

入門ガイド

Sun™ ONE Application Server

Version 7, Enterprise Edition

817-5547-10

2003 年 9 月

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.

Copyright © 2003 Sun Microsystems, Inc. All rights reserved.

Sun、Sun Microsystems、Sun のロゴマーク、Java、Solaris、Sun ONE、iPlanet およびすべての Sun、Java、および Sun ONE ベースの商標およびロゴマークは、米国およびその他の国における米国 Sun Microsystems, Inc. の商標もしくは登録商標です。

UNIX は、X/Open Company, Ltd が独占的にライセンスしている米国およびその他の国における登録商標です。

Netscape は、米国およびその他の国における Netscape Communications Corporation 社の登録商標です。

Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions

本書で説明されている製品は著作権法により保護されており、その使用、複製、頒布および逆コンパイルを制限するライセンスのもとにおいて頒布されます。Sun および Sun のライセンサーの書面による事前の許可なく、本製品および関連する文書のいかなる部分も、いかなる方法によっても複製することが禁じられます。

本書は、「現状のまま」をベースとして提供され、商品性、特定目的への適合性または第三者の権利の非侵害の黙示の保証を含みそれに限定されない、明示的であるか黙示的であるかを問わない、なんらの保証も行われないものとします。

目次

本書について	7
マニュアルの概要	7
マニュアルの構成	8
マニュアルの使用方法	8
マニュアルの表記規則	11
一般的な表記規則	11
ディレクトリ名の表記規則	11
製品サポート	12
 第 1 章 Sun ONE Application Server 7、Enterprise Edition の概要	13
Sun ONE Application Server について	13
Enterprise Edition 機能について	16
クラスタリング	16
ロードバランス	17
セッション持続性	18
セッション持続性のタイプについて	18
セッション持続性の設定について	19
シングルサインオンセッション情報について	19
高可用性データベース	19
クラスタリングのシナリオの例	20
Application Server の設定と管理に使用するツール	22
 第 2 章 エンタープライズ機能の設定のチュートリアル	23
チュートリアルを使用する準備	23
インストールの要件	24
チュートリアルを使用する前の必要な手順	24

チュートリアルの手順の概要	26
サーバーの起動	27
PATH 変数の設定	27
asadmin start-domain の実行	28
サーバーの起動の確認	29
管理サーバーの起動の確認	30
管理インタフェースへのアクセス	30
管理サーバーのイベントログの表示	31
サーバーインスタンスの起動の確認	32
asadmin を使ったインスタンスの確認	32
HTTP サーバーへのアクセスによるインスタンスの確認	33
loadbalancer.xml ファイルの作成	35
loadbalancer.xml ファイルへのクラスタの追加	36
ロードバランスの設定	39
ヘルスチェッカーの設定	39
ロードバランサの監視の有効化	40
ロードバランサのその他のプロパティ	42
再読み込みのポーリング間隔を使った動的再設定	42
応答タイムアウト	42
HTTPS ルーティング	43
loadbalancer.xml ファイルのサンプル	43
 第 3 章 クラスタ JSP サンプルアプリケーションのチュートリアル	 45
クラスタ JSP サンプルアプリケーションのチュートリアルを使用する準備	46
クラスタ JSP のサンプルアプリケーションのクラスタへの配備	47
cladmin コマンドの入力ファイル	48
cladmin コマンドの構文	48
cladmin deploy の実行	49
cladmin コマンドでサポートされている asadmin コマンド	50
要件と制限事項	51
cladmin を使ったアプリケーションサーバーインスタンスの起動	51
アプリケーション配備の確認	52
アプリケーションサーバーのサンプルの監視	54
管理インタフェースによるログの表示	54
tail コマンドによるログの表示	56
イベントログのアプリケーション生成のメッセージ	57
アクセスログのアプリケーション生成のメッセージ	57
アプリケーションサーバーインスタンスへの配備に対するトラブルシューティング	58
クラスタへのサンプルアプリケーションの追加	59
設定の変更の適用と Web サーバーの再起動	60
アプリケーションの実行	61
ロードバランスの確認	66

ロードバランスの確認手順	66
ロードバランサプラグインのトラブルシューティング	68
loadbalancer.xml ファイル内のエラーの検出	68
ヘルスチェッカーの使用	69
HTTP セッションの持続性の確認	70
サーバーインスタンスの休止	71
 第 4 章 要約と参照情報	 75
索引	77

本書について

ここでは、『Sun™ Open Net Environment (Sun ONE) Application Server 7 入門ガイド』の内容について説明します。

この章には次の節があります。

- [マニュアルの概要](#)
- [マニュアルの構成](#)
- [マニュアルの使用方法](#)
- [マニュアルの表記規則](#)
- [製品サポート](#)

マニュアルの概要

この『入門ガイド』は、Sun ONE Application Server を初めて使うユーザーを対象としています。このマニュアルでは、ロードバランスや HTTP セッションの持続性など、AS7EE の固有の機能を含め、実務的な知識を得るための簡潔な操作手順を記載しています。アプリケーションサーバーの使用や開発の経験は特に必要ありません。ただし、『Sun ONE Application Server インストールガイド』の指示に従ってサーバーをインストールしておく必要があります。

最初に、クラスタやロードバランス、HTTP セッションの持続性機能の概要を含めて、AS7EE の機能の全体像を紹介します。次に、例にあるサンプルアプリケーションを実行するための環境の設定手順に進みます。ここでは、サンプルアプリケーションを配備する方法と、ロードバランスおよびセッションの持続性を確認する方法を示しています。また、より詳細な情報入手のため目的別に情報の参照先を記載しています。

マニュアルの構成

このマニュアルは次の章から構成されています。

- 第 1 章「Sun ONE Application Server 7、Enterprise Edition の概要」では、クラスタリング、ロードバランス、HTTP セッションの持続性、高可用性データベースの概略説明など、製品の概要を説明します。
- 第 2 章「エンタープライズ機能の設定のチュートリアル」には、チュートリアルを始める前に実行するタスクのリスト、エンタープライズ機能を使うようにサーバーを設定する手順の概要、クラスタと HTTP セッションのフェイルオーバーを使うためのサーバー設定のチュートリアルが記載されています。
- 第 3 章「クラスタ JSP サンプルアプリケーションのチュートリアル」では、クラスタ JSP アプリケーションのクラスタへの配備方法、および実行方法について説明します。また、このアプリケーションを使ってロードバランスと HTTP セッションの持続性を確認する方法も説明します。
- 第 4 章「要約と参照情報」では、このマニュアルで説明した作業を振り返り、次に参照すべき情報について説明します。

マニュアルの使用方法

Sun ONE Application Server のマニュアルは、PDF 形式または HTML 形式で、次のサイトから入手できます。

<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。

表 1 Sun ONE Application Server のマニュアルの概要

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』
サポート対象のハードウェア、オペレーティングシステム、JDK、JDBC、RDBMS の一覧	『Platform Summary』
Sun ONE Application Server 7 の概要と、製品の各エディションで利用可能な機能	製品の概要
サーバーのアーキテクチャの図と説明、Sun ONE Application Server アーキテクチャの利点	『サーバーアーキテクチャの概要』
企業、開発者、および運用向けの、Sun ONE Application Server 7 の新機能	新機能

表 1 Sun ONE Application Server のマニュアルの概要 (続き)

情報の内容	参照するマニュアル
Sun ONE Application Server 7 製品の基本的な使用方法。サンプルアプリケーションのチュートリアルも含まれる	入門ガイド
Sun ONE Application Server ソフトウェア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、高可用性コンポーネントなど) のインストール。高可用性の基本設定の実装手順が記載されている	インストールガイド
Sun ONE Application Server を確実にサイトに適合させるように配備するための、システムのニーズとエンタープライズの評価。アプリケーションサーバーを配備する際に注意する必要がある、一般的な問題と懸案事項も記載されている	システム配備ガイド
アプリケーションの設計者と開発者が、HTTP セッションの可用性を確保するための最適な方法	『Application Design Guidelines for Storing Session State』
J2EE コンポーネント (サーブレット、EJB™ (Enterprise JavaBeans™)、JSP™ (JavaServer Pages™) など) 向け Java オープンスタンダードモデルに準拠した Sun ONE Application Server 7 上で実行することを目的とした Java™ Platform, Enterprise Edition (J2EE™ プラットフォーム) アプリケーションの作成および実装。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方法などについての一般的な情報が含まれる。Sun ONE Application Server のさまざまな用語について解説する用語集も含まれる	『Developer's Guide』
Sun ONE Application Server 7 の Java™ Servlet および JSP (JavaServer Pages) 仕様に準拠した J2EE Web アプリケーションの作成および実装。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。結果キャッシュ機能、JSP のプリコンパイル、セッション管理、セキュリティ、配備、SHTML、CGI などが含まれる	『Developer's Guide to Web Application』
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および実装。Enterprise JavaBeans (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。コンテナ管理持続性、読み取り専用 Bean、エンタープライズ Bean に関連付けられた XML ファイルや DTD ファイルなどが含まれる	『Developer's Guide to Enterprise JavaBeans』
Sun ONE Application Server 7 上で J2EE アプリケーションにアクセスする Application Client Container (ACC) の作成	Developer's Guide to Clients
Sun ONE Application Server 環境での Web サービスの作成	Developer's Guide to Web Services
Java™ Database Connectivity (JDBC™)、トランザクション、Java Naming and Directory Interface™ (JNDI)、Java™ Message Service (JMS)、および JavaMail™ API	『Developer's Guide to J2EE Services and APIs』
カスタム NSAPI プラグインの作成	NSAPI Developer's Guide

表 1 Sun ONE Application Server のマニュアルの概要 (続き)

情報の内容	参照するマニュアル
管理インタフェースとコマンド行インタフェースの両方からの Sun ONE Application Server サブシステムとコンポーネントの設定、管理、および配備の説明と手順。クラスタ管理、高可用性データベース、ロードバランス、およびセッションの持続性について説明する。Sun ONE Application Server のさまざまな用語について解説する用語集も含まれる	管理者ガイド
server.xml ファイルなどの Sun ONE Application Server の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 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	セキュリティ管理者ガイド
Sun ONE Application Server 7 用の J2EE™ CA (Connector Architecture) 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、プーリングモニター、JCA コネクタの配備、サンプルコネクタとサンプルアプリケーションなどについて説明する	J2EE CA Service Provider Implementation Administrator's Guide
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 (特に、iPlanet Application Server 6.x、Netscape Application Server 4.0 からの移行)。移行例も含まれる	サーバーアプリケーションの移行および再配備
Sun ONE Application Server のパフォーマンスを改善する方法と、なぜそうする必要があるか	『Performance Tuning Guide』
Sun ONE Application Server に関する問題の解決	『Troubleshooting Guide』
Sun ONE Application Server 7 の実行中に表示される可能性のあるメッセージ。メッセージが生成された状態について、考えられる原因とガイドラインを説明する	『Error Message Reference』
マニュアルページに記載されている、Sun ONE Application Server で利用できるユーティリティコマンド	『Utility Reference Manual』
Sun™ Open Net Environment (Sun ONE) Message Queue ソフトウェアの使用	Sun ONE Message Queue については次の URL を参照: http://docs.sun.com/db/prod/s1.s1msgqu?l=ja#hic

マニュアルの表記規則

この節では、このマニュアルの表記規則について説明します。

- 一般的な表記規則
- ディレクトリ名の表記規則

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「/」記号で区切って表記)。
- 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、コード例、コードリスト、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 タグに使用します。
 - 斜体文字は、変数および可変部分、およびリテラルに使われる文字に使います。
 - 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。

ディレクトリ名の表記規則

デフォルトでは、Sun ONE Application Server のファイルは複数のルートディレクトリ間に分散しています。ここでは、これらのディレクトリについて説明します。

- *install_dir* は、インストールイメージの静的な要素が保存されるディレクトリを指します。Sun ONE Application Server を構成するユーティリティ、実行可能ファイル、およびライブラリは、すべてここに保存されます。デフォルトの場所は `/opt/SUNWappserver7` です。
- *default_config_dir* は、作成されたドメインのデフォルトの保存場所であるディレクトリを指します。デフォルトの場所は `/var/opt/SUNWappserver7/domains` です。

- `install_config_dir` は、インストール全体の設定情報が保存されるディレクトリを指します。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ストなどが保存されます。デフォルトの場所は `/etc/opt/SUNWappserver7` です。

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/supporttraining/>

- 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他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

Sun ONE Application Server 7、Enterprise Edition の概要

この章では次の項目について説明します。

- [Sun ONE Application Server](#) について
- [Enterprise Edition 機能](#)について
- [Application Server](#) の設定と管理に使用するツール

Sun ONE Application Server について

Sun™ Open Net Environment (ONE) Application Server 7、Enterprise Edition は、アプリケーションサービスや Web サービスを広く配備する場合に適した高性能な Java™ 2 Platform、Enterprise Edition (J2EE™) プラットフォームを提供します。Sun ONE Application Server 7 は、検証済みの HTTP サーバーインフラストラクチャ、Java Message Service (JMS) をはじめとする業界標準コンポーネントと、Java Web Services Developer Pack ソフトウェアに付属の J2EE バージョン 1.3、Java 2 Platform, Standard Edition バージョン 1.4、Java API for XML (JAX) による最新の J2EE および Web サービス仕様を基盤としたモジュールアーキテクチャを採用しています。また、クラスタリング、ロードバランス、および HTTP セッションフェイルオーバーも提供します。機能の詳細については、『Sun ONE Application Server 製品の概要』を参照してください。

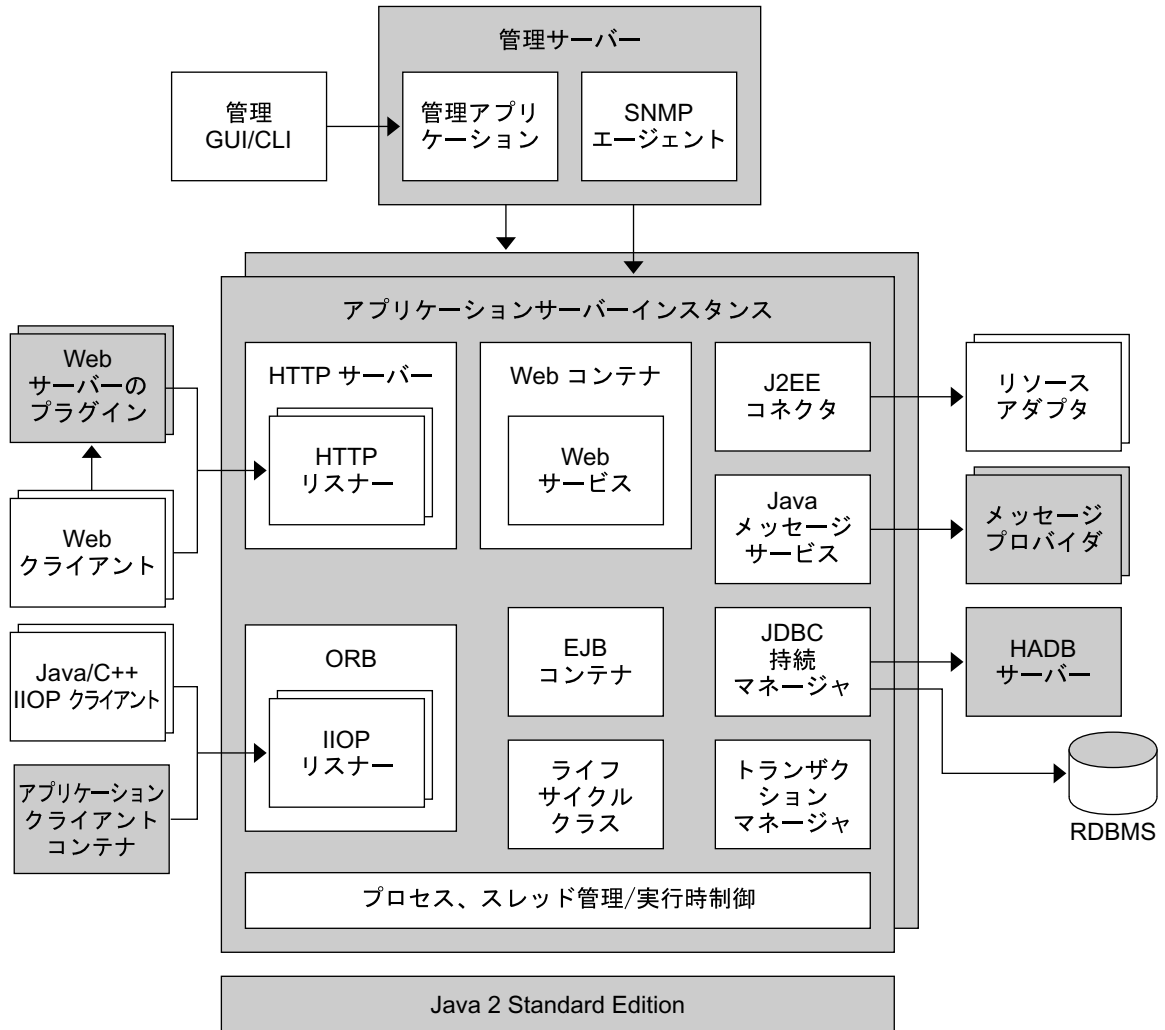
Application Server インスタンスは、Application Server の配備の基本となります。各インスタンスには、それぞれに専用のディレクトリ構造、設定、配備アプリケーションがあります。各サーバーインスタンスには、J2EE プラットフォームの Web および EJB コンテナも組み込まれています。Sun ONE Application Server にも、検証済みの高性能な HTTP サーバーが組み込まれています。オブジェクトリクエストブローカ (Object Request Broker (ORB)) モジュールにより、RMI-IIOP を使って EJB を呼び出すことができます。

