Sun Java System Web Server 7.0
管理ガイド

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

Part No: 820–0875
2006 年 10 月

070530@17466

目次

図目次

表目次

例目次

はじめに

このマニュアルでは、Sun Java™ System Web Server 7.0 の設定方法と管理方法について説明します。

対象読者

このマニュアルの対象読者は、本稼働環境のサーバーを管理する Sun Java System Web Server 管理者です。このマニュアルでは、次の分野の知識があることを前提としています。

- ソフトウェアのインストール
- Web ブラウザの使用
- 基本システムの管理タスクの実行
- 端末ウィンドウでのコマンドの発行

このマニュアルをお読みになる前に

Sun Java System Web Server 7.0 は、スタンドアロン製品としてインストールすることが可能です。あるいは、ネットワークまたはインターネット環境にわたって分散しているエンタープライズアプリケーションをサポートするソフトウェアインフラストラクチャーである Sun Java Enterprise System (Java ES) のコンポーネントとしてインストールすることもできます。Sun Java System Web Server 7.0 を Java ES のコンポーネントとしてインストールする場合は、`http://docs.sun.com/coll/1286.2` にあるシステムマニュアルをよく読むことをお勧めします。

Sun Java System Web Server 7.0 のマニュアルセット

Sun Java System Web Server 7.0 のマニュアルセットでは、Web Server をインストールおよび管理する方法について説明しています。Sun Java System Web Server 7.0 マニュアルの URL は、http://docs.sun.com/coll/1308.3 です。Sun Java System Web Server 7.0 への導入としては、次の表に示されている順序でマニュアルを参照してください。

表 P–1 Sun Java System Web Server 7.0 のマニュアルセットの内容

マニュアルのタイトル	内容
『Sun Java System Web Server 7.0 Documentation Center 』	タスク別、テーマ別に編成された Web Server マニュアルのトピック
『Sun Java System Web Server 7.0 リリースノート (UNIX 版)』	■ ソフトウェアと文書に関する最新情報 ■ サポートされているプラットフォームと、Web Server をインストールするためのパッチ要件
『Sun Java System Web Server 7.0 Installation and Migration Guide 』	以下のインストールおよび移行作業の実行 ■ Web Server とその各種コンポーネントのインストール。 ■ Sun ONE Web Server 6.0 または 6.1 から Sun Java System Web Server 7.0 へのデータ移行
『Sun Java System Web Server 7.0 管理ガイド』	次の管理タスクを実行します。 ■ 管理およびコマンド行インタフェースの使用 ■ サーバー環境の設定 ■ サーバーインスタンスの使用 ■ サーバーアクティビティーの監視およびログ ■ サーバー保護のための証明書および公開鍵暗号の使用 ■ サーバー保護のためのアクセス制御の設定 ■ JavaPlatform Enterprise Edition (Java EE) のセキュリティー機能の使用 ■ アプリケーションの配備 ■ 仮想サーバーの管理 ■ パフォーマンスニーズに合わせたサーバー作業負荷の定義およびシステムのサイズ決定 ■ サーバードキュメントのコンテンツと属性の検索、およびテキスト検索インタフェースの作成 ■ コンテンツ圧縮のためのサーバー設定 ■ WebDAV を使用した Web 発行およびコンテンツオーサリングのためのサーバー設定
『Sun Java System Web Server 7.0 Developer's Guide 』	以下を実行するためのプログラミングテクノロジおよび API の使用 ■ Sun Java System Web Server の拡張および変更 ■ クライアント要求に応じたコンテンツの動的生成、およびサーバーのコンテンツの変更

表 P–1 Sun Java System Web Server 7.0 のマニュアルセットの内容 (続き)

マニュアルのタイトル	内容
『Sun Java System Web Server 7.0 Update 1 NSAPI Developer's Guide』	カスタム NSAPI (Netscape Server Application Programmer's Interface) プラグインの作成
『Sun Java System Web Server 7.0 Developer's Guide to Java Web Applications 』	Sun Java System Web Server における Java サーブレットおよび JavaServer Pages™ (JSP™) テクノロジの実装
『Sun Java System Web Server 7.0 Administrator's Configuration File Reference 』	設定ファイルの編集
『Sun Java System Web Server 7.0 パフォーマンスのチューニング、サイジング、およびスケーリング』	パフォーマンス最適化のための Sun Java System Web Server の調整
『Sun Java System Web Server 7.0 Troubleshooting Guide 』	Web Server のトラブルシューティング

関連マニュアル

Sun Java Enterprise System (Java ES) とそのコンポーネントに関するすべてのマニュアルの URL は、`http://docs.sun.com/app/docs/prod/entsys.06q4` です。

デフォルトのパスとファイル名

次の表は、このマニュアルで使用するデフォルトのパスやファイル名について説明したものです。

表 P–2 デフォルトのパスとファイル名

プレースホルダ	説明	デフォルト値

install_dir	Sun Java System Web Server 7.0 のベースインストールディレクトリを表します。	Solaris™ プラットフォームへの Sun Java Enterprise System (Java ES) のインストール:
		`/opt/SUNWwbserver7`
		Linux および HP-UX プラットフォームへの Java ES のインストール:
		`/opt/sun/webserver/`
		Windows プラットフォームへの Java ES のインストール:
		System Drive:`\Program Files\Sun\JavaES5\WebServer7`
		Solaris、Linux、および HP-UX へのその他のインストールで、root ユーザーでない場合:
		user's home directory`/sun/webserver7`
		Solaris、Linux、および HP-UX へのその他のインストールで、root ユーザーの場合:
		`/sun/webserver7`
		Windows のすべてのインストールの場合:
		SystemDrive`:\Program Files\Sun\WebServer7`

instance_root	インスタンス固有のサブディレクトリを含むディレクトリ。	Solaris 上でのインスタンスのデフォルトの場所は、次のとおりです。 `/var/opt/SUNWwbsvr7`。 Linux および HP-UX 上でのインスタンスのデフォルトの場所は、次のとおりです。 `/var/opt/sun/webserver7` Windows 上でのインスタンスのデフォルトの場所は、次のとおりです。 *System Drive*`:\Program Files\sun\WebServer7` Java ES インストールの場合、Windows 上でのインスタンスのデフォルトの場所は、次のとおりです。 *System Drive*`:\Program Files\Sun\JavaES5\WebServer7`

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–3 表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 machine_name% you have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	machine_name% **su** Password:
aabbcc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、rm *filename* と入力します。
『』	参照する書名を示します。	『コードマネージャー・ユーザーズガイド』を参照してください。
「」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

コード例は次のように表示されます。

- C シェル

  machine_name% **command y|n** [*filename*]

- C シェルのスーパーユーザー

  machine_name# **command y|n** [*filename*]

- Bourne シェルおよび Korn シェル

  $ **command y|n** [*filename*]

- Bourne シェルおよび Korn シェルのスーパーユーザー

```
# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

記号の規則

次の表は、この用語集で使用される記号の一覧です。

表 P–4 記号の規則

記号	説明	例	意味
[ ]	省略可能な引数やコマンドオプションが含まれます。	ls [-l]	-l オプションは必須ではありません。
{ | }	必須のコマンドオプションの選択肢を囲みます。	-d {y|n}	-d オプションには y 引数か n 引数のいずれかを使用する必要があります。
${ }	変数参照を示します。	${com.sun.javaRoot}	com.sun.javaRoot 変数の値を参照します。
-	同時に押すキーを示します。	Control-A	Control キーを押しながら A キーを押します。
+	順番に押すキーを示します。	Ctrl+A+N	Control キーを押してから放し、それに続くキーを押します。
→	グラフィカルユーザーインタフェースでのメニュー項目の選択順序を示します。	「ファイル」→「新規」→「テンプレート」	「ファイル」メニューから「新規」を選択します。「新規」サブメニューから「テンプレート」を選択します。

マニュアル、サポート、およびトレーニング

Sun のサービス	URL	内容
マニュアル	http://jp.sun.com/documentation/	PDF 文書および HTML 文書をダウンロードできます。
サポートおよびトレーニング	http://jp.sun.com/supportraining/	技術サポート、パッチのダウンロード、および Sun のトレーニングコース情報を提供します。

◆ ◆ ◆ 第 1 章

1 基本的な使用方法

この章では、Sun Java System Web Server 7.0 の基本について説明します。具体的には、このマニュアルで使用する用語について簡単に説明します。

概要

Sun Java System Web Server 7.0 はマルチプロセス、マルチスレッド対応のセキュリティー保護された Web サーバーであり、業界標準に基づいています。この製品は、どのような規模の企業にも、高い性能、信頼性、スケーラビリティー、および管理性を提供します。

Web Server 7.0 は、包括的なコマンド行インタフェースのサポート、統合された構成、ECC (Elliptic Curve Cryptography) のサポートによって拡張されたセキュリティー、およびクラスタリングのサポートを提供します。また、アプリケーションと構成を Web Server 6.0 や 6.1 から Sun Java System Web Server 7.0 に移行するのに役立つ、堅牢な組み込み移行ツールも付属しています。

新機能

Sun Java System Web Server 7.0 の新機能や拡張機能の詳細については、『Sun Java System Web Server 7.0 リリースノート (UNIX 版)』の第 1 章「Sun Java System Web Server リリースノート」を参照してください。

管理サーバーの起動

管理インタフェースを使用するには、管理サーバーを起動する必要があります。

Unix/Linux での管理サーバーの起動

管理サーバーを起動するには、次のタスクを実行します。

▼ Unix/Linux での管理サーバーの起動

1. `install_root/admin-server/bin` ディレクトリ (`/usr/sjsws7.0/admin-server/bin` など) に移動します。
2. `./startserv` と入力します。
   このコマンドは、インストール中に指定されたポート番号を使って管理サーバーを起動します。

Windows での管理サーバーの起動

Sun Java System Web Server インストールプログラムは、Windows プラットフォーム用のアイコンをいくつか含むプログラムグループを作成します。このプログラムグループに含まれるアイコンは、次のとおりです。

- リリースノート
- 管理サーバーを起動
- Web Server をアンインストール

管理サーバーはサービスアプレットとして実行されるため、コントロールパネルを使ってこのサービスを直接起動することも可能です。

サーバーの異なる管理方法

Sun Java System Web Server は次のユーザーインタフェースを使って管理できます。

- 管理コンソール (GUI)。
- コマンド行インタフェース (`wadm` シェル)。

`wadm` シェルインタフェース (この章の後のほうで説明)、Web ベースの管理コンソールのいずれかを使ってインスタンスを管理できます。管理ノードでは、ある特定の構成のインスタンスは 1 つしか実行できません。

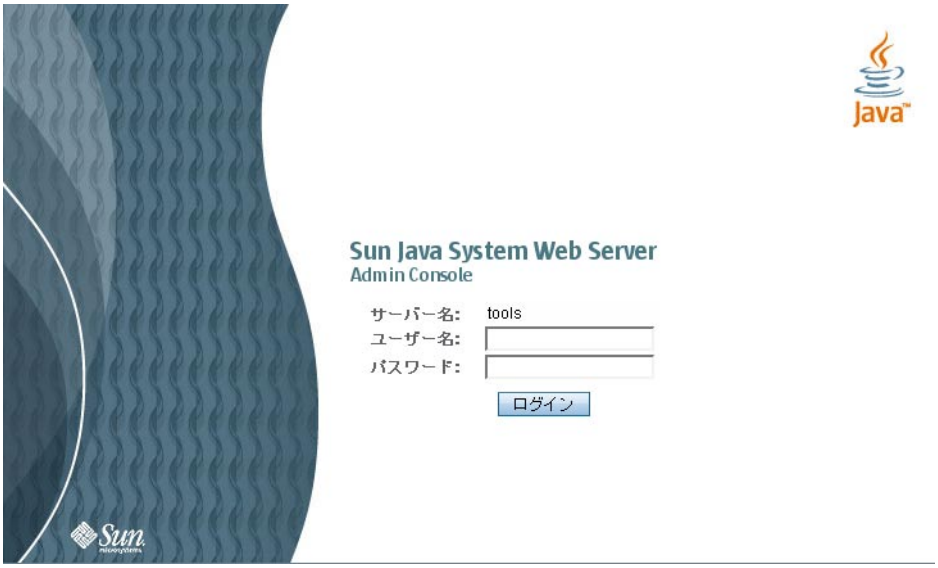
管理コンソールの使用

Sun Java System Web Server のインストールが完了したら、ブラウザを使って管理コンソールにアクセスします。

管理サーバーページへのナビゲーションに使用する URL は、Sun Java System Web Server のインストール時にどのコンピュータホスト名とポート番号を管理サーバー用として選択するかによって異なります。たとえば、SSL ポート 1234 を使って管理サーバーをインストールした場合、URL は次のようになります。

```
https://myserver.sun.com:1234/
```

サーバー管理を実行するには、管理コンソールにログインする必要があります。Sun Java System Web Server をコンピュータにインストールするときには、管理者のユーザー名とパスワードを設定します。次の図に認証画面を示します。



管理サーバーへのアクセス時に最初に表示されるページは、共通操作ページです。このページ上のボタンを使って Sun Java System Web Server の管理、追加、削除、および移行を行います。共通操作ページを次の図に示します。

注– これらのタブのいずれかをクリックすると、ページ上に子タブが表示されることがあります。子タブによって提供されるアクションは、その親タブの機能に固有のものとなります。

次の図に、ある選択されたタブの子タブを示します。

タブをクリックすると、同じウィンドウ内にページが開きます。タスクによっては、一連の手順を通じてユーザーからデータを収集することもあります。そのようなタスク向けに、管理コンソールにはウィザードインタフェースが用意されています。これらのウィザードは常に、新しいウィンドウ内に開きます。

管理コンソール GUI 画面のヘルプ

すべてのフォーム要素と GUI コンポーネントには、検証や省略可能なパラメータに関する情報を提供する詳細なインラインヘルプが用意されています。ウィザードインタフェースの場合、ウィザード内の任意の手順でヘルプタブをクリックすると、その現在のタスクに固有のヘルプが表示されます。

CLI の使用

この節では、Sun Java System Web Server 7.0 のコマンド行インタフェースについて説明するとともに、サーバーの構成および管理用としてサポートされているすべてのコマンドを定義します。

Sun Java System Web Server 7.0 では、`wadm` と呼ばれる新しい CLI が導入されました。

以前のバージョンのサーバーではいくつかの個別のコマンド行がサポートされていましたが、それらのコマンド行をすべて合わせても、GUI で提供される全管理機能の一部にしか相当しませんでした。Sun Java System Web Server 6.1 でサポートされていたコマンド行インタフェースは、`HttpServerAdmin`、`wdeploy`、および `flexanlg` です。新しい CLI (`wadm`) には次の機能が備わっています。

- スクリプティング用の組み込み JACL シェル。
- 拡張可能な CLI — 他社製プラグインを使って別のコマンドを CLI に追加できます。

注 – Sun Java System Web Server 7.0 は `HttpServerAdmin` をサポートしません。

注 – `wdeploy` は、バージョン 6.x との下位互換性を確保するためだけに Sun Java System Web Server 7.0 でサポートされており、管理サーバーノード上でしか動作しません。

CLI のモード

`wadm` は、異なる 3 つのモードでの起動をサポートしています。それらを次に示します。

- スタンドアロンモード — このモードでは、コマンドシェルから `wadm` を起動するときに、目的のコマンド、オプション、およびオペランドを指定します。コマンドの実行が終了すると、CLI も終了してシェルに戻ります。このモードは、コマンドの対話型実行と非対話型実行の両方をサポートできます。デフォルトの対話型実行では、`--password-file` オプション経由で渡されたパスワードファイル内にパスワードがまだ指定されていなかった場合、パスワードの入力が要求されます。非対話型実行では、`--password-file` オプションを指定しないとエラーが発生します。次に例を示します。

  ```
  wadm> create-config --user=admin --password-file=./admin.pwd
  --http-port=2222 --server-name=syrinx myconfig
  ```

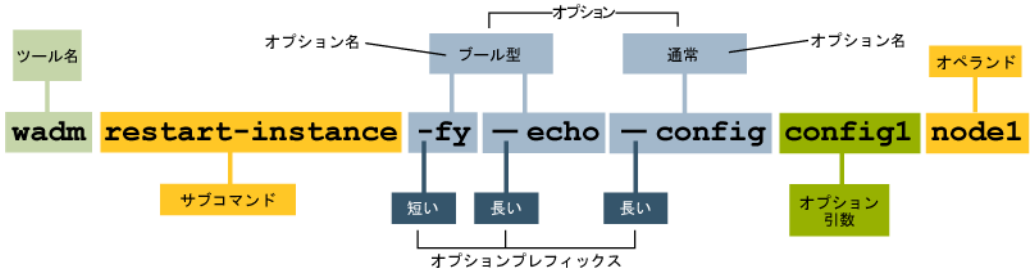
- シェルモード — このモードでは、コマンドシェルから `wadm` を起動するときに、コマンドを指定しません。`wadm` は、コマンドの入力をユーザーに要求します。コマンドの実行が完了すると、このシェルに制御が戻ります。このシェルは、exit または quit コマンドを入力することで終了できます。このモードでは、対話型および非対話型の実行を行えます。次に例を示します。

  ```
  wadm -user=admin -host=serverhost --password-file=admin.pwd --port=8989
  ```

- ファイルモード — このモードでは、一連のコマンドをファイル内に追加し、そのファイルを引数として `wadm` に渡すことができます。次に例を示します。

  ```
  wadm -user=admin -host=serverhost --password-file=admin.pwd
  --port=8989 -commands-file=/space/scripts/admscr
  ```

下図に、`wadm` コマンドを起動するための構文を示します。



注 – wadm CLI を使えば、管理コンソールで実行可能なタスクのすべてを実行できます。

wadm CLI の場所

質問: Sun Java System Web Server 7.0 管理用 CLI はどこにありますか。

回答: 管理 CLI の場所は、`install-root/bin/wadm` です。CLI を使用するには、次のことを知る必要があります。

- 管理サーバーのホスト名 (デフォルトは `localhost`)。
- 管理サーバーの SSL ポート (デフォルトは 8989)。
- 管理サーバーのユーザー名 (デフォルトは `admin`)。
- 管理サーバーのパスワード。

注 – CLI を使用するには、管理サーバーが稼働している必要があります。このサーバーは、`install-root/admin-server/bin/startserv` を実行することで起動できます。

CLI での認証

`wadm` は、管理者のユーザー名とパスワードを使って管理サーバーへの認証を行います。シングルモードで実行中の各コマンドには、有効なユーザー名とパスワードのファイルを引数として渡す必要があります。シェルモードでは、`wadm` 実行可能ファイルの起動時に、ユーザー名とパスワードのファイルを指定できます。シェルモードで起動されたコマンドでは、`user`、`password-file`、`host`、`port`、および `ssl` などの接続オプションを指定する必要はありません。指定しても、それらは無視されます。

CLI がサポートするコマンドのなかには、パスワードの入力を必要とするものもあります。たとえば、`bindpw`、`user-password`、`token-pin` などです。それらのパスワード

は、管理ユーザーのパスワードが格納されているのと同じファイル内に指定できます。コマンドに password-file を指定しないと、ユーザーはパスワードの入力を求められます。

管理サーバーで SSL が有効になっている場合には、wadm と管理サーバーとの通信は SSL 経由で行われます。管理サーバーから渡された証明書は、「トラストストア」*(~/.wadmtruststore)* に対して検証されます。証明書が存在していて、かつ有効である場合、コマンドは通常どおりに処理されます。それ以外の場合、wadm はその証明書を表示し、それを受け入れるかどうかをユーザーに尋ねます。ユーザーが証明書を受け入れた場合、その証明書は「トラストストア」に追加され、コマンドは通常どおりに処理されます。

注– 「トラストストア」をパスワードで保護する必要はありません。なぜなら、このストアには機密データが格納されないからです。

Web Server 7.0 の理解

Web Server には、サーバーファーム内のすべてのサーバーに対する拡張された分散管理機能を提供する新しい管理フレームワークが含まれています。堅牢な管理機能により、グラフィカルインタフェースとコマンド行インタフェースの両方を使用して、Web Server をリモートで管理および配備できます。サーバーは、サーバーファームの中央の 1 箇所で管理し、1 つ以上のノードに配備してサーバーインスタンスを作成できます。また、これらのサーバーインスタンスの監視機能やライフサイクル管理機能も提供されています。

Web Server は、ユーザーが各種機能を有効化/無効化したり、個々のクライアント要求への応答方法を決定したり、サーバーの動作に基づいて実行され、その動作と相互作用するようなプログラムを記述したりできるように構成されています。これらのオプションを識別する命令 (指令と呼ばれる) は、構成ファイル内に格納されます。Sun Java System Web Server は、起動時やクライアント要求の処理時に構成ファイルを読み取り、ユーザーの選択を、目標とするサーバーのアクティビティーに対応付けます。

これらのファイルの詳細については、『Sun Java System Web Server 7.0 *Administrator's Configuration File Reference Guide*』を参照してください。

Web Server 7.0 では、Web アプリケーション、構成ファイル、検索コレクションのインデックスなど、サーバーインスタンスのすべての構成可能要素は論理的にまとめられ、構成と呼ばれます。構成の作成、変更、または削除は、CLI または Web ベースの管理インタフェースを使って行えます。構成は一度に複数管理できます。また、構成という用語は、サーバーの実行時サービスを構成する一連のメタデータを指すこともあります。たとえば、ある実行時サービスは、構成されたドキュメントルートに基づいて Web ページを提供します。構成メタデータは、組み込みサービス

や他社製プラグインを読み込み、Web ページや動的 Web アプリケーションを提供するためのデータベースドライバなど、その他のサーバー拡張を設定するために、サーバーランタイムによって使用されます。

注 – 構成に関するファイルはすべて、ファイルシステム内の構成ストアと呼ばれるリポジトリ内に格納されます。このマニュアル内で明示的に指示されていないかぎり、このリポジトリ内のどのファイルも手動で編集しないようにしてください。

Web Server では、CLI または Web ベースの管理インタフェース経由で構成が変更されるたびに、その変更がまず構成ストアに対して適用され、続いてその構成が配備されます。その結果、変更結果がインスタンスのディレクトリにコピーされます。Web アプリケーションを配備するとき、アプリケーションは次の場所の下に配備されます。

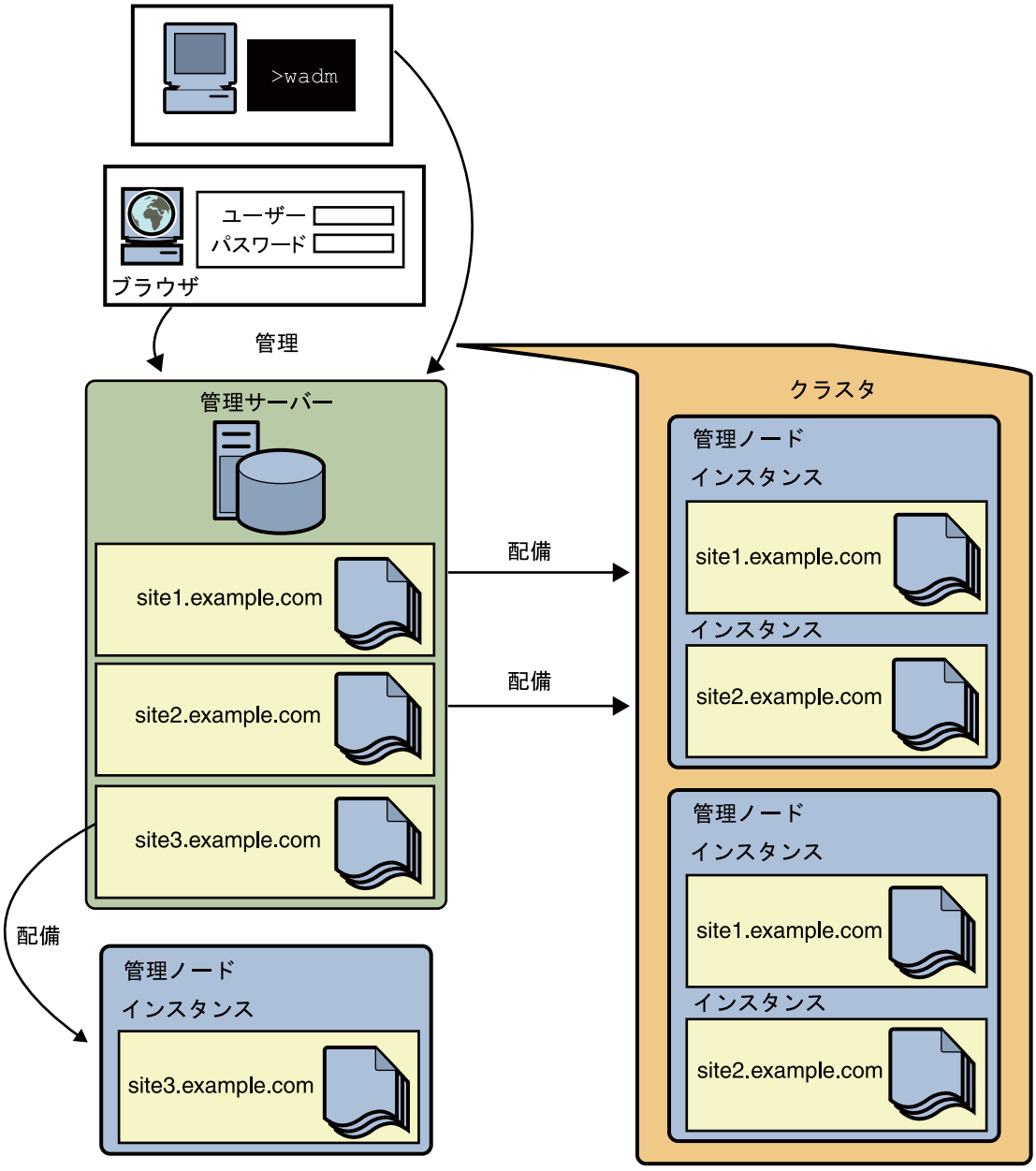
```
<install_dir>/admin-server/config-store/<config_name>/web-app/<virtual_servername>/
```

構成を配備すると、`config-store` の下にある Web アプリケーションディレクトリと構成ディレクトリの全体が圧縮され、その内容がサーバーインスタンスのディレクトリにコピーされます。このファイルは、次の場所にある `current.zip` です。

```
<install_dir>/admin-server/config-store/<config_name>
```

したがって、Web アプリケーションのサイズによっては、選択した構成の配備が完了するまでに若干の時間がかかることがあります。

次の図は、管理ノードへの構成の配備方法を示す概要図です。



構成をノード (サーバーやホストなどのネットワークリソース) に配備すると、その構成のインスタンスが作成されます。インスタンスには、インスタンスが必要とするログファイルや、ロックデータベース、キャッシュ、一時ファイルなどのその他

の実行時ファイルが含まれます。これらのインスタンスは、CLI または Web ベースの管理インタフェースを使って管理できます。

インスタンスは、1 つ以上のノードをまたがってクラスタを形成することもできます。クラスタの場合、クラスタを形成するすべてのノードの構成が同じである必要があります。1 つのクラスタ内のすべてのノードは同種である必要があります。それらは、同じオペレーティングシステムを持ち、まったく同様に構成されており、かつ同じサービスを提供している必要があります。

サーバーファーム内のある 1 つのノードでは、管理アプリケーションの配備先となるサーバーが実行されます。この特別に構成されたサーバーは管理サーバーと呼ばれ、そこに配備される管理アプリケーションは、Web ベースの管理コンソールになります。ユーザーは、管理コンソールを使ってサーバーインスタンスのライフサイクルを制御します。

管理サーバーは、そのノードにあるほかのサーバーの動作 (管理ノードと呼ばれる) を制御します。管理ノードは GUI インタフェースを提供しません。サーバーファーム内のある 1 つのノードに、管理サーバーがインストールされます。サーバーファーム内のその他のすべてのノードには、管理ノードがインストールされます。管理ノードはインストール時に管理サーバーに登録されます。このアクションにより、管理サーバーがその管理ノードを認識するようになります。

管理サーバーと管理ノードは常に SSL 経由で通信します。管理サーバーが管理ノードのサーバー証明書を信頼し、管理ノードが管理サーバーから提示されたクライアント証明書を信頼することによって、管理サーバーと管理ノードの相互認証が実現されます。ある管理ノードを登録するときに、管理サーバーによってその管理ノードのサーバー証明書が生成され、続いてその証明書がその管理ノードにダウンロードおよびインストールされます。管理ノードには、サーバー証明書の発行者の情報もインストールされます。

◆ ◆ ◆ 第 2 章

2 構成、インスタンス、およびノード

前の章では、Web Server 7.0 の新しい概念をいくつか紹介しました。管理者の主なタスクは、サーバーの実行時サービスを構成および管理することです。この章では、構成を管理するためのさまざまな方法や、構成を配備してノード上でインスタンスを起動する方法について説明します。

概要

インスタンスとは、ある特定のノード上における Web サーバーデーモンの環境のことであり、構成やログファイルを含むほか、ロックデータベースやキャッシュ、一時ファイルといったその他の実行時アーティファクトも含みます。

ノードとは、サーバーやホストなどのネットワークリソースのことです。典型的なデータセンターでは、ノードのネットワークは「サーバーファーム」と呼ばれます。この節では、管理コンソール GUI を使ってノードを構成する方法について説明します。

1 つのノードには、1 つまたは多数のインスタンスを配備できます。また、同じインスタンスを複数のノードに配備し、それらのインスタンスがそれぞれ異なるクラスタの一部となるようにすることもできます。

インスタンスを管理する目的で、インスタンスの起動、停止、再起動、および動的再構成が可能となっています。

構成の管理

構成の作成

Web Server を使い始めるには、構成を作成する必要があります。

新しい構成を作成するには、次のタスクを実行します。

1. 「構成」タブをクリックします。
2. 「新規」ボタンをクリックします。

構成作成時に利用可能な設定を案内するウィザードが表示されます。次の各節では、各ウィザードページで利用可能なフィールドについて説明します。

手順 1 – 構成の情報を設定します

このウィザードページでは、新しい構成の全般情報を設定できます。

このウィザードページで設定するパラメータは、次のとおりです。

- 構成名 — 構成の新しい一意名を追加します。
- サーバー名 — 新しい構成のサーバー名を追加します。これは構成名と同じにしてもかまいません。
- ドキュメントルート— 有効なドキュメントルートを入力します。ここでは、すべての配備済み Web アプリケーションが自身のディレクトリを維持します。デフォルト値は ../docs です。サーバー上の任意の有効なディレクトリのパスを入力できます
- **64** ビット — Web Server の 64 ビットサポートを有効化/無効化します。デフォルトは「無効」です。
- サーバーユーザー — サーバーが UNIX ベースのシステム上で実行されている場合、サーバープロセスに対する有効なユーザー名を入力します。例: root。

手順 2 — 構成のリスナーを作成します

このウィザードページでは、新しい構成の HTTP リスナーのプロパティーを設定できます。

このウィザードページで設定するパラメータは、次のとおりです。

- ポート— 構成がバインドし、要求を待機するポート番号。

- **IP** アドレス — ホストマシンの IP アドレス。利用可能なすべての IP アドレスを設定するには、* を入力します。

手順 3 — Java、CGI、および SHTML を構成します。

このウィザードページでは、Java/CGI と SHTML に関するプロパティーを構成できます。

このウィザードページで設定するパラメータは、次のとおりです。

- **Java** — 「有効」。Java はデフォルトで有効になっています。警告:この構成を使って Java ベースの Web アプリケーションを配備する必要がある場合には、Java を無効にしないでください。Java SE ディレクトリのホームを設定します。デフォルト値は、バンドルされた Java SE のディレクトリを指しているディレクトリです。デフォルトの Java SE ディレクトリを選択することも、新しいパスを指定することもできます。
- **CGI** — 「なし」(CGI サポートを無効化する)、「有効なファイルタイプ」(CGI サポートを有効化する)、および「ディレクトリ」(CGI ドキュメントの格納先となる URI とパスを指定する)。
- **SHTML** — SHTML はデフォルトで無効になっています。

手順 4 — インスタンスを作成します

このウィザードページでは、新しい構成のインスタンスを作成できます。

このウィザードページで設定するパラメータは、次のとおりです。

- 構成 — 新しい構成の名前。
- ノードの選択 — 新しい構成のインスタンスを作成するためのノードを選択します。選択可能なリストからノードを選択し、「追加」または「すべてを追加」ボタンをクリックしてノードを追加します。

注 – **CLI** の使用

CLI 経由で構成を作成するには、次のコマンドを実行します。

```
wadm> create-config --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --http-port=8800 --server-user=user
--server-name=servername config1
```

config1 は新しい構成の名前です。

CLI リファレンスの `create-config`(1) を参照してください。

サーバー構成の複製

サーバー構成をコピーして新しい構成を作成できます。新しくコピーされた構成は既存の構成と同一です。ただし、コピー元の構成がインスタンスを持っていたとしても、新しい構成はインスタンスを持ちません。

構成を複製するには、次のタスクを実行します。

1. 「構成」タブをクリックします。
2. リストから構成を選択します。
3. 「コピー」ボタンをクリックします。
4. ポップアップウィンドウで新しい構成の名前を入力し、「了解」をクリックします。

注 – **CLI** の使用

CLI 経由でこのアクションを実行するには、次のコマンドを実行します。

```
wadm> copy-config --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 copyconfig1
```

copyconfig1 は新しい構成の名前です。

CLI リファレンスの `copy-config`(1) を参照してください。

サーバー構成の配備

構成をノードに配備するには、まず構成を作成する必要があります。

既存の構成を配備するには、次のタスクを実行します。

1. 「構成」タブをクリックします
2. 構成のチェックボックスを選択することで構成を特定します。
3. 構成アクションリストから「配備の構成」を選択します。

サーバー構成の削除

注 – ある構成のインスタンスがノードに配備されている場合、その構成を削除することはできません。配備済みのインスタンスが停止中であっても、そのサーバー構成を削除することはできません。構成を削除するには、稼働中のインスタンスを停止し、その配備を取り消します。

構成を削除するには、次のタスクを実行します。

1. 「構成」タブをクリックします
2. 構成のチェックボックスを選択することで構成を特定します。
3. 構成アクションリストから「構成を削除」を選択します。

サーバーインスタンスの管理

サーバーインスタンスの作成

新しいサーバーインスタンスを作成する前に、次のチェックを行います。

1. 構成が作成済みかどうかをチェックします。新しいサーバーインスタンスを作成するときには、既存のインスタンス構成を指定する必要があります。
2. サーバーファーム内の利用可能なすべてのノードに、要求された構成のインスタンスがすでに存在していないかチェックします。重複するインスタンスを作成することはできません。

新しいサーバーインスタンスを作成するには、次のタスクを実行します。

1. 「構成」タブをクリックし、アクションリストから「新規インスタンス」を選択します。
2. 「新規インスタンスウィザード」ページで、インスタンスを作成する必要のある構成を選択し、「次へ」ボタンをクリックします。
3. 「手順 2」で選択された構成のインスタンスが存在すべきノードを選択します。「次へ」ボタンをクリックします。
4. 選択の概要を表示します。「次へ」ボタンをクリックして処理の結果を表示します。

注 – **CLI** の使用

サーバーインスタンスを作成するには、次のコマンドを実行します。

```
wadm> create-instance --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 serverhost
```

CLI リファレンスの `create-instance`(1) を参照してください。

サーバーインスタンスの起動

1. 「ノード」タブをクリックすることで、サーバー内で構成されているノードのリストを表示します。
2. ノード名のチェックボックスを選択してノードを選択します。
3. 「インスタンスを起動」ボタンをクリックします。そのノードによって制御されているすべてのインスタンスのリストを含むページウィンドウが開きます。
4. インスタンスを選択し、「インスタンスを起動」ボタンをクリックします。そのインスタンスが起動されます。
5. インスタンスの状態が「稼働中」になっているか確認し、ウィンドウを閉じます。

注 – **CLI** の使用

CLI 経由でサーバーインスタンスを起動するには、次のコマンドを実行します。

```
wadm> start-instance --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 nodehost1
```

CLI リファレンスの `start-instance`(1) を参照してください。

サーバーインスタンスの停止

1. 「ノード」タブをクリックすることで、サーバー内で構成されているノードのリストを表示します。
2. ノード名のチェックボックスを選択してノードを選択します。
3. 「インスタンスを停止」ボタンをクリックします。そのノードによって制御されているすべてのインスタンスのリストを含むページウィンドウが開きます。
4. インスタンスを選択し、「インスタンスを停止」ボタンをクリックします。そのインスタンスが停止します。
5. インスタンスの状態が「停止中」になっているか確認し、ウィンドウを閉じます。

注 – **CLI** の使用

CLI 経由でサーバーインスタンスを停止するには、次のコマンドを実行します。

```
wadm> stop-instance --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 nodehost1
```

CLI リファレンスの `stop-instance`(1) を参照してください。

サーバーインスタンスの再起動

1. 「ノード」タブをクリックすることで、サーバー内で構成されているノードのリストを表示します。
2. ノード名のチェックボックスを選択してノードを選択します。
3. 「インスタンスを再起動」ボタンをクリックします。そのノードによって制御されているすべてのインスタンスのリストを含むページウィンドウが開きます。
4. インスタンスを選択し、「インスタンスを再起動」ボタンをクリックします。そのインスタンスが再起動されます。
5. インスタンスの状態が「稼働中」になっているか確認し、ウィンドウを閉じます。

注 – **CLI** の使用

```
wadm> restart-instance --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 nodehost1
```

CLI リファレンスの `restart-instance`(1) を参照してください。

サーバーインスタンスの再構成

構成に変更を加えてもインスタンスを再起動する必要がない場合があります。管理サーバーは、サーバーインスタンスを再構成して構成ストアへの変更を取得する機能をサポートしています。この場合、サーバーを再起動しなくても構成の変更がインスタンスに反映されます。対象となるのは、動的再構成可能な構成変更だけです。

注 – user、temp-path、log、thread-pool、pkcs11、statistics、CGI、DNS、DNS-cache、file-cache、ACL-cache、SSL-session-cache、access-log-buffer、および JVM (except log-level) の設定を変更した場合、その変更は再構成後も有効になりません。そのような再起動を必要とする変更はすべて、再構成実行時にログに記録されます。ファイルキャッシュを再構成した場合、サーバーを再起動する必要があります。

1. 「ノード」タブをクリックすることで、サーバー内で構成されているノードのリストを表示します。
2. ノード名のチェックボックスを選択してノードを選択します。
3. 「インスタンスを再構成」ボタンをクリックします。そのノードに配備されているすべてのインスタンスのリストを含むページウィンドウが開きます。
4. インスタンスを選択し、「インスタンスを再構成」ボタンをクリックします。そのインスタンスが再構成されます。
5. インスタンスの状態が「稼働中」になっているか確認し、ウィンドウを閉じます。

注 – **CLI** の使用

CLI 経由でサーバーインスタンスを再構成するには、次のコマンドを実行します。

```
wadm> reconfig-instance --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 serverhost
```

CLI リファレンスの `reconfig-instance`(1) を参照してください。

サーバーインスタンスの削除

注 – 稼働中のサーバーインスタンスは削除できません。

1. 「構成」タブをクリックして利用可能な構成のリストを表示します。
2. 構成のリストから構成を選択します。
3. 「インスタンス」サブタブをクリックします。
4. 「ノード」セクションの下にある配備済みインスタンスのリストから、インスタンスを選択します。
5. アクションドロップダウンリストから「インスタンスを削除」を選択します。選択されたインスタンスが削除されます。

注 – **CLI** の使用

CLI 経由でサーバーインスタンスを削除するには、次のコマンドを実行します。

```
wadm> delete-instance --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 serverhost
```

CLI リファレンスの `delete-instance`(1) を参照してください。

インスタンスの自動構成

スケジュールされたイベントに基づいてインスタンスを再構成または再起動できます。特定の時刻や間隔を設定することで、インスタンスの自動再構成をスケジュールできます。

イベントをスケジュールするには、次のタスクを実行します。

1. 「構成」タブをクリックし、構成を選択します。
2. 「一般」サブタブ>「スケジュールされたイベント」サブタブをクリックします。

▼ スケジュールされたイベントを追加する

1 構成を選択します。
「構成」タブのクリック後に表示されるリストから、構成を選択します。

2 「一般」>「スケジュールされたイベント」サブタブをクリックします。

3 「新規」ボタンをクリックします。

4 次のプロパティーを構成します。

- イベント
  - インスタンスの再起動 — このスケジュールされたイベントは、この構成のすべての稼働中の配備済みインスタンスを再起動します。
  - インスタンスの再構成 — このスケジュールされたイベントは、この構成のすべての稼働中の配備済みインスタンスを再構成します。
  - カスタムコマンド行 — 実行されるファイルへの絶対パスを提供します。
- スケジュール

  構成されたイベント起動時刻。ドロップダウンボックスから時間と分の値を選択します。

- 毎日 — 指定されたイベントを指定された時刻に毎日起動します。
- 指定日 — 指定されたイベントを指定された日に起動します。

  1. 曜日 — 日曜から土曜までの間で、任意の曜日を指定します。

  2. 日付 — 1 から 31 までの間で、任意の日付をコンマ区切りエントリとして指定します。例: 4,23,9
- 指定月 — 指定された月の指定された時刻に指定されたイベントを起動します。1 月から 12 月までの間で、月を指定します。
- 間隔

  この期間後に指定されたイベントを起動します。

  1. 時間おき — ドロップダウンボックスから時間数を選択します。

  2. 秒おき — テキストフィールドに秒数を入力します。

注 – **CLI** の使用

CLI 経由でイベントをスケジュールするには、次のコマンドを実行します。

```
wadm> create-event --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --time=10:10 --command=restart
```

CLI リファレンスの `create-event`(1) を参照してください。

▼ スケジュールされたイベントを削除する

1. 構成を選択します。
   「構成」タブのクリック後に表示されるリストから、構成を選択します。
2. 「一般」>「スケジュールされたイベント」サブタブをクリックします。
3. スケジュールされたイベントを選択し、「削除」ボタンをクリックします。

◆ ◆ ◆ 第 3 章

3 サーバーファームとクラスタ

これまでの章では、構成について紹介し、構成をノードに配備する方法を説明しました。この章では、単純なサーバーファームとクラスタを設定します。

Sun Java System Web Server でのクラスタサポート

クラスタとは、一連のインスタンスが 1 つ以上のノードをまたがって存在し、そのすべてが同一の構成を実行し、同一の実行時サービスセットを提供する場合の、その一連のインスタンスのことです。各クラスタには、管理サーバーとして指定されたサーバーが 1 つ含まれている必要があります。クラスタが複数存在する場合、そのすべてのクラスタを単一のマスター管理サーバーから管理することができます。マスター管理サーバーは、すべてのクラスタに関する情報を取得し、各クラスタにインストールされた Sun Java System Web Server を管理するためのインタフェースを提供します。

注 – クラスタ内のすべてのインスタンスは同種である必要があります。たとえば、それらは同じバージョンのオペレーティングシステム (とパッチ) およびサービスパック上で稼働し、同じ Web サーバー構成を実行し、同じサービスを提供します。

サーバーファームの設定

クラスタを設定するには、まず 1 つの管理サーバーと 1 つ以上の管理ノードをインストールする必要があります。管理ノードを管理するには、各管理ノードを個別に管理サーバーに登録する必要があります。このアクションは、ノードのインストール中に行うことも、インストール後に wadm CLI 経由で行うこともできます。

▼ サーバーファームを設定する

1 管理サーバーと管理ノードのインストール

管理サーバーをインストールします。管理サーバーのインストールは、Sun Java System Web Server インストーラ GUI または wadm CLI を使って行えます。

「高速インストール」オプションを選択できます。このオプションを選択した場合、管理サーバーがポート 8989 上にインストールされます。あるいは、カスタムインストールオプションを選択してユーザー独自の設定を行います。管理サーバーをインストールするには、インストーラの設定画面でオプション「サーバーを管理サーバーとしてインストール」を選択します。SSL ポートは指定する必要がありますが、非 SSL ポートは指定してもしなくてもかまいません。

注 – 非 SSL ポートを指定すると管理サーバーノード内に管理ノードが作成されますが、その管理ノードを管理サーバーに明示的に登録する必要はありません。

管理ノードをインストールするには、「カスタムインストール」を選択したあと、「サーバーを管理ノードとしてインストール」を選択します。インストール用のポートを指定します。非 SSL ポートを選択するオプションは用意されていません。なぜなら、管理サーバーと管理ノード間の通信では常に、セキュリティー保護されたチャネルが使用されるからです。インストール中に、そのノードを管理サーバーに登録する必要があるか尋ねられます。インストール中にノードを登録しないことを選択した場合、wadm CLI を使えばそのアクションを実行できます。

注 – 高速インストール経由で管理ノードをインストールすることはできません。

---

2 管理ノードを管理サーバーに登録します

管理ノードがクラスタやサーバーファームの一部となるには、それらの管理ノードを管理サーバーに登録する必要があります。管理ノードを管理サーバーに登録しないかぎり、それらの管理ノードが起動されることはありません。管理ノードを登録するには、wadm CLI 経由で次のコマンドを実行します。

