



Sun Java™ System

Application Server 7 Enterprise Edition 入門ガイド

2004Q2

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

Part No: 817-6876

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. は、この製品に含まれるテクノロジーに関する知的所有権を保持しています。特に限定されることなく、これらの知的所有権は <http://www.sun.com/patents> に記載されている 1 つ以上の米国特許および米国およびその他の国における 1 つ以上の追加特許または特許出願中のものが含まれている場合があります。

このソフトウェアは SUN MICROSYSTEMS, INC. の機密情報と企業秘密を含んでいます。SUN MICROSYSTEMS, INC. の書面による許諾を受けることなく、このソフトウェアを使用、開示、複製することは禁じられています。

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

Use is subject to license terms. この配布には、第三者が開発したソフトウェアが含まれている可能性があります。

Sun、Sun Microsystems、Sun のロゴマーク、Java、Solaris、Sun™ ONE、Sun™ ONE Studio、iPlanet、J2EE、J2SE、Enterprise JavaBeans、EJB、JavaServer Pages、JSP、JDBC、JDK、JVM、Java Naming and Directory Interface、JavaMail および Java Coffee Cup のロゴは、米国およびその他の国における米国 Sun Microsystems, Inc. (以下、米国 Sun Microsystems 社とします) の商標もしくは登録商標です。

すべての SPARC 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標が付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャに基づくものです。

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。

この製品は、米国の輸出規制に関する法規の適用および管理下にあり、また、米国以外の国の輸出および輸入規制に関する法規の制限を受ける場合があります。核、ミサイル、生物化学兵器もしくは原子力船に関連した使用またはかかる使用者への提供は、直接的にも間接的にも、禁止されています。このソフトウェアを、米国の輸出禁止国へ輸出または再輸出すること、および米国輸出制限対象リスト (輸出が禁止されている個人リスト、特別に指定された国籍者リストを含む) に指定された、法人、または団体に輸出または再輸出することは一切禁止されています。

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らないものとします。

目次

本書について	7
マニュアルの概要	7
マニュアルの使用方法	8
マニュアルの構成	11
マニュアルの表記規則	12
一般的な表記規則	12
ディレクトリ名の表記規則	13
連絡先	14
ご意見、ご要望	14
トレーニング	14
製品サポートの問い合わせ	14
第1章 Sun Java System Application Server 7, Enterprise Edition の概要	17
Sun Java System Application Server について	17
Application Server インスタンスについて	18
管理ドメインについて	20
Enterprise Edition 機能について	20
クラスタリング	20
高可用性	21
スケーラビリティ	22
ロードバランス	23
セッション持続性	24
セッション持続性のタイプについて	24
セッション持続性の設定について	26
シングルサインオンセッション情報について	27
SFSB チェックポイントについて	27
高可用性データベース	28

Application Server の設定と管理に使用するツール	29
---	----

第 2 章 クラスタリングのシナリオ	31
HTTP クラスタリングのシナリオ	31
RMI/IIOP クラスタリングのシナリオ	33
サーバー側の設定	33
クライアント側の設定	34
スタンドアロンのクライアント	34
ACC クライアント	35

第 3 章 エンタープライズ機能の設定のチュートリアル	37
チュートリアルを使用する準備	37
インストールの要件	38
チュートリアルを使用する前の必要な手順	38
チュートリアルの手順の概要	40
サーバーの起動	41
PATH 変数の設定	41
asadmin start-domain の実行	42
サーバーの起動の確認	43
管理サーバーの起動の確認	44
管理インタフェースへのアクセス	44
管理サーバーのイベントログの表示	45
サーバーインスタンスの起動の確認	46
asadmin を使ったインスタンスの確認	46
HTTP サーバーへのアクセスによるインスタンスの確認	46
loadbalancer.xml ファイルの作成	48
loadbalancer.xml ファイルへのクラスタの追加	49
ロードバランスの設定	52
ヘルスチェッカーの設定	52
ロードバランサの監視の有効化	53
ロードバランサのその他のプロパティ	54
再読み込みのポーリング間隔を使った動的再設定	55
応答タイムアウト	55
HTTPS ルーティング	55
loadbalancer.xml ファイルのサンプル	56

第 4 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	57
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	58
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	58
cladmin コマンドの入力ファイル	60
cladmin コマンドの構文	60
cladmin deploy の実行	61

cladmin コマンドでサポートされている asadmin コマンド	62
要件と制限事項	63
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	63
アプリケーション配備の確認	64
アプリケーションサーバーのサンプルの監視	66
管理インタフェースによるログの表示	66
tail コマンドによるログの表示	67
イベントログに記録されるアプリケーション生成メッセージ	67
アクセスログに記録されるアプリケーション生成メッセージ	68
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	68
クラスタへのサンプルアプリケーションの追加	69
設定の変更の適用と Web サーバーの再起動	70
アプリケーションの実行	71
HTTP ロードバランスの確認	76
ロードバランスの確認手順	76
ロードバランサプラグインのトラブルシューティング	77
loadbalancer.xml ファイル内のエラーの検出	78
ヘルスチェッカーの使用	79
HTTP セッションの持続性の確認	80
サーバーインスタンスの休止	81
 第 5 章 要約と参照情報	 83
 索引	 85

本書について

ここでは、『Sun Java™ System Application Server 7 入門ガイド』の内容について説明します。

この章では次の項目を取り上げます。

- [マニュアルの概要](#)
- [マニュアルの使用法](#)
- [マニュアルの構成](#)
- [マニュアルの表記規則](#)
- [連絡先](#)

マニュアルの概要

この『入門ガイド』は、Sun Java System 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、Sun Java System Application Server, Enterprise Edition 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 Java System 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、セッションの持続性機能の概要を含めて、Sun Java System Application Server の機能の全体像を紹介します。次に、サンプルアプリケーションの実行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。

マニュアルの使用方法

Sun Java System Application Server Standard および Enterprise Edition 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。

次の表は、Sun Java System Application Server のマニュアルに記述されている概要と内容を示しています。(AS 7 2004Q2 用に更新) と表示されたマニュアルは、Sun Java System Application Server Standard および Enterprise Edition 7 2004Q2 リリース用に更新されたことを表します。この表示のないマニュアルは、Enterprise Edition バージョン 7 のリリース以降更新されていません。

表 1 Sun Java System 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
(AS 7 2004Q2 用に更新) ソフトウェアおよびマニュアルの最新情報。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 の一覧も含まれる	『リリースノート』
Sun Java System 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 Java System 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 Java System Application Server 7 の新機能	『新機能』
Sun Java System Application Server 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』
(AS 7 2004Q2 用に更新) Sun Java System Application Server Standard Edition および Enterprise Edition 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など)のインストール。Enterprise Edition ソフトウェアについては、高可用性設定の実装手順が記載されている	『インストールガイド』
(AS 7 2004Q2 用に更新) Sun Java System 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、システム要件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	『System Deployment Guide』
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』

表 1 Sun Java System 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
J2EE コンポーネント (サブレット、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 Java System Application Server 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、構成、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 Java System 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 Java System Application Server 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』
(AS 7 2004Q2 用に更新) Sun Java System Application Server のエンタープライズ Bean 向け Java オープンスタンダードに準拠した 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 Technology』
(AS 7 2004Q2 用に更新) Sun Java System Application Server 上で J2EE アプリケーションにアクセスする Application Client コンテナ (ACC) の作成	『Developer's Guide to Clients』
Sun Java System 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』
(AS 7 2004Q2 用に更新) 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 Java System 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 Java System 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	『Administration Guide』
server.xml ファイルなどの Sun Java System 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 Java System 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』

表 1 Sun Java System 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 Java System 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』
(AS 7 2004Q2 用に更新) 新しい Sun Java System Application Server 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、Sun ONE Application Server 7.0 からの移行)。移行例も含まれる	『Migrating and Redeploying Server Applications Guide』
(AS 7 2004Q2 用に更新) Sun Java System Application Server のパフォーマンスを改善する方法とその理由	『Performance Tuning Guide』
(AS 7 2004Q2 用に更新) Sun Java System Application Server に関する問題の解決	『Troubleshooting Guide』
(AS 7 2004Q2 用に更新) Sun Java System Application Server に関するエラーメッセージの解決	『Error Message Reference』
(AS 7 2004Q2 用に更新) マニュアルページに記載されている、Sun Java System Application Server で利用できるユーティリティコマンド	『Utility Reference Manual』
Sun Java System Message Queue 3.5 SP1 ソフトウェアの使用	Sun Java System Message Queue については次の URL を参照 : http://docs.sun.com/db/prod/s1.s1msgqu?l=ja#hic

