管理者ガイド

Sun™ ONE Application Server J2EE CA Service Provider Implementation

Version 7

816-6481-10
2002 年 10 月

目 次

このマニュアルについて

Sun™ ONE Application Server 7 Java 2 Enterprise Edition Connector Architecture (J2EE CA) Service Provider Implementation (SPI) を使用すると、J2EE コネクタをアプリケーションサーバーにプラグインすることができます。このマニュアルでは、SPI をビルド、インストール、設定、および管理する方法について説明します。

この章には次の項目があります。

- 製品ラインの概要
- マニュアルの使用法
- マニュアルの表記規則
- 製品サポート
- マニュアルの構成

製品ラインの概要

Sun ONE Application Server 7 は J2EE 1.3 仕様に準拠したアプリケーションサーバーです。また、新しい Java Web Service 規格や標準の HTTP サーバープログラミング機能もサポートしています。本稼動環境と開発環境の両方に幅広く対応するため、次の 3 種類のアプリケーションサーバーが用意されています。

- Platform Edition
- Standard Edition
- Enterprise Edition

Platform Edition

Platform Edition は、Sun ONE Application Server 7 製品ラインの中核を成す製品です。この製品は無償で提供されます。J2EE 1.3 仕様に準拠した高パフォーマンスで小さな実行時環境を備えているため、基礎的な運用開発に適しています。また、こうした実行時環境をサードパーティアプリケーションに組み込むこともできます。Web サービス対応の Platform Edition には、Sun ONE Web Server や Sun ONE Message Queue で実証済みの技術が組み込まれています。

Platform Edition の配備先は、単一アプリケーションサーバーインスタンスに限られています。このアプリケーションサーバーインスタンスは、Java プラットフォーム 用の仮想マシン、すなわち Java 仮想マシン (JVM™) になります。Platform Edition では、複数層の配備トポロジがサポートされます。ただし、Web サーバー層のプロキシはロードバランスを行いません。さらに、管理ユーティリティを使用できるのは、ローカルクライアントに限定されています。

Platform Edition は、Solaris 9 に統合されています。

Standard Edition

この『入門ガイド』で、対象としているエディションです。Platform Edition の機能に加えて、拡張されたリモート管理機能が追加されています。拡張された管理機能、リモートコマンド行、および Web ベースの管理機能はすべて、Standard Edition に組み込まれています。また、Web サーバー層のプロキシにより Web アプリケーションのトラフィックを分割する機能もあります。Standard Edition では、1 台のマシンに複数のアプリケーションサーバーインスタンス (JVM) を設定できます。

Enterprise Edition

Enterprise Edition では、アプリケーションサーバープラットフォームの基本機能に、高可用性機能、ロードバランス機能、およびクラスタ機能が追加されています。これにより、要求の多い J2EE ベースのアプリケーションもスムーズに配備できます。Standard Edition より高度な管理機能により、複数のインスタンスの配備、複数のマシンへの配備にも対応しています。

クラスタリング機能では、複数のアプリケーションサーバーインスタンスの複製をグループとして構成し、クライアント要求のロードバランスを行うことができます。Enterprise Edition では、外部ロードバランサとロードバランスを行う Web 層ベースのプロキシが両方ともサポートされます。また、HTTP セッション、ステートフルセッション Bean インスタンス、および Java Message Service (JMS) リソースのフェイルオーバー機能があります。「Always On (常時配信)」という独自の高可用性データベーステクノロジを、高可用性 (HA) 持続ストアの基盤として採用しています。

製品の詳細は、Sun Microsystems の Web サイト の Sun ONE Application Server のページを参照してください。

マニュアルの使用法

このマニュアルは、PDF 形式または HTML 形式でも入手できます。次のサイトを参照してください。

`http://docs.sun.com/`

次の表は、Sun ONE Application Server のマニュアルに記述されているタスクと概念を示しています。左側の列にタスクと概念、右側の列に参照するマニュアルを示します。

Sun ONE Application Server マニュアルの概要

情報の内容	参照するマニュアル
ソフトウェアおよびマニュアルの最新情報	リリースノート
サポート対象のプラットフォームと環境	プラットフォーム
アプリケーションサーバーの紹介。アプリケーションサーバーの新機能、評価 (Evaluation) バージョンのインストール、アーキテクチャの概要など	入門ガイド
Sun ONE Application Server とそのコンポーネント (サンプルアプリケーション、管理インタフェース、Sun ONE Message Queue など) のインストール	インストールガイド
Sun ONE Application Server 7 の Java オープンスタンダードモデルに準拠した J2EE アプリケーションの作成方法と実装方法。アプリケーション設計、開発ツール、セキュリティ、アセンブリ、配備、デバッグ、ライフサイクルモジュールの作成に関する情報など	開発者ガイド

Sun ONE Application Server マニュアルの概要 (続き)

情報の内容	参照するマニュアル
Sun ONE Application Server 7 の Web アプリケーション向け Java オープンスタンダードモデルに準拠した J2EE アプリケーションの作成方法と実装方法。Web アプリケーションプログラミングの概念とタスクの説明、サンプルコード、実装のヒント、関連資料の紹介など	Web アプリケーション開発者ガイド
Sun ONE Application Server 7 のエンタープライズ Bean 向け Java オープンスタンダードモデルに準拠した J2EE アプリケーションの作成方法と実装方法。EJB プログラミングの概念とタスクの説明、サンプルコード、実装のヒント、関連資料の紹介など	Enterprise JavaBeans 開発者ガイド
Web サービス、RMI-IIOP、Sun ONE Application Server 7 上の J2EE アプリケーションにアクセスするその他のクライアントの作成方法	Developer's Guide to Clients
JDBC、JNDI、JTS、JMS、JavaMial、リソース、コネクタなどの J2EE 機能	Developer's Guide to J2EE Features and Services
カスタム NSAPI プラグインの作成方法	NSAPI Developer's Guide
次の管理タスクの実行 • 管理インタフェースとコマンド行インタフェースの使用 • サーバーの作業環境の設定 • 管理ドメインの使用 • サーバーインスタンスの使用 • サーバーの稼動状況の監視およびログ記録 • Web サーバープラグインの設定 • Java Messaging Service の設定 • J2EE 機能の使用 • CORBA ベースのクライアント機能の設定 • データベース接続の設定 • トランザクション管理の設定 • Web コンテナの設定 • アプリケーションの配備 • 仮想サーバーの管理	管理者ガイド

Sun ONE Application Server マニュアルの概要 (続き)

情報の内容	参照するマニュアル
サーバー設定ファイルの編集	管理者用設定ファイル リファレンス
Sun ONE Application Server 7 運用環境のセキュリティの設定および管理。一般的なセキュリティ、証明書、および SSL/TLS 暗号化に関する情報など。HTTP サーバーベースのセキュリティについても説明	Administrator's Guide to Security
Sun ONE Application Server 7 用の J2EE CA コネクタのサービスプロバイダ実装の設定と管理。管理ツール、DTD に関する情報やサンプル XML ファイルなど	J2EE CA Service Provider Implementation Administrator's Guide
Netscape Application Server バージョン 2.1 から新しい Sun ONE Application Server 7 プログラミングモデルへのアプリケーションの移行。Sun ONE Application Server に付属するオンラインバンクアプリケーションの移行サンプルなど	移行ガイド
Sun ONE Message Queue の使用法	Sun ONE Message Queue については次の URL を参照 : http://docs.sun.com/db/coll/S1_MessageQueue_30?l=ja

マニュアルの表記規則

この節では、このマニュアルで使用する表記規則について説明します。

- 一般的な表記規則
- ディレクトリの表記規則

一般的な表記規則

このマニュアルは、次の表記規則に従っています。

- **ファイルとディレクトリのパス**は、UNIX の形式で表記します (ディレクトリ名をスラッシュで区切って表記)。Windows バージョンでは、ディレクトリパスについては UNIX と同じですが、ディレクトリの区切り記号にはスラッシュではなく円記号を使用します。
- URL は次の書式で記述します。

  http://*server.domain*/*path*/*file*.html

server はアプリケーションを実行するサーバー名、domain はユーザーのインターネットドメイン名、path はサーバー上のディレクトリの構造、file は個別のファイル名を示します。URL の斜体文字の部分は可変部分です。

- **フォント**は、次のように使い分けます。
  - モノスペースフォントは、サンプルコード、コードの一覧表示、API および言語要素 (関数名、クラス名など)、ファイル名、パス名、ディレクトリ名、および HTML タグに使います。
  - 斜体文字はコード変数に使います。
  - 変数および可変部分、およびリテラルに使われる文字にも斜体を使います。
  - **太字**は、段落の先頭またはリテラルに使われる文字の強調に使います。
- このマニュアルでは、ほとんどのプラットフォームの**インストールルートディレクトリ**を *install_dir* と記述します。例外については、page 13 の「ディレクトリの表記規則」を参照してください。

  デフォルトでは、**ほとんどの**プラットフォームの *install_dir* は次の場所になります
  - Solaris 8 のパッケージベースでない評価バージョンインストール :

    ユーザーのホームディレクトリ `/usn/appserver7`
  - Solaris にバンドルされていない非評価バージョンインストール :

    `/opt/SUNWappserver7`
  - Windows のインストール :

    `C:¥Sun¥AppServer7`

  上記の *default_config_dir* および *install_config_dir* は、*install_dir* と同義です。例外と追加情報については、13 ページの「ディレクトリの表記規則」を参照してください。
- このマニュアルでは、**インスタンスルートディレクトリ**は、*instance_dir* と記述します。これは以下のパスの省略形式です。

  default_config_dir`/domains/`*domain*`/`*instance*
- このマニュアルを通じて、特に明記のない限り、すべての **UNIX 固有の表記**は、Linux オペレーティングシステムにも適用されます。

ディレクトリの表記規則

デフォルトでは、Solaris 8 および 9 のパッケージベースのインストールと Solaris 9 のバンドル版インストールを使用する際、アプリケーションサーバーファイルはいくつかのルートディレクトリに分散してインストールされます。この節では、これらのディレクトリについて説明します。

- **Solaris 9 のバンドル版インストール**のデフォルトのインストールディレクトリは、次のように表記します。
  - *install_dir* は `/usr/appserver/` を表します。このディレクトリには、インストールイメージの静的な部分が格納されます。アプリケーションサーバーを構成するすべてのユーティリティ、実行可能ファイル、およびライブラリがここに格納されます。
  - *default_config_dir* は作成されたドメインのデフォルトの格納先となる `/var/appserver/domains` ディレクトリを表します。
  - *install_config_dir* は `/etc/appserver/config` ディレクトリを表します。このディレクトリには、インストール全体の設定情報が格納されます。たとえば、このインストールのライセンス、管理ドメインのマスターリストなどが格納されます。
- **Solaris 8 および 9 のパッケージベースのバンドルされていない非評価バージョンインストール**のデフォルトのディレクトリは、次のように表記します。
  - *install_dir* は `/opt/SUNWappserver7` ディレクトリを表します。このディレクトリには、インストールイメージの静的な部分が格納されます。アプリケーションサーバーを構成するすべてのユーティリティ、実行可能ファイル、およびライブラリがここに格納されます。
  - *default_config_dir* は 作成されたドメインの格納先となる `/var/opt/SUNWappserver7/domains` ディレクトリを表します。
  - *install_config_dir* は `/etc/opt/SUNWappserver7/config` ディレクトリを表します。このディレクトリには、インストール全体の設定情報が格納されます。たとえば、このインストールのライセンス、管理ドメインのマスターリストなどが格納されます。

製品サポート

ご使用のシステムに問題が発生した場合は、次のいずれかの方法でカスタマサポートにお問い合わせください。

- 次のオンラインサポート Web サイトをご利用ください。

  `http://www.sun.com/supportraining/`

- 保守契約を結んでいるお客様の場合は、専用ダイヤルをご利用ください。

サポートのご依頼の前に、次の情報を用意してください。サポート担当がお客様の問題を解決するために必要な情報です。

- 問題が発生した箇所や動作への影響など、問題の具体的な説明
- マシン機種、OS バージョン、および、問題の原因と思われるパッチやそのほかのソフトウェアなどの製品バージョン
- 問題を再現するための具体的な手順の説明
- エラーログやコアダンプ

マニュアルの構成

『Sun ONE Application Server 管理者ガイド (J2EE CA SPI)』では、Sun ONE Application Server J2EE CA SPI ソフトウェアのすべての側面を理解、設定、および管理するために必要な情報を提供します。

このマニュアルは、次のような内容で構成されています。

- 第 1 章「J2EE Connector Architecture の概要」
- 第 2 章「コネクタのアセンブルと配備」
- 第 3 章「リソースアダプタの管理」
- 第 4 章「サンプルコネクタとサンプルアプリケーションでの使用方法」
- 付録 A「トラブルシューティング」

このマニュアルの最後には、用語集と索引があります。

注	このマニュアルを通じて、特に明記のない限り、すべての UNIX 固有の表記は、Linux オペレーティングシステムにも適用されます。

第 1 章

J2EE Connector Architecture の概要

このマニュアルは、J2EE CA リソースアダプタを配備し、Sun ONE Application Server で使用する必要のある管理者を対象にしています。J2EE CA リソースアダプタのアセンブルおよび配備の方法について説明します。

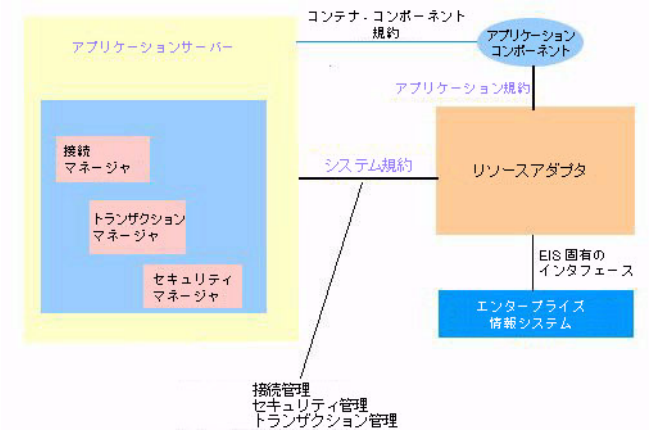
この章では、次の項目について説明します。

- J2EE CA の概要

J2EE CA の概要

J2EE Connector Architecture では、J2EE プラットフォームを異機種の EIS (Enterprise Information System (エンタープライズ情報システム)) に接続するための標準アーキテクチャを定義します。EIS の例には、ERP、メインフレームトランザクション処理、データベースシステム、および Java プログラミング言語以外で記述された旧バージョンのアプリケーションが含まれます。スケーラブルでセキュリティで保護された、トランザクションに基づく一連の機構を定義することにより、J2EE Connector Architecture を使用して EIS をアプリケーションサーバーや企業アプリケーションと統合することができます。

J2EE Connector アーキテクチャを使用すると、EIS ベンダーは EIS の標準リソースアダプタを提供できます。リソースアダプタはアプリケーションサーバーのプラグインになり、EIS、アプリケーションサーバー、および企業アプリケーションの間の接続を可能にします。アプリケーションサーバーのベンダーが J2EE Connector アーキテクチャをサポートするためにシステムを拡張する場合、複数の EIS へのシームレスな接続が約束されます。EIS ベンダーは、J2EE Connector アーキテクチャをサポートするすべてのアプリケーションサーバーにプラグインできる、標準リソースアダプタを 1 つだけ提供する必要があります。アプリケーションサーバーには複数のリソースアダプタをプラグインできます。これにより、アプリケーションサーバー上に配備されたアプリケーションコンポーネントがその下にある EIS にアクセスすることができるようになります。次の図に J2EE Connector アーキテクチャを示します。



アプリケーションサーバー - トランザクション、プール (接続)、セキュリティなど、J2EE CA サーバーサイドの規約を実装しています。

リソースアダプタ (Resource Adapter) - サーバーサイド規約を使ってアプリケーションサーバーに接続します。Sun One Connector Builder を使ってリソースアダプタを生成できます。

エンタープライズ情報システム - システム規約を介してアプリケーションサーバーに接続するバックエンドシステムです。

アプリケーションコンポーネント - EJB、MDB、JSP、サーブレットなどのサーバサイドコンポーネントで、アプリケーションサーバー上で配備、管理、および実行されます。また、Web クライアント層で実行される一方で、アプリケーションサーバーによって Web クライアントに公開されるコンポーネントを指すこともあります。後者のアプリケーションコンポーネントの例には、Java アプレットや DHTML ページがあります。

