

公開されたサーブレットは、クラス `SYS:oracle.aurora.mts.ServletActivation` の JNDI オブジェクトになります。HTTP クライアントからアクセスするには、サーブレット・コンテキストで、サーブレットを仮想パスまたはワイルドカードと対応付ける必要があります。

### 例 7-5 サーブレットの公開

```
$ publishservlet -virtualpath /tastings \  
/webdomains/contexts/winecellar tastingServlet \  
SCOTT:winemasters.tasting.Tasting
```

---

**注意：** 行の継続には、バックスラッシュを使用します。これは、非常に長いエントリを処理する場合のシェル・ツールのコマンドライン表記規則です。

---

### 例 7-6 新しいサーブレットと仮想パスのマッピングの確認

```
# サーブレットの確認  
$ getproperties /webdomains/contexts/winecellar/named_servlets/tastingServlet  
servlet.class=SCOTT:winemasters.tasting.Tasting
```

```
# 仮想パスのマッピングの確認
$ getgroup /webdomains/contexts/winecellar/config context.servlets
/errors/internal=internalError
/tastings=tastingServlet
```

新しく公開されたサーブレットには、1つのプロパティ、`servlet.class`しかありません。`servlet.class`は、サーブレット・クラスの完全な名前（スキーマおよびスキーマ内の完全なパス）を指定します。スキーマを指定しない場合は、現行のスキーマが使用されます。

プロパティを追加するには、セッション・シェルのコマンドを使用し、サーブレットのソースコードから追加のプロパティにアクセスするには、`getInitParameter()` メソッドを使用します。

#### 例 7-7 サーブレットへのプロパティの追加

```
$ setproperties invoker \
"servlet.class=SCOTT:winemasters.tasting.Tasting
details=high
style=parker"
```

```
$ getproperties invoker
servlet.class=SCOTT:winemasters.tasting.Tasting
details=high
style=parker
```

サーブレット・プロパティには、次のコードでアクセスできます。

```
public void doGet (HttpServletRequest request, HttpServletResponse response)
{
    ...
    String details = getServletConfig().getInitParameter ("details");
    if (details.equals ("high")) {
        ...
    }
}
```

## サーブレットの作成

OSE を設定すると、サーブレットおよび JSP を作成および配置することができます。

---

**注意：** この項を理解するには、「[アーキテクチャ](#)」の章の「[エンド・ポイント](#)」の項を理解しておくことが前提条件になります。

---

サーブレット構築の基礎を学ぶためには、『Java Servlet Programming』（O'Reilly and Associates）などの、サーブレットの参考文献が役に立ちます。

---

「Servlet 2.2 仕様」をサポートします。

次の手順は、サーブレットの作成と公開の方法を示します。

1. サーブレットのソースコードを作成します。
2. サーブレットをコンパイルします。
3. サーブレットのクラスをデータベースにロードします。
4. サーブレットを公開します。
5. Web ブラウザからサーブレットを要求します。

## サーブレットのソースコードの作成

サーブレットの参考文献には、様々な例とチュートリアルが示されています。コードの作成については、サーブレットの参考文献を参照してください。

## サーブレットのコンパイル

サーブレットをコンパイルするには、クラスパスに次の jar が必要です。

1. http クラスを格納する jar は、次のようになります。  
`$(ORACLE_HOME)/jis/lib/servlet.jar`
2. サーブレットでデータベースにアクセスする場合は、次の JDBC クラスも必要です。  
`-(ORACLE_HOME)/jdbc/lib/classes12.zip`  
`-% javac -classpath .:$(ORACLE_HOME)/jis/lib/servlet.jar HelloWorld.java`

## サーブレット・コードのロード

サーブレットをロードするには、loadjava コマンドを使用します。Oracle8i JVM 関連のマニュアルで loadjava に関する項目を参照してください。

```
% loadjava -u scott/tiger -r HelloWorld.class
```

## サーブレットの公開

この例では、default のコンテキストでサーブレットを公開します。サーブレットを仮想パス /hello にマッピングします。

```
$ publishservlet -virtualpath /hello /webdomains/contexts/default helloServlet SCOTT:HelloWorld
```

サーブレットを変更する場合は、データベースに変更したサーブレットを再ロードする必要がありますが、Web サーバーでサーブレットを再発行する必要はありません。

## サーブレットへのアクセス

Web ブラウザでサーブレットを参照するには、次の URL を使用します。  
`http://<host>:8080/hello`



---

# セキュリティ HTTP の管理

## 概要

この章では、OSE で HTTP セキュリティを設定する場合に役立つ一連の操作を説明します。

この章では、次の項目について説明します。

- [Web サーバー・セキュリティの前提条件](#)
- [認証と許可](#)
- [プリンシパルの宣言](#)
- [ツール](#)
- [リソースの保護](#)
- [アクセス権の宣言](#)
- [セキュリティ・サーブレットの宣言](#)
- [トラブルシューティング](#)

## Web サーバー・セキュリティの前提条件

OSE でセキュリティ設定を定義およびプログラムするには、JNDI 構造の設定と操作を十分に理解しておく必要があります。次のチェックリストで、どの程度理解しているか確認してください。

- ☐ シングル・ドメイン (1-7 ページ)
- ☐ マルチ・ドメイン (1-8 ページ)
- ☐ HTTP 仮想パス (2-4 ページ)
- ☐ 仮想パスのサーブレット・コンテキストへのマッピング (2-8 ページ)
- ☐ HTTP 要求 (4-2 ページ)
- ☐ サーブレットにより処理される要求 (2-5 ページ)
- ☐ デフォルトのサーブレットにより処理される要求 (4-4 ページ)
- ☐ サービスの作成 (7-2 ページ)
- ☐ コンテキストの作成 (7-3 ページ)

前述のトピックの中でよくわからないものがある場合は、OSE で JNDI 名前空間を処理する方法および Web ブラウザで OSE と通信する方法を示す図や例を再度参照してください。各トピックのリンクを使用すると、その説明が表示されます。例の完全なリストは、[付録 B「例」](#)に示されています。

この章で基本的なセキュリティの設定を学ぶと、デフォルトの設定を使用して、独自の Web サービス・セキュリティの構成を安全に変更できます。

## 認証と許可

HTTP セキュリティでは、保護付きのリソースへのアクセスは、有効な資格証明の認証およびユーザーの許可の 2 つの部分で構成されます。認証は、ユーザーがシステムにより認識および検証されたことを証明する発行済み資格証明の検証です。許可は、認証されたユーザーが要求されたアクションを実行できるかどうかの判断です。

セキュリティ保護装置を設置するときは、次の 4 つの段階を宣言します。

1. サービスのプリンシパル
2. 保護するリソースと保護の方法
3. サーブレット・コンテキスト内のプリンシパルのアクセス権
4. サーブレット・コンテキストの root のセキュリティ・サーブレット

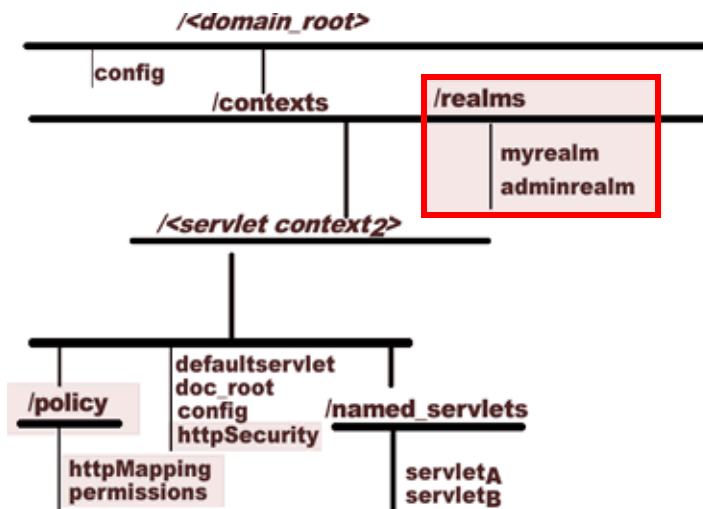
前述の手順を使用して、Web リソースの HTTP セキュリティを定義するときに適切な基本情報が使用されているか確認します。前述の手順が 1 つでも欠けると、セキュリティは存在しなくなるか、保護付きリソースへのアクセスが制限されます。