マニュアルの構成

このマニュアルでは、Sun Java System Application Server を初めて使用するユーザーを対象に、Sun Java System Application Server の Enterprise Edition 機能について概説します。

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 Java System 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「クラスタリングのシナリオ」](#)では、HTTP と RMI/IIOP ロードバランスおよびフェイルオーバーの設定手順を取り上げます。
- [第3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4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5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。

最後に、[索引](#)が記載されています。

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)。Windows の場合はディレクトリのパスは同じです。ただし、ディレクトリは「¥」記号で区切られます。

- **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、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。
 - 斜体文字はコード変数に使用します。
 - 斜体文字は、強調、変数、可変部分、およびリテラルに使われる文字にも使います。
 - **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* で表しています。例外は、13 ページの「ディレクトリ名の表記規則」に記載しています。

ほとんどのプラットフォームの *install_dir* は、デフォルトで次の場所にあります。

- Solaris および Linux ファイルベースのインストール環境：

user's home directory/sun/appserver7

- Windows のすべてのインストール環境：

system drive:¥Sun¥AppServer7

上記のプラットフォームでは、*default_config_dir* と *install_config_dir* は、*install_dir* と同じです。例外と詳細については、13 ページの「ディレクトリ名の表記規則」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**を *instance_dir* で表しています。これは次のパスを省略したものです。

default_config_dir/domains/*domain*/*instance*

- このマニュアルでは、**UNIX 特有の説明**は、特に Linux と明記していないかぎり、Linux オペレーティングシステムにも適用されます。

ディレクトリ名の表記規則

デフォルトでは、Solaris PKG 形式でインストールする場合または Linux RPM 形式でインストールする場合、Application Server のファイルは複数のルートディレクトリに分散しています。このマニュアルは、デフォルトの複数のインストールディレクトリに対応した、次の表記規則を使用しています。

- *install_dir* は、/opt/SUNWappserver7 を示します。このディレクトリには、インストールイメージの静的な要素が保存されます。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。
- *default_config_dir* は、/var/opt/SUNWappserver7/domains を示します。このディレクトリは、作成されたドメインのデフォルトの保存場所です。
- *install_config_dir* は、/etc/opt/SUNWappserver7/config を示します。このディレクトリには、インストール全体に適用される設定情報が保存されます。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保存されます。

注	Forte for Java は、このマニュアル全体で Sun Java Studio に名前を変更しています。
----------	--

連絡先

次の場合には、Sun Microsystems まで連絡してください。

- [ご意見、ご要望](#)
- [トレーニング](#)
- [製品サポートの問い合わせ](#)

ご意見、ご要望

製品またはマニュアルに関する一般的なご意見は、
appserver-feedback@sun.com までご連絡ください。

トレーニング

Application Server トレーニングコースは、次の URL から申し込むことができます。

http://training.sun.com/US/catalog/enterprise/web_application.html/

Sun Java System Application Server の新しいコースについては、定期的にこのサイトを参照して確認してください。

製品サポートの問い合わせ

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。バージョンを調べるために一般に使用されるコマンドは次のとおりです。
 - **Solaris** : pkginfo、showrev

- **Linux** : rpm
- **すべて** : asadmin version --verbose
- 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
- 設定ファイル:
 - `instance_dir/config/server.xml`
 - Web アプリケーションの `web.xml` ファイル
Web アプリケーションが問題に関わっている場合
- アプリケーションの場合、クラスタ内で実行したときに問題が発生するのか、スタンドアロンで実行したときに発生するのか

Sun Java System 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 Java System 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 Java System Application Server について

Sun Java™ System 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。

さらに、Sun Java System Application Server は、HTTP/S 要求および RMI/IIOP 要求のロードバランスと、リモート EJB 参照および HTTP セッションの高可用性をもたらします。機能の詳細については、『Sun Java System Application Server 製品の概要』を参照してください。

この節では次の項目について説明します。

- [Application Server インスタンスについて](#)
- [管理ドメインについて](#)

Application Server インスタンスについて

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 Java System Application Server には、検証済みの高性能な HTTP サーバーも組み込まれています。オブジェクトリクエストブローカー (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。

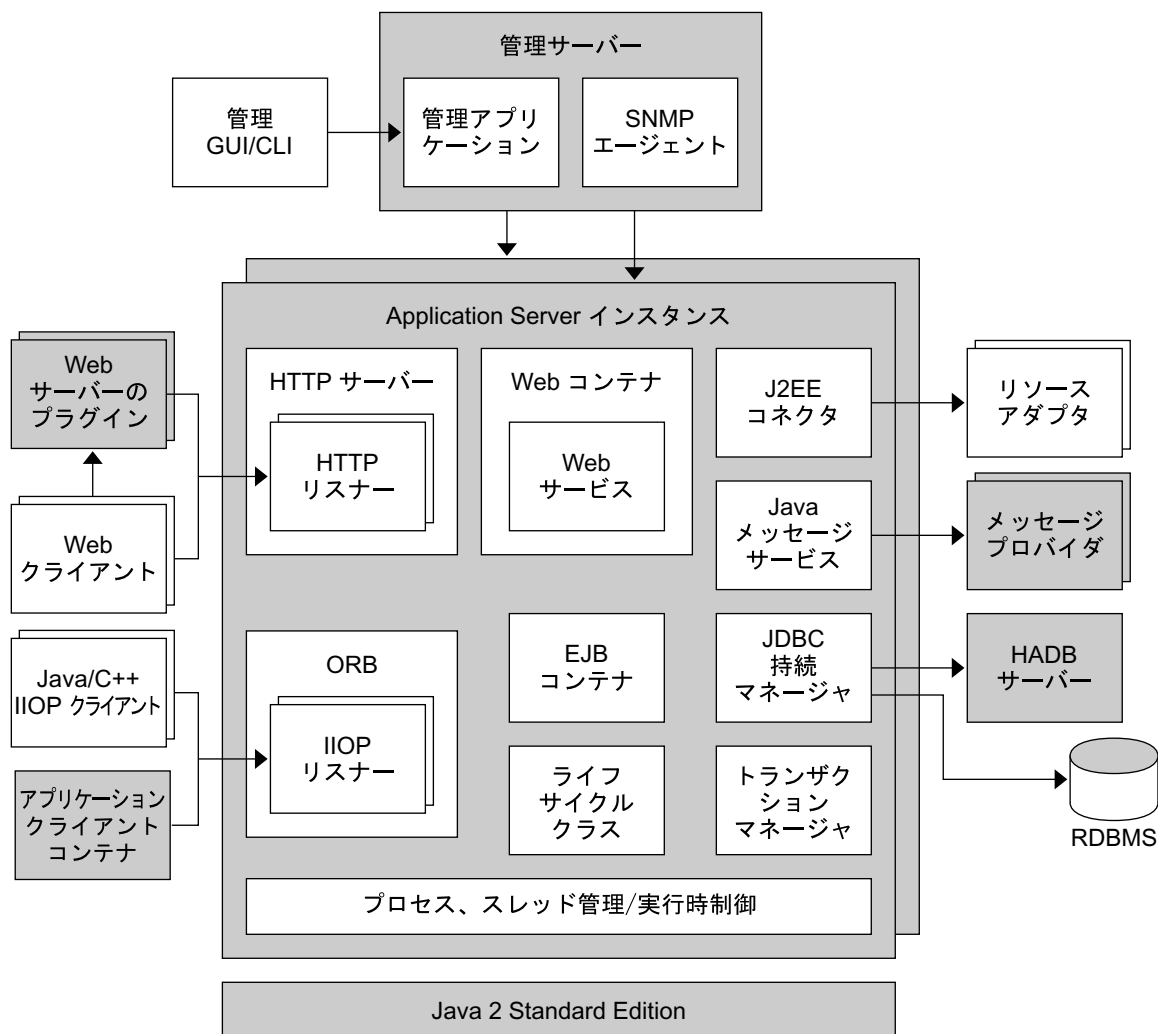
管理サーバーは、コア管理アプリケーションと SNMP エージェントを実行する、特殊なサーバーインスタンスです。リモート管理は、すべて管理サーバー経由で行われます。管理サーバーには、コマンド行や Web ブラウザベースの管理クライアントから HTTP を使用するか、セキュリティ保護された HTTPS を使って安全に直接アクセスします。

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、Application Server 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 Application Server (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、Application Server 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 に、Application Server インスタンスの詳細を示します。Application Server インスタンスは、20 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 Java System Application Server インスタンス



管理ドメインについて

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 Application Server 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 Application Server インスタンスを制御します。図 1-1 では、単一の管理ドメインを示しています。