第 2 章

コネクタのアセンブルと配備

この章では、Sun ONE Application Server コネクタモジュールの内容と、このモジュールをアプリケーション内に個別にまたは一括してアセンブルする方法について説明します。

Sun ONE Application Server コネクタのモジュールには、J2EE 標準の要素と Sun ONE Application Server 固有の要素が組み込まれています。この章では、Sun ONE Application Server 固有の要素についてのみ詳細に説明します。

この章では、次のトピックについて説明します。

- アセンブリと配備の概要
- モジュールとアプリケーションの配備
- コネクタ配備記述子ファイル
- セキュリティ管理
- .rar ファイルを配備するための準備

アセンブリと配備の概要

アプリケーションアセンブリ (パッケージ化とも呼ばれる) は、アプリケーションの個別のコンポーネントを、J2EE に準拠するアプリケーションサーバーに配備できる単位に結合するプロセスです。パッケージは、モジュールまたは独立したアプリケーションとして利用できます。この節には次のトピックがあります。

- モジュール
- J2EE 標準記述子
- Sun ONE Application Server 記述子
- 命名規則
- 実行時環境

- クラスローダー
- サンプルアプリケーション

モジュール

J2EE モジュールは、1 つまたは複数の J2EE コンポーネントの集まりで、各コンポーネントは同一コンテナタイプの 2 つの配備記述子を持っています。一方の配備記述子は J2EE 標準で、もう一方の記述子は Sun ONE Application Server 固有のものです。コネクタモジュールは次のとおりです。

- **リソース RAR ファイル** : J2EE CA コネクタにだけ適用される RAR ファイルです。各 Sun ONE Application Server コネクタには、`sun-ra.xml` ファイルと、J2EE アプリケーション配備記述子の `ra.xml` ファイルがあります。

モジュールを配備した後にクラスローダーが正しいクラスを検索できるように、すべてのモジュールのソースコードでパッケージ定義を使う必要があります。

配備記述子内の情報は宣言型であるため、ソースコードを変更しなくても変更できます。J2EE サーバーは、実行時に読み込んだ配備記述子内の情報に従って動作します。

コネクタモジュールには、Sun ONE Application Server 配備記述子と J2EE 配備記述子があります。Sun ONE Application Server 管理インタフェースは、配備記述子を使って、アプリケーションコンポーネントを配備し、Sun ONE Application Server にリソースを登録します。

アプリケーションは、1 つ以上のモジュール、Sun ONE Application Server 配備記述子、および J2EE アプリケーション配備記述子で構成されます。これらのすべてのモジュールが、Java ARchive (`.jar`) ファイル形式で、拡張子 `.ear` を持つ 1 つのファイルにアセンブルされます。

J2EE 標準記述子

J2EE プラットフォームでは、アセンブリおよび配備機能が提供されます。これらの機能では、コンポーネントおよびアプリケーションの標準パッケージとして JAR ファイルが使われ、パラメータのカスタマイズには XML ベースの配備記述子が使われます。

J2EE 標準配備記述子については、J2EE 仕様書のバージョン 1.3 に規定されています。J2EE 標準配備記述子の詳細は、次の仕様書を参照してください。

- 『Java 2 Enterprise Edition, J2EE Connector Architecture Specification v1.0』の第 10 章「Packaging and Deployment」

仕様書は、次の URL に掲載されています。

http://java.sun.com/products/

Sun ONE Application Server 記述子

Sun ONE Application Server では、Sun ONE Application Server 固有の機能を設定するために追加の配備記述子を使用します。

注 Sun ONE Application Server 配備記述子は、UNIX システム上で 600 のアクセス権限を持っている必要があります。

すべての Sun ONE Application Server 配備記述子用の DTD スキーマファイルは、*install_dir*/lib/dtds ディレクトリにあります。

命名規則

配備されたコネクタの RAR モジュール名は、Sun ONE Application Server 内で一意である必要があります。モジュールのファイル名には、Java パッケージ方式の命名規則の使用をお勧めします。Java パッケージ方式の命名規則を使えば、名前の衝突は発生しません。この命名規則は、Sun ONE Application Server だけでなく、ほかの J2EE アプリケーションサーバーでも使うことをお勧めします。

実行時環境

コンポーネントの配備をスタンドアロンモジュールまたはアプリケーションのどちらとして行っても、配備はファイルシステムおよびサーバーの設定の両方に影響します。詳細は、『Sun ONE Application Server 開発者ガイド (J2EE CA SPI)』を参照してください。

クラスローダー

Java 仮想マシン (JVM) のクラスローダーは、依存関係の解決に必要な Java クラスファイルを動的に読み込みます。たとえば、java.util.Enumeration のインスタンスを作成する場合は、クラスローダーの 1 つが関連するクラスを実行時環境に読み込みます。詳細は、『Sun ONE Application Server 開発者ガイド (J2EE CA SPI)』を参照してください。

サンプルアプリケーション

Sun ONE Application Server には、参照したり配備したりできるサンプルアプリケーションがあり、*install_dir*/samples/j2ee ディレクトリに含まれています。各サンプルには専用のドキュメントが用意されています。

指定したパラメータと一致するコンポーネントファイルを選択します。fileset がサブ要素として含まれている場合、fileset の各ファイルに対して、含んでいる要素の name および contextroot 属性にデフォルト値を使う必要があります。詳細は、次のサイトを参照してください。

http://jakarta.apache.org/ant/manual/CoreTypes/fileset.html

J2EE CA リソースアダプタのアセンブル

この節では、J2EE CA リソースアダプタのアセンブルについて簡単に説明します。

コネクタをアプリケーションサーバーに配備するために次の 2 つの XML コネクタファイルが必要です。

- sun-ra.xml
- ra.xml

ra.xml ファイルは J2EE CA 仕様に基づいており、コネクタにパッケージ化されています。sun-ra.xml ファイルは Sun ONE Application Server 固有の情報を含んでいます。

コネクタ RAR モジュールをアセンブルするには

1. 作業ディレクトリを作成し、モジュールの内容をコピーします。
2. sun-ra.xml および ra.xml という名前の 2 つの配備記述子を META_INF ディレクトリに作成します。
3. 次のコマンドを実行して、RAR ファイルを作成します。

```
jar -cvf module_name.rar *
```

モジュールとアプリケーションの配備

この節では、J2EE のコネクタモジュールを Sun ONE Application Server に配備する方法について説明します。

配備 ID とエラー

アプリケーションまたはモジュールを配備するとき、一意の ID が `server.xml` ファイルに作成されます。この ID は変更しないでください。配備中、サーバーでは ID の衝突を検出し、一意でない ID を持つアプリケーションやモジュールは読み込まれません。この衝突が起こったときには、衝突のメッセージがサーバーログに記録されます。

配備中にエラーが発生すると、アプリケーションやモジュールは配備されません。アプリケーション内のモジュールにエラーがある場合、そのアプリケーション全体が配備されません。

`server.xml` の詳細は、『Sun ONE Application Server 管理者用設定ファイルリファレンス』を参照してください。

配備のライフサイクル

アプリケーションははじめに配備されて、修正および再読み込み、再配備、無効化、再有効化され、最後に配備取り消しされてサーバーから削除されます。この節では、配備のライフサイクルに関連する次のトピックについて説明します。

- 動的配備
- リソースアダプタの無効化
- 動的再読み込み

動的配備

サーバーを再起動せずにアプリケーションまたはモジュールを配備、再配備、および配備解除することができます。これを動的配備と呼びます。

アプリケーションまたはモジュールを再配備するときに、一部のファイルシステムの内容と Application Server の設定が上書きまたは削除されないことがあります。この場合、再配備した後でも古い設定が残ることがあります。クリーンな環境に再配備するには、アプリケーションまたはモジュールを再配備する前に配備取り消しします。また、再配備を行うと、再配備中に実行されていたセッションが無効になります。クライアントはセッションを実行し直す必要があります。

リソースアダプタの無効化

配備されたリソースアダプタをサーバーから削除しないで無効にすることができます。各アプリケーションまたはモジュールの server.xml ファイルには enabled 属性があり、対応するオプションが管理インタフェースにあります。これらのオプションは変更可能です。server.xml の詳細は、『Sun ONE Application Server 管理者用設定ファイルリファレンス』を参照してください。

動的再読み込み

動的再読み込みを有効にすると、コードを変更したときにアプリケーションまたはモジュールを再配備する必要がありません。必要となるのは、変更したクラスファイルをアプリケーションまたはモジュールの配備ディレクトリにコピーすることだけです。サーバーは、定期的に変更を確認して、アプリケーションを変更に合わせて自動的かつ動的に再配備します。

この機能は、変更したコードをすぐにテストできるため、開発環境で役に立ちます。動的再読み込みは、パフォーマンスが低下することがあるので本稼動環境にはお勧めしません。また、再読み込みを行うと、再読み込み中に実行されていたセッションが無効になります。クライアントはセッションを実行し直す必要があります。

アプリケーションの配備記述子ファイルは、自動的には再読み込みされません。配備記述子ファイルを変更した場合、アプリケーションを再配備する必要があります。

動的再読み込みを有効にするには、次のいずれかを行います。

- 管理ツールインタフェースを使用する :
  a. サーバーインスタンスの下にある「アプリケーション」コンポーネントを開きます。
  b. 「アプリケーション」ページに移動します。
  c. 「再読み込みを有効」ボックスをオンにして動的再読み込みを有効にします。
  d. 「再読込のポーリング間隔」フィールドに秒数を入力して、アプリケーションとモジュールにコードの変更がないか確認して動的に再読み込みする間隔を設定します。
  e. 「保存」ボタンをクリックします。
- server.xml ファイルの applications 要素の次の属性を編集する :
  - dynamic-reload-enabled="true" に設定して、動的再読み込みを有効にします。
  - dynamic-reload-poll-interval-in-seconds で、アプリケーションとモジュールにコードの変更がないか確認して動的に再読み込みする間隔を設定します。

  server.xml の詳細は、『Sun ONE Application Server 管理者用設定ファイルリファレンス』を参照してください。

さらに、新しいサーブレットファイルの読み込み、変更に関連する EJB の再読み込み、または配備記述子の変更の再読み込みを行うには、次の操作を行う必要があります。

- 配備されたアプリケーションのルートに .reload という名前の空のファイルを作成します。

  instance_dir/applications/j2ee-apps/*app_name*/.reload

  または個別に配備されたモジュールに作成します。

  instance_dir/applications/j2ee-modules/*module_name*/.reload

- 上記の変更を行うたびに、.reload ファイルのタイムスタンプ (UNIX では touch.reload) を明示的に更新します。

JSP では、sun-web.xml ファイルの jsp-config 要素にある reload-interval プロパティで設定した頻度で、変更が自動的に再読み込みされます。JSP の動的再読み込みを無効にするには、reload-interval="-1" に設定します。

J2EE CA リソースアダプタの配備

コネクタモジュールを配備するには、次のツールを使います。

- asadmin の deploy コマンド
- 管理インタフェース

asadmin の deploy コマンド

asadmin の deploy コマンドは、RAR ファイルを配備します。スタンドアロンのコネクタモジュールを配備するには、--type connector を指定します。RAR スタンドアロンモジュールを配備するには、次のコマンドを使います。

```
asadmin deploy -- connector--instance inst1 -name connector_name rar
filename
```

コネクタモジュールを配備取り消しするには、次のコマンドを使います。

```
asadmin undeploy -- connector--instance inst1 -name connector_name
```

管理インタフェース

管理インタフェースを使ってコネクタモジュールを配備することもできます。

コネクタモジュールを配備するには

1. サーバードメインの下にある「アプリケーション」コンポーネントを開きます。
2. 「コネクタモジュール」ページに移動します。

3. 「配備」ボタンをクリックします。
4. 次の情報を入力します。

   ファイルパス - コネクタモジュールの .rar ファイルへのファイルパス
5. 「了解」をクリックします。
6. コネクタ名を入力します。「了解」をクリックします。

共有フレームワークへのアクセス

J2EE のアプリケーションとモジュールで共有フレームワーククラス (コンポーネント、ライブラリなど) を使用する場合、それらのクラスはアプリケーションやモジュールではなくシステムクラスローダーまたは共有クラスローダーのパスに置くことがきます。サイズが大きい共有ライブラリを、そのライブラリを使用するすべてのモジュールにアセンブルする場合、サーバーへの登録に多くの時間がかかります。また、同一クラスの複数のインスタンスが独自のクラスローダーを使用すると、リソースの浪費になります。

システムクラスローダーについては、19 ページの「クラスローダー」を参照してください。

セキュリティ管理

コネクタは、J2EE セキュリティスキーマ全体に統合されます。ただし、通常 EIS には独自のセキュリティシステムがあるので、J2EE セキュリティ主体を EIS 主体にマップする機能が必要です。J2EE SPI は次の 2 つのメカニズムをサポートしています。

- 主体マッピング
- 設定識別

主体マッピング

主体マッピングは、J2EE 主体と EIS 主体間のマッピングを設定します。このマッピングは、sun-ra.xml に含まれています。詳細とサンプルについては、27 ページの「sun-ra.xml ファイルの要素」を参照してください。

設定識別

設定識別メカニズムを使うと、1 つのユーザー ID ですべての J2EE 主体のバックエンドにログインできるように設定して使うことができます。

J2EE CA は、現在パスワード認証をサポートしています。

コネクタ配備記述子ファイル

Sun ONE Application Server コネクタには次の 2 つの配備記述子ファイルがあります。

- J2EE 標準ファイル (`ra.xml`)。『Java 2 Enterprise Edition, J2EE Connector Architecture Specification v1.0』の第 10 章「Packaging and Deployment」で説明します。
- Sun ONE Application Server 固有のファイル (`sun-ra.xml`)。この節で説明します。

この節では次のトピックについて説明します。

- sun-connector_1_0-0.dtd
- sun-ra.xml ファイルの要素
- サンプルアプリケーション XML ファイル

sun-connector_1_0-0.dtd

`sun-connector_1_0-0.dtd` ファイルには、`sun-ra.xml` ファイルに含めることのできるさまざまな要素と、その要素に含めることができるサブ要素や属性が定義されています。`sun-connector_1_0-0.dtd` ファイルは、`install_dir/lib/dtds` ディレクトリにあります。

注 `sun-connector_1_0-0.dtd` ファイルは編集しないでください。このファイルの内容は、Sun ONE Application Server のバージョンの改訂にともなって変更されます。

注 `sun-connector_1_0-0.dtd` インタフェースは、変更される可能性があります。このような試験的または一時的なインタフェースは、次のリリースで互換性が変わったり、削除されたり、または変更の可能性の少ないインタフェースに置き換えられたりする場合があります。

DTD ファイルおよび XML の全般的な情報については、次のサイトにある XML 仕様書を参照してください。

`http://www.w3.org/TR/REC-xml`

DTD ファイル (およびスキーマを指定するための XML ファイル) の各要素には、次のものを含めることができます。

- サブ要素
- データ
- 属性

サブ要素

要素にはサブ要素を含めることができます。たとえば、次のコードは `sun-connector` 要素を定義しています。

```
<!ELEMENT sun-connector (resource-adapter, role-map?)>
```

`ELEMENT` 行では、`sun-connector` 要素に `(resource-adapter, role-map?)` サブ要素を含めることができると明示しています。

次の表では、サブ要素の終了文字によって決定されるサブ要素の必要指定数について説明しています。左側の列にはサブ要素の終了文字、右側の列には対応する必要指定数を示しています。

サブ要素の必要指定数

サブ要素の終了文字	必要指定数
*	このサブ要素を含まないか、1 個以上含めることができる
?	このサブ要素を含まないか、1 個含めることができる
+	このサブ要素を 1 個以上含まなければならない
なし	このサブ要素を 1 個だけ含まなければならない

要素にほかの要素を含めることができない場合は、カッコで囲まれた要素名のリストの代わりに、`EMPTY` または `(#PCDATA)` が表示されます。

データ

要素の中には、サブ要素の代わりにデータを含むものもあります。これらの要素は、次の形式で定義されます。

`<!ELEMENT` *element-name* `(#PCDATA)>`

次に例を示します。

```
<!ELEMENT description (#PCDATA)>
```

sun-ra.xml ファイルでは、空白はデータ要素内のデータの一部として扱われます。そのため、データ要素で区切られたデータの前後には余分な空白がないようにする必要があります。次に例を示します。

```
<principal user-name="keren"></principal>
```