```
wadm> register-node --user <admin-user> --port <SSL Port> --host <node name>
```

このポートは、管理サーバーのインストール時に指定されたポートです。ホストは、管理サーバーがインストールされているノードのホスト名です。

このアクションによって、管理サーバーにノードが登録されます。

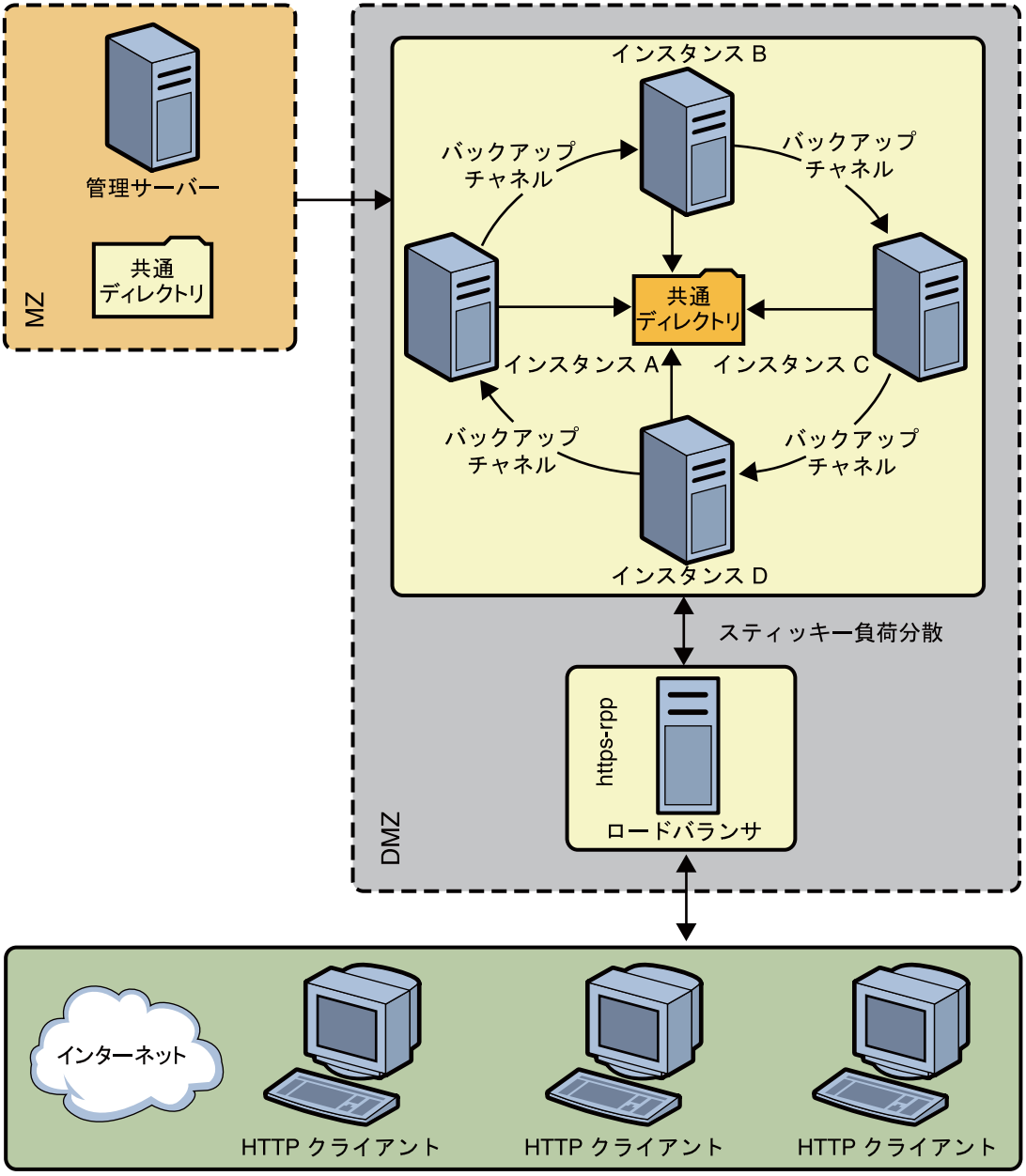
注 – ノードの登録はそのノードからしか行えません。管理サーバーの CLI にアクセスして任意のノードを登録する、といったことは行えません。また、管理サーバーへのノードの登録は、SSL モードでしか行えません。

作成されたクラスタのセッションレプリケーションを設定する方法については、192 ページの「セッションレプリケーションの構成」を参照してください。

単純なクラスタの設定

この例の一部として、1 つのロードバランサ、1 つの管理サーバー、および 4 つの Web サーバーインスタンスを持ち、セッションレプリケーションが有効化されたクラスタを設定します。セッションレプリケーションは、Java Web アプリケーションのセッションに対して高い可用性を提供します。それは、メモリー内に常駐している、ある Web サーバーインスタンスのセッションのコピーを、別の Web サーバーインスタンスにコピーすることによって実現されます。このため、通常の動作条件下では、各セッションのコピーが少なくとも 2 つ存在し、それぞれが個別の JVM 内に、そして理想的には個別のマシン上に存在することになります。

次の図は、単純なクラスタを示したものです。


管理サーバー
共通
ディレクトリ
MZ
インスタンス B
バックアップ
チャネル
バックアップ
チャネル
共通
ディレクトリ
インスタンス A
インスタンス C
バックアップ
チャネル
バックアップ
チャネル
インスタンス D
スティッキー負荷分散
https-rpp
ロードバランサ
DMZ
インターネット
HTTP クライアント
HTTP クライアント
HTTP クライアント

▼ クラスタを構成する

始める前に　次の各マシンを特定します。

- MachineA — ロードバランサと管理サーバーの両方が存在します。
- MachineB、MachineC、MachineD、および MachineE — 管理ノードと Web サーバーインスタンスが稼働します。

1. 管理サーバーを **MachineA** にインストールします。

   管理サーバーのインストール方法については、50 ページの「サーバーファームを設定する」を参照してください。典型的なインストール処理では、Web サーバーインスタンスのインストールも行われます。このシナリオでは、そのインスタンスは使用しません。

2. 管理ノードを **MachineB**、**MachineC**、**MachineD**、および **MachineE** にインストールします。

   管理ノードを 4 台のマシンすべてにインストールします。管理ノードを管理サーバーに登録します。

3. **Web** アプリケーションを構成します。

   Web アプリケーションのセッションレプリケーションを有効にします。`WEb-INF/sun-web.xml` ファイルを次のように変更します。

   ```
   <session-manager persistence-type="replicated"/>
   ```

4. インスタンスを構成します。

   - `wadm` を起動します。

     ```
     wadm --host MachineA --port 8089
     ```

   - ロードバランサ用に新しい構成を作成します。

     ```
     wadm> create-config --http-port=8080 --server-name=SampleCluster lb
     ```

   - 逆プロキシ (ロードバランサ) を設定します。

     ```
     wadm> create-reverse-proxy --config=lb --vs=lb
     -uri-prefix=/ --server="http://MachineB:8080,http://MachineC:8080,
     ttp://MachineD:8080,http://MachineE:8080"
     ```

   - インスタンスを作成します。

     ```
     wadm> create-instance --config=lb MachineA
     ```

   - 構成を配備します。

```
wadm> deploy-config lb
wadm> start-instance --config=lb
```

5 クラスタを作成して起動します。

4 つのインスタンスを持つ新しい構成を作成します。

- クラスタ用に新しい構成を作成します。

```
wadm> create-config --http-port=8080 --server-name=SampleCluster clusterOf4
```

- セッションレプリケーションを有効にします。

```
wadm> set-session-replication-prop --config=clusterOf4 enabled=true
```

- Web アプリケーションを追加します。

```
wadm> add-webapp --config=clusterOf4 --uri=/simple webapps-simple.war
```

- インスタンスを作成します。

```
wadm> create-instance --config=clusterOf4 MachineB MachineC MachineD MachineE
```

- クラスタを起動します。

```
wadm> start-instance --config=clusterOf4
```

注 – start-instance コマンドでホスト名を指定しなかった場合、このアクションによって、この構成が配備されているすべてのノードのインスタンスが起動されます。

◆ ◆ ◆ 第 4 章

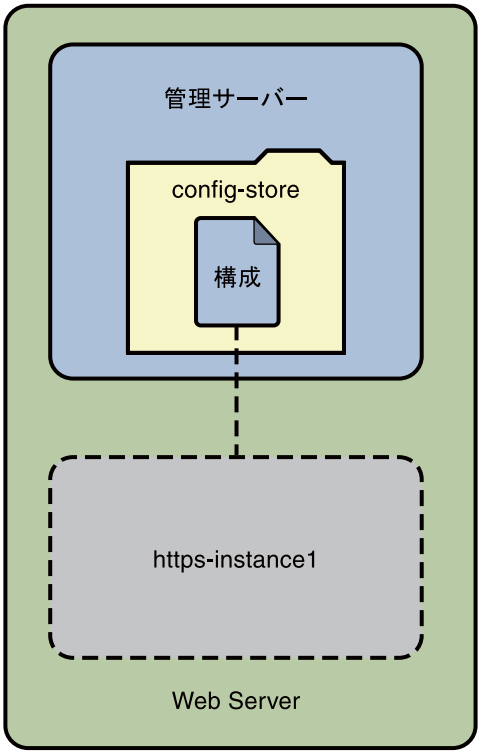
4 配備シナリオ

この章では、Sun Java System Web Server 7.0 を単一ノード上およびクラスタ環境に配備する方法について説明します。この章の内容は次のとおりです。

配備アーキテクチャー

この節では、単一ノード配備アーキテクチャーについて説明します。

次の図は、単一ノード配備設定の Web Server を表現したものです。



この図の Web Server 配備設定は、次の各要素から構成されています。

- 「管理サーバー」 - 管理サーバーとは、特別に構成された Web サーバーインスタンスのことです。管理サーバーには Web アプリケーションを配備できます。
- 「管理ノード」 - 管理ノードはサーバーファーム内のあるノード上、つまりサーバー/ホスト上に配備され、リモートの管理サーバーとの通信機能を備えています。管理サーバー内で利用可能なサーバー構成は、このノードに配備することができます。サーバーファーム内のすべての管理ノードは同種である必要があります。つまり、すべてのノードが同じオペレーティングシステムを使用し、同じハードウェアアーキテクチャーを持つ必要があります。
- 「構成」 - 構成とは、Web アプリケーション、構成ファイル、検索コレクションのインデックスなど、Web Server インスタンスのすべての一連の構成可能要素のことです。構成は作成、変更、または削除できます。Web Server では複数の構成を管理できます。1 つの構成に対して複数のインスタンスを作成できます。変更された構成を配備すると、その構成のインスタンスが更新されます。
- *config-store* これは、すべての構成の格納先となるファイルシステムベースのリポジトリです。

注意 – config-store ディレクトリの下のファイルは一切編集しないでください。このディレクトリの下のファイルは、Sun Java System Web Server によって内部用として作成されます。

config-store ディレクトリの下の構成ファイルを手動で編集する必要がある場合には、wadm deploy-config コマンドを使って構成を配備してください。

このコマンドの使用方法の詳細については、『Sun Java System Web Server 7.0 CLI Reference Manual』を参照してください。

- 「インスタンス」 - インスタンスとは、ある特定のノード上における Web サーバーの環境のことであり、構成やログファイルを含むほか、ロックデータベースやキャッシュ、一時ファイルといったその他の実行時アーティファクトも含みます。インスタンスを管理する目的で、インスタンスの起動、停止、再起動、および動的再構成が可能となっています。

配備の概要

単一ノードへの Web Server の配備は、次の目的の場合に検討できます。

- 単純な Web または CGI アプリケーションのホスティング。
- Web アプリケーションの開発やテスト。

次のフローチャートは、1 つのノード上に Web Server を配備する方法を大まかに表現したものです。

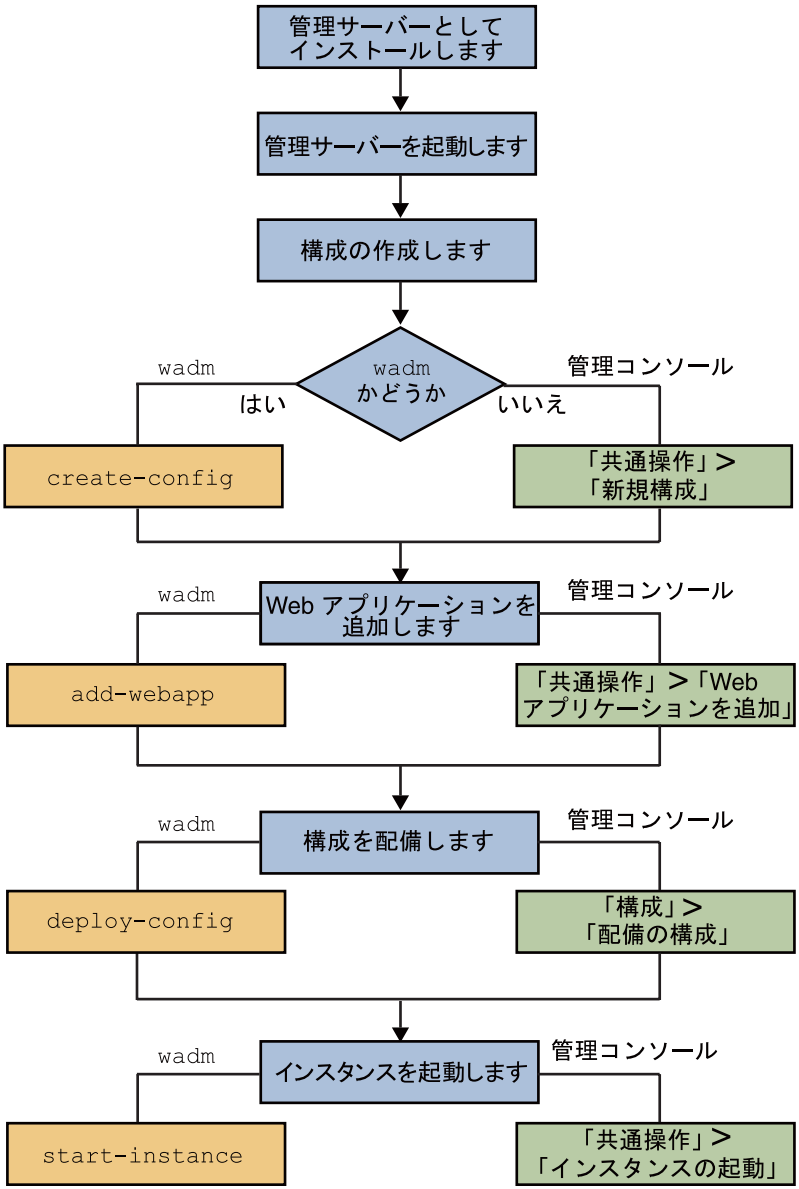


図 4–1　単一ノードへの Web Server の配備を表現したフローチャート

配備手順については、次の節で説明します。

- 59 ページの「配備前の要件」
- 59 ページの「Web Server の配備」

配備前の要件

単一ノード上に Web Server を配備するには、次のタスクを実行してシステムの準備を整えます。

1. 1 つのノード上に Web Server をインストールします。

   Web Server のインストール時に高速インストールオプションを選択した場合、次のデフォルトエンティティーが作成されます。

   - 管理サーバー。
   - HTTP リスナーと仮想サーバーを 1 つずつ含むデフォルト構成。構成と仮想サーバーの名前はホスト名と同じになります。
   - デフォルト構成のインスタンス。

   Web Server のインストール方法については、『Sun Java System Web Server 7.0 Installation and Migration Guide』の第 2 章「Installing the Web Server」を参照してください。

   サポートされているプラットフォームやシステム要件については、『Sun Java System Web Server 7.0 リリースノート (UNIX 版)』の「対応プラットフォーム」を参照してください。
2. 管理サーバーを起動します。

   指定された SSL ポート上で管理サーバーの実行が開始されます。

Web Server の配備

1 つのノード上に Web Server を配備するには、次の手順を使用します。

1. デフォルト構成を使用することも、新しい構成を作成することもできます。

   新しい構成を作成する場合は、構成の一意名を指定します。新しい構成は、仮想サーバーとデフォルト HTTP リスナーを 1 つずつ作成します。

   注 – 管理コンソールを使って構成を作成する場合、ウィザードによって新しいインスタンスを作成するためのプロンプトが表示されます。CLI を使用する場合、`create-instance` コマンドを使って構成のインスタンスを明示的に作成する必要があります。

   <*install_dir*>`/admin-server/` ディレクトリの下にある `config-store` ディレクトリ内に、すべての構成が格納されます。

注意 – config-store ディレクトリの下のファイルは一切編集しないでください。このディレクトリの下のファイルは、Sun Java System Web Server によって内部用として作成されます。

2. 変更済みの構成を配備します。

クラスタ環境

「クラスタ」とは複数のノードにまたがって存在する、複数のサーバーインスタンスのグループのことであり、それらすべてのインスタンスで同一の構成が実行されます。クラスタ内のすべてのインスタンスが連携して動作することで、高い可用性、信頼性、およびスケーラビリティーが実現されます。

クラスタで負荷分散を使えば、フェイルオーバーとセッションレプリケーションによって、中断されることのないサービスとセッションデータの持続性が実現されます。

ハードウェアとソフトウェアの要件

この節で説明するユースケースでは、Web Server クラスタは次のエンティティーから構成されます。

1) 4 つのインスタンス (4 つの同一ノード上で稼働)
2) 管理サーバー
3) HTTP 要求を負荷分散するための逆プロキシ

クラスタを設定するには、同じバージョンのオペレーティングシステムとパッチがインストールされた 2 つ以上の同一ノードが必要となります。たとえば、Solaris® 9 SPARC® オペレーティングシステムを含むマシンを選択した場合、クラスタ内のほかのマシンにも Solaris 9 SPARC をインストールする必要があります。

サポートされているプラットフォームやパッチの要件については、『Sun Java System Web Server 7.0 リリースノート (UNIX 版)』を参照してください。

次の図は、クラスタ環境を記述したものです。

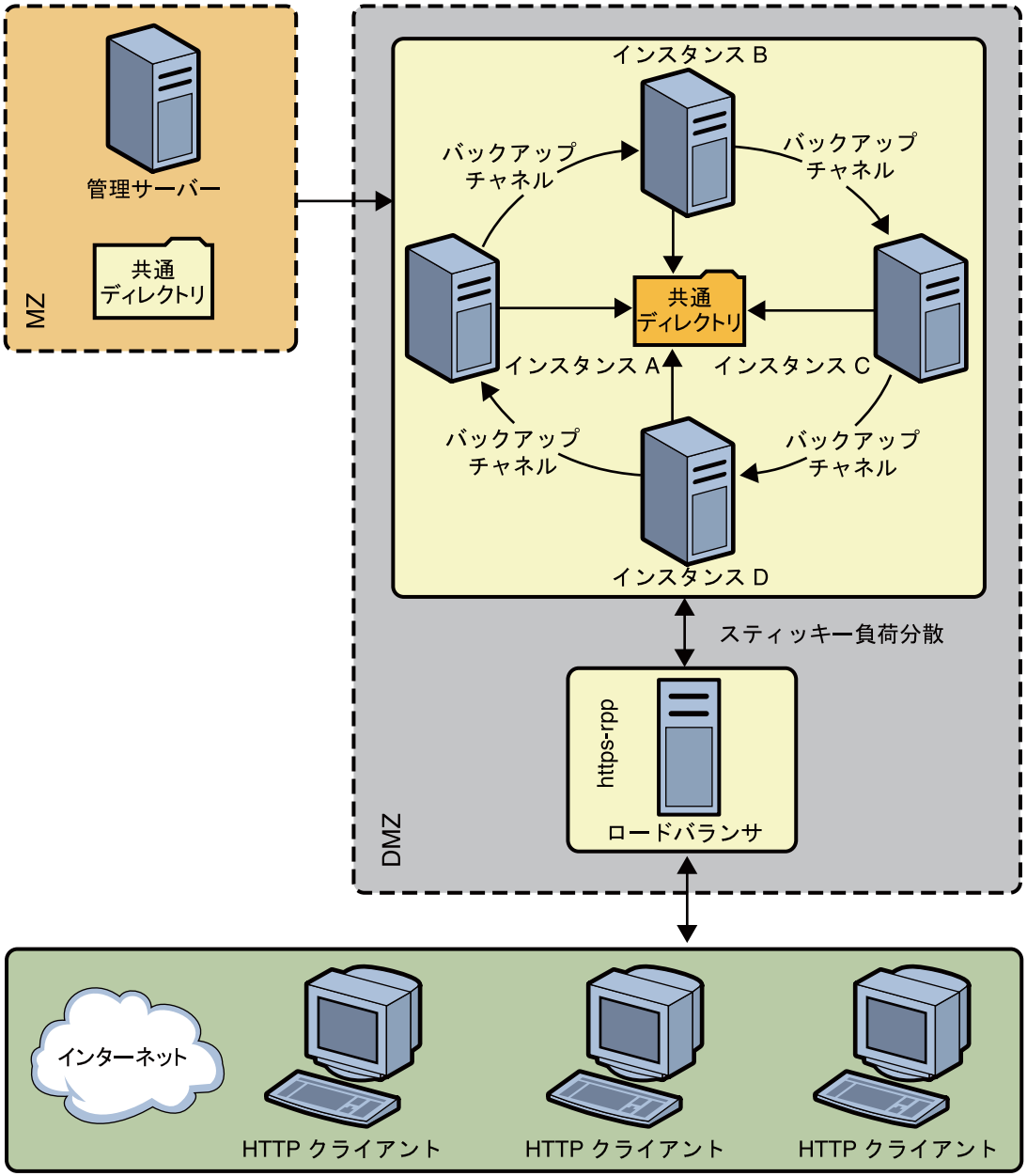


図4–2　クラスタ設定

この図では、ノードは非武装地帯 (DMZ) に構成されています。管理サーバーは、一般的なアクセスから制限/保護するために、ファイアウォールの背後にある武装地帯

に構成されています。別のノードが逆プロキシサーバーとして構成されています。逆プロキシサーバーは、セキュリティーを強化する目的で DMZ の内側に存在しています。

注 – Solaris のゾーン機能は、Solaris 10 オペレーティングシステム上でしかサポートされていません。

クラスタの設定

この節では、クラスタを設定し、逆プロキシを有効にして HTTP 要求の負荷分散をサポートする手順について説明します。

次のフローチャートは、クラスタの設定手順を示したものです。

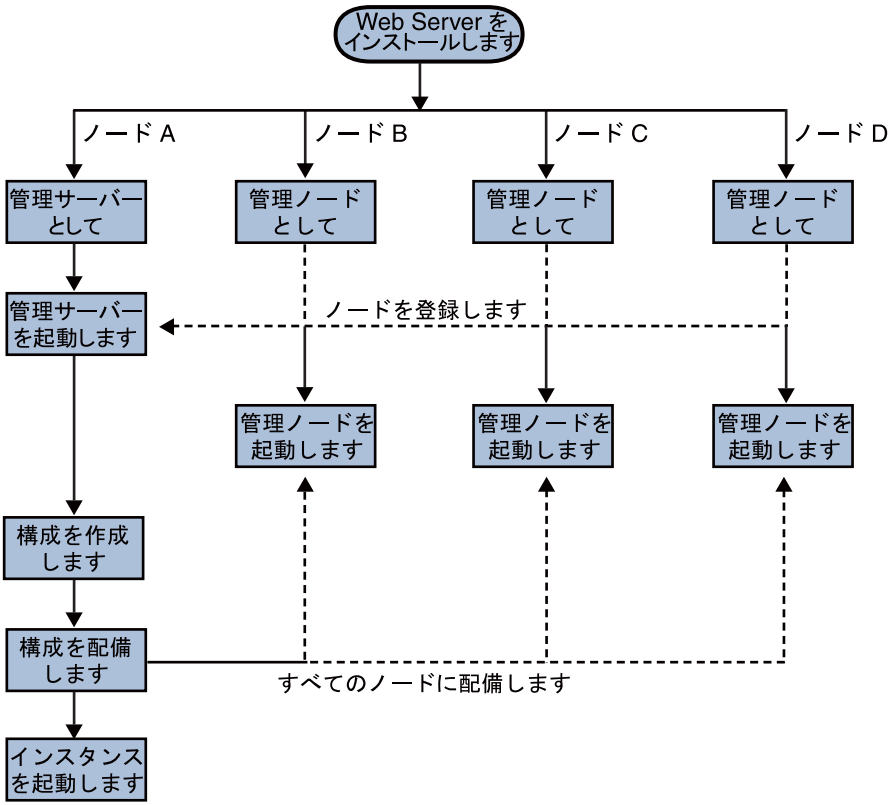


図 4–3 クラスタの設定を示すフローチャート

1. ノードの 1 つに、クラスタ内の管理サーバーとして機能する Web Server をインストールします。

2. ほかの 3 つのノードに Web Server をインストールします。Web Server を管理ノードとしてインストールするオプションを選択します。インストール時に、サーバーにノードを登録するオプションを選択します。
3. 管理サーバーが通信時に SSL ポートを使用していることを確認します。なぜなら、セキュリティー保護されたモードでしか、管理ノードをサーバーに登録できないからです。
4. 管理サーバーと管理ノードがインストールされているすべてのノードのシステム日時が、同一であることを確認します。サーバーに関連付けられる証明書は、管理サーバーがインストールされたノードのシステム日時に基づいて作成されます。管理ノードのシステム日時が管理サーバーの日時よりも遅れていると、管理サーバーの証明書がまだ有効になっていないため、登録が失敗します。必然的に、その証明書は、有効期限が切れたあとも有効であるとみなされる可能性があります。
5. *install_dir*`/admin-server/bin/` ディレクトリから管理サーバーを起動します。

   install_dir`/admin-server/bin>./startserv`
6. 管理ノードから `wadm` コマンド行ツールを起動します。`wadm` コマンド行ツールは、*install_dir*`/bin` ディレクトリに格納されています。

   install_dir`/bin>./wadm`
7. 各管理ノードを管理サーバーに登録します`register-node` コマンドを使って各ノードをサーバーに登録します。

   次に例を示します。

   ```
   ./wadm register-node -user=admin --host=abc.sfbay.sun.com --port=8989
   ```

   説明:

   `abc.sfbay.sun.com`　ノードの登録先となる管理サーバーのホスト名です。

   `port`　管理サーバーの SSL ポート番号です。
8. 管理パスワードの入力を求められます。管理サーバーの管理パスワードを入力します。

   管理サーバーが管理ノードのサーバー証明書を信頼し、管理ノードが管理サーバーから提示されたクライアント証明書を信頼することによって、管理サーバーの相互認証が実現されます。ある管理ノードを登録するときに、管理サーバーによってその管理ノードのサーバー証明書が生成され、続いてその証明書がその管理ノードにダウンロードおよびインストールされます。管理ノードには、サーバー証明書の発行者の情報もインストールされます。

注 – 登録は SSL 経由でしか行えません。

ノードの登録方法については、『Sun Java System Web Server 7.0 Installation and Migration Guide』の「Registering the Administration Node From the Command-Line」を参照してください。

9. *install_dir*/admin-server/bin/ ディレクトリの startserv コマンドを使って、すべての管理ノードを起動します。
10. 管理コンソールまたはCLI を使って、新しい構成を管理サーバー内に作成します。

    新しい構成の構成名、HTTP リスナーポート、サーバー名など、構成情報を入力します。
11. すべてのノード上で構成のインスタンスを作成します。
12. すべてのノード上でインスタンスを起動します。

注 – Web Server では、クラスタを柔軟に拡張または縮小できるようになっています。クラスタに対するインスタンスの追加や削除は、任意のタイミングで行えます。

負荷分散のための逆プロキシの構成

Web Server 7.0 には、高度な組み込みロードバランサである逆プロキシが用意されています。逆プロキシは、サーバーファーム内の Web Server に対するゲートウェイになります。逆プロキシを構成すると、同じように構成された複数の Web サーバーに要求が転送されるようになります。

Web Server 7.0 で逆プロキシを有効にするには、次の手順を使用します。

1. 逆プロキシ構成用として使用するノードに、Web Server をインストールします。
2. 構成を作成します。例: rp。
3. 管理コンソールで「構成」>「仮想サーバー」>「コンテンツ処理」>「逆プロキシ」タブを選択します。「新規」ボタンをクリックします。
4. 逆プロキシの URI を入力するとともに、クラスタ内のすべてのマシンのサーバー URL をコンマで区切って入力します。

    サーバー URL の入力形式は、*hostname*:*portnumber* です。
5. 変更結果を保存します。
6. 変更済みの構成を配備することで、構成への変更内容を適用します。
7. この変更済みの構成のすべてのインスタンスを起動します。

これで、HTTP 要求を負荷分散するための逆プロキシの構成が完了しました。

セッションレプリケーション

セッションレプリケーションとは、あるセッション内に格納されたデータを異なるインスタンス間でレプリケートするために使用される機構のことです。ただし、レプリケート対象インスタンスは、同じクラスタの一部になっている必要があります。クラスタ環境でセッションレプリケーションが有効になると、セッションデータの全体がレプリケート対象インスタンスにコピーされます。ただし、セッションレプリケーションの処理では、セッション内の直列化可能でない属性やインスタンスに固有のあらゆるデータはコピーされません。

セッションレプリケーションと負荷分散を組み合わせると、Web アプリケーションのフェイルオーバー機能を効率的に実現できます。

セッションレプリケーションとフェイルオーバーの動作

この節では、セッションレプリケーションの動作について詳しく説明します。

Web Server は Web 要求の終了時に、サーバー構成ファイル `server.xml` 内に格納されたセッションレプリケーション構成に基づいてセッションデータをコピーする必要があるかどうかを判定します。

ここで、4 つのインスタンスが 1 つのクラスタを形成しており、管理サーバー上でセッションレプリケーションが有効になっている、というユースケースを考えます。

4 つのインスタンス (A、B、C、および D) が 4 つのノード上で稼働している Web Server クラスタにおけるセッションレプリケーションの手順は、次のとおりです。

- インスタンス A は D のバックアップであり、B は A のバックアップであり、C は B のバックアップであり、D は C のバックアップです。これで完全なバックアップリングが形成されます。
- クラスタ内の各インスタンスは、クラスタ内のすべてのインスタンスの静的リストと、アクティブなバックアップインスタンスを追跡します。
- 構成によっては、各要求の終了時に、セッションデータがバックアップインスタンスに同期的に送信されます。

Web Server クラスタ環境におけるフェイルオーバーの手順は、次のとおりです。

- インスタンス A で障害が発生すると、ロードバランサによってインスタンス A 宛のすべての受信 Web 要求がクラスタ内の残りのインスタンスへリダイレクトされるとともに、バックアップリングが次のように構成し直されます。

- Dは、自身のバックアップであるAが停止したことを検出し、順序付きリスト上でAの次にあるインスタンスを、自身の新しいバックアップインスタンスとして選択します。
- この場合はBが選択され、Dは、Bとのバックアップ接続を新たに確立します。Bはこの時点で、2つのバックアップを保持しています。Aの読み取り専用バックアップと、Dのアクティブなバックアップです。
- これでバックアップリングが完結し、BがCにバックアップされ、CがDにバックアップされ、DがBにバックアップされるようになります。
- 障害の発生したインスタンスAが再度利用可能になると、Aは、自身の指定されたバックアップインスタンスであるBに再結合メッセージを送信することでバックアップリングに再度参加し、Bとのバックアップ接続を確立します。
- Dが、Aから正常なpingリターンを受信するかAから何らかのメッセージを受信することによって、Aがオンラインになったことを検出すると、
- Dは、Aとのバックアップ接続を確立し、Bとのバックアップ接続を終了します。

Web Server 7.0のセッションレプリケーションでサポートされていない機能は、次のとおりです。

- 2つ以上のインスタンスで同時に発生した障害を復旧すること。
- 2つの障害間の間隔が、復活したインスタンスが完全に復旧するのに必要な時間よりも長くなければいけません。
- 複数のインスタンスへのセッションのバックアップ。通常の動作では、どのセッションのコピーも2つしか存在しません。主要セッションとバックアップセッションです。
- セッションの持続性: セッションは、フェイルオーバーの目的で別のインスタンスのメモリー内にバックアップされるだけです
- Web Serverがセッションレプリケーションをサポートするのは、Java Webアプリケーションに対してだけです。CGIやPHPなど、Java以外のアプリケーションを使用する場合には、セッションデータをレプリケートできません。

セッションレプリケーションの有効化

クラスタのセッションレプリケーションの有効化は、管理コンソール、CLIのいずれかを使って行えます。セッションレプリケーションを有効化する前に、ブラウザのCookieが有効になっていることを確認してください。

`server.xml`ファイルには、セッションレプリケーションに関する情報が含まれています。セッションレプリケーションが有効化されたサンプル`server.xml`ファイルを、次に示します。

```
<cluster>
        <local-host>hostA</local-host>
            <instance>
          <host>hostB</host>
      </instance>
      <instance>
          <host>hostC</host>
      </instance>
       <instance>
          <host>hostD</host>
      </instance>
      <instance>
           <host>hostA</host>
    <session-replication/>
</cluster>
```

次の各要素のデフォルト値を使用する場合、それらの要素のエントリは server.xml 構成ファイル内に含まれません。

Port number (デフォルトは 1099)
Protocol (デフォルトは jrmp)
Encrypted (デフォルトは false)
Getattribute Triggers Replication (デフォルトは true)
Replica Discovery MaxHopss (デフォルトは –1)
Startup Discovery Timeout (デフォルトは ?
)
Cookie Name (デフォルトは CLUSTERSESSIONLOCATOR)

これらのセッションレプリケーションプロパティーの詳細については、『Sun Java System Web Server 7.0 Administrator's Configuration File Reference』を参照してください。

セッションレプリケーションのための Web アプリケーションの構成

サーバーがセッションをレプリケートできるようにするには、Web アプリケーションでもセッションレプリケーションを有効にする必要があります。

1. Web アプリケーションのセッションレプリケーションを有効にするには、*<web-application>*/WEB-INF ディレクトリに格納された sun-web.xml 構成ファイルを変更します。

   sun-web.xml で次の変更を行う必要があります。

要素 <session-manager/> を <session-manager persistence-type="replicated"> に変更します。

セッションレプリケーションが有効化されたサンプル sun-web.xml ファイルを、次に示します。

```
<sun-web-app>
        <session-config>
     <session-manager persistence-type="replicated">
     </session-manager>
  </session-config>
</sun-web-app>
```

2. sun-web.xml ファイルを変更したあと、Web アプリケーションをビルドし直すかアプリケーションの JAR ファイルを作成し直すことで、Web アプリケーションアーカイブ (WAR ファイル) を作成します。
3. すべてのインスタンスを再起動することで、その Web アプリケーションがすべてのインスタンスで利用可能になるようにします。
4. Web アプリケーションには、クラスタ内のすべてのインスタンスからアクセスできます。Web アプリケーションにアクセスするには、ブラウザで次のように入力します。

   http://*webserver-name*/*webapplication-name*/

注 – すべてのノードからアクセス可能なディレクトリが、配備用のアプリケーションを格納するための最良の方法です。ただし、このディレクトリには、管理サーバーからアクセスできる必要はありません。Web アプリケーションのサイズが 1M バイトを超える場合には、ディレクトリベースの配備を行うことをお勧めします。

検索コレクションを作成する場合、すべてのノードからアクセス可能な共通ディレクトリ内に検索コレクションが存在していることを確認してください。

クラスタの監視

管理サーバーは、クラスタ内のすべてのインスタンスを監視できます。Web Server の監視機能は、実行時のコンポーネントやプロセスの状態に関する情報を提供します。その用途には次のものが考えられます。

- パフォーマンス障害の特定
- 最適なパフォーマンスを実現するためのシステム調整
- 容量計画の支援
- 障害の予測
- 障害発生時の根本原因の解析

Solaris ゾーン

Solaris ゾーンは、Solaris 10 のアプリケーションおよびリソース管理機能です。ゾーン環境は通常、プロセス管理、メモリー、ネットワーク構成ファイル、ファイルシステム、パッケージレジストリ、ユーザーアカウント、共有ライブラリなどのリソースから構成されますが、場合によってはインストールされたアプリケーションも含みます。ゾーンは、1 つの Solaris インスタンス内に仮想化されたオペレーティングシステム環境を作成する手段を提供します。このため、システム上のほかのアクティビティーから遮断して 1 つ以上のプロセスを実行することが可能となります。ゾーンは、物理デバイスパス、ネットワークインタフェース名、ネットワークルーティングテーブルなど、アプリケーションの配備先マシンの物理属性からアプリケーションを分離するための抽象層も提供します。この遮断により、ユーザー ID などの資格情報が何であれ、あるゾーン内で実行されているプロセスが、ほかのゾーン内で実行されているプロセスを監視したり影響を与えたりできなくなります。

ゾーンは、システムの残りの部分に影響を与えたりそうした部分と相互作用することなしに 1 つ以上のアプリケーションを実行することのできる「サンドボックス」です。

Solaris ゾーンの詳細については、『Solaris のシステム管理 (Solaris コンテナ: 資源管理と Solaris ゾーン)』(`http://docs.sun.com/app/docs/doc/819-0385`) を参照してください。

◆ ◆ ◆ 第 5 章

仮想サーバーの使用

仮想サーバーの概要

仮想サーバーを使用すると、企業や個人のドメイン名、IP アドレス、および一部のサーバー監視機能を、インストール済みの単一サーバーを使って提供できます。それはユーザーにとって、自分の Web サーバーを所有しているのとほぼ同じです。ただし、そのハードウェアや Web サーバーの保守を提供するのは皆さんです。

すべての仮想サーバーで HTTP リスナーが指定されます。サーバーは新しい要求を受信すると、その要求をどの仮想サーバーに送信するかを、構成された HTTP リスナーに基づいて判定します。

ユースケース

Sun Java System Web Server のサーバーインスタンスは、セキュリティー保護された HTTP リスナー、セキュリティー保護されていない HTTP リスナーのどちらでも、任意の数だけ持つことができます。IP アドレスベースの仮想サーバー、URL ホストベースの仮想サーバーのいずれも作成できます。

どの仮想サーバーも、独自の ACL リスト、独自の `mime.types` ファイル、および独自の一連の Java Web アプリケーションを持つことができます (ただし、必ずしも持つ必要はない)。

このような設計になっているため、ユーザーは、さまざまなアプリケーション向けに極めて柔軟にサーバーを構成できます。次の例では、Sun Java System Web Server で利用可能な構成のいくつかについて説明します。

デフォルトの構成

Sun Java System Web Server を新規インストールした場合、サーバーインスタンスが 1 つ存在しています。このサーバーインスタンスには HTTP リスナーが 1 つだけ存在しますが、このリスナーは、使用するコンピュータが構成されている任意の IP アドレスを、ポート 80 (またはインストール時に選択されたポート) 上で待機します。

ローカルネットワーク内の何らかの機構により、使用するコンピュータが構成されているアドレスのそれぞれについて、名前とアドレスのマッピングが確立されます。次の例では、コンピュータは 2 つのネットワークインタフェースを持っています。アドレス 127.0.0.1 のループバックインタフェース (ネットワークカードが存在しなくても存在しているインタフェース) と、アドレス 10.0.0.1 の Ethernet インタフェースです。

名前 example.com は DNS 経由で 10.0.0.1 にマップされています。待機ソケットは、そのマシンが構成されている任意のアドレスをポート 80 上で待機するように構成されています (「ANY:80」または「0.0.0.0:80」)。

この構成の場合、次の各場所に接続するとサーバーにアクセスでき、仮想サーバー VS1 のサービスを受けることができます。

- `http://127.0.0.1/` (`example.com` 上で起動)
- `http://localhost/` (`example.com` 上で起動)
- `http://example.com/`
- `http://10.0.0.1/`

従来型の Web サーバー用途では、この構成を使用します。別の仮想サーバーや HTTP リスナーを追加する必要はありません。

サーバーのセキュリティー保護

90 ページの「サーバーの SSL 設定」を参照してください。

イントラネットホスティング

Sun Java System Web Server のより複雑な構成の 1 つとして、サーバーがイントラネット配備向けにいくつかの仮想サーバーをホスティングする、というものがあります。たとえば、内部用のサイトが 3 つあり、それらのサイト上で従業員は、ほかのユーザーの電話番号の検索、構内の地図の閲覧、および情報サービス部門への要求の状態追跡を行えるものとします。これまで (この例の場合)、これらのサイトは、名前 `phone.example.com`、`maps.example.com`、および `is.example.com` にマップされた異なる 3 つのコンピュータ上にホスティングされていました。

ハードウェアと管理のオーバーヘッドを最小限に抑えたければ、この 3 つのサイトすべてを、マシン example.com 上に存在する 1 つの Web サーバーに統合化することができます。この設定の実現方法は 2 つあります。URL ホストベースの仮想サーバーを使用する方法と、個別の HTTP リスナーを使用する方法です。どちらにも固有の利点と欠点があります。

URL ホストベースの仮想サーバーを使用したイントラネットホスティング

URL ホストベースの仮想サーバーは、設定は容易ですが、次の欠点があります。

- この構成で SSL をサポートするには、ワイルドカード証明書を使用する標準的でない設定が必要となる。
- URL ホストベースの仮想サーバーは、旧バージョンの HTTP クライアントでは正しく動作しない

また、アドレスごとに HTTP リスナーが 1 つずつ用意される、IP アドレスベースの構成を設定することも可能です。

個別の **HTTP** リスナーを使用したイントラネットホスティング

IP アドレスベースの仮想サーバーに対する利点は、次のとおりです。

- HTTP/1.1 Host ヘッダーをサポートしない古いクライアントでも正常に動作する。
- SSL サポートの提供方法が単純である。

欠点は次のとおりです。

- ホストコンピュータの構成 (実際のネットワークインタフェースや仮想ネットワークインタフェースの構成) を変更する必要がある
- 数千の仮想サーバーを含む構成に拡大することができない

どちらの構成の場合にも、3 つの名前について、名前とアドレスのマッピングを設定する必要があります。IP アドレスベースの構成では、それぞれの名前が異なるアドレスにマップされます。ホストマシンがこれらすべてのアドレスへの接続を受信するように、ホストマシンを設定する必要があります。URL ホストベースの構成では、すべての名前を同じアドレスに、具体的にはマシンがもともと持っていたアドレスに、マップできます。

複数の HTTP リスナーを含む構成を使用すると、パフォーマンスがわずかに向上することがあります。なぜなら、受信した要求の対象となるアドレスをサーバーが調べる必要がないからです。ただし、複数の HTTP リスナーを使用する場合には、追加のアクセプタスレッドが必要となるため、メモリーとスケジューリングに関する別のオーバーヘッドも発生します。

大規模ホスティング

大規模ホスティングとは、低トラフィックの仮想サーバーを多数有効にするような構成のことです。たとえば、低トラフィックの個人のホームページを多数ホスティングする ISP は、このカテゴリに該当します。

仮想サーバーは通常、URL ホストベースとなります。たとえば、ある構成では静的コンテンツのみを有効にし、別の構成では静的コンテンツと CGI を有効にする、といったことが可能になります。

仮想サーバーの管理

仮想サーバーの追加

▼ 仮想サーバーを追加する

始める前に
- 仮想サーバーを作成する必要のある構成の作成/特定が完了しているべきです。
- HTTP リスナーの作成/特定が完了しているべきです。
- 新しい仮想サーバーに対する 1 つ以上のホストの特定が完了しているべきです。

1. 仮想サーバーを追加する必要のある構成を選択します。「構成」タブに表示される構成のリストから、構成を選択できます。
2. 「仮想サーバー」タブ>「新規」ボタンをクリックします。
3. 仮想サーバーの構成手順を案内するポップアップウィザードページが表示されます。このウィザードページから次のタスクを実行します。
   - 新しい仮想サーバーの情報を入力します。
     a. 新しい仮想サーバーを識別する名前を入力します。この名前には英数字はもちろん、ピリオド (.)、ダッシュ (-)、および下線 (_) 文字も含めることができます。
     b. (省略可能) 新しい仮想サーバーに追加するホストのリストを入力します。
     c. (省略可能) 仮想サーバーのドキュメントルートを入力します。
   - 新しく構成される仮想サーバーの HTTP リスナーを選択します。既存の HTTP リスナーを選択することも、新しい HTTP リスナーを作成することもできます。

4 この時点でウィザードの概要ページが表示されます。構成を変更するには、「戻る」をクリックして前のページに戻ります。新しい仮想サーバーの構成処理を完了させるには、「完了」をクリックします。

5 この時点で「結果」ページが表示されます。エラーが表示された場合には、ウィザードの前のページに戻って仮想サーバーを構成し直します。

注 – **CLI** の使用

CLI 経由で仮想サーバーを追加するには、次のコマンドを実行します。