## プリンシパルの宣言

プリンシパルは、ユーザーまたはグループのいずれかを表す一般的な用語です。レルムは、サービス内のオブジェクトで、宣言されたプリンシパルが格納されます。図 8-1 は、Web サービスの最上位に位置するレルムオブジェクトの位置を示しています。これは、サービスの config オブジェクトと同じレベルに位置します。

図 8-1 ドメイン・ルート内の /realms を示す JNDI 構造



## グループ

グループには、他のプリンシパル（ユーザーまたはその他のグループ）が格納されます。グループの個別のメンバは、グループ・オブジェクトのアクセス権を継承します。

## ユーザー

ユーザーはシングル・オブジェクトです。グループとは異なり、ユーザーに属するその他のプリンシパルのサブセットはありません。

## レルム

各レルムは、プリンシパルの個別のセットを定義します。レルムとその実体は、すべての HTTP セキュリティの中心となります。サービス内には複数のレルムを置くことができます。レルムは、次のソースになります。

- プリンシパルの有効なセット
- サーバーに渡されるプリンシパルのタイプ

レルムはすべてのプリンシパルの基本となるため、次の点で重要な役割も果たします。

- プリンシパルを識別するために使用する資格証明のタイプの判別
- 資格証明の管理に対してプリンシパルを支援または資格証明を持つエンティティの判別
- レルム内でのすべてのプリンシパルの関係の確立

デフォルトでは、HTTP セキュリティで次の 3 つのレルムの実装が指定されています。

- JNDI – 名前空間に JNDI エントリのすべての情報を格納します。
- DBUSER – データベースのローカルのユーザー定義に従います。
- RDBMS – より一般的なデータベース表にすべてのプリンシパルとその関係を格納します。

これらの名前は単なる短縮形です。レルムをインスタンス化する場合は、JNDI 名前空間でレルムクラスを宣言するときの適切な名前を使用してください。

### DBUSER の考慮事項

DBUSER レルムは、データベース内で定義されたユーザーとロールから導出されたプリンシパルの定義です。次のいくつかの含意があります。

- どのセキュリティ・コマンドでも、プリンシパルの管理は許可されません。プリンシパルの作成、削除およびロール・メンバーシップは、セキュリティ・ツールではなく、データベースにより管理されます。
- すべてのインスタンスで、プリンシパルの定義に同じソースを利用するため、すべてのインスタンスは必ず等価になります。
- プリンシパルを表す場合は、Java システムで表示されるように、名前の形式に対して大 / 小文字の変換は実行されません。つまり、データベース・エンティティが作成されると、大 / 小文字を明示的に表さないかぎり、名前はすべて大文字になります。たとえば、SYS および PUBLIC は大文字で表しても、次に示すようにユーザーを小文字で作成した場合は、

```
create user "joe" identified by welcome;
```

ユーザー名は小文字の joe になります。

---

**注意：** 大 / 小文字の区別は、ブラウザからユーザー名とパスワードを指定する場合重要です。

---

## ツール

realm コマンドでセキュリティ・レベルを設定する場合は、様々なフラグとオプションを使用します。すべての有効なレルムの定義は、『Oracle8i Java Tools リファレンス』で参照できます。様々な選択項目のリストを参照するには、セッション・シェルから realm コマンドを実行します。

### 例 8-1 有効なセキュリティ・コマンドをリストする Realm コマンドの実行

```
$ realm
usage:
realm
realm list -d <webServiceRoot>
realm echo [0|1]
realm secure -s <servletContextPath>
realm map -s <servletContextPath> [- add|remove <path>] [-scheme <auth>:<realm>]
realm publish -d <webServiceRoot> [- add|remove <realmName>] [-type RDBMS | DBUSER | JNDI]
realm user -d <webServiceRoot> -realm <realmName> [- add|remove <userName> [-p <user password>]]
realm group -d <webServiceRoot> -realm <realmName> [- add|remove <groupName> [-p <group password>]]
realm parent -d <webServiceRoot> -realm <realmName> [-group <groupName> [- add|remove <principalName>]]
[-query <principalName>]
realm perm -d <webServiceRoot> -realm <realmName> -s <servletContextPath> -name <principalName> [-path <path>
(+|-) <permList>]
```

## コマンド構造

中心的なロールであるため、realm は、シェル内のすべてのセキュリティ・コマンドの先頭に置かれます。次の項では、シェルからレルムを作成および管理するコマンドの例を示します。

### レルム

RDBMS レルムを作成するには、次を入力します。

```
realm publish -w /myService -add testRealm -type RDBMS
```

JNDI と DBUSER の場合は、その文字列をタイプ引数として使用します。

レルムを削除するには、次を入力します。

```
realm publish -w /myService -remove testRealm
```

レルム宣言は、JNDI 名前空間に常駐します。一度作成されて、カスタマイズされたレルムを配置するには、名前空間エントリを若干カスタマイズする必要があります。

ユーザー定義のレルムを公開するには、次を入力します。

```
realm publish -w /myService -add testRealm -classname foo.bar.MyRealm
```

### プリンシパル

user を作成するには、次を入力します。

```
realm user -w /myService -realm testRealm1 -add user1 -p upswdl1
```

group を作成するには、次を入力します。

```
realm group -w /myService -realm testRealm1 -add group1 -p gpswd1
```

前述のコマンドのいずれかで、パスワードを空白のままにすると、プリンシパル名がかわりに使用されます。

user を削除するには、次を入力します。

```
realm user -w /myService -realm testRealm1 -remove user1
```

group を削除するには、次を入力します。

```
realm group -w /myService -realm testRealm1 -remove group1
```

レルムの user をリストするには、次を入力します。

```
realm user -w /myService -realm testRealm1
```

レルムの group をリストするには、次を入力します。

```
realm group -w /myService -realm testRealm1
```

group にプリンシパルを追加するには、次を入力します。

```
realm parent -w /myService -realm testRealm -group group1 -add user1
```

group からプリンシパルを削除するには、次を入力します。

```
realm parent -w /myService -realm testRealm -group group1 -remove user1
```

group 内のプリンシパルをリストするには、次を入力します。

```
realm parent -w /myService -realm testRealm -group group1
```

プリンシパルがメンバーであるグループを問い合わせるには、次を入力します。

```
realm parent -w /myService -realm testRealm -q user1
```

(この問合せオプションをサポートしないレルムもあります。)

---

**注意：** プリンシパルの変更をサポートしないレルムもあります。たとえば、DBUSER レルムでは、プリンシパルの操作はサポートされません。

---

## 詳細

サービスでレルムが宣言されている場合は、それらのレルムは、サービスの realms サブコンテキストに位置します。JNDI レルムの場合は、そのプリンシパルの宣言を格納する realms コンテキスト内にサブコンテキストが追加されます。

/realms が削除されると、すべてのレルム定義も削除されます。ただし、外部ソース（表エントリなど）はそのまま残ります。セッション・シェル realm ツールを使用すると、効率的なレルム管理をより安全に実行できます。

realms のサブコンテキストを削除すると、JNDI タイプのレルムに影響する場合があります。RDBMS レルムは、次の表を使用するように定義します。

- JAVA\$HTTP\$REALM\$PRINCIPAL\$ – すべてのプリンシパルとコード化された形式のパスワードを格納します。
- JAVA\$HTTP\$REALM\$GROUP\$ – プリンシパルとグループの関係を格納します。

## リソースの保護

OSE の HTTP セキュリティ・リソースの保護は、サーブレット・コンテキストに対してローカルにあります。保護付きのリソースを宣言するには、情報の 2 つの部分を、保護スキームに組み込んで指定する必要があります。スキームの形式は、次のようになります。