次の表記規則は、特に明示されている場合を除き、すべての J2EE 配備記述子要素に適用されます。

PCDATA を含む要素では、データの前後にある空白は無視されます。

値が「列挙型」の要素では、値の大文字と小文字が区別されます。

同じ JAR ファイル内のファイルへのパス名を指定する要素では、先頭が「/」で始まらない相対ファイル名は JAR ファイルの名前空間のルートに対して相対であるとみなされます。先頭が「/」で始まる絶対ファイル名も JAR ファイルの名前空間のルートで名前を指定します。通常は、相対名をお勧めします。例外として、.war ファイルについては、サーブレット API との整合性のために絶対名をお勧めします。

属性

ATTLIST 行を持つ要素には属性が含まれています。

sun-ra.xml ファイルの要素

この節では、sun-connector_1_0-0.dtd と sun-ra.xml ファイルにおける次の XML 要素について説明します。

- sun-connector
- resource-adapter
- role-map
- map-element
- principal
- backend-principal
- description
- property

sun-connector

Sun ONE Application Server 固有の sun-connector 要素を定義します。リソースアダプタの配備記述子のルート要素で、sun-ra.xml ファイルに sun-connector 要素は 1 つしか指定できません。

サブ要素

次の表は、sun-connector 要素のサブ要素を説明しています。左側の列にはサブ要素名、中央の列には必要指定数、右側の列には要素の説明を示しています。

sun-connector サブ要素

要素	必要指定数	説明
resource-adapter		jndi-name およびプール設定属性で構成される
role-map		セキュリティマッピング情報を含む

注 JNDI のその他の企業リソースとの名前の衝突や移植の際の問題を回避するため、Sun ONE Application Server のアプリケーションの名前は、すべて文字列 java: comp/env で始めてください。

resource-adapter

プールサイズおよび JNDI - lookup 名を定義します。

属性

次の表は、resource-adapter 要素の属性を説明しています。左側の列には属性名があり、中央の列はデフォルト値を示し、右側の列ではその属性を説明します。

resource adapter 属性

属性	デフォルト値	説明
jndi-name		このリソースアダプタを JNDI 名前空間に表示するための名前 以下の注を参照
max-pool-size	32	EIS への接続の最大数
steady-pool-size	4	保持する接続の最小の数

resource adapter 属性 (続き)

属性	デフォルト値	説明
max-wait-in-millis	10000	接続が利用できない場合、呼出側は接続が作成されるまで、この時間待機しなければならない。値が 0 の場合、接続が利用できないと、例外がスローされる。プールが完全に利用でき、タイマーが時間切れになった場合、アプリケーションに例外が送信される
idle-timeout-in-seconds	1000	タイマースレッドが未使用の接続を定期的に削除する。このパラメータは、このスレッドを実行する間隔を定義する。このスレッドは、タイムアウトになった未使用の接続を削除する

注 ここで記述される jndi-name は、このコネクタを使用するすべてのアプリケーションで jndi 検索名として使用されます。jndi 名は一意である必要があります。たとえば、jndi-name の comet は、jndi 名前空間検索で java:comp/env/eis/comet として参照されます。

サブ要素

次の表は、resource-adapter 要素のサブ要素を説明しています。左側の列にはサブ要素名、中央の列には必要指定数、右側の列には要素の説明を示しています。

resource-adapter サブ要素

要素	必要指定数	説明
description		リソースアダプタを説明します。
property		プロパティを名前と値のペアで提示するための構文を定義します。

description

リソースアダプタを説明します。

サブ要素

なし

property

プロパティを名前と値のペアで提示するための構文を定義します。

サブ要素

なし

属性

次の表は、property 要素の属性を説明しています。左側の列には属性名があり、中央の列はデフォルト値を示し、右側の列ではその属性を説明します。

property 属性

属性	デフォルト値	説明
name		プロパティの名前
value		プロパティの値を含む

role-map

サーブレット /EJB 認証中に受信した主体から EIS の承認した証明書へのマッピングを定義します。このマッピングはオプションです。マップはいくつかの 2 つのタプルからなっています。

サブ要素

次の表は、role-map 要素のサブ要素を説明しています。左側の列にはサブ要素名、中央の列には必要指定数、右側の列には要素の説明を示しています。

role map サブ要素

要素	デフォルト値	説明
description		マップ先となるバックエンドを記述
map-element		主体からバックエンド主体へのマッピングを定義

属性

次の表は、role-map 要素の属性を説明しています。左側の列には属性名があり、中央の列はデフォルト値を示し、右側の列ではその属性を説明します。

role-map 属性

属性	デフォルト値	説明
map-id		マッピングの ID を定義

map-element

map-element を定義します。複数の (サーバー) 主体を同じバックエンド主体にマップすることができます。

サブ要素

次の表は、map-element 要素のサブ要素を説明しています。左側の列にはサブ要素名、中央の列には必要指定数、右側の列には要素の説明を示しています。

map-element サブ要素

要素	デフォルト値	説明
principal		サーブレットおよび EJB クライアントの主体
backend-principal		EIS のバックエンド主体

principal

サーブレットおよび EJB クライアントの主体を定義

サブ要素

次の表は、principal 要素のサブ要素を説明しています。左側の列にはサブ要素名、中央の列には必要指定数、右側の列には要素の説明を示しています。

principal サブ要素

要素	デフォルト値	説明
description		主体を説明

属性

次の表は、principal 要素の属性を説明しています。左側の列には属性名があり、中央の列はデフォルト値を示し、右側の列ではその属性を説明します。

principal 属性

属性	デフォルト値	説明
user-name		ユーザー名を含む

backend-principal

バックエンド EIS 主体を定義

サブ要素

なし

属性

次の表は、backend-principal 要素の属性を説明しています。左側の列には属性名があり、中央の列はデフォルト値を示し、右側の列ではその属性を説明します。

backend-principal 属性

属性	デフォルト値	説明
user-name		バックエンドユーザー名を含む
password		バックエンドパスワードを含む
credential		証明書のタイプを含む

description

リソースアダプタを定義します。

サブ要素

なし

サンプルアプリケーション XML ファイル

sun-ra.xml ファイルは、配備のためにカスタマイズする必要があります。「コード例」には、コードの例、およびカスタマイズの必要な各パラメータやアイテムの説明があります。

この節では次のトピックについて説明します。

- sun-ra.xml ファイルのサンプル
- ra.xml ファイルのサンプル

sun-ra.xml ファイルのサンプル

次のコードは、sun-ra.xml ファイルの例です。

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sun-connector PUBLIC "-//Sun Microsystems, Inc.//DTD Sun
ONE Application Server 7.0 Connector 1.0//EN"
"http://www.sun.com/software/sunone/appserver/dtds/sun-connector_1_
0-0.dtd">

<sun-connector>

    <resource-adapter jndi-name="Comet" max-pool-size="20"
steady-pool-size="10" max-wait-time-in-millis="300000"
idle-timeout-in-seconds="5000">

    </resource-adapter>

    <role-map map-id="mainframe">

         <map-element>

           <principal user-name="keren"></principal>

            <backend-principal user-name="pazit" password="tulip"
credential="credential">

             </backend-principal>

         </map-element>

        </role-map>

    </sun-connector>
```

ra.xml ファイルのサンプル

ra.xml ファイルの例は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>


<!--
   Copyright 2002 Sun Microsystems, Inc. All rights reserved.
 -->


<!DOCTYPE connector PUBLIC '-//Sun Microsystems, Inc.//DTD Connector
1.0//EN' '
http://java.sun.com/dtd/connector_1_0.dtd'>


<connector>
    <display-name>CometResourceAdapter</display-name>
    <vendor-name>sun</vendor-name>
    <spec-version>1.0</spec-version>
    <eis-type>Comet</eis-type>
    <version>1.0</version>
    <resourceadapter>


<managedconnectionfactory-class>samples.connectors.simple.CometMana
gedConnectionFactory</managedconnectionfactory-class>


<connectionfactory-interface>javax.resource.cci.ConnectionFactory</
connectionfactory-interface>


<connectionfactory-impl-class>samples.connectors.simple.CometConnec
tionFactory</connectionfactory-impl-class>


<connection-interface>javax.resource.cci.Connection</connection-int
erface>
```

```
<connection-impl-class>samples.connectors.simple.CometConnection</c
onnection-impl-class>
        <transaction-support>NoTransaction</transaction-support>
        <config-property>
            <config-property-name>Host</config-property-name>

<config-property-type>java.lang.String</config-property-type>

<config-property-value>localhost</config-property-value>
        </config-property>
        <config-property>
            <config-property-name>Port</config-property-name>

<config-property-type>java.lang.String</config-property-type>
            <config-property-value>8020</config-property-value>
        </config-property>

        <authentication-mechanism>

<authentication-mechanism-type>BasicPassword</authentication-mechan
ism-type>

<credential-interface>javax.resource.security.PasswordCredential</c
redential-interface>
        </authentication-mechanism>

        <reauthentication-support>false</reauthentication-support>
    </resourceadapter>
</connector>
```

.rar ファイルを配備するための準備

.rar ファイルは次のものを含む必要があります。

- connector_1-0.dtd ファイルで定義した標準配備情報を含む配備記述子ファイル。

  配備記述子は、META-INF/ra.xml の名前を付けて .rar ファイルに保存する必要があります。

- Sun ONE Application Server コネクタ `sun-ra.xml` ファイル。追加の Application Server 固有の配備情報を指定します。

  `sun-ra.xml` は、META-INF/sun-ra.xml の名前を付けて .rar ファイルに保存する必要があります。

- リソースアダプタに必要な Java インタフェース、実装およびユーティリティクラスは、リソースアダプタモジュールの一部として、1 つまたは複数の .jar ファイルにパッケージ化する必要があります。
- リソースアダプタに必要なプラットフォームに依存するライブラリも、リソースアダプタモジュールでパッケージ化する必要があります。

.rar ファイルのディレクトリ構造

リソースアダプタモジュールのサンプルに含まれているファイルは、次のとおりです。

- /META-INF/ra.xml
- /META-INF/sun-ra.xml
- /readme.html
- /ra.jar
- /client.jar
- /win.dll
- /solaris.so

上の例では、ra.xml は配備記述子であり、sun-ram.xml は Sun 固有の配備記述子です。ra.jar と client.jar には、リソースアダプタの Java インタフェースと実装クラスが含まれています。win.dll と solaris.so は、ネイティブライブラリの例です。

第 3 章

リソースアダプタの管理

この章では、J2EE CA SPI 実装の管理タスクについて説明します。

この章では、次のトピックについて説明します。

- 概要
- 管理タスク

概要

J2EE CA 仕様では、コネクタの配備について規定していますが、リソースアダプタの管理については言及していません。

リソースアダプタを配備した後に、既存のリソースアダプタのパラメータを変更する必要がある場合があります。

管理タスク

プール、設定、セキュリティなどのリソースアダプタのすべてのパラメータを、次のいずれかの方法で変更できます。

- .rar ファイルで ra.xml と sun-ra.xml のいずれかまたは両方のファイルを編集または変更して、リソースアダプタを再配備する
- `<AS_inst_dir>/SUNWappserver7/domains/<domain>/<server>/applications/j2ee_modules/<connector_name>/META-INF` にある配備された ra.xml と sun-ra.xml のいずれかまたは両方のファイルを編集または変更してから、サーバーを再起動する

  詳細は、「コネクタのアセンブルと配備」の「コネクタ配備記述子ファイル」を参照してください。

リソースアダプタの複数のインスタンスの作成

5 つの異なる CICS システムなど、同じタイプのバックエンドシステムが複数ある場合は、バックエンドシステムごとにリソースアダプタを配備する必要があります。

各リソースアダプタには、一意のアプリケーションサーバー名、jndi 名、およびバックエンドに固有の接続パラメータが必要です。

セキュリティの設定

J2EE CA 仕様によると、個人証明書を読み取るために、リソースアダプタにアクセス権が必要です。AS7 には、個人証明書を読み取るためのデフォルトのユーザー、ANONYMOUS (匿名) を認めるデフォルトのサーバー (セキュリティ) ポリシーが定義されています。コンテナ管理によるセキュリティと ANONYMOUS 以外のユーザーでリソースアダプタを使用する予定である場合は、リソースアダプタがコネクタユーザーの個人証明書を読み取ることができるように、`server.policy` ファイルを変更する必要があります。サーバーポリシーの詳細については、『Sun ONE Application Server Developer's Guide (J2EE CA SPI)』を参照してください。

ユーザーの数に基づいて使用できる 2 つのオプションがあります。

- 多数のユーザー
- 限定された数のユーザー

多数のユーザー

次のアクセス権を server.policy ファイルに追加します。

```
grant codeBase
"file:/AS_inst_dir>/SUNWappserver7/domains/<domain>/<server>/
applications/j2ee-modules/<Connector_directory>/-"{

 permission javax.security.auth.PrivateCredentialPermission
"javax.resource.spi.security.PasswordCredential

com.sun.enterprise.security.PrincipalImpl ¥"*¥"", "read";

};
```

このアクセス権は、コネクタコードにだけすべてのユーザーの個人証明書を読み取ることを許可します。

限定された数のユーザー

コネクタのユーザーの数が限られている場合は、次のようにしてアクセス権をそれらのユーザーに限定できます。

すべてのユーザーに対して、次のアクセス権を server.policy ファイルに追加します。

```
grant codeBase
"file:/AS_inst_dir>/SUNWappserver7/domains/<domain>/<server>/
applications/j2ee-modules/<Connector_directory>/-"{

 permission javax.security.auth.PrivateCredentialPermission
"javax.resource.spi.security.PasswordCredential

com.sun.enterprise.security.PrincipalImpl ¥"<user_name>¥"", "read";

};
```

第 4 章

サンプルコネクタとサンプルアプリケーションでの使用方法

この章では、次のトピックについて説明します。

- Comet サンプルアプリケーションの使用法の概要
- サンプルアプリケーションのコンパイルとアセンブル
- サンプルコネクタ
- サンプルアプリケーション

Comet サンプルアプリケーションの使用法の概要

Comet サンプルコネクタは、J2EE CA に準拠するコネクタの配備方法と、J2EE CA に準拠するアプリケーションサーバーでの運用方法に関する説明に使用します。このサンプルには次のコンポーネントがあります。

- サンプル J2EE CA コネクタ
- サンプルアプリケーション
- サンプルバックエンド

サンプルアプリケーションのコンパイルとアセンブル

アプリケーションの再コンパイル、アセンブル、および配備を簡単に行うには、これらのタスクをすばやく行うためのビルド機能の使い方の詳細を記載した「Sun ONE Studio 4,Enterprise Edition for Java」のヘルプから「J2EE アプリケーションアセンブラ」を参照してください。

たとえば、アプリケーション全体を初めから再ビルドするには、次の手順を実行します。

1. アプリケーションのコンパイルおよびアセンブル

   comet/src/ 以下を構築します。

   デフォルトのターゲットコアが実行されて WAR、JAR および EAR ファイルが再ビルドされます。

2. アプリケーションを再配備します。

   comet/src/ の下を配備します。

3. Application Server を再起動します。

   配備記述子を修正した場合は、アプリケーションサーバーの再起動が必要です。サーブレットまたは JSP あるいはその両方の修正した場合は、再起動する必要はありません。

4. サンプルアプリケーションのプロジェクト領域を削除するには、「構築」→「生成物を削除」を実行します。

サンプルコネクタ

`comet.rar` ファイルには、コネクタ仕様アーキテクチャに従って .rar ファイルにアーカイブされたサンプルコネクタファイルが含まれています。ファイルは、`[sun7 インストールディレクトリ ]/sun70/samples/j2ee/connector/assemble/comet.rar` にあります。