```
wadm> create-virtual-server --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --document-root=../docs config1_vs_1
```

CLI リファレンスの `create-virtual-server`(1) を参照してください。

仮想サーバーの構成

- 75 ページの「仮想サーバーを構成する」

仮想サーバーの一般設定を構成するには、次のタスクを実行します。

▼ 仮想サーバーを構成する

1 構成を選択します。

構成のリストから構成を選択します。利用可能な構成のリストを取得するには、「構成」タブをクリックします。

2 仮想サーバーを選択します。

仮想サーバーのリストから仮想サーバーを選択します。選択された構成で利用可能な仮想サーバーを取得するには、「仮想サーバー」タブをクリックします。

3 「一般」タブをクリックします。次の設定を行います。

- 有効— 実行時に仮想サーバーを有効にするかどうか。
- ドキュメントルート— 仮想サーバーのデータの格納先となる、仮想サーバーのドキュメントルートのパス。これには、展開された Web アプリケーションディレクトリやログファイルが含まれます。
- ホスト — 複数の URL ホストをコンマで区切ったものを入力できます。

仮想サーバーの複製

仮想サーバーを複製するには、次のタスクを実行します。

▼ 仮想サーバーを複製する

1. 構成を選択します。
   構成のリストから構成を選択します。利用可能な構成のリストを取得するには、「構成」タブをクリックします。
2. 仮想サーバーを選択します。
   仮想サーバーのリストから仮想サーバーを選択します。選択された構成で利用可能な仮想サーバーを取得するには、「仮想サーバー」タブをクリックします。
3. 「コピー」ボタンをクリックします。
   新しい仮想サーバーの名前を入力します。

注 – **CLI** の使用

CLI 経由で仮想サーバーを複製するには、次のコマンドを実行します。

```
wadm> copy-virtual-server --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --vs=config1_vs_1 copiedVs
```

copiedVS は新しい仮想サーバーの名前です。

CLI リファレンスの `copy-virtual-server`(1) を参照してください。

HTTP リスナーの設定

サーバーは、HTTP リスナー経由で HTTP 要求を受け入れたあと、構成された仮想サーバーにその要求を転送します。このページでは、HTTP リスナーを追加および構成できます。

HTTP リスナーは、ポート番号と IP アドレスの一意の組み合わせを持つ必要があります。IPv4 アドレス、IPv6 アドレスのいずれかを使用できます。IP アドレスを「*」に設定すると、そのポート上のすべての IP アドレスに対して待機する HTTP リスナーが作成されます。

HTTP リスナーの作成

受信 HTTP 要求を処理するための、仮想サーバーの新しい HTTP リスナーを作成できます。それには次の手順を実行します。

1. 「構成」タブの下の「仮想サーバー」タブをクリックします。

2. 「**HTTP** リスナー」サブタブを クリックして、構成された HTTP リスナーのリストを表示します。
3. 「新規」ボタンをクリックします。新しい HTTP リスナーを作成するための ウィザードページがポップアップします。

ウィザードページで次の情報を入力します。

- 名前 — 新しい HTTP リスナーの名前。
- ポート — HTTP リスナーがバインドし、受信 HTTP 要求に対して待機するポート。
- **IP** アドレス — 有効な IPv4 または IPv6 アドレス。「*」は、HTTP リスナーが、構成されたポートのすべての IP アドレスに対して待機することを意味します。
- サーバー名 — サーバー名を入力します。例: *sales.mycomp.com*
- デフォルト仮想サーバー — ドロップダウンリストから仮想サーバーを選択します。このアクションにより、この新しい HTTP リスナーが選択された仮想サーバーに関連付けられます。
- 説明 (省略可能) — HTTP リスナーの簡単な説明を入力します。

注 – **CLI** の使用

CLI 経由で HTTP リスナーを作成するには、次のコマンドを実行します。

```
wadm> create-http-listener --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --listener-port=18003 --config=config1 --server-name=config1.com
--default-virtual-server-name=config1_vs_1 config1_ls_1
```

CLI リファレンスの `create-http-listener`(1) を参照してください。

HTTP リスナーの設定

既存の HTTP リスナーの設定を編集できます。それには次のタスクを実行します。

1. 既存の HTTP リスナーの設定を編集するには、あるサーバー構成の「仮想サーバー」タブをクリックします。
2. 「**HTTP** リスナー」サブタブを クリックして、構成された HTTP リスナーのリストを表示します。
3. 「リスナー名」表列の下で、設定を編集する必要のある HTTP リスナーをクリックします。

HTTP リスナーの一般設定とセキュリティー関連設定の両方を編集できます。

HTTP リスナーのパラメータの変更

HTTP リスナーの基本設定と詳細設定を編集するには、「一般」タブをクリックします。次のオプションを構成します。

- 名前 — 新しい HTTP リスナーの名前
- ポート — HTTP リスナーがバインドし、受信 HTTP 要求に対して待機するポート。
- IP アドレス — 有効な IPv4 または IPv6 アドレス。「*」は、HTTP リスナーが、構成されたポートのすべての IP アドレスに対して待機することを意味します。
- サーバー名 — サーバー名を入力します。例: *sales.mycomp.com*

HTTP リスナーの詳細設定を編集するには、「詳細」セクションの「詳細設定」オプションを選択します。次のオプションを構成します。

- アクセプタスレッド — このリスナーが受信した接続を受け付けるための専用スレッドの数。指定可能な値は 1 から 128 までです。
- プロトコルファミリ — リスナーが使用するプロトコル。この値は変更しないでください。デフォルトは HTTP です。
- 待機キューサイズ — オペレーティングシステムの待機キューバックログの最大サイズ。
- 受信バッファーサイズ — オペレーティングシステムのソケット受信バッファーのサイズ (バイト)。
- 送信バッファーサイズ — オペレーティングシステムのソケット送信バッファーのサイズ (バイト)。
- ブロック入出力 — HTTP リスナーのソケットをブロックモードで動作させるかどうかを決定します。デフォルトでは無効です。

◆ ◆ ◆ 第 6 章

6 証明書と鍵

この章では、証明書と鍵の認証を使用した、Sun Java System Web Server のセキュリティー保護について説明します。ここでは、データを保護し、侵入者をブロックし、正規ユーザーのアクセスを許可するように設計された、各種セキュリティー機能を有効にする方法について説明します。

この章を読む前に、公開鍵暗号方式の基本概念に慣れ親しんでおくべきです。具体的には、暗号化と復号化、公開鍵と非公開鍵、デジタル証明書、暗号化プロトコルなどの概念です。

証明書を使用した認証

認証とは、識別情報を確認するためのプロセスのことです。ネットワーク通信の文脈では、認証とは一方が他方を確信をもって特定することです。証明書は、認証をサポートする方法の 1 つです。

証明書は、個人、企業、またはその他の実体の名前を指定するデジタルデータで構成されており、証明書に含まれている公開鍵がその実体に属していることを証明します。クライアントとサーバーの両方が証明書を持つことができます。

証明書は、認証局つまり CA によって発行され、デジタル署名されます。CA は、インターネット経由で証明書を販売する企業の場合も、企業でイントラネットやエク

ストラネットの証明書の発行を担当する部門の場合もあります。他人の身元の証明者として十分に信頼できる CA はどれであるかを決定します。

証明書には、その証明書によって識別される実体の公開鍵と名前のほかに、有効期限、その証明書を発行した CA の名前、およびその発行元 CA の「デジタル署名」も含まれています。

注 – 暗号化機能を有効にするには、事前にサーバー証明書をインストールしておく必要があります。

サーバー認証

サーバー認証とは、クライアントがサーバーを確信をもって特定することであり、具体的には、ある特定のネットワークアドレスに存在しているサーバーの責任元と考えられる組織を確信をもって特定することです。

クライアントの認証

クライアント認証とは、サーバーがクライアントを確信をもって特定することであり、具体的には、クライアントソフトウェアを使用していると考えられる個人を確信をもって特定することです。クライアントは複数の証明書を持つことができますが、これはちょうど、個人が異なる ID をいくつか持つことができるのに似ています。

証明書のキーの種類

Sun Java System Web Server 7.0 では RSA キーに加え、ECC (Elliptic Curve Cryptography) のサポートが追加されました。

ECC は昨今、魅力的な公開鍵暗号系として台頭しつつあります。というのも、ECC は、RSA などの従来の暗号系と同等のセキュリティーをより小さなキーサイズで実現できるため、計算の高速化や電力消費の低下、メモリーや帯域幅の節約が図れるからです。ECC (Elliptic Curve Cryptography) は米国政府によって支持されました。

今回、証明書要求や自己署名付き証明書を RSA キー、ECC キーのどちらを使って生成するかを選択できるようになりました。

RSA キーの場合、さまざまなキーサイズを指定できます (キーサイズが大きいほど暗号化も強固になる。デフォルトのキーサイズは 1024)。ECC キーの場合、鍵ペア生成時に使用する曲線を選択すべきです。ANSI X9.62、NIST、SECG など、さまざまな組織によって多くの曲線に名前が付けられていますが、Sun Java System Web Server 7.0 は現在規定されているすべての曲線をサポートしています。

自己署名付き証明書を使用しないでCAの証明書を要求する場合には、必ず選択したCAにまず連絡をとり、ECCの使用法に関する最新情報を入手してください。自身のユースケースに推奨のECC曲線があるかどうかを尋ねてください。曲線の選択について、CAからも組織の内部ポリシーからも指針が得られない場合のために、推奨する曲線を次にいくつか列挙します。ただし、ECCは台頭しつつあるテクノロジであるため、特定のユースケースに推奨の曲線が、このマニュアルの執筆時点のものから変更される可能性もあります。

サポートされているECC曲線のいくつかを、次に列挙します。

prime256v1
secp256r1
nistp256
secp256k1
secp384r1
nistp384
secp521r1
nistp521
sect163k1
nistk163
sect163r1
sect163r2
nistb163
sect193r1
sect193r2
sect233k1
nistk233k1
nistk233
sect233r1
nistb233
sect239k1
sect283k1
nistk283
sect283r1
nistb283
sect409k1
nistk409
sect571k1
nistk571
sect571r1
nistb571
secp160k1
secp160r1
secp160r2

secp192k1
secp192r1
nistp192
secp224k1
secp224r1
nistp224
prime192v1

自己署名付き証明書の作成

証明書に CA の署名が必要ない場合や、CA が証明書の署名処理を行っている最中に新しい SSL 実装をテストしたい場合、自己署名付き証明書を生成できます。この一時的な証明書を使用すると、「認証局が不明であり、信頼できない」という意味のエラーがクライアントのブラウザで発生します。

CLI 経由で自己署名付き証明書を作成するには、次のコマンドを実行します。

```
wadm> create-selfsigned-cert --user=admin --port=8989 --password-file=admin.pwd
--config=config1 --token=internal --org-unit=org1 --locality=XYZ --state=DEF
--validity=10 --org=sun --country=ABC --server-name=serverhost --nickname=cert1
```

CLI リファレンスの `create-selfsigned-cert`(1) を参照してください。

証明書の管理

証明書の要求

証明書は、個人、企業、またはその他の実体の名前を指定するデジタルデータで構成されており、証明書に含まれている公開鍵がその個人に属していることを証明します。SSL 対応のサーバーは証明書を持つ必要があり、クライアントはオプションで証明書を持つことができます。

証明書は、認証局つまり CA によって発行され、デジタル署名されます。CA は、インターネット経由で証明書を販売する企業の場合も、企業でイントラネットやエクストラネットの証明書の発行を担当する部門の場合もあります。他人の身元の証明者として十分に信頼できる CA はどれであるかを決定します。

証明書の要求を作成し、それを認証局 (CA) に提出できます。会社に独自の内部 CA がある場合には、そこから発行される証明書を要求します。商用 CA からの証明書購入を予定している場合には、CA を選定し、CA が必要とする情報の特定のフォーマットを入手してください。サーバーの自己署名付き証明書を作成することもできます。自己署名付き証明書は、インターネットとの接点を持つ配備には適しませんが、開発やテストには非常に役立ちます。なぜなら、CA を通さずにテストサーバーを設定できるからです。

前述したように、証明書には実体 (この場合は Web サーバー) の公開鍵が含まれています。公開鍵は、ある特定のアルゴリズム (このアルゴリズムのタイプも証明書内に復号化される) に基づいて生成されます。次の節では、Web Server でキー用にサポートされているアルゴリズムタイプの背景について説明します。

▼ 証明書を要求する

1 「サーバー証明書」タブ>「要求」ボタンをクリックします。

2 構成を選択します

証明書をインストールする必要がある構成を、構成のリストから選択します。

3 トークンを選択します

キーを含むトークン (暗号化デバイス) を選択します。Sun Java System Web Server 7.0 によって維持されるローカルキーデータベース内にキーが格納されている場合には、「Internal」を選択します。スマートカードなど、外部のデバイスやエンジンにキーが格納されている場合には、その外部トークンの名前をドロップダウンリストボックスから選択します。選択されたトークンのパスワードを入力します。

4 詳細を入力します

要求処理を開始する前に、CA が必要とする情報が明確になっているか確認してください。サーバー証明書を民間の CA、内部 CA のいずれに要求する場合でも、次の情報を提供する必要があります。

- サーバー名: *www.sun.com* など、DNS 検索で使用される完全修飾ホスト名でなければいけません。これは、サイトへの接続時にブラウザが使用する URL に含まれるホスト名になります。これら 2 つの名前が一致しない場合、証明書の名前とサイトの名前が一致しないことがクライアントに通知され、証明書の信頼性が疑われることになります。

  内部 CA が発行する証明書を要求する場合には、このフィールドにワイルドカードや正規表現も入力できます。ほとんどのベンダーは、共通名にワイルドカードや正規表現が入力された証明書の要求を承認しません。
- 組織: 企業、教育機関、提携先などの正式な法律上の名前です。CA の多くは、ここに入力された情報を、営業許可証の複写などの法的文書で確認することを要求します。

- 組織単位: 省略可能なフィールドであり、企業内の組織を記述します。また、*Inc.* や *Corp.* を省略するなど、より非公式な企業名をここに記載することもできます。
- 地域: 省略可能なフィールドであり、通常は組織の市区町村を記述します。
- 都道府県: 省略可能なフィールドです。
- 国名: 国名を表す 2 文字の略号 (ISO 形式) です。米国の国コードは US です。

これらすべての情報が、識別名 (DN) と呼ばれる一連の属性と値のペアとして結合され、その DN が証明書のサブジェクトとなります。

5 証明書オプションを選択します

キー情報の入力を求められます。キーの種類としては、RSA または ECC を選択できます。キーの種類が RSA の場合、キーサイズは 1024、2048、または 4098 になります。キーの種類が ECC の場合は曲線も選択する必要があります。新しい鍵ペアの生成には時間がかかることに注意してください。キーの長さが長いほど、ウィザードでキーを生成する時間も長くなります。

注意 – キーの種類は、あとで署名要求を送る CA がサポートしている種類を必ず選択するようにしてください。

6 証明書タイプを選択します

証明書の認証局 (CSA) を選択します (自己署名付き、CA 署名付きのいずれか)。自己署名付き証明書を選択する場合、証明書に HTTP リスナーを関連付けることもできます。このアクションをあとで実行することもできます。

7 要求を生成します

CA 署名付き証明書の場合、生成された証明書要求が ASCII 形式で利用可能になります。自己署名付き証明書の場合、証明書は直接インストールされます。タイプが自己署名付きの場合、セキュア要求を処理するためのニックネーム、有効期間 (月)、および HTTP リスナー名の値を入力します。

8 結果を表示します

このページには、選択したオプションの概要が表示されます。要求の生成を完了させるには、「完了」をクリックします。

注 – **CLI** の使用

CLI 経由で証明書を要求するには、次のコマンドを実行します。

```
wadm> create-cert-request --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --server-name=servername.org
--org=sun --country=ABC --state=DEF --locality=XYZ --token=internal
```

CLI リファレンスの `create-cert-request`(1) を参照してください。

証明書のインストール

CA から証明書を取得したら、管理コンソールを使ってその証明書をある構成用としてインストールできます。

▼ 証明書をインストールする

1 「サーバー証明書」タブ>「インストール」ボタンをクリックします。

2 構成を選択します
証明書をインストールする必要がある構成を、構成のリストから選択します。

3 トークンを選択します
キーを含むトークン (暗号化デバイス) を選択します。Sun Java System Web Server 7.0 によって維持されるローカルキーデータベース内にキーが格納されている場合には、「Internal」を選択します。スマートカードなど、外部のデバイスやエンジンにキーが格納されている場合には、その外部トークンの名前をドロップダウンリストボックスから選択します。選択されたトークンのパスワードを入力します。

4 証明書のデータを入力します
提供されたテキスト領域に、証明書のテキストをペーストします。テキストをコピーしてペーストするときは、必ず、開始と終了のハイフンを含む「Begin Certificate」および「End Certificate」というヘッダーを含めてください。「参照」ボタンをクリックし、.DER ファイルを手動で選択することもできます。

5 証明書の詳細を入力します
証明書で使用されるニックネームを入力します。選択可能なリストから、セキュア要求を処理するための HTTP リスナーを選択します。

6 結果を表示します
このページには、選択したオプションの概要が表示されます。インストール処理を完了させるには、「完了」をクリックします。

注 – **CLI** の使用

CLI 経由で証明書をインストールするには、次のコマンドを実行します。

```
wadm> install-cert --user=admin --port=8989  --password-file=admin.pwd
--config=config1 --token=internal --cert-type=server --nickname=cert1 cert.req
```

ここで、cert.req には証明書データが含まれています。

CLI リファレンスの `install-cert`(1) を参照してください。

証明書の更新

次の手順に従えば、既存の証明書を更新できます。

▼ 証明書を更新する

1. 「サーバー証明書」タブ>「証明書名」>「更新」ボタンをクリックします。
2. トークン情報を入力します
   必要に応じてトークンのパスワードを入力します。それ以外の場合は、「次へ」をクリックして処理を続行します。
3. 証明書の詳細を更新します
   証明書の詳細を確認し、有効期間を月数で入力します。
4. キー情報を更新します
   キーの種類としては、RSA または ECC を選択できます。キーの種類が RSA の場合、キーサイズは 1024、2048、または 4098 になります。キーの種類が ECC の場合は曲線も選択する必要があります。新しい鍵ペアの生成には時間がかかることに注意してください。
5. 概要を表示します
   このページには、選択したオプションの概要が表示されます。更新処理を完了させるには、「完了」をクリックします。

証明書の削除

構成を削除するには、次のタスクを実行します。

▼ 証明書を削除する

1. 「サーバー証明書」タブをクリックします。
2. 証明書を選択します
   証明書のリストから証明書名を選択します。
3. 証明書を削除します
   選択した証明書を削除するには、「削除」ボタンをクリックします。

注 – **CLI** の使用

CLI 経由で証明書を削除するには、次のコマンドを実行します。

```
wadm> delete-cert --user=admin --port=8989  --password-file=admin.pwd
--token=internal --config=config1 cert1
```

CLI リファレンスの `delete-cert`(1) を参照してください。

管理サーバー証明書の更新

管理サーバー証明書を更新するには、`renew-admin-certs` CLI コマンドを実行します。このコマンドは、ニックネーム `Admin-CA-Cert`、 `Admin-Server-Cert`、および `Admin-Client-Cert` を含む管理サーバー証明書を更新する場合に使用します。このコマンドは、現在稼働している、更新後の証明書を使ってアクセス可能なノードの更新も行います。

このコマンドの実行後、新しい証明書が有効になるように、管理サーバーと各ノードを再起動することをお勧めします。証明書の更新中にオフラインになっていた (停止中であったか、ネットワークの問題でアクセス不能になっていた) ノードについては、登録し直す必要があります。管理サーバー証明書を更新するには、次のコマンドを実行します。

```
wadm> renew-admin-certs --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --validity=120
```

CLI リファレンスの `renew-admin-certs`(1) を参照してください。

証明書失効リスト (CRL) の管理

証明書失効リスト (CRL) は、クライアントユーザー、サーバーユーザーのいずれかが今後信頼すべきでない証明書と鍵をすべて通知します。証明書の有効期限が切れる前にユーザーが事務所を変更したり、その組織を離れるような場合など、証明書のデータが変わった場合には、その証明書は無効になり、そのデータが CRL に表示されます。CRL は CA によって生成され、定期的に更新されます。

▼ CRL をインストールする

CA から取得された CRL をインストールするには、次の手順を実行します。

1. **CA から CRL をファイルとして取得します。**
2. 管理コンソールの構成のページに移動します。
3. 「証明書」>「認証局」タブをクリックします。
4. 「**CRL** をインストール」ボタンをクリックします。
5. 関連ファイルへのフルパス名を入力します。
6. ［了解］をクリックします。

注 – CRL がすでにデータベース内に存在している場合には、「証明書失効リストを置換」ページが表示されます。

---

7. 「配備」をクリックしないと変更が有効にならない可能性があります。

注 – **CLI** の使用

CLI 経由で CRL をインストールするには、次のコマンドを実行します。

```
wadm> install-crl --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 data/install-crl/ServerSign.crl
```

CLI リファレンスの `install-crl(1)` を参照してください。

▼ CRL を削除する

1. 管理コンソールの構成のページに移動します。
2. 「証明書」>「認証局」タブをクリックします。
3. **CRL** エントリを選択し、「削除」をクリックします。
4. 「配備」をクリックしないと変更が有効にならない可能性があります。

注 – **CLI** の使用

CLI 経由で CRL を削除するには、次のコマンドを実行します。

```
wadm> delete-crl --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 issuer
```

CLI リファレンスの `delete-crl`(1) を参照してください。

内部トークンのパスワードの設定

内部 PKCS11 トークンのパスワードを設定するには、次のタスクを実行します。

▼ トークンのパスワードを設定する

1. 管理コンソールの構成のページに移動します。
2. 「証明書」>「**PKCS11** トークン」タブをクリックします。
3. **PKCS11** トークンの名前 **(**デフォルトは **internal)** をクリックします。
4. 「トークンの状態」チェックボックスを選択します。
5. パスワードの情報を入力します。
6. インスタンス起動時にパスワードの入力を求められないようにするには、チェックボックス「インスタンス起動時に新規パスワードの入力を求めない」を選択します。「了解」をクリックします。
7. パスワードが構成に保存されます。パスワードを削除するには、上記手順を実行し、「パスワードの設定解除」オプションを選択します。

注 – **CLI** の使用

CLI 経由で内部 PKCS11 トークンのパスワードを設定するには、次のコマンドを実行します。

```
wadm> set-token-pin --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --token=internal
```

CLI リファレンスの `set-token-pin`(1) を参照してください。

サーバーの SSL 設定

要求を生成し、その要求を CA に送信するには、コマンド `create-cert-request` を使用します。その後、CA から証明書を受け取ったら、その証明書を `install-cert` コマンドでインストールする必要があります。移行対象の鍵と証明書が Java キーストア内に存在している場合には、コマンド `migrate-jks-keycert` を使用します。開発/テストサーバーを設定するもっとも簡単な方法は、コマンド `create-selfsigned-cert` を使って自己署名付き証明書を生成することです。

```
wadm> create-selfsigned-cert --server-name=hostname --nickname=MyServerCert
--token=internal
```

その他のオプションや例については、マニュアルページを確認してください。

証明書がインストールされたら、SSL を有効にするリスナーが、何らかのポート上で必要になります。

```
wadm> create-http-listener --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --listener-port=18003 --config=config1 --server-name=config1.com
--default-virtual-server-name=config1_vs_1 config1_ls_1
```

次に、このリスナーの SSL を有効にし、このリスナーに証明書のニックネームを関連付けます。

```
wadm> set-ssl-prop --http-listener=http-listener-ssl enabled=true
wadm> set-ssl-prop --http-listener=http-listener-ssl server-cert-nickname=MyServerCert
```

この設定が完了したら、構成を配備し、インスタンスを起動します。

```
wadm> deploy-config config_name
wadm> start-instance --config config_name hostname
```

構成の SSL 暗号化方式の有効化

構成の SSL 暗号化方式を有効にするには、次のコマンドを実行します。

```
wadm> enable-ciphers --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --http-listener=http-listener-1
--cipher-type=ssl2 SSL_RC4_128_WITH_MD5
```

CLI リファレンスの `enable-ciphers(1)` を参照してください。

HTTP リスナーのセキュリティーの有効化

注 – HTTP リスナーのセキュリティーを有効にできるのは、利用可能なインストール済み証明書が存在する場合だけです。

証明書を入手したら、その証明書を HTTP リスナーに関連付け、サーバーをセキュリティー保護することができます。

暗号化とは、情報を対象とした受信者以外の人が読めないような内容にするための、変換プロセスのことです。復号化とは、暗号化された情報を判読可能な状態に戻すための、変換プロセスのことです。Sun Java System Web Server には、SSL および TLS プロトコルのサポートが含まれています。

暗号化方式とは、暗号化または復号化に使用される暗号化アルゴリズム (数学関数) のことです。SSL と TLS プロトコルには、多数の暗号化方式のセットが含まれています。安全度は、暗号化方式によって異なります。一般に、暗号化方式が使用するビット数が多くなるほど、データの復号化が難しくなります。

任意の双方向の暗号化処理では、両者が同じ暗号化方式を使用する必要があります。多数の暗号化方式が利用可能であるため、使用頻度のもっとも高い暗号化方式をサーバーで有効にする必要があります。

セキュリティー保護された接続時には、クライアントとサーバーは、通信に、その双方が持てる最も強力な暗号化方式を使用します。SSL2、 SSL3、および TLS プロトコルから暗号化方式を選択できます。

注 – SSL バージョン 2.0 のあとでセキュリティーとパフォーマンスが改善されたため、SSL 3 を使用できないクライアントを使用するのでないかぎり、SSL 2 を使用すべきではありません。クライアント証明書は、SSL 2 暗号化方式との組み合わせで正しく動作することが保証されていません。

暗号化プロセスだけでは、サーバーの機密情報のセキュリティー保護には十分ではありません。実際に暗号化結果を生成したり、すでに暗号化された情報を復号化するためには、暗号化方式と一緒に鍵を使用する必要があります。暗号化処理では、2つの鍵を使ってこの結果が実現されます。これが、公開鍵と非公開鍵です。公開鍵を使用して暗号化された情報は、対応する非公開鍵を使用した場合にのみ復号化できます。公開鍵は証明書の一部として発行されます。保護されるのは、関連付けられた非公開鍵だけです。

Sun Java System Web Server は暗号化通信用として、Secure Sockets Layer (SSL) および Transport Layer Security (TLS) プロトコルをサポートします。SSL と TLS はアプリケーションに依存しないため、それらの上に上位レベルのプロトコル層を透過的に配置できます。

SSL および TLS プロトコルは、サーバーとクライアントでお互いを認証するために使用される多くの暗号化方式のサポート、証明書の送信、およびセッション鍵の確立を行います。クライアントとサーバーは、サポートしているプロトコルや、暗号化の強度についての会社の方針および暗号化されたソフトウェアの輸出に対する行政上の制約条件などの要因に基づいて、別の暗号化方式セットをサポートすることができます。他の機能の中でも特に、SSL と TLS ハンドシェイクプロトコルは、どの暗号化方式のセットを通信に使用するかをサーバーとクライアントが交渉する方法を決定します。

HTTP リスナーのセキュリティー設定を編集するには、「構成」>「**HTTP** リスナー」>「**SSL**」タブをクリックします。次の表に、このページで設定可能なプロパティーの一覧を示します。

表 6–1 HTTP リスナーのセキュリティープロパティー

プロパティー	説明
名前	HTTP リスナーの名前。
セキュリティー	選択された HTTP リスナーのセキュリティーを有効化/無効化します。
証明書	利用可能な証明書からサーバー証明書を選択します。このアクションを実行するには、RSA または ECC 証明書がインストール済みになっているべきです。
クライアント認証	クライアント認証を必須、省略可能のいずれにするかを指定します。クライアント認証を無効にするには、「無効」オプションを選択します。
認証タイムアウト	このタイムアウト後に、クライアント認証のハンドシェークが失敗します。[0.001–3600]。デフォルト値は 60 秒です。
認証データの最大値	バッファーに格納する認証データの最大量。[0–2147.0483647.0]。デフォルト値は 104857.06 です。

表 6–1 HTTP リスナーのセキュリティープロパティー (続き)

プロパティー	説明
SSL バージョン **2**/バージョン **3**	SSL バージョン 2、SSL バージョン 3 を有効化/無効化します。
TLS	TLS を有効化/無効化します。「バージョンロールバックを検出」はデフォルトで有効になっています。「有効」に設定すると、人が介在するバージョンロールバック攻撃を検出するようにサーバーが設定されます。TLS の仕様を間違って実装している一部のクライアントとの相互運用性を確保するためには、これを無効にしなければいけない可能性があります。
SSL3/SSL2/TLS 暗号化方式	Web サーバーのセキュリティーを保護するには、SSL を有効にすることをお勧めします。SSL 2.0、SSL 3.0、および TLS 暗号化プロトコルを有効にし、さまざまな暗号化方式群を選択することができます。管理サーバーの待機ソケットで、SSL および TLS を有効にできます。 デフォルト設定では、使用頻度のもっとも高い暗号化方式が有効になります。特定の暗号化方式群を使用したくない特別な理由がないかぎり、それらすべてを有効にすべきです。

◆ ◆ ◆ 第 7 章

7 サーバーへのアクセスの制御

Web サーバー上に存在するリソースは、認証、承認、アクセス制御など、いくつかのセキュリティーサービスおよび機構を使って保護できます。この章では、Sun Java System Web Server 7.0 へのアクセスを制御するためにサポートされているいくつかの機構について説明します。

アクセス制御とは

認証とは、識別情報を確認するためのプロセスのことです。承認とは、ある制限されたリソースへのアクセスをあるアイデンティティーに許可することを意味しますが、それらの制限はアクセス制御機構によって実現されます。認証と承認は、さまざまなセキュリティーモデル (Web アプリケーションセキュリティー、htaccess、認証レルムなど) やサービスを使って実現できます。

アクセス制御を使えば次のことを決定できます。

- 管理サーバーにアクセスできるユーザー
- それらのユーザーがアクセスできるアプリケーション
- Web サイト上のファイルまたはディレクトリにアクセスできるユーザー

アクセス制御は、サーバーの全体、サーバーの一部、Web サイト上のファイルやディレクトリのいずれに対しても行えます。アクセスを許可または拒否するには、アクセス制御エントリ (ACE) と呼ばれる規則の階層を作成します。作成された ACE のコレクションは、アクセス制御リスト (ACL) と呼ばれます。

サーバーにはデフォルトで、複数の ACL を含む ACL ファイルが 1 つあります。Sun Java System Web Server は、受信要求に使用する仮想サーバーを決定したあと、その仮想サーバー用に構成された ACL の有無をチェックします。現在の要求に当てはまる ACL が見つかった場合、サーバーはそれらの ACL の ACE を評価することで、そのアクセスを許可すべきかどうかを決定します。

アクセスの許可または拒否は、次の情報に基づいて行われます。

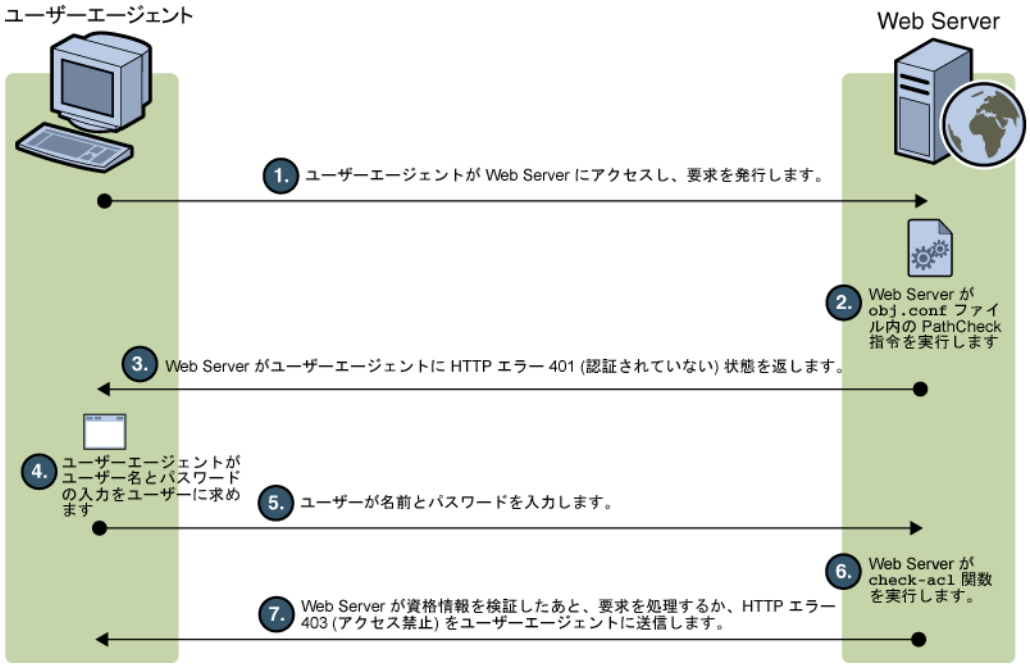
- 要求を行っているユーザー (ユーザー - グループ)
- 要求の送信元 (ホスト - IP)
- 要求が発生した日時 (時刻など)
- 使用されている接続のタイプ (SSL)

アクセス制御のしくみ

サーバーは、あるページの要求を受信すると、ACL ファイル内の規則を使ってそのアクセスを許可すべきかどうかを決定します。規則は、要求を送信しているコンピュータのホスト名や IP アドレスを参照できます。また、規則が LDAP ディレクトリに格納されているユーザーやグループを参照するように設定することもできます。

注 – 一致する ACL が複数ある場合、サーバーは最後に一致した ACL 文を使用します。`uri` ACL が最後に一致する文であるため、`default` ACL は無視されます。

上図は、Web Server 7.0 でのアクセス制御の処理手順を示したものです。ユーザー



エージェント (クライアント) が Web Server にアクセスします。Web Server が `obj.conf` ファイル内の `PathCheck` 指令を実行します。Web Server が HTTP 401 (認証されない) をクライアントに返します。クライアントがユーザーに認証を要求します。クライアントがブラウザの場合、ログインダイアログボックスが表示されます。ユーザーがログイン情報を入力します。Web Server が内部 `check-acl` 関数を実行します。Web Server がユーザーの資格情報を検証し、要求を処理します。

ユーザー - グループに対するアクセス制御の設定

特定のユーザーまたはグループに対して、Web サーバーへのアクセスを制限することができます。ユーザー - グループのアクセス制御を設定した場合、サーバーにアクセスする前に、ユーザーはユーザー名とパスワードの入力を求められます。サーバーは、クライアント証明書内の情報をディレクトリサーバーのエントリと比較します。

管理サーバーでは、基本認証だけを使用します。管理サーバーでクライアント認証を要求する場合、ACL ファイルを手動で編集し、認証方法を SSL に変更する必要があります。

ユーザー - グループ認証は、Web Server がユーザーグループデータベース内のエントリを読み取ることによって実行されます。ディレクトリサービスがアクセス制御の実装に使用する情報は、次のソースのいずれかから入手できます。

- 内部フラットファイルタイプのデータベース
- 外部 LDAP データベース

サーバーは、外部 LDAP ベースのディレクトリサービスを使用するとき、サーバーインスタンスに対して次のタイプのユーザーグループ認証方法をサポートします。

- デフォルト
- 基本
- SSL
- ダイジェスト
- その他

サーバーが内部ファイルベースのディレクトリサービスを使用するときに、サーバーインスタンスに対してサポートされるユーザー - グループ認証方法には、次のものがあります。

- デフォルト
- 基本
- ダイジェスト

ユーザー - グループ認証では、ユーザーがサーバーにアクセスしたり、Web サイト上のファイルやディレクトリにアクセスしたりするには、まず自身を認証する必要があります。認証を行う場合、ユーザーは、ユーザー名とパスワードを入力したりクライアント証明書を使用したりすることで、自身の身元を確認します。クライアント証明書が必要となるのは、SSL 通信を行う場合だけです。

デフォルト認証

デフォルト認証は、推奨される方法です。「標準」設定では、`server.xml` ファイル内のデフォルトの方法が使用されます。ただし、`server.xml` に設定が含まれていない場合は「基本」が使用されます。「デフォルト」を選択した場合、ACL 規則によって ACL ファイル内のメソッドが指定されることはありません。「標準」を選択すれば、`obj.conf` ファイル内で 1 行編集することですべての ACL の方法を簡単に変更できます。

基本認証

基本認証では、ユーザー名とパスワードを入力しないと、Web サーバーまたは Web サイトにアクセスできません。これがデフォルトの設定です。一連のユーザーとグループを作成し、それらを Sun Java System Directory Server などの LDAP データベース内またはファイル内に格納する必要があります。ディレクトリサーバーは、Web サーバーとは異なるサーバールートにインストールされたものか、リモートマシンにインストールされたものを使用する必要があります。

ユーザーが、管理サーバー内や Web サイト上のユーザー - グループ認証が必要なリソースにアクセスしようとすると、ユーザー名とパスワードの入力を求めるダイアログボックスが Web ブラウザによって表示されます。サーバーで暗号化が設定されているかどうかに応じて、サーバーはこの情報を暗号化された状態、または暗号化されていない状態で受信します。

注 – SSL 暗号化なしで基本認証を使用する場合、暗号化されていないユーザー名とパスワードがネットワークを経由して送信されます。ネットワークパケットは傍受される可能性があり、ユーザー名とパスワードが不正に知られてしまう可能性があります。基本認証は、SSL 暗号化またはホスト - IP 認証、あるいはその両方と組み合わせるのがもっとも効果的です。ダイジェスト認証を使えばこの問題を回避できます。

SSL 認証

サーバーは、次の 2 つの方法で、セキュリティー証明書付きのユーザーの識別情報を確認できます。

- クライアント証明書の情報を識別情報の証明として使用する
- LDAP ディレクトリで発行されたクライアント証明書を確認する (追加)

クライアントの認証で証明書の情報を使用するようにサーバーを設定した場合、サーバーは次の処理を実行します。

- まず、証明書が信頼できる CA からのものであるか確認します。そうでない場合、認証は失敗し、トランザクションが終了します。
- 証明書が信頼できる認証局 (CA) からのものである場合、`certmap.conf` ファイルを使ってその証明書をユーザーのエントリにマップします。
- 証明書が正しくマップされている場合は、そのユーザーに対して指定されている ACL 規則を確認します。証明書が正しくマップされている場合でも、ACL 規則によってユーザーのアクセスが拒否される可能性もあります。

特定のリソースへのアクセスを制御するためにクライアント認証を要求することは、サーバーへの接続のすべてに対してクライアント認証を要求することとは異なります。すべての接続に対してクライアント認証を要求するようにサーバーを設定した場合、クライアントは信頼できる CA によって発行された有効な証明書のみを提示する必要があります。SSL の方法を使ってユーザーとグループを認証するようにサーバーのアクセス制御を設定した場合、クライアントで次のことが必要になります。

- 信頼できる CA から発行された有効な証明書を提供する
- 証明書は LDAP 内の有効なユーザーにマッピングされている
- アクセス制御リストで、適切に評価される

アクセス制御でクライアント認証を要求する場合、Web サーバーの SSL 暗号化方式を有効にする必要があります。

SSL で認証されるリソースにアクセスするには、Web サーバーが信頼している CA から、クライアント証明書が発行されている必要があります。ブラウザのクライアント証明書とディレクトリサーバーのクライアント証明書を比較するように Web サーバーの `certmap.conf` ファイルが設定されている場合、クライアント証明書はディレクトリサーバーで発行されている必要があります。ただし、証明書から選択した情報とディレクトリサーバーのエントリだけを比較するように、`certmap.conf` ファイルを設定することもできます。たとえば、ブラウザ証明書のユーザー ID および電子メールアドレスとディレクトリサーバーのエントリだけを比較するように、`certmap.conf` ファイルを設定することができます。

注 – SSL 認証方法の場合にのみ、`certmap.conf` ファイルを変更する必要があります。なぜなら、この方法では、LDAP ディレクトリに基づいて証明書がチェックされるからです。サーバーへのすべての接続に対してクライアント認証を要求する場合は、その必要はありません。クライアント証明書を使用することを選択する場合、`magnus.conf` の `AcceptTimeout` 指令の値を増やすべきです。

ダイジェスト認証

サーバーは、LDAP ベースまたはファイルベースのいずれかのディレクトリサービスを使用して、ダイジェスト認証を行うように設定できます。

ダイジェスト認証を使えば、ユーザー名とパスワードを平文として送信することなしに、ユーザー名とパスワードに基づく認証を行えます。ブラウザは MD5 アルゴリズムを使用して、ユーザーのパスワードと Web Server によって提供される情報の一部を使用するダイジェスト値を作成します。

サーバーが LDAP ベースのディレクトリサービスを使用してダイジェスト認証を行うとき、このダイジェスト値は、ダイジェスト認証プラグインを使用してサーバー側で計算され、クライアントによって提示されたダイジェスト値と比較されます。ダイジェスト値が一致した場合、ユーザーは認証されます。これが機能するには、ディレクトリサーバーが平文形式のユーザーのパスワードにアクセスできる必要があります。Sun Java System Directory Server にはリバーシブルパスワードプラグインが組み込まれています。このプラグインは、対称暗号化アルゴリズムを使用し、暗号化された形式にしてデータを格納し、あとで元の形式に復号化することができます。データへの鍵を持っているのは Directory Server だけです。

LDAP ベースのダイジェスト認証の場合、リバーシブルパスワードプラグインと、サーバーに組み込まれているダイジェスト認証専用のプラグインを有効にする必要があります。ダイジェスト認証を処理するように Web サーバーを構成するには、`dbswitch.conf` 内のデータベース定義の `digestauth` プロパティーを設定します。

ACL 方式を指定しない場合、認証が必要であればダイジェスト認証または基本認証のいずれかが使用され、認証が必要ない場合は基本認証が使用されます。これが推奨される方法です。

表 7–1 ダイジェスト認証要求の生成

ACL 方式	認証データベースでダイジェスト認証がサポートされている	認証データベースでダイジェスト認証がサポートされていない
「標準」 指定されていない	ダイジェストと基本	基本
「基本」	基本	基本
「ダイジェスト」	ダイジェスト	エラー

`method = digest` を含む ACL を処理する場合、サーバーは次のようにして認証を試みます。

- Authorization 要求ヘッダーをチェックします。見つからない場合は、ダイジェスト要求に対して 401 の応答が生成され、プロセスが停止します。
- 認証のタイプを確認します。認証のタイプがダイジェストの場合、サーバーは次のことを実行します。
  - ノンスをチェックします。このサーバーによって生成された、有効で新しい nonce がない場合は、401 の応答が生成されてプロセスが停止します。期限切れの場合は、`stale=true` で 401 応答が生成され、処理が停止します。

    ノンスの有効期間を構成するには、`magnus.conf` ファイル内のパラメータ `DigestStaleTimeout` の値を変更します。このファイルは `server_root/https-server_name/config/` にあります。この値を設定するには、次の行を `magnus.conf` に追加します。

    ```
    DigestStaleTimeout seconds
    ```

    ここで *seconds* は、nonce の有効期限 (秒) です。指定されている秒数が経過すると、nonce の有効期限は切れ、ユーザーからの新しい認証が必要になります。
  - レルムをチェックします。これが一致しない場合は、401 応答が生成され、処理が停止します。
  - 認証ディレクトリが LDAP ベースの場合は LDAP ディレクトリにユーザーが存在するかどうかを確認します。認証ディレクトリがファイルベースの場合はファイルデータベースにユーザーが存在するかどうかを確認します。見つからない場合は、401 応答が生成され、処理が停止します。
  - ディレクトリサーバーまたはファイルデータベースから要求ダイジェスト値を取得し、クライアントの要求ダイジェストと一致するかどうかをチェックします。一致しない場合は、401 応答が生成され、処理が停止します。

- Authorization-Info ヘッダーを構築し、サーバーヘッダーに挿入します。

ホスト - IP に対するアクセス制御の設定

管理サーバーまたは Web サイト上のファイルとディレクトリに対して、特定のコンピュータを使用しているクライアントだけが利用できるように、アクセスを制限できます。許可または拒否するコンピュータのホスト名または IP アドレスを指定します。ワイルドカードパターンを使えば、複数のコンピュータやネットワーク全体を指定できます。ホスト - IP 認証経由でのファイルやディレクトリへのアクセスは、ユーザーにはシームレスに感じられます。このため、ユーザーは、ユーザー名やパスワードを入力することなく、すぐにファイルやディレクトリにアクセスできます。

ある特定のコンピュータを複数のユーザーが使用する可能性があるため、ホスト - IP 認証は、ユーザー - グループ認証と組み合わせると、より効果的です。両方の認証方法が使用されている場合、アクセス時にユーザー名とパスワードが必要になります。

ホスト - IP 認証の場合、サーバー上で DNS を構成する必要はありません。ホスト - IP 認証を使用することを選択した場合、ネットワーク内で DNS を稼働させ、その DNS を使用するようにサーバーを構成する必要があります。サーバーの DNS を有効にするには、サーバーマネージャーの「構成」タブの「パフォーマンス」ページを使用します。

DNS を有効にすると、サーバーで DNS 検索が強制的に実行されるため、サーバーのパフォーマンスが低下します。サーバーパフォーマンスへの DNS 検索の影響を減らすには、すべての要求で IP アドレスを解決するのではなく、アクセス制御および CGI でのみ IP アドレスを解決するようにします。それには、`obj.conf` ファイル内の `AddLog fn="flex-log" name="access"` に `iponly=1` を追加します。