アプリケーションは、バックエンドシステムにアクセスするため、J2EE コネクタアーキテクチャに対応したリソースアダプタ、サードパーティのリソースアダプタ、組み込みの JMS プロバイダまたはサードパーティのプロバイダに加えて、一般的なサードパーティ JDBC ドライバを組み合わせて使用します。分散トランザクションの範囲内であれば、バックエンドシステムへのアクセスは、すべて Java で記述された内蔵のトランザクションマネージャを使って管理できます。

管理サーバーは、コア管理アプリケーションと SNMP エージェントを実行する、特殊なサーバーインスタンスです。リモート管理は、すべて管理サーバー経由で行われます。管理サーバーには、コマンド行や Web ブラウザベースの管理クライアントから HTTP を使用するか、セキュリティ保護された HTTPS を使って安全に直接アクセスします。

図 1-1 に、アプリケーションサーバーインスタンスの詳細を示します。アプリケーションサーバーインスタンスは、16 ページの「Enterprise Edition 機能について」に記載されているクラスタリング、ロードバランス、およびセッション持続性機能の基本的要素です。

図 1-1 Sun ONE Application Server インスタンス



Web サーバーのプラグイン (ロードバランサプラグインなど) を使用すると、1つ以上のファイアウォール層によって保護された非武装地帯 (DMZ) に置かれた 1つまたは複数の Web サーバーの背後に、アプリケーションサーバーを配備できます。フロントエンドの Web サーバー層は、このプラグインを使って、インターネットから受信した HTTP/HTTPS トラフィックをバックエンドのアプリケーションサーバー層にあるアプリケーションサーバー (複数可) に送信します。

さまざまなクライアントアプリケーションが、アプリケーションサーバーに配備されたビジネスサービスにアクセスできます。Web サービスクライアントとブラウザベースのクライアントは、HTTP または HTTPS を使って、Web サービス、サーバー側の終点、および J2EE Web アプリケーションにアクセスできます。

図 1-1 では、単一の管理ドメインを示しています。管理ドメインを使用すると、同じインストールイメージを再使用する、複数の完全に別個のアプリケーションサーバーの実行時設定を定義できます。それぞれの管理ドメインには管理サーバーがあり、この管理サーバーが 1 つまたは複数のアプリケーションサーバーインスタンスを制御します。このチュートリアルでは、製品のインストール時に設定される単一の管理ドメインと、クラスタ内の複数のサーバーインスタンスを扱います。

Enterprise Edition 機能について

アプリケーションサーバーインスタンスをクラスタにグループ化でき、Web サーバーとロードバランサとともに、ロードバランスと HTTP セッションの持続性を提供できます。このマニュアルには、次の節で説明する Enterprise Edition 機能の使用方法が記載されています。

- クラスタリング
- ロードバランス
- セッション持続性
- 高可用性データベース
- クラスタリングのシナリオの例

クラスタリング

クラスタは、1 つの論理エンティティとして機能する、Application Server インスタンスの集まりです。クラスタ内の各アプリケーションサーバーインスタンスには、同じ設定と同じアプリケーションが配備されています。

Sun ONE Application Server のクラスタを使用すると、次のことを実現できます。

- 高可用性。クラスタ内のアプリケーションサーバーに対し、フェイルオーバーによる保護を実現します。1 つのアプリケーションサーバーインスタンスが停止した場合、別のアプリケーションサーバーインスタンスが、利用できなくなったサーバーが処理していたセッションを引き受けます。Sun ONE Application Server は、HTTP セッション内の HTTP セッションと EJB 参照のフェイルオーバーをサポートしています。

- スケーラビリティ。クラスターへのアプリケーションサーバーインスタンスの追加によって、システム全体の容量が増加します。Sun ONE Application Server のロードバランサは、クラスター内の稼働しているアプリケーションサーバーインスタンスに要求を分散します。サービスを中断せずに、アプリケーションサーバーインスタンスをクラスターに追加できます。

クラスター内のアプリケーションサーバーインスタンスは、異なるマシン上でも同じマシン上でもホストすることができます。つまり、クラスター内の複数のマシン上にあるアプリケーションサーバーインスタンスをグループ化できます。このマニュアルのデフォルトの設定では、同じマシン上の2つのインスタンスになっています。

クラスタリングの詳細については、『Sun ONE Application Server 管理者ガイド』を参照してください。

ロードバランス

ロードバランスの目的は、複数の Sun ONE Application Server インスタンスにかかる作業負荷を均等に分散させることです。Sun ONE Application Server に付属のロードバランサプラグインを使用するか、サードパーティのハードウェアおよびソフトウェアのロードバランサを使用することができます。このマニュアルでは、ロードバランサプラグインを使って説明します。

Sun ONE Application Server のロードバランサは、スティッキーなラウンドロビンアルゴリズムを使用して、受信した HTTP および HTTPS 要求のロードバランスを行います。新しい HTTP 要求がロードバランサプラグインに送られると、その要求は、単純なラウンドロビンスキームに基づいてアプリケーションサーバーインスタンスに転送されます。その後、この要求は、Cookie を使用するか、または明示的な URL の書き換えによって、この特定のアプリケーションサーバーインスタンスに「スタック」されます。

ロードバランサプラグインは最初に、スティッキー情報から、以前に要求が転送されたインスタンスを判別します。ロードバランサプラグインは、そのインスタンスが正常に稼働していることがわかると、要求をその特定のアプリケーションサーバーインスタンスに転送します。

特定のセッションに対するすべての要求は同じアプリケーションサーバーインスタンスに送信されるため、セッションデータは、クラスター内のすべてのインスタンスに分散されるのではなく、単一の Application Server のキャッシュに保存されます。このキャッシュは、スティッキーなラウンドロビンによりパフォーマンスが大幅に向上することを意味します。通常これは、純粋なラウンドロビンを使って取得できる、より均等な分散による利点を無効にします。

ロードバランスの詳細については、『Sun ONE Application Server 管理者ガイド』を参照してください。

セッション持続性

セッション持続性の機能によって、Sun ONE Application Server のインスタンスやマシンそのものに不具合が発生した場合でも、HTTP/HTTPS の要求は、別のサーバーに引き継がれることが保証されます。Sun ONE Application Server は、HTTP セッションの持続性と、HTTP セッション内の EJB 参照のフェイルオーバーをサポートしています。

高可用性データベース (HADB) は、Web アプリケーションに高可用性を提供する持続性ストアとして、Sun ONE Application Server にバンドルされています。高可用性データベースの詳細については、[19 ページの「高可用性データベース」](#)を参照してください。

この節には次の項目があります。

- [セッション持続性のタイプについて](#)
- [セッション持続性の設定について](#)
- [シングルサインオンセッション情報について](#)

セッション持続性のタイプについて

Sun ONE Application Server は、ha、file、および memory という、3 つのタイプの持続性をサポートしています。

- ha 持続性タイプは、HADB のセッション情報を格納します。ha 持続性タイプは、フェイルオーバー機能を必要とする本稼働環境に対応しています。
- file 持続性タイプはセッション情報を定期的にファイルに格納しますが、インスタンスが利用できなくなったときにデータが失われることがあります。file 持続性タイプは、フェイルオーバー機能を必要とする本稼働環境には対応していませんが、アプリケーションのテストのために開発環境で使われることがあります。
- memory 持続性タイプは、サーバーインスタンスがシャットダウンした場合に、ファイル内のセッション情報を格納します。しかし、予期しないシャットダウンの場合はセッション情報を格納しません。memory 持続性タイプは、フェイルオーバー機能を必要とする本稼働環境には対応していません。

このマニュアルの例では、ha セッション持続性のみ説明しています。その他のタイプのセッション持続性の設定と使用の詳細については、『Sun ONE Application Server 管理者ガイド』を参照してください。

セッション持続性の設定について

Sun ONE Application Server をインストールして、`clsetup` コマンドを実行すると、セッション持続性情報がデフォルト値に設定されます。ただし、パフォーマンス、信頼性、高可用性の特定のニーズに合うように、デフォルト値を変更することができます。

たとえば、データを保存するたびに、セッション全体を格納することができます。また、セッションが変更された場合は、セッションのみを格納できます。つまり、セッションの変更された属性のみが格納されるように、持続性を設定することもできます。

同様に、持続性の頻度を選択できます。たとえば、それぞれの Web 要求のあとにセッションを格納するように選択して、更新されたセッションの状態の高可用性と信頼性を提供したり、指定した時間間隔のあとにセッションを格納するように選択して、パフォーマンスを向上させることができます。

セッション持続性の各種設定オプションの詳細については、『Sun ONE Application Server 管理者ガイド』を参照してください。

シングルサインオンセッション情報について

単一のアプリケーションサーバーインスタンスでは、Web アプリケーションで 1 度認証されると、同じインスタンス上で実行中のその他の Web アプリケーションに対して個別に再度認証を要求されません。これを、シングルサインオンといいます。

この機能で、セッションがクラスタ内の別のインスタンスにフェイルオーバーしたときでも動作を続けるためには、シングルサインオン情報が HADB に継続して置かれている必要があります。この持続性は、アプリケーションサーバーインスタンスの高可用性が有効なときに有効です。高可用性は、`clsetup` コマンドを実行すると、自動的に設定されます。

高可用性データベース

高可用性データベース (HADB) はセッション情報の格納に使われ、HTTP セッションが持続できるようにします。高可用性は、ハードウェアやソフトウェアの障害によって予定外の停止があっても、引き続きシステムを使用できることを意味します。HADB は JDBC 準拠のデータベースであり、「Always-On (常時配信)」テクノロジーに基づいています。HADB は、99.999% を超えるデータの有効性を実現できることから、高い負荷が継続するアプリケーションサーバーの商用環境において、すべての種類の HTTP セッションの状態を持続的に処理するための、理想的なプラットフォームを提供します。

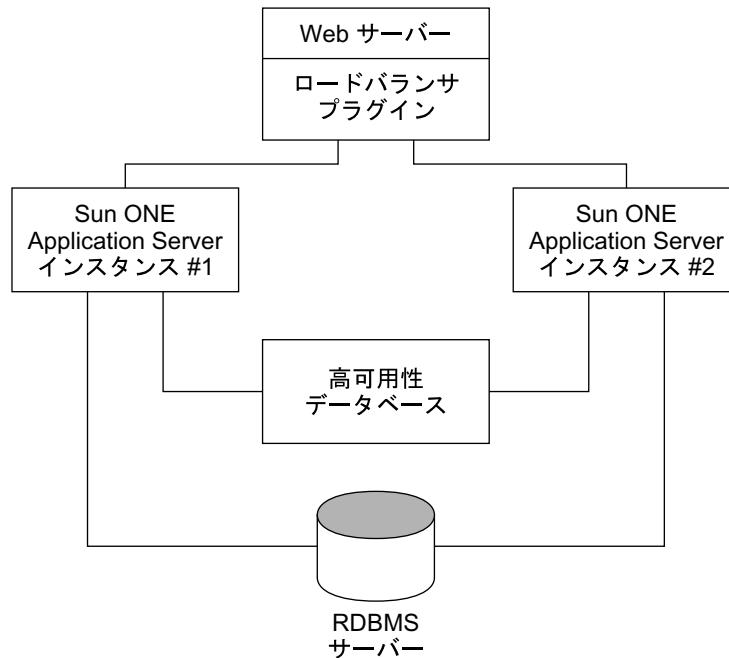
HADB は、データの分割とレプリケーションによって、このようなデータの可用性を提供します。このデータベース内のすべてのテーブルは、フラグメントというほぼ同じサイズのサブセットを作成するように区切られています。この分割のプロセスは、データベースのノード間のデータを分割して均等に分散させる、ハッシュ機能に基づいています。それぞれのフラグメントは、データベースのミラーノードに、2 回格納されます。これにより、データの耐障害性と高速回復が保証されます。また、ノードが使用不能になるか、あるいはシャットダウンした場合は、そのノードが再度アクティブになるまで、予備のノードが代行します。

高可用性の機能を実装していないシステムでは、予定外の停電が発生した場合など、該当する Web コンテナから別なコンテナへのフェイルオーバーが実行されている間、HTTP セッションの状態が失われてしまいます。しかし、HADB に実装されたセッション持続性のメカニズムを使用することによって、このような状態を事前に回避できます。HADB では、分割されているが確実に統合されている持続性ストレージ層の中で、すべての状態情報の格納や検索が可能だからです。

クラスタリングのシナリオの例

次の図で、1 つのロードバランサプラグイン、HADB を使って HTTP セッションデータを格納するように設定された 2 つの Sun ONE Application Server インスタンス、およびアプリケーションデータを格納するリモートデータベース管理システム (RDBMS) を持つ Web サーバーで構成される、単純なクラスタリングのシナリオを示します。実際の配備とは異なる場合があることに注意してください。たとえば、ロードバランサプラグインの代わりにサードパーティのロードバランサを使用することもできます。

図 1-2 クラスタリングのシナリオの例



要求が処理される手順は、次のとおりです。

1. 要求を受信したクライアントは、HTTP 要求を Web サーバーが処理する URL に送信します。この Web サーバーは、ロードバランサプラグインが着信 HTTP 要求を処理できるように設定されています。
2. そのあと、ロードバランサプラグインは要求を、クラスタ内の Sun ONE Application Server インスタンスの 1 つに転送します。プラグインは、ターゲットインスタンスを特定するためにスティッキーなラウンドロビンのロードバランサを使用します。
3. ターゲットインスタンスは、ロードバランサプラグインから転送された要求を受信し、HTTP セッションデータを HADB、J2EE アプリケーションデータを RDBMS に格納して、HTTP セッションを開始します。アプリケーションによるクライアントの処理の進捗状況に従って、HTTP セッションデータが更新されて HADB に格納され、RDBMS のアプリケーションデータが更新されます。
4. インスタンスに、システムクラッシュなどの問題が発生すると、ロードバランサは、インスタンスが要求への応答を中止したことを検出します。それ以降に要求が届くと、ロードバランサは、その要求をクラスタ内の正常なインスタンスに転送します。