サンプルコネクタ (comet.rar) の説明

Comet コネクタサンプルのファイルを次に示します。

- META-INF/ra.xml - 標準配備情報が含まれます。
- comet.jar - JAVA クラスおよびエラーメッセージファイルが含まれます。

- META-INF/sun-ra.xml - 標準的な配備記述子の情報以外の追加の配備情報が含まれます。

META-INF/ra.xml

Comet サンプル ra.xml ファイルには、ホストおよびポートの設定プロパティが含まれます。

それらの値を次のように設定します。

Host: localhost

Port: 8020

これらの値は配備前に修正できます。

次のコードはサンプルの ra.xml ファイルです。

サンプルの ra.xml ファイルの例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>


<!--
   Copyright 2002 Sun Microsystems, Inc. All rights reserved.
 -->


<!DOCTYPE connector PUBLIC '-//Sun Microsystems, Inc.//DTD Connector
1.0//EN' '
http://java.sun.com/dtd/connector_1_0.dtd'>


<connector>
    <display-name>CometResourceAdapter</display-name>
    <vendor-name>sun</vendor-name>
    <spec-version>1.0</spec-version>
    <eis-type>Comet</eis-type>
    <version>1.0</version>
    <resourceadapter>


<managedconnectionfactory-class>samples.connectors.simple.CometMana
gedConnectionFactory</managedconnectionfactory-class>
```

```
<connectionfactory-interface>javax.resource.cci.ConnectionFactory</
connectionfactory-interface>


<connectionfactory-impl-class>samples.connectors.simple.CometConnec
tionFactory</connectionfactory-impl-class>


<connection-interface>javax.resource.cci.Connection</connection-int
erface>


<connection-impl-class>samples.connectors.simple.CometConnection</c
onnection-impl-class>
        <transaction-support>NoTransaction</transaction-support>
        <config-property>
            <config-property-name>Host</config-property-name>


<config-property-type>java.lang.String</config-property-type>


<config-property-value>localhost</config-property-value>
        </config-property>
        <config-property>
            <config-property-name>Port</config-property-name>


<config-property-type>java.lang.String</config-property-type>
            <config-property-value>8020</config-property-value>
        </config-property>


        <authentication-mechanism>


<authentication-mechanism-type>BasicPassword</authentication-mechan
ism-type>
```

```
<credential-interface>javax.resource.security.PasswordCredential</c
redential-interface>
        </authentication-mechanism>
        <reauthentication-support>false</reauthentication-support>
    </resourceadapter>
</connector>
```

comet.jar

comet.jar ファイルには、リソースアダプタおよびエラーメッセージファイルに必要な、Java インタフェース、実装およびユーティリティクラスが含まれます。ファイル samples.connectors.simple.Messages.properties は、メッセージファイルで国際化標準 (I18N) に準拠するために修正されることがあります。

META-INF/sun-ra.xml

sun-ra.xml には、次のような追加の配備情報が含まれます。

- プールの設定
- セキュリティロールマップ
- JNDI 名

プールの設定

次の表に、プール設定属性を示します。リストの左側の列には属性、中央の列にはそのデフォルト値、右側の列には属性の説明があります。

プール属性

属性	デフォルト値	説明
max-pool-size	32	EIS への接続の最大数
steady-pool-size	4	保持する接続の最小の数
max-wait-in-millis	10000	接続が利用できない場合、呼出側は接続が作成されるまで、この時間待機しなければならない。値が 0 の場合、接続が利用できないと例外がスローされる。プールが完全に利用でき、タイマーが時間切れになった場合、アプリケーションに例外が送信される

プール属性

属性	デフォルト値	説明
idle-timeout-in-seconds	1000	タイマースレッドが未使用の接続を定期的に削除する。このパラメータは、このスレッドを実行する間隔を定義する。このスレッドは、タイムアウトになった未使用の接続を削除する

セキュリティロールマップ

セキュリティロールマップ属性は、サーブレット /EJB 認証中に受信した主体から EIS の承認した証明書へのマッピングに使われます。

複数の主体を同じバックエンド主体にマップすることができます。

JNDI 名

jndi-name は、このコネクタを使用するすべてのアプリケーションで jndi 検索名として使用されます。jndi 名は一意である必要があります。たとえば、jndi-name の comet は、jndi 名前空間検索で `java: comp/env/eis/comet` として参照されます。

次のコードはサンプルの `sun-ra.xml` ファイルです。

サンプルの `sun-ra.xml` ファイルのコード例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sun-connector PUBLIC "-//Sun Microsystems, Inc.//DTD Sun
ONE Application Server 7.0 Connector 1.0//EN"
"http://www.sun.com/software/sunone/appserver/dtds/sun-connector_1_
0-0.dtd">

<sun-connector>

    <resource-adapter jndi-name="Comet" max-pool-size="20"
steady-pool-size="10" max-wait-time-in-millis="300000"
idle-timeout-in-seconds="5000">

    </resource-adapter>

    <role-map map-id="mainframe">

         <map-element>

           <principal user-name="keren"></principal>

             <backend-principal user-name="pazit" password="tulip"
credential="credential">

           </backend-principal>
```

```
        </map-element>
      </role-map>
  </sun-connector>
```

サンプルコネクタの配備

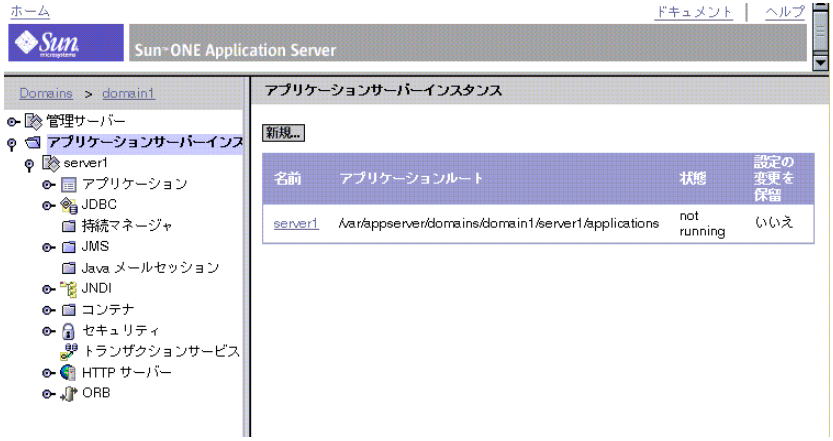
サンプルコネクタをアプリケーションサーバーに配備して、サンプルコネクタを使用できるようにする必要があります。

サンプルコネクタを配備するには

1. 管理ツールを起動します。

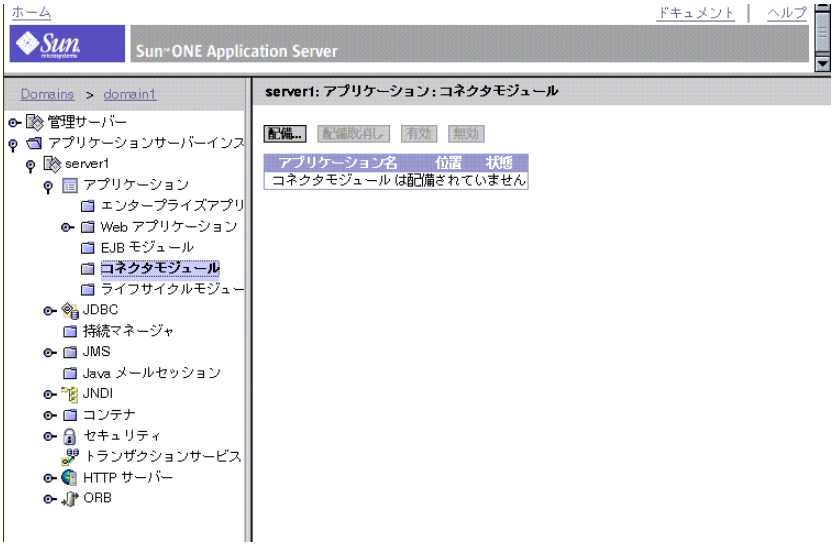
注	デフォルトのインスタンス名は「servler1」です。

次の図が表示されます。



2. 「アプリケーションサーバーインスタンス」-「[インスタンス名]」-「アプリケーション」>「コネクタモジュール」を選択します。

次の図が表示されます。

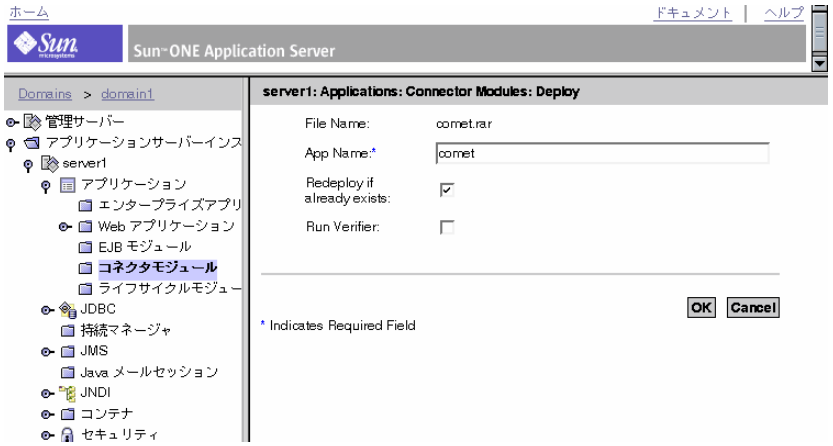


3. 「配備」をクリックします。
4. comet.rar ファイルのある位置を参照し、移動します。

   comet.rar ファイルには、コネクタ仕様アーキテクチャに従って .rar ファイルにアーカイブされたサンプルコネクタファイルが含まれています。

   このディレクトリは、次のとおりです。

   [sun7 インストールディレクトリ]/samples/connectors/simple
5. 「了解」をクリックします。次の図が表示されます。



6. 「アプリケーション名 (App Name:)」に「comet」を入力します。

   「comet」は、アプリケーションがコネクタの検索に使う、コネクタの JNDI 名です。

7. 「了解」をクリックします。

   コネクタが「コネクタモジュール」リストに追加されます。次の図が表示されます。

サンプルアプリケーション

comet サンプルアプリケーションは、バックエンドシステムからメッセージを送受信する comet コネクタを操作します。サンプルアプリケーションとコネクタを使用または操作する前に、それらを配備する必要があります。

サンプルアプリケーションのディレクトリ構造

一般に、どのサンプルアプリケーションも最上位ディレクトリに次の項目があります。

- src ディレクトリ - サンプルアプリケーションのソースコードを含む
- docs - すべてのドキュメントが収められているディレクトリ
- assemble - アプリケーションサーバーに配備する必要のある .EAR ファイルを含む

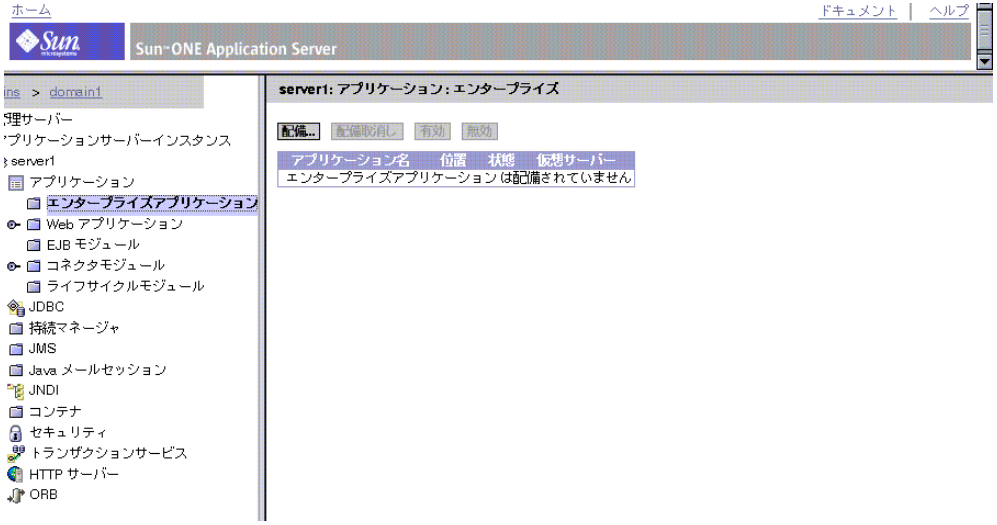
サンプルアプリケーションの配備

次の手順では、アプリケーションの配備方法を説明します。

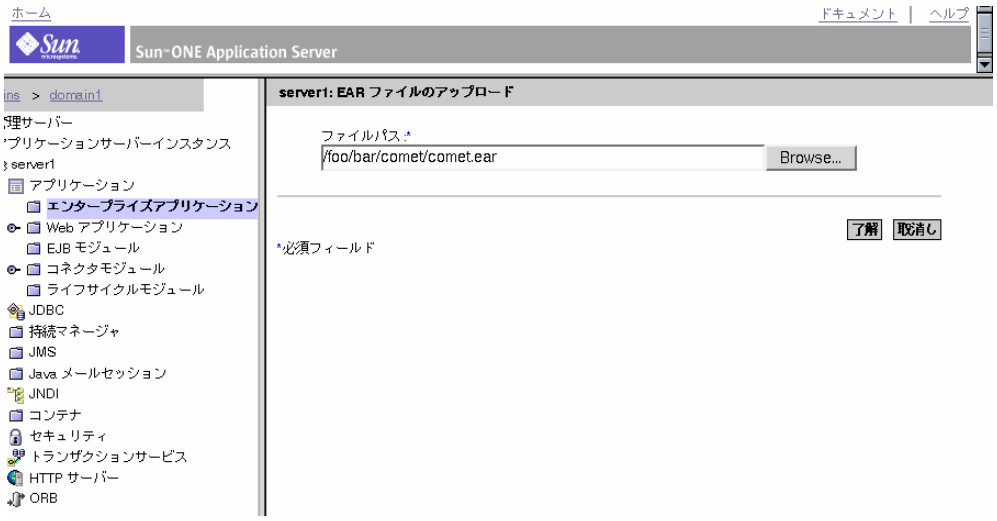
サンプルアプリケーションを配備するには

1. 管理ツールを起動します。
2. 「アプリケーションサーバーインスタンス」-「[インスタンス名]」-「アプリケーション」>「エンタープライズアプリケーション」を選択します。次の図を参照してください。

注　　デフォルトのインスタンス名は「servler1」です。



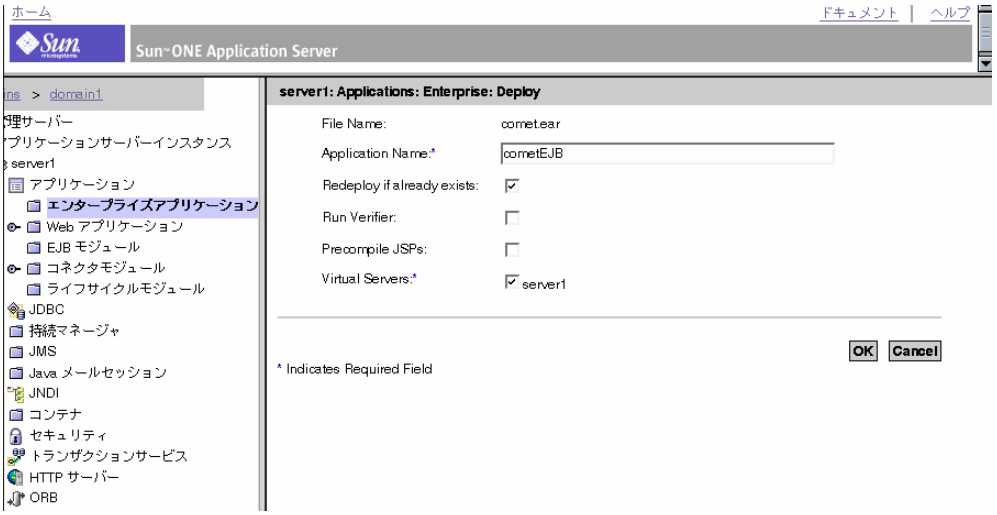
3. 「配備」をクリックします。次の図を参照してください。



4. `comet.ear` ファイルパスに移動します。

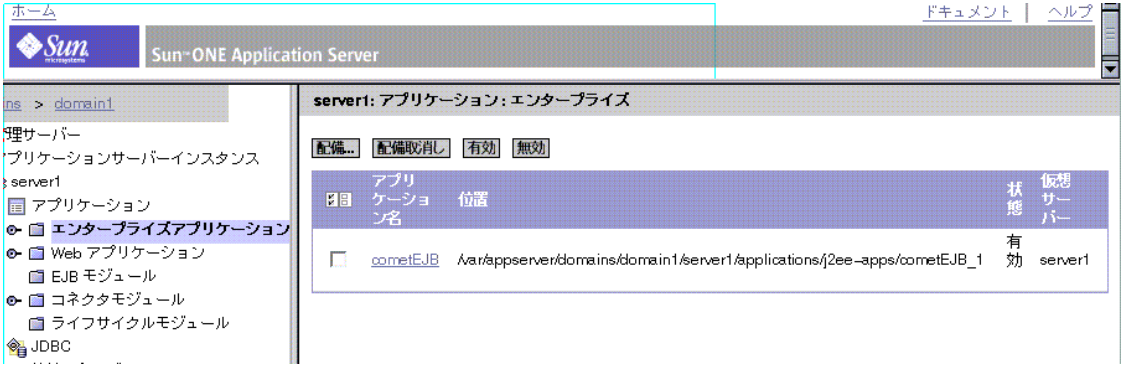
   `comet.ear` ファイルには、.ear ファイルにアーカイブされたサンプルアプリケーションファイルが含まれています。