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

Sun Java System Application Server Enterprise Edition 7 2004Q2 は、高度なクラスタリングおよびフェイルオーバーテクノロジーを備えています。これらの機能によって、拡張性と可用性の高い J2EE アプリケーションを実行できます。

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
- 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

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。クラスタ内のそれぞれの Application Server インスタンスには、同じ設定と同じアプリケーションが備えられています。Application Server クラスタの詳細については、31 ページの第 2 章「クラスタリングのシナリオ」を参照してください。

クラスタ内の Application Server インスタンスは、異なるマシン上でも同じマシン上でも実行することができます。つまり、複数のマシン上にある Application Server インスタンスを、1 つの論理クラスタに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は、同じマシン上の 2 つのインスタンス用に設定されています。

クラスタリングの詳細については、『Sun Java System Application Server Administration Guide』を参照してください。

Sun Java System Application Server のクラスタを使用すると、次の機能を実現できます。

- [高可用性](#)
- [スケーラビリティ](#)
- [ロードバランス](#)

高可用性

Sun Java System Application Server は、クラスタ内の Application Server インスタンスをフェイルオーバーによって保護することで、高可用性を実現します。ある Application Server インスタンスが停止しても、別の Application Server インスタンスが、その利用できなくなったサーバーに割り当てられていたセッションを引き継ぎます。

セッション情報は、Sun Java System Application Server Enterprise Edition に付属する高可用性データベース (HADB) に格納されます。HADB の詳細については、[28 ページの「高可用性データベース」](#)を参照してください。

Sun Java System Application Server は、次の場合でのフェイルオーバーをサポートしています。

- [HTTP/S セッション](#)
- [HTTP セッション内の EJB 参照](#)
- [RMI/IIOP による、EJB および JNDI 初期コンテキストのリモート参照](#)
- [JMS 接続](#)

HTTP/S セッション

ロードバランサプラグインは、セッションを処理していた元の Application Server インスタンスが利用できなくなると、HTTP/S 接続と関連のセッション情報を、別の Application Server インスタンスへ引き継ぎます。高可用性を有効にしている場合、セッション情報は HADB に格納されます。

ロードバランサプラグインは、次の 2 つの方法で HTTP/S セッションを追跡します。

- [Cookie](#)
- [URL の明示的な書き換え](#)

クラスタ設定は、loadbalancer.xml ファイルを作成する必要があります。このファイルは、ロードバランサプラグインとともにインストールされる loadbalancer.xml.example ファイルに基づいて、手動で作成する必要があります。デフォルトで、この設定例のファイルは、Web サーバーの config ディレクトリに格納されています。

HTTP/S セッションの高可用性設定の詳細については、[57 ページの第 4 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」](#)を参照してください。

HTTP セッション内の EJB 参照

HTTP セッションに格納された、`java.io.Serializable` インタフェースを実装する EJB 参照と J2EE オブジェクトのフェイルオーバーがサポートされます。詳細については、『Sun Java System Application Server Administration Guide』の「Configuring Session Persistence」を参照してください。

RMI/IIOP による、EJB および JNDI 初期コンテキストのリモート参照

RMI/IIOP アプリケーションの高可用性を有効にするには、IIOP クラスタに指定の IIOP エンドポイントを組み込むように設定する必要があります。IIOP クラスタに組み込まれた ORB リスナーは、IIOP エンドポイントと呼ばれます。IIOP エンドポイントは、管理コンソールまたは CLI を使用して設定します。

RMI/IIOP パスを介して要求を受信すると、Application Server は、クラスタ内の利用可能ないずれかの IIOP エンドポイントをランダムに、プライマリエンドポイントとして選択します。クラスタ内の他の IIOP エンドポイントは、代替エンドポイントとして指定されます。

プライマリエンドポイントが利用できなくなると、この接続に関連したリモート参照は、いずれかの代替エンドポイントに引き継がれます。

IIOP クラスタは、管理コンソールまたは CLI のどちらかを使用して設定できます。変更は、Application Server 設定ファイル、`server.xml` に登録されます。詳細については、[33 ページの「RMI/IIOP クラスタリングのシナリオ」](#)を参照してください。

注	IIOP および HTTP クラスタは、同じ Application Server インスタンスから構成されている必要があります。
---	---

JMS 接続

Sun Java System Message Queue Enterprise Edition 3.5 SP1 では、JMS 接続によるフェイルオーバーに対応しています。

スケーラビリティ

Sun Java System Application Server では、Application Server インスタンスをクラスタに追加して、システム容量を拡張できるので、J2EE アプリケーションに高度なスケーラビリティがもたらされます。サービスを中断せずに、Application Server インスタンスをクラスタに追加できます。

HTTP および RMI/IIOP ロードバランスシステムにより、要求は、クラスタ内の Application Server インスタンスに適切に配分されます。

ロードバランス

ロードバランスの目的は、複数の Sun Java System 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。ロードバランスは、HTTP/S と RMI/IIOP パスを介したアプリケーション要求に対して設定できます。

この項では次の項目を取り上げます。

- [HTTP のロードバランス](#)
- [RMI/IIOP のロードバランス](#)

HTTP のロードバランス

Sun Java System Application Server は、着信 HTTP および HTTPS 要求を、クラスタ内に設定した Application Server インスタンスに分散させます。ロードバランスは、サポートする Web サーバーにインストールされた、付属のロードバランサプラグインによって行われます。

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 Application Server インスタンスに転送されます。同一のコンテキストルートを持つこのあとの要求は、Cookie または URL の明示的な書き換えに基づいて、最初に要求を処理した Application Server インスタンスに割り当てられます。

HTTP のロードバランスおよびフェイルオーバーを設定するには、Application Server クラスタをロードバランサプラグインに対して指定する必要があります。設定の変更は、loadbalancer.xml ファイルで行う必要があります。ロードバランサ設定ファイルの詳細については、[48 ページの「loadbalancer.xml ファイルの作成」](#)を参照してください。

Sun Java System Application Server に付属のロードバランサプラグインを使用するか、サードパーティ製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、付属の HTTP ロードバランサプラグインについて説明します。ロードバランスの詳細については、『Sun Java System Application Server Administration Guide』を参照してください。

RMI/IIOP のロードバランス

Sun Java System Application Server, Enterprise Edition 7 では、JNDI 初期コンテキストに基づいて、RMI/IIOP パスでのリモート EJB 参照のロードバランスを行います。新しい JNDI 初期コンテキストが開始されるたびに、新しいターゲット Application Server インスタンスが、設定された IIOP クラスタから選択されます。

RMI/IIOP ベースの要求のロードバランスを有効にするには、RMI/IIOP クライアントアプリケーションで、ロードバランスを有効にするように、若干のコード変更を行う必要があります。付属の ORB には、ロードバランスに必要な機能が備わっています。

次の RMI/IIOP クライアントについて、ロードバランスがサポートされています。

- Application Client コンテナ (ACC) 内で実行し、Application Server インスタンス上に配備された EJB にアクセスする Java アプリケーション
- ACC 内では実行せず、Application Server インスタンス上に配備された EJB にアクセスする Java アプリケーション

RMI/IIOP クライアントでロードバランスを有効にする設定は、クライアントの種類によって異なります。設定例については、[33 ページの「RMI/IIOP クラスタリングのシナリオ」](#)を参照してください。さまざまな RMI/IIOP クライアントの設定に関する詳細については、『Sun Java System Application Server Developer's Guide to Clients』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 Java System 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/S または EJB セッションは、別のサーバーインスタンスに引き継がれることが保証されます。Sun Java System Application Server は次の持続性をサポートしています。

- HTTP セッション
- HTTP セッション内の EJB 参照
- ステートフルセッション Beans (SFSB)

Sun Java System Application Server に付属の高可用性データベース (HADB) は、持続性ストアとして機能します。高可用性データベースの詳細については、[28 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](#)
- [シングルスサインオンセッション情報について](#)
- [SFSB チェックポイントについて](#)

セッション持続性のタイプについて

Sun Java System 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 (高可用性) 持続性タイプは、HADB に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。

管理インターフェースを使用して、HADB 持続性ストアを選択します。次の手順に従って、HADB を持続性ストアに設定します。

- a. サーバーインスタンスの下の「可用性サービス」コンポーネントを開きます。
- b. 「可用性サービス」ページに移動します。
- c. 「インスタンスレベルの可用性」ボックスをチェックします。
- d. 「保存」ボタンをクリックします。
- e. 「持続性ストアプロパティ」の下に「プロパティ」をクリックします。
- f. 「名前」フィールドに、store-pool-jndi-name と入力します。
- g. 「値」フィールドに、HADB JDBC リソースの JNDI 名を入力します。仮のデフォルト名は、jdbc/hastore です。
- h. 「了解」ボタンをクリックします。
- i. サーバーインスタンスページに移動します。
- j. 変更を適用し、サーバーを再起動します。