5. 新しいターゲットインスタンスはフェイルオーバーした HTTP セッション情報を HADB から受信し、クライアントの要求に対する応答を続けます。このため、クライアントはセッションデータを失うことなく HTTP セッションを完了できます。

Sun ONE Application Server の配備のシナリオの詳細については、『Sun ONE Application Server システム配備ガイド』を参照してください。

Application Server の設定と管理に使用するツール

Sun ONE Application Server には、設定と管理に使用できる以下のツールが用意されています。これらのツールを使って、サーバーの起動と停止や、さまざまなその他の機能を実行できます。特定の設定作業を行うために使用する一部のツールについては、その使用方法がチュートリアルに記載されています。

- **asadmin ユーティリティ** : asadmin ユーティリティは、単一のアプリケーションサーバーインスタンスまたは管理ドメインでの管理タスクの実行に使用できる、コマンド行インタフェースです。
- **cladmin コマンド** : cladmin コマンドにより、クラスタ内のすべてのアプリケーションサーバーインスタンスで、特定の asadmin コマンドが同時に実行されます。cladmin コマンドを使用すれば、クラスタ内のすべてのインスタンスが同じように設定されます。このため、可能な限り、asadmin ではなく cladmin を使用してください。cladmin はすべての asadmin コマンドをサポートしているわけではありません。
- **管理インタフェース** : 管理インタフェースは、管理サーバーと個々のアプリケーションサーバーインスタンスの設定に使用できる、Web ベースのユーザーインタフェースです。また、個々のアプリケーションサーバーインスタンスのログファイルの表示にも使用できます。
- **clsetup コマンド** : clsetup コマンドにより、HADB データベースの初期設定など、クラスタを簡単に設定することができます。clsetup コマンドの詳細については、『Sun ONE Application Server インストールガイド』を参照してください。

注	いくつかのサーバー設定とその対応するインタフェースは、現在も開発が進んでいます。不安定なインタフェースは、次期製品リリースでより安全で安定したものに置き換えられる可能性があります。ほとんどのサーバー設定および管理インタフェースがそのまま残るか互換性を持ったまま変更されますが、いくつかの点で互換性が保たれない場合もあります。互換性のない機能の変更内容については、製品の以降のリリースで変更が発生した場合に、ドキュメントに明確に記述することを予定しています。
----------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

エンタープライズ機能の設定のチュートリアル

この章では、Sun ONE Application Server 7, Enterprise Edition 環境のセットアップと使用方法に関する、一連のチュートリアルを記載しています。この章には次の節があります。

- [チュートリアルを使用する準備](#)
- [チュートリアルの手順の概要](#)
- [サーバーの起動](#)
- [サーバーの起動の確認](#)
- [loadbalancer.xml ファイルの作成](#)
- [loadbalancer.xml ファイルへのクラスタの追加](#)
- [ロードバランスの設定](#)
- [loadbalancer.xml ファイルのサンプル](#)

チュートリアルを使用する準備

このマニュアルでは、クラスタ環境のセットアップ、その環境へのアプリケーションの配備、アプリケーションを使用したロードバランスと HTTP セッション持続性の機能のテスト手順について説明しています。このマニュアルには 2 つのチュートリアルがあります。「[エンタープライズ機能の設定のチュートリアル](#)」ではクラスタとロードバランサの設定について説明しています。「[クラスタ JSP サンプルアプリケーションのチュートリアル](#)」では、アプリケーションの配備と、ロードバランスとセッション持続性の機能の確認について説明しています。

この節には次の項目があります。

- [インストールの要件](#)
- [チュートリアルを使用する前の必要な手順](#)

インストールの要件

ここで例示しているインストールの設定は、このチュートリアルで使用するための、基本的な単一マシンの構成です。この設定は、必ずしも、実務の環境に適切なものではないかもしれません。Sun ONE Application Server の導入に先立ち、実際の要求事項について、慎重な検討が必要です。推奨される設定の詳細については、『Sun ONE Application Server システム配備ガイド』を参照してください。

このマニュアルに記載されているチュートリアルを実行するために、次の基本設定を使います。

- 1 つの Sun ONE Web Server
- 1 つのロードバランサプラグイン
- 1 つの Sun ONE Application Server (高可用性データベースを使うように設定された 2 つのサーバーインスタンスを含む)
- 2 つの高可用性データベース

注 このマニュアルでは、すべての手順で Sun ONE Web Server を使用することを前提としています。Apache Web Server を使用することもできます。Apache Web Server の使用方法の詳細については、『Sun ONE Application Server 管理者ガイド』を参照してください。

チュートリアルを使用する前の必要な手順

チュートリアルを始める前に、次の手順をすべて行なっておく必要があります。

表 2-1 チュートリアルを使用する準備の手順

手順	情報の参照先
1. Sun ONE Web Server をインストールする	『Sun ONE Web Server 6.1 Installation and Migration Guide』
2. 次のものを含む Sun ONE Application Server をインストールする	『Sun ONE Application Server インストールガイド』
• ロードバランサプラグイン	
• HADB	

表 2-1 チュートリアルを使用する準備の手順 (続き)

手順	情報の参照先
3. マシンに次の設定をする <ul style="list-style-type: none"> 共有メモリ rsh または ssh を使った通信 HADB を使うための環境変数 	『Sun ONE Application Server インストールガイド』
4. clsetup を実行して次の設定を行う <ul style="list-style-type: none"> ドメイン domain1 に 2 つのアプリケーションサーバーインスタンス (server1 と server2) を作成して設定する HADB を作成する HADB にセッション情報を格納するために必要なデータベーステーブルを作成する インスタンスに接続プールを作成する すべてのインスタンスに JDBC リソースを作成する アプリケーションサーバーインスタンスにセッション持続性情報を設定する アプリケーションサーバーインスタンスで高可用性を有効にする 	

チュートリアルの手順の概要

次の表には、チュートリアル内で必要となる、システム設定やアプリケーションの配備、Enterprise Edition の独自の機能の確認について、手順の概要を記載しています。左側の列には手順を示し、右側の列には該当の手順を実行するために必要な情報の参照先を示しています。

表 2-2 チュートリアルの手順	
手順	情報の参照先
1. 管理サーバーおよびアプリケーションサーバーインスタンスを起動する	27 ページの「サーバーの起動」
2. 管理サーバーおよびアプリケーションサーバーインスタンスが稼働中であることを確認する	29 ページの「サーバーの起動の確認」
3. loadbalancer.xml ファイルを作成する	35 ページの「loadbalancer.xml ファイルの作成」
4. loadbalancer.xml のクラスタを設定する	36 ページの「loadbalancer.xml ファイルへのクラスタの追加」
5. loadbalancer.xml ファイルにロードバランスを設定する	39 ページの「ロードバランスの設定」
6. 設定した loadbalancer.xml ファイルをサンプルの loadbalancer.xml ファイルと照らし合わせて検証する	43 ページの「loadbalancer.xml ファイルのサンプル」
7. cladmin を使ってクラスタ内のインスタンスにサンプルアプリケーションを配備する	47 ページの「クラスタ JSP のサンプルアプリケーションのクラスタへの配備」
8. cladmin を使ってアプリケーションサーバーインスタンスを起動する	51 ページの「cladmin を使ったアプリケーションサーバーインスタンスの起動」
9. アプリケーションが正常に配備されたことを確認する	52 ページの「アプリケーション配備の確認」
10. サンプルアプリケーションを loadbalancer.xml 内のクラスタ情報に追加する	59 ページの「クラスタへのサンプルアプリケーションの追加」
11. 変更を適用し、Web Server を再起動する	60 ページの「設定の変更の適用と Web サーバーの再起動」
12. サンプルを実行する	61 ページの「アプリケーションの実行」

表 2-2 チュートリアルの手順 (続き)

手順	情報の参照先
13. ロードバランスを確認する	66 ページの「ロードバランスの確認」
14. セッション持続性を確認する	70 ページの「HTTP セッションの持続性の確認」
15. サーバーインスタンスを休止する	71 ページの「サーバーインスタンスの休止」

サーバーの起動

このマニュアルのチュートリアルを使用するには、クラスタ内の管理サーバーとすべてのアプリケーションサーバーインスタンスを起動する必要があります。Sun ONE Web Server も稼働させる必要があります。

すべてのサーバーインスタンスと管理サーバーが 1 つのドメイン内にあるときに管理サーバーとアプリケーションサーバーインスタンスを起動するもっとも簡単な方法は、asadmin ユーティリティの start-domain コマンドを使うことです。

この節には次の項目があります。

- [PATH 変数の設定](#)
- [asadmin start-domain の実行](#)

PATH 変数の設定

asadmin の実行に先立ち、PATH 変数を設定して、簡単に使用できるようにできます。

PATH 変数を設定すれば、asadmin、cladmin、および asant (アプリケーションサーバーの Ant ユーティリティ) を、どこからでもコマンド行で実行できます。PATH 環境変数に次のディレクトリを追加します。

```
install_dir/bin
```

Application Server の bin ディレクトリをログインプロファイルに追加すると、環境の PATH 設定がログイン時に自動的に追加されます。

たとえば、.cshrc ファイルに次のように追加します。

```
set path=( /opt/SUNWAppserver7/bin)
```

ファイルを保存し、source コマンドを使用して設定を有効にします。

```
source .cshrc
```

コマンドプロンプトで `asadmin` ユーティリティを実行することで、変更内容をテストできます。次のように入力します。

```
asadmin
```

実行結果が、次のように表示されます。

```
Use "exit" to exit and "help" for online help
```

```
asadmin>
```

`exit` と入力して、`asadmin` インタフェースを終了します。

コマンドが見つからない場合は、次のようにします。

- `PATH` 設定が正しく更新されていることを確認します。
- `.cshrc` ファイルなどのログインファイルを更新した場合には、その設定が有効になっていることを確認します。新しい端末ウィンドウを起動する必要がある場合もあります。

`PATH` 環境変数を設定しない場合でも、コマンドとユーティリティが置かれているディレクトリ (`install_dir/bin`) からそれらを実行できます。たとえば、次のようにします。

```
cd /opt/SUNWappserver7/bin
```

```
./asadmin
```

asadmin start-domain の実行

管理サーバーとすべてのアプリケーションサーバーインスタンスを起動するには、`asadmin start-domain` コマンドを実行します。コマンドプロンプトに次のように入力します。

```
asadmin start-domain --domain domain1
```

`domain1` は、サーバーをインストールして `clsetup` コマンドを実行したときに設定されたデフォルトのドメインです。

注

このマニュアルで示している設定では、すべてのインスタンスが同じドメイン内にあります。サーバーインスタンスが同じドメイン内でない場合は、使用するドメインごとに `asadmin start-domain` を実行する必要があります。いったんクラスタを作成したら、`cladmin start-instance` を使ってクラスタ内のすべてのインスタンスを起動することもできます。`cladmin` 構文については、[48 ページの「cladmin コマンドの構文」](#)を参照してください。

次のコマンドを使って、最初に設定したドメインの管理サーバーとアプリケーションサーバーインスタンスを両方とも停止します。

```
asadmin stop-domain --domain domain1
```

domain1 は、アプリケーションサーバーのインストール時に定義された管理ドメインの名前です。

すべての asadmin コマンドの完全なリストは、asadmin のヘルプを参照してください。ヘルプを表示するには、コマンドプロンプトに次のように入力します。

```
asadmin
```

PATH 変数が正しく設定されている場合、プロンプトが「asadmin」に変わります。ここで help と入力すると、asadmin のコマンドの一覧が表示されます。また、asadmin の特定のコマンドのヘルプを表示するには、「asadmin」のプロンプトから、次のように、コマンドに help オプションを付けて入力します。次に例を示します。

```
asadmin> start-domain --help
```

サーバーの起動の確認

管理サーバーとアプリケーションサーバーインスタンスを起動したら、起動が正常であることを確認します。具体的な手順については、次に示す節で説明します。

- [管理サーバーの起動の確認](#)
- [サーバーインスタンスの起動の確認](#)
- [HTTP サーバーへのアクセスによるインスタンスの確認](#)

管理サーバーの起動の確認

次の項目で説明するように、Sun ONE Application Server の管理インタフェースにアクセスするか、管理サーバーのイベントログを確認することによって、管理サーバーが起動したことを確認できます。

- [管理インタフェースへのアクセス](#)
- [管理サーバーのイベントログの表示](#)

管理インタフェースへのアクセス

Sun ONE Application Server の Web ベースの管理インタフェースにアクセスして、管理サーバーが正常に起動したことを確認できます。このインタフェースを使って、Sun ONE Application サーバーインスタンスを管理できます。ただし、クラスタ化されたインスタンスの場合は `cladmin` コマンドを使うほうが便利で、エラーが発生しにくくなります。

注	管理インタフェースと互換性のあるブラウザのリストについては、『Sun ONE Application Server Platform Summary』を参照してください。
---	---------------------------------------------------------------------------------------

管理インタフェースにアクセスするには、次の手順に従います。

1. ブラウザウィンドウを開き、管理サーバーのポートを指定します。

インストール時に管理サーバーのデフォルト番号は **4848** に指定されます。インストール時にこのポートが使用中か、別のポート番号を選択した場合は、そのポート番号を指定します。

次の **4848** の代わりにほかの番号を指定してください。

`http://test.sun.com:4848`

2. 製品のインストール時に指定した管理ユーザー名およびパスワードを使って管理インタフェースにサインインします。

ヒント	ユーザー名やパスワードを忘れてしまった場合： インストール時に設定した、管理サーバーのユーザー名を思い出せない場合は、ユーザー名として <code>admin</code> と入力してみてください。これは、インストール時にサーバー設定ダイアログで指定されているデフォルトのユーザー名です。
-----	---------------------------------------------------------------------------------------------------------------------------------------------------------

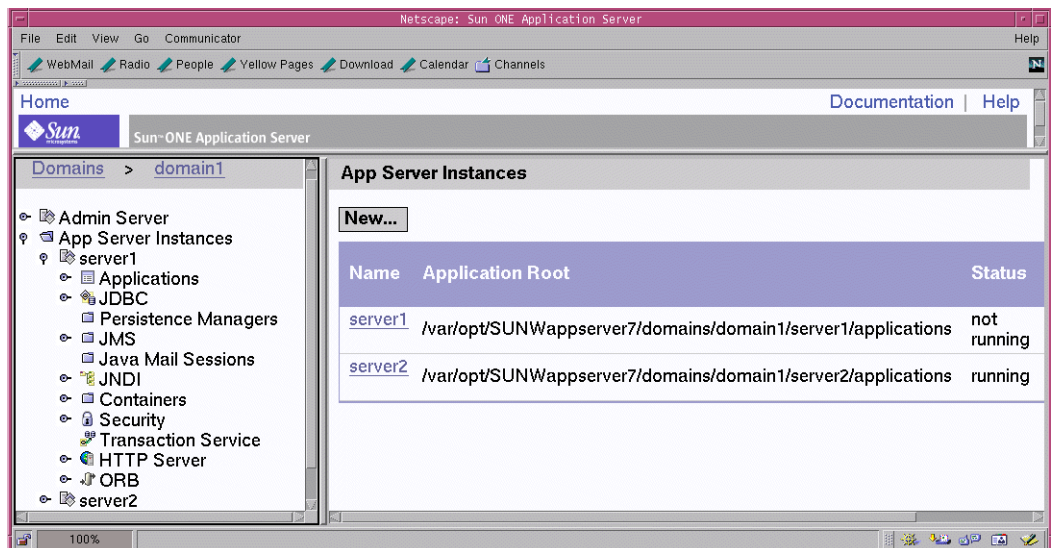
管理者のパスワードを思い出せない場合は、`install_config_dir` にある、サーバーをインストールしたときに作成された `clpassword.conf` ファイルを確認してください。このファイルにアクセスするにはルートになる必要があります。