5. 「了解」をクリックします。次の図が表示されます。



6. 「アプリケーション名」に「CometEJB」を入力して「了解」をクリックします。

   アプリケーションが「エンタープライズアプリケーション」リストに追加されます。次の図を参照してください。



サンプルの操作方法

次の手順では、Comet サンプルの操作方法を説明します。

サンプルを操作するには

1. サンプルバックエンドを実行します。

   バックエンドは、http ポート 8020 を使ってサンプルコネクタからデータを送受信します。

2. `[sun7 インストールディレクトリ ]samples/connectors/simple`

   から次のコマンドを実行します。`java -classpath ./backend.jar Server 8020`

3. アプリケーションサーバーに読み込みます。

   ディレクトリ `[sun7 インストールディレクトリ ]/domains/[ ドメイン名 ]/[ インスタンス名 ]/bin` から `startserv` スクリプトを実行します。

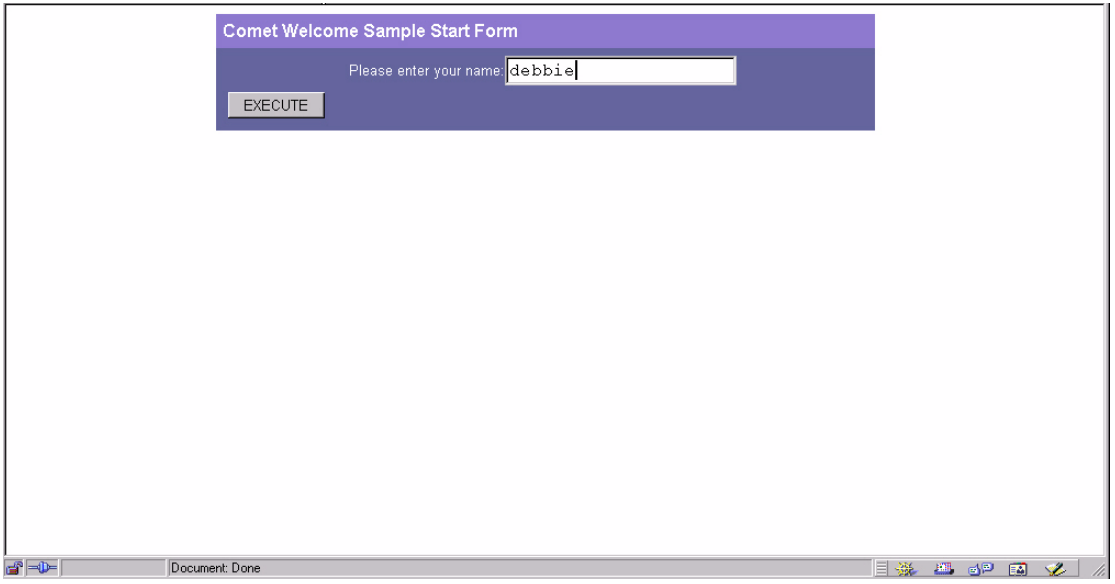
注 デフォルトのインスタンス名は「servler1」です。

4. ブラウザを開いて次のサンプル URL を入力します。

   `http://[ ホスト名 . ドメイン名 ]:[ サーバーポート ]/CometEJB/startForm.jsp`

注 デフォルトのサーバーポートは 1024 です。

次の図が表示されます。



5. ユーザー名を入力し、「EXECUTE」をクリックします。

   「Hello」メッセージが表示されます。

付録 A

トラブルシューティング

この付録では、一般的なエラーメッセージとエラー回避方法を説明します。

コネクタがロガーの作成を試みると、例外がスローされます。
次の例外がスローされます。

```
access denied(java.util.logging.LoggingPermission control)
```

server.policy で必要なアクセス権の基本セットに次の行を追加します。

```
permission java.util.logging.LoggingPermission "control";
```

用語集

この用語集では、Sun ONE Application Server の配備および開発環境を説明するために使われる一般的な用語を定義します。標準 J2EE の用語については、次のサイトにある用語集を参照してください。

```
http://java.sun.com/j2ee/glossary.html
```

ACL　アクセス制御リスト (Access Control List)。Sun ONE Application Server に格納されているリソースにアクセスできるユーザーの ID リストを記録したテキストファイル。「汎用 ACL (general ACL)」も参照。

API　Application Program Interface の略。コンピュータプログラムが、API を解釈するために設計されたほかのソフトウェアまたはハードウェアと通信するために使われる命令の集まり。

Bean 管理によるトランザクション (bean-managed transaction)　エンタープライズ Bean が、開発者が記述したプログラムで制御されるトランザクション境界設定。「コンテナ管理によるトランザクション (container-managed transaction)」も参照。

Bean 管理による持続性 (bean-managed persistence)　エンティティ Bean の変数とデータストアの間で行われるデータ転送。通常、データアクセスロジックは、JDBC (Java Database Connectivity) またはそれ以外のデータアクセステクノロジを使って、開発者によって決定される。「コンテナ管理による持続性 (container-managed persistence)」も参照。

BLOB　Binary Large OBject の略。複合オブジェクトフィールドの格納と取り出しに使うデータ型。BLOB は、画像などのバイナリまたは直列化可能なオブジェクトで、大きなバイト配列に変換された後、コンテナ管理による持続性フィールドに直列化される。

BMP　「Bean 管理による持続性 (bean-managed persistence)」を参照。

BMT　「Bean 管理によるトランザクション (bean-managed transaction)」を参照。

CA　「証明書発行局 (certificate authority)」または「コネクタアーキテクチャ (connector architecture)」を参照。

CKL　Compromised Key List の略。証明書発行局が発行するリスト。クライアントユーザーまたはサーバーユーザーが信頼しなくなった証明書を示す。この場合、鍵は信頼性がなくなっている。「CRL」も参照。

CLI　コマンド行インタフェース (Command-line interface)。ユーザープロンプトで実行型の命令を入力できるインタフェース。「管理インタフェース (Administration interface)」も参照。

CMP　「コンテナ管理による持続性 (container-managed persistence)」を参照。

CMR　「コンテナ管理による関係 (container-managed relationship)」を参照。

CMT　「コンテナ管理によるトランザクション (container-managed transaction)」を参照。

cookie　呼び出し側である Web ブラウザに対して送信され、その後、そのブラウザから呼び出しが行われるたびにブラウザ側に記録される情報の小さなコレクション。サーバーは、cookie によって、同じクライアントからの呼び出しであるかどうかを認識できる。cookie はドメイン特有である。cookie は、アプリケーションとサーバー間の、ほかのデータ交換の場合と同じ Web サーバーセキュリティ機能を利用できる。

CORBA　Common Object Request Broker Architecture の略。オブジェクト指向型分散コンピューティングでの標準的なアーキテクチャ定義。

COSNaming サービス (COSNaming Service)　IIOP ベースのネーミングサービス。

CosNaming プロバイダ (CosNaming provider)　グローバルな JNDI ネームスペースをサポートする (IIOP アプリケーションクライアントにアクセスできる) ために、Sun ONE Application Server には J2EE ベースの CosNaming プロバイダが含まれる。このプロバイダは、CORBA 参照 (リモート EJB 参照) のバインドをサポートする。

CRL　Certificate Revocation List の略。証明書発行局が発行するリスト。クライアントユーザーまたはサーバーユーザーが信頼しなくなった証明書を示す。この場合、証明書は無効になっている。「CRL」も参照。

DataSource オブジェクト (DataSource Object)　実際のデータソースを識別する一連のプロパティを持ったオブジェクト。

DN　識別名 (Distinguished Name)。ディレクトリサーバーのエントリ名を表す文字列。

DN 属性 (DN attribute)　識別名の属性。関連するユーザー、グループ、オブジェクトの識別情報を含むテキスト文字列。

DTD　ドキュメントタイプ定義 (Document Type Definition) 。XML ファイルのクラスの構造とプロパティを記述したもの。

EAR ファイル (EAR file)　Enterprise ARchive ファイル。J2EE アプリケーションを含むアーカイブファイル。EAR ファイルの拡張子は `.ear`。「JAR ファイル (JAR file)」も参照。

EIS　Enterprise Information System の略。EIS は、パッケージ化された企業アプリケーション、トランザクションシステム、またはユーザーアプリケーションと言い換えることができる。通常は、EIS と呼ばれている。EIS の例には、R/3、PeopleSoft、Tuxedo、CICS などがある。

EJB QL　EJB クエリ言語 (EJB Query Language)。コンテナ管理の関係によって定義されるエンティティ Bean のネットワーク上を移動するためのクエリ言語。

EJB コンテナ (EJB container)　「コンテナ (container)」を参照。

EJB テクノロジ (EJB technology)　エンタープライズ Bean は、アプリケーションのビジネスロジックをカプセル化したサーバーサイドコンポーネントである。ビジネスロジックは、アプリケーションの目的をすべて含むコードである。たとえば、在庫管理アプリケーションでは、エンタープライズ Bean はビジネスロジックを `checkInventoryLevel` や `orderProduct` などのメソッドに実装する。これらのメソッドを呼び出すことで、クライアントはアプリケーションが提供する在庫サービスにアクセスできる。「コンテナ (container)」、「エンティティ Bean (entity bean)」、「メッセージ駆動型 Beans (message-driven bean)」、「セッション Bean (session bean)」も参照。

ejbc ユーティリティ (ejbc utility)　エンタープライズ Bean のコンパイラ。すべての EJB クラスとインタフェースが EJB 仕様に合っているかどうかを調べ、スタブとスケルトンを作成する。

ERP　Enterprise Resource Planning の略。企業のリソースの計画をサポートするマルチモジュールのソフトウェアシステム。通常、ERP システムには、購買、在庫、人事、顧客サービス、出荷、資金計画などのビジネスの重要な面を管理するためのリレーショナルデータベースおよびアプリケーションが含まれている。

finder メソッド (finder method)　クライアントがグローバルに利用可能なディレクトリで、Bean または Bean のコレクションを調べることができるようにするメソッド。

FQDN　完全指定のドメイン名 (Fully Qualified Domain Name)。システムの完全指定された名前で、ホスト名とドメイン名の両方を含む。

HTML　Hypertext Markup Language の略。Web ブラウザに表示できるドキュメントを記述するためのマークアップ言語。テキストの各ブロックは、テキストの種類を指定したコードで囲む。

HTML ページ (HTML page)　HTML でコード化され、Web ブラウザで表示することを目的としたページ。

HTTP　HyperText Transfer Protocol の略。リモートホストからハイパーテキストオブジェクトをフェッチするインターネットプロトコル。TCP/IP を基本としている。

HTTP サーブレット　`javax.servlet.HttpServlet` を拡張するサーブレット。HTTP サーブレットには、HTTP プロトコルのサポートが組み込まれている。「汎用サーブレット (generic servlet)」と対照的。

HTTPS　HyperText Transmission Protocol, Secure の略。安全なトランザクション用の HTTP。

IDE　統合開発環境 (Integrated Development Environment)。1 つの使いやすいインタフェースでコードを作成、アセンブル、配備、およびデバッグするためのソフトウェア。

IIOP　Internet Inter-ORB Protocol の略。IIOP 経由の RMI (Remote Method Invocation) と CORBA (Common Object Request Broker Architecture) の両方で使用されるトランスポートレベルプロトコル。

IIOP リスナー (IIOP Listener)　特定のポートで待機して、CORBA ベースのクライアントアプリケーションから送信される接続を受け付ける待機ソケット。

IMAP　インターネットメッセージアクセスプロトコル (Internet Message Access Protocol)。

IP アドレス (IP address)　TCP/IP ネットワーク上のコンピュータまたはその他のデバイスを識別する構造化された数値 ID。IP アドレスの形式は、4 つの数値をピリオドで区切って記述される 32 ビットの数値アドレスである。各数値は 0 ～ 255 の範囲で指定できる。たとえば、123.231.32.2 は IP アドレスにできる。

J2EE　Java 2 Enterprise Edition の略。多層 Web ベースエンタープライズアプリケーションを開発し、配備するための環境。J2EE プラットフォームは、一連のサービス、アプリケーションプログラミングインタフェース (API)、およびこれらのアプリケーションを開発する機能を提供するプロトコルから構成されている。

JAF　JavaBeans Activation Framework の略。MIME データタイプのサポートを Java プラットフォームに統合する。「Mime タイプ」を参照。

JAR ファイル (JAR file)　Java ARchive ファイル。多数のファイルを 1 つのファイルに統合するためのファイル。JAR ファイルの拡張子は `.jar`。

JAR ファイル形式 (JAR file format)　Java ARchive ファイル形式。多数のファイルを 1 つのファイルに統合できるファイル形式で、プラットフォームに依存しない。複数のアプレットと必要なコンポーネント (クラスファイル、イメージ、サウンド、その他のリソースファイル) を JAR ファイルにまとめて、1 回の HTTP トランザクションでブラウザにダウンロードできる。JAR ファイル形式はファイルの圧縮とデジタルシグネチャもサポートしている。

JAR ファイルの規約 (JAR file contract)　エンタープライズ Bean パッケージに含める情報を指定する Java ARchive の規約。

Java IDL　Java インタフェース定義言語 (Java Interface Definition Language)。Java プログラミング言語で記述した API で、Common Object Request Broker Architecture (CORBA) との標準ベースの互換性と接続性を提供する。

JavaBean　移植可能でプラットフォームに依存しない、再利用できるコンポーネントモデル。

JavaMail セッション (JavaMail session)　メールストアとの通信でアプリケーションが使用するオブジェクト。アプリケーションコードは、JNDI 名を使う JavaMail セッションリソースを JNDI サービスを使って特定する。

JAX-RPC　XML ベースのリモートプロシージャ呼び出し用 Java API (Java API for XML-based Remote Procedure Calls)。開発者が、XML ベースの RPC プロトコルに基づいた相互利用可能な Web アプリケーションや Web サービスを作成できるようにする。

JAXM　Java API for XML Messaging の略。アプリケーションが、SOAP 標準を使って、ドキュメント指向の XML メッセージを送受信できるようにする。これらのメッセージにファイルが添付されていても構わない。

JAXP　Java API for XML Processing の略。DOM、SAX、および XSLT を使った XML ドキュメントの処理をサポートしている Java API。アプリケーションが、特定の XML 処理実装に依存せずに、XML ドキュメントを解析および変換できるようにする。

JAXR　Java API for XML Registry の略。さまざまな種類の XML レジストリにアクセスするための、統一された標準の Java API を提供する。ユーザーが、Web サービスを作成、配備、および検索できるようにする。

JDBC　Java Database Connectivity の略。開発者がデータ認識コンポーネントを作成するときに使う、標準ベースの一連のクラスおよびインタフェース。JDBC は、プラットフォームやベンダーとは無関係にデータソースと接続して対話するためのメソッドを実装する。

JDBC 接続プール (JDBC connection pool)　データベースへの接続を指定するための JDBC データソースのプロパティと接続プールのプロパティを組み合わせたプール。

JDBC リソース (JDBC resource)　アプリケーションサーバー上で稼動しているアプリケーションとデータベースを接続するリソースで、既存の JDBC 接続プールを使用する。JNDI 名 (アプリケーション側で使用) と既存の JDBC 接続プールの名前から構成される。