HADB を設定すると、server.xml ファイルでの要素階層は次のようになります。

```
<server name="server1" ... >
    ...
    <availability-service availability-enabled="true">
        <persistence-store>
            <property name="store-pool-jndi-name"
value="jdbc/hastore"/>
        </persistence-store>
    </availability-service>
    ...
</server>
```

HADB 持続性ストアの設定方法の詳細については、『Administration Guide』を参照してください。

- file 持続性タイプでは、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。

server.xml ファイルの session-store 属性は、SFSB 状態の持続性にローカルファイルシステムが使用されている場合に、SFSB 状態が格納されている場所を指定します。次に例を示します。

```
<server name="server1" ... session-store="/export/sfsbstore">
```

- 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 Java System Application Server Administration Guide』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 Java System 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。これらのオプションについては、次の 2 つの項目で説明します。

- [持続性適用範囲の設定](#)
- [持続性頻度の設定](#)

持続性適用範囲の設定

HADB を持続性ストアとして使用する場合、持続性適用範囲を指定して、格納するセッション状態の容量を設定する必要があります。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も、セッションが変更された場合にかぎりセッションを格納することもできます。また、セッションの変更された属性だけを格納するように、持続性を設定することもできます。

次の 3 つのオプションを使用できます。

modified-session セッションは変更されたときにのみ格納されます。

session セッション状態が HADB に格納されるごとに、セッション全体が格納されます。

modified-attribute セッションの変更された属性だけが格納されます。

持続性頻度の設定

HADB を使用してセッション状態を格納する場合、HADB データベースにセッション状態を格納する頻度を設定できます。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。

次の 2 つのオプションを使用できます。

web-method web-method の持続性頻度を使用すると、Web 要求が終了するごとにセッションを格納します。

time-based time-based の持続性頻度を使用すると、server.xml ファイル (インスタンスレベル設定の場合) または sun-web.xml ファイル (アプリケーションレベル設定の場合) の manager-properties 要素の reapIntervalSeconds プロパティで定義した間隔で、セッション状態が格納されます。

注 ha 持続性タイプについてのみ、持続性頻度を選択できます。

セッション持続性のさまざまな設定オプションの詳細については、『Sun Java System Application Server Administration Guide』の「Configuring Session Persistence」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 Application Server 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再度認証は要求しません。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引き継いだ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。この持続性を使用できるのは、Application Server インスタンスの高可用性を有効にした場合であり、次のプロパティを server.xml の virtual-server 要素に追加する必要があります。

```
<property name="sso-enabled" value="true" />
```

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。

SFSB チェックポイントについて

SFSB の状態は、ライフサイクルにおける既定の時点で、持続性ストアに格納されます。

SFSB チェックポイントは、次の 5 つの異なるレベルで有効にできます。

- サーバーインスタンス
- EJB コンテナ
- アプリケーション
- EJB モジュール
- SFSB 自体

SFSB チェックポイントを所定のレベルで有効にするには、それより高いレベルでも有効にする必要があります。たとえば、SFSB チェックポイントをアプリケーションレベルで有効にするには、サーバーインスタンスレベルと EJB コンテナレベルでも有効にする必要があります。

SFSB チェックポイントの設定方法の詳細については、『Sun Java System Application Server 7 Developer's Guide to Enterprise JavaBeans Technology』を参照してください。

高可用性データベース

高可用性データベース (HADB) は、HTTP と SFSB セッション情報の格納に使用されます。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おり、99.999% を超えるデータの可用性を実現できます。このため、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、すべての種類のセッション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になります。

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。

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。HADB のセットアップと設定の詳細については、『Sun Java System Application Server インストールガイド』の「HADB の設定準備」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 Java System 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 Application Server 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。

asadmin の詳細については、『Sun Java System Application Server Administration Guide』の「Appendix A, Using the Command Line Interface」を参照してください。

- **cladmin コマンド** : cladmin コマンドにより、クラスタ内のすべての Application Server 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、可能なかぎり、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。

cladmin の詳細については、『Sun Java System Application Server Administration Guide』の「Appendix F, Using the cladmin Script for Administration」を参照してください。

- **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 Application Server 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 Application Server インスタンスのログファイルの表示にも使用できます。

管理コンソールの詳細については、『Sun Java System Application Server Administration Guide』の「Part 1, Server Basics and Administering Global Settings」を参照してください。

- **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 Java System Application Server インストールガイド』を参照してください。さらに、clsetup で提供されるマニュアルページも参照してください。

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。不確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だけ変更されますが、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。

クラスタリングのシナリオ

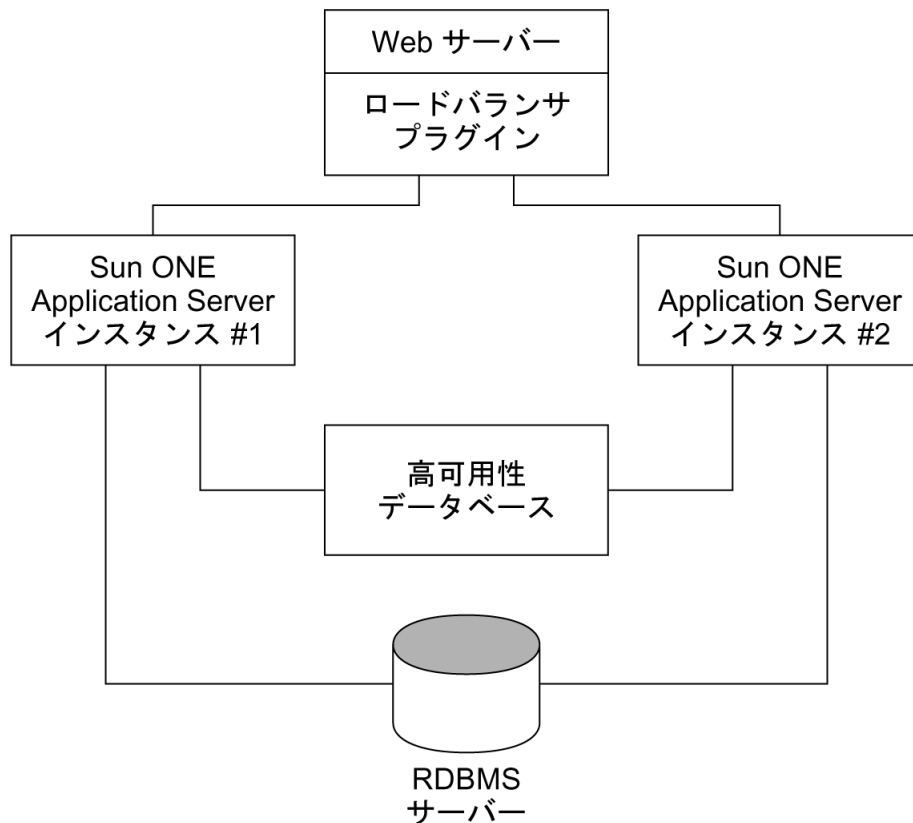
この章では次の項目について説明します。

- [HTTP クラスタリングのシナリオ](#)
- [RMI/IIOP クラスタリングのシナリオ](#)

HTTP クラスタリングのシナリオ

次の図に、単純なクラスタリングのシナリオを示します。この例では、ロードバランサプラグインを持つ Web Server、HADB を使用して 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 Java System 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) で構成されています。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 2-1 クラスターリングのシナリオの例



HTTP 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. 続いて、ロードバランサプラグインが、クラスタ内のいずれかの Sun Java System Application Server インスタンスに、この要求を転送します。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引き継いだ 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 Java System Application Server の配備のシナリオの詳細については、『Sun ONE Application Server System Deployment Guide』を参照してください。

RMI/IIOP クラスタリングのシナリオ

Sun Java System Application Server は、RMI/IIOP パス上でロードバランスおよびフェイルオーバーのメカニズムを使用して、可用性の高い J2EE アプリケーションを実現します。

高可用性を設定するには、RMI-IIOP 要求の処理に使用される、IIOP リスナーを組み込んだ IIOP クラスタ内に、Application Server インスタンスを構成する必要があります。IIOP クラスタに組み込まれた IIOP リスナーは、IIOP エンドポイントと呼ばれます。

以降の項では、RMI/IIOP パス上で、J2EE アプリケーションに対してロードバランス (クライアント側の設定) と高可用性 (サーバー側の変更) を有効にするために必要な変更について説明します。