<authType>:<realmName>

次の 2 つの認証タイプが有効です。

- 基本 – 通常の base64 エンコーディングで、あまり安全ではありません。
- ダイジェスト – 両方の当事者がパスワードを自己管理し、高度に暗号化されたコードを渡します。コードは、タイム・スタンプ、要求されている URL、シークレット・キーおよびリクエストの IP などの、多くの状況に固有な値に埋め込まれます。

ダイジェストは基本よりもはるかに安全ですが、一部のブラウザではサポートされません。通常のインストラクションでは、IE5 はダイジェストをサポートしますが、Netscape 4.7 はサポートしません。

リソースを保護しないことを宣言することもできます。これは、サーブレット・コンテキスト・ルートを保護しない場合に役に立ちます。ただし、ルートが保護されると、ツリーの一部となるエラー・ページも保護されます。エラー・ページの送信は、認証プロセスの一部です。エラー・ページが保護されると、サイクルが作成されて、目的の動作は認識されません。

エラー・ページのデフォルトをツリーの一部にするかわりに、エラー・ページを保護しないことを明示的に宣言します。この場合、<NONE> の保護スキームを使用します。たとえば、次のようになります。

```
realm map -s /myService/contexts/myContext -a /system/* -scheme <NONE>
realm map -s /myService/myService/contexts/myContext -a /* -scheme basic:testRealm1
```

## コマンド構造

保護付きのパスは、サーブレット・コンテキストに対してローカルにあります。内部的に、保護されたパスは正規化されて、安定した、予測可能な一致パターンを有効化できます。これにより、内部の表示は、保護スキームを作成するのに使用される元のパスと異なる場合があります。HTTP セキュリティでは、保護ルールを適用する場合、最も長く一致するもの（可能な限り完全一致のもの）が使用されます。

保護スキームを使用したリソースに対するパスの保護は、次のようになります。

```
realm map -s /myService/contexts/myContext -a /doc/index.html -scheme
basic:testRealm1
realm map -s /myService/contexts/myContext -a /doc -scheme basic:testRealm2
realm map -s /myService/contexts/myContext -a /doc/* -scheme basic:testRealm3
```

前述の例のように宣言すると、次の例に示すように、パスはレルムと一致します。

```
/doc/index.html -> testRealm1
/doc/foo -> testRealm3
/doc -> testRealm2
/doc/ -> testRealm2
/doc/index -> testRealm3
```

パスの保護を削除するには、次を入力します。

```
realm map -s /myService/contexts/myContext -r /doc/index.html
```

サーブレット・コンテキスト内のすべての保護付きパスをリストするには、次を入力します。

```
realm map -s /myService/contexts/myContext
```

パスを保護しないことを明示的に宣言するには、次を入力します。

```
realm map -s /myService/contexts/myContext -a /system/* -scheme
<NONE>
```

サーブレット・コンテキスト内のすべての保護付きパスをリストするには、次を入力します。

```
realm map -s /myService/contexts/myContext
```

## 詳細

保護マッピングの JNDI エントリは、サーブレット・コンテキストのサブディレクトリ、*policy* に位置します。そのサブディレクトリ内のサブコンテキストは、エントリ *httpMapping* で、セキュリティ・サーブレットの保護マッピングを処理するオブジェクトを作成します。デフォルトでは、このオブジェクトは `JAVA$HTTP$REALM$MAPPING$` 表の索引として使用されます。HTTP レルムのマッピング表には、すべてのマップされたパスが格納されます。単純な JNDI エントリの操作で、`HttpProtectionMapping` をカスタマイズできます。

## アクセス権の宣言

アクセス権は、ドメインで適用されるエンティティとサブレット・コンテキストで適用されるエンティティを結びつけ、サブレット・コンテキスト自体に常駐しているため、すべての HTTP セキュリティ宣言の中で最も影響力があります。

1つのアクセス権宣言は、次の部分のいくつかで構成されます。

1. サービス
2. 指定されたサービス内のレルム
3. 指定されたサービス内のサブレット・コンテキスト
4. 指定されたレルム内のプリンシパル
5. 適用するアクセス権へのパス
6. アクセス権を付与するか制限するかどうかの判別
7. 割り当てる HTTP アクション

このように1つのアクセス権宣言に結合されているすべての構成部分をリストすると、アクセス権が最も複雑な宣言である理由が簡単にわかります。

前述の項目の中で、HTTP アクションについては説明していません。HTTP セキュリティのアクセス権は、有効な HTTP 要求メソッド、GET、POST、PUT、DELETE、HEAD、TRACE および OPTIONS のみと関連します。

## コマンド構造

GET および POST のために、user1 に対し、/foo/index.html でアクセス権の付与を宣言するには、次を入力します。

```
realm perm -w /myService -realm testRealm1 -s  
/myService/contexts/myContext -n user1 -u /foo/index.html + get,post
```

PUT および DELETE のために、user1 に対し、/foo/\* でアクセス権の制限を宣言するには、次を入力します。

```
realm perm -w /myService -realm testRealm1 -s  
/myService/contexts/myContext -n user1 -u /foo/* - put,delete
```

user1 に対して、/foo/index.html でアクセス権の付与を消去するには、次を入力します。

```
realm perm -w /myService -realm testRealm1 -s  
/myService/contexts/myContext -n user1 -u /foo/index.html +
```

ユーザーのすべてのアクセス権をリストするには、次を入力します。

```
realm perm -w /myService -realm testRealm1 -s  
/myService/contexts/myContext -n user1
```

### 詳細

サーブレット・コンテキストの *policy* サブコンテキストには、*config* エントリがあります。これは、すべてのアクセス権の宣言をチェックするためのオブジェクトを作成する場合に使用されるエントリです。また、オブジェクトはアクセス権テーブル、`JAVA$HTTP$REALM$POLICY$` のキーとしても使用されます。

## セキュリティ・サーブレットの宣言

すべての HTTP セキュリティは、JNDI 名前空間エントリを使用して宣言されます。これは、セキュリティを施行するサーブレットの場合も同じです。サーブレット・コンテキストに、*httpSecurity* という名前の特別なサーブレットがある場合、このサーブレットは、サーブレット・コンテキスト内のすべての要求に対する最初のフィルタとして追加されます。

`PrivilegedServlet` インタフェースが実装されているかぎり、すべてのカスタマイズが許可されます。このサーブレットは、主に次のことを実行します。

- 認識可能なセキュリティ違反がある場合に、その  
`service(HttpRequest.PrivilegedAccess, HttpRequest, HttpResponse)`  
中に `AccessControlException` を呼び出します。  
または
- 要求が許可される場合は、例外を呼びません。

認証または許可が実行された後に、サーブレットは、特定の認証されたプリンシパルの値を要求そのものに設定する必要があります。これは、実行されているすべてのサーブレットによる要求から取得できるユーザー情報です。

## コマンド構造

セキュリティ・サーブレットを作成するには、次を入力します。

```
realm secure -s /myService/contexts/myContext
```

---

---

**注意：** サーブレットは、`named_servlets` ではなく、そのサーブレットのコンテキスト・ディレクトリ内で公開されます。

---

---

## トラブルシューティング

HTTP セキュリティをデバッグするときに絞り込むいくつかの問題の段階があります。次の最低限のチェックリストを使用すると、トラブルシューティングをする場合に役に立ちます。

- （レルム名、ユーザー名、または URI 指定などの）スペルをチェックしてください。
- `DBUSER` レルムを使用している場合は、大 / 小文字をチェックしてください。