ヒント

ポートにアクセスできない場合：管理サーバーの管理インタフェースに接続しようとしても拒否される場合は、管理サーバーが稼働していない可能性があります。起動手順と管理サーバーのログファイルの内容をチェックして、サーバーが稼働していない理由を確認してください。ログファイルの表示方法については、31 ページの「管理サーバーのイベントログの表示」を参照してください。

正しく認証されると、管理インタフェースの最初の画面が表示されます。

図 2-1 管理インタフェース



左側のペインの項目をクリックすると、右側のペインに対応するページが表示されます。

管理サーバーのイベントログの表示

管理サーバーのイベントログでサーバーの起動メッセージを表示することもできます。イベントログファイルを開いて表示するには、次の手順に従います。

1. 管理サーバーのサーバーログに移動します。

`domain_config_dir/domain1/admin-server/logs/`

2. エディタで `server.log` を開きます。

サーバーが起動済みの場合は、ログファイルの最後に `successful server startup` という情報が記録されています。

```
[20/Feb/2003:00:06:00] INFO ( 4318):CORE3274: successful server startup
```

正常な起動を示すメッセージが表示されない場合、管理サーバーが起動手順を完了する前にイベントログファイルを開いてしまった可能性があります。ログファイルを閉じてから開き直し、最新のイベントメッセージを確認してください。

`tail -f` コマンドを使ってサーバーログを表示することもできます。

```
tail -f server.log
```

`-f` オプションにより `tail` コマンドの実行が保持されるため、新しいログエントリはファイルに書き込まれたとおりに表示されます。

管理インタフェースから、管理サーバーとアプリケーションサーバーインスタンスのイベントログファイルにアクセスすることもできます。左側のペインでサーバー名 (管理サーバーまたはアプリケーションサーバーインスタンス) をクリックし、右側のペインで「Logging (ログ)」タブをクリックします。

詳細は、『Sun ONE Application Server 管理者ガイド』を参照してください。

サーバーインスタンスの起動の確認

`clsetup` を実行したあとは、`domain1` に `server1` と `server2` という 2 つのサーバーインスタンスがあるはずです。この節では、これらのインスタンスが存在することと、稼働していることを確認する方法について説明します。

- [asadmin を使ったインスタンスの確認](#)
- [HTTP サーバーへのアクセスによるインスタンスの確認](#)

asadmin を使ったインスタンスの確認

確認には `asadmin list-instances` コマンドを使います。

1. コマンドプロンプトに `asadmin` と入力して、`asadmin` を起動します。

```
asadmin
```

プロンプトは `asadmin>` になります。

2. `asadmin` プロンプトで `list-instances` コマンドを使って、現在のすべてのインスタンスを表示します。

```
list-instances
```

このコマンドにより、インスタンスが一覧表示され、それらが現在稼働中かどうかが表示されます。`server1` と `server2` という 2 つのインスタンスが稼働中として表示されます。

HTTP サーバーへのアクセスによるインスタンスの確認

アプリケーションサーバーインスタンスが起動したかどうかを確認するもう 1 つの方法は、Web ブラウザからインスタンスの HTTP サーバーのトップページにアクセスすることです。

ブラウザで次の URL にアクセスします。

```
http://server_instance:server_instance_port_number
```

`server_instance_port_number` は、インストール時またはサーバーインスタンスの作成時に指定した HTTP サーバーのポート番号です。

ヒント HTTP サーバーのポート番号がわからないときは、アプリケーションサーバーインスタンスの設定ファイルを調べます。

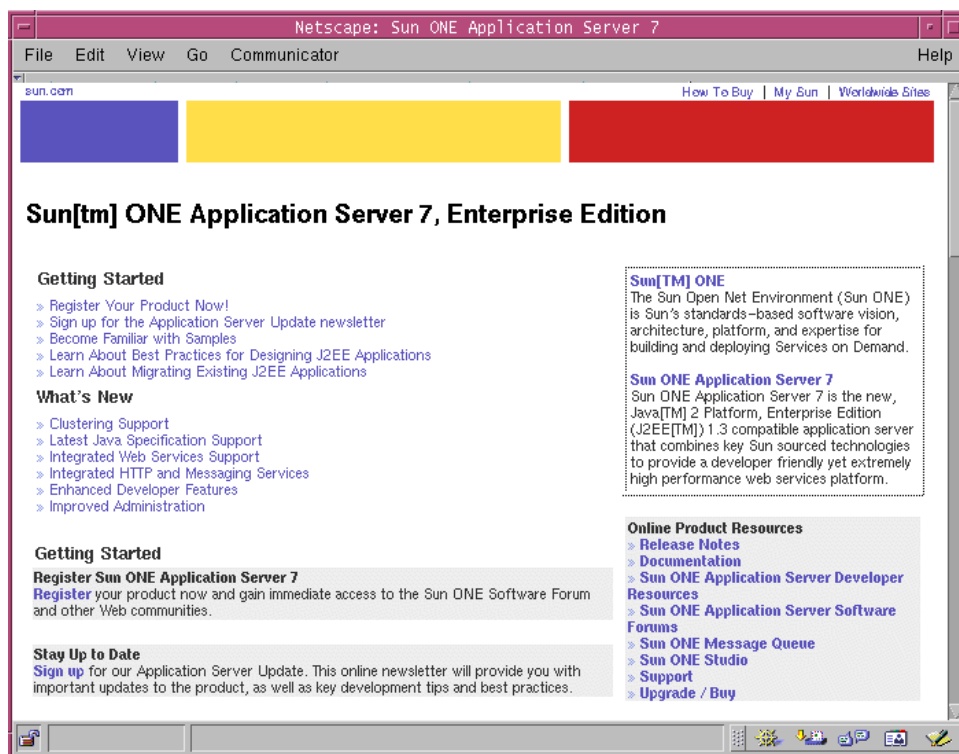
1. `domain_config_dir/domain1/server_instance/config/` ディレクトリに移動して、`server.xml` ファイルをエディタで開きます。
2. たとえば次のように、`http-listener` 要素を探します。

```
http-listener id="http-listener-1" address="0.0.0.0"
port="81"...
```

この例では、ポート 81 が使用されている HTTP ポート番号です。

アプリケーションサーバーのインスタンスが正常に稼働している状態では、HTTP サーバーのデフォルトのトップページがブラウザに表示されます。

図 2-2 HTTP サーバーのトップページ



ヒント HTTP サーバーのトップページ: HTTP サーバーのトップページは、アプリケーションサーバーインスタンスのデフォルトのドキュメントディレクトリにある、`index.html` という HTML ファイルです。アプリケーションサーバーインスタンスのデフォルトのドキュメントルートは、そのインスタンスの `server.xml` 設定ファイルに設定されています。インストールが完了すると、インスタンスのドキュメントルートである `server1` は `domain_config_dir/domain1/server1/docroot/` に設定されます。トップページはこの場所にあります。

loadbalancer.xml ファイルの作成

コンポーネントのインストールと clseup の実行が完了し、管理サーバーを起動したなら、インストール設定の準備が完了したことになります。設定の最初の手順は、クラスタの作成です。

このマニュアルのクラスタの設定では1つのロードバランサプラグインを使用するため、Web Server の loadbalancer.xml 設定ファイルでクラスタを定義する必要があります。このファイルは、Web Server の設定ファイルディレクトリにあります。loadbalancer.xml ファイルはデフォルトでは存在しません。このファイルは作成する必要があります。

loadbalancer.xml ファイルを作成するには、次の手順に従います。

1. サンプルの loadbalancer.xml ファイルを見つけます。このファイルは loadbalancer.xml.example という名前で、Web Server の config ディレクトリにあります。

デフォルトでは、このディレクトリは次のとおりです。

/usr/iplanet/servers/https-server_name/config

2. loadbalancer.xml.example のコピーを loadbalancer.xml として保存します。

このコピーを編集して、クラスタを作成してロードバランスを機能させる情報など、環境のための情報を追加します。

サンプルファイルは次のとおりです。

コード例 2-1 loadbalancer.xml.example ファイル

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE Application
Server 7.0//EN" "sun-loadbalancer_1_0.dtd">

<loadbalancer>

  <cluster name="cluster1">

    <instance name="instance1" enabled="true" disable-timeout-in-minutes="60"
listeners="<REPLACE_WITH_LISTENER1> <REPLACE_WITH_LISTENER2>"/>

    <web-module context-root="/abc" enabled="true"
disable-timeout-in-minutes="60" enabled="true" />

    <health-checker url="/" interval-in-seconds="10" timeout-in-seconds="30" />

  </cluster>

  <property name="reload-poll-interval-in-seconds" value="60"/>

</loadbalancer>
```

```
<property name="response-timeout-in-seconds" value="30"/>
<property name="https-routing" value="true"/>
<property name="require-monitor-data" value="false"/>
</loadbalancer>
```

サンプルファイルは sun-loadbalancer_1_0.dtd を指し、Web Server の config ディレクトリにもあります。

loadbalancer.xml ファイルへのクラスタの追加

次に、クラスタ (このチュートリアルでは cluster1) を loadbalancer.xml ファイルに追加します。このクラスタには、2 つの Sun ONE Application Server またはサーバーインスタンスがあります。

新たに作成した loadbalancer.xml を編集して、インスタンスと URL を HTTP リスナーに組み込みます。

1. Web Server の config ディレクトリに移動し、loadbalancer.xml をテキストエディタで開きます。

サンプルファイルには、cluster1 というクラスタがあらかじめ組み込まれています。

```
<cluster name=cluster1>
```

2. clsetup を使って作成した 2 つのアプリケーションサーバーインスタンス (server1 と server2) を cluster1 に追加します。

インスタンスのそれぞれに対して、クラスタのインスタンスのサブ要素を作成します。インスタンスは有効 (true) にする必要があります。また、URL を HTTP リスナーに追加することも必要です。デフォルトのサーバーインスタンス (server1) と、追加した 2 番目のサーバーインスタンス (server2) の場合、それぞれのインスタンスには HTTP リスナーは 1 つだけしかありません。別の環境ではもっと多い場合があります。

サンプルの loadbalancer.xml ファイルには、編集可能な次の行があります。

```
<instance name="instance1" enabled="true"
disable-timeout-in-minutes="60"
listeners="<REPLACE_WITH_LISTENER1> <REPLACE_WITH_LISTENER2>"/>
```

3. このチュートリアルでは、「instance1」を「server1」と置き換えます。

4. <REPLACE_WITH_LISTENER1> を、次のようなサーバーインスタンスのリスナー URL と置き換えます。

```
http://test.sun.com:81
```

リスナー URL は、サーバー名とポートで構成されます。リスナーのセキュリティが有効になっている場合、URL は HTTPS で始まります。リスナーがセキュリティ保護されていない場合、URL は HTTP で始まります。Sun ONE Application Server の各インスタンスの HTTP リスナーに関する情報を検索するには、次のようにします。

- a. 管理インタフェースでインスタンス名をクリックして、ツリー表示を展開します。
 - b. 「HTTP Server (HTTP サーバー)」をクリックして展開します。
 - c. 「HTTP Server (HTTP サーバー)」の下にある「HTTP Listener (HTTP リスナー)」をクリックします。
 - d. HTTP リスナーページには、ポート、サーバー名、セキュリティが有効かどうかが表示されます。
5. 変更を loadbalancer.xml に保存します。この時点で、ファイルには次のような行があります。

```
<instance name="server1" enabled="true"
disable-timeout-in-minutes="60"
listeners="http://test.sun.com:81"/>
```

6. disable-timeout-in-minutes の値を 5 に変更します。

disable timeout in minutes はインスタンスレベルの休止時間です。この間に、サーバーインスタンスのために終了するセッション時間をみておきます。サーバーは稼働を続けますが、新しいセッションになるような要求は送信されません。ただし、サーバーは既存のセッションに対する要求は受け入れます。実際には、この数はかなり大きくなる場合があります。例で指定した時間は 60 分です。ただし、このチュートリアルではサーバーをシャットダウンするため、タイムアウトは短く設定してください。

```
<instance name="server1" enabled="true"
disable-timeout-in-minutes="5"
listeners="http://test.sun.com:81"/>
```

7. 2 番目のインスタンスをクラスタ要素に追加します。これは、上記の行をコピーして既存の行の下に貼り付けてから、サーバー名とリスナー URL を変更することによって実行できます。次に例を示します。

```
<instance name="server2" enabled="true"
disable-timeout-in-minutes="5"
listeners="http://test.sun.com:82"/>
```

8. 変更を保存します。

この時点で、loadbalancer.xml ファイルは次のようになっているはずです。

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE Application
Server 7.0//EN" "sun-loadbalancer_1_0.dtd">

<loadbalancer>

  <cluster name="cluster1">

    <instance name="server1" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:81"/>

    <instance name="server2" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:82"/>

    <web-module context-root="/abc" enabled="true"
disable-timeout-in-minutes="60" enabled="true" />

    <health-checker url="/" interval-in-seconds="10" timeout-in-seconds="30" />

  </cluster>

  <property name="reload-poll-interval-in-seconds" value="60"/>
  <property name="response-timeout-in-seconds" value="30"/>
  <property name="https-routing" value="true"/>
  <property name="require-monitor-data" value="false"/>

</loadbalancer>
```

後述の手順で、配備したアプリケーションに関連する情報 (サンプルでは「web-module」で始まる行) を追加します。

注	loadbalancer.xml ファイルのアプリケーションサーバーインスタンスを有効にすることは、アプリケーションサーバーインスタンスを起動することとは異なります。サーバーが稼働中で、クラスタに追加されている場合でも、instance 要素の enabled 属性を true に設定して loadbalancer.xml を有効にするまで、ロードバランサは要求をサーバーインスタンスに配信しません。
---	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ロードバランスの設定

loadbalancer.xml ファイルを作成し、クラスタを追加したら、loadbalancer.xml ファイルを編集してロードバランサ設定を組み込みます。

この節には次の項目があります。

- [ヘルスチェッカーの設定](#)
- [ロードバランサの監視の有効化](#)
- [ロードバランサのその他のプロパティ](#)

ヘルスチェッカーの設定

ヘルスチェッカーが有効な場合、ロードバランサは定期的に、ダウンしているというフラグが立てられたすべての Sun ONE Application Server インスタンスをチェックして、稼働したかどうかを確認します。ダウンしているとマークされたアプリケーションサーバーインスタンスが稼働すると、そのインスタンスは稼働インスタンスのリストに追加され、要求がもう一度配信されます。

ヘルスチェッカーは、loadbalancer.xml の health-checker 要素で設定します。ヘルスチェッカーで ping してサーバーが稼働しているかどうかを確認するリスナー URL と、ヘルスチェッカーでサーバーを ping する頻度、サーバーがダウンしているとマークするまでにヘルスチェッカーが応答を待機する時間を設定します。

サンプルの loadbalancer.xml ファイルには次の行が組み込まれています。このマニュアルで説明しているタスクを完了させるためには、編集は必要ありません。

```
health-checker url="/" interval-in-seconds="10"
timeout-in-seconds="30" />
```

- health-checker url は、サーバーインスタンスが稼働しているかどうか (つまり、応答するかどうか) を ping で確認する URL を指定します。例では "/" の URL です。たとえば、リスナー URL が http://www.example.com:80 の場合、"/" のヘルスチェッカー URL は http://www.example.com:80/ を ping します。このマニュアルの目的のために、この値は "/" のままにします。
- interval-in-seconds 属性は、ロードバランサのヘルスチェックの時間間隔を指定します。この値はデフォルトの 10 から変更する必要はありません。
- timeout-in-seconds 属性は、サーバーインスタンスが稼働しているとみなすために、ロードバランサが ping されたサーバーインスタンスから応答を受け取る必要がある時間間隔を指定します。この値はデフォルトの 30 から変更する必要はありません。