- [サーバー側の設定](#)
- [クライアント側の設定](#)

サーバー側の設定

IIOP 要求のフェイルオーバーを有効にするには、Sun Java System Application Server 内に IIOP クラスタを構築するように IIOP エンドポイントを設定する必要があります。IIOP エンドポイントは、管理コンソールかコマンド行インタフェースのいずれかを使用して定義します。

RMI/IIOP 要求のフェイルオーバー先の、クラスタ内の非 SSL エンドポイントをすべて定義します。availability-service 要素の iiop-cluster プロパティで、IIOP エンドポイントを定義します。フェイルオーバーを有効にするため、availability-enabled="true" を必ず設定します。

次の例では、server.xml ファイルの IIOP クラスタ設定プロパティを示しています。

```

<availability-service availability-enabled="true">
  <iiop-cluster>
    <iiop-server-instance name=server1>
      <iiop-endpoint id=s1_ep1 host=trident port=3700 />
      <iiop-endpoint id=s1_ep2 host=trident port=3800 />
      <iiop-endpoint id=s1_ep3 host=trident port=3900 />
    </iiop-server-instance>
    <iiop-server-instance name=server2>
      <iiop-endpoint id=s2_ep1 host=jupiter port=4700 />
      <iiop-endpoint id=s2_ep2 host=jupiter port=4800 />
      <iiop-endpoint id=s2_ep3 host=jupiter port=4900 />
    </iiop-server-instance>
  </iiop-cluster>
</availability-service>

```

クライアント側の設定

RMI/IIOP 要求のロードバランスを有効にするには、クライアントアプリケーションを設定する必要があります。次の 2 つのタイプの RMI/IIOP クライアントについて、ロードバランスがサポートされています。

- [スタンドアロンのクライアント](#)
- [ACC クライアント](#)

スタンドアロンのクライアント

スタンドアロンのクライアントでロードバランス機能を有効にするには、JNDI 環境プロパティまたはシステムプロパティで、次のプロパティを定義する必要があります。

- `java.naming.factory.initial`
このプロパティを `com.sun.appserv.naming.S1ASCtxFactory` に設定します。
- `com.sun.appserv.iiop.endpoints`
`server.xml` で定義した IIOP エンドポイントのリストを指定します。
- `com.sun.appserv.iiop.loadbalancingpolicy`

エンドポイントが指定されている場合、このプロパティが、ロードバランスポリシーを指定するために使用されます。このプロパティの定義に使用される値は、JNDI 初期コンテキストに基づきます。

RMI/IIOP クライアントが、リモートオブジェクトで、InitialContext 検索を呼び出すと、Application Server インスタンス (IIOP クラスタの一部) がランダムに選ばれて、Bean を作成します。この InitialContext を使用した以降のすべての操作は、同じ Application Server インスタンスに割り当てられるか、スタックされます。

RMI/IIOP クライアントにロードバランス機能を実装するには、次の手順を行います。

1. 次の JVM プロパティを指定して、ORB を設定します。

```
com.sun.CORBA.connection.ORBConnectionFactoryClass=com.sun.enterprise.iiop.EEIIOPSocketFactory
org.omg.PortableInterceptor.ORBInitializerClass=com.sun.enterprise.iiop.EEORBInitializer
```

2. appserv-rt.jar と appserv-rt-ee.jar へのクラスパスを設定します。これらの jar ファイルは *install_dir/lib* ディレクトリにあります。
3. InitialContext のインスタンス化の前に、S1ASCtxFactory クラスの次のプロパティを使用します。

```
Properties env = new Properties();
env.put("java.naming.factory.initial",
"com.sun.appserv.naming.S1ASCtxFactory");

env.put("com.sun.appserv.iiop.endpoints"."trident:3600,
exodus:3700");

env.put("com.sun.iiop.loadbalancingpolicy", "ic-based");

//create an initial naming context
Context initial = new InitialContext(env);
```

このクライアントコードにより、新しい InitialContext (env) が呼び出され、JNDI 初期コンテキストオブジェクトがインスタンス化されます。ただし、env は、JNDI SPI プロパティのリストです。

ACC クライアント

ACC クライアントでロードバランス機能とフェイルオーバー機能を有効にするには、sun-acc.xml で次のプロパティを定義する必要があります。

- com.sun.appserv.iiop.endpoints
- com.sun.appserv.iiop.loadbalancingpolicy

sun-acc.xml ファイルで、ロードバランスプロパティを定義して、高可用性 ACC クラ イアントを有効にします。これらのプロパティは、sun-acc.xml ファイルで、プロパ ティ要素として定義されます。

次に例を示します。

```
<client-container>

    <target-server name="qasol-e1" address="qasol-e1"
port="3700">

        <property name="com.sun.appserv.iiop.loadbalancingpolicy"
value="ic-based" />

        <property name="com.sun.appserv.iiop.endpoints"
value="qasol-e1:3700,jupiter:3800"/>

    </client-container>
```

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 Java System 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が、Sun Java System 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 Java System Application Server System Deployment Guide』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 Java System Web Server
- 1 つのロードバランサプラグイン
- 1 つの Sun Java System 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)
- 1 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 Java System 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 Java System Application Server Administration Guide』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 3-1 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 Java System Web Server をインストールする	『Sun Java System Web Server インストールガイド』
2. 次のものを含む Sun Java System Application Server をインストールする	『Sun Java System 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 3-1 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 Java System 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 3-2 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	41 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	43 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	48 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	49 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サを設定する	52 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	56 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	58 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	63 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	64 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	69 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	70 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	71 ページの「アプリケーションの実行」

表 3-2 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	76 ページの「HTTP ロードバランスの確認」
14. セッション持続性を確認する	80 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	81 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 Java System 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、簡単に使用できるように、PATH 変数を設定します。

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。

終了するには「`exit`」を使用し、オンラインヘルプを表示するには「`help`」を使用します。

```
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にない場合は、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、[60 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 Java System 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 Java System 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 Java System 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 Java System Application Server プラットフォームの概要』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のポート番号は **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。

次に例を示します。

`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 `admin` 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、[45 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。左側の区画の項目をクリックすると、右側の区画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ことが原因と考えられます。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。左側の区画でサーバー名 (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、右側の区画で「ログ」タブをクリックします。

詳細は、『[Sun Java System Application Server Administration Guide](#)』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を示されます。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	<p>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。</p> <ol style="list-style-type: none">1. <code>domain_config_dir/domain1/server_instance/config/</code> ディレクトリに移動して、<code>server.xml</code> ファイルをエディタで開きます。2. たとえば次のように、<code>http-listener</code> 要素を探します。 <pre>http-listener id="http-listener-1" address="0.0.0.0" port="81"...</pre> <p>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。</p>
------------	--

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。

ヒント	<p>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、<code>index.html</code> 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 <code>server.xml</code> 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 <code>server1</code> は <code>domain_config_dir/domain1/server1/docroot/</code> に設定されます。トップページはこの場所にあります。</p>
------------	--

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。

webserver_install_dir/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。

loadbalancer.xml ファイルの例は、[コード例 3-1](#) に記載されています。

コード例 3-1 loadbalancer.xml.example ファイル

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE Application
Server 7.0//EN" "sun-loadbalancer_1_1.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"/>
  <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_1.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 Java System 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 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。

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 リスナーが 1 つあります。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 Java System 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ことがあります。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必要があるため、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_1.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、ダウンしているというフラグが立てられたすべての Sun Java System Application Server インスタンスを定期的にチェックして、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引き継が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. 次の行を見つけます。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 Java System 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 Java System 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 Java System 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 ポートのみを引き継がれます。

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 3-2 loadbalancer.xml ファイル

```
<!DOCTYPE loadbalancer PUBLIC "-//Sun Microsystems Inc.//DTD Sun ONE Application
Server 7.0//EN" "sun-loadbalancer_1_1.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4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

この章では、ロードバランサプラグインを使用した単純な HTTP フェイルオーバーについて記載しています。HTTP ロードバランサプラグインがサポートする機能をすべて記述しているわけではありません。ロードバランサプラグインが備える他の機能の詳細については、『Sun Java System Application Server Administration Guide』を参照してください。

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
- HTTP 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3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 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、必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

セッションファ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 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 <code>./cladmin <i>asadmin_command</i></code> 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。ただし、configure-session-persistence は除きます。

表 4-1 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。このコマンドは、同一マシン上で実行しているアプリケーションサーバーのすべてのインスタンスについて、1 度だけ実行する必要がある
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 Java System Application Server Administration Guide』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。