```
AddLog fn="flex-log" name="access" iponly=1
```

ACL ユーザーキャッシュの設定

サーバーはデフォルトで、ユーザーおよびグループ認証の結果を ACL ユーザーキャッシュに書き込みます。ACL ユーザーキャッシュの有効期間を制御するには、`magnus.conf` ファイル内の `ACLCacheLifetime` 指令を使用します。キャッシュ内のエントリが参照されるたびにその継続時間が計算され、`ACLCacheLifetime` に基づいてチェックされます。継続時間が `ACLCacheLifetime` に等しいかそれより大きい場合、そのエントリは使用されません。デフォルト値は 120 秒です。この値を 0 (ゼロ) に設定すると、キャッシュが無効になります。この値に大きな値を使用する場合、LDAP エントリを変更するたびにサーバーの起動が必要となる可能性があります。たとえば、この値を 120 秒に設定した場合は、サーバーと LDAP ディレクトリの同期が

2 分間にわたって取られない可能性があります。LDAP ディレクトリが頻繁に変更される可能性が低い場合にだけ、大きな値を設定します。

magnus.conf の ACLUserCacheSize パラメータを使用すると、キャッシュ内に保存できるエントリの最大数を設定できます。このパラメータのデフォルト値は 200 です。新しいエントリはリストの先頭に追加されます。キャッシュが最大サイズに達すると、リストの末尾のエントリは新しいエントリを作成するために再利用されます。

また、magnus.conf に含まれるパラメータである ACLGroupCacheSize を使用して、ユーザーエントリごとにキャッシュできるグループメンバーシップの最大数を設定することもできます。このパラメータのデフォルト値は 4 です。残念ながら、グループ内のユーザーの非メンバーシップはキャッシュに書き込まれないため、要求が発生するたびに数回の LDAP ディレクトリアクセスが発生します。

ACL ファイルの指令の詳細については、『*NSAPI Developer's Guide*』を参照してください。

ACL キャッシュのプロパティーの設定

CLI 経由で ACL キャッシュのプロパティーを設定するには、次のコマンドを実行します。

```
wadm> set-acl-cache-prop --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 property=value
```

CLI リファレンスの set-acl-cache-prop(1) を参照してください。

設定できる有効なプロパティーは、次のとおりです。

- enabled — サーバーがファイルの内容とメタ情報をキャッシュに書き込むかどうかを示します。デフォルト値は true です。
- max-age — ファイルの内容とメタ情報をキャッシュ内に保持する最大時間 (秒)。値の範囲は 0.001 から 3600 までです。
- max-groups-per-user — サーバーがメンバーシップ情報をキャッシュに書き込む対象となるグループの、1 ユーザーあたりの最大数。値の範囲は 1 から 1024 までです。
- max-age — 認証情報をキャッシュ内に保持する最大時間 (秒)。値の範囲は 0.001 から 3600 までです。

アクセス制御の構成

サーバーは、ローカルに格納されたアクセス制御リスト (ACL) を使用することで、認証と承認をサポートします。ACL は、あるユーザーがあるリソースに対してどのようなアクセス権限を持つかを記述します。たとえば、ある ACL のエントリを使って、John という名前のユーザーに、特定のフォルダ `misc` に対する `read` アクセス権を許可することができます。

この節では、Web サイト上のファイルやディレクトリへのアクセスを制限するための手順について説明します。すべてのサーバーに対してグローバルアクセス制御規則を設定することも、特定のサーバーに対して個別に設定することもできます。たとえば、人事部門であれば、すべての認証済みユーザーが自分の給与データを参照できるが、そのデータを更新する機能へのアクセスは人事部門の給与担当者のみに制限するような ACL を作成できます。

サーバーがサポートするコア ACL では、3 種類の認証が使えます。基本、SSL、およびダイジェストです。

アクセス制御の設定を編集するには、次のタスクを実行します。

1. 「構成」タブをクリックし、構成を選択します。
2. 「セキュリティー」サブタブ>「アクセス制御」サブタブをクリックします。
3. 「**ACL** を追加」ボタンをクリックして新しい ACL を追加するか、既存の ACL をクリックして設定を編集します。

アクセス制御リスト **(ACL)** の追加

次の節では、新しい ACL を構成に追加する手順について説明します。

1. 「構成」タブをクリックし、構成を選択します。
2. 「アクセス制御」サブタブ>「アクセス制御リスト **(ACL)**」サブタブをクリックします。
3. 「新規」ボタンをクリックして新しい ACL を追加します。

下記のパラメータを設定します。

表 7–2 ACL パラメータ

パラメータ	説明
リソース	名前/URI/パス。アクセス制限を設定するリソースの種類を選択し、その値を指定します。URI リソースの例: — 「/sales」。パスリソースの例: — 「/usr/sun/server4/docs/cgi-bin/*」。

表 7–2 ACL パラメータ (続き)

パラメータ	説明
認証データベース	「認証データベース」では、サーバーがユーザーの認証に使用するデータベースを選択できます。 デフォルトは **keyfile** です
認証方法	1. 基本 — HTTP メソッドを使用してクライアントから認証情報を取得します。ユーザー名とパスワードがネットワーク上で暗号化されるのは、サーバーで SSL が有効になっている場合だけです。 2. **SSL** — クライアント証明書を使用してユーザーの認証を行います。このメソッドを使用するには、サーバーの SSL を有効にする必要があります。暗号化が有効になっていれば、基本と SSL の方法を組み合わせることができます。 3. ダイジェスト — ユーザー名とパスワードを平文として送信することなしにユーザー名とパスワードに基づく認証をブラウザが行うための手段を提供する認証機構を使用します。ブラウザは *MD5* アルゴリズムを使用して、ユーザーのパスワードと Web Server によって提供される情報の一部を使用するダイジェスト値を作成します。「ダイジェスト」を使用するには、背後の auth-db もダイジェストをサポートしている必要があります。これは、ダイジェスト認証プラグインがインストールされている場合にのみ、digestfile を使用したファイル auth-db、LDAP auth-db のいずれかを意味します 4. その他 — アクセス制御 API を使って作成されたカスタム方法を使用します。
認証確認テキスト	「認証確認テキスト」オプションでは、認証ダイアログボックス内に表示されるメッセージテキストを入力できます。このテキストを使用して、ユーザーが入力する必要のある項目について説明することができます。ブラウザによっては、最初の 40 文字程度しか表示されません。 Web ブラウザは通常、ユーザー名とパスワードをキャッシュし、それらをプロンプトのテキストと関連付けます。ユーザーが、サーバーの同じ確認テキストを持つファイルやディレクトリにアクセスしても、ユーザー名とパスワードを再度入力する必要はありません。特定のファイルやディレクトリでユーザーに再度認証させたい場合は、そのリソースの ACL の確認テキストを変更すればよいだけです。
アクセス拒否時の応答	あるリソースへのアクセスが拒否された場合の応答アクションを指定します。 1. デフォルトメッセージで応答 — サーバーから標準のアクセス拒否メッセージを表示する場合に、このオプションを選択します。 2. URL で応答 — ほかの外部 URL やエラーページに要求を転送する場合に、このオプションを選択します。

注 – **CLI** の使用

CLI 経由で ACL を追加するには、次のコマンドを実行します。

```
wadm> set-acl --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --vs=config1_vs_1 --config=config1
--aclfile=aclfile1
```

CLI リファレンスの `set-acl`(1) を参照してください。

アクセス制御エントリ **(ACE)** の追加

この節では、選択された構成に対して新しいアクセス制御エントリ (ACE) を追加するプロセスについて説明します。

1. 「構成」タブをクリックし、構成を選択します。
2. 「アクセス制御」サブタブ>「アクセス制御リスト **(ACL)**」サブタブをクリックします。
3. 「新規」ボタンをクリックします。
4. 「アクセス制御エントリ (ACE)」の「新規」ボタンをクリックします。

次の ACE パラメータを設定します。

表 **7–3** ACE パラメータ

パラメータ	説明
アクセス権	■ 許可するユーザーまたはシステムが、要求されたリソースにアクセスできることを意味します ■ 許可しないユーザーまたはシステムが、要求されたリソースにアクセスできないことを意味します サーバーはアクセス制御エントリ (Access Control Entry、ACE) のリストを参照して、アクセス権を決定します。

表 7–3 ACE パラメータ *(続き)*

パラメータ	説明
ユーザー	1.すべて (認証なし) — 認証は行われません。すべてのユーザーにアクセスを許可します。 2. 認証データベース中のすべて — 認証データベース内に指定されたすべてのユーザーにアクセスを許可します。 3. 認証データベース中の以下だけ — 認証データベースから選択されたユーザーだけにアクセスを制限します。 認証データベースに対するクエリーを、名、姓、電子メールアドレスなどの共通属性に基づいて実行できます。
グループ	グループの認証が行われる場合、ユーザーがアクセス制御規則で指定されているリソースにアクセスするには、ユーザー名とパスワードを入力する必要があります。 特定のグループにアクセスを制限する場合に、このオプションを使用します。

表 7–3 ACE パラメータ *(続き)*

パラメータ	説明
アクセスを許可するホスト	どのコンピュータから要求が送られたかに基づいて、管理サーバーまたは Web サイトへのアクセスを制限できます。 ■ 「すべて」では、すべてのユーザーとシステムに対してアクセスを許可します ■ 「次の転送元からのみ」では、特定のホスト名または IP アドレスへのアクセスが制限できます 「次の転送元からのみ」オプションを選択する場合は、「ホスト名」フィールドまたは「IP アドレス」フィールドに、ワイルドカードパターンまたはコンマで区切ったリストを入力します。IP アドレスよりホスト名で制限する方が、より柔軟にできます。ユーザーの IP アドレスが変更された場合でも、このリストを更新する必要がありません。ただし、IP アドレスで制限する方が、より確実です。接続したクライアントの DNS 検索が失敗した場合、ホスト名による制限が使用できないためです。 コンピュータのホスト名または IP アドレスに一致するワイルドカードパターン用として使用できるのは、* ワイルドカード記号だけです。たとえば、指定ドメインのすべてのコンピュータに対してアクセスを許可または拒否する場合、 `*.sun.com` のように、特定ドメイン内のすべてのホストと一致するワイルドカードパターンを指定します。管理サーバーにアクセスするスーパーユーザーに対しては、その他のユーザーとは異なるホスト名と IP アドレスを設定することができます。 ホスト名の場合、* は名前の構成要素全体を表している必要があります。つまり、`*.sun.com` は許容されますが、`*users.sun.com` は許容されません。* がホスト名に含まれる場合、それは左端の文字でなければいけません。 たとえば、`*.sun.com` は許容されますが、`users.*.com` は許容されません。IP アドレスの場合、* はアドレスのバイト全体を表している必要があります。たとえば、`198.95.251.*` は許容されますが、`198.95.251.3*` は許容されません。IP アドレスで * を使用する場合、この記号は文字列の一番右に使用する必要があります。たとえば、`198.*` は許容されますが、`198.*.251.30` は許容されません。

表 7–3 ACE パラメータ (続き)

パラメータ	説明
権限	アクセス権は、Web サイトのファイルやディレクトリへのアクセスを制限します。すべてのアクセス権の許可または拒否に加えて、一部のアクセス権の許可または拒否を行うための規則を指定することもできます。たとえば、ユーザーに対してファイルへの読み取り専用アクセスを許可することができます。この設定では、ユーザーは情報を表示することはできますが、ファイルを変更することはできません。 ■ 「すべてのアクセス権限」はデフォルトで、すべてアクセス権を許可または拒否します。 ■ 「次の権限のみ」では、許可または拒否するアクセス権の組み合わせを選択できます。 ■ 「読み込み」を選択すると、ユーザーはファイルを表示できます。これには HTTP メソッド GET、HEAD、POST、および INDEX が含まれます ■ 「書き込み」を選択すると、ユーザーはファイルを変更または削除できます。これには、HTTP メソッド PUT、DELETE、MKDIR、RMDIR、および MOVE.が含まれます。ファイルを削除するには、ユーザーは書き込み権限と削除権限の両方を持っている必要があります ■ 「実行」を選択すると、ユーザーは CGI プログラム、Java アプレット、エージェントなどのサーバー側アプリケーションを実行できます ■ 「削除」を選択すると、書き込み特権も持つユーザーがファイルやディレクトリを削除できます。 ■ 「リスト」を選択すると、ユーザーは、`index.html` ファイルを含んでいないディレクトリ内のファイルのリストにアクセスできます。 ■ 「情報」を選択すると、ユーザーは、`http_head` などの URI に関する情報を受け取れます。
継続	サーバーはアクセス制御エントリ (Access Control Entry、ACE) のリストを参照して、アクセス権を決定します。たとえば、最初の ACE は通常、すべてのユーザーを拒否します。最初の ACE に「継続」が設定されている場合、サーバーはリストの 2 番目の ACE を確認し、一致している場合は、次の ACE を使用します。 「継続」チェックボックスが選択されていない場合は、すべてのユーザーがリソースへのアクセスを拒否されます。サーバーは、一致しない ACE か、一致しているが継続しないように設定されている ACE のどちらかに到達するまでリストを参照し続けます。一致する最後の ACE によって、アクセスが許可されるか拒否されるかが決まります。

.htaccess ファイルの使用

サーバーは、.htaccess 動的構成ファイルをサポートします。.htaccess ファイルを有効にするには、ユーザーインタフェースを使用するか、構成ファイルを手動で変更します。

.htaccess ファイルは、サーバーの標準アクセス制御と組み合わせて使用できます。標準アクセス制御は常に、PathCheck 指令の順序にかかわらず、あらゆる .htaccess アクセス制御の前に適用されます。ユーザー - グループ認証が「基本」の場合、標準アクセス制御と .htaccess アクセス制御の両方を使ってユーザー認証を要求しないでください。サーバーの標準アクセス制御経由で SSL クライアント認証を使用し、さらに .htaccess ファイル経由で HTTP「基本」認証を要求することは可能です。

.htacess ファイルを使用可能にすると、サーバーはリソースを提供する前に、.htaccess ファイルを確認します。サーバーはリソースと同じディレクトリおよびそのディレクトリの親ディレクトリで .htaccess ファイルを検索します。この検索はドキュメントのルートまで続けられます。たとえば「Primary Document Directory」が /sun/server/docs に設定されているときに、クライアントが /sun/server/docs/reports/index.html を要求すると、サーバーは /sun/server/docs/reports/.htaccess および /sun/server/docs/.htaccess を確認します。

サーバーの「Addtional Document Directories」および「CGI Directory」機能で、管理者は代わりのドキュメントルートを定義できます。代わりのドキュメントルートが存在すると、.htaccess ファイルの処理に影響します。たとえば、サーバーの「Primary Document Directory」が /sun/server/docs に設定されていて、CGI プログラムが /sun/server/docs/cgi-bin/program.cgi にあるとします。CGI を「File Type」として有効にした場合、クライアントが CGI プログラムに要求を発行すると、サーバーは /sun/server/docs/.htaccess と /sun/server/docs/cgi-bin/.htaccess の両方の内容を評価します。しかし、「CGI Directory」として /sun/server/docs/cgi-bin を設定すると、サーバーは /sun/server/docs/cgi-bin/.htaccess は検査しますが、/sun/server/docs/.htaccess は検査しません。これは、「CGI Directory」で /sun/server/docs/cgi-bin を指定したことで、代替のドキュメントルートとしてマークされたためです。

サービス拒否攻撃からのサーバーの保護

サービス拒否 (DoS) 攻撃とは、サーバーの悪意のあるユーザーが故意に、正当なユーザーのサービス利用を妨害しようとすることです。そのような攻撃は、次の操作によって実現されます。

- ある特定の Web リソースに対する要求を、サーバーに継続的に送信する

Sun Java System Web Server は、要求の発生頻度が非常に高い場合に、頻繁にアクセスされる URI を監視して要求を拒否することにより、DoS 攻撃を検出できます。

次の各節では、仮想サーバーレベルで DoS 攻撃を防ぐ方法について説明します。

サーバーへの要求の制限

今回、サーバーを調整することでサービス拒否攻撃を防止できるようになりました。具体的には、要求の制限を構成し、仮想サーバーごとの最大接続数を監視します。これらの値のいくつかを構成すると、サーバーのパフォーマンスに影響が及ぶ可能性があります。

サーバーの要求の制限を構成するには、「構成」>「仮想サーバー」>「サーバー設定」>「要求の制限」をクリックします。次の表に記載したパラメータを構成します。

表 7–4 要求の制限の構成

パラメータ	説明
要求の制限	この仮想サーバーで要求の制限を有効化/無効化します。要求の制限オプションはデフォルトで無効になっています。
最大接続数	この仮想サーバーで許可する最大同時接続数。
最大 **RPS**	1 クライアントに許可される 1 秒あたりの最大要求数。
RPS 計算間隔	秒あたりの平均要求数 (RPS) が計算される時間間隔。デフォルト値は 30 秒です。
継続条件	ブロックされた要求タイプが再度サービスを受けられるようになるために満たす必要のある条件を決定します。 サイレンス — サービスが再開するには、拒否された要求の数が後続の期間内でゼロまで落ちる必要があります。 しきい値 — サービスが再開するには、拒否された要求の発生頻度が低下して RPS のしきい値を下回る必要があります。 デフォルト値は「しきい値」です。
エラーコード	ブロックされた要求に使用する HTTP 状態コード。デフォルトのコードは、HTTP 503 (サービス利用不可) です。
属性の監視	省略可能な監視対象の要求属性。

注 – **CLI** の使用

CLI 経由でサーバーへの要求を制限するには、次のコマンドを実行します。

```
wadm> enable-request-limits --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1
```

CLI リファレンスの `enable-request-limits`(1) を参照してください。

▼ 最大接続数を制限する

最大同時接続数を制限できます。処理中の要求が少なくとも指定された数だけ存在している状態で、一致する要求が受信されると、その要求は拒否されます。要求が拒否されるのはその特定の期間内だけである点に注意してください。同時要求数が低下してこの制限を下回るようになると、新しい要求が処理されます。

1. 「構成」タブをクリックします。
2. リストから構成を選択します。
3. 「仮想サーバー」タブで仮想サーバーを選択します。
4. 「サーバー設定」>「要求の制限」をクリックします。
5. 「最大接続数」セクションに値を入力します。

◆ ◆ ◆ 第 8 章

ユーザーとグループの管理

この章では、Sun Java System Web Server にアクセス可能なユーザーとグループを追加、削除、および編集する方法について説明します。

ユーザーとグループに関する情報へのアクセス

管理サーバーを使用して、ユーザーアカウント、グループリスト、アクセス特権 (ACL)、組織単位、およびその他のユーザーやグループに固有の情報に関するアプリケーションデータにアクセスできます。

ユーザーとグループの情報は、テキスト形式のフラットファイルに格納されるか、あるいは Sun Java System Directory Server など、LDAP (Lightweight Directory Access Protocol) をサポートするディレクトリサーバー内に格納されます。LDAP は TCP/IP 上で実行されるオープンなディレクトリアクセスプロトコルであり、グローバルなサイズと数百万件のエントリにも対応可能なスケーラビリティーを備えています。

ディレクトリサービスについて

Sun Java System Directory Server などのディレクトリサーバーを使えば、単一のアプリケーションからすべてのユーザー情報を管理できます。また、サーバーを設定して、簡単にアクセスできる複数のネットワークロケーションからユーザーがディレクトリ情報を引き出せるようにすることもできます。

Web Server 7.0 では、ユーザーやグループを認証および承認するために、異なる 3 つのタイプのディレクトリサービスを構成できるようになっています。ディレクトリサービスが設定されていない場合、作成される新しいディレクトリサービスには、種類に関係なく `default` という値が設定されます。

ディレクトリサービスを作成すると、そのディレクトリサービスの詳細情報に基づいて `server.xml` ファイルが更新されます。

ディレクトリサービスのタイプ

Web Server 7.0 でサポートされているディレクトリサービスのタイプは、次のとおりです。

- **LDAP** — ユーザーとグループの情報を、LDAP ベースのディレクトリサーバー内に格納します。
- 鍵ファイル — 鍵ファイルとは、ハッシュ形式のユーザーパスワード、およびそのユーザーが所属するグループのリストが含まれているテキストファイルです。鍵ファイル内に格納されたユーザーやグループは、`file` レルムによってのみ、認証や承認用として使用されます。これらは、システムのユーザーやグループとは何の関係もありません。

  鍵ファイル形式は、HTTP 基本認証の使用を目的としている場合にだけ使用できます。
- ダイジェストファイル — ユーザーとグループの情報を、暗号化されたユーザー名とパスワードに基づいて格納します。

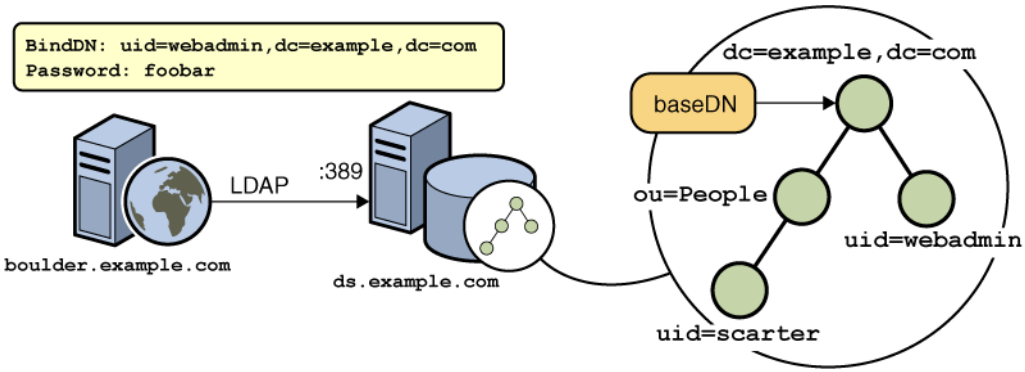
  ダイジェストファイル形式の目的は、HTTP ダイジェスト認証の使用をサポートすることです。ただし、これは基本認証もサポートしており、どちらの認証方法にも使用できるようになっています。

注 – 分散管理を設定する場合、デフォルトのディレクトリサービスは LDAP ベースのディレクトリサービスでなければいけません。

識別名 (DN) の理解

ユーザーとは、企業の社員などのような、LDAP データベース内の個人を意味します。グループとは、同じ属性を共有する複数のユーザーを意味します。組織単位とは企業内の 1 部門のことです。

企業内のユーザーやグループはそれぞれ、識別名 (DN) 属性によって表現されます。DN 属性は、関連するユーザー、グループ、またはオブジェクトを識別する情報が含まれているテキスト文字列です。ユーザーまたはグループのディレクトリエントリを変更する場合は常に、DN を使用します。たとえば、ディレクトリエントリを作成または変更したり、アクセス制御を設定したり、メールやバブリッシングといったアプリケーションのユーザーアカウントを設定したりするたびに、DN 情報を指定する必要があります。



上図にサンプルの DN 表現を示します。次の例は、Sun Microsystems の従業員の典型的な DN を表したものです。

```
uid=doe,e=doe@sun.com,cn=John Doe,o=Sun Microsystems Inc.,c=US
```

この例で、各等号の前にある略語の意味は次のとおりです。

- uid: ユーザー ID
- e: 電子メールアドレス
- cn: ユーザーの共通名
- o: 組織
- c: 国

DN には、さまざまな名前と値のペアを含めることができます。これらは、LDAP をサポートするディレクトリ内における、証明書のサブジェクトとエントリの両方を識別するために使用されます。

LDIF の使用

現在、ディレクトリが存在しない、または既存のディレクトリに新規のサブツリーを追加したい場合、Directory Server の管理サーバー LDIF インポート機能を使用できます。この機能を使用すれば、LDIF を含むファイルを取り扱うことができ、ディレクトリを構築したり、LDIF エントリから新規のサブツリーを構築したりすることが可能です。また、Directory Server の LDIF エクスポート機能を使用して、現在のディレクトリを LDIF へエクスポートすることもできます。この機能は、該当するディレクトリを表す LDIF 形式のファイルを作成します。`ldapmodify` コマンドと適切な LDIF 更新文を組み合わせることで、エントリを追加または編集します。

LDIF を使ってデータベースにエントリを追加するには、まず `LDIF` ファイル内でエントリを定義し、次にその `LDIF` ファイルを Directory Server でインポートします。

認証データベースの操作

「認証データベース」(auth-db とも呼ばれる) は、既知のユーザーのデータベースを表すとともに、そのデータベースに基づいてクライアント要求を認証するための機構を表します。サーバーでは複数の auth-db エントリを同時に構成することができます。これらは同じタイプであってもかまいません。auth-db ユーザーデータベースは ACL 処理モジュールによって使用されます。

サーバーでサポートされる認証データベースは、次のとおりです。

1. **LDAP** — Sun Java System Directory Server などの LDAP ディレクトリサーバー内にユーザーデータが格納されます。
2. ファイル— ディスクファイル内にユーザーデータが格納されます。この auth-db は、ユーザーの集中管理が利用可能でない (あるいは望ましくない) ような開発用途や小規模配備用途に特に便利です。ファイル auth-db は、いくつかの異なるファイル形式をサポートしています。
   a. keyfile — keyfile 形式では、ユーザーのリスト (およびオプションで各ユーザーのグループメンバーシップ) が格納されます。パスワードは単方向の (復元不可能な) ハッシュとして格納されます。これがデフォルトの形式です。
   b. digestfile — digestfile は、keyfile によく似ていますが、さらに HTTP ダイジェスト認証方法もサポートします。
   c. htaccess — これは旧バージョンの形式であり、決して新規インストール時や新しいユーザーを追加する場合に使用すべきではありません。
3. **PAM** — PAM は、Sun Java System Web Server 7.0 でサポートされる新しい auth-db です。PAM auth-db は、Solaris PAM スタックに認証を委任します。これにより、Web Server システム上の既存の Solaris ユーザーを Web Server に対しても認証することが可能となります。

注 – PAM auth-db は Solaris 9 および 10 (またはそれ以上) でしかサポートされておらず、さらに Web Server インスタンスを root として実行する必要があります。

認証データベースの作成

管理コンソール経由で認証データベースを作成するには、「構成」>「構成名」>「アクセス制御」>「認証データベース」>「新規」ボタンをクリックします。フィールドの説明については、管理コンソールのインラインヘルプを確認してください。選択された認証データベースに応じてフィールドが変わります。たとえば、PAM ベースの認証データベースの場合、認証データベース名だけが必須です。

認証データベース作成時の必須オプションを、次に列挙します。

LDAP	■ 認証データベース名 ■ ホスト名 ■ ポート ■ ベース DN
鍵ファイル	■ 認証データベース名 ■ ファイルパス
ダイジェストファイル	■ 認証データベース名 ■ ファイルパス
PAM	■ 認証データベース名

CLI 経由で認証データベースを作成するには、次のコマンドを実行します。

```
wadm> create-authdb --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1
--url=ldap://ldapserver.com:20002/dc=xxx,dc=sun,dc=com LDAP1
```

CLI リファレンスの `create-authdb`(1) を参照してください。

上の例では、認証データベースの URL が指定されています。認証データベースのタイプは、この URL のスキーマで指定します。たとえば、`ldap://ds.example.com/dc=example,dc=com` の場合、LDAP ディレクトリサーバーが認証データベースとして構成されます。

ユーザーとグループの設定

管理サーバーでは、LDAP、ファイルのどちらの auth-db タイプの場合でも、ユーザーアカウント、グループリスト、アクセス特権、組織単位など、ユーザー固有、グループ固有の情報を編集できます。

▼ ユーザーを追加する

1 構成を選択します。
構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「アクセス制御」>「ユーザー」タブをクリックします。

3 「新規」ボタンをクリックします。

4 ユーザー情報を追加します。
ユーザー ID とパスワードを入力します。任意で、ユーザーが属するグループを入力します。ユーザー ID は一意である必要があります。LDAP ベースの認証データベースの場合、管理サーバーは、ユーザー ID が一意であることを確認するために、検索ベース (ベース `DN`) から下方にディレクトリ全体を検索し、そのユーザー ID が使用されているか確認します。ただし、Directory Server の `ldapmodify` コマンド行ユーティリティー (使用可能な場合) を使ってユーザーを作成する場合、ユーザー ID の一意性の確認は行われないことに注意してください。

注 – CLI の使用

CLI 経由でユーザーを作成するには、次のコマンドを実行します。

```
wadm> create-user --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --authdb=KEYFILE1 --full-name=keyfile-config1-u1
keyfile-config1-u1
```

CLI リファレンスの `create-user`(1) を参照してください。

▼ グループを追加する

1 構成を選択します。
構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「アクセス制御」>「グループ」タブをクリックします。

3 「新規」ボタンをクリックします。

4 グループ名を入力します。

5 「グループへのユーザーの追加」セクションで、既存のユーザーを検索してグループに追加します。

注 – keyfile や digestfile などの認証データベースでグループを作成するには、少なくとも 1 つのユーザーを指定する必要があります。

注 – **CLI** の使用

CLI 経由でグループを作成するには、次のコマンドを実行します。

```
wadm> create-group --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --authdb=LDAP1 group1
```

CLI リファレンスの `create-group`(1) を参照してください。

▼ ユーザーを削除する

1 構成を選択します。
構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「アクセス制御」>「ユーザー」タブをクリックします。

3 ユーザーを削除する必要のある認証データベースを選択します。

4 「グループの検索」テキストフィールドにグループ名を入力し、「検索」ボタンをクリックします。

5 「グループ名」列からグループを選択し、「削除」ボタンをクリックします。

注意 – keyfile/digestfile 認証データベースから 1 人以上のユーザーを削除すると、それらのユーザーの削除後に関連する 1 つ以上のグループのメンバーが 1 人もいなくなる場合には、それらのグループも削除されます。これは、メンバーを持たないグループは、keyfile/digestfile 認証データベースでは許可されないからです。

注 – **CLI** の使用

CLI 経由でユーザーを削除するには、次のコマンドを実行します。

```
wadm> delete-user --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config config1 --authdb KEYFILE1 user1
```

CLI リファレンスの `delete-user`(1) を参照してください。

▼ グループを削除する

1. 構成を選択します。
   構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。
2. 「アクセス制御」>「グループ」タブをクリックします。
3. グループを削除する必要のある認証データベースを選択します。
4. 「ユーザーの検索」テキストボックスにユーザー **ID** を入力し、「検索」ボタンをクリックします。
5. 「ユーザー **ID**」列からユーザーを選択し、「削除」ボタンをクリックします。

注 – グループを削除しても、そのグループに属するユーザーは削除されません。ユーザーを手動で削除するか、グループを割り当て直す必要があります。

注 – **CLI** の使用

CLI 経由でグループを削除するには、次のコマンドを実行します。

```
wadm> delete-group --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config config1 --authdb LDAP1 group1
```

CLI リファレンスの `delete-group(1)` を参照してください。

スタティックグループとダイナミックグループ

グループとは、LDAP データベース内の一連のオブジェクトを記述するオブジェクトのことです。Web Server 7.0 のグループは、共通の属性を共有するユーザーから構成されます。たとえば、会社のマーケティング部門で働く多数の従業員がオブジェクトのセットになります。これらの従業員は、Marketing という名前のグループに所属させることができます。

LDAP サービスに対しては、グループのメンバーシップを定義する方法が 2 つあります。スタティックとダイナミックです。スタティックグループは、自身のメンバーオブジェクトを明示的に列挙します。スタティックグループは CN であり、`uniqueMember`、`memberURL`、`memberCertDescription` のいずれかまたはその任意の組み合わせを含みます。スタティックグループでは、メンバーは `CN=<Groupname>` 属性以外の共通属性を共有しません。

ダイナミックグループでは、LDAP URL を使用してグループメンバーにだけ一致する規則セットを定義できます。ダイナミックグループでは、メンバーは `memberURL` フィルタに定義される 1 つの属性または一連の属性を共有します。たとえば、営業部門のすべての従業員を含むグループが必要で、それらの従業員がすでに LDAP データベース内の

「`ou=Sales,o=sun`」の下に存在している場合、次の `memberurl` を含むダイナミックグループを定義します。

```
ldap:///ou=Sales,o=sun??sub?(uid=*)
```

結果として、このグループには、「`ou=Sales,o=sun`」ポイントの下のツリーに `uid` 属性を持つすべてのオブジェクト、つまりすべての Sales メンバーが含まれます。

スタティックおよびダイナミックグループで、`memberCertDescription` を使用している場合は、メンバーは証明書から共通属性を共有できます。ただし、これらは、ACL が SSL メソッドを使用している場合にだけ機能します。

新しいグループの作成が完了すると、そのグループにユーザーつまりメンバーを追加できます。

スタティックグループ

LDAP サービスでは、管理サーバーを使用し、任意のユーザー数の DN に同じグループ属性を指定して、スタティックグループを作成できます。ユーザーが追加または削除されないかぎり、スタティックグループが変わることはありません。

スタティックグループ作成時の指針

管理サーバーのフォームを使って新しいスタティックグループを作成する場合には、次の指針を参考にしてください。

- スタティックグループには、その他のスタティックグループまたはダイナミックグループを含めることができます。
- オプションで、新しいグループの説明を追加することもできます。
- 定義済みの組織単位がディレクトリに存在している場合、「新規グループの追加先」リストを使って新しいグループの配置場所を指定できます。デフォルトの場所は、ディレクトリのルートポイント、つまり最上位のエントリです。

ダイナミックグループ

ダイナミックグループは `groupOfURLs` オブジェクトクラスと 0 個以上の `memberURL` 属性を持ちます。各 `memberURL` 属性は、一連のオブジェクトを記述する LDAP URL です。

LDAP サービスに対しては、任意の属性に基づいてユーザーを自動的にグループ化する場合、または一致する DN を含む特定のグループに ACL を適用する場合に、Web Server でダイナミックグループを作成できます。たとえば、属性 `department=marketing` 持つすべての DN を自動的に含めるグループを作成できます。department=marketing を目的とする検索フィルタを適用すると、この検索によって、属性 `department=marketing` を持つすべての DN を含むグループが返されます。次に、このフィルタに基づいて検索結果からダイナミックグループを定義できます。さらに、結果として生成されるダイナミックグループの ACL を定義できます。

Web Server でのダイナミックグループの実装方法

Web Server は、LDAP サーバースキーマ内でダイナミックグループを `objectclass = groupOfURLs` として実装しています。`groupOfURLS` クラスは `memberURL` 属性を複数持つことができます。各 `memberURL` 属性は、ディレクトリ内の一連のオブジェクトを列挙する単一の LDAP URL で構成されます。グループのメンバーは、これらのセットの和集合です。たとえば、次のグループにはメンバー URL が 1 つだけ含まれています。

```
ldap:///o=mcom.com??sub?(department=marketing)
```

この例は、「o=mcom.com」の下にある、department が「marketing」になっているすべてのオブジェクトから成るセットを示したものです。LDAP URL には、検索ベース DN、スコープ、フィルタは含められますが、ホスト名やポートは含められません。つまり、同じ LDAP サーバー上のオブジェクトだけを参照できます。すべてのスコープがサポートされます。

グループに DN を個別に追加しなくても、すべての DN が自動的に組み込まれます。Sun Java System Web Server は ACL 検証でグループ検索が必要になるたびに LDAP サーバー検索を行うため、グループは動的に変化します。ACL ファイルで使用されるユーザー名とグループ名は、LDAP データベース内のオブジェクトの cn 属性に対応します。

注 – Web Server は、cn (commonName) 属性を ACL のグループ名として使用します。

ACL から LDAP データベースへのマッピングは、dbswitch.conf 構成ファイル (ACL データベース名と実際の LDAP データベース URL を関連付ける) と ACL ファイル (どの ACL でどのデータベースが使用されるかを定義する) の両方に定義されます。たとえば、「staff」というグループのメンバーシップに基本アクセス権を設定する場合、ACL コードは groupOf<anything> というオブジェクトクラスを持ち、CN が「staff」に設定されているオブジェクトを検索します。オブジェクトは、スタティックグループの groupOfUniqueNames のようにメンバーの DN を明示的に列挙するか、groupOfURLs のように LDAP URL を指定することによって、グループのメンバーを定義します。

グループはスタティックかつダイナミックにすることができる

グループオブジェクトは、objectclass = groupOfUniqueMembers と objectclass = groupOfURLs の両方を持つことができます。したがって、「uniqueMember」属性と「memberURL」属性のどちらも有効になります。グループのメンバーシップは、スタティックメンバーとダイナミックメンバーの和集合になります。

サーバーのパフォーマンスに対するダイナミックグループの影響

ダイナミックグループを使用すると、サーバーのパフォーマンスに影響があります。グループメンバーシップをテストするときに、DN がスタティックグループのメンバーではない場合、Web Server はデータベースのベース DN に含まれるすべてのダイナミックグループをチェックします。Web Server はこのタスクを実行するために、各 memberURL が一致するかどうかをチェックします。具体的には、そのベース DN とスコープに基づいてユーザーの DN をチェックしたあと、ベース DN としてのユーザー DN と memberURL のフィルタを使ってベース検索を実行します。この処理では、膨大な数の検索が行われることがあります。

ダイナミックグループ作成時の指針

新規のダイナミックグループを作成するために管理サーバーを使用するときには、次のガイドラインを考慮してください。

- ダイナミックグループにほかのグループを含めることはできません。
- グループの LDAP URL を次の形式で入力します (ホストとポートの情報は不要。これらのパラメータは無視される)。

```
ldap:///<base_dn>?<attributes>?<scope>?<(filter)>
```

必須パラメータについて、次の表で説明します。

表 8–1 ダイナミックグループ:必須パラメータ

パラメータ名	説明
<base_dn>	検索ベースの識別名 (DN)、またはポイント。すべての検索は、LDAP ディレクトリ内のこの場所から実行されます。多くの場合、このパラメータは、o=mcom.com のようにディレクトリのサフィックスまたはルートに設定されます。
<attributes>	検索によって返される属性のリスト。複数の属性を指定するには、「cn,mail,telephoneNumber」のように、属性をコンマで区切ります。属性を指定しないと、すべての属性が返されます。このパラメータは、ダイナミックグループメンバーシップのチェックでは無視されます。
<scope>	検索のスコープ。次のいずれかの値を指定します。 ■ base は URL に指定された識別名 (<base_dn>) に関する情報だけを取得します。 ■ one は URL に指定された識別名 (<base_dn>) の 1 レベル下のエントリに関する情報を取得します。このスコープにはベースエントリは含まれません。 ■ sub は URL に指定された識別名 (<base_dn>) の下のすべてのレベルのエントリに関する情報を取得します。このスコープにはベースエントリが含まれます。 このパラメータは必須です。
<(filter)>	検索範囲内のエントリに適用される検索フィルタです。管理サーバーのフォームを使用する場合は、この属性を指定する必要があります。括弧は必須であることに注意してください。 このパラメータは必須です。

<attributes>、<scope>、および<(filter)>パラメータは、URL 内での位置によって識別されます。どの属性も指定しない場合でも、疑問符を含めてそのフィールドを区切る必要があります。

- オプションで、新しいグループの説明を追加することもできます。
- 定義済みの組織単位がディレクトリに存在している場合、「新規グループの追加先」リストを使って新しいグループの配置場所を指定できます。デフォルトの場所は、ディレクトリのルートポイント、つまり最上位のエントリです。

◆ ◆ ◆ 第 9 章

9 サーバーコンテンツの管理

この章では、複数の仮想サーバーにわたるコンテンツを構成および管理する方法について説明します。

ドキュメントディレクトリの構成

基本ドキュメントディレクトリ (ドキュメントルートとも呼ばれる) は、リモートクライアントに対して使用可能にするすべてのファイルの格納先となる、中央ディレクトリです。

基本ドキュメントディレクトリ以外にも、ドキュメントディレクトリを作成できます。そうすれば、あるユーザーが、基本ドキュメントルートにアクセスすることなしに、一連のドキュメントを管理できるようになります。

▼ ドキュメントディレクトリを作成する

1 構成を選択します。

構成のリストから構成を選択します。利用可能な構成を取得するには、「構成」タブをクリックします。

2 仮想サーバーを選択します。

新しいドキュメントディレクトリを追加する必要のある仮想サーバーを選択します。選択された構成に設定された仮想サーバーのリストを取得するには、「仮想サーバー」タブをクリックします。

3 「コンテンツ処理」>「ドキュメントディレクトリ」タブをクリックします。

4 「新規」ボタンをクリックします。下記のパラメータを設定します。

- **URL** プレフィックス — ディレクトリにマップする必要のある URI プレフィックス。
- ディレクトリパス — ドキュメントを格納するための絶対サーバーパスと有効なディレクトリ。

注 – **CLI** の使用

CLI 経由でドキュメントディレクトリを作成するには、次のコマンドを実行します。

```
wadm> create-document-dir --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1
--uri-prefix=/config1_uri --directory=../docs1
```

CLI リファレンスの `create-document-dir`(1) を参照してください。

デフォルト **MIME** タイプの変更

サーバーはクライアントにドキュメントを送信する際に、ドキュメントのタイプを識別するセクションを含めることで、クライアントがそのドキュメントを正しく表示できるようにします。ただし、サーバーでドキュメントの拡張子が定義されていないために、ドキュメントの適切なタイプをサーバーが決定できない場合があります。そうした場合にはデフォルト値が送信されます。

デフォルトは通常 text/plain ですが、これは、サーバーでもっともよく格納されるファイルのタイプに設定すべきです。よく使用される MIME タイプのいくつかを、次に示します。

■ text/plain	■ text/html
■ text/richtext	■ image/tiff
■ image/jpeg	■ image/gif
■ application/x-tar	■ application/postscript
■ application/x-gzip	■ audio/basic

▼ デフォルト MIME タイプを変更する

1 構成を選択します。
構成のリストから構成を選択します。利用可能な構成を取得するには、「構成」タブをクリックします。

2 仮想サーバーを選択します。
選択された構成に設定された仮想サーバーのリストを取得するには、「仮想サーバー」タブをクリックします。

3 「コンテンツ処理」>「一般」タブをクリックします。

4 「その他」セクションの「デフォルト MIME タイプ」の値を変更します

注 – CLI の使用

CLI 経由で MIME タイプを作成するには、次のコマンドを実行します。