- ❑ ブラウザのキャッシュを設定して、ページが最新バージョンになっているか常にチェックしてください。
- ❑ ブラウザのキャッシュを消去してください。
- ❑ Web サーバー・セッション・プロパティの設定後、必ず新しい Web サーバー・セッションをテストしてください。この情報は、現行のアクティブなセッションに伝播されない場合があります。実行しているすべてのブラウザを閉じて、新しいブラウザを起動すると情報は伝搬されます。
- ❑ セキュリティ宣言の 4 つのすべての段階が適切に実行されているか確認してください。どれかが欠けていたり、不適切であると、予測不可能な結果になります。
- ❑ 指定した認証のタイプは、使用しているブラウザでサポートされるか確認してください。たとえば、デフォルトでは、Netscape 4.7 はダイジェスト認証をサポートしません。Netscape では、この認証は単なる基本認証とみなされます（ダイアログ・ボックスを表示）。ただし、基本認証の応答は、ダイジェスト認証には通用しません。これは、実際には不正な理由で表示されているため、通常の Netscape のプロンプトが表示されると誤解を招きます。
- ❑ シェルを使用して、関連するエンティティを問い合せてください。情報がセキュリティの目標を定義する方法で宣言されているか確認してください。  
たとえば、/doc/index.html が、基本認証を使用して myRealm の user1 のみにアクセスできるようにする場合は、次のようになります。
  1. myRealm という名前のレルムがドメイン内に必要です。
  2. レルムに、認識されるパスワードを持つ user1 という名前のユーザーを含める必要があります。
  3. /doc/index.html のマッピング、またはサブレット・コンテキスト内の保護スキーム basic:myRealm へのより一般的なパスが必要です。
  4. サブレット・コンテキストに対して宣言されたセキュリティが必要です。
  5. /doc/index.html（またはより一般的なパス）のユーザー user1 に対して GET 権限を付与するアクセス権が必要です。



---

## PL/SQL サブレットの作成

ここでは、リリース 8.1.7 の新しい機能と変更された機能を説明します。

この付録では、次の項目について説明します。

- [PL/SQL サブレットの概要](#)
- [アプリケーションからのデータベース・アクセス記述子の構成](#)
- [パッケージ DBMS\\_EPGC](#)

## PL/SQL サブレットの概要

Oracle Internet Application Server (iAS) を使用する場合、`mod_plsql` モジュールを使用して、Web 上で PL/SQL ストアド・プロシージャにアクセスします。このモジュールは、ステートレスな PL/SQL プロシージャに対して使用することをお勧めします。ステートレスな PL/SQL プロシージャでは、元のプロシージャ・コールが完了すると、パッケージ変数のトランザクション状態と値は保持されません。

iAS の `mod_ose` モジュールで Oracle Servlet Engine を使用して、PL/SQL ストアド・プロシージャを実行することもできます。このモジュールは、Java サブレットと同じように動作する、ステートフルな PL/SQL プロシージャに対して使用することをお勧めします。ステートフルな PL/SQL プロシージャでは、複数の HTTP 要求に対して状態（パッケージ変数やトランザクション状態など）を保持できます。

---

**関連資料：** Web 上での PL/SQL プロシージャ実行の詳細は、Oracle HTTP Server のマニュアルの「`mod_plsqlno` の使用」を参照してください。

---

## PL/SQL サブレット実行のための `mod_ose` の構成

PL/SQL ストアド・プロシージャをサブレットとして実行するには、ゲートウェイ（埋込み PL/SQL ゲートウェイ）として機能する 1 つのサブレットを最初にロードおよび公開する必要があります。これは、一度のみの操作です。この操作を一度実行すると、コードを変更したり、プロシージャごとにステップをロードおよび公開せずに、Web 上で PL/SQL プロシージャを実行できます。

埋込み PL/SQL ゲートウェイ・サブレットをデータベース・サーバーにロードするには、SQL\*Plus から、`SYS` として接続し、スクリプト `rdbms/admin/initplgs.sql` を実行します。

URL でサブレットにアクセスするには、システムのコマンドラインから、`sess_sh` コマンドを使用して、サブレットを公開します。この操作で仮想パスが登録され、その仮想パスを使用するドキュメントに対するすべての要求は、埋込み PL/SQL ゲートウェイ・サブレットで処理されます。このサブレットは、適切な PL/SQL ストアド・プロシージャを実行します。たとえば、次のようになります。

```
% $ORACLE_HOME/jis/bin/unix/sess_sh -s http://webserver:portnumber -u sys/change_on_install
--Session Shell--
--type "help" at the command line for help message
$ publishservlet -virtualpath pls/* /webdomains/contexts/default plsGateway
SYS:oracle.plsql.web.PLSQLGatewayServlet
```

この例の場合、デフォルトのコンテキストを使用して名前 `plsGateway` でゲートウェイ・サブレットが公開されます。この例では、次のようになります。

- PL/SQL ストアド・プロシージャには、仮想パス /pls を使用してアクセスできます。異なる仮想パスを指定して、それぞれ設定が異なる、サブプレットの複数のインスタンスを設定する場合もあります。
- plsGateway ではなく、別の名前を選択できます。これは、別のサブプレットから要求を転送する場合に使用する名前です。
- SYS: パラメータは、例に示されるとおりに指定する必要があります。これは、実際の Java クラス・ファイルの名前です。

ゲートウェイを使用してストアド・プロシージャにアクセスする場合の URL は、次の中の 1 つになります。

```
http://webserver/pls/dadname/procedurename
http://webserver/pls/dadname/schemaname.procedurename
http://webserver:portnumber/pls/dadname/procedurename
http://webserver/pls/dadname/procedurename?param1=value1&param2=value2
```

これらの URL のプロシージャ名には、PL/SQL プロシージャを指定します。次の項で説明しますが、これらのプロシージャでは、DAD の構成方法に応じて、ステートフルまたはステートレスな実行モデルのいずれかを使用できます。

#### 関連資料：

- DAD 構成パラメータの詳細は、Oracle HTTP Server のマニュアルの「mod\_plsql の使用」に記載されています。
- sess\_sh コマンドの構文は、『Oracle8i Java Tools リファレンス』に記載されています。

## ステートフルな PL/SQL ストアド・プロシージャの作成

通常、PL/SQL ストアド・プロシージャを Web 上で実行する場合、プロシージャが終了するとそのステータスは消去されます。このステータス情報には、アクセスするパッケージ変数の値、トランザクション状態およびテンポラリ表に挿入されたすべての行が含まれます。

いくつかのプロシージャを順番に呼び出す場合（いくつかの異なる HTML フォームを使用する登録プロシージャなど）、この動作を変更することがあります。CGI スタイルのパラメータを使用して情報を 1 つのプロシージャから別のプロシージャへ渡すかわりに、プロセス全体が終了するまでパッケージ変数内に情報を格納できます。登録が成功または失敗した場合に、単一のコミットまたはロールバックを実行できます。

PL/SQL ストアド・プロシージャのコールの間ステータス情報を保つには、次の手順を実行します。

1. 前述したように、埋込み PL/SQL ゲートウェイ・サブプレットを、Oracle Servlet Engine を使用して公開します。この操作を実行する必要があるのは、一度のみです。
2. DAD の stateful 属性を Yes に設定します。デフォルトの値は、No です。値を一度設定すると、この DAD を使用して呼び出されるすべてのパッケージとストアド・プロ

シー ज्याの値もそのまま保持されます。すべての新しい DAD で同じ設定が継承されるように、この属性をグローバル・レベルで設定することもできます。

3. コールの間保持するデータの格納が必要な場合は、いくつかの変数を格納するパッケージを作成します。
4. パッケージ変数にアクセスする 1 つ以上の PL/SQL ストアド・プロシー ज्याを作成し、1 つのトランザクションに含まれる様々な部分を実行して、ステートフルな実行を活用します。
5. すべてのデータを準備すると、処理が成功した場合は明示的にコミットし、処理が失敗した場合はロールバックします。プロシー ज्याが終了するときは暗黙的コミットはありません。例外が呼び出された場合、現行のプロシー ज्या・コールの開始時の状態への暗黙的ロールバックはありませんが、トランザクションはオープンされたままのため、コールの終了時にコミットまたはロールバックが必要になります。