たとえば、デフォルト値を使うと、ヘルスチェッカーの動作は次のようになります。

1. ヘルスチェッカーは、ヘルスチェッカー URL を使ってダウンしているリスナーを ping します。
2. リスナーごとに、応答を 30 秒間 (timeout-in-seconds) 待機します。
サーバーが 30 秒以内に応答すると、リスナーに稼働しているというマークが付けられます。サーバーが応答しないと、ヘルスチェッカーはサーバーがまだダウンしているとみなします。
3. ヘルスチェッカーは、ヘルスチェックの次のサイクルが始まるまで、10 秒間 (interval-in-seconds) 待機します。
ヘルスチェッカーが起動したときにダウンしているインスタンスがない場合は、最初の 2 つの手順がスキップされます。

ロードバランサの監視の有効化

ロードバランサプラグインは、Web Server のログメカニズムを使ってログメッセージを書き込みます。監視が有効になっている場合は、Web Server のログファイルに次の情報が記録されます。

- 各要求の開始と停止の情報
- 要求がダウンしているインスタンスから稼働しているインスタンスにフェイルオーバーされたときのフェイルオーバー要求情報
- ヘルスチェッカーの各サイクルの最後に記録される、ダウンしているインスタンスのリスト

警告

ロードバランサプラグインでログ機能が有効になっているときに、Web Server のログレベルが DEBUG に設定されているか、詳細メッセージを印刷するように設定されている場合、ロードバランサは Web Server のログファイルに HTTP セッション ID を書き込みます。このため、ロードバランサプラグインをホストする Web Server が DMZ にある場合、本稼働環境では DEBUG や同様のログレベルを使用しないことをお勧めします。

DEBUG ログレベルを使う必要がある場合は、loadbalancer.xml で require-monitor-data プロパティを false に設定して、ロードバランサのログ機能をオフにしてください。

ロードバランスを監視するには、次のことを行う必要があります。

- loadbalancer.xml で監視を有効にする
- 詳細ログを使用するように Web Server を設定する

loadbalancer.xml で監視を有効にするには、次の手順に従います。

1. loadbalancer.xml をテキストエディタで開きます。
2. 次の行を見つけてください。これはサンプルの loadbalancer.xml ファイルの一部です。

```
<property name="require-monitor-data" value="false"/>
```

3. 「false」を「true」に変更して、監視を有効にします。
4. 変更を保存し、ファイルを終了します。

さらに、Web Server のデフォルトのログレベルを変更するには、次の手順に従います。

Sun ONE Web Server の詳細ログを有効にする場合

1. Web Server の config ディレクトリに移動します。Sun ONE Web Server がデフォルトの場所にインストールされている場合、パスは次のとおりです。

```
/web_server_install_dir/https-server_name/config
```

2. 編集のために magnus.conf ファイルを開きます。
3. 次の行を追加します。

```
LogVerbose on
```

4. ファイルに対する変更を保存します。
5. Web Server を再起動して変更を適用します。

ロードバランサのその他のプロパティ

例の `loadbalancer.xml` には、次のような追加のロードバランサプロパティが含まれています。

```
<property name="reload-poll-interval-in-seconds" value="60"/>
<property name="response-timeout-in-seconds" value="30"/>
<property name="https-routing" value="true"/>
```

このチュートリアルでは、デフォルト値を変更する必要はありません。これらのプロパティの詳細については、次の節を参照してください。

- [再読み込みのポーリング間隔を使った動的再設定](#)
- [応答タイムアウト](#)
- [HTTPS ルーティング](#)

再読み込みのポーリング間隔を使った動的再設定

初期設定のあと、ロードバランサプラグインはその設定に対する変更を検出し、それを自動的に読み込みます。変更は、`loadbalancer.xml` ファイルのタイムスタンプを調べることによって検出されます。タイムスタンプが変更された場合、ロードバランサは自動的に再設定されます。再読み込みのポーリング間隔では、ロードバランサがタイムスタンプをチェックする頻度を指定します。

注

- `loadbalancer.xml` ファイルに対する変更が `sun-loadbalancer_1_0.dtd` ファイルに示されているような正しい形式になっていないと、再設定は失敗します。再設定に失敗すると、**Web Server** のエラーログファイルに記録されます。ロードバランサは、すでにメモリに読み込まれている古い設定を引き続き使用します。
- ロードバランサは、再設定しようとしているときにハードディスク読み取りエラーが発生した場合、現時点でメモリ内にある設定を使用します。ディスク読み取りエラーが発生した場合は、**Web Server** のエラーログファイルに警告メッセージが記録されます。

Sun ONE Web Server のエラーログファイルは、
`web_server_install_dir/web_server_instance/logs/` にあります。

応答タイムアウト

応答タイムアウトは、サーバーが要求に応答するためにインスタンスを待機する時間を指定します。指定した秒数の間、インスタンスが応答しなかった場合、その秒数の経過後にブラウザにエラーメッセージが送信されます。

HTTPS ルーティング

HTTPS ルーティングが無効になっているとき、loadbalancer.xml ファイルのリストに指定されている HTTPS リスナーは無視され、ロードバランスには HTTP リスナーのみが使用されます。HTTPS ルーティングが有効になっているとき、HTTPS 要求は HTTPS ポートのためにフェイルオーバーされます。

loadbalancer.xml ファイルのサンプル

次のサンプルの loadbalancer.xml ファイルには、2つのインスタンスの1つのクラスタが含まれています。loadbalancer.xml ファイルの構文をこの例と照らし合わせてチェックしてください。

コード例 2-2 loadbalancer.xml ファイル

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE Application
Server 7.0//EN" "sun-loadbalancer_1_0.dtd">

<loadbalancer>
  <cluster name="cluster1">
    <instance name="server1" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:81"/>
    <instance name="server2" enabled="true" disable-timeout-in-minutes="5"
listeners="http://test.sun.com:82"/>
    <web-module context-root="/abc" enabled="true"
disable-timeout-in-minutes="60" enabled="true" />
    <health-checker url="/" interval-in-seconds="10" timeout-in-seconds="30" />
  </cluster>
  <property name="reload-poll-interval-in-seconds" value="60"/>
  <property name="response-timeout-in-seconds" value="30"/>
  <property name="https-routing" value="true"/>
  <property name="require-monitor-data" value="true"/>
</loadbalancer>
```

これでエンタープライズ機能の設定が完了したので、サンプルアプリケーションを配備して実行することができます。[第3章「クラスタ JSP サンプルアプリケーションのチュートリアル」](#)に進んでください。

クラスタ JSP サンプルアプリケーションの チュートリアル

この章には次の節があります。

- クラスタ JSP サンプルアプリケーションのチュートリアルを使用する準備
- クラスタ JSP のサンプルアプリケーションのクラスタへの配備
- `cladmin` を使ったアプリケーションサーバーインスタンスの起動
- アプリケーション配備の確認
- クラスタへのサンプルアプリケーションの追加
- 設定の変更の適用と Web サーバーの再起動
- アプリケーションの実行
- ロードバランスの確認
- HTTP セッションの持続性の確認
- サーバーインスタンスの休止

クラスタ JSP サンプルアプリケーションのチュートリアルを使用する準備

このチュートリアルを使用するためには、次の準備が必要になります。

- 第2章「エンタープライズ機能の設定のチュートリアル」のすべてのチュートリアル手順が正常に完了していること
- 必要に応じて、サンプルアプリケーションのコピーを作成すること

アプリケーションサーバーのインストールをほかのユーザーと共有しているか、システムユーザー ID にアプリケーションサーバーがインストールされている場所への書き込み権がない場合は、サンプルのコピーを作成する必要があります。

サンプルをコピーするには、*install_dir/samples* ディレクトリを、ユーザー ID に書き込み権がある場所にコピーします。

次に例を示します。

```
cp -r install_dir/samples user_sample_directory
```

samples ディレクトリには、このマニュアルで使用するサンプルが含まれているサブディレクトリ *ee-samples* があります。

このマニュアルの手順に従ってサンプルのコピーを使用する場合、*install_dir/samples/* と表記されるディレクトリは、サンプルアプリケーションの専用コピーが保存されているディレクトリを指します。

クラスタ JSP のサンプルアプリケーションのクラスタへの配備

クラスタ JSP のサンプルアプリケーションは、JSP に対する要求をクラスタ内のアプリケーションサーバー間でロードバランスを行う方法を示します。アプリケーションサーバーが停止した場合でも、HTTP セッション情報を持続させる方法を示すショッピングカートがあります。

このサンプルアプリケーションには、そのままの状態でも Web アプリケーションの配備に使用できる WAR ファイルが付属しています。このため、配備にあたっては、かならずしもコンパイルやアセンブルは必要ではありません。

注

セッションフェイルオーバーを使用するアプリケーションは、すべて配布可能でなければなりません。clusterjsp サンプルアプリケーションは、その web.xml ファイルにあるかどうかを確認できるよう、あらかじめ配布可能になっています。

これを確認するには、次のようにします。

1. クラスタ JSP のサンプルの src ディレクトリに移動します。

```
cd /install_dir/samples/ee-samples/clusterjsp/src
```

2. web.xml ファイルを調べます。次のコードを確認します。

```
<web-app>
    <display-name>clusterjsp</display-name>
    <distributable/>
</web-app>
```

web-app 要素には、指定した distributable サブ要素があります。アプリケーションコードを変更する必要はありません。

サンプルアプリケーションをクラスタに配備するには、最初にクラスタ内のすべてのインスタンスに配備する必要があります。

cladmin コマンドを使って、アプリケーションをクラスタ内のすべてのインスタンスに、同時に配備します。cladmin コマンドにより、クラスタ内のすべてのインスタンスで同時に asadmin コマンドが実行されます。cladmin コマンドは *install_dir/bin* にあります。

この節には次の項目があります。

- [cladmin コマンドの入力ファイル](#)
- [cladmin コマンドの構文](#)
- [cladmin deploy の実行](#)

- [cladmin](#) コマンドでサポートされている [asadmin](#) コマンド
- [要件と制限事項](#)

cladmin コマンドの入力ファイル

cladmin コマンドは、clinstance.conf と clpassword.conf という、2つの入力ファイルを使います。

- `clinstance.conf`: このファイルには、クラスタの一部であるアプリケーションサーバーインスタンスに関する情報が含まれています。
- `clpassword.conf`: このファイルには管理サーバーのパスワードが含まれています。標準インストール時にこの正しいパスワードが事前設定されています。

これらのファイルは `install_config_dir` にあります。これらのファイルは以前に `clsetup` を実行するために使用したので、環境に適した値が含まれているはずです。

`cladmin deploy` を実行するために指定する必要がある値の多くはこれらのファイルに含まれているため、`deploy` コマンドは、最低限のオプションを付けるだけで実行できます。

cladmin コマンドの構文

cladmin コマンドの構文は次のとおりです。

```
cladmin [--help] [--instancefile instance_file_location] [--passwordfile  
password_file_location] asadmin_command
```

各変数の意味は次のとおりです。

- `instance_file_location` は、入力ファイル `clinstance.conf` の場所
- `password_file_location` は、入力ファイル `clpassword.conf` の場所
- `asadmin_command` は、クラスタ内のアプリケーションサーバーインスタンスで実行する `asadmin` コマンド

入力ファイルがデフォルトの場所である `/etc/opt/SUNWappserver7` にある場合は、`instancefile` オプションと `passwordfile` オプションを省略して、コマンドを次のように実行できます。

```
cladmin asadmin_command
```


cladmin deploy の実行

アプリケーションをクラスタ内のすべてのインスタンスに配備するには、次のように入力します。

```
cladmin deploy filepath
```

サンプルに付属の WAR ファイルを使って、サンプルを Web アプリケーションとして配備できます。*filepath* は、WAR ファイルへのパスです。

たとえば、クラスタ JSP アプリケーションがデフォルトの場所にある場合は、次のように入力します。

```
cladmin deploy /opt/SUNWappserver7/samples/ee-samples/clusterjsp/clusterjsp.war
```

PATH 変数が設定されている場合は、WAR ファイルが置かれているディレクトリに移動して、そこからアプリケーションを配備することもできます。次に例を示します。

```
cd install_dir/samples/ee-samples/clusterjsp
```

```
cladmin deploy clusterjsp.war
```

cladmin deploy コマンドを実行すると、アプリケーションがクラスタ内の各インスタンスに配備されるときにメッセージが表示されます。

エラーが発生した場合は、/var/tmp/cladmin.log にあるログファイルで詳細を確認してください。

注	ルートとして実行しているときに、ルートの PATH 変数が設定されていない場合は、任意のディレクトリから cladmin を実行することができない場合があります。 <i>install_dir/bin</i> ディレクトリに変更してから、コマンドプロンプトに <i>./cladmin asadmin_command</i> と入力します。
---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

cladmin コマンドでサポートされている asadmin コマンド

cladmin コマンドを使って、クラスタ内の各アプリケーションサーバーインスタンスに対して、次の asadmin コマンドを同時に実行できます。

表 3-1 cladmin コマンドでサポートされている asadmin コマンド

コマンド	用途
start-instance	アプリケーションサーバーインスタンスを起動する
stop-instance	アプリケーションサーバーインスタンスを停止する
deploy	指定したディレクトリ内の EJB、WEB、コネクタ、appclient、またはアプリケーションコンポーネントをアプリケーションサーバーインスタンスに配備する
undeploy	配備されたコンポーネントをアプリケーションサーバーインスタンスから削除する
create-jdbc-resource	アプリケーションサーバーインスタンスに JDBC リソースを作成する
create-jdbc-connection-pool	アプリケーションサーバーインスタンスに JDBC 接続プールを作成する
configure-session-persistence	アプリケーションサーバーインスタンスにセッション持続性を設定する
delete-jdbc-resource	アプリケーションサーバーインスタンスの JDBC リソースを削除する
delete-jdbc-connection-pool	アプリケーションサーバーインスタンスの JDBC 接続プールを削除する

cladmin コマンドの詳細については、『Sun ONE Application Server 管理者ガイド』を参照してください。

要件と制限事項

cladmin コマンドには、次のような要件と制限があります。

- クラスタに含まれるすべてのインスタンスには、管理者のための同一のユーザー名と同一のパスワードが必要になります。
- このコマンドの起動に先立ち、クラスタに含まれるすべてのアプリケーションサーバーインスタンスで、対応する管理サーバーを起動しておく必要があります。
- このコマンドの入力ファイルに指定した値は、クラスタ内のすべてのインスタンスに対し、一律に設定されます。cladmin コマンドは、各インスタンスを別の値で設定するようには設計されていません。たとえば、このコマンドでは、インスタンスごとに設定が異なる JDBC 接続プールは作成できません。

cladmin を使ったアプリケーションサーバーインスタンスの起動

サンプルアプリケーションを配備したら、サーバーインスタンスが稼働していることを確認します。

1. コマンドプロンプトに次のように入力します。

```
asadmin list-instances
```

このコマンドにより、インスタンスと、それらが稼働中かどうかが表示されます。

2. インスタンスが稼働していない場合は、次のように、cladmin を使って起動します。