```
wadm> create-mime-type --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --extensions=sxc application/sxc
```

CLI リファレンスの create-mime-type(1) を参照してください。

仮想サーバーごとに MIME タイプファイルを作成する必要はありません。代わりに、MIME タイプファイルを必要な数だけ作成し、それらを仮想サーバーに関連付けることができます。サーバー上にはデフォルトで、MIME タイプファイルが 1 つ (mime.types) 存在していますが、このファイルを削除することはできません。

ユーザー公開情報ディレクトリのカスタマイズ (UNIX/Linux)

ユーザーが自分の Web ページを保守したい場合があります。サーバー上のすべてのユーザーが管理者の介入なしにホームページやその他のドキュメントを作成できるようにするための、公開情報ディレクトリを構成できます。

クライアントはこのシステムを使用することで、サーバーが公開情報ディレクトリとして認識する特定の URL を使ってサーバーにアクセスすることができます。たとえば、プレフィックス ~ とディレクトリ `public_html` を選択したとします。サーバーは、`http://www.sun.com/~jdoe/aboutjane.html` に対する要求を受信すると、`~jdoe` があるユーザーの公開情報ディレクトリを参照していると認識します。サーバーは、システムのユーザーデータベース内で `jdoe` を検索し、Jane のホームディレクトリを見つけます。そこでサーバーは、`~/jdoe/public_html/aboutjane.html` を検索します。

公開ディレクトリを使用するようにサーバーを構成するには、次の手順に従います。

▼ ドキュメントディレクトリの構成

1. 仮想サーバーのページから「コンテンツ処理」タブをクリックします。
2. 「ドキュメントディレクトリ」をクリックします。
3. 「ユーザードキュメントディレクトリ」の下でユーザー **URL** プレフィックスを選択します。

   通常使用されるプレフィックスは ~ です。なぜなら、チルド文字が、ユーザーのホームディレクトリにアクセスするための標準 UNIX/Linux プレフィックスであるからです。
4. **HTML** ファイルの検索場所として使用する、ユーザーのホームディレクトリ内のサブディレクトリを選択します。

   典型的なディレクトリは、`public_html` です。
5. パスワードファイルを指定します。

   サーバーは、システム上のユーザーのリストを含むファイルをどこで検索すべきかを知る必要があります。サーバーは、このファイルを使用することで、有効なユーザー名を決定し、それらのユーザーのホームディレクトリを見つけます。システムのパスワードファイルをこの目的で使用する場合、サーバーは標準ライブラリ呼び

出しを使ってユーザーの検索を行います。あるいは、別のユーザーファイルをユーザー検索用として作成することもできます。そのユーザーファイルは、絶対パスを使って指定できます。

ファイル内の各行が次の構造になるようにしてください (/etc/passwd ファイル内の不要な要素は * で示してある)。

```
username:*:*:groupid:*:homedir:*
```

6 起動時にパスワードデータベースを読み込むかどうかを選択します。

7 「保存」をクリックします。

詳細については、「ユーザードキュメントディレクトリ」ページに対するオンラインヘルプを参照してください。

ユーザーに個別のディレクトリを割り当てるための、もう 1 つの方法は、すべてのユーザーが変更可能な中央ディレクトリへの URL マッピングを作成することです。

コンテンツ発行の制限

状況によっては、システム管理者が、ユーザードキュメントディレクトリ経由でのユーザーアカウントによるコンテンツ発行を制限したい場合があります。ユーザーの発行を制限するには、/etc/passwd ファイルで、ユーザーのホームディレクトリパスの末尾にスラッシュを追加します。

```
jdoe::1234:1234:John Doe:/home/jdoe:/bin/sh
```

これは次のようになります。

```
jdoe::1234:1234:John Doe:/home/jdoe/:/bin/sh
```

この変更が完了すると、Sun Java System Web Server は、このユーザーのディレクトリからページを提供しないようになります。この URI を要求したブラウザは「404 ファイルが見つからない」エラーを受け取り、Web サーバーのアクセスログに 404 エラーが記録されます。エラーログには何のエラーも記録されません。

あとになって、このユーザーにコンテンツ発行を許可することにした場合は、/etc/passwd エントリの末尾のスラッシュを削除したあと、Web サーバーを再起動します。

起動時のパスワードファイル全体の読み込み

パスワードファイルの全体を起動時に読み込むオプションも用意されています。このオプションを選択すると、サーバーが起動時にパスワードファイルをメモリー内

に読み込むようになるため、ユーザー検索が大幅に高速化されます。ただし、パスワードファイルが非常に大きい場合にこのオプションを選択すると、メモリー使用量が多くなりすぎる可能性があります。

URL リダイレクションの設定

URL リダイレクションを使えば、ある HTTP URL へのドキュメント要求を別の HTTP URL にリダイレクトできます。URL 転送つまりリダイレクションは、ファイルを別のディレクトリやサーバーに移動したなどの理由により URL が変わったことを、サーバーがユーザーに通知するための手段の 1 つです。リダイレクションを使用すれば、あるサーバー上のドキュメントに対する要求を別のサーバー上のドキュメントにシームレスに送信することもできます。

たとえば、`http://www.sun.com/info/movies` をプレフィックス`film.sun.com` に転送するようにした場合、URL `http://www.sun.com/info/movies` は `http://film.sun.com/info/movies` にリダイレクトされます。

あるサブディレクトリ内のすべてのドキュメントに対する要求を、ある特定の URL にリダイレクトしたい場合があります。たとえば、トラフィックが多すぎた、ドキュメントが何らかの理由で提供できなくなった、などの理由により、ディレクトリを削除しなければいけなかった場合、それらのドキュメントのどれか 1 つに対する要求を、ドキュメントが利用できなくなった理由を説明したページに転送できます。たとえば、`/info/movies` のプレフィックスを `http://www.sun.com/explain.html` にリダイレクトできます。

URL リダイレクションの設定は、仮想サーバーレベルで行えます。

URL リダイレクションを構成するには、次の手順を実行します。

1. 「構成」タブをクリックし、構成のリストから構成を選択します。
2. 「仮想サーバー」サブタブをクリックし、仮想サーバーのリストから仮想サーバーを選択します。
3. 「コンテンツ処理」サブタブ、「**URL** リダイレクト」サブタブを順次クリックします。
4. 「新規」ボタンをクリックして新しい URL リダイレクト規則を追加します。
5. 次の表で説明するフィールドに、必要な値を入力します。「了解」ボタンをクリックします。必要に応じて、構成の「配備」ボタンをクリックしなければいけない可能性があります。

次の表では、新しい URL リダイレクト規則を追加する場合の必須パラメータについて説明します。

表9–1 URLリダイレクトのパラメータ

パラメータ	説明
リダイレクト元**URL**	要求のリダイレクト元となるURL。このURLへのHTTP要求はすべて、「ターゲットURL」で指定されたURLにリダイレクトされます。
ターゲット**URL**	要求のリダイレクト先となるURL。「リダイレクト元URL」で指定されたURLからのHTTP要求はすべて、このURLにリダイレクトされます。
URLタイプ	固定。有効/無効。固定されたURLとは、HTMLページへのリンクなど、静的なURLのことです。固定されないURLとは、要求パラメータを持つ動的URLや、プレフィックスのみを持つURLのことです。

注 – **CLI**の使用

CLI経由で新しいURLリダイレクション規則を追加するには、次のコマンドを実行します。

```
wadm> create-url-redirect --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --no-ssl --config=config1 --vs=config1_vs_1 --uri-prefix=/redirect
--target-url=http://www.cnet.com
```

CLIリファレンスの`create-url-redirect`(1)を参照してください。

正規表現を使用した**URL**リダイレクション

Sun Java System Web Server 7.0は、正規表現(パターンとも呼ばれる)と、要求時にパラメータを構成ファイルに挿入する機能をサポートするように拡張されています。さらに、ワイルドカードパターンマッチングのサポートが、server.xmlにまで拡張されています。URLリダイレクションはSAFとして実装されています。リダイレクトSAFを使えば、ある特定のプレフィックスに一致するURIをリダイレクトできます。プレフィックスは`from`パラメータを使って、リダイレクト先のURLは`url`または`url-prefix`パラメータを使って、それぞれ指定できます。Sun Java System Web Server 7.0では、`from`パラメータは省略可能です。`from`を省略すると、すべてのURIがリダイレクトされます。

obj.confファイル内のSAFパラメータでは、新しいタグ`<If>`、`<Elseif>`、および`<Else>`がサポートされています。付録 - obj.conf - 構文と使用法を参照してください。これらのタグには指令が含まれます。これらのタグを使えば、指令の実行条件を定義できます。また、これらのタグは、SAFパラメータを動的に生成するために使用することもできます。

Sun Java System Web Server 7.0 が提供する URL 書き換え機能は、Apache HTTP サーバーの mod_rewrite モジュールのスーパーセットになっています。<If> タグは Apache の mod_rewrite 機能とは異なり、次の機能を提供します。

- URI、パス、ヘッダーフィールド、および応答本文を操作できる。
- 要求処理のどの段階でも機能する。
- 他社製プラグインを含む任意の SAF と連携できる。

次のような指令があるとします。

```
NameTrans fn="redirect"
                    from="/site1"
                url="http://site1.mycompany.com"
```

この指令は、正規表現を使って次のように書き換えられます。

```
<If $uri =~ '^/site1'>
           NameTrans fn="redirect"
           url="http://site1.mycompany.com"
</If>
```

このコードでは、from パラメータの代わりに正規表現が使用されています。/site1/* に対するすべての要求を、http://site1.mycompany.com/*/index.html にリダイレクトする必要がある場合には、次のテクニックを検討してください。

```
<If $uri =~ '^/site1/(.*)'>
           NameTrans fn="redirect"
           url="http://site1.mycompany.com/$1/index.html"
</If>
```

この場合、<If> タグによって、(.*) に一致する値が何であれ、その値が変数 $1 に代入されます。url パラメータの $1 は、元の要求からの値で動的に置換されます。つまり、上記の obj.conf コードによって、/site1/download に対する要求が http://site1.mycompany.com.com/download/index.html にリダイレクトされます。

<If> と redirect の組み合わせれば、mod_rewrite の柔軟性の一部を実現できます。ただし、<If> は mod_rewrite とは異なり、URL のリダイレクションや書き換え以外の用途にも使用できます。<If> は、他社製プラグインを含む任意の SAF と組み合わせて使用することもできます。

上記の方法では、302 一時的に移動リダイレクトが構成されます。Sun Java System Web Server 7.0 では、status="301" パラメータを追加することで、301 永久に移動リダイレクトを代わりに必要としていることを示すこともできます。

```
NameTrans fn="redirect" from="/path" url="http://server.example.com" status="301"
```

CGIの概要

CGI (Common Gateway Interface) プログラムは、任意の数のプログラミング言語を使って定義できます。UNIX/Linux マシン上にはおそらく、Bourne シェルや Perl スクリプトとして記述された CGI プログラムが存在しています。

注 – UNIX/Linux の下ではさらに `CGIStub` プロセスも実行されますが、これらは、サーバーが CGI 実行の支援用として使用するものです。これらのプロセスは、CGI への初回アクセス時にのみ作成されます。その個数は、サーバーでの CGI 負荷に応じて変わります。これらの CGIStub プロセスを終了しないでください。これらはサーバー停止時に削除されます。

詳細については、オンラインの『Sun Java System Web Server パフォーマンスのチューニング、サイジング、およびスケーリング』に含まれている、`MinCGIStub`、`MaxCGIStub`、および `CGIStubIdleTimeout` に関する説明を参照してください。

Windows コンピュータにはおそらく、C++ またはバッチファイルで記述された CGI プログラムが存在しています。Windows の場合、Visual Basic などの Windows ベースのプログラミング言語で記述された CGI プログラムは、異なる機構を使ってサーバーとの連携動作を行います。これらは Windows CGI プログラムと呼ばれます。

注 – コマンド行ユーティリティーを実行するには、`Path` 変数を手動で設定して *server_root*`/bin/https/bin` を含める必要があります。

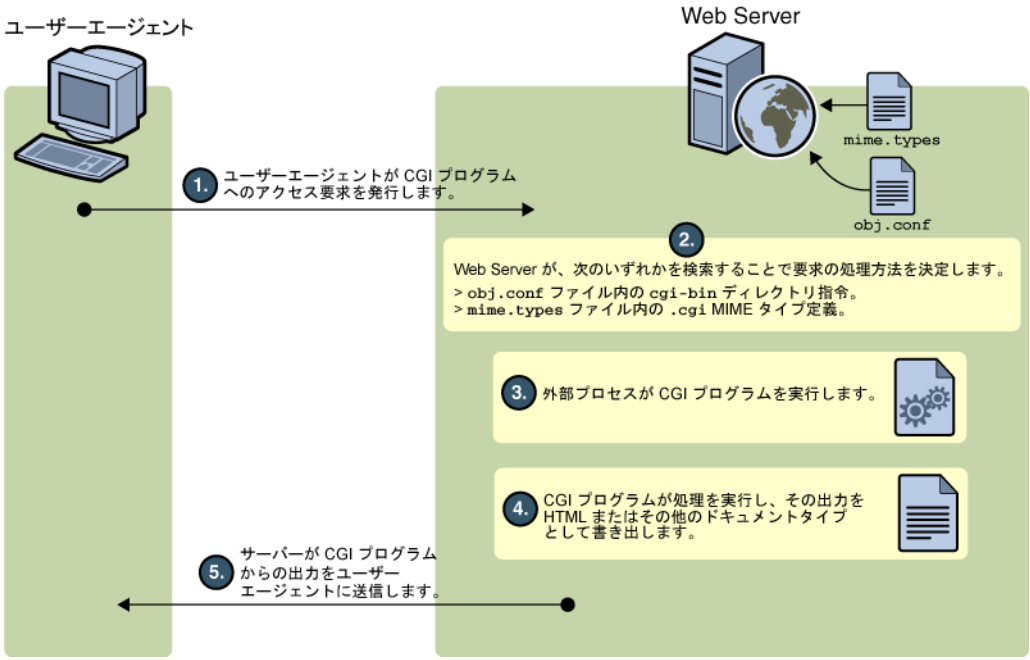
プログラミング言語にかかわらず、データを受け取ったり返したりする方法は、どの CGI プログラムでも同じです。CGI プログラムの記述方法については、次の情報源を参照してください。

- 『Sun Java System Web Server *Developer's Guide*』
- 「*The Common Gateway Interface*」

  `http://hoohoo.ncsa.uiuc.edu/cgi/overview.html`
- 次のオンラインドキュメント Web サイトで利用可能な、CGI に関する記事

  `http://docs.sun.com`

次の図は、Web Server 7.0 での CGI 要求の処理手順を示したものです。



CGI プログラムをサーバーマシンに格納する方法は 2 つあります。

- CGI プログラムだけを格納するディレクトリを指定します。拡張子にかかわらず、すべてのファイルがプログラムとして実行されます。
- すべての CGI プログラムがある特定のファイルタイプを持つことを指定します。つまり、それらはすべて、ファイル拡張子 `.cgi`、`.exe`、または `.bat` を使用します。ドキュメントルートディレクトリの中またはその下にある任意のディレクトリに、プログラムを格納できます。

必要であれば、両方のオプションを同時に有効にすることも可能です。

どちらの実装にも利点があります。ある特定のユーザーグループだけが CGI プログラムを追加できるようにするには、CGI プログラムを特定のディレクトリ内に格納し、それらのディレクトリへのアクセスを制限します。HTML ファイルを追加できるすべてのユーザーが CGI プログラムも追加できるようにするには、ファイルタイプの選択肢を使用します。ユーザーは、HTML ファイルと同じディレクトリ内に CGI ファイルを格納できます。

ディレクトリのオプションが選択されると、サーバーはそのディレクトリ内のすべてのファイルを CGI プログラムとして解釈しようとします。同様に、ファイルタイプのオプションが選択されると、サーバーは、ファイル拡張子 `.cgi`、`.exe`、または `.bat` を持つファイルをすべて、CGI プログラムとして処理しようとします。あるファイルがこれらの拡張子を持つにもかかわらず、CGI プログラムではなかった場合、ユーザーがそのファイルにアクセスしようとした時点でエラーが発生します。

注 – デフォルトでは、CGI プログラムのファイル拡張子は .cgi、.exe、および .bat. です。ただし、MIME タイプファイルを変更すれば、CGI プログラムを表す拡張子を変更できます。これを行うには、「構成」タブを選択し、「MIME タイプ」リンクをクリックします。

サーバーの CGI サブシステムの構成

Sun Java System Web Server では、管理コンソール GUI を使って CGI ドキュメントディレクトリを追加できます。

新しい CGI ドキュメントディレクトリを追加するには、次のタスクを実行します。

1. 「構成」タブをクリックし、構成のリストから構成を選択します。
2. 「仮想サーバー」サブタブをクリックし、仮想サーバーのリストから仮想サーバーを選択します。
3. 「コンテンツ処理」サブタブ、「**CGI**」サブタブを順次クリックします。
4. 「新規」ボタンをクリックして新しい CGI ドキュメントディレクトリを追加します。
5. 次の表で説明するフィールドに、必要な値を入力します。「了解」ボタンをクリックします。必要に応じて、構成の「配備」ボタンをクリックしなければいけない可能性があります。

次の表では、新しい CGI ドキュメントディレクトリを追加する場合の必須フィールドについて説明します。

表 9–2 CGI パラメータ

パラメータ	説明
プレフィックス	このディレクトリに対して使用する URL プレフィックスを入力します。つまり、ここで入力したテキストが、CGI プログラムのディレクトリとして URL 内に現れます。 たとえば、cgi-bin を URL プレフィックスとして入力した場合、これらの CGI プログラムへの URL はすべて、次の構造を持ちます。 http://*yourserver.domain.com* /cgi-bin/*program-name*

表 9–2 CGI パラメータ (続き)

パラメータ	説明
CGI ディレクトリ	「CGI ディレクトリ」テキストフィールドには、ディレクトリの場所を絶対パスで入力します。このディレクトリはドキュメントルートの下になくてもかまわないことに注意してください。これが、URL プレフィックスを指定する必要がある理由です。 注 – ユーザーが指定する URL プレフィックスは、実際の CGI ディレクトリと異なっていてもかまいません。
ユーザー	CGI プログラムの実行時に使用するユーザーの名前を指定します。
グループ	CGI プログラムの実行時に使用するグループの名前を指定します。
ディレクトリ変更	実行開始前のディレクトリ変更先となるディレクトリを指定します。
優先順位	優先順位値、つまり CGI プログラムの優先順位を決定する、サーバーからの相対的な増分を指定します。 通常、サーバーは優先順位値 0 で実行され、優先順位の増分は、0 (CGI プログラムがサーバーと同じ優先順位で実行される) から 19 (CGI プログラムがサーバーよりも大幅に低い優先順位で実行される) までになります。CGI プログラムの優先順位をサーバーの優先順位よりも上げるために優先順位の増分に -1 を指定することも可能ですが、これはお勧めできません。

既存の CGI ディレクトリを削除するには、CGI ディレクトリを選択し、「削除」ボタンをクリックします。既存のディレクトリの URL プレフィックスや CGI ディレクトリを変更するには、そのディレクトリのリンクをクリックします。

指定したディレクトリ内に CGI プログラムをコピーします。これらのディレクトリ内のファイルはすべて CGI ファイルとして処理されるため、HTML ファイルを CGI ディレクトリに格納しないでください。

CGI をファイルタイプとして指定するには、次のタスクを実行します。

1. 「構成」タブをクリックし、構成のリストから構成を選択します。
2. 「仮想サーバー」サブタブをクリックし、仮想サーバーのリストから仮想サーバーを選択します。
3. 「コンテンツ処理」サブタブ、「**CGI**」サブタブを順次クリックします。
4. 「ファイルタイプとしての **CGI**」ラジオボックスをクリックして有効にします。

CGI ファイルはファイル拡張子 .bat、 .exe、または .cgi を持つ必要があります。CGI ファイルでなくてもこれらの拡張子を持っていれば、サーバーによって CGI ファイルとして処理されるため、エラーが発生します。

注 – **CLI** の使用

サーバーによって処理される CGI プログラムを格納する CGI ディレクトリを作成できます。CGI プログラムは、.cgi や .exe、.bat といった特定のファイルタイプを持ちます。ドキュメントルートディレクトリの中またはその下にある任意のディレクトリに、プログラムを格納できます。

CLI 経由で CGI ディレクトリを追加するには、次のコマンドを実行します。

```
wadm> create-cgi-dir --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --vs=config1_vs_1 --uri-prefix=/config1_urlprefix
--directory=/cgi-dir
```

CLI リファレンスの create-cgi-dir(1) を参照してください。

実行可能ファイルのダウンロード

.exe を CGI ファイルタイプとして使用している場合、.exe ファイルを実行可能ファイルとしてダウンロードすることはできません。

この問題の解決方法の 1 つは、ユーザーがダウンロードできるようにする実行可能ファイルを圧縮することです。そうすれば、拡張子が .exe でなくなります。この解決方法には、ダウンロード時間が短縮されるという利点もあります。

もう 1 つの選択可能な解決方法は、ファイル拡張子としての .exe を magnus-internal/cgi タイプから削除し、代わりに application/octet-stream タイプ (通常のダウンロード可能ファイル用の MIME タイプ) に追加することです。サーバーマネージャーからこれを行うには、「構成」タブを選択し、「MIME タイプ」リンクをクリックします。ただし、この方法には欠点があります。それは、この変更を行うと、.exe ファイルを CGI プログラムとして使用できなくなる、という点です。

もう 1 つの解決方法は、サーバーの obj.conf ファイルを編集してダウンロードディレクトリを設定することです。そうすれば、そのディレクトリ内のすべてのファイルが自動的にダウンロードされるようになります。サーバーのほかの部分に影響を及ぼすことはありません。詳細については、次を参照してください。

http://developer.netscape.com/docs/manuals/enterprise/admnunix/programs.htm

Windows 用シェル CGI プログラムのインストール

Windows 用シェル CGI プログラムの概要

シェル CGI とは、Windows で設定されたファイル関連付けを使って CGI アプリケーションを実行できるようにするサーバー構成です。

たとえば、サーバーは、`hello.pl` という名前のシェル CGI ファイルの要求を受け取ると、Windows のファイル関連付けを使って、具体的には `.pl` 拡張子に関連付けられたプログラムを使って、そのファイルを実行します。`.pl` 拡張子がプログラム `C:\\bin\\perl.exe` に関連付けられていた場合、サーバーは `hello.pl` ファイルを次のようにして実行しようとします。

```
c:\\bin\\perl.exe hello.pl
```

シェル CGI を設定するもっとも簡単な方法は、サーバーのドキュメントルート内に、シェル CGI ファイルだけを格納するディレクトリを作成することです。ただし、サーバーを構成して特定のファイル拡張子をシェル CGI に関連付けることもできます。それには、Sun ONE Web Server から MIME タイプを編集します。

注 – Windows のファイル拡張子の設定方法については、Windows のドキュメントを参照してください。

エラー応答のカスタマイズ

仮想サーバーでのエラー発生時にクライアントに詳細メッセージを送信するための、カスタムのエラー応答を指定できます。送信するファイル、実行する CGI プログラムのいずれかを指定できます。

たとえば、ある特定のディレクトリでエラーが発生した際のサーバーの動作方法を変更できます。アクセス制御によって保護されたサーバーの部分にクライアントが接続しようとした場合、アカウントの取得方法に関する情報を含むエラーファイルを返すことができます。

カスタムエラー応答を有効にするには、まず、エラーへの応答として、送信する HTML ファイル、実行する CGI プログラムのいずれかを作成する必要があります。

カスタムエラーページを追加するには、次の手順に従います。

1. 「構成」タブをクリックし、構成のリストから構成を選択します。
2. 「仮想サーバー」サブタブをクリックし、仮想サーバーのリストから仮想サーバーを選択します。

3. 「コンテンツ処理」サブタブ、「エラーページ」サブタブを順次クリックします。
4. 「新規」ボタンをクリックしてカスタムエラーページを追加します。

   変更するエラーコードごとに、エラー応答が格納されたファイルまたはCGIへの絶対パスを指定します。
5. 「了解」をクリックしてエラーページのリストに戻ります。

注 – **CLI** の使用

CLI経由でエラーページをカスタマイズするには、次のコマンドを実行します。

```
wadm> set-error-page --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --vs=config1_vs_1 --code=500
--error-page=/server-error-uri-new
```

CLIリファレンスの `set-error-page(1)` を参照してください。

文字セットの変更

ドキュメントの文字セットの決定要因の1つは、その記述に使われている言語です。1つのドキュメント、一連のドキュメント、または1つのディレクトリに対するクライアントのデフォルト文字セット設定を上書きするには、あるリソースを選択し、そのリソースの文字セットを入力します。

ほとんどのブラウザは、HTTPに含まれるMIMEタイプの `charset` パラメータに基づいて自身の文字セットを変更できます。サーバーがこのパラメータを応答に含めると、ブラウザはそれに応じて自身の文字セットを変更します。次に例を示します。

- `Content-Type: text/html;charset=iso-8859-1`
- `Content-Type: text/html;charset=iso-2022-jp`

次の `charset` 名は一般的なブラウザの一部で認識されますが、これらはRFC 17.000で規定されたものです (ただし、`x-` で始まる名前は除く)。

■ `us-ascii`	■ `iso-8859-1`
■ `iso-2022-jp`	■ `x-sjis`
■ `x-euc-jp`	■ `x-mac-roman`

さらに、`us-ascii` では次の別名が認識されます。

■ ansi_x3.4-1968	■ iso-ir-6
■ ansi_x3.4-1986	■ iso_646.irv:1991
■ ascii	■ iso646-us
■ us	■ ibm367.0
■ cp367.0	

iso_8859-1 では次の別名が認識されます。

■ latin1	■ iso_8859-1
■ iso_8859-1:1987.0	■ iso-ir-100
■ ibm819	■ cp819

文字セットを変更するには、次の手順に従います。

▼ 文字セットの変更

1. 仮想サーバーのページから「コンテンツ処理」タブをクリックします。
2. 「一般」タブをクリックします。
3. 「その他」セクションでデフォルト文字セットを設定します。
   このフィールドを空白のままにすると、文字セットが「なし」に設定されます。
4. 「保存」をクリックします。

ドキュメントフッターの設定

サーバーのある特定のセクション内のすべてのドキュメントに対し、ドキュメントフッターを指定できます。このフッターには最終更新時刻を含めることができます。このフッターは、CGI スクリプトの出力と解析対象 HTML (.shtml) ファイルを除く、すべてのファイルで機能します。CGI スクリプトの出力や解析対象 HTML ファイルにドキュメントフッターを表示させる必要がある場合には、そのフッターテキストを別のファイルに入力したあと、コード行または別のサーバー側インクルードを追加することで、そのファイルがページ出力の末尾に追加されるようにします。

ドキュメントフッターを設定するには、次の手順に従います。

▼ ドキュメントフッターを設定する

1. 仮想サーバーのページから「コンテンツ処理」タブをクリックします。
2. 「一般」サブタブをクリックし、「ドキュメントフッター」セクションに移動します。
3. フッターを含めるファイルのタイプを指定します。
4. 日付の形式を指定します。
5. フッターに表示させる任意のテキストを入力します。
   ドキュメントフッターの最大文字数は 7,065 です。ドキュメントの最終更新日付を含めるには、文字列 `:LASTMOD:` を入力します。
6. 「保存」をクリックします。

注 – **CLI** の使用

CLI 経由でドキュメントフッターを設定するには、次のコマンドを実行します。

```
wadm> enable-document-footer --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1
--mime-type=text/html --date-format=%B --footer="config1 footer"
```

CLI リファレンスの `enable-document-footer`(1) を参照してください。

シンボリックリンクの制限 (UNIX/Linux)

サーバーでのファイルシステムリンクの使用を制限できます。ファイルシステムリンクとは、ほかのディレクトリまたはファイルシステム内のファイルへの参照のことです。参照を使えば、リモートファイルがまるで現在のディレクトリ内に存在するかのように、リモートファイルにアクセスできるようになります。次の 2 種類のファイルシステムリンクが存在します。

- ハードリンク — ハードリンクは、同一のデータブロックのセットを指している、実際の 2 つのファイル名です。元のファイルとリンクは同一です。このため、ハードリンクは異なるファイルシステム上に存在できません。
- シンボリック (ソフト) リンク — シンボリックリンクは 2 つのファイル、つまりデータが格納された元のファイルと、その元のファイルを指す別のファイルから構成されます。シンボリックリンクはハードリンクよりも柔軟です。シンボリックリンクは、異なるファイルシステムをまたがって使用できるほか、ディレクトリにリンクすることもできます。

ハードリンクとシンボリックリンクの詳細については、UNIX/Linux システムのドキュメントを参照してください。

ファイルシステムリンクを使えば、基本ドキュメントディレクトリの外側にあるドキュメントへのポインタを簡単に作成できますが、そうしたリンクは誰でも作成できます。このため、機密ドキュメントやシステムパスワードファイルといった機密性のあるファイルへのポインタをユーザーが作成してしまう心配が生じます。

シンボリックリンクを制限するには、次の手順に従います。

▼ シンボリックリンクを制限する

1. 仮想サーバーのページから「コンテンツ処理」タブをクリックします。
2. 「一般」サブタブをクリックします。
3. 「その他」セクションの下にある「シンボリックリンク」セクションに移動します。
4. ソフトリンクまたはハードリンクあるいはその両方を有効にするかどうかと、その起動元ディレクトリを選択します。
5. 「保存」をクリックします。

注 – **CLI** の使用

CLI 経由でシンボリックリンクを制限するには、次のコマンドを実行します。

```
wadm> set-symlinks-prop --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1
allow-soft-links=true allow-hard-links=false directory=/abc
```

CLI リファレンスの `set-symlinks-prop`(1) を参照してください。

サーバーにより解析される **HTML** の設定

HTML は通常、サーバーが介入することなしに、ディスク上に存在する状態のままでクライアントに送信されます。ところがサーバーは、ドキュメントを送信する前に、HTML ファイル内で特殊なコマンドを検索する、つまり HTML を解析することができます。サーバーがこれらのファイルを解析し、要求に固有の情報やファイルをドキュメント内に挿入するようにするには、まず HTML の解析機能を有効にする必要があります。

HTML を解析するには、次の手順に従います。

▼ サーバーにより解析される **HTML** を設定する

1. 仮想サーバーのページから「コンテンツ処理」タブをクリックします。
2. 「一般」サブタブをクリックします。
3. 「解析対象 **HTML/SSI** 設定」で、サーバーにより解析される **HTML** を有効にするかどうかを選択します。

   HTML ファイルで有効にするが exec タグでは有効にしないこともできますし、HTML ファイルでも exec タグでも有効にすることもできます。後者の場合、HTML ファイルがサーバー上のほかのプログラムを実行できるようになります。
4. 解析対象ファイルを選択します。

   解析対象として、.shtml 拡張子を持つファイルだけ、すべての HTML ファイルのいずれかを選択できます。後者の場合はパフォーマンスが低下します。UNIX/Linux を使用している場合には、実行権が有効になっている UNIX/Linux ファイルも、解析対象として選択できます。ただし、これはあまり信頼できない可能性があります。
5. 「保存」をクリックします。

   サーバーにより解析される HTML の使用方法の詳細については、『Sun Java System Web Server Developer's Guide.』を参照してください。

注 – **CLI** の使用

CLI 経由でサーバーにより解析される HTML を設定するには、次のコマンドを実行します。

```
wadm> enable-parsed-html --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs1
```

CLI リファレンスの `enable-parsed-html`(1) を参照してください。

キャッシュ制御指令の設定

キャッシュ制御指令は、Sun Java System Web Server がプロキシサーバーのキャッシュにどの情報を書き込むかを制御するための方法の 1 つです。キャッシュ制御指令を使えば、プロキシのデフォルトキャッシュを上書きすることで、機密情報がキャッシュに書き込まれ、あとでおそらく読み出されるのを防ぐことができます。この指令を利用するには、プロキシサーバーが HTTP 1.1 に準拠している必要があります。

HTTP 1.1 の詳細については、次の「Hypertext Transfer Protocol--HTTP/1.1」仕様 (RFC 2068) を参照してください。

```
http://www.ietf.org/
```

キャッシュ制御指令を設定するには、次の手順に従います。

▼ キャッシュ制御指令を設定する

1. 仮想サーバーのページから「コンテンツ処理」タブをクリックします。
2. 「一般」サブタブをクリックし、「その他」セクションの「キャッシュ制御指令」フィールドに移動します。
3. 各フィールドに値を入力します。応答指令の有効な値は、次のとおりです。
   - 公開:任意のキャッシュに応答を書き込めます。これは標準設定です。
     - 非公開:非公開 (非共有) キャッシュにしか応答を書き込めません。
     - キャッシュなし:どのキャッシュにも応答を書き込めません。
     - ストアなし:キャッシュは、要求または応答を非揮発性記憶領域のどこにも格納できません。
     - 再検証が必要:元のサーバーに基づいてキャッシュエントリを再検証する必要があります。
     - 最大継続時間 (秒):クライアントは、この継続時間よりも大きな継続時間を持つ応答を受け取りません。
4. 「保存」をクリックします。

注 – **CLI** の使用

CLI 経由でキャッシュ制御指令を設定するには、次のコマンドを実行します。

```
wadm> set-cache-control-directives --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1 public=true
private=true must-revalidate=true
```

CLI リファレンスの `set-cache-control-directives`(1) を参照してください。

コンテンツを圧縮するためのサーバー設定

Sun Java System Web Server 7.0 は HTTP コンテンツ圧縮をサポートします。コンテンツを圧縮することで、ハードウェアに負担をかけることなくクライアントへの配信速度を向上させ、コンテンツのボリュームを増やすことができます。コンテンツ圧縮により、コンテンツのダウンロード時間が減少します。これは、ダイアルアップ接続やトラフィックの多い接続を使用するユーザーにとって非常に重要な利点です。

コンテンツを圧縮した場合、Web Server は圧縮されたデータを送信し、そのデータを直ちに展開 (解凍) するようにブラウザに指示を出します。このため、送信するデータの容量が減り、ページの表示速度が速くなります。

事前に圧縮されたコンテンツを提供するためのサーバー構成

ある指定されたディレクトリで、事前に圧縮されたバージョンのファイルを生成して格納するように、Sun Java System Web Server を構成できます。そのように構成され、かつ `Accept-encoding: gzip` ヘッダーが受信された場合にのみ、事前に圧縮されたコンテンツを提供するように構成されたディレクトリ内のファイルへのすべての要求が、そのディレクトリ内のそれと同等な圧縮済みファイルへの要求にリダイレクトされます。ただしそれには、そのようなファイルが存在している必要があります。たとえば、Web サーバーが `myfile.html` の要求を受信し、かつ `myfile.html` と `myfile.html.gz` がどちらも存在していた場合、それらの要求に適切な `Accept-encoding` ヘッダーが含まれていれば、圧縮されたファイルが送信されます。

事前に圧縮されたコンテンツを提供するようにサーバーを構成するには、次の手順を実行します。

▼ 事前に圧縮されたコンテンツの設定を変更する

1. 仮想サーバーのページから「コンテンツ管理」タブをクリックします。
2. 「一般」サブタブをクリックします。
3. 「圧縮」>「事前に圧縮されたコンテンツ」セクションに移動し、次のオプションから選択します。
   - 事前に圧縮されたコンテンツ — 有効化/無効化。選択されたリソースに対して事前に圧縮されたコンテンツを提供するように、サーバーに指示できるようにします。
   - 経過時間チェック — 圧縮版が非圧縮版より古いかどうかをチェックするかどうかを指定します。

     これを選択すると、圧縮版が非圧縮版より古い場合に圧縮版が選択されません。

これを選択しないと、圧縮版が非圧縮版より古い場合でも圧縮版が常に選択されます。

- **Vary** ヘッダーを挿入 — `Vary: Accept-encoding` ヘッダーを使用するかどうかを指定します。

  これを選択すると、圧縮版のファイルが選択されたときに `Vary: Accept-encoding` ヘッダーが常に挿入されます。

  これを選択しないと、`Vary: Accept-encoding` ヘッダーは挿入されません。

4 「保存」をクリックします。

コンテンツをオンデマンドで圧縮するためのサーバー設定

転送データを動的に圧縮するように Sun Java System Web Server 7.0 を構成することもできます。動的に生成される HTML ページは、ユーザーがそれを要求するまで生成されません。これは特に、電子商取引ベースの Web アプリケーションやデータベース駆動型のサイトで役立ちます。

オンデマンドでコンテンツを圧縮するようにサーバーを構成するには、次の手順を実行します。

▼ オンデマンドでコンテンツを圧縮する

1 仮想サーバーのページから「コンテンツ処理」タブをクリックします。

2 「一般」サブタブをクリックします。「圧縮」セクションの「オンデマンドのコンテンツ圧縮」セクションに移動します。

3 次のオプションがあります。

- オンデマンド圧縮 — 選択されたリソースのオンデマンド圧縮を有効化/無効化します。
- **Vary** ヘッダーを挿入 — `Vary: Accept-encoding` ヘッダーを挿入するかどうかを指定します。

  これを選択すると、圧縮版のファイルが選択されたときに `Vary: Accept-encoding` ヘッダーが常に挿入されます。

  これを選択しないと、`Vary: Accept-encoding` ヘッダーは挿入されません。
- フラグメントサイズ — 圧縮ライブラリ (zlib) が一度に圧縮する分量を制御するために使用する、メモリーのフラグメントサイズをバイトで指定します。デフォルト値は `8096` です。

- 圧縮レベル — 圧縮のレベルを指定します。1 ～ 9 の値を選択します。値 1 では速度が最高になり、値 9 では圧縮率が最高になります。デフォルト値は、速度と圧縮率の両方を考慮した 6 です。

4 「保存」をクリックします。

注 – **CLI** の使用

CLI 経由でオンデマンド圧縮を有効にするには、次のコマンドを実行します。

```
wadm> enable-on-demand-compression --user=admin
--password-file=admin.pwd --host=serverhost --port=8989 --config=config1
--vs=config1_vs_1 --insertvaryheader=true
--fragment-size=100 --compression-level=5
```

CLI リファレンスの `enable-on-demand-compression`(1) を参照してください。

逆プロキシの構成

逆プロキシとは、クライアントからは Web サーバー (元のサーバー) のように見えるが、実際には受信した要求を 1 つ以上の元のサーバーに転送するようなプロキシのことです。逆プロキシは自身を元のサーバーとして提示するため、逆プロキシを使用するようにクライアントを構成する必要はありません。ある特定の逆プロキシを、同様に構成された複数の元のサーバーに要求を転送するように構成すれば、その逆プロキシは、アプリケーションレベルのソフトウェアロードバランサとしての役割を果たすことができます。

注 – 典型的な配備では、ブラウザと元のサーバーとの間に逆プロキシが 1 つ以上配備されます。

▼ プロキシ **URI** を追加する

1 「構成」タブをクリックし、構成を選択します。

2 「仮想サーバー」タブをクリックし、仮想サーバーを選択します。

3 「コンテンツ処理」>「逆プロキシ」タブをクリックします。

4 「新規プロキシ **URI**」ボタンをクリックします。
次の各パラメータの値を指定します。

- **URI** — 逆プロキシの URI

- サーバー **URL** — リモートサーバーの URL をコンマで区切ったもの。複数の値が指定された場合、サーバーはその指定されたサーバー間で負荷分散を行います。

▼ 逆プロキシのパラメータを変更する

1. 「構成」タブをクリックし、構成を選択します。
2. 「仮想サーバー」タブをクリックし、仮想サーバーを選択します。
3. 「コンテンツ処理」>「逆プロキシ」タブをクリックします。
4. **URI** をクリックします。
   次のパラメータを編集できます。
   - **URI** — 逆プロキシの URI
   - サーバー **URL** — リモートサーバーの URL をコンマで区切ったもの。複数の値が指定された場合、サーバーはその指定されたサーバー間で負荷分散を行います。
   - **Sticky Cookie** — Cookie の名前。この Cookie が応答に含まれていると、後続の要求がその元のサーバーに固定されます。
   - **Sticky URI** パラメータ — ルート情報を調べる URI パラメータの名前。要求 URI にこの URI パラメータが含まれていて、かつその値にコロン「:」とルート ID が含まれていると、そのルート ID で特定される元のサーバーに要求が「固定」されます。
   - ルートヘッダー — 元のサーバーにルート ID を伝えるために使用される HTTP 要求ヘッダーの名前。
   - ルート **Cookie** — サーバーが応答内で sticky-cookie Cookie を検出したときに生成する Cookie の名前。route-cookie Cookie に格納されたルート ID を使えば、サーバーは後続の要求を同じ元のサーバーに転送できます。

注 – **CLI** の使用

1. `create-reverse-proxy` コマンドを呼び出します。

```
wadm> create-reverse-proxy --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=test --vs=test --uri-prefix=//
--server=http://rick.india.sun.com:8080
```

CLI リファレンスの `create-reverse-proxy`(1) を参照してください。

2. `obj.conf` ファイルを変更します。

```
NameTrans fn="map" from="/" name="reverse-proxy-/" to="http:/"
...
<Object name="reverse-proxy-/">
Route fn="set-origin-server" server="http://rick.india.sun.com:8080"
</Object>

<Object ppath="http:*">
Service fn="proxy-retrieve" method="*"
</Object>
```

セキュリティー保護されたサイトにリダイレクトするには、同じ手順に従い、`--server` オプションで `https` アドレスを指定します。

P3P の設定

- 151 ページの「仮想サーバーの P3P 設定の構成」

P3P (Platform for Privacy Preferences) を使えば、Web サイトは自身のプライバシ運用規定を、ユーザーエージェントが自動的に取得して簡単に解釈できるような標準形式で表明することができます。P3P ユーザーエージェントを使えば、ユーザーは、サイトの運用規定を (マシン可読、人間可読の両形式で) 通知されます。詳細については、http://www.w3.org/P3P/ を参照してください。

▼ 仮想サーバーの P3P 設定の構成

1 構成を選択します。

構成のリストから構成を選択します。利用可能な構成のリストを取得するには、「構成」タブをクリックします。

2 仮想サーバーを選択します。

仮想サーバーのリストから仮想サーバーを選択します。選択された構成で利用可能な仮想サーバーを取得するには、「仮想サーバー」タブをクリックします。

3 「一般」タブをクリックします。「**P3P**」セクションで次の設定を構成します。

- 有効 — 選択された仮想サーバーで P3P を有効にします。
- ポリシー **URL**— 関連する P3P ポリシーファイルの場所を入力します。
- コンパクトポリシー — コンパクトポリシーは、ユーザーエージェント (ブラウザまたはその他の P3P アプリケーション) にヒントを提供することで、そのユーザーエージェントが、ポリシーの適用に関する判断をすばやく同期の取れたかたちで下せるようにします。コンパクトポリシーはパフォーマンスを最適化するためのものであり、P3P 仕様によれば、ユーザーエージェント、サーバーのいずれかのオプションとして提供されています。

注 – **CLI** の使用

仮想サーバーの P3P を有効にするには、次のコマンドを実行します。