パッケージのステータス情報を明示的に削除するには、DBMS\_SESSION.RESET\_PACKAGE をコールします。この方法を使用すると、ステータス情報のデフォルトの動作を維持しながら、ステートフルなプロシー ज्याのパフォーマンスの利点が得られます。

## アプリケーションからのデータベース・アクセス記述子の構成

Web サーバーを設定して Oracle のストアド・プロシー ज्याを実行する場合は、通常、ブラウザ・インタフェースを使用して、データベース・アクセス記述子 (DAD) を設定します。この操作を自動化するには、パッケージ DBMS\_EPGC 内のプロシー ज्याをコールして、DAD を設定します。次の例は、アプリケーションから DAD 構成を設定または変更する方法を示しています。次の項で、パッケージ DBMS\_EPGC の各プロシー ज्याを説明します。

```
--
-- A sample procedure that configures an embedded gateway
-- instance for the given port number.
--
CREATE OR REPLACE PROCEDURE sample1_init_cfg(port IN PLS_INTEGER) IS
BEGIN

    --
    -- reset instance (port) configuration.
    --
    dbms_epgc.drop_instance(port);
    dbms_epgc.create_instance(port);

    --
    -- set global attributes for the embedded PL/SQL Gateway instance.
    --
    dbms_epgc.set_global_attribute(port, 'defaultdad', 'HR');
    dbms_epgc.set_global_attribute(port, 'adminPath', '/admin/');
    dbms_epgc.set_global_attribute(port, 'stateful', 'Yes');
```

```
--
-- create a database access descriptor (DAD) called APPS
--
dbms_epgc.create_dad(port, 'APPS');
dbms_epgc.set_dad_attribute(port, 'APPS', 'default_page', 'APPS.pkg.home');
dbms_epgc.set_dad_attribute(port, 'APPS', 'document_table', 'APPS.doc_tab');
dbms_epgc.set_dad_attribute(port, 'APPS', 'document_path', 'docs');
dbms_epgc.set_dad_attribute(port, 'APPS', 'upload_as_blob', 'jpeg, gif, txt');
dbms_epgc.set_dad_attribute(port, 'APPS', 'document_proc',
                           'APPS.doc_pkg.process_download');
-- override global setting for stateful attribute
dbms_epgc.set_dad_attribute(port, 'APPS', 'stateful', 'No');

--
-- create a database access descriptor (DAD) called HR.
--
dbms_epgc.create_dad(port, 'HR');
--
dbms_epgc.set_dad_attribute(port, 'HR', 'username', 'scott');
dbms_epgc.set_dad_attribute(port, 'HR', 'password', 'tiger');
dbms_epgc.set_dad_attribute(port, 'HR', 'default_page', 'HR.hello');
dbms_epgc.set_dad_attribute(port, 'HR', 'document_table', 'wpg_new_doctab');
dbms_epgc.set_dad_attribute(port, 'HR', 'document_path', 'docs');
dbms_epgc.set_dad_attribute(port, 'HR', 'upload_as_blob', 'txt');
dbms_epgc.set_dad_attribute(port, 'HR', 'upload_as_long_raw', 'sql');
dbms_epgc.set_dad_attribute(port, 'HR', 'document_proc',
                           'HR.docpkg.process_download');

--
-- Commit the changes.
--
COMMIT;

END;
/
show errors;

--
-- Configure the embedded gateway for port 8080.
--
EXECUTE sample1_init_cfg(8080);
```

これまでに DAD を扱ったことがある場合は、WebDB および OAS で使用されている構成ファイルの構文に似ていることが理解できるかと思います。次のプログラムに示すように、一度の操作でこの情報をインポートできます。

```
-- A sample procedure that configures an embedded gateway
-- using the import method.
--
CREATE OR REPLACE PROCEDURE sample2_init_cfg(port IN PLS_INTEGER) IS
    string VARCHAR2(2000);
BEGIN

    string := '
[PLSQL_GATEWAY]
adminpath = /admin_/
defaultdad = HR
stateful = yes
[DAD_APPS]
DEFAULT_PAGE=APPS.pkg.home
DOCUMENT_PATH=docs
DOCUMENT_PROC=APPS.doc_pkg.process_download
DOCUMENT_TABLE=APPS.doc_tab
STATEFUL=no
UPLOAD_AS_BLOB=jpeg, gif, txt
[DAD_HR]
DEFAULT_PAGE=HR.hello
DOCUMENT_PATH=docs
DOCUMENT_PROC=HR.docpkg.process_download
DOCUMENT_TABLE=wpkg_new_doctab
USERNAME=scott
PASSWORD=tiger
UPLOAD_AS_BLOB=txt
UPLOAD_AS_LONG_RAW=sql
';

    dbms_epgc.drop_instance(port);
    dbms_epgc.create_instance(port);

    dbms_epgc.import(port, string);
    --
    -- Commit the changes.
    --
    COMMIT;

END;
/
show errors;
```



```
--
-- Configure the embedded gateway for port 8080.
--
EXECUTE sample2_init_cfg(8080);
```

**関連資料：** この構成パラメータについては、Oracle HTTP Server のマニュアルの「mod\_plsql の使用」に記載されています。ストアド・プロシージャに mod\_plsql 以外でアクセスする場合は、適用されないキャッシング・パラメータおよび接続プーリング・パラメータもあります。

## パッケージ DBMS\_EPGC

このパッケージを使用すると、Oracle Servlet Engine でデータベース・アクセス記述子 (DAD) を構成できます。

埋込み PL/SQL ゲートウェイは、Oracle データベースに埋め込まれている Oracle Servlet Engine のプラグインとして実行されます。Oracle Servlet Engine の複数のインスタンスがある場合は、各インスタンスは固有のポートで HTTP 要求をリスニングします。各ポートには、埋込み PL/SQL ゲートウェイのインスタンスを対応付けできます。HTTP 要求のポート番号によって、使用される（対応付けられた構成の）埋込み PL/SQL ゲートウェイのインスタンスが判別されます。

埋込み PL/SQL ゲートウェイの各インスタンスは、個別に構成できます。この構成情報を設定および取得するには、DBMS\_EPGC パッケージを使用します。

構成情報は、中間層ではなくデータベースに格納されるため、Oracle HTTP Server から DAD を処理しません。中間層で構成情報を DAD と交換するには、このパッケージの IMPORT/EXPORT プロシージャを使用します。

### セキュリティ・モデル

すべてのユーザーがこのパッケージに対する実行権限を付与されている場合でも、パッケージでは、管理ユーザーのプライベート・リストを保守して、その独自のセキュリティ・チェックが実行されます。これらの管理ユーザーのみがこのパッケージのメソッドをコールできます。SYS および SYSTEM は、デフォルトで常に管理ユーザーになります。GRANT\_ADMIN および REVOKE\_ADMIN プロシージャで、他のデータベース・ユーザーの埋込みゲートウェイ管理権限を制御します。

### トランザクションの動作

すべての操作は、呼び出し側のトランザクション・コンテキストで実行されます。呼び出し側は、インポート、設定または削除などの更新操作をコールした後に明示的にコミットする必要があります。

個別のトランザクション・コンテキストで構成操作を実行するには、自律型 PL/SQL ブロックでこのパッケージのコールをラップします。

## 型

このパッケージのプロシージャでは、パラメータの受渡しに次の型を使用します。

```
TYPE varchar2_table IS TABLE OF VARCHAR2(4000) INDEX BY BINARY_INTEGER;
```