- クラスタに組み込まれたすべてのインスタンスには、管理者用の同じパスワードが必要になります。ユーザー名は、clinstance.conf ファイルの user プロパティ名で指定できます。
- 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 Java System 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。

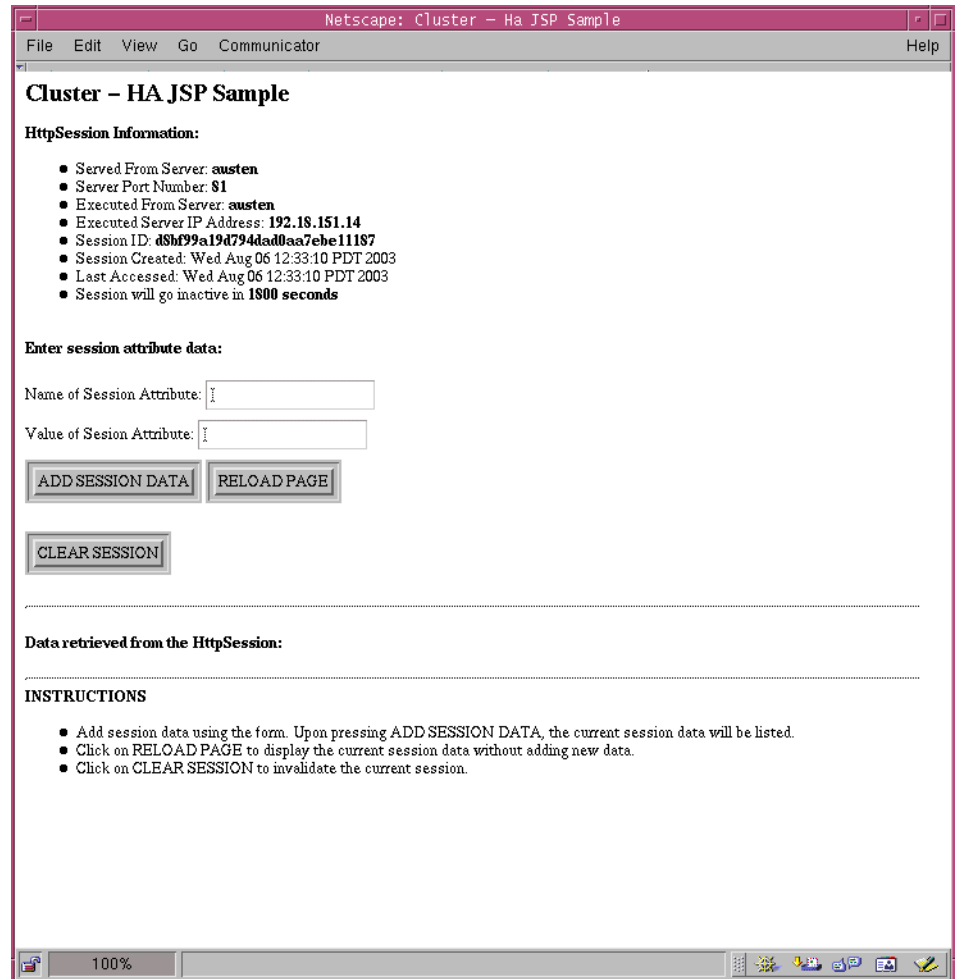
`http://host:application_server_instance_port/clusterjsp`

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 4-1 クラスタ JSP のサンプルアプリケーションページ



エラーが発生した場合は、68 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。アプリケーションサーバーインスタンスごとに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](#)
- [イベントログに記録されるアプリケーション生成メッセージ](#)
- [アクセスログに記録されるアプリケーション生成メッセージ](#)

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. 左側の区画で、ログをチェックするインスタンスをクリックします。
3. 「ログ」タブをクリックします。
4. 「イベントログを表示」をクリックします。
5. ログの表示を更新するには「了解」をクリックします。

25 項目より多くのログを表示するには、「表示するイベント数」に表示する項目数を入力し、「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. 左側の区画で、ログをチェックするインスタンスをクリックします。
3. 「ログ」タブをクリックします。
4. 「HTTP アクセスログを表示」をクリックします。

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は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。