```
wadm> enable-p3p --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --vs=config1_vs_1 --policy-url=http://xyz.com/policyurl
```

CLI リファレンスの `enable-p3p`(1) を参照してください。

◆ ◆ ◆ 第 10 章

WebDAV を使用した Web パブリッシング

Sun Java System Web Server 7.0 は、Web ベースのコラボレーションの標準である WebDAV、つまり Web ベースの分散オーサリングおよびバージョン管理 (Web-based Distributed Authoring and Versioning) をサポートしています。WebDAV は HTTP/1.1 プロトコルの拡張機能であり、クライアントがリモート Web コンテンツのオーサリング操作を実行できるようにします。

完全な WebDAV トランザクションには、Sun Java System Web Server 7.0 など、WebDAV リソースへの要求を処理できる WebDAV 対応サーバーと、Adobe® GoLive® や Macromedia® DreamWeaver® など、WebDAV 対応の Web パブリッシング要求をサポートする WebDAV 対応クライアントが必要になります。

サーバー側では、WebDAV 要求を処理できるように Sun Java System Web Server 7.0 を有効化および構成する必要があります。

WebDAV を構成する理由としては、サーバーのパフォーマンスチューニング、セキュリティーリスクの解消、衝突の発生しないリモートオーサリングなど、いくつかの理由が考えられます。

ユーザーは構成の要件に応じて、サーバーが WebDAV リソースのロックを保持する時間の最小値、コレクションに対する PROPFIND 要求の実行範囲、要求の本文内に指定可能な XML コンテンツの最大サイズなどを変更することができます。

仮想サーバーレベルでは、ある仮想サーバーのすべてのコレクションに対するデフォルトの WebDAV 属性を構成できます。ここで構成された値は、`server.xml` ファイル内の `DAV` 要素に対応します。

WebDAV 属性はコレクションレベルでも構成可能であり、その値は、仮想サーバーレベルでそのコレクションに対して構成されたどの属性よりも優先されます。コレクションレベルで構成された属性値は、`server.xml` ファイル内の `DAVCOLLECTION` 要素に対応します。

WebDAV について

WebDAV は HTTP/1.1 プロトコルの拡張機能であり、新しい HTTP メソッドやヘッダーが追加されていますが、それらのメソッドやヘッダーは、HTML や XML だけでなく、テキスト、グラフィックス、スプレッドシート、およびその他のすべての形式を含む、あらゆるタイプの Web リソースに対してオーサリングサポートを提供します。WebDAV を使って実現可能なタスクを、いくつか次に示します。

- プロパティー (メタデータ) の操作: WebDAV メソッドの PROPFIND と PROPPATCH を使えば、作成者や作成日付など、Web ページに関する情報の作成、削除、およびクエリーを行えます。
- コレクションおよびリソースの管理: WebDAV メソッドの GET、PUT、DELETE、および MKCOL を使えば、一連のドキュメントを作成したり、ファイルシステムのディレクトリ一覧に似た階層的なメンバーシップ一覧を取得したりできます。
- ロック: WebDAV を使えば、あるドキュメントを複数のユーザーが同時に操作しないようにできます。WebDAV メソッドの LOCK や UNLOCK による排他ロックや共有ロックを使えば、「失われた更新」(変更の上書き) の問題が発生しにくくなります。
- 名前空間の操作: WebDAV では、WebDAV メソッドの COPY や MOVE を使って Web リソースのコピーや削除をサーバーに指示できます。

Sun Java System Web Server 7.0 の WebDAV サポートによって提供される機能は、次のとおりです。

- RFC 2518 準拠による RFC 2518 クライアントとの相互運用性
- パブリッシングのセキュリティーおよびアクセス制御。
- ファイルシステムベースの WebDAV コレクションおよびリソースに対する効率的な発行操作。

WebDAV の一般的な用語

この節では、WebDAV を扱う際に目にする一般的な用語について、簡単に説明します。

URI: URI (Uniform Resource Identifier) はファイル識別子の 1 つであり、省略形の URL を使用することで追加のセキュリティー層を提供します。URL の最初の部分が URL マッピングで置換されるため、ファイルの完全な物理パス名がユーザーからわからないようになります。

ソース URI: ソース URI という用語は、リソースのソースへのアクセスが可能な URI を意味します。ソース URI の概念を理解するために、次のような例を考えます。

JSP ページ `foo.jsp` が URI `/docs/date.jsp` に格納されています。このページには HTML マークアップと Java コードが含まれていますが、この Java コードは実行時に、クライアントのブラウザ上に現在日付を出力します。サーバーは、`foo.jsp` の GET 要求をクライアントから受信すると、そのページを送信する前に Java コードを実行します。クライアントが受信するのは、サーバー上に存在している状態の `foo.jsp` ではなく、動的に生成された、現在日付が表示されたページです。

たとえば、`/publish/docs` というソース URI を作成し、それを `foo.jsp` を含む `/docs` ディレクトリにマップした場合、`/publish/docs/foo.jsp` に対する要求は、`/docs/foo.jsp` JSP ページのソースコードに対する要求になります。この場合、サーバーは Java コードを実行せずにページを送信します。クライアントは、ディスク上に格納された状態のままの、未処理のページを受信します。

したがって、ソース URI に対する要求は、リソースのソースに対する要求になります。

コレクション: WebDAV コレクションとは、WebDAV 操作に対応したリソースまたは一連のリソースのことです。コレクションには一連の URI が含まれますが、これらはメンバー URI と呼ばれ、WebDAV に対応した各メンバーリソースを識別します。

メンバー URI: コレクション内の一連の URI のメンバーになっている URI。

内部メンバー URI: コレクションの URI の直接の相対パスになっているようなメンバー URI。たとえば、URL `http://info.sun.com/resources/info` のリソースが WebDAV に対応しており、URL `http://info.sun.com/resources/` のリソースも WebDAV に対応している場合、URL `http://info.sun.com/resources/` のリソースはコレクションであり、`http://info.sun.com/resources/info` を内部メンバーとして含みます。

プロパティー: リソースについての記述情報を含む名前と値のペア。プロパティーは、リソースの検索や管理を効率的に行うために使用されます。たとえば、「creationdate」プロパティーを使えば、すべてのリソースのインデックス作成をリソースの作成日付に基づいて行えますし、「author」プロパティーを使えば、それを作成者名に基づいて行えます。

ライブプロパティー:サーバーによって管理されるプロパティー。たとえば、`getcontentlength` ライブプロパティーはその値として、GET 要求によって返されるエンティティーの長さを持ちますが、この値はサーバーによって自動計算されます。ライブプロパティーには次のものがあります。

- 値がサーバーによって維持される、読み込み専用のプロパティー
- 値がクライアントによって維持されるプロパティー。ただし、サーバーは送信されてきた値に対して構文チェックを実行します。

デッドプロパティー:サーバーによって管理されないプロパティー。デッドプロパティーの場合、サーバーはその値を記録するだけです。その整合性を維持する責任はクライアントにあります。

Sun Java System Web Server がサポートするライブプロパティーは、次のとおりです。

- `creationdate`
- `displayname`
- `getcontentlanguage`
- `getcontentlength`
- `getcontenttype`
- `gettag`
- `getlastmodified`
- `lockdiscovery`
- `resourcetype`
- `supportedlock`
- `executable`

注 – Sun Java System Web Server がサポートするライブプロパティー executable を使えば、クライアントがリソースに関連付けられたファイルアクセス権を変更できます。

executable ライブプロパティーに対する PROPPATCH 要求の例を、次に示します。

```
PROPPATCH /test/index.html HTTP/1.1

Host: sun

Content-type: text/xml

Content-length: XXXX

<?xml version="1.0"?>

<A:propertyupdate xmlns:A="DAV:" xmlns:B="http://apache.org/dav/props/">

<A:set>

<A:prop>

<B:executable>T</B:executable>

</A:prop>

</A:set>

</A:propertyupdate>
```

ロック: リソースをロックする機能は、あるユーザーが編集しているリソースを別のユーザーが変更しないことを保証する機構を提供します。ロックを使えば上書きの衝突を回避でき、「失われた更新」の問題を解決できます。

Sun Java System Web Server がサポートするロックは 2 種類あります。共有と排他です。

新しい HTTP ヘッダー: WebDAV は HTTP/1.1 プロトコルを拡張することで機能します。クライアントが WebDAV リソースに対する要求を伝えるための HTTP ヘッダーが、新たに定義されています。それらのヘッダーを次に示します。

- Destination:
- Lock-Token:
- Timeout:
- DAV:
- If:
- Depth:

- Overwrite:

新しい HTTP メソッド: WebDAV では、WebDAV 対応サーバーに要求の処理方法を指示する新しい HTTP メソッドが、いくつか導入されました。GET、PUT、DELETE などの既存の HTTP メソッドに加えてそれらのメソッドを使用することで、WebDAV トランザクションが実行されます。新しい HTTP メソッドの簡単な説明を、次に示します。

- COPY: リソースをコピーするために使用されます。コレクションのコピー時には、Depth: ヘッダーが使用され、Destination: ヘッダーによってコピー先が指定されます。COPY メソッドでは、Overwrite: ヘッダーも必要に応じて使用されます。
- MOVE: リソースを移動するために使用されます。コレクションの移動時には、Depth: ヘッダーが使用され、Destination: ヘッダーによって移動先が指定されます。MOVE メソッドでは、Overwrite: ヘッダーも必要に応じて使用されます。
- MKCOL: 新しいコレクションを作成するために使用されます。このメソッドは、PUT メソッドのオーバーロードを回避するために使用されます。
- PROPPATCH: 単一リソース上のプロパティーを設定、変更、または削除するために使用されます。
- PROPFIND: 1 つ以上のリソースに属する 1 つ以上のプロパティーを取得するために使用されます。クライアントがあるコレクションの PROPFIND 要求をサーバーに送信する場合、その要求には、0、1、infinity のいずれかの値を持つ Depth: ヘッダーが含まれている可能性があります。
  - 0: 指定された URI のコレクションのプロパティーを取得することを指定します。
  - 1: コレクションのプロパティーと、指定された URI のすぐ下にあるリソースのプロパティーを取得することを指定します。
  - infinity: コレクションのプロパティーと、そのコレクションに含まれるすべてのメンバー URI のプロパティーを取得することを指定します。無限の実行範囲を持つ要求ではコレクションの全体が検索されるため、サーバーに大きな負荷がかかる可能性があるので、注意が必要です。

  LOCK: リソースにロックを追加します。Lock-Token: ヘッダーを使用します。
- UNLOCK: リソースからロックを削除します。Lock-Token: ヘッダーを使用します。

インスタンスレベルで WebDAV を有効にする

管理サーバーを使えば、WebDAV をサーバー全体で有効にすることができます。そのようにした場合、WebDAV プラグインを読み込むための次の指令が、magnus.conf file に追加されます。

```
Init fn="load-modules" shlib="/s1ws6.1/lib/libdavplugin.so" funcs="init-dav,ntrans-dav,pcheck-dav,service-dav"
shlib_flags="(global|now)"
Init fn="init-dav" LateInit=yes
```

init-dav Init 関数は、WebDAV サブシステムを初期化して登録します。

WebDAV を有効にするには、CLI で次のコマンドを実行します。

```
wadm> enable-webdav --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=test
```

CLI リファレンスの enable-webdav(1) を参照してください。

WebDAV コレクションの管理

WebDAV コレクションの有効化

WebDAV コレクションを有効にするには、次のコマンドを実行します。

```
wadm> enable-dav-collection --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1 --uri=/dav_config1
```

CLI リファレンスの enable-dav-collection(1) を参照してください。

WebDAV コレクションの無効化

WebDAV コレクションを無効にするには、次のコマンドを実行します。

```
wadm> disable-dav-collection --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1 --uri=/dav_config1
```

CLI リファレンスの disable-dav-collection(1) を参照してください。

WebDAV コレクションの追加

WebDAV コレクションを追加するには、次のコマンドを実行します。

```
wadm> create-dav-collection --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1 --uri=/dav_config1
--source-uri=/dav_config1
```

CLI リファレンスの create-dav-collection(1) を参照してください。

WebDAV コレクションの一覧表示

すべての WebDAV コレクションを一覧表示するには、次のコマンドを実行します。

```
wadm> list-dav-collections --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --vs=config1_vs_1
```

CLI リファレンスの list-dav-collections(1) を参照してください。

WebDAV コレクションの削除

WebDAV コレクションを削除するには、次のコマンドを実行します。

```
wadm> delete-dav-collection --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --vs=config1_vs_1 --uri=/dav_config1
```

CLI リファレンスの delete-dav-collection(1) を参照してください。

WebDAV プロパティーの構成

WebDAV プロパティーの設定

サーバーレベルの WebDAV プロパティーを設定するには、次のコマンドを実行します。

```
wadm> set-webdav-prop --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 acl-max-entries=120
```

CLI リファレンスの set-webdav-prop(1) を参照してください。

WebDAV プロパティーの表示

サーバーレベルの WebDAV プロパティーを表示するには、次のコマンドを実行します。

```
wadm> get-webdav-prop --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1
```

CLI リファレンスの `get-webdav-prop`(1) を参照してください。

WebDAV コレクションのプロパティーの設定

WebDAV コレクションのプロパティーを設定するには、次のコマンドを実行します。

```
wadm> set-dav-collection-prop --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --vs=config1_vs_1 --uri=/dav_config1 min-lock-timeout=1
```

CLI リファレンスの `set-dav-collection-prop`(1) を参照してください。

WebDAV コレクションのプロパティーの表示

WebDAV コレクションのプロパティーを表示するには、次のコマンドを実行します。

```
wadm> get-dav-collection-prop --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 -config=config1 --vs=config1_vs_1 --uri=/dav_config1
```

CLI リファレンスの `get-dav-collection-prop`(1) を参照してください。

WebDAV パラメータの変更

一般的な WebDAV プロパティーのいくつかを、次の表に示します。

表 **10–1** WebDAV パラメータ

パラメータ	説明
ロックデータベースのパス	ロックデータベースが維持されるディレクトリを指定します。
最小ロックタイムアウト	ロックの最短の寿命を秒単位で指定します。値 **-1** は、ロックが決して期限切れにならないことを意味します。この値は、ある要素のロックが自動的に解除されるまでの時間を示します。
最大要求サイズ	XML 要求本文の最大サイズを指定します。サービス拒否攻撃に備えてこの値を構成すべきです。デフォルト値は **8192 (8K)** です。

表 10–1 WebDAV パラメータ *(続き)*

パラメータ	説明
最大拡張プロパティー **Depth**	最大拡張プロパティー Depth 要求の実行範囲を指定します。**0** の場合、指定されたリソースのみが処理対象になります。これがデフォルト値です。**1** の場合、指定されたリソースとその次のレベルが処理対象になります。infinity の場合、指定されたリソースとそれに含まれるすべてのリソースが処理対象になります。また、このパラメータのサイズを制限することで、過剰なメモリー消費を防止するようにしてください。
デフォルトの所有者	コレクションのデフォルトの所有者。
URI	WebDAV が有効になっている既存のルート URI。
最大 PROPFIND Depth	コレクションに送信される PROPFIND 要求の実行範囲の最大値。
ロックデータベースの更新間隔	WebDAV ロックデータベースがディスクとの同期を取る間隔。WebDAV ロック情報のキャッシュを無効にするには、0 を使用します。
認証データベース	使用する ACL 認証データベース。
認証方法	使用する認証方法。デフォルトの認証方法は「基本」です。
認証確認テキスト	認証要求時にクライアントに表示される確認テキスト。
DAV ACL データベース	
最大エントリ数	単一のリソースで許可される ACE の最大数。0–2147.0483647.0。制限なしの場合は —1 を指定します。
最大サイズ	1 つのコレクションに対する WebDAV ACL データベースのメモリー表現の最大サイズ。0–2147.0483647.0。制限なしの場合は —1 を指定します。
更新間隔	WebDAV ACL データベースがディスクとの同期を取る間隔。0.001–3600 秒。WebDAV ACL リストのキャッシュを無効にする場合は、0 を指定します。
DAV プロパティーデータベース	
最大サイズ	WebDAV プロパティーデータベースファイルの最大サイズ。0–2147.0483647.0。制限なしの場合は —1 を指定します。
更新間隔	WebDAV プロパティーデータベースがディスクとの同期を取る間隔。0.01–3600 秒。WebDAV プロパティーのキャッシュを無効にする場合は、0 を指定します。

サーバーレベルでの WebDAV の無効化

サーバーレベルで WebDAV を無効にするには、次のコマンドを実行します。

```
wadm> disable-webdav --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1
```

CLI リファレンスの `disable-webdav`(1) を参照してください。

WebDAV 認証データベースの管理

管理コンソールで WebDAV 認証データベースの設定を編集するには、選択された構成から「**WebDAV**」タブをクリックします。次の表に、そのページ内の各フィールドの簡単な説明を示します。

表 **10–2** WebDAV 認証データベースのプロパティー

プロパティー	説明
認証データベース	「認証データベース」では、サーバーがユーザーの認証に使用するデータベースを選択できます。 デフォルトは **keyfile** です

表 10–2 WebDAV 認証データベースのプロパティー (続き)

プロパティー	説明
認証方法	■ 基本 — HTTP メソッドを使用してクライアントから認証情報を取得します。ユーザー名とパスワードがネットワーク上で暗号化されるのは、サーバーで SSL が有効になっている場合だけです。 ■ **SSL** — クライアント証明書を使用してユーザーの認証を行います。このメソッドを使用するには、サーバーの SSL を有効にする必要があります。暗号化が有効になっていれば、基本と SSL の方法を組み合わせることができます。 ■ ダイジェスト — ユーザー名とパスワードを平文として送信することなしにユーザー名とパスワードに基づく認証をブラウザが行うための手段を提供する認証機構を使用します。ブラウザは *MD5* アルゴリズムを使用して、ユーザーのパスワードと Web Server によって提供される情報の一部を使用するダイジェスト値を作成します。「ダイジェスト」を使用するには、背後の auth-db もダイジェストをサポートしている必要があります。これは、ダイジェスト認証プラグインがインストールされている場合にのみ、digestfile を使用したファイル auth-db、LDAP auth-db のいずれかを意味します ■ その他 — アクセス制御 API を使って作成されたカスタム方法を使用します。
認証確認テキスト	「認証確認テキスト」オプションでは、認証ダイアログボックス内に表示されるメッセージテキストを入力できます。このテキストを使用して、ユーザーが入力する必要のある項目について説明することができます。ブラウザによっては、最初の 40 文字程度しか表示されません。 Web ブラウザは通常、ユーザー名とパスワードをキャッシュし、それらをプロンプトのテキストと関連付けます。ユーザーが、サーバーの同じ確認テキストを持つファイルやディレクトリにアクセスしても、ユーザー名とパスワードを再度入力する必要はありません。特定のファイルやディレクトリでユーザーに再度認証させたい場合は、そのリソースの ACL の確認テキストを変更すればよいだけです。

WebDAV 対応サーバーでのソース URI と Translate:f ヘッダーの使用

WebDAV メソッドはリソースまたはコレクションのソースを処理します。GET や PUT などの HTTP メソッドは WebDAV プロトコルによってオーバーロードされた

め、これらのメソッドを含む要求は、リソースのソースへの要求、リソースのコンテンツ (出力) への要求のいずれかになります。

Microsoft やその他の多くの WebDAV ベンダーは、要求と一緒に `Translate:f` ヘッダーを送信し、その要求がソースに対するものであることをサーバーに知らせるようにすることで、この問題を解決しました。広く利用されている WebDAV クライアントである Microsoft WebFolders との相互運用性を確保するため、Sun Java System Web Server 7.0 は、`Translate:f` ヘッダーをリソースのソースへの要求として認識します。`Translate:f` ヘッダーを送信しないクライアントに対応するため、Sun Java System Web Server はソース URI を定義します。

WebDAV に対応したコレクションの場合、URI を要求するとリソースのコンテンツ (出力) が取得され、ソース URI を要求するとリソースのソースが取得されます。`Translate:f` ヘッダーを含む URI への要求は、ソース URI への要求として扱われます。

デフォルトでは、サーバーインスタンスに固有の ACL ファイル内にある、次の宣言を含む `dav-src` ACL によって、リソースのソースへのアクセスのすべてが拒否されることに注意してください。

```
deny (all) user = "anyone";
```

ソースへのアクセスをあるユーザーに許可するには、ソース URI へのアクセス権限を追加してください。

リソースのロックとロック解除

Sun Java System Web Server では、サーバー管理者は、リソースをロックすることでそのリソースへのアクセスを直列化できます。ある特定のリソースにアクセスするユーザーは、ロックを使用することで、別のユーザーがその同じリソースを変更することがない、という安心感が得られます。こうして、サーバー上のリソースを複数のユーザーが共有する場合に生じる「失われた更新」の問題が解決されます。サーバーによって維持されるロックデータベースは、発行され、クライアントによって使用されるロックトークンを追跡します。

Sun Java System Web Server がサポートする `opaquelocktoken` URI スキーマは、すべての時間にわたってすべてのリソース間で一意になるように設計されています。これには、ISO-1157.08 で説明されている汎用一意識別子 (UUID) 機構が使用されています。

Sun Java System Web Server は次の 2 種類のロック機構を認識します。

- 排他ロック。
- 共有ロック。

排他ロック

排他ロックとは、リソースへのアクセスを単一ユーザーにのみ許可するロックのことです。ほかのユーザーがその同じリソースにアクセスできるのは、そのリソースの排他ロックが解除された後です。

排他ロックは、リソースをロックする機構としては、厳格でコストがかかりすぎる場合があると言われています。たとえば、プログラムがクラッシュしたりロック所有者がリソースのロック解除を忘れたりした場合、ロックタイムアウトや管理者の介入がないと排他ロックを解除できません。

共有ロック

共有ロックを使えば、あるリソースのロックを複数のユーザーが取得できます。したがって、適切なアクセス権を持つユーザーであれば、誰でもロックを取得できます。

共有ロックを使用する場合、複数のロック所有者は、何らかの通信手段を使って作業の調整を行えます。共有ロックの目的は、共同作業者に、ほかのどのユーザーがリソースを操作する可能性があるかを知らせることです。

最小ロックタイムアウト

ロックを制御するには、`server.xml` ファイル内の `DAV` または `DAVCOLLECTION` オブジェクトの `minlocktimeout` 属性の値を構成します。`minlocktimeout` 属性は、ロックの最小の寿命を秒単位で指定します。この値は、ある要素のロックが自動的に解除されるまでの時間を示します。

これは省略可能な属性です。値を -1 に設定すると、ロックが決して期限切れにならなくなります。値を `0` に設定すると、要求に指定された Timeout ヘッダーのタイムアウトで、コレクション内のすべてのリソースをロックできるようになります。

Timeout ヘッダーが指定されていないと、無限大のタイムアウトでリソースがロックされます。要求の Timeout ヘッダーが値 `Infinite` に設定されていると、やはり無限大のタイムアウトでリソースがロックされます。

WebDAV リソースの要求に含まれる Timeout ヘッダー値が、`server.xml` に指定された `minlocktimeout` 値に等しいかそれより大きい場合、その要求に指定された期間の間、リソースがロックされます。

これに対し、要求の Timeout ヘッダー値が `server.xml` に指定された `minlocktimeout` 値より小さい場合には、その `server.xml` に指定された `minlocktimeout` 値でリソースがロックされます。

次の表に、Sun Java System Web Server がロック要求を処理する方法を示します。

表 10–3　Sun Java System Web Server がロック要求を処理する方法

要求の **Timeout** ヘッダーの設定値	リソースのロック方法
`Infinite`	タイムアウトが -1 (無限大) に設定されてロックされます
なし	タイムアウトが -1 (無限大) に設定されてロックされます
`Second-`*xxx*	■ *xxx* が `server.xml` に設定された `minlocktimeout` 値に等しいかそれより大きい場合、*xxx* 値でロックされます または、 ■ *xxx* が `server.xml` に設定された `minlocktimeout` 値よりも小さい場合、その `server.xml` に指定された `minlocktimeout` 値でロックされます。

注 – **CLI** の使用

CLI 経由でロックの期限切れを設定するには、次のコマンドを実行します。

```
wadm> expire-lock  --user=admin --password-file=admin.pwd
--host=serverhost--port=8989 --config=config1 --vs=config1
--collection-uri=/dav1 --lock-uri=/dav1/file.html
--opaque-token=opaquelocktoken
```

CLI リファレンスの `expire-lock`(1) を参照してください。

上の例の opaque-token は、期限切れに設定するロックの ID を指定しています。

CLI 経由で既存のロックを表示するには、次のコマンドを実行します。

```
wadm> list-locks --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --vs config1 --collection-uri=/dav1
```

CLI リファレンスの `list-locks`(1) を参照してください。

◆ ◆ ◆ 第 11 章

11 Java と Web アプリケーションの操作

この章では、仮想サーバーの Java 設定を編集する手順について説明します。Java 設定の編集は、管理コンソール、`wadm` コマンド行ツールのいずれかから行えます。この章では、Sun Java System Web Server で構成可能な各種 Java リソースについても説明します。

この章では、Sun Java System Web Server での Java Web アプリケーションの配備方法についても説明します。

Sun Java System Web Server と連携動作するように Java を構成

この節では、選択された構成で Java を有効にし、Java ホーム変数を設定できるようにします。

▼ 構成の Java の有効化

1 構成を選択します。
構成のリストから構成を選択します。利用可能な構成のリストを取得するには、「構成」タブをクリックします。

2 「**Java**」>「一般」タブをクリックします。

3 「**Java** を有効にする」チェックボックスをクリックします。
構成の Java サポートの有効/無効を切り替えます。Java を有効にすると、サーバーが Java アプリケーションを処理できるようになります。

4 「**Java** ホーム」を設定します。
Java SE の場所を指定します。絶対パス、サーバーの `config` ディレクトリからの相対パスのいずれかを指定します。

5 「スティキーを張り付ける」を設定します。
サーバーが各 HTTP 要求処理スレッドを JVM に一度だけ接続するかどうかを指定します (一度だけ接続しない場合、サーバーは要求が発生するたびに HTTP 要求処理スレッドの接続/接続解除を行う)。

注 – **CLI** の使用

構成の Java を有効にするには、次のコマンドを実行します。

```
wadm> enable-java --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1
```

CLI リファレンスの `enable-java`(1) を参照してください。

Java クラスパスの設定

この節では、選択された構成の JVM クラスパスを追加できるようにします。

▼ Java クラスパスを設定する

1 構成を選択します。
構成のリストから構成を選択します。利用可能な構成のリストを取得するには、「構成」タブをクリックします。

2 「Java」>「パス設定」タブをクリックします。
次のパラメータを編集します。

- 環境クラスパスを無視 — デフォルトで有効になっています。
- クラスパスのプレフィックス — システムクラスパスのプレフィックス。システムクラスパスのプレフィックスを追加するのは、XML パーサークラスなどのシステムクラスを上書きする場合だけにすべきです。これは注意して使用してください。
- サーバークラスパス — サーバークラスを含むクラスパス。読み取り専用のリスト。
- クラスパスのサフィックス — クラスパスの末尾に追加します。
- ネイティブライブラリパスのプレフィックス — オペレーティングシステムのネイティブライブラリパスのプレフィックス。
- バイトコードプリプロセッサクラス — `com.sun.appserv.BytecodePreprocessor` を実装するクラスの完全修飾名。実行時クラスインストゥルメンテーションを実行するための典型的な方法は、前処理機構を使用することです。この機構では、プロファイリングおよび監視ツールが、JVM によって Java クラスが読み込まれる直前に、クラスプリプロセッサを使って Java クラス内の必要な場所にインストゥルメンテーションコードを挿入します。この目的のために、クラスプリプロセッサはクラスローダーと連携して動作します。

JVM の構成

管理インタフェースで JVM コマンド行オプションを設定するには、次のタスクを実行します。

▼ JVM を構成する

1 構成を選択します。
構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「Java」>「JVM 設定」タブをクリックします。
JVM の設定を構成します。

JVM オプションの追加

ここで値を指定することで、コマンド行 JVM オプションを追加/削除できます。

JVM オプションを追加するには、「JVM オプションを追加」ボタンをクリックします。

JVM オプションの例を次にいくつか示します。
`-Djava.security.auth.login.config=login.conf`、
`-Djava.util.logging.manager=com.iplanet.ias.server.logging.ServerLogManager`、
および `-Xms128m -Xmx256m`

注 – CLI の使用

CLI 経由で JVM オプションを追加するには、次のコマンドを実行します。

```
wadm> create-jvm-options --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 -Dhttp.proxyHost=proxyhost.com -Dhttp.proxyPort=8080
```

CLI リファレンスの `create-jvm-options`(1) を参照してください。

JVM プロファイラの追加

JVM プロファイラは、アプリケーションの最高レベルの安定性とスケーラビリティーを確実にするために、Java アプリケーションのパフォーマンスの問題、メモリーのリーク、マルチスレッドの問題、およびシステムリソース使用率の問題の診断と解決に役立ちます。

▼ JVM プロファイラを追加する

1. 構成を選択します。
   構成のリストから構成を選択します。利用可能な構成のリストを取得するには、「構成」タブをクリックします。
2. 「**Java**」>「**JVM 設定**」タブをクリックします。
3. 「プロファイラ」セクションの「新規」ボタンをクリックします。
4. 次の各パラメータの値を入力します。
   - 名前 — 新しい JVM プロファイラの簡易名を入力します。
   - 有効 — 実行時にプロファイラを有効にするかどうかを決定します。
   - クラスパス — プロファイラの有効なクラスパスを入力します。(省略可能)。
   - ネイティブライブラリパス — 有効なネイティブライブラリパスを入力します。(省略可能)。
   - **JVM** オプション — CLI の追加の JVM オプションを指定できます。

注 – **CLI** の使用

CLI 経由で JVM プロファイラを追加するには、次のコマンドを実行します。

```
wadm> create-jvm-profiler --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1
```

CLI リファレンスの `create-jvm-profiler`(1) を参照してください。

サーバーの **Java** デバッグの有効化

JVM はデバッグモードで起動でき、JPDA (Java Platform Debugger Architecture) デバッガと接続できます。デバッグを有効にすると、ローカルとリモートのデバッグがどちらも有効になります。

Sun Java System Web Server のデバッグは JPDA ソフトウェアに基づいています。デバッグを有効にするには、次のタスクを実行します。

▼ **JVM** デバッグの有効化

1. 構成を選択します。
   構成のリストから構成を選択します。利用可能な構成のリストを取得するには、「構成」タブをクリックします。
2. 「**Java**」>「**JVM** 設定」タブをクリックします。
3. 「**Java** 設定のデバッグ」で「デバッグを有効にする」チェックボックスを選択します。
4. 必要であれば、「新規」ボタンをクリックして **JVM** オプションを入力します。
   デフォルトの JPDA オプションは次のとおりです。

   ```
   -Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=7896
   ```

   代わりに `suspend=y` に使用すると、JVM は中断モードで起動され、デバッグが接続するまで中断された状態に保たれます。これは、JVM の起動後すぐにデバッグを開始したい場合に便利です。JVM をデバッガに接続するときに使用するポートを指定するには、`address=port_number` を指定します。デバッグオプションの一覧については、JPDA のドキュメントを確認してください。

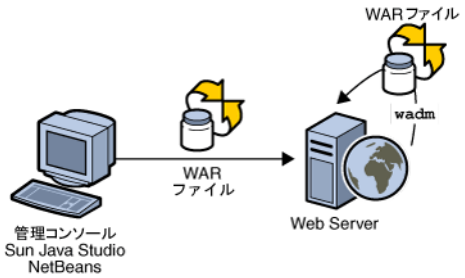
Java Web アプリケーションの配備

Web アプリケーションの追加

Web アプリケーションは、どの既存の仮想サーバーにも配備できます。

▼ Web アプリケーションを配備する

始める前に
- Web アプリケーションを配備する必要のある仮想サーバーの特定が完了しているべきです。
- Web アプリケーションアーカイブ (.war ファイル) が手元にあるか、サーバー内での Web アプリケーションのパスを知っている必要があります。



Web アプリケーションの配備は、`wadm`、管理コンソール、およびその他のサポートされている IDE 経由で行えます。

1. **Web** アプリケーションを配備するには、あるサーバー構成の下で「仮想サーバー」タブをクリックします。
2. **Web** アプリケーションを配備する必要のある仮想サーバーを選択します。
3. 「**Web** アプリケーション」タブ>「新規」ボタンをクリックします。
4. **Web** アプリケーションのパッケージを指定します。
   Web アプリケーションアーカイブをアップロードする必要がある場合は、「参照」ボタンをクリックしてアーカイブを選択します。あるいは、サーバーに格納されている Web アプリケーションアーカイブを指定することもできます。
5. **Web** アプリケーションの **URI** を指定します。これがアプリケーションのコンテキストルートになりますが、これはサーバーホストに対する相対パスです。
6. **Web** アプリケーションについての簡単な説明を入力します。

7 「JSP プリコンパイル」を有効化/無効化します。
この指令を有効にすると、Web アプリケーション内に存在するすべての JSP がプリコンパイルされるため、パフォーマンスが向上します。

8 アプリケーションを有効にします。
Web アプリケーションの状態を「無効」に設定すると、そのアプリケーションは要求を処理できません。ただし、このオプションは、アプリケーションをインスタンスに再配備することなしに、いつでも切り替えられます。

9 アプリケーションを配備します。
「配備」ボタンをクリックして Web アプリケーションを配備します。

コンテキストルートを指定することでアプリケーションにアクセスできます。例:
`http://<your-server>:<port>/<URI>`

注 – CLI の使用

```
wadm> add-webapp --user=admin --password-file=admin.passwd --host=localhost
 --port=8888 --config=config1 --vs=HOSTNAME --uri=/hello /home/test/hello.war
```

CLI リファレンスの `add-webapp`(1) を参照してください。

Web アプリケーションのディレクトリの配備

管理サーバーホストマシン上のディレクトリをある構成に配備するには、`–file-on-server` オプションを使用します。次のコマンドを実行します。

```
wadm> add-webapp --user=admin-user --password-file=admin.passwd
--port=8989 --vs=vs1 --config=config1 --file-on-server
--uri=/mywebapp /space/tmp/mywebapp
```

配備中の JSP のプリコンパイル

Web アプリケーションの配備中にそのアプリケーションに含まれる JSP をプリコンパイルするには、次のように `–precompilejsp` オプションを指定してコマンドを実行します。

```
wadm> add-webapp --user=admin-user --password-file=admin.passwd
--port=8989 --vs=vs1 --config=config1 --file-on-server --uri=/mywebapp
--precompilejsp mywebapp.war
```

サーブレットコンテナの構成

この節では、サーブレットコンテナの構成手順について説明します。

▼ サーブレットコンテナを設定する

1 構成を選択します。
構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「**Java**」>「サーブレットコンテナ」をクリックします。

サーブレットコンテナのグローバルパラメータ

次の表では、サーブレットコンテナのページで利用可能なパラメータについて説明します。

表 11–1 サーブレットコンテナのパラメータ

パラメータ	説明
ログレベル	サーブレットコンテナのログの冗長レベル。値は、もっとも詳細 (もっとも冗長)、より詳細、詳細、情報、警告、失敗、構成、セキュリティー、または重大 (もっとも冗長でない) にすることができます。
動的再読み込み間隔	このパラメータは、配備済み Web アプリケーションが変更されていないかサーバーがチェックする時間間隔を定義します。値の範囲は 1 から 60 までです。ただし、動的再読み込みを無効にする場合は –1 にします。
匿名ロール	すべての主体割り当てられるデフォルトまたは匿名のロールの名前。デフォルトのロールは ANYONE です。
サーブレットプールサイズ	`SingleThreadedServlet` ごとにインスタンス化するサーブレットインスタンスの数。値の範囲は 1 から 4096 までです。
ディスパッチャー最大実行範囲	入れ子の要求ディスパッチを許可するサーブレットコンテナの最大実行範囲。値の範囲は 0 から 2147.0483647.0 までです。デフォルト値は 20 です。
クロスコンテキストを許可	要求ディスパッチに別のコンテキストへのディスパッチを許可するかどうか。デフォルト値は false です。

表 11–1 サーブレットコンテナのパラメータ (続き)

パラメータ	説明
Cookie を符号化	サーブレットコンテナが Cookie の値を符号化するかどうか。デフォルト値は true です。
セッション **ID** を再利用	既存のセッション ID の番号をそのクライアントの新しいセッションを作成するときに再利用するかどうか。デフォルト値は false です。
セキュアセッション **Cookie**	動的/True/False。このパラメータは、どのような条件下で JSESSIONID Cookie がセキュアとマークされるかを制御します。セキュリティー保護された接続 (HTTPS) で要求が受信された場合にのみ Cookie をセキュアとマークするには、「動的」(デフォルト) を使用します。 常にセキュアとマークする場合は「True」を、決してセキュアとマークしない場合は「False」を、それぞれ選択します。

サーバーライフサイクルモジュールの構成

Java サーバーライフサイクルモジュールはサーバーのライフサイクルイベントを待機する Java クラスであり、その目的は、サーバーの起動や停止といったサーバーイベントが発生するたびに特定のタスクを実行することにあります。

このサーバーは、Web サーバー環境内での Java ベースの短期または長期タスクの実行をサポートします。これらのタスクは、サーバー起動時に自動的に起動され、サーバー停止時にその旨を通知されます。このため、シングルトンや RMI サーバーのインスタンス化などのタスクを組み込むことが可能となります。

サーバーのライフサイクルについての簡単な説明を、次に示します。

サーバーのライフサイクルの概要

- 初期化 — このフェーズには、構成の読み取り、組み込みサブシステム (ネーミング、セキュリティー、およびロギングのサービス) の初期化、および Web コンテナの作成が含まれます。
- 起動 — このフェーズには、配備済みアプリケーションの読み込みと初期化が含まれます
- サービス — サーバーが要求を処理する準備が整いました
- シャットダウン — このフェーズでは、読み込み済みのアプリケーションが停止および破棄されます。システムは停止準備中です。
- 終了 — このフェーズでは、組み込みサブシステムとサーバー実行時環境が終了されます。このフェーズ後に別のアクティビティーが発生することはありません。

- 再構成 — サーバーがサービス状態を保ちつつサーバースレッドが動的に再構成を行っているという、サーバーの一時的な状態。このフェーズは、サーバーの存続期間中に何回か発生する可能性があります。

▼ ライフサイクルモジュールを追加する

1. 構成を選択します。
   構成のリストから構成を選択します。構成のリストを表示するには「構成」タブをクリックします。
2. 「**Java**」>「ライフサイクルモジュール」タブをクリックします。
3. 「新規」ボタンをクリックします。
   次の各パラメータの値を入力します。
   - 名前 — 新しいライフサイクルモジュールの有効な一意名を入力します。
   - 有効 — このライフサイクルモジュールを有効にするには、このオプションを使用します。
   - クラス名 — 完全修飾 Java クラス名。このクラスは、`com.sun.appserv.server.LifecycleListener` インタフェースを実装しているべきです。このインタフェースの詳細については、『*Developer's Guide*』を参照してください。
   - クラスパス — 省略可能です。リスナークラスへのクラスパスを指定できます。
   - 読み込み順序— 100 より大きい値。ライフサイクルイベントリスナーの読み込み順序。数の順番に従います。100 と等しいかそれより大きい読み込み順序を選択することをお勧めします。そうすれば、内部ライフサイクルモジュールとの衝突を回避できます。
   - 読み込み時の障害 — このオプションを有効にすると、サーバーがリスナークラスからスローされた例外を致命的な障害として扱わないため、通常の起動が引き続き行われます。デフォルトでは無効です。
   - 説明 — ライフサイクルモジュールについての簡単な説明を入力します。
   - プロパティー — プロパティーを使えば、Java ライフサイクルモジュールに引数を渡せます。新しいプロパティーを追加するには、「プロパティーを追加」ボタンをクリックし、名前、値、および説明のテキストを入力します。

注意 – サーバーライフサイクルリスナークラスはメインのサーバースレッドから同期的に呼び出されるため、リスナークラスがサーバーをブロックしないように、特別な注意を払う必要があります。リスナークラスは必要に応じてスレッドを作成できますが、それらのスレッドがシャットダウン/終了フェーズで停止されるようにする必要があります。

▼ ライフサイクルモジュールを削除する

1. 構成を選択します。
   構成のリストから構成を選択します。構成のリストを表示するには「構成」タブをクリックします。
2. 「**Java**」>「ライフサイクルモジュール」タブをクリックします。
3. ライフサイクルモジュールを選択し、「ライフサイクルモジュールを削除」ボタンをクリックします。

注 – **CLI** の使用

次の例は、myLifecycleModule という名前の Java ライフサイクルモジュールを構成 test に対して作成する方法を示したものです。このモジュールはクラス com.MyLifecycleModule によって実装されています。

```
wadm> create-lifecycle-module --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1
--class=com.sun.webserver.tests.LifecycleClass LifecycleTest
```

CLI リファレンスの create-lifecycle-module(1) を参照してください。

Java ライフサイクルモジュールを一覧表示するには、次のコマンドを実行します。

```
wadm> list-lifecycle-modules --config=test
```

CLI リファレンスの list-lifecycle-modules(1) を参照してください。

Java ライフサイクルモジュールにプロパティーを追加するには、次のコマンドを実行します。

```
wadm> create-lifecycle-module-userprop --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --module=LifecycleTest info=Testing
```

CLI リファレンスの create-lifecycle-module-userprop(1) を参照してください。

Java ライフサイクルモジュールのプロパティーを変更するには、次のコマンドを実行します。

```
wadm> set-lifecycle-module-prop --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --module=LifecycleTest
class-path=/space
```

CLI リファレンスの set-lifecycle-module-prop(1) を参照してください

Java リソースの構成

Web アプリケーションは、リソースマネージャー、データソース (SQL データソースなど)、メールセッション、URL 接続ファクトリなど、さまざまなリソースにアクセスする可能性があります。Java EE プラットフォームはそのようなリソースを、Java Naming and Directory Interface (JNDI) サービスを介してアプリケーションに公開します。

Sun Java System Web Server で作成および管理可能な Java EE リソースは、次のとおりです。

- JDBC データソース
- JDBC 接続プール
- Java メールセッション。
- カスタムリソース。
- 外部 JNDI リソース。

JDBC リソースの構成

JDBC データソースは、Sun Java System Web Server で作成および管理可能な Java EE リソースの 1 つです。

JDBC API は、リレーショナルデータベースシステムと接続するための API です。JDBC API は次の 2 つの部分から成ります。

- アプリケーションコンポーネントがデータベースへのアクセスに使用する、アプリケーションレベルのインタフェース。
- JDBC ドライバを Java EE プラットフォームに接続するためのサービスプロバイダインタフェース。

JDBC データソースオブジェクトは、データソースを Java プログラミング言語で実装したものです。簡単に言えば、データソースとはデータを格納する機能のことです。それは、大企業向けの複雑なデータベースのように高度なものでもかまいませんし、行と列を含むファイルのように単純なものでもかまいません。JDBC データソースは、Sun Java System Web Server 経由で作成および管理可能な Java EE リソースの 1 つです。

JDBC API は標準 SQL データベースアクセスインタフェースを備えた Java 向けの一連のクラスを提供しますが、それらのクラスを使えば、広範なリレーショナルデータベースに統一的な方法で確実にアクセスできます。

JDBC を使えば、事実上すべてのデータベース管理システム (DBMS) に SQL 文を送信できます。これは、リレーショナル DBMS、オブジェクト DBMS のどちらのインタフェースとしても使用されます。

JDBC リソースの追加

CLI 経由で JDBC リソースを追加するには、次のコマンドを実行します。