JDK　Java Development Kit の略。Java 2 より前のバージョンの Java プラットフォームに対応したアプリケーションの開発に必要な API やツールを含むソフトウェア。

JMS　Java Message Service の略。JMS クライアントが JMS メッセージサービスの機能にアクセスする方法を定義するインタフェースとセマンティックの標準セット。これらのインタフェースは、Java プログラムによるメッセージの作成、送信、受信、読み込みの標準の方法を提供する。

JMS 管理オブジェクト (JMS-administered object)　1 つまたは複数の JMS クライアントを使用できるように、管理者が作成した設定済みの JMS オブジェクト (接続ファクトリまたは送信先)。

管理オブジェクトを使うことで、プロバイダごとに別の JMS クライアントを使用せずに、同じクライアントを使用できるようになる。管理者はこれらのオブジェクトを JNDI ネームスペースに保存し、JMS クライアントは JNDI ルックアップによってこれらのオブジェクトにアクセスする。

JMS クライアント (JMS client)　JMS メッセージサービスを使ってメッセージを交換する別の JMS クライアントと通信するアプリケーションまたはソフトウェアコンポーネント。

JMS サービス (JMS Service)　JMS クライアントとの接続、メッセージのルーティングと配信、持続性、セキュリティ、ログなど、JMS メッセージシステムの配信サービスを提供するソフトウェア。メッセージサービスは、JMS クライアントのメッセージ送信先、およびメッセージをコンシュームするクライアントに配信されるメッセージの送信元である物理的送信先を維持する。

JMS 接続ファクトリ (JMS connection factory)　JMS クライアントが JMS メッセージサービスとの接続に使用する JMS 管理オブジェクト。

JMS 送信先 (JMS destination)　JMS メッセージに含まれる物理的送信先。生成されたメッセージの、ルーティング先またはコンシューマへの配信先。この物理的送信先は、JMS 管理オブジェクトによって識別され、カプセル化される。JMS クライアントは、プロデュースするメッセージの配信先、コンシュームするメッセージの送信元、またはその両方を決定するときに、この JMS 管理オブジェクトを使用する。

JMS プロバイダ (JMS provider)　メッセージシステム用の JMS インタフェースを実装した製品。完全な製品用に必要な管理機能と制御機能を追加する。

JMS メッセージ (JMS messages)　JMS クライアントがコンシュームする非同期の要求、報告、またはイベント。メッセージにはヘッダーとボディがある (ヘッダーにはフィールドを追加できる)。メッセージヘッダーは、標準フィールドとオプションプロパティを指定する。メッセージボディには、転送するデータが含まれる。

JNDI　Java Naming and Directory Interface の略。企業の複数のネーミングサービスやディレクトリサービスに対する統一インタフェースを Java 技術が使用可能なアプリケーションに提供する、Java プラットフォームの標準拡張。Java Enterprise API セットの一部として、JNDI は、企業の異種ネーミングサービスおよび異種ディレクトリサービスへのシームレスな接続を可能にする。

JNDI 名 (JNDI name)　JNDI ネーミングサービスに登録されているリソースへのアクセスに使用する名前。

JRE　Java 実行時環境 (Java Runtime Environment)。Java 仮想マシンと Java コアクラスに加え、Java プログラミング言語で書かれたアプリケーションの実行時サポートを提供するファイルから構成された、Java Development Kit (JDK) のサブセット。「JDK」も参照。

JSP　JavaServer Pages の略。HTML または XML タグ、JSP タグ、および Java コードを組み合わせて記述したテキストページ。JSP はプログラミング言語の能力と標準ブラウザページのレイアウト機能をあわせ持つ。

jspc ユーティリティ (jspc utility)　JSP のコンパイラ。JSP 仕様に準拠しているかすべての JSP をチェックする。

JTA　Java Transaction API の略。アプリケーションおよび J2EE サーバーによるトランザクションへのアクセスを可能にする API。

JTS　Java Transaction Service の略。トランザクションを処理する Java サービス。

LDAP　Lightweight Directory Access Protocol の略。LDAP は、TCP/IP 上で実行するオープンディレクトリアクセスプロトコルである。グローバルなサイズおよび多数のエントリに拡張できる。アプリケーションサーバーにバンドルされている LDAP サーバーである、Sun ONE Directory Server を使うと、アプリケーションサーバーがネットワーク経由でアクセスできる 1 つの一元化されたディレクトリ情報リポジトリに社内情報をすべて保存できる。

LDIF　LDAP Data Interchange Format の略。Sun ONE Directory Server エントリをテキスト形式で表す形式。

MDB　「メッセージ駆動型 Beans (message-driven bean)」を参照。

MIME データタイプ (MIME Data Type)　MIME (Multi-purpose Internet Mail Extension) タイプを使って、ユーザーのシステムでサポートされるマルチメディアファイルのタイプを制御できる。

NTV　名前 (Name)、タイプ (Type)、値 (Value)。

O/R マッピングツール (O/R mapping tool)　Object-to-relational mapping tool の略。Sun ONE Application Server 管理インタフェースのマッピングツールで、Entity Beans の XML 配備記述子を作成する。

POP3　Post Office Protocol の略。

QOS　QOS (Quality of Service、サービス品質) は、サーバーインスタンス、または仮想サーバーなどに対して設定するパフォーマンスの制限である。たとえば、ISP は、許可する帯域幅に応じて仮想サーバーの課金額を変えたいことがある。この場合、帯域幅の量と接続数に制限を課すことができる。

RAR ファイル (RAR file)　Resource ARchive の略。リソースアダプタを持つ JAR アーカイブ。

RDB　リレーショナルデータベース。

RDBMS　リレーショナルデータベース管理システム。

ResultSet　`java.sql.ResultSet` インタフェースを実装するオブジェクト。`ResultSets` は、データベースまたはほかのソースの表形式データから取得した一連の行のカプセル化に使われる。

RMI　Remote Method Invocation の略。オブジェクトをリモートプロセスに渡せるようにリモートインタフェースを記述するための一連の Java 標準 API。

RMIC　Remote Method Invocation Compiler の略。

RowSet　データベースまたはほかのソースの表形式データから取得した一連の行をカプセル化するオブジェクト。`RowSet` は、`java.sql.ResultSet` インタフェースを拡張して、`ResultSet` が JavaBeans コンポーネントとして機能できるようにする。

RPC　Remote Procedure Call の略。リモートオブジェクトまたはサービスにアクセスするメカニズム。

SAF　Server Application Function の略。要求の処理やその他のサーバーアクティビティに関与する機能。

Secure Socket Layer　「SSL」を参照。

SMTP　Simple Mail Transport Protocol の略。

SNMP　Simple Network Management Protocol の略。ネットワークの稼動状況に関するデータを交換するために使用されるプロトコル。管理対象デバイスとネットワークマネージメントステーション (NMS) 間のデータのやりとりは、SNMP によって行われる。SNMP を使用するすべてのデバイス (ネットワーク上のホスト、ルーター、Web サーバー、その他のサーバーなど) が管理の対象となる。NMS は、そのネットワークのリモート管理を行うマシンである。

SOAP　Simple Object Access Protoco の略。XML ベースのデータ構築と HTTP (Hyper Text Transfer Protocol) の組み合わせを使って、インターネットを介して多様なオペレーション環境に配布されたオブジェクト内のメソッドを呼び出すための標準的な方法を定義している。

SQL　Structured Query Language の略。リレーショナルデータベースアプリケーションで一般的に使用される言語。SQL2 および SQL3 は、この言語のバージョンを表す。

SSL　Secure Sockets Layer の略。インターネットで安全に通信できるようにするためのプロトコル。

Sun ONE Directory Server　Lightweight Directory Access Protocol (LDAP) の Sun ONE バージョン。Sun ONE Application Server の各インスタンスは、Sun ONE Directory Server を使ってユーザーおよびグループに関する情報などの共有サーバー情報を保存する。「LDAP」も参照。

Sun ONE Message Queue　JMS (Java Message Service) オープン標準を実装する Sun ONE エンタープライズメッセージングシステム。JMS プロバイダでもある。

TLS　Transport Layer Security の略。トランスポート層で暗号化と証明書を提供するプロトコル。クライアントおよびサーバーアプリケーションに対して大きな変更を加える必要なく、データをセキュリティの保護されたチャンネル経由で送受信することができる。

UDDI　Universal Description, Discovery, and Integration の略。検索および統合用に、Web サービスのワールドワイドなレジストリを提供する。

URI　Uniform Resource Identifier の略。ドメインの固有リソースを記述する。ローカルではベースディレクトリのサブセットとして記述され、`/ham/burger` はベースディレクトリになり、URI は `toppings/cheese.html` を指定する。対応する URL は、`http://domain:port/toppings/cheese.html` となる。

URL　Uniform Resource Locator の略。HTML ページまたはほかのリソースを一意に指定するアドレス。Web ブラウザは URL を使って、表示するページを指定する。URL では、転送プロトコル (HTTP、FTP など)、ドメイン (`www.my-domain.com` など)、URI (オプション) などを記述する。

WAR ファイル (WAR file)　Web ARchive の略。Web モジュールを含む Java アーカイブ。WAR ファイルの拡張子は `.war`。

Web アプリケーション (web application)　サーブレット、JavaServer Pages、HTML ドキュメント、およびその他の Web リソース (イメージファイル、圧縮アーカイブなどのデータを含む) の集まり。Web アプリケーションは、アーカイブ (WAR ファイル) にパッケージされている場合や、オープンディレクトリ構造に配備されている場合がある。

Sun ONE Application Server では、SHTML や CGI など、Java 以外の Web アプリケーションテクノロジもサポートしている。

Web キャッシュ (web cache)　Sun ONE Application Server の機能の 1 つ。パフォーマンスの向上のため、サーブレットまたは JSP がその結果を指定した一定の時間キャッシュすることを可能にする。その時間内にサーブレットまたは JSP を呼び出すと、キャッシュに保存された結果が返されるので、サーブレットまたは JSP を実行し直す必要がない。

Web コネクタプラグイン (web connector plug-in)　Sun ONE Application Server との通信を可能にする Web サーバーの拡張機能。

Web コンテナ (web container)　「コンテナ (container)」を参照。

Web サーバー (web server)　HTML ページと Web アプリケーションを格納、管理するホスト。完全な J2EE アプリケーションではない。Web サーバーは、Web ブラウザからのユーザーリクエストに応答する。

Web サーバープラグイン (Web Server Plugin)　HTTP リバースプロキシプラグイン。これを使って、ユーザーから Sun ONE Web Server または Sun ONE Application Server に指示を送り、特定の HTTP 要求を別のサーバーへ転送することができる。

Web サービス (web service)　Web 経由で提供されるサービス。インターネットまたはイントラネットを経由してシステムからの要求を受け入れ、それを処理し、応答を返す、完全な自己記述式のモジュラーアプリケーション。

Web モジュール (web module)　個別に配備された Web アプリケーション。「Web アプリケーション (web application)」を参照。

WSDL　Web Service Description Language の略。標準化された方法で Web サービスを定義するために使用される、XML ベースの言語。主に、Web サービスの 3 つの 基本的なプロパティ (Web サービスの定義、Web サービスにアクセスする方法、および Web サービスの場所) を記述する。

XA プロトコル (XA protocol)　分散トランザクション対応のデータベース業界標準プロトコル。

XML　Extensible Markup Language の略。HTML スタイルタグを使って、ドキュメントをフォーマットするだけでなく、ドキュメントで使われるさまざまな種類の情報を識別する。

アクセス制御 (access control)　誰が、どんなアクセス権を持つかを制御することによって、Sun ONE Application Server 製品の安全を確保する方法。

アクティベーション (activation)　エンタープライズ Bean の状態を補助記憶装置からメモリに転送するプロセス。

アセンブリ (assembly)　アプリケーションの個別コンポーネントを配備可能な単位に結合するプロセス。「配備 (deployment)」も参照。

アプリケーション (application)　`.ear` ファイルにパッケージ化されたコンポーネント群。J2EE アプリケーション配備記述子を伴う。「コンポーネント (component)」、「モジュール (module)」も参照。

アプリケーションクライアントコンテナ (application client container)　「コンテナ (container)」を参照。

アプリケーションサーバー (application server)　ビジネスアプリケーションを実行する、信頼性が高く、安全で、スケーラブルなソフトウェアプラットフォーム。通常、アプリケーションサーバーは、コンポーネントのライフサイクル、場所、リソースの分配とトランザクションアクセスなど、高レベルのサービスをアプリケーションに提供する。

アプリケーション層 (application tier)　J2EE アプリケーションの概念的な分割。

クライアント層 : ユーザーインタフェース (UI)。エンドユーザーは、クライアントソフトウェア (Web ブラウザなど) と対話してアプリケーションを使う。

サーバー層 : アプリケーションを構成し、アプリケーションのコンポーネント内で定義されているビジネスロジックおよびプレゼンテーションロジック。

データ層 : アプリケーションがデータソースと対話できるようにするデータアクセスロジック。

アプレット (applet)　Web ブラウザで実行する、Java で書かれた小さなアプリケーション。通常、アプレットは、特別な機能を提供する Web ページに呼び出されたり、埋め込まれたりする。これに対し、サーブレットは、サーバーで実行される小さなアプリケーション。

暗号化 (encryption)　目的の受信者以外が認識できないように情報を変換するプロセス。

委譲 (delegation)　オブジェクトの構成を実装方法として使うオブジェクト指向技術の 1 つ。ある処理の結果に責任を持つオブジェクトが、委譲相手となる別のオブジェクトに実装を任せる。たとえば、クラスローダーは一部のクラスのロードを親に委譲することが多い。

イベント (event)　モジュールまたはアプリケーションからの応答をトリガする名前付きのアクション。

エンティティ Bean (entity bean)　エンタープライズ Bean は、データベースの行などの物理的なデータに関連している。エンティティ Beans は、持続データに結び付けられるので生存期間が長い。エンティティ Beans は、常にトランザクションおよびマルチユーザーを認識する。「メッセージ駆動型 Beans (message-driven bean)」、「読み込み専用 Bean (read-only bean)」、「セッション Bean (session bean)」を参照。

オブジェクトの持続性 (object persistance)　「持続 (persistence)」を参照。

外見 (facade)　アプリケーション固有の、ステートフルセッション Bean を使用してさまざまな Enterprise JavaBeans (EJBs) を管理する状態。

外部 JDNI リソース (resource)　JNDI サービスをリモート JNDI サーバーへの橋渡しとして機能させるリソース。

会話型状態 (conversational state)　同一のクライアントと何度も対話した結果、オブジェクトの状態が変更される状態。「持続状態 (persistent state)」も参照。

仮想サーバー (virtual server)　指定した URL のターゲットとなるコンテンツを処理する仮想 Web サーバー。同じまたは異なるホスト名、ポート番号、または IP アドレスを使って、複数の仮想サーバーがコンテンツを処理できる。HTTP サービスは、受け取った Web 要求を URL に基づいて別の仮想サーバーに送ることができる。仮想ホストとも呼ばれる。

特定の仮想サーバーに Web アプリケーションを割り当てることができる。サーバーインスタンスには、複数の仮想サーバーを持たせることができる。「サーバーインスタンス (server instance)」も参照。

カプセル化 (encapsulate)　モジュールの知識をローカライズすること。オブジェクトはデータと実装をカプセル化するので、サービスを提供するブラックボックスとして表示することができる。インスタンスの変数とメソッドを追加、削除、変更できるが、オブジェクトの提供するサービスが同じであれば、オブジェクトの使用するコードを書き換えずに使い続けることができる。

カラム (column)　データベーステーブル内のフィールド。

監査 (auditing)　エラーやセキュリティ違反などの重大なイベントが発生した場合に、それを後から調べることができるようにイベントを記録するメソッド。

管理インタフェース (Administration interface)　Sun ONE Application Server の設定と管理に使用するブラウザベースの書式の集り。「CLI」も参照。

管理サーバー (administration server)　Sun ONE Application Server の管理機能を担う専用のアプリケーションサーバーインスタンス。管理機能には、配備、ブラウザベースの管理、コマンド行インタフェース (CLI) と統合開発環境 (IDE) からのアクセスなどがある。