イベントログに記録されるアプリケーション生成メッセージ

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。

```
[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変更に示されます。

アクセスログに記録されるアプリケーション生成メッセージ

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 4-2 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、 41 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	58 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 Java System 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 4-1 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
Java System 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 Server Manager 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. Server Manager 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 Server Manager 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、[55 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。

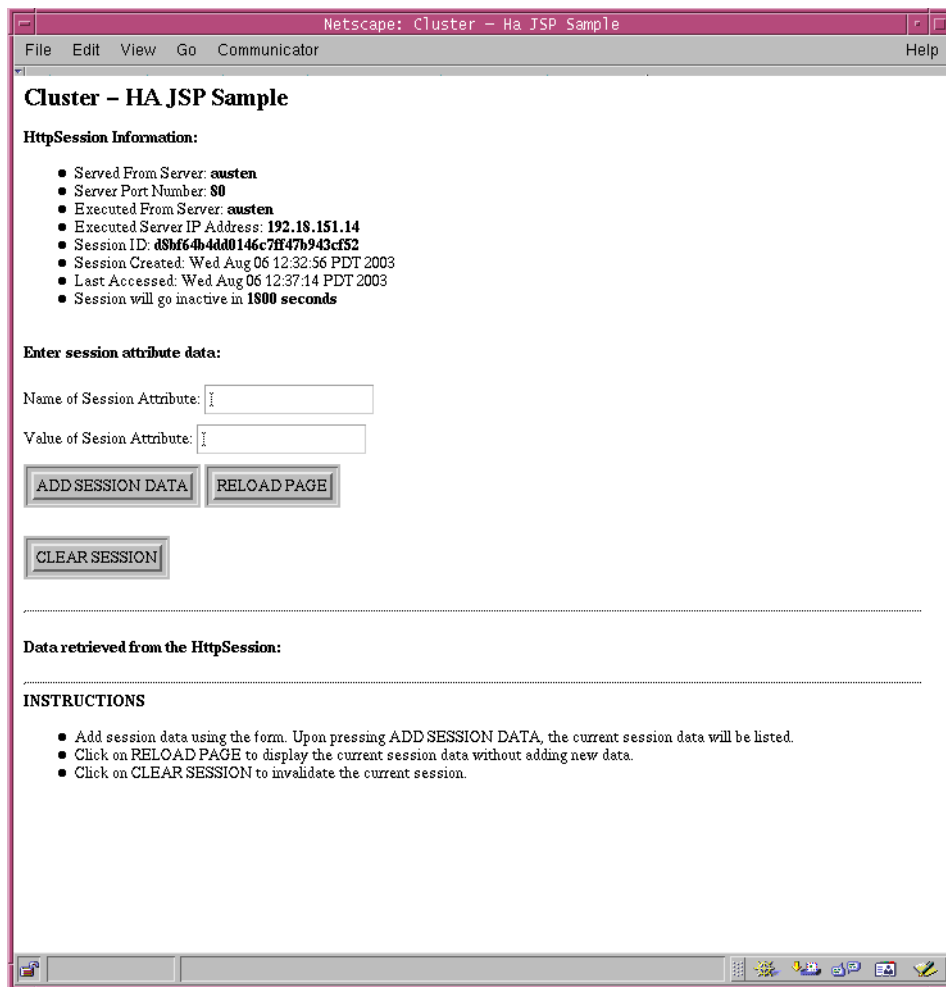
`http://host:web_server_port/clusterjsp`

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 4-2 クラスタ JSP のサンプルアプリケーションページ

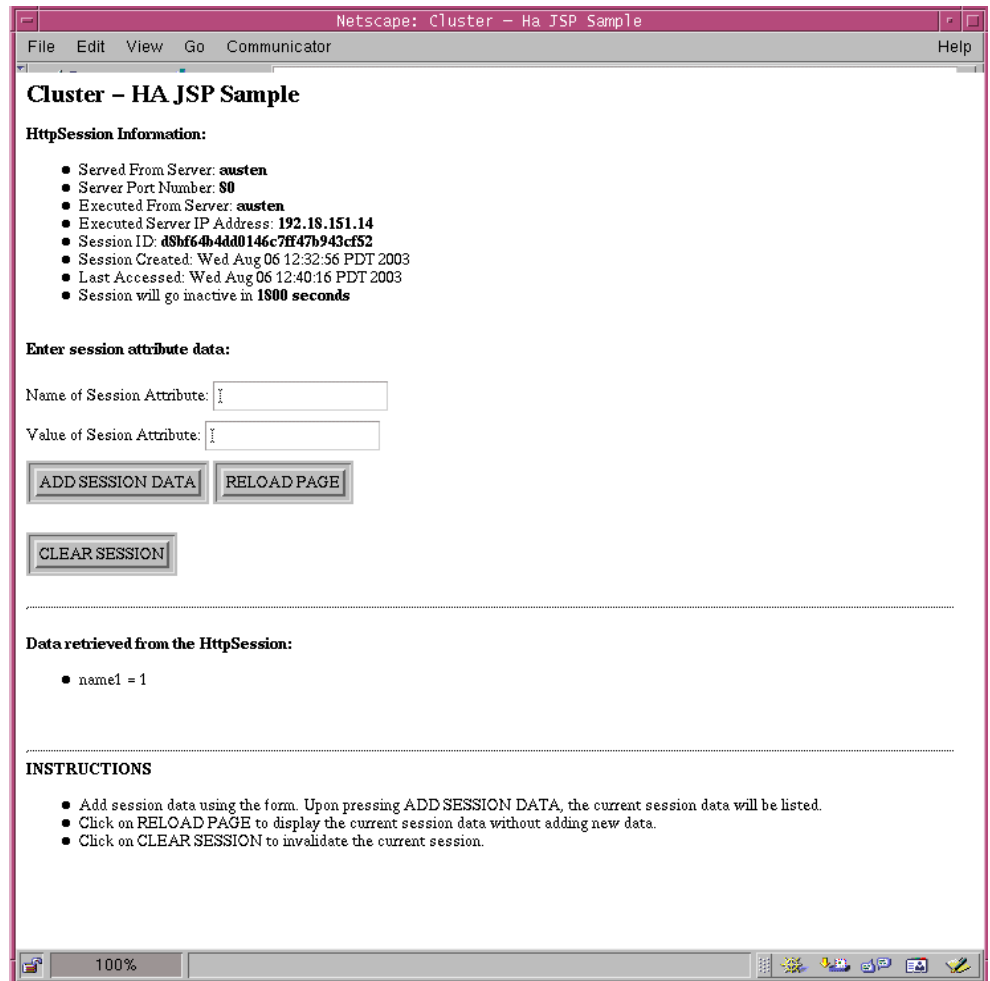


サーバー情報は、一意なセッション ID とともにページの上部に表示されます。アプリケーションサーバーインスタンス上で直接このアプリケーションにアクセスしている場合は、ページの上部に表示されるポート情報がアプリケーションサーバー情報です。クラスタ化されたアプリケーションに Web サーバーからアクセスしている場合、ポート情報が Web サーバー情報です。

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 4-3 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。

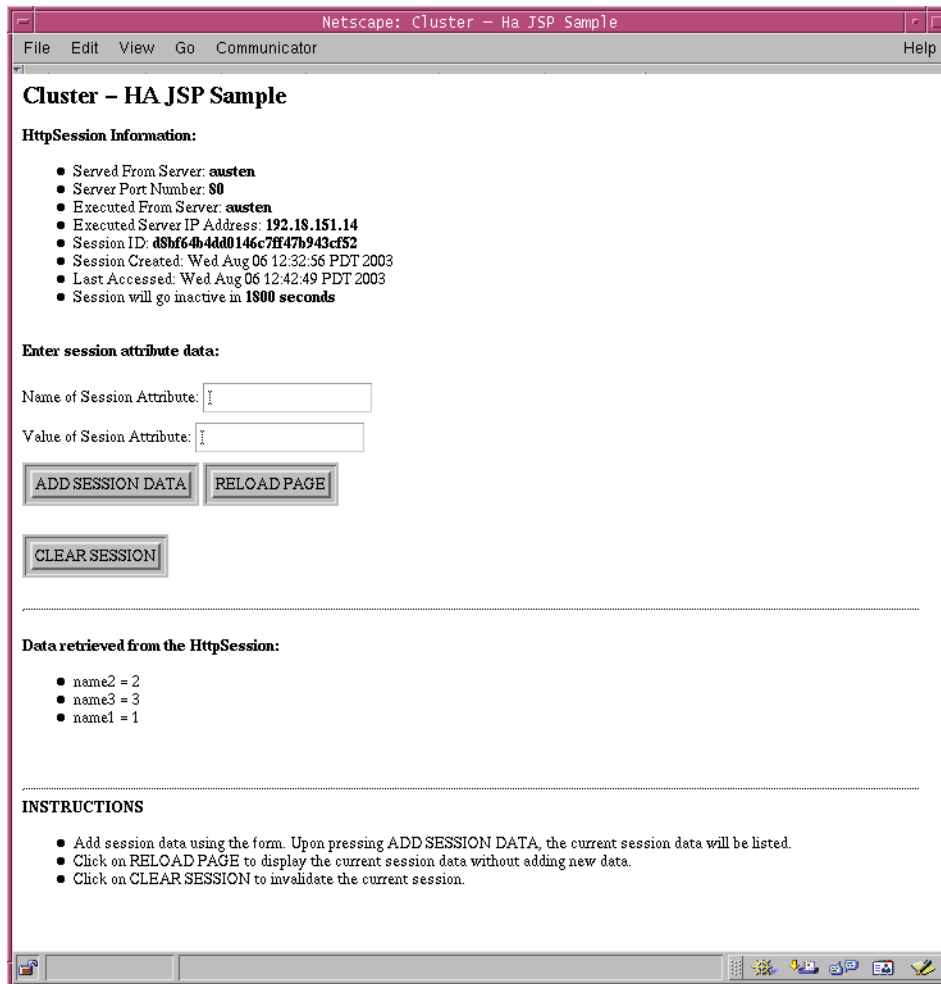
上の例では、セッション ID である d8bf64b4dd0146c7ff47b943cf52 はどちらの場合でも同じです。

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。

- 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。

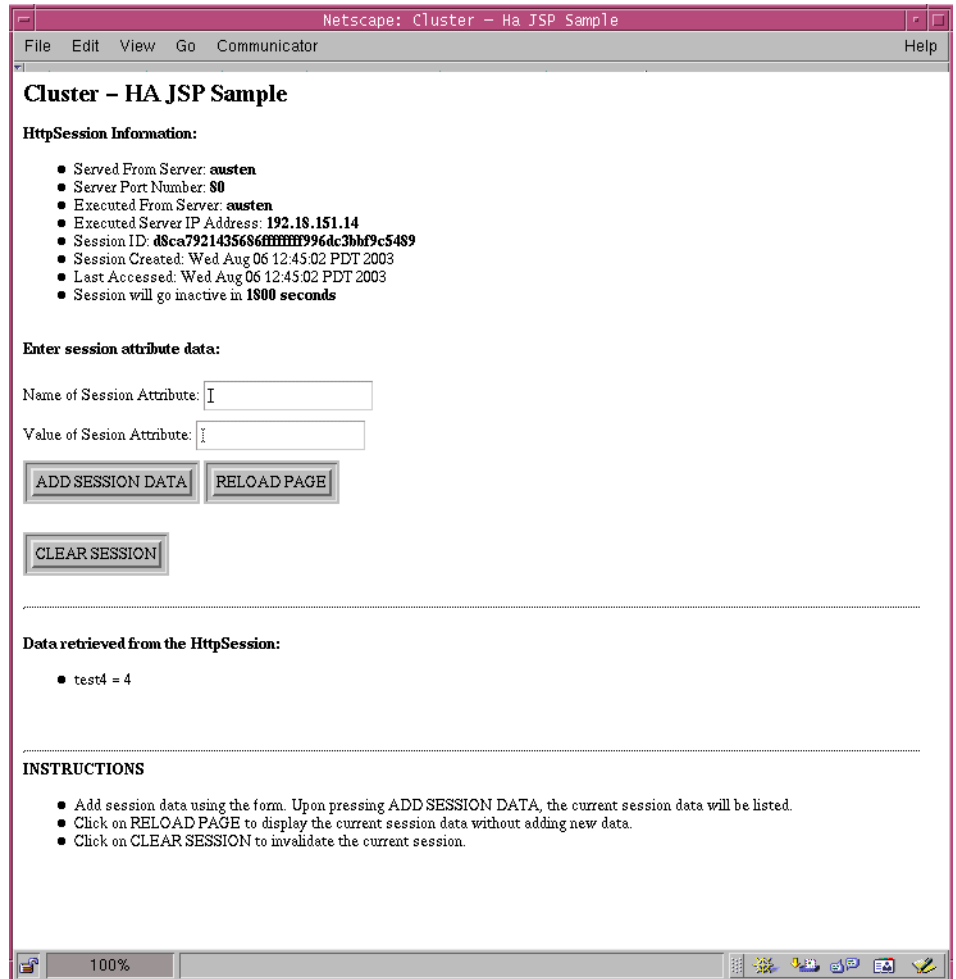
図 4-4 複数の HTTP セッション属性を含むクラスタ JSP のサンプルアプリケーション



- 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 4-5 新しい HTTP セッション情報を持つクラスタ JSP



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。

HTTP 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 Java System 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サ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 Java System 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 Server Manager で、「Logs (ログ)」タブをクリックします。
- d. 左側の区画の「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。
目で見えて調べるか、「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 Java System 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 Java System 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、66 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、53 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、次項「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 4-3 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	Web 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	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

表 4-3 ロードバランスに対するトラブルシューティング (続き)

状態	考えられる原因	解決法
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	58 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、78 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、79 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 Java System Web Server のエラーログは、Server Manager 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 Java System 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 Server Manager で、「Logs (ログ)」タブをクリックします。
4. 左側の区画の「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:Pars
ing 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:UnhealthyI
nstances cluster1 1049920562631 NoUnhealthyInstances
```