```
wadm> create-jdbc-resource --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --datasource-class=oracle.jdbc.pool.OracleDataSource jdbc
```

CLI リファレンスの `create-jdbc-resource`(1) を参照してください。

この例の `com.pointbase.jdbc.jdbcDataSource` は、JDBC ドライバクラスを表しています。

サポートされている JDBC ドライバの一覧については、182 ページの「Sun Java System Web Server でサポートされている JDBC ドライバ」を参照してください。

Sun Java System Web Server でサポートされている JDBC ドライバ

次の表に、一般的な JDBC ドライバと、新しい JDBC リソースの追加時に構成する必要のあるプロパティーの一覧を示します。184 ページの「新しい JDBC リソースの追加」を参照してください。

表 11–2　サポートされている一般的な JDBC ドライバの一覧

ドライバ	クラス名	プロパティー
Oracle ドライバ	`oracle.jdbc.pool.OracleDataSource`	■ url ■ user ■ password
Oracle 用 **SJS JDBC** ドライバ	`com.sun.sql.jdbcx.oracle.OracleDataSource`	■ serverName ■ portNumber ■ user ■ password ■ SID
DB2 IBM ドライバ	`com.ibm.db2.jdbc.DB2DataSource`	■ serverName ■ databaseName ■ portNumber ■ user ■ password ■ driverType
DB2 用 **SJS JDBC** ドライバ	`com.sun.sql.jdbcx.db2.DB2DataSource`	■ databaseName ■ locationName ■ packageName ■ password ■ portNumber ■ serverName ■ user
MS SQLServer ドライバ	`com.ddtek.jdbcx.sqlserver.SQLServerDataSource`	■ databaseName ■ password ■ user ■ serverName ■ portNumber

表 11–2 サポートされている一般的な JDBC ドライバの一覧 *(続き)*

ドライバ	クラス名	プロパティー
MS 用 **SJS JDBC** ドライバ	`com.sun.sql.jdbcx.sqlserver.SQLServerDataSource`	■ databaseName ■ password ■ user ■ serverName ■ portNumber
Sybase ドライバ	`com.sybase.jdbcx.SybDataSource`	■ databaseName ■ password ■ portNumber ■ serverName ■ user
Sybase 用 **SJS JDBC** ドライバ	`com.sun.sql.jdbcx.sybase.SybaseDataSource`	■ databaseName ■ password ■ user ■ portNumber ■ serverName
MySQL MM ドライバ	`org.gjt.mm.mysql.jdbc2.optional.MysqlDataSource`	■ serverName ■ port ■ databaseName ■ user ■ password
Informix ドライバ	`com.informix.jdbcx.IfxDataSource`	■ portNumber ■ databaseName ■ IfxIFXHOST (Informix データベースが稼働しているコンピュータの IP アドレスまたはホスト名) ■ serverName ■ user ■ password

表 11–2 サポートされている一般的な JDBC ドライバの一覧 (続き)

ドライバ	クラス名	プロパティー
Informix 用 **SJS JDBC** ドライバ	`com.sun.sql.jdbcx.informix.InformixDataSource`	■ databaseName ■ informixServer (接続先の Informix データベースサーバーの名前) ■ password ■ portNumber ■ severName
PostgreSQL ドライバ	`org.postgresql.ds.PGSimpleDataSource`	■ serverName ■ databaseName ■ portNumber ■ user ■ password
Apache Derby ドライバ	`org.apache.derby.jdbc.EmbeddedDataSource`	■ databaseName ■ user ■ password

JDBC リソースの管理

▼ 新しい JDBC リソースの追加

1 構成を選択します。
構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「**Java**」>「リソース」タブをクリックします。

3 「**JDBC** リソース」セクションの「新規」ボタンをクリックします。

4 ドライバベンダーを選択します。
JNDI 名として一意の値を指定し、選択可能なリストから JDBC ドライバベンダーを選択します。

5 **JDBC** リソースのプロパティーを入力します。
1 つ前の手順で選択した JDBC ドライバベンダーに基づいて、ドライバのクラス名と JDBC リソースプロパティーが自動的に設定されます。

6 確認します。
概要を表示し、「完了」クリックして新しい JDBC リソースを作成します。

JDBC 接続プールの管理

JDBC 接続プールの設定

Sun Java System Web Server 7.0 で JDBC 接続プールを構成するには、`jdbc-resource` 要素を使用します。次の手順に従えば、もっとも簡単な接続プールを構成できます。この例の接続プールは Oracle JDBC ドライバを使用します。

▼ JDBC 接続プールを作成する

1 **wadm** を起動します。

2 **JDBC** リソースを作成します。
基本的な構成の JDBC リソースを作成します。ほかの属性も利用可能であり、それらを使えば接続プールを細かく調整できます。ほかの属性や使用例については、マニュアルページを参照してください。

```
wadm> create-jdbc-resource --config=test
--datasourceclass=oracle.jdbc.pool.OracleDataSource jdbc/MyPool
```

3 ベンダー固有のプロパティーを構成します。
ドライバのベンダー固有のプロパティーを構成するには、プロパティーを使用します。次の例では、プロパティー `url`、`user`、および `password` が、JDBC リソースに追加されています。

```
wadm> add-jdbc-resource-userprop --config=test --jndi-name=jdbc/MyPool
url=jdbc:oracle:thin:@hostname:1521:MYSID user=myuser password=mypassword
```

4 接続検証を有効にします。
プールでは接続検証を有効にすることができます。このオプションを使用すると、接続がアプリケーションに渡される前に、その接続が検証されます。これにより、ネットワークやデータベースサーバーに障害が発生してデータベースが利用できなくなった場合でも、Web サーバーが自動的にデータベース接続を再確立できます。接続の検証は追加オーバーヘッドとなるため、パフォーマンスに若干の影響が生じます。

```
wadm> set-jdbc-resource-prop --config=test --jndi-name=jdbc/MyPool
connection-validation-table-name=test connection-validation=table
```

5 デフォルトのプール設定を変更します。

この例では、最大接続数を変更しています。

```
wadm> set-jdbc-resource-prop --config=test --jndi-name=jdbc/MyPool
max-connections=100
```

6 構成を配備します。

```
wadm> deploy-config test
```

7 **JDBC** ドライバを含む **JAR** ファイルを提供します。

ドライバを実装するクラスをサーバーに知らせる必要があります。これを実行する方法には次の 2 つがあります。

- ドライバの JAR ファイルをサーバーインスタンスの lib ディレクトリ内にコピーします。これがもっとも簡単な方法です。なぜなら、インスタンスの lib ディレクトリに含まれる JAR ファイルは自動的にサーバーに読み込まれ、サーバーから利用可能になるからです。
- JVM の *class-path-suffix* を変更して、JDBC ドライバの JAR ファイルが含まれるようにします。

```
wadm> set-jvm-prop  --config=test class-path-suffix=/export/home/lib/classes12.jar
```

8 **Web** アプリケーションでの使用方法。

- `WEB-INF/web.xml` の変更。

```
<web-app>
...
  <resource-ref>
    <description>JDBC Connection Pool</description>
    <res-ref-name>jdbc/myJdbc</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
...
</web-app>
```

- `WEB-INF/sun-web.xml` の変更。

```
<sun-web-app>
...
  <resource-ref>
    <res-ref-name>jdbc/myJdbc</res-ref-name>
    <jndi-name>jdbc/MyPool</jndi-name>
  </resource-ref>
...
</sun-web-app>
```

- 接続プールの使用。

```
Context initContext = new InitialContext();
    Context webContext = (Context)context.lookup("java:/comp/env");

    DataSource ds = (DataSource) webContext.lookup("jdbc/myJdbc");
    Connection dbCon = ds.getConnection();
```

カスタムリソースの登録

次のタスクを実行すれば、カスタムリソースをインスタンスに登録できます。

▼ カスタムリソースを追加する

1 構成を選択します。
構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「**Java**」>「リソース」タブをクリックします。

3 「カスタムリソース」セクションの「新規」ボタンをクリックします。

カスタムリソースのプロパティー

次の表では、カスタムリソースの作成時に利用可能なプロパティーについて説明します。

表 **11–3** カスタムリソースのプロパティー

プロパティー	説明
JNDI 名	カスタムリソースの一意の JNDI 名を入力します。
有効	実行時にこのカスタムリソースを有効にするかどうかを決定します。
リソースタイプ	このリソースの完全修飾の型。
ファクトリクラス	この型のリソースをインスタンス化するクラス。`javax.naming.spi.ObjectFactory` を実装する、ユーザーが記述したファクトリクラスの完全修飾名。
説明	カスタムリソースの簡単な説明を入力します。
プロパティー	オプションで、「プロパティーを追加」ボタンをクリックして CLI プロパティーを入力します。

注 – **CLI** の使用

CLI 経由でカスタムリソースを作成するには、次のコマンドを実行します。

```
wadm> create-custom-resource --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 --res-type=samples.jndi.customResource.MyBean
--factory-class=samples.jndi.customResource.MyCustomConnectionFactory custom
```

CLI リファレンスの `create-custom-resource`(1) を参照してください。

外部 JNDI リソースの操作

外部 JNDI リソースの作成

このオプションを使えば、外部 Java Naming and Directory Interface (JNDI) リソースを作成できます。外部 JNDI リポジトリに格納されたリソースにアクセスするには、JNDI リソースが必要になります。

▼ 外部 JNDI リソースを追加する

1 構成を選択します。
構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「**Java**」>「リソース」タブをクリックします。

3 「外部 **JNDI** リソース」セクションの「新規」ボタンをクリックします。

外部 JNDI リソースのプロパティー

次の表では、新しい外部 JNDI リソースの追加時に利用可能なプロパティーについて説明します。

表 11–4 外部 JNDI リソースのプロパティー

プロパティー	説明
JNDI 名	新しい外部 JNDI リソースの一意名を入力します。
有効	実行時にこの外部 JNDI リソースを有効にするかどうかを決定します。

表 11–4 外部 JNDI リソースのプロパティー (続き)

プロパティー	説明
外部 **JNDI** 名	外部 JNDI リソースの名前。
リソースタイプ	このリソースの完全修飾の型。
ファクトリクラス	この型のリソースをインスタンス化するクラス。
説明	外部 JNDI リソースの簡単な説明を入力します。
プロパティー	オプションで、「プロパティーを追加」ボタンをクリックして CLI プロパティーを入力します。

注 – **CLI** の使用

CLI 経由で外部 JNDI リソースを作成するには、次のコマンドを実行します。

```
wadm> create-external-jndi-resource --user=admin
--password-file=admin.pwd --host=serverhost --port=8989 --config=config1
--res-type=org.apache.naming.resources.Resource
--factory-class=samples.jndi.externalResource.MyExternalConnectionFactory
--jndilookupname=index.html external-jndi
```

CLI リファレンスの `create-external-jndi-resource`(1) を参照してください。

メールリソースの構成

JMS デスティネーションは、Sun Java System Web Server 経由で作成および管理可能な Java EE リソースです。

多くのインターネットアプリケーションで電子メール通知を送信する機能が必要となるため、Java EE プラットフォームには JavaMail API と JavaMail サービスプロバイダが含まれています。これにより、アプリケーションコンポーネントによるインターネットメールの送信が可能となります。

▼ メールリソースを追加する

1. 構成を選択します。
   構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。
2. 「**Java**」>「リソース」タブをクリックします。
3. 「メールリソース」セクションの「新規」ボタンをクリックします。

メールリソースのプロパティー

次の表では、新しいメールリソースの追加時に利用可能なプロパティーについて説明します。

表 11–5 メールリソースのプロパティー

プロパティー	説明
JNDI 名	新しいメールリソースの一意名を入力します。
有効	実行時にこのメールリソースを有効にするかどうかを決定します。
ユーザー	メールサーバーに登録されている有効なユーザー名。
送信元	サーバーがメール送信時に使用する電子メールアドレス。
ホスト	メールサーバーのホスト名/IP アドレス。
ストアプロトコル	メッセージの取得に使用するプロトコル。
ストアプロトコルクラス	store-protocol の記憶領域サービスプロバイダの実装。store-protocol を実装するクラスの完全修飾クラス名。デフォルトのクラスは、`com.sun.mail.imap.IMAPStore` です。
転送プロトコル	メッセージの送信に使用するプロトコル。
転送プロトコルクラス	transport-protocol のトランスポートサービスプロバイダの実装。transport-protocol を実装するクラスの完全修飾クラス名。デフォルトのクラスは、`com.sun.mail.smtp.SMTPTransport` です。

注 – **CLI** の使用

メールリソースを作成するには、次のコマンドを実行します。

```
wadm> create-mail-resource --config=test --server-host=localhost
--mail-user=nobody --from=xyz@foo.com mail/Session
```

CLI リファレンスの `create-mail-resource`(1) を参照してください。

SOAP 認証プロバイダの構成

Java Authentication Service Provider Interface for Containers 仕様は、認証メカニズムプロバイダをコンテナに統合できるようにするための標準サービスプロバイダインタフェースを定義します。管理コンソールを使えば、新しい SOAP 認証プロバイダを追加できます。

▼ SOAP 認証プロバイダを追加する

1 構成を選択します。

構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「**Java**」>「認証」タブをクリックします。

3 「**SOAP** 認証プロバイダ」セクションの「新規」ボタンをクリックします。

SOAP 認証プロバイダのパラメータ

次の表では、「新規 SOAP 認証プロバイダ」ページで利用可能なパラメータについて説明します。

表 **11–6** SOAP 認証プロバイダのパラメータ

パラメータ	説明
名前	新しい SOAP 認証プロバイダの簡易名を入力します。
クラス名	プロバイダを実装するクラス名。`javax.security.auth.XXX` を実装するクラスの完全修飾クラス名
要求認証の送信元	この属性は、ユーザー名/パスワードなどのメッセージ層送信側認証、要求メッセージに適用すべきデジタル署名などのコンテンツ認証、のいずれかの要件を定義します。値 (auth-policy) は「送信側」、「コンテンツ」のいずれかです。この引数が指定されない場合、要求のソース認証は必須ではありません。
要求認証の受信先	この属性は、XML 暗号化などによる、送信側に対するメッセージ受信側のメッセージ層認証の要件を定義します。値は「コンテンツの前」「コンテンツのあと」のいずれかです。
応答認証の送信元	この属性は、ユーザー名/パスワードなどのメッセージ層送信側認証、応答メッセージに適用すべきデジタル署名などのコンテンツ認証、のいずれかの要件を定義します。値 (auth-policy) は「送信側」、「コンテンツ」のいずれかです。この引数が指定されない場合、応答のソース認証は必須ではありません
応答認証の受取先	この属性は、XML 暗号化などによる、送信側に対する応答メッセージ受信側のメッセージ層認証の要件を定義します。
プロパティー	「プロパティーを追加」ボタンをクリックしてその他の CLI プロパティーを入力します。

注 – **CLI** の使用

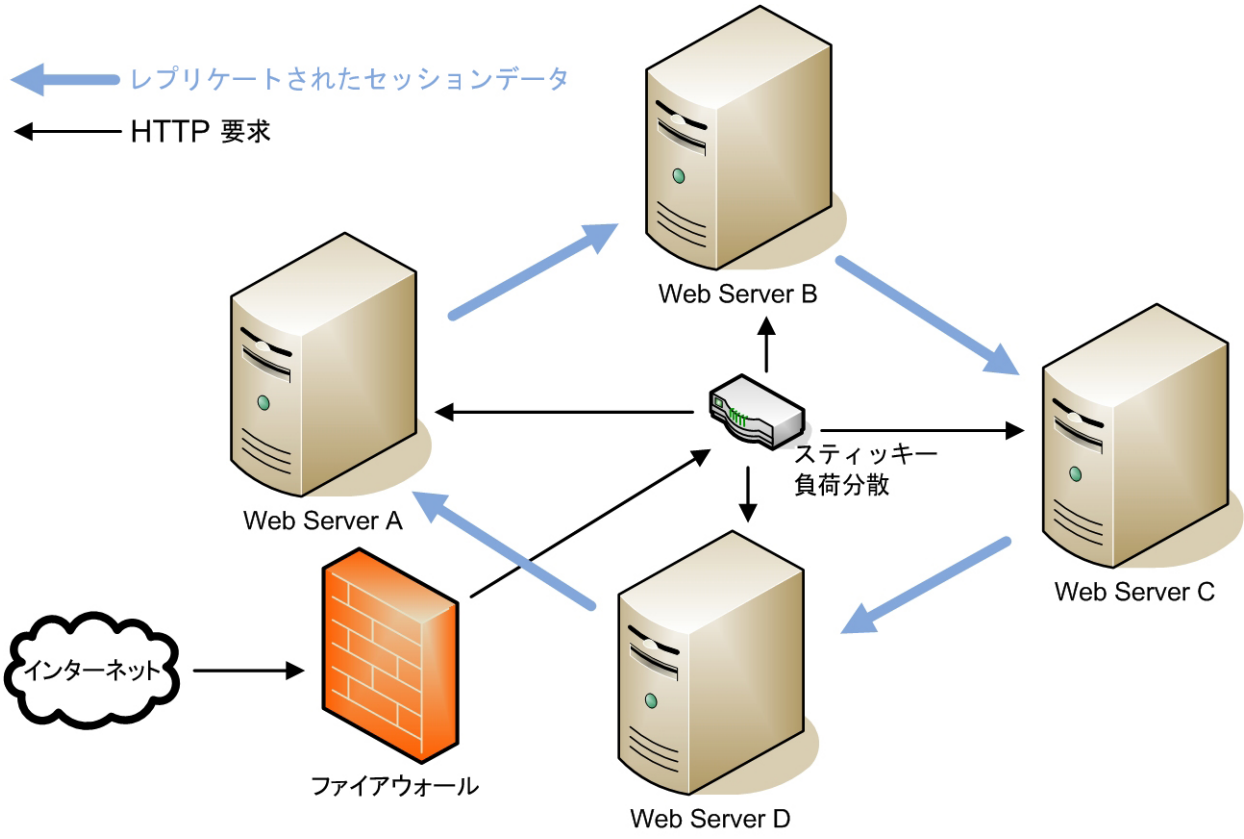
CLI を使って SOAP 認証プロバイダを追加するには、次のコマンドを実行します。

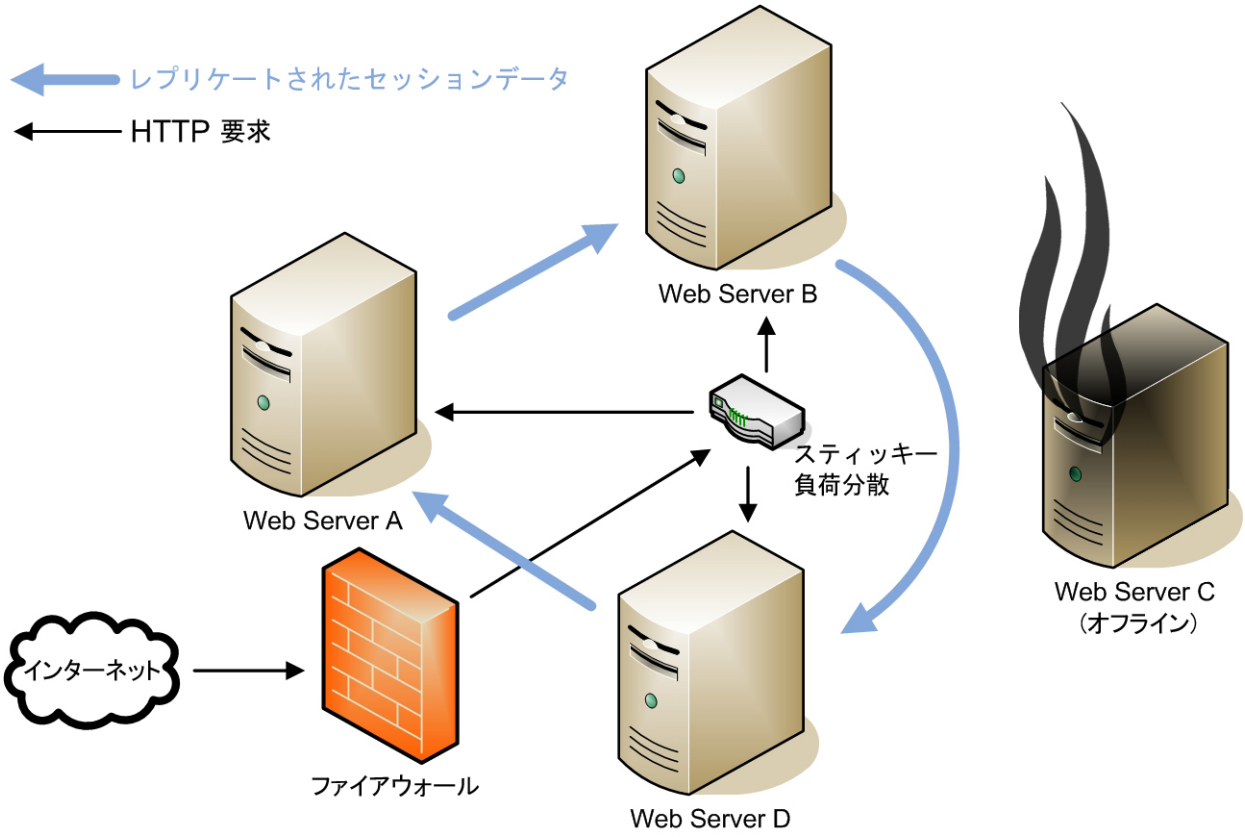
```
wadm> create-soap-auth-provider --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1
--class-name=javax.security.auth.soapauthprovider soap-auth
```

CLI リファレンスの `create-soap-auth-provider`(1) を参照してください。

セッションレプリケーションの構成

Sun Java System Web Server 7.0 は、Web アプリケーションに高可用性を提供するセッションレプリケーションをサポートしています。セッションレプリケーションは、あるインスタンスから同じクラスタ内の別のサーバーインスタンスに HTTP セッションをレプリケートすることで、これを実現します。したがって、どの HTTP セッションもリモートインスタンス上にバックアップコピーを持ちます。クラスタ内のあるインスタンスが利用不能になるような障害が発生した場合でも、クラスタはセッションの継続性を維持します。


レプリケートされたセッションデータ
HTTP 要求
Web Server B
Web Server A
スティッキー
負荷分散
Web Server C
インターネット
ファイアウォール
Web Server D



上図は、逆プロキシが設定された状態で 4 つのノード間でセッションレプリケーションが行われるという、典型的なシナリオを示したものです。Web Server C がオフラインになると、Web Server B から Web Server D にセッションデータがレプリケートされます。

セッションレプリケーションの設定

この節では、選択された構成のセッションレプリケーションのプロパティーを設定する手順について説明します。

▼ セッションレプリケーションを設定する

1 構成を選択します。

構成のリストから構成を選択します。リストを取得するには「構成」タブをクリックします。

2 「**Java**」>「セッションレプリケーション」をクリックします。

セッションレプリケーションのパラメータの変更

次の表では、セッションレプリケーションのページで利用可能なパラメータについて説明します。

表 11–7 セッションレプリケーションのパラメータ

パラメータ	説明
ポート	管理サーバーが待機するポート番号。デフォルトのポート番号は、8888 です。
有効	選択された構成のセッションレプリケーションを有効にします。
暗号化	レプリケーションの前にセッションデータを暗号化するかどうか。デフォルト値は false です。
暗号化方式	クラスタメンバーがセッションデータのレプリケートに使用する暗号化方式群 (アルゴリズム、モード、パディング)。
Getatrribute でレプリケーションをトリガー	`HttpSession.getAttribute` メソッドの呼び出し時にセッションをバックアップすべきかどうか。デフォルト値は true です。
レプリカ検出の最大数	セッションのバックアップの検索を試みる間接続する必要があるインスタンスの最大数。値の範囲は 1 から 2147.0483647.0 までです。ただし、制限なしの場合は -1 を使用します。
起動時検出のタイムアウト	インスタンスが指定されたバックアップインスタンスへの接続を試みる最大時間 (秒)。値の範囲は 0.001 から 3600 までです。
Cookie 名	セッションを所有するインスタンスを追跡する Cookie の名前を入力します。

認証レルムの管理

Java EE ベースのセキュリティーモデルは、ユーザーを識別および認証するセキュリティーレルムをサポートします。

認証プロセスは、Java レルムによってユーザーを確認します。レルムは、ユーザーのセット、オプションのグループマッピング、および認証要求を検証できる認証ロジックで構成されます。構成されたレルムと確立されたセキュリティーコンテキストによって認証要求が検証されたあと、このアイデンティティーが以降のすべての認証決定に適用されます。

注 – Java レルムは auth-db (認証データベース) に似ていますが、auth-db が ACL エンジンによって (ACL ファイル内の規則に基づいて) 使用されるのに対し、Java レルムは (各 Web アプリケーションの web.xml ファイル内に指定された) Java サーブレットアクセス制御規則によって使用される、という点が異なります。

サーバーインスタンスは任意の数の構成済みレルムを持つことができます。構成情報は、`server.xml` ファイルの `auth-realm` 要素内に存在します。

次の表では、Sun Java System Web Server 7.0 でサポートされているさまざまなタイプのレルムを定義します

表 11–8 レルムのタイプ

レルム	説明
ファイル	`file` レルムは、Sun Java System Web Server を初めてインストールしたときのデフォルトのレルムです。このレルムは設定が簡単かつ単純であり、開発者に多大な利便性を提供します。 `file` レルムは、テキストファイルに格納されたユーザーデータに基づいて、ユーザーを認証します。Java レルムは auth-db (認証データベース) に似ていますが、auth-db が ACL エンジンによって (ACL ファイル内の規則に基づいて) 使用されるのに対し、Java レルムは (各 Web アプリケーションの web.xml 内に指定された) Java サーブレットアクセス制御規則によって使用される、という点が異なります。
LDAP	`ldap` レルムを使えば、LDAP データベースをユーザーのセキュリティー情報用として使用できます。LDAP ディレクトリサービスは、一意の識別子を持つ属性のコレクションです。`ldap` レルムは、本稼働システムへの配備に最適です。 `ldap` レルムに基づいてユーザーを認証するには、1 人以上の必要なユーザーを LDAP ディレクトリに作成する必要があります。これは、管理サーバーの「ユーザー」および「グループ」タブから行えます。このアクションは、LDAP ディレクトリ製品のユーザー管理コンソールからも行えます。
PAM	PAM レルム (Solaris レルムとも呼ばれる) は、Solaris PAM スタックに認証を委任します。このレルムは PAM auth-db の場合と同じく Solaris 9 と 10 でしかサポートされておらず、また、サーバーインスタンスを root で実行する必要があります。

表 11–8 レルムのタイプ (続き)

レルム	説明
証明書	certificate レルムは、SSL 認証をサポートしています。証明書レルムは、Sun Java System Web Server のセキュリティーコンテキスト内にユーザーのアイデンティティーを設定し、クライアント証明書に含まれるユーザーデータをそのアイデンティティーに設定します。その後、Java EE コンテナが、証明書に含まれる各ユーザーの DN に基づいて承認処理を行います。このレルムは、X.509 証明書による SSL または TLS クライアント認証を使って、ユーザーの認証を行います。
ネイティブ	native レルムは、ACL ベースのコア認証モデルと Java EE/サーブレット認証モデルを連結する役割を果たす、特殊なレルムです。Java Web アプリケーションでネイティブレルムを使用すれば、Java Web コンテナに認証を実行させる代わりに ACL サブシステムに認証を実行させながらも、Java Web アプリケーションからそのアイデンティティーを利用できるようにする、といったことが可能になります。 認証処理が呼び出されると、ネイティブレルムはその認証をコア認証サブシステムに委任します。これは、ユーザーの立場から見れば、LDAP レルムが構成済み LDAP サーバーに認証を委任するのと、本質的には同じことです。グループメンバーシップクエリーがネイティブレルムによって処理される場合、その処理もコア認証サブシステムに委任されます。Java Web モジュールや開発者の立場から見れば、ネイティブレルムは、Web モジュールで利用可能なほかの Java レルムと何の違いもありません。
カスタム	ユーザー固有の要求に対応するため、プラグイン可能な JAAS ログインモジュールとレルム実装を使用することで、Oracle など、その他のデータベース用のレルムを構築することができます。

次の節では、新しい認証レルムを追加する場合の手順について説明します。

▼ 認証レルムを追加する

1 構成を選択します。
新しい認証レルムを追加する必要のある構成を選択します。「構成」タブをクリックし、構成を選択します。

2 「**Java**」>「認証」タブをクリックします。

3 「新規」ボタンをクリックします。

4 レルムの詳細を入力します。

- 名前 — レルムの簡易名を入力します。この名前は、web.xml などからこのレルムを参照する場合に使用されます。
- クラス — カスタムレルムを構成する場合に、そのカスタムレルムを実装する完全な Java クラス名を入力します。組み込みレルムの場合、クラスを入力する必要はありません。
- タイプ — レルムのタイプを選択します。Java レルムのタイプについて説明した前の節を参照してください。
- プロパティー — レルム固有のプロパティーを追加します。例: `property name="file" value="instance_dir/config/keyfile" and property name="jaas-context" value="fileRealm`。

注 – **CLI** の使用

CLI 経由で認証レルムを追加するには、次のコマンドを実行します。

```
wadm> create-auth-realm --user=admin --password-file=admin.pwd --host=serverhost
--port=8989 --config=config1 basic
```

CLI リファレンスの `create-auth-realm`(1) を参照してください。

組み込み認証レルムのタイプ名を指定します。タイプは `file`、`ldap`、`pam`、`native`、`certificate` のいずれかです。

◆ ◆ ◆ 第 12 章

検索コレクションの操作

サーバーに含まれている検索機能を使用すると、ユーザーはサーバー上のドキュメントを検索して、結果を Web ページに表示できます。サーバー管理者は、検索対象となるドキュメントのインデックス (コレクションと呼ばれる) を作成し、ユーザーの要求に合うように検索インタフェースをカスタマイズできます。

検索コレクションのクエリー方法の詳細については、「検索機能のオンラインヘルプ」を参照してください。

検索について

検索機能は、Sun Java System Web Server のインストール時にほかの Web コンポーネントとともにインストールされます。検索の構成や管理は、サーバーインスタンスのレベルではなく仮想サーバーのレベルで行います。

管理コンソールの「仮想サーバー」タブの下にある「検索」タブから、次のことが行えます。

- 検索機能を有効および無効にする
- 検索コレクションの作成、変更、削除、およびインデックスの再作成を行う
- 検索コレクションのスケジュールされた保守タスクを作成、変更、および削除する

管理インタフェースから得られる情報は、*<server-root>*/config/server.xml ファイル内に格納されており、VS 要素内にマップされています。

サーバー管理者は、検索クエリーおよび検索結果ページをカスタマイズできます。これには、企業のロゴを使ってページのブランドを変更したり、検索結果の表示方法を変更したりすることが含まれます。以前のリリースでは、パターンファイルを使用することでこの機能が実現されていました。

検索には、グローバルな「オン/オフ」機能はありません。代わりに、デフォルトの検索 Web アプリケーションが提供されており、このアプリケーションを特定の仮想サーバー上で有効または無効にするようになっています。この検索アプリケーションは、コレクションのクエリーと結果の表示に使用する基本的な Web ページを提供します。この検索アプリケーションには、検索タグライブラリを使ってカスタマイズされた検索インタフェースを構築する方法を示すサンプル JSP が含まれています。

注 – Sun Java System Web Server では、検索結果に対するアクセスチェック機能は提供されていません。使用される可能性のあるセキュリティーモデルやレルムが多岐にわたるため、セキュリティーチェックの実行と結果のフィルタリングを、検索アプリケーション内から行うことは不可能です。適切なセキュリティー機構を使ってコンテンツを確実に保護することは、サーバー管理者の責任です。

検索プロパティーの構成

仮想サーバーの検索を有効にするには、サーバーに含まれている検索アプリケーションを有効にします。

注 – 検索を有効にするには、Java Web コンテナが有効になっている必要があります。

構成する仮想サーバーの Java が有効になっているのを確認したあとで、次の手順を実行して検索を有効にします。

1. 「構成」タブをクリックします。
2. 構成のリストから構成を選択します。
3. 「仮想サーバー」タブをクリックします。
4. 仮想サーバーのリストから仮想サーバーを選択します。
5. 「検索」タブをクリックします。
6. 「検索アプリケーション」セクションで「有効」チェックボックスをクリックすることで、検索アプリケーションを有効にします。

その他の構成可能なパラメータは、次のとおりです。

- **URI**: カスタム検索アプリケーションを使用する予定である場合は、その URI を入力します。デフォルトの検索アプリケーションを使用する場合、ここに値を指定する必要はありません。
- 最大ヒット数: 1 つの検索クエリーで取得される結果の最大数を指定します。
- 有効: デフォルトの検索アプリケーションを有効にするには、これをチェックします。

注 – **CLI** の使用

CLI 経由で検索プロパティーを設定するには、CLI で次のコマンドを実行します。

```
wadm> set-search-prop --user=admin --password-file=admin.pwd --host=serverhost
--port=8888 --no-ssl --rcfile=null --config=config1 --vs=config1_vs_1
enabled=true max-hits=1200
```

CLI リファレンスの `set-search-prop(1)` を参照してください。

検索コレクションの構成

検索を行うには、その検索対象となる検索可能データのデータベースが必要となります。このコレクションと呼ばれるデータベースはサーバー管理者によって作成されます。コレクションは、サーバー上のドキュメントに関する情報のインデックスを作成し、それを格納します。サーバー管理者がサーバーのドキュメントの全部または一部のインデックスを作成し終わると、タイトル、作成日付、作成者などの情報を検索できるようになります。

注 – 検索コレクションについて

- コレクションは管理対象の仮想サーバーに固有である
- 仮想サーバーから可視のドキュメントのみが、管理インタフェースに表示され、インデックス作成可能となる
- サーバー上に存在できるコレクションの数に制限はない
- 検索コレクション内のドキュメントは 1 つの文字エンコーディングに固有のものではない。つまり、検索コレクションは、複数のエンコーディングに関連付けることができる。

サポートする形式

インデックス作成および検索が可能なファイルの形式は、次のとおりです。

1. HTML ドキュメント — .html および .htm
2. ASCII プレーンテキスト — .txt

3. PDF

検索コレクションの追加

新しいコレクションを追加するには、次のタスクを実行します。

1. 「構成」タブをクリックします。
2. 構成のリストから構成を選択します。
3. 「仮想サーバー」タブをクリックします。
4. 仮想サーバーのリストから仮想サーバーを選択します。
5. 「検索」タブをクリックします。
6. 「検索コレクション」セクションで「ドキュメントの追加」ボタンをクリックすることで、新しい検索コレクションを追加します。

次の節では、新しい検索コレクションを作成するためのページに含まれるフィールドについて説明します。

1. 検索コレクションの情報を入力します
   a. コレクション名 — 検索コレクションの一意名を入力します。

   注 – 複数バイト文字はコレクション名として許可されません。

   b. 表示名 — (省略可能) これが検索クエリーページのコレクション名として表示されます。表示名を指定しないと、コレクション名が表示名として使用されます。
   c. 説明 — (省略可能) 新しいコレクションを説明するテキストを入力します。
   d. パス — デフォルトの場所にコレクションを作成することも、コレクションの格納先となる有効なパスを入力することもできます。
2. インデックス作成の情報を入力します
   a. インデックスを作成するディレクトリ — インデックスを作成してコレクション内に格納するドキュメントが入っているディレクトリを入力します。インデックスを作成できるのは、この仮想サーバーから可視のディレクトリだけです。
   b. サブディレクトリ — インデックスを作成してコレクション内に格納するドキュメントが入っているサブディレクトリを入力します。サブディレクトリのパスは、上で指定したディレクトリパスからの相対パスにすべきです。
   c. パターン — インデックスを作成するファイルを選択するためのワイルドカードを指定します。

ワイルドカードパターンは、特定のファイルだけが確実にインデックス作成されるよう、適切な使い方をしてください。たとえば、*.* と指定すると、実行可能ファイルや Perl スクリプトまでインデックス作成されてしまいます。

d. サブディレクトリ — 有効/無効。このオプションを選択すると、選択されたディレクトリのサブディレクトリ内のドキュメントもインデックス作成されます。これがデフォルトの動作です。

e. デフォルトエンコーディング —

コレクション内のドキュメントは、単一の言語やエンコーディングに制限されません。ドキュメントを追加するたびに、ある単一のエンコーディングのみを指定できます。しかし、次回ドキュメントをコレクションに追加するときには、異なるデフォルトエンコーディングを選択できます。

3. 手順 **3:**概要を表示します

a. 概要を表示したあと、「完了」ボタンをクリックして新しいコレクションを追加します。

注 – **CLI** の使用

CLI 経由で検索コレクションを追加するには、次のコマンドを実行します。

```
wadm> create-search-collection --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1 --uri=/search_config1
--document-root=../docs searchcoll
```

CLI リファレンスの `create-search-collection`(1) を参照してください。

検索コレクションの削除

検索コレクションを削除するには、次のタスクを実行します。

1. 「構成」タブをクリックします。
2. 構成のリストから構成を選択します。
3. 「仮想サーバー」タブをクリックします。
4. 仮想サーバーのリストから仮想サーバーを選択します。
5. 「検索」タブをクリックします。
6. 「検索コレクション」セクションでコレクション名を選択し、「削除」ボタンをクリックします。そのコレクションが削除されます。

注 – **CLI** の使用

CLI 経由で検索コレクションを削除するには、次のコマンドを実行します。

```
wadm> delete-search-collection --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 --config=config1 --vs=config1_vs_1 searchcoll
```

CLI リファレンスの `delete-search-collection`(1) を参照してください。

コレクション更新のスケジュール

検索コレクション上で定期的に実行する保守タスクをスケジュールすることができます。スケジュールできるタスクにはインデックスの再作成と更新があります。特定のコレクションのタスクをスケジュールするには管理インタフェースを使用します。指定できる情報は次のとおりです。

- 実行するタスク (インデックスの再作成、更新のいずれか)
- タスクを実行する時間
- タスクを実行する曜日

コレクションのイベントをスケジュールするには、次のタスクを実行します。

1. 「構成」タブをクリックします。
2. 構成のリストから構成を選択します。
3. 「仮想サーバー」タブをクリックします。
4. 仮想サーバーのリストから仮想サーバーを選択します。
5. 「検索」タブをクリックします。
6. 「スケジュールされたイベント」タブをクリックします。
7. 「検索イベント」タブで「新規」ボタンをクリックします。

次の表では、「新規検索イベントスケジュール」ページのフィールドについて説明します。

表 12–1 フィールド説明 > 新規検索イベントスケジュール

フィールド	説明
コレクション	保守をスケジュールするコレクションをドロップダウンリストから選択します。
イベント	1. コレクションのインデックスを再作成 — このスケジュールされたイベントは、指定された時刻に指定されたコレクションのインデックスを再作成します。 2. コレクションを更新 — コレクションの作成後にファイルを追加または削除できます。ドキュメントの追加は、コレクション作成時に指定されたディレクトリの下でしか行えません。ドキュメントを削除する場合、そのファイルとメタデータのエントリだけがコレクションから削除されます。実際のファイル自体がファイルシステムから削除されるわけではありません。このスケジュールされたイベントは、指定された時刻にコレクションを更新します。 3. パターン — インデックスを作成するファイルを選択するためのワイルドカードを指定します。 4. 含まれているサブディレクトリ — このオプションを選択すると、選択されたディレクトリのサブディレクトリ内のドキュメントもインデックス作成されます。これがデフォルトの動作です。 5. エンコーディング — インデックスを作成するドキュメントの文字エンコーディングを指定します。デフォルトは ISO-8859-1 です。インデックス作成エンジンは、**HTML** ドキュメントのエンコーディングを、そこに組み込まれているメタタグに基づいて決定しようとします。これが指定されていない場合にデフォルトエンコーディングが使用されます。
時間	構成されたイベント起動時刻。ドロップダウンボックスから時間と分の値を選択します。 毎日 — 指定されたイベントを指定された時刻に毎日起動します。 指定日 — 指定されたイベントを指定された日に起動します。 1. 曜日 — 日曜から土曜までの間で、任意の曜日を指定します。 2. 日付 — **1** から **31** までの間で、任意の日付をコンマ区切りエントリとして指定します。例: **4,23,9** 指定月 — 指定された月の指定された時刻に指定されたイベントを起動します。**1** 月から **12** 月までの間で、月を指定します。
間隔	この期間後に指定されたイベントを起動します。 1. 時間おき — ドロップダウンボックスから時間数を選択します。 2. 秒おき — ドロップダウンリストから秒数を選択します。

検索の実行

ユーザーの主な関心事は、検索コレクション内のデータについて質問し、その応答としてドキュメントのリストを取得することです。Sun Java System Web Server にインストールされている検索 Web アプリケーションは、デフォルトの検索クエリーおよび検索結果ページを提供します。それらのページは、そのまま使用することも、「検索ページのカスタマイズ」で説明する一連の JSP タグを使ってカスタマイズすることもできます。

ユーザーは、サーバー管理者によって作成されたコレクションに対して検索を行います。ユーザーは次のことを行えます。

- 検索する一連のキーワードと省略可能なクエリー演算子を入力する
- 仮想サーバーから可視のコレクション内でのみ検索する
- 仮想サーバーから可視の単一コレクションに対して、あるいは複数のコレクションをまたがって、検索を行う

サーバー管理者は、仮想サーバーの検索クエリーページへのアクセスに必要な URL を、ユーザーに知らせる必要があります。

検索ページ

エンドユーザーが検索機能にアクセスするのに使用できるデフォルトの URL は、次のとおりです。

`http://<`*server-instance*`>:`*port number*`/search`

例:

```
http://plaza:8080/search
```

エンドユーザーがこの URL を呼び出すと、Java Web アプリケーションである「検索」ページが開きます。

注 – キーワード、およびオプションのクエリ演算子を含む、基本および高度な検索を実行する詳細な手順については、検索エンジンに付属のオンラインヘルプを参照してください。これらの情報にアクセスするには、「検索」ページの「ヘルプ」リンクをクリックしてください。

クエリーの実行

あるコレクションに対して検索を行うには、検索クエリーページを使用します。一連のキーワードと省略可能なクエリー演算子を入力すると、ブラウザに表示されたWebページ上にその結果が表示されます。結果ページには、検索条件に一致するサーバー上のドキュメントへのリンクが含まれます。

注–サーバー管理者は、「検索ページのカスタマイズ」で説明した方法でこの検索クエリーページをカスタマイズできます。

クエリーを行うには、次の手順を実行します。

▼ クエリーの実行

1 ブラウザのロケーションバーに次の形式で**URL**を入力することで、検索**Web**アプリケーションにアクセスします。

`http://<server-instance>:port number /search`

2 表示された検索クエリーページの「検索対象」フィールドで、検索するコレクションを表すチェックボックスをオンにします。

3 関連のある**Web**ページのリストを得るために、クエリーを記述するいくつかの単語を入力したあと、「**Enter**」キーを押すか「検索」ボタンをクリックします。

検索時により細かい調整を行いたい場合は、「詳細検索」ページで提供される検索パラメータを使用できます。このページについては次の節で説明します。

詳細検索

キーワードを細かく調整する演算子を追加すれば、検索精度を高めることができます。これらのオプションは「詳細検索」ページから選択できます。

詳細検索クエリーを行うには、次の手順を実行します。

▼ 詳細検索クエリーを行う

1 ブラウザのロケーションバーに次の形式で**URL**を入力することで、検索**Web**アプリケーションにアクセスします。

`http://<server-instance>:port number /search`

2. 「詳細」リンクをクリックします。
3. 次の情報のいずれかまたはすべてを入力します。
   - 検索対象: 検索対象となるコレクションを選択します。
     - 検索: 次の 4 つのオプションがサポートされています。
       - すべての単語に一致: 「検索」に指定されたすべてのキーワードを含むページを検索します。
       - いずれかの単語に一致: 「検索」に指定されたいずれかのキーワードを含むページを検索します。
       - 完全に一致: 「検索」に指定された語句に完全に一致するページを検索します。
       - パセッージ検索: 取得されたページ内でキーワードまたは単語を含むパセッジを強調表示します。

       対象外の単語: 検索時に、指定された単語を含む Web ページが除外されます。
   - タイトルに「含む/含まない」: 指定されたキーワードを含むタイトルを持つページに、検索を制限します。
   - 検索対象期間: 選択された期間内にインデックス作成された Web ページに、検索処理を制限します。

ドキュメントフィールド

Sun JavaTMSystem Web Server はドキュメントのインデックスを維持します。インデックスには各ドキュメントのエントリが含まれます。各インデックスエントリには、Title、Author、URL などのフィールドが 1 つ以上含まれます。クエリーを特定のドキュメントフィールドに制限することができますが、その場合、その指定されたフィールドで条件に一致するドキュメントのみが検出されます。

たとえば、Einstein を単純に検索した場合、Title、Author、Keywords のいずれかのフィールドに単語 Einstein を含むドキュメントのすべてが検出されます。これには、Einstein に関するドキュメント、Einstein を参照しているドキュメント、Einstein によって書かれたドキュメントなどが含まれます。これに対し、`Author = "Albert Einstein"` と指定した場合、Albert Einstein によって書かれたドキュメントのみが検出されます。

デフォルトで検索可能なインデックスフィールドは、次のとおりです。

1. **Author** — `<author>` メタタグに指定された、ドキュメントを作成した著者、一連の著者、または組織。
2. **Keywords** — `<keywords>` メタタグに指定されたキーワード。
3. **Date** — このドキュメントが最後に編集または変更された日付。

4. **Title** — HTML の <title> タグに指定された、ドキュメントのタイトル。

検索クエリー演算子

検索クエリー演算子の詳細な一覧については、『*Administration Console Search Online Help*』を参照してください。

検索結果の表示

検索結果はユーザーのブラウザ内の Web ページ上に表示されます。このページには、検索条件に一致するサーバー上のドキュメントへの HTML ハイパーリンクが含まれます。各ページにはデフォルトで 10 レコード (ヒット) が表示され、それらは関連性に基づいて降順でソートされます。 各レコードには、ファイル名、サイズ、作成日付などの情報が表示されます。さらに、一致した単語が強調表示されます。

検索ページのカスタマイズ

Sun Java System Web Server には、基本的な検索クエリーおよび検索結果ページを提供するデフォルトの検索アプリケーションが含まれています。これらの Web ページは、そのまま使用することも、ユーザーの特定の要求に合うようにカスタマイズすることもできます。そのようなカスタマイズには、別のロゴを使って Web ページのブランドを変更するような単純なものから、検索結果の表示順を変更するような複雑なものまであります。

デフォルトの検索アプリケーションは、検索タグライブラリを使ってカスタマイズされた検索インタフェースを構築する方法を示すサンプル JSP を提供しています。 `/bin/https/webapps/search` に格納されているデフォルト検索アプリケーションを、カスタマイズ可能な検索タグの使用方法を示すサンプルアプリケーションとして参照することができます。

デフォルト検索インタフェースの主要コンポーネントは 4 つあります。ヘッダー、フッター、クエリーフォーム、および結果です。

これらの基本要素は、タグの属性の値を変更するだけで簡単にカスタマイズすることができます。より細かなカスタマイズは、タグライブラリを使用することで実現できます。

検索インタフェースのコンポーネント

検索インタフェースは次のコンポーネントから構成されています。

ヘッダー

ヘッダーにはロゴ、タイトル、および簡略説明が含まれます。

フッター

フッターには著作権情報が含まれます。

フォーム

クエリーフォームには、検索コレクションを表す一連のチェックボックス、クエリー入力ボックス、および送信ボタンとヘルプボタンが含まれます。

結果

結果はデフォルトで、1 ページに 10 レコードずつ表示されます。レコードごとに、タイトル、パセッジ、サイズ、作成日付、URL などの情報が表示されます。パセッジとはページの短い断片のことですが、そこでは一致した単語が強調表示されます。

検索クエリーページのカスタマイズ

クエリーフォームには、検索コレクションの一連のチェックボックス、クエリー入力ボックス、および送信ボタンが含まれます。フォームは、`<s1ws:form>` タグに加えて `<collElem>`、`<queryBox>`、`<submitButton>` の各タグ、およびそれらのデフォルト値を使って作成されます。