```
cladmin start-instance
```

サーバーインスタンスが起動されるたびにメッセージが表示されます。

アプリケーション配備の確認

アプリケーションが配備されたら、それを Sun ONE Application Server インスタンスで実行して、配備をテストできます。サンプルアプリケーションを実行する手順は、次のとおりです。

1. ブラウザから次の URL にアクセスします。

`http://host:application_server_instance_port/clusterjsp`

次に例を示します。

`http://test.sun.com:81/clusterjsp`

2. このサンプルアプリケーションのページが表示されます。

図 3-1 クラスタ JSP のサンプルアプリケーションページ

Netscape: Cluster - Ha JSP Sample

File Edit View Go Communicator Help

Cluster - HA JSP Sample

HttpSession Information:

- Served From Server: **austen**
- Server Port Number: **\$1**
- Executed From Server: **austen**
- Executed Server IP Address: **192.18.151.14**
- Session ID: **d5bf99a19d794dad0aa7ebe11187**
- Session Created: Wed Aug 06 12:33:10 PDT 2003
- Last Accessed: Wed Aug 06 12:33:10 PDT 2003
- Session will go inactive in **1800 seconds**

Enter session attribute data:

Name of Session Attribute:

Value of Session Attribute:

Data retrieved from the HttpSession:

INSTRUCTIONS

- Add session data using the form. Upon pressing ADD SESSION DATA, the current session data will be listed.
- Click on RELOAD PAGE to display the current session data without adding new data.
- Click on CLEAR SESSION to invalidate the current session.

100%

エラーが発生した場合は、58 ページの「アプリケーションサーバーインスタンスへの配備に対するトラブルシューティング」を参照してください。

3. セッション属性フィールドに情報を入力して、「ADD SESSION DATA (セッションデータを追加)」をクリックします。

データが表示されます。

4. 2 番目のインスタンスに対する配備を確認するには、URL にその他のインスタンスのポートとホスト名を入力します。

`http://host:application_server_instance_port/clusterjsp`

次に例を示します。

`http://test.sun.com:82/clusterjsp`

これらの URL で、アプリケーションがアプリケーションサーバーインスタンスに配備されたことを確認します。アプリケーションは `loadbalancer.xml` 内のクラスタにまだ追加されていないため、まだ Web サーバーからは確認できません。

アプリケーションサーバーのサンプルの監視

アプリケーションサーバーインスタンスのイベントログファイルを調べることで、サンプルアプリケーションを監視できます。それぞれのアプリケーションサーバーインスタンスには、独自のイベントログがあります。

この節には次の項目があります。

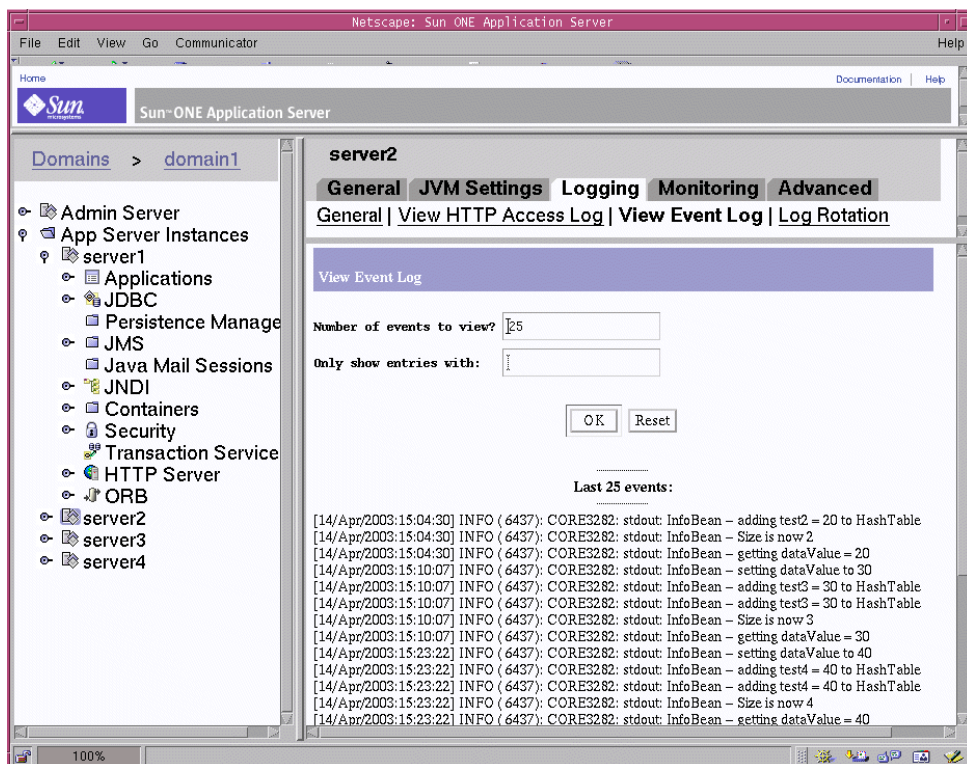
- [管理インタフェースによるログの表示](#)
- [tail コマンドによるログの表示](#)
- [イベントログのアプリケーション生成のメッセージ](#)
- [アクセスログのアプリケーション生成のメッセージ](#)

管理インタフェースによるログの表示

管理インタフェースを使ってサーバーインスタンスのログファイルを表示するには、次のようにします。

1. 管理インタフェースにアクセスします。
2. 左側のペインで、ログをチェックするインスタンスをクリックします。
3. 「Logging (ログ)」 タブをクリックします。
4. 「View Event Log (インスタンスのイベントログを表示)」 をクリックします。

図 3-2 インスタンスのイベントログを表示



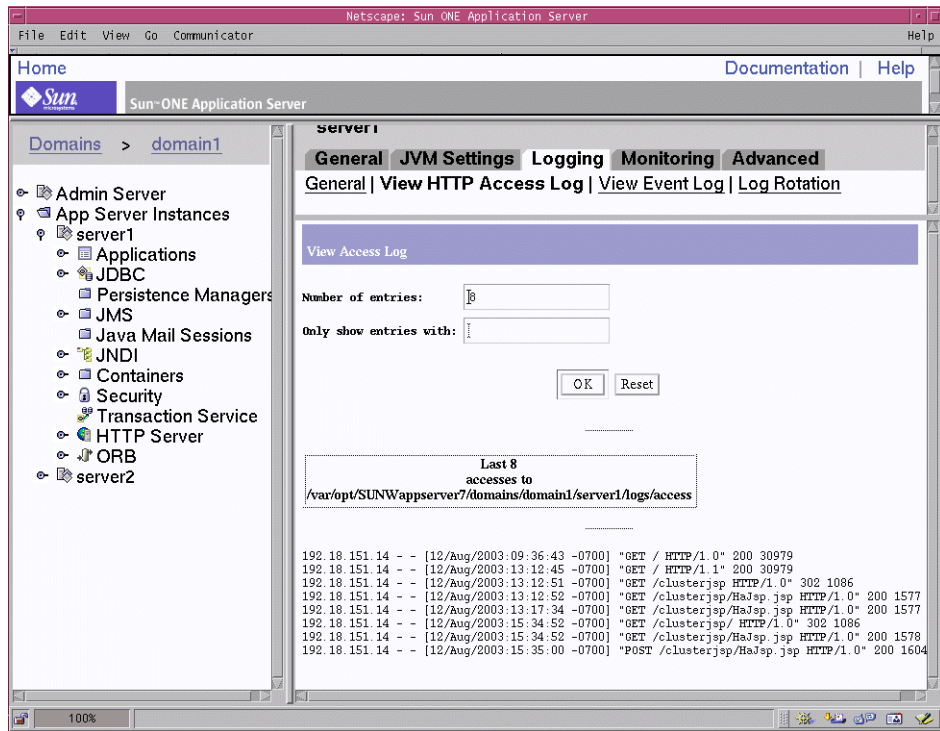
「OK (了解)」をクリックするとログの表示が更新されます。

25 項目より多くのログを表示するには、「Number of errors to view? (表示するイベント数)」に表示する項目数を入力し、「OK (了解)」をクリックして表示を更新します。

管理インタフェースを使ってアプリケーションサーバーインスタンスの HTTP アクセスログを表示するには、次のようにします。

1. 管理インタフェースにアクセスします。
2. 左側のペインで、ログをチェックするインスタンスをクリックします。
3. 「Logging (ログ)」タブをクリックします。
4. 「View HTTP Access Log (インスタンスの HTTP アクセスログを表示)」をクリックします。

図 3-3 インスタンスの HTTP アクセスログを表示



サーバーインスタンスの HTTP アクセスログには、サーバーインスタンスに対するサンプルアプリケーションへのアクセスが表示されます。

アクセスログの名前は `access` です。デフォルトの設定では、アクセスログファイルはサーバーイベントログと同じディレクトリに保存されます。

`domain_config_dir/domain1/server1/logs/`

tail コマンドによるログの表示

ログファイルを監視するために `tail -f` コマンドを使うと、ログが記録されるときに自動的にメッセージが表示されるようにできます。管理インタフェースでボタンをクリックして、ページ上の新しい情報を反映する必要があります。

`tail -f` コマンドを使用するには、次の手順に従います。

1. 監視するサーバーインスタンスのログディレクトリに移動します。次に例を示します。

```
cd domain_config/domain1/server1/logs/
```


2. サーバーイベントのログファイルで `tail` を実行します。

```
tail -f server.log
```

`-f` オプションにより `tail` コマンドの実行が保持されるため、新しいログエントリはファイルに書き込まれたとおりに表示されます。

イベントログのアプリケーション生成のメッセージ

アプリケーションが `stdout` または `stderr`、あるいはその両方に情報を書き込むと、同じ情報が **INFO** レベルのメッセージとしてサーバーインスタンスのイベントログファイルにも記録されます。このメッセージの ID は、それぞれ **CORE3282** (`stdout`) および **CORE3283** (`stderr`) になります。クラスタ JSP のサンプルの実行中は、アプリケーションから多数のメッセージが生成され、サーバーのイベントログに記録されます。次に、出力されるメッセージの一部を引用します。

次に例を示します。

```
[13/Aug/2003:17:32:28] INFO ( 9657):CORE3282:stdout: No parameter
entered for this request
```

```
[13/Aug/2003:17:32:34] INFO ( 9657):CORE3282:stdout: Add to session:
name1 = 1
```

```
[13/Aug/2003:17:32:45] INFO ( 9657):CORE3282:stdout: Add to session:
name2 = 2
```

```
[13/Aug/2003:17:32:52] INFO ( 9657):CORE3282:stdout: Add to session:
name3 = 3
```

これらのメッセージには、データがアプリケーションのユーザーインタフェースに入力されるときの値の変更が示されます。

アクセスログのアプリケーション生成のメッセージ

クラスタ JSP のサンプルを実行すると、アクセスログメッセージが、要求を処理したアプリケーションサーバーインスタンスに記録されます。

```
192.18.151.14 - - [12/Aug/2003:15:34:52 -0700] "GET /clusterjsp/
HTTP/1.0" 302 1086
```

```
192.18.151.14 - - [12/Aug/2003:15:34:52 -0700] "GET
/clusterjsp/HaJsp.jsp HTTP/1.0" 200 1578
```

アプリケーションサーバーインスタンスへの配備に対するトラブルシューティング

サンプルアプリケーションの実行に関する一般的な問題を、次の表にまとめます。左側の列には状態、中央の列には問題の考えられる原因、右側の列には解決策を示します。

表 3-2 アプリケーションサーバーインスタンスへの配備に対するトラブルシューティング

状態	考えられる原因	解決法
最初のページにアクセスできない	アプリケーションサーバーが稼動していない	アプリケーションサーバーが稼動していることを確認する
	URL に別のポート番号を指定している	HTTP サーバーの正しいポート番号を指定する 詳細は、 27 ページの「サーバーの起動」 を参照
メインページの表示時の 404 エラー	アプリケーションが配備されていない	47 ページの「クラスタ JSP のサンプルアプリケーションのクラスタへの配備」 を参照

問題を解決したら、必ずアプリケーションサーバーのログファイルを確認してください。また、HTTP アクセスログファイルの内容を確認すれば、HTTP 要求が正しくアプリケーションサーバーに送られているかどうかを確認できます。

クラスタへのサンプルアプリケーションの追加

サンプルアプリケーションをクラスタ内の2つのサーバーインスタンスに配備したら、アプリケーションを loadbalancer.xml で作成したクラスタに追加する必要があります。

1. Web サーバーの config ディレクトリに移動し、loadbalancer.xml をテキストエディタで開きます。
2. サンプルの loadbalancer.xml ファイルには、次の行があります。

```
<web-module context-root="/abc" enabled="true"
disable-timeout-in-minutes="60" enabled="true" />
```

clusterjsp サンプルアプリケーションを反映するように、この行を変更します。

```
<web-module context-root="clusterjsp" enabled="true"
disable-timeout-in-minutes="60"/>
```

この行により clusterjsp アプリケーションがクラスタに追加され、有効になります。

コンテキストルートは、Sun ONE Application Server インスタンスの server.xml ファイルのアプリケーションに設定されているものと同じ値です。

disable-timeout-in-minutes 属性は 60 に設定されています。アプリケーションが無効になっている場合は、アプリケーションに対する未処理の要求を処理するために、サーバーは 60 分間アプリケーションを休止状態にしておくことができます。インスタンスレベルの disable-timeout-in-minutes はすでに設定しています。これは、サーバーインスタンスの休止時間を制御します。

3. 変更を保存します。

この時点で、loadbalancer.xml ファイルは次のようになっているはずです。

コード例 3-1 サンプルアプリケーションが設定されている loadbalancer.xml ファイル

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE
Application Server 7.0//EN" "sun-loadbalancer_1_0.dtd">
```

```
<loadbalancer>
```

```
  <cluster name="cluster1">
```

```
    <instance name="server1" enabled="true"
disable-timeout-in-minutes="5" listeners="http://test.sun.com:81"/>
```

```
    <instance name="server2" enabled="true"
disable-timeout-in-minutes="5" listeners="http://test.sun.com:82"/>
```

```
<web-module context-root="clusterjsp" enabled="true"
disable-timeout-in-minutes="60"/>

<health-checker url="/" interval-in-seconds="10"
timeout-in-seconds="30" />

</cluster>

<property name="reload-poll-interval-in-seconds" value="60"/>

<property name="response-timeout-in-seconds" value="30"/>

<property name="https-routing" value="true"/>

<property name="require-monitor-data" value="true"/>

</loadbalancer>
```

設定の変更の適用と Web サーバーの再起動

loadbalancer.xml を編集したら、Web サーバーを再起動する必要があります。
Web サーバーがすでに稼働している場合は、設定の変更を適用して、Web サーバーを再起動する必要があります。

Web サーバーを起動するには、次のようにします。

1. `web_server_install_dir/https-instance_name` に移動します。

次に例を示します。

```
/usr/iplanet/servers/https-test.sun.com
```

2. `./start` と入力してサーバーを起動します。

`web_server_install_dir/https-admin-serv` にある Web サーバーの管理サーバーも起動する必要があります。

稼働している Web サーバーに変更を適用して再起動するには、次のようにします。

1. Web ブラウザに適切な URL を入力して、Web サーバーのサーバーマネージャにアクセスします。

```
http://web_server:web_server_admin_port
```

次に例を示します。

```
http://test.sun.com:8888
```

2. サーバーマネージャからサーバーを選択して、「Manage (管理)」をクリックします。

Web サーバーインスタンスのサーバーマネージャが表示されます。

3. 設定ファイルに手動で変更が加えられたため、変更を適用する必要があるという警告メッセージが表示されます。
4. ページの右上にある「Apply (適用)」リンクをクリックします。
5. 「Apply Changes (変更の適用)」をクリックして変更を適用し、サーバーを再起動します。

それ以降の `loadbalancer.xml` の変更には、変更を適用してサーバーを再起動する必要はありません。再読み込みのポーリング間隔が設定されているため、それ以降のすべての変更は自動的に読み込まれます。詳細は、[42 ページの「再読み込みのポーリング間隔を使った動的再設定」](#)を参照してください。

アプリケーションの実行

アプリケーションをすべてのインスタンスに配備して、`loadbalancer.xml` ファイルを更新したら、Web サーバー上のアプリケーションをテストして、機能することを確認します。