## 例外

このパッケージのプロシージャは、次の例外を呼び出せます。

```
config_error EXCEPTION;
PRAGMA EXCEPTION_INIT(config_error, -20000);
config_error_num CONSTANT PLS_INTEGER := -20000;

user_already_exists EXCEPTION;
PRAGMA EXCEPTION_INIT(user_already_exists, -20001);
user_already_exists_num CONSTANT PLS_INTEGER := -20001;

invalid_port EXCEPTION;
PRAGMA EXCEPTION_INIT(invalid_port, -20002);
invalid_port_num PLS_INTEGER := -20002;

invalid_username EXCEPTION;
PRAGMA EXCEPTION_INIT(invalid_username, -20003);
invalid_username_num PLS_INTEGER := -20003;

not_an_admin EXCEPTION;
PRAGMA EXCEPTION_INIT(not_an_admin, -20004);
not_an_admin_num PLS_INTEGER := -20004;

privilege_error EXCEPTION;
PRAGMA EXCEPTION_INIT(privilege_error, -20005);
privilege_error_num PLS_INTEGER := -20005;

dad_not_found EXCEPTION;
PRAGMA EXCEPTION_INIT(dad_not_found, -20006);
dad_not_found_num PLS_INTEGER := -20006;

invalid_dad_attribute EXCEPTION;
PRAGMA EXCEPTION_INIT(invalid_dad_attribute, -20007);
invalid_dad_attribute_num PLS_INTEGER := -20007;

invalid_global_attribute EXCEPTION;
PRAGMA EXCEPTION_INIT(invalid_global_attribute, -20008);
invalid_global_attribute_num PLS_INTEGER := -20008;
```

```
instance_already_exists EXCEPTION;  
PRAGMA EXCEPTION_INIT(instance_already_exists, -20009);  
instance_already_exists_num PLS_INTEGER := -20009;
```

## サブプログラムの概要

### CREATE\_INSTANCE プロシージャ

ポート番号で識別されるゲートウェイ・インスタンスを作成します。このコールは、インスタンスに属性および権限が構成される前に実行する必要があります。

このインスタンス（ポート）がすでに使用されている場合は、この操作はエラーになります。

バルク構成プロシージャ（IMPORT および EXPORT）は、明示的にインスタンスを作成せずに使用できます。

このルーチンのコール側のユーザーは、このゲートウェイ・インスタンスの管理権限を自動で取得できます。

```
PROCEDURE create_instance(port IN PLS_INTEGER);
```

### DROP\_INSTANCE プロシージャ

ポート番号で識別されるゲートウェイ・インスタンスの構成情報を削除します。インスタンスを削除して再作成した方が、変更するよりも簡単な場合があります。

```
PROCEDURE drop_instance(port IN PLS_INTEGER);
```

### DROP\_ALL\_INSTANCES プロシージャ

データベース内のすべてのゲートウェイ・インスタンスの構成情報を削除します。

このプロシージャのコール側には SYS を指定するか、データベース内のすべてのゲートウェイ・インスタンスに対する管理権限が必要です。

```
PROCEDURE drop_all_instances;
```

## GRANT\_ADMIN プロシージャ

次の API では、ゲートウェイ管理権限をデータベース・ユーザーに付与し、またその権限を取り消します。SYS および SYSTEM ユーザーは、デフォルトでは常に管理ユーザーになります。

ゲートウェイ管理権限をユーザーに付与します。

```
PROCEDURE grant_admin(port IN PLS_INTEGER, username IN VARCHAR2);
```

## REVOKE\_ADMIN プロシージャ

ユーザーのゲートウェイ管理権限を取り消します。

```
PROCEDURE revoke_admin(port IN PLS_INTEGER, username IN VARCHAR2);
```

## GET\_ADMIN\_LIST プロシージャ

SYS および SYSTEM 以外のゲートウェイ管理ユーザーのリストを取得します。そのようなユーザーが存在しない場合、結果はゼロ要素を示す空の表になります。

```
PROCEDURE get_admin_list(port IN PLS_INTEGER,  
                          users OUT NOCOPY VARCHAR2_TABLE);
```

## SET\_GLOBAL\_ATTRIBUTE プロシージャ

すべての DAD に適用されるグローバル属性の値を設定します。属性が特定のポート番号にすでに設定されている場合は、古い値が新しい値で上書きされます。

属性名は大 / 小文字を区別しません。属性値は、UNIX ファイル名を表す場合など、大 / 小文字を区別する場合がありますが、Yes および No などの値は、大 / 小文字を区別しません。

```
PROCEDURE set_global_attribute(port IN PLS_INTEGER,  
                               attrname IN VARCHAR2,  
                               attrvalue IN VARCHAR2);
```

## GET\_GLOBAL\_ATTRIBUTE プロシージャ

グローバル属性の値を取得します。属性が設定されていない場合は、NULL を戻します。属性が無効な場合は、例外を呼び出します。

```
FUNCTION get_global_attribute(port IN PLS_INTEGER,  
                             attrname IN VARCHAR2)  
RETURN VARCHAR2;
```

## DELETE\_GLOBAL\_ATTRIBUTE プロシージャ

グローバル属性を削除します。

```
PROCEDURE delete_global_attribute(port          IN PLS_INTEGER,
                                attrname       IN VARCHAR2);
```

## GET\_ALL\_GLOBAL\_ATTRIBUTES プロシージャ

埋込みゲートウェイ・インスタンスのすべてのグローバル属性および値を取得します。出力は、2つの索引付き表になります。1つの表には属性名が示され、もう1つの表には対応する属性値が示されます。ゲートウェイ・インスタンスにグローバル属性が設定されていない場合、出力配列は空になります。

```
PROCEDURE get_all_global_attributes(port          IN PLS_INTEGER,
                                attrnamearray    OUT NOCOPY VARCHAR2_TABLE,
                                attrvaluearray   OUT NOCOPY VARCHAR2_TABLE);
```

## CREATE\_DAD プロシージャ

属性が設定されていない新しい DAD を作成します。DAD 名は、大 / 小文字を区別しません。指定した名前の DAD がすでに存在する場合は、古い情報が削除されます。

```
PROCEDURE create_dad(port IN PLS_INTEGER, dadname IN VARCHAR2);
```

## DROP\_DAD プロシージャ

ゲートウェイ構成から DAD を削除します。

```
PROCEDURE drop_dad(port IN PLS_INTEGER, dadname IN VARCHAR2);
```

## SET\_DAD\_ATTRIBUTE プロシージャ

DAD（データベース・アクセス記述子）に属性を設定します。DAD が存在しない場合は作成されます。属性の古い値はすべて上書きされます。

DAD 名および DAD 属性名は、大 / 小文字を区別しません。DAD 属性値は、属性によって大 / 小文字を区別する場合があります。

```
PROCEDURE set_dad_attribute(port          IN PLS_INTEGER,
                            dadname       IN VARCHAR2,
                            attrname      IN VARCHAR2,
                            attrvalue     IN VARCHAR2);
```

## 例

```
set_dad_attribute(8080, 'myApp', 'default_page', 'myApp.home');
set_dad_attribute(8080, 'myApp', 'document_path', 'docs');
```

## GET\_DAD\_ATTRIBUTE プロシージャ

DAD 属性の値を取得します。DAD が存在しない場合、または属性が無効な場合は、エラーを呼び出します。属性が設定されていない場合は、NULL を戻します。

```
function get_dad_attribute(port      IN PLS_INTEGER,
                           dadname   IN VARCHAR2,
                           attrname  IN VARCHAR2) return VARCHAR2;
```

## DELETE\_DAD\_ATTRIBUTE プロシージャ

DAD 属性を削除します。

```
PROCEDURE delete_dad_attribute(port      IN PLS_INTEGER,
                               dadname   IN VARCHAR2,
                               attrname  IN VARCHAR2);
```

## GET\_DAD\_LIST プロシージャ

埋込みゲートウェイ・インスタンスのすべての DAD のリストを取得します。DAD が存在しない場合、結果はゼロ要素を示す空の表になります。

```
PROCEDURE get_dad_list(port IN PLS_INTEGER,
                       dadarray OUT NOCOPY VARCHAR2_TABLE);
```