詳細ログを有効にする手順については、[53 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.c
om: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 Java System 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 Java System 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. 左側の区画で、停止するサーバーインスタンスをクリックします。
 - c. 右側の区画で「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ながざり、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、休止時間中も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 Java System Application Server Administration Guide』を参照してください。

要約と参照情報

本書『入門ガイド』では、Sun Java System Application Server 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 5-1 参照情報

目的	参照情報
Sun Java System 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	http://application_server_host_name:port/samples 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 Java System 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 5-1 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 Java System Application Server 7 の管理について調べる	『Sun Java System 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 Java System 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 Java System Application Server 7 を使った一般的な開発事例を詳しく調べる	『Sun Java System Application Server 7 開発者ガイド』を参照

索引

A

Application Server インスタンス , 18
asadmin ユーティリティ , 29, 62
 configure-session-persistence コマンド , 62
 create-jdbc-connection-pool コマンド , 62
 create-jdbc-resource コマンド , 62
 delete-jdbc-connection-pool コマンド , 62
 delete-jdbc-resource コマンド , 62
 deploy コマンド , 62
 list-instances コマンド , 46, 63
 PATH 変数の設定 , 41
 start-instance コマンド , 62
 stop-instance コマンド , 62

C

cladmin.log ファイル , 61
cladmin コマンド , 29, 59
 deploy コマンド , 61
 PATH 変数の設定 , 41
 start-instance , 63
 構文 , 60
 サポートされた asadmin コマンド , 62
 制限 , 63
 場所 , 59
 要件 , 63
 ログファイル , 61
clinstance.conf ファイル , 60

clpassword.conf ファイル , 60
clsetup コマンド , 29, 39
configure-session-persistence コマンド , 62
create-jdbc-connection-pool コマンド , 62
create-jdbc-resource コマンド , 62

D

delete-jdbc-connection-pool コマンド , 62
delete-jdbc-resource コマンド , 62
deploy コマンド , 62
disable-timeout-in-minutes, 50

F

file セッション持続性 , 25
Forte for Java, 13

H

HADB, 28
ha セッション持続性 , 24
http-listener 要素、server.xml ファイル , 47
HTTPS リスナー , 50, 55
HTTPS ルーティング , 55

HTTP アクセスログ , 66

HTTP クラスタリングのシナリオ , 31

HTTP サーバー

 トップページ , 47

 ポート , 46

 リスナー , 46

HTTP リスナー , 49

 RequestExit ログファイルエントリ , 77

I

interval-in-seconds, 52

L

list-instances コマンド , 46, 63

loadbalancer.xml.example ファイル , 48

loadbalancer.xml ファイル

 エラー , 78

 作成 , 48

 例 , 48, 56

LogVerbose, 54

M

memory セッション持続性 , 26

modified-attribute, 26

modified-session, 26

P

PATH 環境変数、設定 , 41

R

RMI/IIOP クラスタリングのシナリオの例 , 33

rpm, 15

S

session, 26

SFSB チェックポイント , 27

showrev, 14

start-domain コマンド , 41

start-instance コマンド , 62, 63

stderr, 67

stdout, 67

stop-instance コマンド , 62

Sun Java Studio, 13

sun-loadbalancer.dtd ファイル , 49

Sun カスタマサポート , 14

T

tail コマンド , 45, 67

time-based, 27

timeout-in-seconds, 52

U

undeploy コマンド , 62

W

web-method, 26

 持続性頻度の設定 , 27

Web Server

 Apache, 38

 エラーログ , 55

ログの設定 , 53
Web サーバー
変更の適用 , 70
Web サーバーのエラーログ , 78
Web サーバーへの変更の適用 , 70

あ

アクセスログ , 68
アプリケーションサーバー
起動 , 41
起動の確認 , 43
アプリケーションサーバーインスタンス
起動の確認 , 46
休止 , 81
クラスタに追加 , 49
アプリケーションサーバーインスタンスの起動 , 63
アプリケーション、サンプル
監視 , 67
休止 , 69
クラスタに追加 , 69
コピー , 58
実行 , 64
トラブルシューティング , 68

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 , 45
アプリケーションメッセージ , 67
管理サーバー , 45
インストールの要件 , 38

え

エラーログ、Web Server , 55
エラーログ、Web サーバー , 78

お

応答タイムアウト , 55

か

確認
アプリケーションの配備 , 64
サーバーインスタンスの起動 , 46
監視、ロードバランサ , 53
管理インタフェース , 29
アクセス , 44
ログの表示 , 66
管理サーバー , 18
イベントログ , 45
起動の確認 , 44
パスワード , 44
ポート , 44
ユーザー名 , 44
管理ドメイン , 20

き

休止
アプリケーション , 69
サーバーインスタンス , 81

く

クライアント側での RMI/IIOP ロードバランスの設定 , 34
クラスタ , 20
アプリケーションの追加 , 69
アプリケーションの配備 , 59
作成 , 49
シナリオの例 , 31
クラスタ JSP のサンプルアプリケーション , 58
実行 , 71

こ

高可用性のセッション持続性 , 24

コマンド行ユーティリティ , 29

さ

サーバーインスタンス , 18

 起動の確認 , 46

 クラスタに追加 , 49

サーバー側での RMI/IIOP フェイルオーバーの設定 , 33

再読み込みのポーリング間隔 , 55

サンプルアプリケーション

 WAR ファイル , 58

 監視 , 67

 クラスタ JSP , 58

 クラスタに追加 , 69

 クラスタへの配備 , 59

 コピー , 58

 実行 , 64, 71

 トラブルシューティング , 68

 配備の確認 , 64

サンプルアプリケーションの監視 , 67

し

持続性適用範囲の設定 , 26

持続性頻度

 web-method , 27

持続性頻度の設定 , 26

詳細ログ , 54

シングルサインオンセッション情報 , 27

シングルサインオン、セッション情報の利用可能性 , 27

せ

セッション持続性 , 24

 file , 25

 ha , 24

 memory , 26

 確認 , 80

 設定 , 26

セッション持続性の設定 , 26

セッション持続性のタイプ , 24

セッション持続性のタイプについて , 24

セッション持続性の有効化 , 24

ち

チュートリアル

 エンタープライズ機能の設定 , 41

 クラスタ JSP のサンプル , 58

 使用する準備 , 37

 手順 , 40

と

トップページ、HTTP サーバー , 47

ドメイン、管理 , 20

トラブルシューティング

 サンプルアプリケーション , 68

 ロードバランサ , 77

は

配備

 確認 , 64

 サンプルアプリケーションをクラスタへ , 59

 トラブルシューティング , 68

配布可能なアプリケーション , 59

パスワード、管理サーバー , 44

へ

ヘルスチェッカー , 52, 79
interval-in-seconds, 52
timeout-in-seconds, 52
URL, 52

ほ

ポート

HTTP サーバー , 46
Web サーバー , 71
アクセスできない , 45
管理サーバー , 44

ま

マニュアル

マニュアルの構成 , 11

ゆ

ユーザー名、管理サーバー , 44

ろ

ロードバランサ , 23, 53
HTTPS ルーティング , 55
loadbalancer.xml のサンプル , 56
応答タイムアウト , 55
動的再設定 , 55
トラブルシューティング , 77
プロパティ , 54
ヘルスチェッカー , 52, 79

ログ

HTTP アクセス , 66
tail コマンドを使用したモニタリング , 45, 67
Web Server エラー , 55

Web Server の詳細ログを有効にする , 54

アクセス , 68

イベント、アプリケーションサーバーインスタンス , 67

イベント、管理サーバー , 45

エラー、Web Server, 55

表示 , 66