1. ブラウザから次の URL にアクセスします。

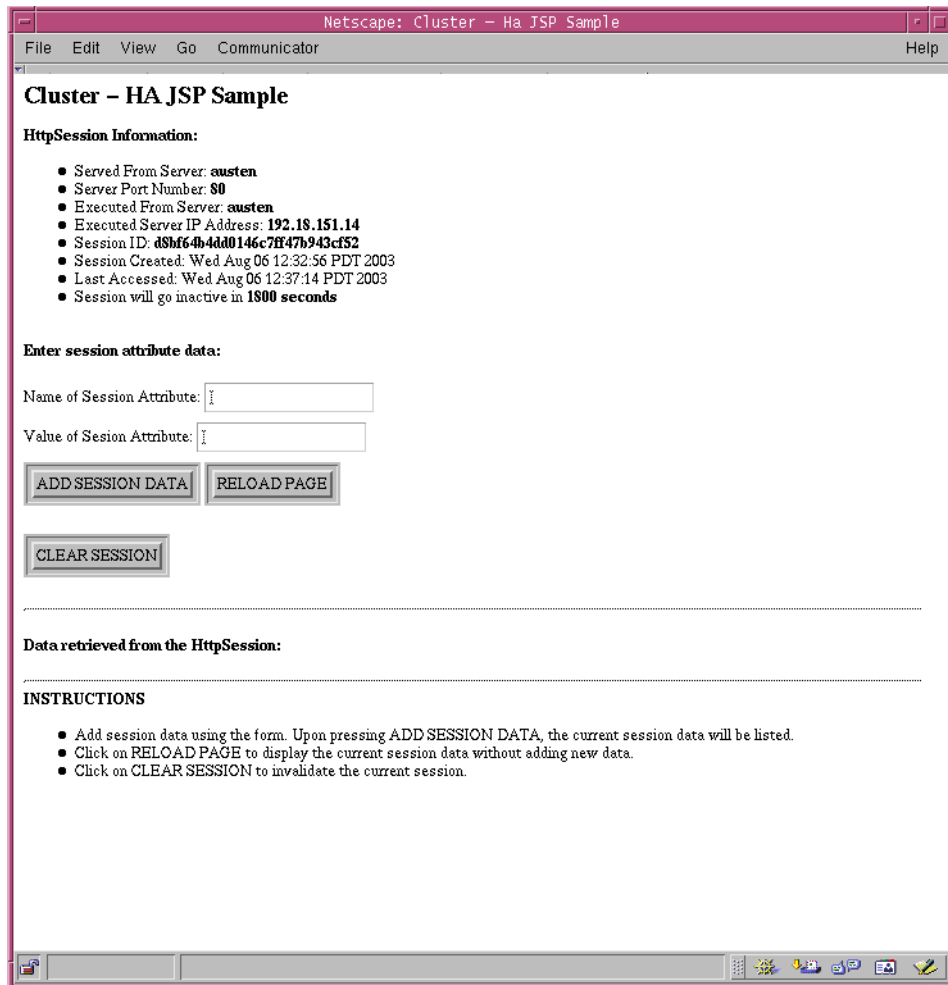
`http://host:web_server_port/clusterjsp`

次に例を示します。

`http://test.sun.com:80/clusterjsp`

このサンプルアプリケーションのページが表示されます。

図 3-4 クラスタ JSP のサンプルアプリケーションページ

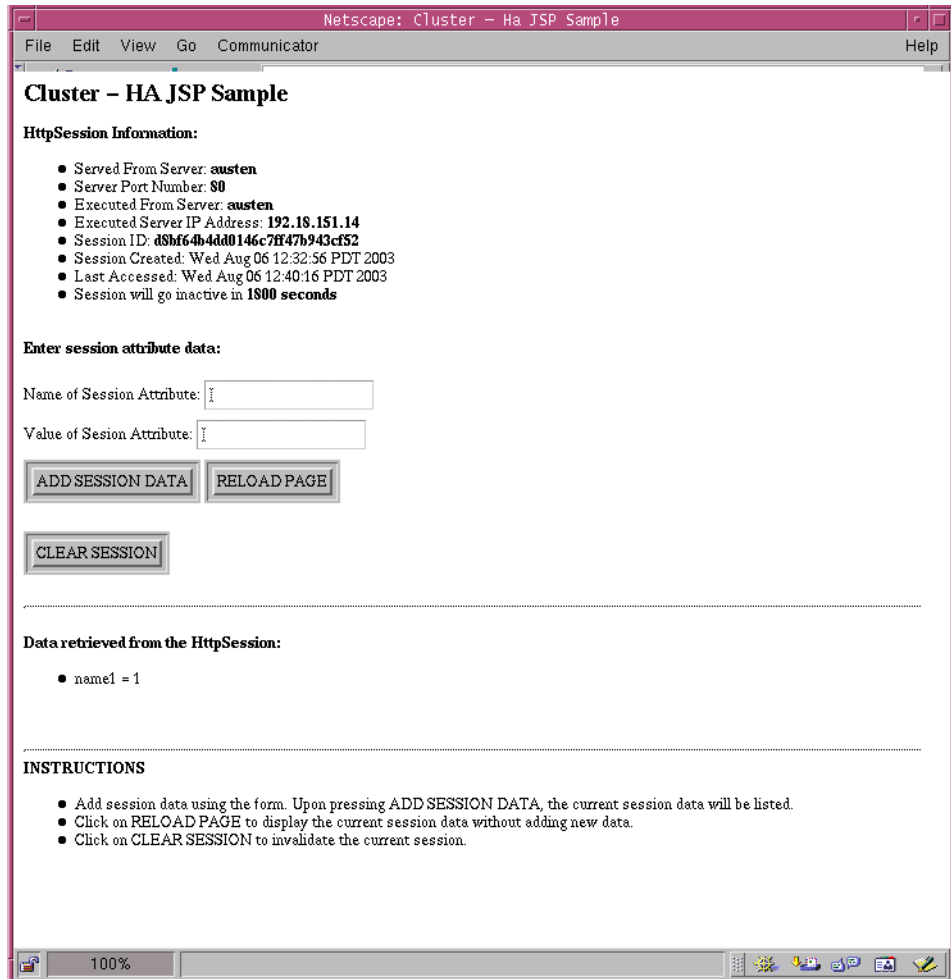


サーバー情報は、一意なセッション ID とともにページの上部に表示されます。このアプリケーションにアプリケーションサーバーインスタンス上で直接アクセスしている場合は、ページの上部に表示されるポート情報がアプリケーションサーバー情報です。クラスタ化されたアプリケーションに Web Server からアクセスしている場合、ポート情報は Web サーバー情報です。

2. セッション属性の名前と値を入力して、「Add Session Data (セッションデータを追加)」をクリックします。

セッションデータが表示されます。

図 3-5 クラスタ JSP に表示されたセッションデータ



ページ上部のセッション ID に注意してください。セッションがタイムアウトになる前にセッション属性を入力して「ADD SESSION DATA (セッションデータを追加)」をクリックした場合、ページの上部のセッション ID は最初にアクセスしたときと同じものになります。

上の例では、セッション ID である d8bf64b4dd0146c7ff47b943cf52 はどちらの場合でも同じです。

セッションがタイムアウトになった場合、セッション ID は異なります。

3. 新しいセッション属性の名前と値を入力します。セッションがタイムアウトになっていない場合は、すべての属性が「Data Retrieved from HttpSession (HttpSession から受信されたデータ)」領域に表示されます。

たとえば、セッションがタイムアウトになる前にセッション属性データを 2 回以上入力した場合、次のようなページが表示されます。

図 3-6 複数の HTTP セッション属性を含むクラスタ JSP のサンプルアプリケーション

Netscape: Cluster - Ha JSP Sample

File Edit View Go Communicator Help

Cluster - HA JSP Sample

HttpSession Information:

- Served From Server: **austen**
- Server Port Number: **80**
- Executed From Server: **austen**
- Executed Server IP Address: **192.18.151.14**
- Session ID: **d8bf64b4dd0146c7ff47b943cf52**
- Session Created: Wed Aug 06 12:32:56 PDT 2003
- Last Accessed: Wed Aug 06 12:42:49 PDT 2003
- Session will go inactive in **1800 seconds**

Enter session attribute data:

Name of Session Attribute:

Value of Session Attribute:

Data retrieved from the HttpSession:

- name2 = 2
- name3 = 3
- name1 = 1

INSTRUCTIONS

- Add session data using the form. Upon pressing ADD SESSION DATA, the current session data will be listed.
- Click on RELOAD PAGE to display the current session data without adding new data.
- Click on CLEAR SESSION to invalidate the current session.

4. セッションがタイムアウトするまで (1800 秒) 待機してから新しいセッション属性データを入力した場合は、ページの上部に、最後に入力したデータと新しいセッション ID のみが表示されます。

「CLEAR SESSION (セッションをクリア)」をクリックして、タイムアウトをシミュレートすることもできます。これにより、セッションデータがクリアされ、新しいセッション ID が指定されます。

図 3-7 新しい HTTP セッション情報を持つクラスタ JSP

Cluster - HA JSP Sample

HttpSession Information:

- Served From Server: **austen**
- Server Port Number: **80**
- Executed From Server: **austen**
- Executed Server IP Address: **192.18.151.14**
- Session ID: **d8ca7921435686f1111f996dc3bbf9c5489**
- Session Created: Wed Aug 06 12:45:02 PDT 2003
- Last Accessed: Wed Aug 06 12:45:02 PDT 2003
- Session will go inactive in **1800 seconds**

Enter session attribute data:

Name of Session Attribute:

Value of Session Attribute:

ADD SESSION DATA **RELOAD PAGE**

CLEAR SESSION

Data retrieved from the HttpSession:

- test4 = 4

INSTRUCTIONS

- Add session data using the form. Upon pressing **ADD SESSION DATA**, the current session data will be listed.
- Click on **RELOAD PAGE** to display the current session data without adding new data.
- Click on **CLEAR SESSION** to invalidate the current session.

100%

ページの上部には新しいセッション ID (この例では d8ca7921435686ffffff996dc3bbf9c5489) が表示され、最後に入力した情報 (この例では test4=4) は、「Data Retrieved from HttpSession (HttpSession から受信されたデータ)」領域に表示されています。

ロードバランスの確認

これで、サンプルアプリケーションの実行が完了し、セッション情報がどのように表示されるかがわかりました。これでロードバランサがどのように動作するかを見ることができます。または、ロードバランサの問題を解決できます。

この節には次の項目があります。

- [ロードバランスの確認手順](#)
- [ロードバランサプラグインのトラブルシューティング](#)

ロードバランスの確認手順

次の手順に従って、ロードバランサの動作を見ることができます。

1. 2つの別々のブラウザを開き、
`http://web_server_name:web_server_port/clusterjsp` と入力して、アプリケーションを Web サーバーから実行します。次に例を示します。
`http://test.sun.com:80/clusterjsp`
ロードバランサを使用するときはセッションデータがスティッキーであるため、ブラウザウィンドウを開いてアプリケーションにアクセスする場合、そのブラウザからの再読み込みなどのすべての操作は同じセッションにあるとみなされ、単一の Sun ONE Application Server インスタンスによって処理されます。
アプリケーションに対してロードバランスが機能することを確認するために、2番目のブラウザを開いてアプリケーションにアクセスします。セッションデータは Cookie に格納されているため、別のマシンにある 2 番目のブラウザを開くか、同じマシンにある別のブラウザソフトウェアを使用する必要があります。セッション ID をチェックして、両方のブラウザウィンドウに個別のセッションデータがあることを確認してください。
2. 2つの異なるセッションウィンドウでアプリケーションにアクセスしたら、もう 1つのブラウザウィンドウを開いて、Web サーバーのエラーログを確認します。
 - a. Web サーバーのマニュアルに記載されている URL で、Sun ONE Web Server の管理サーバーにアクセスします。次に例を示します。
`http://test.sun.com:8888`

- b. Web サーバーインスタンスを選択して、「Manage (管理)」をクリックします。
 - c. Web サーバーインスタンスのサーバーマネージャで、「Logs (ログ)」タブをクリックします。
 - d. 左側のペインの「View Error Logs (エラーログを表示)」をクリックします。
3. その中の RequestExit を含むエントリを探します。
- 目で見て調べるか、「Only show entries with (表示するエントリタイプ)」フィールドに「RequestExit」と入力することができます。
4. RequestExit を含むエントリは、Sun ONE Application Server の両方のインスタンスの HTTP リスナー ID を示すため、両方のアプリケーションサーバーインスタンスが要求を処理していることを確認できます。
5. Sun ONE Application Server インスタンスが要求を処理したことを確認するには、それぞれのアプリケーションサーバーインスタンスの HTTP アクセスログをチェックします。
- clusterjsp アプリケーションに対する要求が表示されます。アクセスログのチェックの詳細については、[54 ページの「管理インタフェースによるログの表示」](#)を参照してください。

注	RequestExit メッセージを表示するためには、Web サーバーの詳細ログが有効になっている必要があります。手順については、 40 ページの「ロードバランサの監視の有効化」 を参照してください。
---	----------------------------------------------------------------------------------------------------------------------

ロードバランサを動作させるときにトラブルが発生する場合は、次の節の「[ロードバランサプラグインのトラブルシューティング](#)」を参照してください。

ロードバランサプラグインのトラブルシューティング

次の表は、ロードバランスのトラブルシューティングに役立ちます。左側の列には状態、中央の列には考えられる原因、右側の列には解決策が示されています。

表 3-3 ロードバランスに対するトラブルシューティング

状態	考えられる原因	解決法
アプリケーションにアクセスできない	Web サーバーが起動していない URL に別のポート番号を指定している	Web サーバーが稼働していることを確認する Web サーバーの正しいポート番号を指定する 詳細は、Web サーバーのマニュアルを参照
アプリケーションへのアクセス時の 404 エラー	アプリケーションが配備されていない loadbalancer.xml ファイル内のエラー	47 ページの「クラスタ JSP のサンプルアプリケーションのクラスタへの配備」 を参照 loadbalancer.xml ファイル内のエラーは Web サーバーのエラーログに書き込まれる。詳細は、 68 ページの「loadbalancer.xml ファイル内のエラーの検出」 を参照
アプリケーションへのアクセス時のエラーページ	クラスタ内の 1 つまたは複数のサーバーが応答していない	クラスタ内のサーバーインスタンスが稼働していることを確認する エラーログをチェックして、ヘルスチェッカーによりダウンしているインスタンスが見つかったかどうかを確認する。詳細は、 69 ページの「ヘルスチェッカーの使用」 を参照

loadbalancer.xml ファイル内のエラーの検出

Web サーバーの URL からサンプルアプリケーションにアクセスできない場合は、ロードバランサに問題がある可能性があります。個々の Application Server からアプリケーションにアクセスできる場合は、Application Server インスタンスには問題がないと判断できます。

ロードバランサプラグインは、Web サーバーのログファイルにメッセージを記録するので、さらに詳細な情報については、Web サーバーのエラーログから確認してください。

Sun ONE Web Server のエラーログは、サーバーマネージャから表示できます。

1. Web サーバーのマニュアルに記載されている URL で、Sun ONE Web Server の管理サーバーにアクセスします。次に例を示します。

```
http://test.sun.com:8888
```

2. Web サーバーインスタンスを選択して、「Manage (管理)」をクリックします。
3. Web サーバーインスタンスのサーバーマネージャで、「Logs (ログ)」タブをクリックします。
4. 左側のペインの「View Error Logs (エラーログを表示)」をクリックします。
5. デフォルトでは 25 個のメッセージが表示されます。この数は必要に応じて増やすことができます。

ロードバランサで問題が発生した場合は、「Failed to initialize load balancing subsystem (ロードバランスのサブシステムの初期化に失敗しました)」に示されているメッセージを確認します。

loadbalancer.xml ファイルにエラーがあるという問題の場合は、loadbalancer.xml ファイル内のエラーがある行を示すパーサーエラーが表示されることがあります。たとえば、次のように表示されます。

```
[17/Jun/2003:09:57:44] catastrophe ( 5643): LBConfigParser.cpp@434:
reports: lb.configurator: CNFG1000: Parsing of file
:/usr/iplanet/servers/https-test.sun.com/config/loadbalancer.xml

Failed at line : 7 and column : 3.The error message is : No character
data is allowed by content model
```