## GET\_ALL\_DAD\_ATTRIBUTES プロシージャ

DAD のすべての属性を取得します。出力は、2 つの索引付き表になります。1 つの表には属性名が示され、もう 1 つの表には対応する属性値が示されます。DAD に属性が設定されていない場合、出力配列は空になります。

```
PROCEDURE get_all_dad_attributes(port      IN PLS_INTEGER,
                                  dadname    IN VARCHAR2,
                                  attrnamearray OUT NOCOPY VARCHAR2_TABLE,
                                  attrvaluearray OUT NOCOPY VARCHAR2_TABLE);
```

## IMPORT プロシージャ

埋込み PL/SQL ゲートウェイの構成情報をバルク・ロードするには、次のプロシージャを使用します。入力は、次のいずれかのフォームで表します。

- VARCHAR2 (最大長 32KB)
- VARCHAR2 変数の索引付き表
- CLOB

構成情報の構文は、iAS の Oracle HTTP Server で使用されている構文と同じにする必要があります。最も簡単に構文を作成するには、既存の DAD から構文をエクスポートします。

```
PROCEDURE import (port IN PLS_INTEGER,
                  cfg IN VARCHAR2);
```

```
PROCEDURE import (port IN PLS_INTEGER,
                  cfg IN DBMS_EPGC.VARCHAR2_TABLE);
```

```
PROCEDURE import (port IN PLS_INTEGER,
                  cfg IN CLOB);
```

## EXPORT プロシージャ

iAS の Oracle HTTP Server で埋込み PL/SQL ゲートウェイの構成情報を使用するには、次のプロシージャで、この構成情報をフラット化されたフォームにエクスポートします。出力は、次のいずれかになります。

- VARCHAR2 (最大長 32KB)
- VARCHAR2 変数の索引付き表
- CLOB

```
PROCEDURE export (port IN PLS_INTEGER,
                  cfg OUT NOCOPY VARCHAR2);
```

```
PROCEDURE export (port IN PLS_INTEGER,
                  cfg OUT NOCOPY dbms_epgc.VARCHAR2_TABLE);
```

```
PROCEDURE export (port IN PLS_INTEGER,
                  cfg OUT NOCOPY CLOB);
```





# B

## 例

この付録では、Oracle8i Oracle Servlet Engine ユーザーズ・ガイドのすべての例に対するリンクを示します。

シングル・ドメイン：URL 内のポート関係とドメイン・ルート 1-7

マルチ・ドメイン：URL 内のポート関係とサービス・ルート 1-8

仮想パスのサーブレット・コンテキストへのマッピング 1-10

HTTP 仮想パスに対応付けられたサーブレット 1-11

コンテンツへのクライアント・アクセスを示す URL 1-11

doc\_root パスの変更 2-4

publishservlet コマンドを使用したサーブレットの公開 2-6

setproperties コマンドを使用した、公開されたサーブレットのプロパティの変更 2-6

サーブレット・コンテキストを検索する HTTP 要求 2-7

サーブレットを検索する HTTP 要求 2-8

defaultervlet で処理される要求 2-8

HTTP 要求 4-4

サーブレットにより処理される要求 4-4

デフォルトのサーブレットにより処理される要求 4-5

サービスの作成とグローバル・タイムアウトの設定 4-8

このシナリオの Apache 接続として、エントリ inst1\_http を定義する tnsnames.ora 5-5

Web ドメイン構造の Apache CFG ファイルへのエクスポート 5-8

exportwebdomain コマンドの出力結果：webdomains.cfg 5-9

OSE へ接続する接続記述子の定義 5-12

OSE により処理されるファイル拡張子を指定する AddHandler 5-12

---

すべての接続要求を指定し、OSE へ送信されるアドレス・セグメントを解析する `SetHandler` 5-12

要求を `mod_ose` から OSE へルーティングする Apache 5-13

静的ファイルを処理する Apache と `mod_ose` 5-13

**winecellar** という名前のサーブレット・コンテキストの作成 7-4

構成とコンテンツを表す **winecellar** のサーブレット・コンテキストのエントリ 7-4

`config` プロパティの参照と変更 7-6

サーブレット・コンテキスト `getAttribute` メソッド 7-7

サーブレットの公開 7-8

新しいサーブレットと仮想パスのマッピングの確認 7-8

サーブレットへのプロパティの追加 7-9

有効なセキュリティ・コマンドをリストする `Realm` コマンドの実行 8-5

## 記号

" : 継続標識, 2-3  
: 継続標識, 2-3  
\>: 継続標識, 2-3

## 数字

404.htm, 4-5

## A

Apache 構成  
tnsnames.ora を使用した, 5-4

## C

context.params, 7-7  
context.properties, 7-6  
cookie, 4-2  
およびタイムアウト, 4-8  
ステートフル・セッション, 4-8

## D

DBMS\_EPGC パッケージ, A-7  
CONFIG\_ERROR 例外, A-8  
CREATE\_DAD プロシージャ, A-11  
CREATE\_INSTANCE プロシージャ, A-9  
DAD\_NOT\_FOUND 例外, A-8  
DELETE\_DAD\_ATTRIBUTE プロシージャ, A-12  
DELETE\_GLOBAL\_ATTRIBUTE プロシージャ,  
A-10  
DROP\_ALL\_INSTANCES プロシージャ, A-9  
DROP\_DAD プロシージャ, A-11

DROP\_INSTANCE プロシージャ, A-9  
EXPORT プロシージャ, A-13  
GET\_ADMIN\_LIST プロシージャ, A-10  
GET\_ALL\_DAD\_ATTRIBUTES プロシージャ,  
A-12  
GET\_ALL\_GLOBAL\_ATTRIBUTES プロシージャ,  
A-11  
GET\_DAD\_ATTRIBUTE プロシージャ, A-12  
GET\_DAD\_LIST プロシージャ, A-12  
GET\_GLOBAL\_ATTRIBUTE プロシージャ, A-10  
GRANT\_ADMIN プロシージャ, A-10  
IMPORT プロシージャ, A-13  
INSTANCE\_ALREADY\_EXISTS 例外, A-8  
INVALID\_DATA\_ATTRIBUTE 例外, A-8  
INVALID\_GLOBAL\_ATTRIBUTE 例外, A-8  
INVALID\_PORT 例外, A-8  
INVALID\_USERNAME 例外, A-8  
NOT\_AN\_ADMIN 例外, A-8  
PRIVILEGE\_ERROR 例外, A-8  
REVOKE\_ADMIN プロシージャ, A-10  
SET\_DAD\_ATTRIBUTE プロシージャ, A-11  
SET\_GLOBAL\_ATTRIBUTE プロシージャ, A-10  
USER\_ALREADY\_EXISTS 例外, A-8  
VARCHAR2\_TABLE 型, A-8  
defaultervlet, 2-8  
エラー・メッセージ, 4-5  
doc\_root オブジェクト, 7-5

## G

get properties  
ls, 2-6  
getSession(true), 4-7

## H

---

http\_sh  
  getproperties  
    サーブレットへのプロパティの追加, 2-6  
  publishservlet -virtualpath, 2-6  
  setproperties, 2-3, 2-6  
httpSecurity, 7-6  
HTTP セキュリティ, 8-1  
  アクセス権のコマンド構造, 8-9  
  アクセス権の宣言, 8-9  
  セキュリティ・サーブレット作成のコマンド構造,  
    8-10  
  セキュリティ・サーブレットの宣言, 8-10  
  トラブルシューティング, 8-10  
  認証と許可, 8-2  
  リソースの保護のコマンド構造, 8-8  
HTTP セッション  
  getSession(true), 4-7  
HTTP 要求  
  サーブレット, 2-7  
  適切なサーブレットの検索, 4-1  
HTTP 要求の処理, 2-7

## J

---

Java  
  概要, vii  
  ドキュメント, viii  
Java Namespace and Directory Interface (JNDI), 1-12  
Java の詳細情報を示す Web サイト, viii  
Java プロパティ・ファイル, 2-3  
Java リファレンス, 2-2  
JDBC  
  Web 情報, viii  
JDK  
  Web の場所, viii  
JLS  
  Web 情報, viii  
JNDI  
  アクセス権, 3-2  
  セッション・シェルの起動, 3-2  
  ナビゲーション, 3-2  
JNDI オブジェクト  
  仮想バス, 1-11  
JNDI およびセッション・シェル, 3-1