管理情報ベース (MIB)　ツリーに似た構造を持ち、マスター SNMP エージェントからアクセス可能な変数を定義する。HTTP サーバのネットワーク設定、状態、および統計へのアクセスが MIB によって提供される。これらの情報は、SNMP を使用して、ネットワークマネージメントステーション (NMS) から取得できる。「ネットワーク管理ステーション (NMS)」および「SNMP」も参照。

管理ドメイン (administrative domain)　Sun ONE Application Server の機能の 1 つ。複数の管理ドメインに対応することで、複数の管理ユーザーのそれぞれが専用のドメインを作成、管理できる。ドメインは、1 つのシステムにインストールされた共通バイナリファイルセットから作成されるインスタンスの集り。

キーペアファイル (key-pair file)　「信頼データベース (trust database)」を参照。

キャッシュされた行セット (cached rowset)　`CachedRowSet` オブジェクトを使うと、データソースからデータを取り込み、そのデータを確認したり変更したりしながらデータソースから切り離すことができる。キャッシュされた行セットには、取得した元のデータ、およびアプリケーションによるデータの変更の両方が記録される。アプリケーションが元のデータソースを更新しようとすると、行セットはデータソースに再び接続され、変更された行だけがデータベースにマージされる。

キャッシュ制御指令 (Cache Control Directives)　プロキシサーバーにどの情報をキャッシュさせるかを制御する Sun ONE Application Server の機能。キャッシュ制御指令を使うことで、プロキシによるデフォルトのキャッシングがオーバーライドされ、機密情報をキャッシュっせずに後から検索することができる。この指令を利用するには、プロキシサーバーが HTTP 1.1 に準拠している必要がある。

キュー (queue)　管理者が作成するオブジェクトで、ポイントツーポイント配信モデルが実装される。キューは、メッセージをコンシュームするクライアントが非活性化されている状態でも、メッセージを保持する。キューは、プロデューサとコンシューマの間のメッセージの保管場所として機能する。

行 (row)　テーブル内の各列の値を格納する 1 つのデータレコード。

クライアント規約 (client contract)　クライアントと EJB コンテナ間の通信ルールを決め、Enterprise Bean を使うアプリケーションのために均一な開発モデルを設定し、クライアントとの関係を統一することによって Bean を効率よく再利用できるように保証する規約。

クライアント認証 (client authentication)　クライアントの証明書を認証するプロセス。このプロセスでは暗号を使用して、証明書の署名と証明書チェーンが信頼できる CA のリストに載っている CA からのものであることを検証する。「認証 (authentication)」、「証明書発行局 (certificate authority)」も参照。

クラスパス (classpath)　Java クラスが格納されるディレクトリと JAR ファイルを識別するパス。「クラスローダー (classloader)」も参照。

クラスローダー (classloader)　特定のルールに従って Java クラスを読み込む機能を果たす Java コンポーネント。「クラスパス (classpath)」も参照。

グループ (group)　何らかの関連があるユーザーの集まり。通常、グループのメンバーシップはローカルシステム管理者が管理する。「ユーザー (user)」、「ロール (role)」を参照。

グローバルデータベースコネクション (global database connection)　複数のコンポーネントに対して利用可能なデータベースコネクション。データベースコネクションにはリソースマネージャが必要。

グローバルトランザクション (global transaction)　トランザクションマネージャによって管理および調整され、1 つのデータベースおよびプロセスに制限されないトランザクション。トランザクションマネージャは通常、XA プロトコルを使ってデータベースのバックエンドと対話する。「ローカルトランザクション (local transaction)」を参照。

公開鍵暗号法 (public key cryptography)　各ユーザーが公開鍵と秘密鍵を持つ暗号法。メッセージは受信者の公開鍵を使って暗号化され、受信者は秘密鍵を使ってメッセージを復号化する。この方法では、秘密鍵はユーザー以外に秘密鍵を知らせる必要がない。

コネクタ (connector)　EIS への接続を提供するコンテナ用の標準拡張メカニズム。コネクタは、EIS に固有のもので、EIS 接続用のリソースアダプタおよびアプリケーション開発ツールから構成されている。リソースアダプタは、コネクタアーキテクチャに定義されたシステムレベル規約を使ってコンテナへ接続される。

コネクタアーキテクチャ (connector architecture) J2EE アプリケーションと EIS を統合するためのアーキテクチャ。このアーキテクチャには、EIS ベンダー提供のリソースアダプタと、このリソースアダプタの接続を許可する J2EE サーバーという 2 つの部分がある。このアーキテクチャは、トランザクション、セキュリティ、リソース管理など、リソースアダプタが J2EE サーバーに接続するために必要な規約を定義している。

コミットする (commit) 必要なコマンドをデータベースに送信することによって、トランザクションを実行すること。「ロールバック (rollback)」、「トランザクション (transaction)」を参照。

コンテナ (container) 特定のタイプの J2EE コンポーネントにライフサイクル管理、セキュリティ、配備、実行時サービスを提供するエンティティ。Sun ONE Application Server には Web コンテナと EJB コンテナがあり、アプリケーションクライアントコンテナをサポートしている。「コンポーネント (component)」も参照。

コンテナ管理による関係 (container-managed relationship) クラスペアで表される、一方の動作が他方の動作に影響を与えるようなフィールドの関係

コンテナ管理によるトランザクション (container-managed transaction) Enterprise JavaBean のトランザクション境界設定を EJB コンテナが自動的に宣言して制御する。「Bean 管理によるトランザクション (bean-managed transaction)」も参照。

コンテナ管理による持続性 (container-managed persistence) EJB コンテナがエンティティ Bean の持続性を管理している状態。エンティティ Bean の変数とデータストアの間のデータ転送で、データアクセスロジックが Sun ONE Application Server によって決定される。「Bean 管理による持続性 (bean-managed persistence)」も参照。

コントロール記述子 (control descriptor) Enterprise Bean トランザクションおよびセキュリティロパティだけでなく、Bean メソッドの個々のプロパティオーバーライド (オプション) を指定できるようにする一連の Enterprise Bean 設定エントリ。

コンパイル済みコマンド (prepared command) 実行の繰り返しを効率よくするために、SQL で書かれた、あらかじめコンパイルされているデータベースコマンド。コンパイル済みコマンドにはパラメータを入れることができる。コンパイル済みステートメントには、1 つまたは複数のコンパイル済みコマンドが含まれている。

コンパイル済みステートメント (prepared statement) `QUERY`、`UPDATE`、または `INSERT` ステートメントをカプセル化したクラスで、データをフェッチするために繰り返し使用される。コンパイル済みステートメントには、1 つまたは複数のコンパイル済みコマンドが含まれている。

コンポーネント (component) Web アプリケーション、Enterprise JavaBean、メッセージ駆動型 Bean、アプリケーションクライアント、またはコネクタ。「アプリケーション (application)」、「モジュール (module)」も参照。

コンポーネント規約 (component contract)　Enterprise JavaBean とそのコンテナ間の関係を確立する規約。

サーバーインスタンス (server instance)　Sun ONE Application Server では、同じマシンの同じインストールに複数のインスタンスを持つことができる。各インスタンスには、それぞれに専用のディレクトリ構造、設定、配備アプリケーションがある。各インスタンスに複数の仮想サーバーを持たせることもできる。「仮想サーバー (virtual server)」も参照。

サーブレット　サーブレットクラスのインスタンス。サーブレットは、サーバーで実行する再利用可能なアプリケーションである。Sun ONE Application Server では、サーブレットは、プレゼンテーションロジックの実行、ビジネスロジックの起動、およびプレゼンテーションレイアウトの起動または実行によって、アプリケーションでの対話ごとにセントラルディスパッチャとしての役割を果たす。

サーブレットエンジン (servlet engine)　すべてのサーブレットメタファンクションを処理する内部オブジェクト。インスタンス化および実行などのサービスをサーブレットに提供する一連のプロセス。

サーブレットランナー (servlet runner)　リクエストオブジェクトおよびレスポンスオブジェクトを持つサーブレットを起動するサーブレットエンジンの一部。「サーブレットエンジン (servlet engine)」を参照。

細分レベル (granularity level)　アプリケーションを細分化するアプローチ。細分度が高いとは、アプリケーションが細かく定義された多数の Enterprise JavaBeans (EJBs) に分割されていることを示す。細分度が低いとは、アプリケーションの分割数が少なく、大きなプログラムが生成されていることを示す。

再利用可能なコンポーネント (reusable component)　複数の容量、たとえば複数のリソースまたはアプリケーションが使えるように作成されたコンポーネント。

識別名 (Distinguished Name)　「DN」、「DN 属性 (DN attribute)」を参照。

システム管理者 (system administrator)　Sun ONE Application Server ソフトウェアを管理し、Sun ONE Application Server アプリケーションを配備する人。

持続 (persistence)　エンタープライズ Bean で、インスタンス変数と基礎となるデータベースとの間でエンティティ Beans の状態を転送するプロトコル。「トランジエンス (transience)」とは反対の概念。セッションでは、セッションのストレージメカニズムを意味する。

持続状態 (persistent state)　オブジェクトの状態が持続ストレージ (通常はデータベース) に保存されている状態。

持続性マネージャ (persistance manager)　コンテナにインストールされたエンティティ Bean の持続性に対する責任を持っているエンティティ。

実行時システム (runtime system)　プログラムを実行するソフトウェア環境。実行時システムには、Java プログラミング言語で記述したプログラムのロード、ネイティブメソッドへの動的リンク、メモリー管理、例外処理に必要なコードがすべて含まれている。Java 仮想マシンの実装も含まれており、Java インタプリタになることもある。

主キー (primary key)　クライアントを特定のエンティティ Bean に配備する一意の識別子。

主キークラス名 (primary key class name)　Bean の主キーの完全修飾クラス名を指定する変数。JNDI 検索に使われる。

主体 (principal)　認証の結果として、エンティティに割り当てられる ID。

状態 (state)　1. 指定された時間におけるエンティティの環境または状態。2. Sun ONE Application Server 機能インタフェース `IState2` を使って、アプリケーションの状態を保存する分散データ保存メカニズム。「会話型状態 (conversational state)」、「持続状態 (persistent state)」も参照。

承認 (authorization)　メソッドまたはリソースへのアクセスを決定するプロセス。J2EE プラットフォームでの承認では、承認を必要とする要求に関連するユーザーが、そのセキュリティロールに含まれているかどうかが検証される。たとえば、人事管理アプリケーションでは、管理者には社員全員の個人情報を見ることを承認し、社員には自身の個人情報だけを見ることを承認する。

証明書 (certificate)　個人や企業などのエンティティの名前を指定するデジタルデータ。証明書に含まれる公開鍵がそのエンティティのものであることを証明する。クライアントとサーバーの両方が証明書を持つことができる。

証明書発行局 (certificate authority)　インターネットを通じて証明書を発行する企業。または、企業のイントラネットまたはエクストラネットの証明書の発行を担当する部門。

シングルサインオン (single sign-on)　1 つの仮想サーバーインスタンスの複数の J2EE アプリケーションでユーザーの認証状態を共有している状態。

信頼データベース (trust database)　公開鍵と秘密鍵を含むセキュリティファイル。キーペアファイル (key-pair file) とも呼ばれる。

スキーマ (schema)　基礎となるデータベースの構造で、テーブル名、カラムの種類、索引情報、主キーと外部キーの関係情報が含まれる。

スティッキー cookie (sticky cookie)　常に同じサーバープロセスにクライアントを強制的に接続させるためにクライアントに返される cookie。「セッション cookie (session cookie)」も参照。

ステートフルセッション Bean (stateful session bean)　特定のクライアントとのセッションを表すセッション Beans で、複数のクライアント起動メソッドのステートを自動的に管理する。

ステートレスセッション Bean (stateless session bean)　状態のないサービスを表すセッション Bean。状態のないセッション Bean は、完全にトランジェントであり、特定のクライアントが限られた時間必要とするビジネスロジックの一時的な部分がカプセル化される。

ストアドプロシージャ (stored procedure)　SQL で書かれ、データベースに保存されるステートメントのブロック。ストアドプロシージャを使って、レコードの変更、挿入、または削除などのすべてのタイプのデータベースオペレーションを実行できる。ストアドプロシージャを使うと、ネットワークを介して送信される情報量が減るのでデータベースのパフォーマンスが向上する。

ストリーミング (streaming)　HTTP によるデータの通信方法を管理するための技術。結果がストリーミングされると、そのデータの最初の部分をすぐに利用できる。結果がストリーミングされないと、結果全体が取得されるまで利用できない。ストリーミングを使うと、大量のデータを効率よく返すことができるため、アプリケーションの体感的なパフォーマンスが向上する。

スレッド (thread)　プロセス内部の実行シーケンス。プロセスで複数のスレッドが同時に実行される場合はマルチスレッド。各スレッドが逐次実行される場合はシングルスレッド。

生成メソッド (create method)　Enterprise Bean を作成時にカスタマイズするメソッド。

セキュリティ (security)　認証されたクライアントだけがアプリケーションリソースにアクセスできるようにしたスクリーニングメカニズム。

セッション Bean (session bean)　クライアントによって作成されるエンタープライズ Bean。通常は、1 回のクライアントサーバーセッションの間だけ存在する。セッション Bean は、クライアントのために計算や他の EJB へのアクセスなどを実行する。セッション Bean はトランザクションで使用されることもあるが、システムがクラッシュした場合に復元できない。セッション Bean オブジェクトにはステートレス (特定のクライアントに関連付けられない)、およびステートフル (特定のクライアントと関連付けられる) があり、メソッドやトランザクションの間で対話状態を保持できる。「ステートフルセッション Bean (stateful session bean)」、「ステートレスセッション Bean (stateless session bean)」も参照。

セッション cookie (session cookie)　ユーザーセッション識別子が含まれているクライアントに返される cookie。「スティッキー cookie (sticky cookie)」も参照。

セッション (session)　サーブレットが複数の HTTP リクエストでのユーザーと Web アプリケーションとの対話を追跡するために使用するオブジェクト。

セッションタイムアウト (session timeout)　ユーザーセッションの有効期限。この特定の時間を超えると、Sun ONE Application Server によってユーザーセッションが無効になる。「セッション (session)」を参照。

接続プール (Connection Pool)　物理的な接続をキャッシュおよび再利用することで、データベースへのアクセスを効率的にする方法。接続によるオーバーヘッドを回避し、多数のスレッド間で共有する接続を少数に抑えることができる。「JDBC 接続プール (JDBC connection pool)」も参照。

接続ファクトリ (connection factory)　J2EE コンポーネントがリソースにアクセスできるよう、接続オブジェクトを生成するオブジェクト。提供された JMS 実装をアプリケーションコードが使えるようにする JMS 接続 (TopicConnection または QueueConnection) の作成に使用される。アプリケーションコードは、JNDI 名を使う接続ファクトリを JNDI サービスを使って特定する。

設定 (configuration)　サーバーを調整する、またはコンポーネントのメタデータを提供するプロセス。通常、コンポーネントの設定はコンポーネントの配備記述子ファイルに保存されている。「管理サーバー (administration server)」、「配備記述子 (deployment descriptor)」も参照。

宣言によるセキュリティ (declarative security)　セキュリティプロパティをコンポーネントのコンフィグレーションファイル内で宣言し、コンポーネントのコンテナ (例 : Bean のコンテナやサーブレットエンジン) にセキュリティを暗黙的に管理させること。このタイプのセキュリティには、プログラムの制御は必要ない。「プログラムセキュリティ (programmatic security)」とは反対の概念。「コンテナ管理による持続性 (container-managed persistence)」を参照。

宣言によるトランザクション (declarative transaction)　「コンテナ管理によるトランザクション (container-managed transaction)」を参照。

送信先リソース (destination resource)　Topic 送信先または Queue 送信先を表すオブジェクト。キューの読み出しと書き込み、トピックのパブリッシュとサブスクライブを行うときにアプリケーションが使用する。アプリケーションコードは、JNDI 名を使う JMS リソースを JNDI サービスを使って特定する。

属性 (attribute)　サーブレットによって設定可能な、リクエストオブジェクト内の Name-value ペア。XML ファイル内の要素を修正する Name-value ペアでもある。「パラメータ」と対照的。一般的には、属性はメタデータの単位。

ダイジェスト認証 (digest authentication)　ユーザー名とパスワードをクリアテキストとして送信することなく、ユーザー名とパスワードに基づいてユーザーを認証する認証形態。