このメッセージは、loadbalancer.xml ファイル内のエラーがある行を示しています。

loadbalancer.xml ファイル内の問題を修正し、変更を適用して Web サーバーを再起動します。その後、再試行します。

ヘルスチェッカーの使用

ロードバランサが正常に稼働していて、Web サーバーの LogVerbose が on に設定されている場合は、Web サーバーのログファイルに動作しているヘルスチェッカーが表示されます。

```
[09/Apr/2003:13:36:02] verbose ( 7576): HealthChecker.cpp@153:
reports: lb.monitor: HLCK1006:UnhealthyInstances cluster1
1049920562631 NoUnhealthyInstances
```

詳細ログを有効にする手順については、[40 ページの「ロードバランサの監視の有効化」](#)を参照してください。

詳細ログを使用するかどうかとは関係なく、インスタンスの1つがダウンすると、ロードバランサは、アクセスの試行が失敗したときにダウンしたというフラグを付けます。ダウンしたままになっている場合、ヘルスチェッカーはそれに、ダウンしているとして再度フラグを付けます。

次に例を示します。

```
[17/Jun/2003:10:37:27] warning ( 5700): reports: lb.runtime:  
RNTM2024: Daemon http://test.sun.com:81 is unhealthy.
```

```
[17/Jun/2003:10:37:27] warning ( 5700): reports: lb.healthchecker:  
HLCK3003: Listener: http://test.sun.com:81 is detected to be still  
unHealthy in cluster: cluster1
```

ヘルスチェッカーはダウンしているインスタンスの監視を続け、稼働中になると、稼働しているというフラグを付けます。

HTTP セッションの持続性の確認

ロードバランスの機能が、正常に動作していることを確認したなら、次に、HTTP セッションの持続性の機能を確認することができます。セッション持続性では、何らかの理由により1つの Sun ONE Application Server インスタンスが失敗すると、2 番目のインスタンスがセッションをピックアップして要求を処理します。

注	セッション持続性はクラスタ間では機能せず、クラスタ内のインスタンス間でのみ機能します。
---	---------------------------------------------

HTTP セッションの持続性が機能していることを確認するには、次のようにします。

1. ブラウザウィンドウを開きます。
2. Web サーバーからアプリケーションにアクセスします。次に例を示します。
`http://test.sun.com:80/clusterjsp`
3. Web サーバーのエラーログを表示して、要求を処理したサーバーを確認します (ここでも、RequestExit エントリを検索します)。
4. ページを処理している Sun ONE Application Server インスタンスをシャットダウンします。

管理インタフェースを使ってシャットダウンすることができます。

- a. 管理ポート番号を使って管理インタフェースにアクセスします。次に例を示します。

`http://test.sun.com:4848`

- b. 左側のペインで、停止するサーバーインスタンスをクリックします。
 - c. 右側のペインで「Stop (停止)」をクリックします。
5. クラスタ JSP のサンプルアプリケーションページを再度読み込みます。
セッション ID とセッション属性のデータは保持されています。
 6. Web サーバーのエラーログをチェックします。この時点ではほかの Application Server インスタンスが要求を処理していることに注意してください。

セッション情報は、サーバーがオフラインになったあとで保存されます。障害発生時にセッションデータが失われる可能性があるのは、アプリケーションによるデータの転送中に障害が発生した場合のみです。この場合、少し古いセッションデータが表示されることがあります。

サーバーインスタンスの休止

実際には、既存のセッションを完了させようとしめない限り、サーバーがシャットダウンすることはありません。代わりに、休止というプロセスで、サーバーを段階的にシャットダウンします。

アプリケーションサーバーインスタンスを休止させるには、次の手順で行います。

1. ブラウザウィンドウを開いて、クラスタ JSP アプリケーションに Web サーバーポートからアクセスします。このウィンドウは開いたままにします。
2. 要求を処理するアプリケーションサーバーインスタンスを決定します。
2 番目のブラウザウィンドウを開き、管理インタフェースで **server1** アプリケーションサーバーインスタンスの HTTP アクセスログにアクセスします。要求が送信された時点でのクラスタ JSP アプリケーションのアクセスのログを確認します。**server1** ログにあるアクセスが表示された場合、**server1** アプリケーションサーバーインスタンスは要求を処理しています。アクセスログのこのようなエントリが表示されない場合は、**server2** アプリケーションサーバーインスタンスのアクセスログを確認してください。アクセスログウィンドウは開いたままにします。

3. 要求を処理しているアプリケーションサーバーインスタンスを確認したら、`loadbalancer.xml` ファイルにあるそのサーバーインスタンスを無効にします。

正しいサーバーインスタンスに対して、`enabled="true"` を `enabled="false"` に変更します。次に例を示します。

```
<instance name="server1" enabled="false"
disable-timeout-in-minutes="5"
listeners="http://test.sun.com:81"/>
```

4. `loadbalancer.xml` の変更を保存します。
5. 新しい設定が読み込まれるように、再読み込みポーリング間隔の経過を待ちます。
このチュートリアルでは、再読み込みポーリング間隔は 60 秒に設定されているため、1 分だけ待機する必要があります。
6. 再読み込みポーリング間隔が経過して、新しい設定が読み込まれると、ロードバランサプラグインは、既存のセッションがない要求をそのサーバーインスタンスに送信しなくなります。ただし、既存のセッションがある要求は、休止時間の間引き続きサーバーインスタンスに転送されます。

この状態を確認するには、開いているクラスタ JSP ウィンドウで名前と属性の情報を入力して、「ADD SESSION DATA (セッションデータを追加)」をクリックします。

7. 最初の要求を処理したサーバーの HTTP アクセスログを表示する、ブラウザウィンドウに移動します。「OK (了解)」をクリックして情報を更新します。
サーバーは、`loadbalancer.xml` で無効になっていても、この要求を処理することに注意してください。
8. 終了までの休止時間として 5 分間待機してから、クラスタ JSP アプリケーションウィンドウを開いて追加の名前属性情報を入力して、「ADD SESSION DATA (セッションデータを追加)」をクリックします。

休止時間は、サーバーインスタンスの無効タイムアウトによって制御されます。このチュートリアルの前の手順で、この値を 5 分に設定しています。

9. 休止時間が経過したため、要求は別のアプリケーションサーバーインスタンスに送信されます。
この結果を表示するには、サーバーインスタンスの HTTP アクセスログを更新します。新しい要求は表示されません。
10. 3 番目のブラウザウィンドウを開いて、ほかのサーバーインスタンスの HTTP アクセスログを表示します。2 番目のサーバーインスタンスによって処理されているクラスタ JSP の要求が表示されます。

そのあと、アプリケーションサーバーインスタンスを安全にシャットダウンすることができます。

アプリケーションを休止して、安全にオフラインにすることもできます。アプリケーションサーバーインスタンスとアプリケーションの休止の詳細については、『**Sun ONE Application Server 管理者ガイド**』を参照してください。

サーバーインスタンスの休止

要約と参照情報

本書「入門ガイド」では、Sun ONE Application Server 7, Enterprise Edition の代表的な機能と使用方法について説明しています。

- 主な機能とアーキテクチャ
- アプリケーションサーバーの管理に使用するツールの使用
- J2EE アプリケーションの配備と使用
- ロードバランスの設定と使用
- HTTP セッションの持続性の設定と使用
- サーバーインスタンスの休止

以下の表から、目的に合った資料を特定することで、次に必要な手順の詳細を参照してください。左側の列には目的のトピックを示し、右側の列には、目的を遂行するために必要な詳細情報の入手方法が記載されています。

表 4-1 参照情報

目的	参照情報
Sun ONE Application Server のサンプルアプリケーションで実装されている Enterprise Edition の機能を詳しく調べる	<code>http://application_server_host_name:port/samples</code> で Application Server とともにインストールされた Enterprise Edition のサンプルアプリケーションを参照。索引ページに、入手できるすべての Enterprise Edition のサンプルの情報がある
J2EE の設計および開発の最適事例を詳しく調べる	Java BluePrints を調べ、アプリケーションサーバーに付属するサンプルアプリケーション Java Pet Store および Smart Ticket を試す
アプリケーションのフェイルオーバー、セッション持続性、および高可用性セッションの設定について詳しく調べる	『Sun ONE Application Server Application Guidelines for Storing Session State』を参照し、アプリケーションサーバーに付属している Duke's Bookstore と Session Storage のサンプルアプリケーションを使用する

表 4-1 参照情報 (続き)

目的	参照情報
クラスタリング、セッション持続性、ロードバランス、HADB などの Sun ONE Application Server 7 の管理について調べる	『Sun ONE Application Server 7 管理者ガイド』を参照
チュートリアルを使って J2EE を詳しく調べる	J2EE チュートリアルと Java Web サービスのチュートリアルを参照
特定の J2EE 機能が Sun ONE Application Server 7 にどのように実装されているかを詳しく調べる	アプリケーションサーバーのインストールに付属しているサンプルアプリケーションを参照
Sun ONE Application Server 7 を使った一般的な開発事例を詳しく調べる	『Sun ONE Application Server 7 開発者ガイド』を参照

索引

A

asadmin ユーティリティ, 22, 50
 configure-session-persistence コマンド, 50
 create-jdbc-connection-pool コマンド, 50
 create-jdbc-resource コマンド, 50
 delete-jdbc-connection-pool コマンド, 50
 delete-jdbc-resource コマンド, 50
 deploy コマンド, 50
 list-instances コマンド, 33, 51
 PATH 変数の設定, 27
 start-instance コマンド, 50
 stop-instance コマンド, 50

C

cladmin.log ファイル, 49
cladmin コマンド, 22, 47
 deploy コマンド, 49
 PATH 変数の設定, 27
 start-instance, 51
 構文, 48
 サポートされた asadmin コマンド, 50
 制限, 51
 場所, 47
 要件, 51
 ログファイル, 49
clinstance.conf ファイル, 48
clpassword.conf ファイル, 48

clsetup コマンド, 22, 25
configure-session-persistence コマンド, 50
create-jdbc-connection-pool コマンド, 50
create-jdbc-resource コマンド, 50

D

default_config_dir, 11
delete-jdbc-connection-pool コマンド, 50
delete-jdbc-resource コマンド, 50
deploy コマンド, 50
disable-timeout-in-minutes, 37

F

file セッション持続性, 18

H

HADB, 19
ha セッション持続性, 18
http-listener 要素、server.xml ファイル, 33
HTTPS リスナー, 37, 43
HTTPS ルーティング, 43
HTTP アクセスログ, 55

I

HTTP サーバー

トップページ, 34

ポート, 33

リスナー, 33

HTTP リスナー, 37

RequestExit ログファイルエントリ, 67

I

install_config_dir, 12

install_dir, 11

interval-in-seconds, 39

L

list-instances コマンド, 33, 51

loadbalancer.xml.example ファイル, 35

loadbalancer.xml ファイル

エラー, 68

作成, 35

例, 35, 43

LogVerbose, 41

M

memory セッション持続性, 18

P

PATH 環境変数、設定, 27

S

start-domain コマンド, 27

start-instance コマンド, 50, 51

stderr, 57

stdout, 57

stop-instance コマンド, 50

sun-loadbalancer.dtd ファイル, 36

T

tail コマンド, 32, 56

timeout-in-seconds, 39

U

undeploy コマンド, 50

URL の書式, 11

W

Web サーバー

Apache, 24

エラーログ, 42

変更の適用, 60

ログの設定, 40

Web サーバーのエラーログ, 69

Web サーバーへの変更の適用, 60

あ

アクセスログ, 57

アプリケーションサーバー

起動, 27

起動の確認, 29

アプリケーションサーバーインスタンス, 13

起動の確認, 32

休止, 71

クラスタに追加, 36

アプリケーションサーバーインスタンスの起動, 51

アプリケーション、サンプル

監視, 57

休止, 59

クラスタに追加, 59

コピー, 46

実行, 52

トラブルシューティング, 58

い

イベントログ

アプリケーションサーバーインスタンス, 32

アプリケーションメッセージ, 57

管理サーバー, 31

インストールの要件, 24

え

エラーログ、Web サーバー, 42, 69

お

応答タイムアウト, 42

か

確認

アプリケーションの配備, 52

サーバーインスタンスの起動, 32

カスタマサポート, 12

監視、ロードバランサ, 40

管理インタフェース, 22

アクセス, 30

ログの表示, 54

管理サーバー, 14

イベントログ, 31

起動の確認, 30

パスワード, 30

ポート, 30

ユーザー名, 30

管理ドメイン, 16

き

休止

アプリケーション, 59

サーバーインスタンス, 71

く

クラスタ, 16

アプリケーションの追加, 59

アプリケーションの配備, 47

作成, 36

シナリオの例, 20

クラスタ JSP のサンプルアプリケーション, 47

実行, 61

こ

高可用性のセッション持続性, 18

コマンド行ユーティリティ, 22

さ

サーバーインスタンス, 13

起動の確認, 32

クラスタに追加, 36

再読み込みのポーリング間隔, 42

サンプルアプリケーション

WAR ファイル, 47

監視, 57

し

クラスタ JSP, [47](#)
クラスタに追加, [59](#)
クラスタへの配備, [47](#)
コピー, [46](#)
実行, [52](#), [61](#)
トラブルシューティング, [58](#)
配備の確認, [52](#)
サンプルアプリケーションの監視, [57](#)

し

詳細ログ, [41](#)
シングルサインオン、セッション情報の利用可能性,
[19](#)

す

スティッキーなラウンドロビン, [17](#)

せ

セッション持続性, [18](#)
file, [18](#)
ha, [18](#)
memory, [18](#)
確認, [70](#)
設定, [19](#)

ち

チュートリアル
エンタープライズ機能の設定, [27](#)
クラスタ JSP のサンプル, [47](#)
使用する準備, [23](#)
手順, [26](#)

て

ディレクトリ, [11](#)

と

トップページ、HTTP サーバー, [34](#)
ドメイン、管理, [16](#)
トラブルシューティング
サンプルアプリケーション, [58](#)
ロードバランサ, [68](#)

は

配備
確認, [52](#)
サンプルアプリケーションをクラスタへ, [47](#)
トラブルシューティング, [58](#)
配布可能なアプリケーション, [47](#)
パスの書式, [11](#)
パスワード、管理サーバー, [30](#)

ふ

フォントの表記規則, [11](#)

へ

ヘルスチェッカー, [39](#), [69](#)
interval-in-seconds, [39](#)
timeout-in-seconds, [39](#)
URL, [39](#)

ほ

ポート

HTTP サーバー, 33
 Web サーバー, 61
 アクセスできない, 31
 管理サーバー, 30

ま

マニュアル

URL の書式, 11
 一般的な表記規則, 11
 使用方法, 8
 ディレクトリの表記規則, 11
 パスの書式, 11
 フォントの表記規則, 11
 マニュアルの構成, 8

tail コマンドを使用したモニタリング, 32, 56
 Web サーバーエラー, 42
 Web サーバーの詳細ログを有効にする, 41
 アクセス, 57
 イベント、アプリケーションサーバーインスタンス, 57
 イベント、管理サーバー, 31
 エラー、Web サーバー, 42
 表示, 54

ゆ

ユーザー名、管理サーバー, 30

ら

ラウンドロビンのロードバランス, 17

ろ

ロードバランサ, 17, 40
 HTTPS ルーティング, 43
 loadbalancer.xml のサンプル, 43
 アルゴリズム, 17
 応答タイムアウト, 42
 動的再設定, 42
 トラブルシューティング, 68
 プロパティ, 42
 ヘルスチェッカー, 39, 69
 ログ
 HTTP アクセス, 55