## JVM

  Web 情報, viii

## L

---

loadjava, 1-11

## M

---

mod\_ose  
  tnsnames.ora, 5-4  
  フェイルオーバー構成, 5-2, 5-4  
mod\_ose のロード・バランシング, 5-4

## O

---

Oracle8i JVM  
  定義, viii  
Oracle スキーマ, 1-12  
OSE でのサーブレットの基本  
  セッション・シェル, 2-2

## P

---

PL/SQL サーブレット, A-2

## S

---

setgroup, 2-3  
SQLJ  
  ドキュメント, viii

## T

---

tnsnames.ora  
  mod\_ose, 5-4  
transportURL  
  セッション・シェルの起動, 3-3

## U

---

URL, 2-4  
URL の転送, 5-13

## W

---

Web サービス, 1-5  
    エンド・ポイント, 1-5  
    仮想パス, 7-4  
    単純, 1-5  
    マルチホーム, 1-5  
Web ドメイン  
    構成, 3-4  
    説明, 1-6  
    要求を処理するための検索, 4-3, 4-5  
    要求を処理するための検索方法, 2-7  
Web ドメインのデフォルトの構成, 1-6

## あ

---

アクセス権, 1-12  
    所有権, 3-3  
    スキーマ, 1-12  
アクセス権の宣言, 8-9  
アクセス制御, 1-13  
新しいエントリの作成, 3-3  
アルゴリズム  
    サブレットの検索, 4-1

## い

---

インターネット・ニュースグループ, viii

## う

---

埋込み PL/SQL ゲートウェイ, A-2

## え

---

エクスポート・コマンド, 3-6  
エラー・コード  
    404, 4-5  
エラー・メッセージ  
    クライアントへの送信, 4-5  
エンド・ポイント, 1-5  
エントリ  
    新しい, 3-3  
    バインド・コマンド, 3-3  
    変更, 3-3

## か

---

概要  
    制御階層, 1-13  
    タイムアウト, 1-4  
カスタマイズ  
    policy/map オブジェクト, 8-8  
    レルム, 8-5  
仮想パス, 2-4, 7-4  
    JNDI オブジェクト, 1-11  
    OSE での一致, 2-7  
    マッピング, 2-8  
    例, 4-4  
仮想ホスト  
    マルチ・ドメイン, 1-8

## き

---

許可, 8-2

## く

---

クラスのオブジェクト  
    SYS:oracle.aurora.mts.ServletActivation, 7-5, 7-8  
    SYS:oracle.aurora.namespace.filesystem.FSContextImpl, 7-5  
グループ  
    プロパティ, 2-4  
グループ context.error.uris, 7-7  
グループ context.mime, 7-7  
グループ context.params, 7-7  
グループ context.properties, 7-6  
グループ context.servlets, 7-7

## こ

---

コマンド  
    エントリの変更, 3-3  
コンテキスト・グループ, 6-2

## さ

---

サービスの構成, 3-4  
サブレット  
    HTTP 要求, 2-7  
    PL/SQL での作成, A-2

- 検索, 4-1
- デフォルト, 2-8, 4-4
- サーブレット・コンテキスト, 1-12
  - config, 7-5
  - 管理, 3-5
  - 新規作成, 7-3
  - 内部, 7-4
  - 要求を処理するための, 2-7, 4-3
- サーブレットの管理, 3-5
- サーブレットの公開
  - サーブレットの実行, 2-5
- サーブレット・プロパティ
  - servlet.class, 7-9

## し

---

- 所有権, 1-12, 3-3
- シングル・ドメインの Web サービス
  - 要求, 2-7, 4-3

## す

---

- スキーマ
  - Web ドメイン, 1-12
- ステートフルなプロシージャ
  - PL/SQL での作成, A-3

## せ

---

- 静的コンテンツ
  - デフォルトのサーブレット, 2-8
- セキュリティ, 1-13, 8-2
  - トラブルシューティング, 8-10
- セキュリティ HTTP, 8-1
- セキュリティ・サーブレット, 7-6
- セキュリティのカスタマイズ, 8-5, 8-8
- セキュリティのデフォルト, 8-2
- セッション
  - クライアント用に作成される, 4-2
- セッション・シェル, 3-2
  - cd, 7-4
  - createcontext -virtualpath, 7-3, 7-4
  - createcontext コマンド, 7-3
  - destroycontext コマンド, 7-4
  - getgroup, 2-6, 7-9
  - getproperties, 2-3, 7-6, 7-9
    - サーブレットへのプロパティの追加, 2-6

- ls, 7-4, 7-5
- publishservlet -virtualpath, 7-8
- setproperties, 7-9
- ツール, 2-2
- セッション・シェルの起動
  - transportURL, 3-3
- 接続名ファイル
  - tnsnames.ora, 5-4

## た

---

- タイムアウト
  - HTTP セッション, 4-7
  - データベース・セッション, 4-7

## て

---

- データベース, 4-3, 4-5
- データベース・アクセス記述子
  - アプリケーションからの構成, A-3
- データベース・セッション・タイムアウト
  - service.globalTimeout, 4-7
  - タイムアウト, 4-8
- データベースへのサーブレット・クラスのロード, 2-5
- デフォルトのサーブレット, 2-8, 4-4
- デフォルトの設定
  - Web ドメイン, 1-6
  - セキュリティの変更, 8-2

## と

---

- ドキュメント・ルート, 4-3
- トラブルシューティング
  - セキュリティ, 8-10

## に

---

- 認証, 1-13, 8-2

## は

---

- バージョン, 1-14

## ふ

---

- フェイルオーバー構成, 5-2, 5-4
- プロパティ・グループ, 2-4

## ま

---

### マッピング

- 仮想パスの, 2-7
- 仮想パスへの URI, 4-4
- サブレットへの仮想パス, 2-8
- マルチ・ドメインの Web サービス, 2-7
- 要求, 2-7, 4-3

## よ

---

### 要求

- URI, 2-4
- サブレット, 2-7
- サブレット・コンテキスト, 2-7, 4-3
- サブレットの検索, 4-1
- シングル・ドメインの Web サービス, 2-7, 4-3
- マルチ・ドメインの Web サービス, 2-7, 4-3
- 要求のルーティング
  - mod\_ose から, 5-13

## り

---

- リスナーとロード・バランシング, 5-10
- リソースの保護, 8-7

## れ

---

### 例

- Apache 接続として、エントリ inst1\_http を定義する tnsnames.ora, 5-5
- config プロパティの参照と変更, 7-6
- defaultervlet で処理される要求, 2-8
- doc\_root パスの変更, 2-4
- exportwebdomain コマンドの出力結果、webdomains.cfg, 5-9
- getgroup, 7-7, 7-8
- getproperties config, 7-6
- HTTP 仮想パスに対応付けられたサブレット, 1-11
- HTTP 要求, 4-4
  - サブレット・コンテキストの検索, 2-7
- publishervlet コマンドを使用したサブレットの公開, 2-6
- setproperties コマンドを使用した公開されたサブレットのプロパティの変更, 2-6

- Web ドメイン構造の Apache CFG ファイルへのエクスポート, 5-8
- 仮想パスのサブレット・コンテキストへのマッピング, 1-11
- コンテンツへのクライアント・アクセスを示す URL, 1-12
- サービスの作成とグローバル・タイムアウトの設定, 4-8
- サブレットにより処理される要求, 4-4
- サブレットへのプロパティの追加, 7-9
- サブレットを検索する HTTP 要求, 2-8
- シングル・ドメイン - URL 内のポート関係とドメイン・ルート, 1-7
- デフォルトのサブレットにより処理される要求, 4-5
- マルチ・ドメイン - ポート関係とサービス・ルート, 1-8
- 有効なセキュリティ・コマンドをリストする Realm コマンドの実行, 8-5