直列化可能オブジェクト (serializable object)　解体および再構築できるオブジェクト。複数のサーバーに保存したり分散したりできる。

データアクセスロジック (data access logic)　データソースとの対話を伴うビジネスロジック。

データソース (data source)　データベースなどの、データのソースへのハンドル。データソースは、iPlanet Application Server で登録された後、コネクションを確立してデータソースと対話できるようにするために、プログラムによって取得される。データソース定義により、データのソースへの接続方法を指定する。

データベース (database)　リレーショナルデータベース管理システム (RDBMS) の一般名。関連する組織化された大量のデータの作成および操作が可能なソフトウェアパッケージ。

データベース接続 (database connection)　データベースまたはほかのデータソースとの通信リンク。コンポーネントは、複数のデータベースコネクションを同時に作成および操作して、データにアクセスできる。

テーブル (table)　データベースの行および列内に保存されている関連データの特定のグループ。

ディレクトリサーバー (directory server)　「Sun ONE Directory Server」を参照。

電子商取引 (e-commerce)　電子商取引。インターネットで行うビジネス。

電子署名 (digital signature)　メッセージと署名者の両方の認証に使用される電子的なセキュリティメカニズム。

同一場所に置く (co-locate)　関連するコンポーネントと同じメモリ空間にコンポーネントを配備することによってリモートプロシージャコールを避け、パフォーマンスを向上させること。

動的再配備 (dynamic redeployment)　サーバーを再起動せずにコンポーネントを再配備するプロセス。

動的再読み込み (dynamic reloading)　サーバーを再起動せずにコンポーネントを更新して再読み込みするプロセス。デフォルトでは、サーブレット、JavaServer Page (JSP)、およびエンタープライズ Bean コンポーネントをダイナミックに再読み込みできる。バージョン付けとも呼ぶ。

ドキュメントルート (Document Root)　一次ドキュメントディレクトリ。仮想サーバーの全ファイルを格納してリモートクライアントに提供するための中心的なディレクトリ。

トピック (topic)　管理者が作成するオブジェクトで、パブリッシュ / サブスクライブ配信モデルが実装される。送られてきたメッセージの収集と分配を担当するコンテンツ階層に含まれるノードと考えることもできる。トピックを中間媒体として使うことで、メッセージのパブリッシャとサブスクライバを分離できる。

ドメインレジストリ (Domain Registry)　Sun ONE Application Server のインストールで作成、および設定されるすべてのドメインについて、ドメイン固有の情報 (ドメインの名前、場所、ポート、ホストなど) を含む 1 つのデータ構造。

トランザクション (transaction)　グループとして成功または失敗する一連のデータベースコマンド。トランザクション全体が成功するには、そのトランザクションに関連するすべてのコマンドが成功する必要がある。

トランザクションコンテキスト (transaction context)　ローカルまたはグローバルなトランザクションの範囲。「ローカルトランザクション (local transaction)」、「グローバルトランザクション (global transaction)」を参照。

トランザクション属性 (Transaction Attribute)　トランザクションの範囲を制御する。

トランザクション分離レベル (transaction isolation level)　データベース上で同時に実行されている複数のトランザクションをそれぞれに認識できる度合いを決定する。

トランザクションマネージャ (transaction manager)　通常 XA プロトコルを使ってグローバルトランザクションを制御するオブジェクト。「グローバルトランザクション (global transaction)」を参照。

トランザクションリカバリ (Transaction Recovery)　自動または手動による分散トランザクションのリカバリ。

トランジエンス (transience)　使われていないときにリソースを解放するプロトコル。「持続 (persistence)」とは反対の概念。

認証 (authentication)　ユーザーなどのエンティティが、別のエンティティ (アプリケーションなど) に対し特定の識別情報 (ユーザーのセキュリティ識別情報) を提供して認証されていることを証明するプロセス。Sun ONE Application Server は、基本的な認証のほかにフォームベースと SSL 相互認証もサポートしている。「クライアント認証 (client authentication)」、「ダイジェスト認証 (digest authentication)」、「ホスト -IP 認証 (host-IP authentication)」、「プラグイン対応認証 (pluggable authentication)」も参照。

ネットワーク管理ステーション (NMS)　特定のネットワークをリモートで管理するために使用するマシン。通常、NMS ソフトウェアには収集されたデータをグラフに表示したり、そのデータを使ってサーバが特定の許容範囲内で動作していることを確認する機能がある。「SNMP」も参照。

バージョン付け (versioning)　「動的再読み込み (dynamic reloading)」を参照。

パーミッション (permission)　ユーザーまたはグループに対して付与または拒否する一連の権限。「ACL」も参照。

配備 (deployment)　アプリケーションが必要とするファイルをアプリケーションサーバーに配布し、アプリケーションサーバー上でアプリケーションを実行できるようにするプロセス。「アセンブリ (assembly)」も参照。

配備記述子 (deployment descriptor)　配備方法を記述した XML ファイル。各モジュールおよびアプリケーションに備わっている。配備記述子は、配備ツールに、特定のコンテナオプションでモジュールまたはアプリケーションの配備を指示し、配備ツールが解決する必要のある特定の設定要件を示している。

バックアップストア (backup store)　データのリポジトリ。一般的にはファイルシステムやデータベース。バックアップストアをバックグラウンドスレッド (スィーパスレッド) で監視して、不要なエントリを削除することができる。

パッケージ (package)　共通ディレクトリ内に保存されている、関連するクラスのコレクション。クラスのコレクションは、頻繁に、Java アーカイブ JAR ファイルにパッケージ化される。「アセンブリ (assembly)」、「配備 (deployment)」も参照。

発見的決定 (Heuristic Decision)　特定のトランザクションが使用するトランザクションモデル。トランザクションは、コミットまたはロールバックする必要がある。

パブリッシュ / サブスクライブ配信モデル (publish/subscribe delivery model)　一般に、パブリッシャとサブスクライバは匿名で、トピックに対して動的にパブリッシュまたはサブスクライブできる。このシステムでは、トピックの複数のパブリッシャから受信したメッセージを複数のサブスクライバに配信できる。

パラメータ (parameter)　フォームフィールドデータや HTTP ヘッダー情報など、クライアントから送信される名前 - 値ペアであり、リクエストオブジェクト内にカプセル化されている。「属性」と対照的。一般的には、Java メソッドまたはデータベースコンパイル済みコマンドに渡される引数を指す。

ハンドル (handle)　Enterprise Java Beans を識別するオブジェクト。クライアントはハンドルを直列化した後で直列化を解除し、Beans への参照を取得する。

汎用 ACL (general ACL)　ユーザーまたはグループを 1 つまたは複数の権限に関連付ける、Sun ONE Directory Server 内の特定のリスト。一連の権限を記録するようにこのリストを定義し、自由にアクセスできる。

汎用サーブレット (generic servlet)　`javax.servlet.GenericServlet` を拡張するサーブレット。汎用サーブレットはプロトコルに依存しない。これは、汎用サーブレットは本来、HTTP やその他の転送プロトコルをサポートしていないことを意味する。「HTTP サーブレット」と対照的。

非活性化 (passivation)　Bean を破棄せずに Bean のリソースを解放するメソッド。これによって、Bean は持続的になり、インスタンス化せずに再び呼び出すことができる。

ビジネスロジック (business logic)　データ統合ロジックやプレゼンテーションロジックではなく、不可欠なビジネスルールを含むアプリケーションコード。

非同期通信 (asynchronous communication)　メッセージの送信側が、他の処理を継続する前に送信メソッドの返りを待つ必要のない通信モード。

秘密鍵 (private key) 「公開鍵暗号法 (public key cryptography)」を参照。

プール (pooling) 設定済みのリソースを増やしてパフォーマンスを向上させるプロセス。リソースがプールされていると、コンポーネントは新しくインスタンス化しなくても、プールから既存のインスタンスを使用できる。Sun ONE Application Server では、データベースコネクション、サーブレットインスタンス、および エンタープライズ Bean インスタンスをすべてプールできる。

ファイアウォール (firewall) セキュリティを強化するために、管理者がネットワーク上の情報フローを制限するときに使用する電子的な境界。

ファイルキャッシュ (File Cache) ファイルキャッシュには、ファイルに関する情報と静的なファイルコンテンツが含まれる。ファイルキャッシュはデフォルトで有効である。

ファクトリクラス (factory class) 持続性マネージャを作成するクラス。「接続ファクトリ (connection factory)」も参照。

フェィルオーバー (failover) Bean がサーバークラッシュに透過的に耐えられるようにするリカバリプロセス。

フォームアクションハンドラ (form action handler) フォーム上の特定のボタンに基づいてアクションを実行する、サーブレットまたはアプリケーションロジック内で特別に定義されているメソッド。

復号化 (decryption) 暗号化された情報を認識可能な状態に戻すプロセス。

符号化方式 (cipher) 暗号化と復号化に使用される暗号化アルゴリズム (関数)。

プラグイン対応認証 (pluggable authentication) J2EE アプリケーションが J2SE プラットフォームから JAAS (Java Authentication and Authorization Service) を利用できるようにするメカニズム。開発者は、独自の認証メカニズムをプラグインできる。

プレゼンテーションレイアウト (presentation layout) Web ページコンテンツの形式。

プレゼンテーションロジック (presentation logic) アプリケーションでページを作成するアクティビティ。リクエストの処理、レスポンスコンテンツの生成、クライアントに返すページのフォーマット化など。通常は、Web アプリケーションによって処理される。

ブローカ (broker) JMS メッセージのルーティング、配信、持続性、セキュリティ、ログを管理する Sun ONE Message Queue のエンティティ。管理者がパフォーマンスとリソース使用率の監視と調整に使うインタフェースを提供する。

プログラマによる境界設定トランザクション (programmer-demarcated transaction) 「Bean 管理によるトランザクション (bean-managed transaction)」を参照。

プログラムセキュリティ (programmatic security)　コンポーネントのコンテナ (Bean のコンテナやサーブレットエンジンなど) による処理ではなく、コードを記述して明示的にセキュリティを制御するプロセス。「宣言によるセキュリティ (declarative security)」とは反対の概念。

プロセス (process)　アクティブなプログラムの実行シーケンス。プロセスは、1 つまたは複数のスレッドから構成される。

プロパティ (property)　アプリケーションコンポーネントの動作を定義する 1 つの属性。`server.xml` ファイルでは、プロパティは名前と値のペアを含む要素である。

分散可能セッション (distributable session)　クラスタ内のすべてのサーバー間に分散できるユーザーセッション。

分散トランザクション (distributed transaction)　別個のサーバー上に配備されている複数の異種データベースに適用可能な 1 つのトランザクション。

分離レベル (isolation level)　「トランザクション分離レベル (transaction isolation level)」を参照。

ホームインタフェース (home interface)　クライアントによる Enterprise Bean の作成や削除を可能にするメソッドを定義するメカニズム。

ポイントツーポイント配信モデル (point-to-point delivery model)　プロデューサはメッセージを特定のキューに送り、コンシューマは、そのメッセージを保持するために確立されたキューからメッセージを抽出する。1 つのメッセージは 1 つのメッセージコンシューマだけに配信される。

ホスト -IP 認証 (host-IP authentication)　特定のコンピュータを使うクライアントだけにアクセスを限定することによって、管理サーバー、または Web サイト上のファイルやディレクトリへのアクセスを制限するセキュリティメカニズム。

マッピング (mapping)　オブジェクト指向モデルを、データのリレーショナルモデル (通常はリレーショナルデータベースのスキーマ) に結びつける機能。スキーマを別の構造に変換するプロセス。ユーザーとセキュリティロールとの関連付けも意味する。

メタデータ (metadata)　コンポーネントの名前やその動作の仕様などの、コンポーネントに関する情報。

メッセージ駆動型 Beans (message-driven bean)　非同期メッセージコンシューマの Enterprise JavaBean。メッセージ駆動型 Beans は特定のクライアントのステートを持っていないが、インスタンス変数はクライアントメッセージの処理に関するステート (オープンデータベースコネクションや EJB オブジェクトへのオブジェクト参照など) を持っていることがある。クライアントは、メッセージ駆動型 Bean がメッセージリスナとなっている宛先にメッセージを送信して、メッセージ駆動型 Bean にアクセスする。

メッセージング (messaging)　エンタープライズ Bean が使用する非同期の要求、報告、またはイベントのシステム。緩く結合されたアプリケーション間で確実かつ安全に情報をやり取りするときに利用される。

モジュール (module)　アプリケーションの外部に個別に配備された Web アプリケーション、Enterprise Bean、メッセージ駆動型 Bean、アプリケーションクライアント、またはコネクタ。「アプリケーション (application)」、「コンポーネント (component)」、「ライフサイクルモジュール (lifecycle module)」も参照。

ユーザー (user)　アプリケーションを使う人。プログラム的には、アプリケーションがクライアントを認識する際の手掛かりとなるユーザー名、パスワード、および一連の属性で構成される。「グループ (group)」、「ロール (role)」も参照。

ユーザーセッション (user session)　サーバーによって記録される、ユーザーとアプリケーション間の一連の対話。セッションでは、ユーザーステート、持続オブジェクト、および ID 認証が管理される。

要素 (element)　より大きなセットの集まり。たとえば、配列のデータ単位や、論理要素。XML ファイルでは、これが基本構造単位となる。XML 要素は、サブ要素またはデータを含み、属性を含むこともある。

呼び出し可能なステートメント (callable statement)　ストアドプロシージャからのリザルトセットの戻しをサポートしているデータベースのデータベースプロシージャまたは関数呼び出しがカプセル化されているクラス。

読み込み専用 Bean (read-only bean)　EJB クライアントで修正されることがないエンティティ Bean。「エンティティ Bean (entity bean)」も参照。

ライフサイクルイベント (lifecycle event)　起動や停止など、サーバーのライフサイクルの各段階。

ライフサイクルモジュール (lifecycle module)　サーバーライフサイクルのイベントに応じて、タスクを待機し、実行するモジュール。

リクエストオブジェクト (request object)　クライアントによって生成されたページおよびセッションデータが含まれているオブジェクトであり、入力パラメータとしてサーブレットまたは JavaServer Page (JSP) に渡される。

リスナー (Listener)　ポストするオブジェクトに登録され、イベント発生時の処理を指示するクラス。

リソース参照 (resource reference)　配備記述子の要素で、リソースのコード化されたコンポーネント名を識別する。

リソースマネージャ (resource manager)　リソース (たとえばデータベースやメッセージブローカ) とクライアント (たとえば Sun ONE Application Server プロセス) のまとめ役となるオブジェクト。グローバルに利用可能なデータソースを制御する。

リモートインタフェース (remote interface)　Enterprise JavaBean の 2 つのインタフェースのうちの 1 つ。リモートインタフェースでは、クライアントから呼び出すビジネスメソッドを定義する。

レスポンスオブジェクト (response object)　呼び出しているクライアントを参照して、そのクライアントへの出力を生成するメソッドを提供するオブジェクト。

レルム (realm)　共通セキュリティポリシーが定義され、セキュリティサービスのセキュリティ管理者によって適用されている領域。J2EE 仕様では、セキュリティポリシードメインまたはセキュリティドメインとも呼ばれる。

ローカルインタフェース (local interface)　同じ Java 仮想マシン (JVM) にあるクライアントのメカニズムに、Bean にアクセスするためのセッションやエンティティ Bean を提供するインタフェース。

ローカルセッション (local session)　1 つのサーバーだけに見えるユーザーセッション。

ローカルデータベース接続 (local database connection)　ローカルコネクションのトランザクションコンテキストは現在のプロセスおよびデータソースに対してローカルであり、複数のプロセスまたはデータソース全体に分散できない。

ローカルトランザクション (local transaction)　1 つのデータベースに固有で、1 つのプロセス内に制限されるトランザクション。ローカルトランザクションは、1 つのバックエンドでのみ動作する。ローカルトランザクションは通常、JDBC API を使って区別される。「グローバルトランザクション (global transaction)」も参照。

ロール (role)　アプリケーションにおいてサブジェクトを機能別にグループ分けしたもの。配備環境では 1 つまたは複数のグループによって表される。「ユーザー (user)」、「グループ (group)」も参照。

ロールバック (rollback)　トランザクションの取り消し。

索引

R

S

U

X

あ

い

か

き

く

こ

さ

し

せ

そ

た

て

と

は

ひ

ふ

め

り

れ