```
<s1ws:form>
    <s1ws:collElem>
    <s1ws:queryBox> <s1ws:submitButton>
</s1ws:form>
```

クエリーフォームは、ページの中央やサイドバー上など、ページ内の任意の場所に配置できます。また、コレクション選択ボックス、クエリー文字列入力ボックス、送信ボタンが水平方向に一直線に並ぶクロスバーや、コレクションがチェックボックスとして表示され、その下にクエリー入力ボックスと送信ボタンが配置されるブロックなど、さまざまな形式で表示することもできます。

次の各例は、`<searchForm>` タグセットを使ってさまざまな形式のクエリーフォームを作成する方法を示しています。

水平バー

次のサンプルコードは、すべてのコレクションの選択ボックス、クエリー入力ボックス、および送信ボタンがすべて 1 行に配置されたフォームを作成します。

```
<s1ws:form>
    <table cellspacing="0" cellpadding="3" border="0">
    <tr class="navBar">
        <td class="navBar"><s1ws:collElem type=” select” ></td>
        <td class="navBar">
            <s1ws:querybox size="30">
            <s1ws:submitButton class="navBar" style="padding: 0px; margin: 0px; width: 50px">
        </td>
    </tr>
    </table>
</s1ws:form>
```

サイドバーブロック

フォーム要素がサイドバーに配置され、サイドバーのほかの項目と同じ形式のタイトル「Search」を含むようなフォームブロックを作成できます。そのような配置の効果は、次の図に示すようになります。

フォーム要素がサイドバーに配置された、カスタマイズ後のクエリーページ

次に示すサンプルコードでは、フォームの本体部分に、選択可能な検索コレクションを表示するチェックボックスが3つ含まれており、それらは1列に並んでいます。クエリー入力ボックスと送信ボタンはその下に配置されています。

```
<s1ws:searchForm>
    <table>
<!--... ほかのサイドバー項目 ... -->
    <tr class="Title"><td>Search</td></tr>
    <tr class="Body">
        <td>
        <table cellspacing="0" cellpadding="3" border="0">
        <tr class="formBlock">
            <td class="formBlock"> <s1ws:collElem type="checkbox" cols="1" values="1,0,1,0" /> </td>
        </tr>
        <tr class="formBlock">
            <td class="formBlock"> <s1ws:querybox size="15" maxlength="50"> </td>
        </tr>
        <tr class="formBlock">
            <td class="formBlock"> <s1ws:submitButton class="navBar" style="padding: 0px; margin: 0px; width: 50p
        </tr>
        </table>
        </td>
    </tr>
    </table>
</s1ws:searchForm>
```

検索結果ページのカスタマイズ

検索結果は次のようにして生成されます。

- <formAction> タグがすべてのフォーム要素の値を取得し、基本的な検証処理を行う。
- <search> タグ、<resultIteration> タグ、およびその他のタグが <formAction> タグの内側で発生し、すべてのフォーム要素の値にアクセスする。
- <search> タグが <formAction> からのクエリー文字列とコレクションを使って検索を実行し、その検索結果を pageContext に保存する。
- 続いて、<resultIteration> タグが結果セットを取得し、繰り返し処理を行う。

タグの属性値を変更するだけで、検索結果ページをカスタマイズできます。

次のサンプルコードは、まずタイトルバーを表示し、次に指定された件数のレコードを表示し、最後にナビゲーションバーを表示します。タイトルバーには、検索に使用されたクエリー文字列と、1– 10 などのような、返されたレコード合計数の範囲が含まれます。レコードセクションにはレコードごとに、ファイルへのリンクを含むタイトル、キーワードが強調表示された最大 3 つのパセッジ、URL、作成日付、およびドキュメントのサイズが表示されます。

このセクションの末尾にあるナビゲーションバーは、前ページと次ページへのリンク、および現在のページの前後にある 8 つの追加ページへの直接のリンクを提供します。

```
<s1ws:formAction />
<s1ws:formSubmission success="true" >
    <s1ws:search scope="page" />
    <!--search results-->
    (...html を省略...)
        <s1ws:resultStat formId="test" type="total" /></b> Results Found, Sorted by Relevance</span></td><td>
        <span class="body"><a href="/search/search.jsp?">Sort by Date</a></span></td>
        <td align="right"><span class="body">
        <s1ws:resultNav formId="test" type="previous" caption="<img border=0 src=\\"images/arrow-left.gif\\" alt=\\"Pr
         <s1ws:resultStat formId="test" type="range" />
         <s1ws:resultNav formId="test" type="next" caption="<img border=0 src=\\"images/arrow-right.gif\\" alt=\\
          <!img alt="Next" src="images/arrow-right.gif" border="0" WIDTH="13" HEIGHT="9">
            (...html を省略...)
        <table border=0>
        <s1ws:resultIteration formId="test" start="1" results="15">
            <tr class=body>
                <td valign=top>
                <s1ws:item property='number' />.  
                </td>
                <td>
                    <b><a href="<s1ws:item property='url' />"><s1ws:item property='title' /></a></b>
```

```
                    <br>
                    <s1ws:item property='passages' />
                    <font color="#999999" size="-2">
                    <s1ws:item property='url' /> -
                    <s1ws:item property='date' /> -
                    <s1ws:item property='size' /> KB
                    </font><br><br>
                </td>
            </tr>
        </s1ws:resultIteration>
        </table>
        (...html を省略...)
        <s1ws:resultNav formId="test" type="previous" />
        <s1ws:resultNav formId="test" type="full" offset="8" />
        <s1ws:resultNav formId="test" type="next" />
    (...html を省略...)
</s1ws:formSubmission>
```

次の図に、カスタマイズ後の検索結果ページを示します。

カスタマイズ後の検索結果ページ

この基本的な検索結果インタフェースは、タグの操作や HTML の変更を行うことで、簡単にカスタマイズすることができます。たとえば、ナビゲーションバーをコピーし、それを検索結果の前に配置することができます。また、検索レコードの任意のプロパティーの表示/非表示を選択することもできます。

<search>、<resultIterate>、およびその関連タグは、フォームと組み合わせて使用できるだけでなく、特定のトピックを一覧表示するために使用することもできます。次のサンプルコードは、あるサイト上で Java Web サービスに関する記事のトップ 10 を一覧表示します。

```
<s1ws:search collection="Articles" query="Java Web Services" />
<table cellspacing="0" cellpadding="3" border="0">
  <tr class="Title"><td>Java Web Services</td></tr>
</table>
<table cellspacing="0" cellpadding="3" border="0">
<s1ws:resultIteration>
<tr>
<td><a href="<s1ws:item property='URL' />"> <s1ws:item property='Title'/></a></td>
</tr>
</s1ws:resultIteration>
</table>
```

フォームと結果をカスタマイズしてそれぞれ独立したページに格納する

フォームページと結果ページをそれぞれ独立させる必要がある場合、<form> タグセットを使ってフォームページを作成し、<formAction> タグセットを使って結果ページを作成する必要があります。

スムーズなページフローを実現できるよう、フォームページへのリンクを結果ページに追加する必要があります。

タグ規約

次のタグ規約に注意してください。

- タグのクラスはパッケージ com.sun.web.search.taglibs に属する。
- pageContext のすべての属性は、プレフィックス com.sun.web を持つ。たとえば、検索結果に対する属性は com.sun.web.searchresults.*form_id* である。ここで、*form_id* はフォームの名前である。
- タグライブラリはプレフィックス s1ws を使って参照される。タグとその属性の名前は大文字と小文字の混在形式にする。具体的には、pageContext のように、名前に含まれる各単語の先頭文字を大文字にする。

タグの仕様

Sun Java System Web Server には、検索インタフェースの検索クエリーページと検索結果ページのカスタマイズに使用可能な一連の JSP タグが含まれています。

検索ページのカスタマイズに使用可能な JSP タグの完全な一覧については、『Sun Java System Web Server 7.0 *Developer's Guide to Web Applications*』を参照してください。

◆ ◆ ◆ 第 13 章

13 サーバーの監視

この節では、Sun Java System Web Server の監視機能について説明し、インスタンスレベル、構成レベルの両方で監視可能なサーバーパラメータの詳細なリストを提供します。

Sun Java System Web Server の監視機能

「構成」タブ、「インスタンス」タブのいずれかを「監視」親タブの下で選択すると、監視可能なサーバーパラメータが表示されます。

Sun Java System Web Server 管理コンソールから実行可能なアクションは、次のとおりです。

- インスタンスレベルおよび構成レベルのサーバー統計を表示する。
- 構成レベルで監視を有効/無効にする。
- インスタンスレベルのエラー/アクセスログを表示する。

構成レベルのサーバーパラメータを監視するには、「監視」>「構成」タブをクリックします。この表には、利用可能な構成が次の情報とともに一覧表示されます。

- ノード — その構成が配備されたノードの数。
- 要求数 — 受信された要求の、すべての仮想サーバーにわたる合計数。
- エラー — ログに記録されたエラーの、すべての仮想サーバーにわたる合計数。
- 応答時間 — すべての仮想サーバーのなかでの最大応答時間。

構成レベルの統計を取得するには、構成名をクリックします。一般統計は次の 3 つのタイプに分けられます。

- 要求の統計
- エラーの統計
- 応答時間の統計

管理コンソール経由での監視

管理コンソールでは、サーバー統計を次のカテゴリ別に表示できます。

- 一般統計。
- インスタンスの統計。
- 仮想サーバーの統計。

表 13–1 監視カテゴリ

カテゴリ	説明
一般統計	「一般統計」には、構成の要求、エラー、および応答に関する全体的な統計が表示されます。
インスタンスの統計	「インスタンスの統計」には、インスタンスの要求、エラー、および応答に関する全体的な統計に加え、サーバークラッシュ数および仮想サーバー数の情報も表示されます。
仮想サーバーの統計	「仮想サーバーの統計」には、仮想サーバーの要求、エラー、および応答に関する全体的な統計に加え、開いている接続の数および送信/受信合計バイト数の情報も表示されます。

▼ 統計情報の表示

1. 「監視」タブをクリックします。
2. リストから構成を選択します。
3. 「一般統計」、「インスタンスの統計」、および「仮想サーバーの統計」を表示します。

注 – **CLI** の使用

サーバーの監視は、`get-config-stats`、`get-virtual-serevr-stats`、`get-webapp-stats`、および `get-servlet-stats` コマンドを使って行えます。

- `wadm> get-config-stats --user=admin --password-file=admin.passwd --host=localhost --port=8989 --config=test --node=cat.test.com --ssl=true`

  このコマンドは、指定されたインスタンスの統計を取得します。構成レベルの統計を取得するには、`--node` オプションを指定しないでこのコマンドを使用します。

- `wadm> get-vs-stats --user=admin --password-file=admin.passwd --host=localhost --port=8989 --config=test --vs=www.test.com --node=cat.test.com --ssl=true`

  このコマンドは、ある特定の構成が配備されたすべてのノードにわたる、その構成の集積的な仮想サーバー統計を取得します。ある特定のノードに配備された構成の統計を取得するには、`--node` オプションを使用します。

- `wadm> get-webapp-stats --user=admin --password-file=admin.passwd --host=localhost --port=8989 --config=test --node=cat.test.com --vs=www.test.com --uri=/foo --ssl=true`

  このコマンドは、指定されたインスタンスの指定された仮想サーバー上に配備された、ある特定の Web アプリケーションの統計を取得します。ある特定の構成が配備されたすべてのノードにわたる、その構成の集積的な Web アプリケーション統計を取得するには、`--node` オプションを指定しないでこのコマンドを使用します。

- `wadm> get-servlet-stats --user=admin --password-file=admin.pwd --host=localhost --port=8989 --config=test --node=cat.test.com --vs=www.test.com --uri=/servlet-simple --ssl=true`

  このコマンドは、サーブレット `servlet-simple` の統計を取得します。

監視パラメータの変更

サーバーは、SNMP 経由で監視アクションを実行します。SNMP は、ネットワークアクティビティーに関するデータを交換するために使用されるプロトコルです。SNMP では、管理対象デバイスとネットワーク管理ステーション (NMS) との間でデータが転送されます。管理対象デバイスは、SNMP が実行されている任意のデバイスです。つまり、ネットワーク上のホスト、ルーター、Web サーバー、その他のサーバーなどです。NMS は、そのネットワークをリモート管理する場合に使用するマシンです。NMS ソフトウェアは通常、収集されたデータを表示するグラフを提供したり、そのデータを使ってサーバーがある特定の許容範囲内で稼働しているか確認したりします。

NMS は通常、1 つ以上のネットワーク管理アプリケーションがインストールされた強力なワークステーションです。HP OpenView のようなネットワーク管理アプリケーションでは、Web サーバーなどの管理対象デバイスに関する情報がグラフィカルに表示されます。たとえば、社内のどのサーバーが稼動またはダウンしているかを表示したり、受け取ったエラーメッセージの数と種類を表示したりできます。SNMP と Sun Java System Web Server を組み合わせて使用する場合、サブエージェント、マスターエージェントの 2 種類のエージェントを使って、NMS とサーバーとの間でこの情報が転送されます。

サブエージェントはサーバーに関する情報を収集し、その情報をサーバーのマスターエージェントに渡します。管理サーバー以外の Sun Java System Web Server はすべて、サブエージェントを持ちます。

注 – SNMP 構成を変更した場合には、「保存」ボタンをクリックしてから SNMP サブエージェントを再起動する必要があります。

構成の設定を変更するには、次のタスクを実行します。

1. 「構成」タブをクリックします。
2. 監視設定を変更する必要のある構成を選択します。
3. 「監視設定」サブタブをクリックします。

監視パラメータの構成

構成の一般的な監視設定を変更するには、「一般設定」セクションの値を編集します。次の表に、一般的な監視パラメータのフィールド説明を示します。

表 13–2　フィールド説明 > 一般的な監視設定

フィールド	説明
SNMP サブエージェント	一般に、SNMP を使用するには、1 つのマスターエージェントと少なくとも 1 つのサブエージェントをシステム上にインストールし、それらを実行する必要があります。サブエージェントを有効にするには、まずマスターエージェントをインストールしておく必要があります。 SNMP サブエージェントの有効/無効を切り替える場合に、このオプションを選択します。

表 13–2 フィールド説明 > 一般的な監視設定 (続き)

フィールド	説明
間隔	ポーリング間隔は、表示される統計情報の更新間隔を示す秒数です。 サーバーインスタンスが稼動中で、統計を有効にしている場合、選択した統計情報の種類を示すページが表示されます。このページは、ポーリング間隔として選択した値に応じて、5 ～ 15 秒ごとに更新されます。
プロファイリング	統計/プロファイリング機能を使えば、サーバーの現在のアクティビティーを監視できます。統計情報は、サーバーが処理している要求数と、それらの要求の処理状況を示します。統計のなかには、個々の仮想サーバーに対して表示可能なものもあれば、サーバーインスタンス全体に対して表示可能なものもあります。 プロファイリングの有効/無効を切り替える場合に、このオプションを選択します。

SNMP サブエージェントパラメータの構成

構成の SNMP サブエージェント設定を変更するには、「SNMP サブエージェント設定」セクションの値を編集します。次の表に、SNMP サブエージェントパラメータのフィールド説明を示します。

表 13–3 フィールド説明 > SNMP サブエージェント設定

フィールド	説明
有効	一般に、SNMP を使用するには、1 つのマスターエージェントと少なくとも 1 つのサブエージェントをシステム上にインストールし、それらを実行する必要があります。サブエージェントを有効にするには、まずマスターエージェントをインストールしておく必要があります。 SNMP 統計収集の有効/無効を切り替える場合に、このオプションを選択します。
マスターホスト	サーバーの名前とドメインを入力します (*UNIX* の場合のみ)。
説明	サーバーの簡単な説明 (オペレーティングシステム情報を含む) を入力します。
組織	組織を表す簡易名を入力します。
場所	このフィールドにはサーバーの場所に関する情報を入力します。

表 13–3 フィールド説明 > SNMP サブエージェント設定 (続き)

フィールド	説明
連絡先	このフィールドにはサーバーの連絡先に関する情報を入力します。

SNMP サブエージェントの設定

SNMP は、ネットワークアクティビティーに関するデータをやり取りするために使用されるプロトコルです。SNMP では、管理対象デバイスとネットワーク管理ステーション (NMS) との間でデータが転送されます。管理対象デバイスは、SNMP が実行されている任意のデバイスです。つまり、ネットワーク上のホスト、ルーター、Web サーバー、その他のサーバーなどです。NMS は、そのネットワークをリモート管理する場合に使用するマシンです。NMS ソフトウェアは通常、収集されたデータを表示するグラフを提供したり、そのデータを使ってサーバーがある特定の許容範囲内で稼働しているか確認したりします。

NMS は通常、1 つ以上のネットワーク管理アプリケーションがインストールされた強力なワークステーションです。Sun Management Center のようなネットワーク管理アプリケーションでは、Web サーバーなどの管理対象デバイスに関する情報がグラフィカルに表示されます。たとえば、社内のどのサーバーが稼動またはダウンしているかを表示したり、受け取ったエラーメッセージの数と種類を表示したりできます。SNMP と Sun Java System Web Server を組み合わせて使用する場合、サブエージェント、マスターエージェント の 2 種類のエージェントを使って、NMS とサーバーとの間でこの情報が転送されます。

サブエージェントはサーバーに関する情報を収集し、その情報をサーバーのマスターエージェントに渡します。

SNMP サブエージェントを起動するには、次のタスクを実行します。

1. 「ノード」タブをクリックします
2. ノードのリストから、選択可能なノードをクリックします。
3. 「**SNMP** サブエージェント」タブをクリックします
4. 「**SNMP** サブエージェントを起動」ボタンをクリックしてサブエージェントを起動します。

注 – SNMP サブエージェントを起動する前に、マスターエージェントが稼働中であることを確認してください。サブエージェントが起動されるのは、マスターエージェントが稼働中の場合だけです。

SNMP サブエージェントを停止するには、次のタスクを実行します。

1. 「ノード」タブをクリックします

2. ノードのリストから、選択可能なノードをクリックします。
3. 「**SNMP** サブエージェント」タブをクリックします
4. 「**SNMP** サブエージェントを停止」ボタンをクリックしてサブエージェントを停止します。

一般に、SNMP を使用するには、1 つのマスターエージェントと少なくとも 1 つのサブエージェントをシステム上にインストールし、それらを実行する必要があります。サブエージェントを有効にするには、まずマスターエージェントをインストールしておく必要があります。

SNMP の設定手順は、システムによって異なります。次の表では、従うべき手順の概要を、さまざまな状況ごとに示します。実際の手順については、この章のあとのほうで詳しく説明します。

設定を開始する前に、次の 2 つの点を確認する必要があります。

- SNMP エージェント (使用するオペレーティングシステムのネイティブエージェント) がシステムですでに稼動していること
- 稼働している場合、ネイティブ SNMP エージェントが SMUX 通信をサポートしていること (AIX プラットフォームを使用している場合、システムは SMUX をサポートしている)

この情報を確認する方法については、使用しているシステムのマニュアルを参照してください。

注 – 管理サーバーの SNMP の設定を変更したあと、新しいサーバーをインストールしたあと、または既存のサーバーを削除したあとは、次の手順を実行する必要があります。

- (Windows) Windows SNMP サービスを再起動するか、マシンを再起動します。
- (UNIX) 管理サーバーを使用して SNMP マスターエージェントを再起動します。

表 **13–4** 一般的な指針

サーバーが満たしている条件	実行する手順
■ ネイティブエージェントが現在実行されていない	1. マスターエージェントを起動します。 2. システムにインストールされている各サーバーのサブエージェントを有効にします。

表 13–4 一般的な指針 (続き)

サーバーが満たしている条件	実行する手順
■ ネイティブエージェントが現在実行されている ■ SMUX をサポートしていない ■ ネイティブエージェントの使用を継続する必要がない	1. 管理サーバーのマスターエージェントをインストールする場合は、ネイティブエージェントを停止します。 2. マスターエージェントを起動します。 3. システムにインストールされている各サーバーのサブエージェントを有効にします。
■ ネイティブエージェントが現在実行されている ■ SMUX をサポートしていない ■ ネイティブエージェントの使用を継続する必要がある	1. プロキシ SNMP エージェントをインストールします。 2. マスターエージェントを起動します。 3. プロキシ SNMP エージェントを起動します。 4. マスターエージェントのポート番号以外のポート番号を使用して、ネイティブエージェントを再起動します。 5. システムにインストールされている各サーバーのサブエージェントを有効にします。
■ ネイティブエージェントが現在実行されている ■ SMUX をサポートしている	1. SNMP ネイティブエージェントを再設定します。 2. システムにインストールされている各サーバーのサブエージェントを有効にします。

CLI を使用した SNMP の構成

▼ Solaris で SNMP を有効にする

1 **SNMP** パラメータを構成します。

構成の SNMP パラメータを設定します。

```
wadm> set-snmp-prop --user=admin --host=funland --port=1893
--config=test enabled=true master-host=masterhost-name organization=organization-name
location=location-name contact=contact-name description=description-name
```

2 構成を配備します。

```
wadm> deploy-config --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 config1
```

3 サーバーインスタンスを起動します。

```
$ ./https-test/bin/startserv
```

4 マスターエージェント **(magt)** を **root** として実行します。

注 – magt を実行するには、ネイティブの snmpd を停止する必要があります。

```
$ cd /etc/init.d/
          $ init.dmi stop; init.snmpdx stop; init.sma stop
```

ファイル https-admserv/config/logs/pid.masteragt が存在する場合はそれを削除します。

```
$ rm ./https-admserv/config/logs/pid.masteragt
          wadm>  start-snmp-master-agent --snmp-port 161 hostname
```

5 サブエージェントを起動します。

ファイル https-admserv/config/logs/pid.httpagt が存在する場合はそれを削除します。

```
$ rm ./https-admserv/config/logs/pid.httpagt
```

httpagt がすでに稼働中の場合はそれを終了させます

```
wadm> start-snmp-subagent hostname
```

▼ Linux で SNMP を有効にする

1 **SNMP** パラメータを構成します。

構成の SNMP パラメータを設定します。

```
wadm> set-snmp-prop --user=admin --host=funland --port=1893 --config=test
enabled=true master-host=masterhost-name organization=organization-name
location=location-name contact=contact-name description=description-name
```

2 構成を配備します。

```
wadm deploy-config --user=admin --password-file=admin.pwd
--host=serverhost --port=8989 config1
```

3 サーバーインスタンスを起動します。

```
$ ./https-test/bin/startserv
```

4 ネイティブのマスターエージェント **(snmpd)** を **root** として実行します。

snmpd と直接通信できるようにするには、/etc/snmp/snmpd.conf に次の行を追加し、snmpd を再起動します。

```
smuxpeer 1.3.6.1.4.1.42.2.190.1
```

```
view systemview included .1.3.6.1.4.1.42.2.190.1
```

```
# cd /etc/init.d/
      # ./snmpd stop
      # ./snmpd start
```

5 サブエージェントを起動します。

ファイル `https-admserv/config/logs/pid.httpagt` が存在する場合はそれを削除します。

```
$ rm ./https-admserv/config/logs/pid.httpagt
```

`httpagt` がすでに稼働中の場合はそれを終了させます

```
wadm> start-snmp-subagent hostname
```

▼ Windows で SNMP を有効にする

1 **SNMP** パラメータを構成します。

構成の SNMP パラメータを設定します。

```
wadm> set-snmp-prop --user=admin --host=funland --port=1893 --config=test
enabled=true master-host=masterhost-name organization=organization-name
location=location-name contact=contact-name description=description-name
```

2 システムパス環境変数に `install-root/lib` ディレクトリを追加します。

3 マシンを再起動します。

4 **Windows** のサービスオプションを使って **Web Server** インスタンスを起動します。

5 **SNMP** サービスを起動します。

▼ ピアベースのマスターエージェント (magt) を構成する

次の手順に従えば、Solaris 10 および Linux 上の OS ネイティブマスターエージェントと統合されるようにピアベースのマスターエージェントを構成することができます。

注 – Solaris 10 の OS ネイティブマスターエージェントは、snmpd です。デフォルトでは、これは SNMP のデフォルト UDP ポート 161 上で実行されます。これは、/etc/sma/snmp/snmpd.conf ファイルを使って構成可能です。これは、ほかのマスターエージェントやサブエージェントに要求/応答を転送するためのプロキシ指令を提供します。詳細については、snmpd.conf のマニュアルページを参照してください。

Solaris 8 および 9 の場合、OS ネイティブマスターエージェント snmpd との明確な統合は存在しません。Linux の場合、httpagt を snmpd に直接統合することができます。そうした場合、magt を実行する必要はありません。Windows の場合、Sun Java System Web Server の snmp ライブラリが Windows の SNMP サービスと直接通信します。

1 上の注で説明したように、**SNMP** ポート **(11161)** を指定してマスターエージェントを起動します。

2 **Solaris 10** の **/etc/sma/snmp/snmpd.conf** に次の行を追加します。

```
proxy -v 1 -c public myserver:11161 .1.3.6.1.4.1.42.2.190.1
```

3 **snmpd** を再起動します。

```
# cd /etc/init.d
# init.dmi stop; init.snmpdx stop; init.sma stop
# init.dmi start; init.snmpdx start; init.sma start
```

4 **SNMP** データを取得するには、**snmpwalk** をポート上で使用します。

```
$ snmpwalk -c public -v 1 <host-name>:<port> 1.3.6.1.4.1.42.2.190.1
```

サーバーのロギング設定

管理サーバーのログファイルには、サーバーに関するデータが記録されます。これには、検出したエラーのタイプやサーバーアクセスに関する情報が記録されます。これらのログを参照すれば、検出されたエラーのタイプや、特定ファイルへのアクセス発生時刻などのデータが得られるため、サーバーアクティビティーの監視や問題の解決を行えます。

管理コンソールの「ログの設定」ページを使えば、管理サーバーログに記録されるデータのタイプや形式を指定できます。たとえば、管理サーバーにアクセスするすべてのクライアントに関するデータをログに記録することを選択することも、特定のクライアントをログから除外することもできます。さらに、サーバーについて決まった量の情報を提供する共通のログ形式を選択したり、必要に応じてカスタムログファイル形式を作成したりすることもできます。

ログのタイプ

ログのタイプは大きく次の 2 つにわけられます。

1. アクセスログ — アクセスログには、サーバーへの要求やサーバーからの応答に関する情報が記録されます。
2. サーバーログ — サーバーログには、ログファイルの作成後にサーバーが検出したすべてのエラーが含まれます。このファイルには、サーバーの起動時刻や、サーバーへのログインに失敗したユーザー名などのサーバーに関する情報メッセージも記録されます。

アクセスログとサーバーログの表示

- サーバーログの表示.

  ```
  wadm> get-log --user=admin --password-file=admin.passwd --host=localhost
  --port=8989 --start-date=01/01/2006:09:00:00 --end-date=04/01/2006:10:00:00
  --config=test cat.test.com
  ```

  このコマンドは、ある特定の構成のサーバーログを、日付 `01/Jan/2006:09:00:00` から `04/Jan/2006:10:00:00` までの範囲で表示します。

- アクセスログの表示.

  ```
  wadm> get-access-log --user=admin --password-file=admin.passwd
  --host=localhost --port=8989 --status-code=300 --config=test cat.test.com
  ```

  このコマンドは、ある特定の構成に対する、状態コードが 300 のアクセスログエントリだけを表示します。

上記のコマンドでは、start-date および end-date オプションは、`dd/MM/yyyy:HH:mm:ss` の形式になるようにしてください。この日時形式はカスタマイズすることもできます。rcfile 内で変数 `wadm_log_date_format` を使えば、デフォルトの日時形式を使用する代わりにユーザー独自の日時形式を指定できます。

ログのパラメータの構成

構成のログの設定を有効化および編集するには、次のタスクを実行します。

1. 「構成」タブをクリックします
2. ログの設定を有効化/編集する必要のある構成を選択します。
3. 「一般設定」>「ログの設定」タブをクリックします

アクセスログの設定の編集

「アクセスログの設定」セクションのフィールドについて、次の表で説明します。

表 13–5 フィールド説明>アクセスログの設定の編集

フィールド	説明
アクセスログ	有効/無効。アクセスログはデフォルトで有効になっています。アクセスログを無効にするには、このオプションを選択します。アクセスログを有効にすると、サーバーのパフォーマンスがごくわずかですが低下します。
ファイルの場所	アクセスログのファイルの格納先となるサーバーパス。デフォルト値は *../logs/access* です
ログ形式	1. 共通のログ形式を使用 — このオプションが、ログファイルのデフォルトの形式タイプになります。サーバーは、要求ヘッダーから抽象された関連性のもっとも高い情報を、ログに記録します。 共通のログ形式は、 `IP address – user [date] “request” status content-length` です。 2. これらの詳細のみをログ — このオプションを使えば、要求ヘッダー内の特定の値のみをログに記録できます。次の値から選択します。 ■ クライアントのホスト名 ■ システム日付 ■ HTTP の状態 ■ HTTP ヘッダー ■ HTTP メソッド ■ クエリー文字列 ■ 仮想サーバー名 ■ 認証済みユーザー名 ■ 完全な HTTP 要求 ■ コンテンツ長 ■ 要求 URI ■ プロトコル

サーバーログの設定の編集

「サーバーログの設定」セクションのフィールドについて、次の表で説明します。

表 13–6 フィールド説明>サーバーログの設定の編集

フィールド	説明
ファイルの場所	サーバーログのファイルの格納先となるサーバーパス。デフォルト値は *../logs/errors* です

表 13–6 フィールド説明 > サーバーログの設定の編集 (続き)

ログの冗長レベル	このオプションを使えば、ログの粒度を簡単に設定できます。Web アプリケーションのテストやデバッグにお勧めのレベルは、「もっとも詳細」です。 本稼働環境にお勧めのログレベルは、「失敗」または「セキュリティー」です。「重大」ログレベルでは、ほんのわずかの詳細しかログに記録されません。
仮想サーバー名を出力	このオプションを選択すると、エラーに加えて、その要求を処理した仮想サーバーの名前もログに記録されます。
システムログに出力	すべてのメッセージをシステムログに記録します。
コンソールに出力	このオプションを選択すると、配備済み Web アプリケーションから発行された例外が「コンソール」に書き込まれた場合に、その例外がログに記録されます。 このオプションはデフォルトで有効になっています。
日時形式	時間の形式。エラーメッセージの末尾にタイムスタンプを追加するために使用されます。デフォルト値は *[%d/%b/%Y:%H:%M:%S]* です

ログファイルのアーカイブ

ログファイルが自動的に保存されるように設定することができます。ある特定の時刻に、あるいはある指定された間隔で、サーバーはアクセスログのローテーションを行います。サーバーは古いログファイルを保存し、その保存したファイルに、保存時の日付と時刻を含む名前を付けます。

たとえば、ファイルのローテーションが 1 時間ごとに行われるように設定した場合、サーバーはファイルを保存し、それに「`access.199907.0152400`」という名前を付けます。この場合、「名前|年|月|日付|*24* 時間時刻」が結合されて単一の文字列になります。アクセスログアーカイブファイルの形式は、設定したログローテーションのタイプによって異なります。

アクセスログのローテーションは、サーバー起動時に初期化されます。ローテーションを有効にすると、サーバーはタイムスタンプの付いたアクセスログファイルを作成し、ローテーションがサーバーの起動時に開始されます。

ローテーションが開始されると、アクセスログファイルに記録する必要のある要求が発生し、かつその要求が以前にスケジュールされた「次のローテーション時刻」のあとで発生した場合に、サーバーは新しいタイムスタンプのアクセスログファイルを作成します。

ログローテーションの設定

ログローテーションのオプションを使えば、構成されたインスタンスのエラー/アクセスログのローテーションスケジュールを作成できます。ログローテーションを設定するには、次の手順を実行します。

1. 「構成」タブをクリックします
2. ログの設定を有効化/編集する必要のある構成を選択します。
3. 「一般設定」>「ログの設定」タブをクリックします
4. 「ログの保存」セクションの「新規」ボタンをクリックします

「新規ログローテーションイベント」ページのフィールドについて、次の節で説明します。

表 13–7 フィールド説明>ログローテーションの設定

フィールド	説明
イベント	アクセスログをローテーション/サーバーログのローテーション。これらのオプションのいずれかまたはその両方を選択すると、選択したログタイプのローテーションを構成できます。
時間	構成されたイベント起動時刻。ドロップダウンボックスから時間と分の値を選択します。
	毎日 — 指定されたイベントを指定された時刻に毎日起動します。
	指定日 — 指定されたイベントを指定された日に起動します。
	1. 曜日 — 日曜から土曜までの間で、任意の曜日を指定します。
	2. 日付 — 1 から 31 までの間で、任意の日付をコンマ区切りエントリとして指定します。例: 4,23,9
	指定月 — 指定された月の指定された時刻に指定されたイベントを起動します。1 月から 12 月までの間で、月を指定します。
間隔	この期間後に指定されたイベントを起動します。
	1. 時間おき — ドロップダウンボックスから時間数を選択します。
	2. 秒おき — ドロップダウンリストから秒数を選択します。

スケジュールされたログローテーションを削除する必要がある場合は、「ログの保存」セクションの「削除」ボタンをクリックします。

管理サーバーのログの設定

管理コンソールを使って実行されたすべての構成変更が、管理サーバーによってログに記録されます。ログに頻繁に記録されるアクションとしては、新しい構成の作成、仮想サーバーの作成、インスタンスの設定などが挙げられます。ただし、Web アプリケーションへのアクセスや、Web アプリケーションへのアクセス時に発生した例外などの構成レベルの詳細は、構成ごとに個別のログに記録されます。

▼ サーバーログの場所を変更する

1. 「管理サーバー」>「一般」タブをクリックします。
2. 「ログの設定」セクションに移動します。
3. 「ファイルの場所」フィールドを編集します。

   エラーの格納先となるログの場所。ログファイルを維持するための有効なサーバーパスを入力します。UNIX システムの場合は、指定されたディレクトリへの書き込み権を管理サーバーが持っているかどうかの確認も行います。

   デフォルトの場所は `../log/errors` です

▼ ログの冗長レベルを変更する

1. 「管理サーバー」>「一般」タブをクリックします。
2. 「ログの設定」セクションに移動します。
3. 「ログの冗長レベル」を選択します。

   このオプションを使えば、ログの粒度を簡単に設定できます。テストやデバッグにお勧めのレベルは、「もっとも詳細」です。

   本稼働環境にお勧めのログレベルは、「失敗」または「セキュリティー」です。「重大」ログレベルでは、ほんのわずかの詳細しかログに記録されません。

▼ ログの日時形式を変更する

1. 「管理サーバー」>「一般」タブをクリックします。
2. 「ログの設定」セクションに移動します。

3 「日時形式」フィールドを編集します。

時間の形式。エラーメッセージの末尾にタイムスタンプを追加するために使用されます。デフォルト値は *[%d/%b/%Y:%H:%M:%S]* です

◆ ◆ ◆ 第 14 章

国際化とローカリゼーション

国際化およびローカライズされたバージョンの Sun Java System Web Server は、複数言語および複数エンコーディングのサポートを提供します。

複数バイトデータの入力

管理コンソールのページ上で複数バイトデータを入力したい場合、次の問題に注意する必要があります。

ファイルまたはディレクトリの名前

あるファイルまたはディレクトリの名前を URL 内に表示させる場合、その名前に 8 ビットまたは複数バイトの文字を含めることはできません。

LDAP のユーザーとグループ

電子メールアドレスでは、RFC 17.000 (`ftp://ds.internic.net/rfc/rfc17.000.txt`) で許可されている文字だけを使用してください。ユーザー ID とパスワードの情報は、ASCII で格納する必要があります。

ユーザーやグループの文字を正しい形式で確実に入力できるようにするには、`UTF-8` フォームに対応したクライアントを使って 8 ビットまたは複数バイトのデータを入力します。

複数の文字エンコーディングのサポート

Sun Java System Web Server 7.0 は、次の各機能で複数の文字エンコーディングをサポートします。

- 234 ページの「WebDAV」
- 234 ページの「検索」

WebDAV

Sun Java System Web Server は、`PROPPPATCH` および `PROPFIND` メソッドで、複数バイトのプロパティーの設定および取得をサポートします。要求では任意のエンコーディング形式が使えますが、サーバーからの応答は常に UTF-8 になります。

検索

Sun Java System Web Server 7.0 で使用される Java ベースの検索エンジンは、背後の Java VM がサポートするすべての文字エンコーディングで、ドキュメントのフルテキストでのインデックス作成と検索をサポートします。ドキュメントのデフォルトのエンコーディングは、検索コレクションの作成時に指定できます。HTML ドキュメントの場合、インデクサは、HTML メタタグに基づいてエンコーディングを推測しようとしますが、推測に失敗した場合は、デフォルトのエンコーディングを使用します。

検索インタフェースは JSP タグライブラリに基づいており、必要とする任意の言語やエンコーディングを使ってカスタマイズおよびローカライズできるようになっています。このタグライブラリの一覧については、『Sun Java System Web Server 7.0 *Developer's Guide to Web Applications*』を参照してください。

ローカライズされたコンテンツを提供するためのサーバー構成

エンドユーザーは、アクセス対象となるコンテンツの言語設定を記述した Accept-language ヘッダーを送信するように、ブラウザを構成することができます。Accept-language ヘッダーに基づいてコンテンツを提供するようにサーバーを構成するには、「構成」>(構成を選択)>「仮想サーバー」>(仮想サーバーを選択)>「サーバー設定」>「一般」>「ローカリゼーション」の「クライアント言語のネゴシエーションを行う」チェックボックスを有効にします。

たとえば、このオプションが有効になった状態で、クライアントが次の URL を要求するときに値 `fr-CH,de` を含む Accept-language ヘッダーを送信したとします。

```
http://www.someplace.com/somepage.html
```

サーバーは次の順序でファイルの検索を行います。

▼ 検索順序

1. Accept-language リスト fr-CH,de。

   ```
   http://www.someplace.com/fr_ch/somepage.html
   ```

   ```
   http://www.someplace.com/somepage_fr_ch.html
   ```

   ```
   http://www.someplace.com/de/somepage.html
   ```

   ```
   http://www.someplace.com/somepage_de.html
   ```

2. 国番号を含まない言語コード (fr-CH の場合は fr)。

   ```
   http://www.someplace.com/fr/somepage.html
   ```

   ```
   http://www.someplace.com/somepage_fr.html
   ```

3. en など、magnus.conf ファイル内で定義された DefaultLanguage。

   ```
   http://www.someplace.com/en/somepage.html
   ```

   ```
   http://www.someplace.com/somepage_en.html
   ```

4. これらがどれも見つからない場合、サーバーは次を試みます。

   ```
   http://www.someplace.com/somepage.html
   ```

注 – ローカライズされたファイルに名前を付けるときには、CH や TW などの国番号が小文字に、ダッシュ (-) が下線 (_) にそれぞれ変換される点に注意してください。

注意 – acceptlanguage 設定を有効にするとパフォーマンスが低下します。なぜなら、サーバーは、上記で説明したアルゴリズムに従って、Accept-language に指定されたすべての言語でコンテンツのチェックを行う必要があるからです。

◆ ◆ ◆ 付 録 A

以前のバージョンからのCLIの変更点

次の表では、Sun Java System Web Server 7.0 と以前のバージョンで実行可能な一般的なタスクのいくつかを示します。

表 A–1 以前のバージョンからのCLIの変更点

タスク	6.1 CLI	7.0 CLI
あるインスタンスのすべての配備済みWebアプリケーションを一覧表示します。	`wdeploy list -i INSTANCE_NAME -v VIRTUAL_SERVER`	`wadm> list-webapps --user=admin --port=8888 --password-file=admin.passwd --no-ssl`
新しいWebアプリケーションを配備します。	`wdeploy deploy -i INSTANCE_NAME -v VIRTUAL_SERVER -u URI_PATH war file name`	1. `wadm> add-webapp --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME --vs=VIRTUAL_SERVER --uri=URI_PATH war file name` 2. `wadm> deploy-config --user=admin --port=8888 --password-file=admin.passwd 'HOSTNAME'`
稼働中のインスタンスを再構成します。	サポートなし。	`wadm> reconfig-instance --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME`

表 A–1 以前のバージョンからの CLI の変更点 *(続き)*

タスク	6.1 CLI	7.0 CLI
あるインスタンスのすべての仮想サーバーを一覧表示します。	`HttpServerAdmin list -v -d INSTALL_DIR -sinst https-INSTANCE_NAME`	`wadm> list-virtual-servers --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME`
すべての JDBC リソースを一覧表示します。	`HttpServerAdmin list -r -jdbc -d INSTALL_DIR -sintance https-INSTANCE_NAME`	`wadm> list-jdbc-resources --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME`
カスタムリソースを作成します。	`HttpServerAdmin create -r -custom -jndiname -poolname -enabled true`	`wadm> create-custom-resource --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME --res-type=type --jndi-name NAME`
インスタンスを起動します。	サポートなし。	`wadm> start-instance --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME NODENAME*`
インスタンスを停止します。	サポートなし。	`wadm> stop-instance --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME NODENAME*`
逆プロキシを使用した Web サーバーの構成。	サポートなし。	1. `wadm> create-reverse-proxy --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME --vs='VIRTUAL_SERVER' --from='URI' --server='target-hostname'` 2. `wadm> set-reverse-proxy-prop --user=admin --password-file=admin.pwd --host=serverhost --port=8888 --config=config1 --vs=config1_vs_1 --uri-prefix=/test/ --server=http://java.com:8080 --sticky-cookie=testCookie`

表 A–1 以前のバージョンからの CLI の変更点 (続き)

タスク	6.1 CLI	7.0 CLI
逆プロキシを無効にします。	サポートなし。	`wadm> delete-reverse-proxy --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME --vs='VIRTUAL_SERVER' --uri-prefix='URI'`
WebDAV を有効にします。	サポートなし。	1. `wadm> enable-webdav --user=admin --port=8888 --password-file=admin.passwd --config=HOSTNAME` 2. `wadm> deploy-config --user=admin --port=8888 --password-file=admin.passwd HOSTNAME`
新しい Web サーバーを作成します。	サポートなし。	1. `wadm> create-config --doc-root=[DOCROOT] -http-port=[HTTPPORT] --jdk-home=[JAVAHOME] --server-user=[SERVERUSER] --server-name=[HOSTNAME] CONFIGNAME` 2. `wadm> create-instance --config=CONFIGNAME NODENAME` 3. `wadm> deploy-config CONFIGNAME`

◆ ◆ ◆ 付録 B

FastCGI プラグイン

概要

FastCGI は、外部アプリケーションと Web サーバー間の標準インタフェースである既存の CGI (Common Gateway Interface) を拡張したものです。FastCGI アプリケーションは CGI と同じく、隔離された個別のプロセス内で実行されます。FastCGI を使用する利点を次にいくつか示します。

- クライアント要求をまたがってアプリケーションが持続できるため、アプリケーション起動によるオーバーヘッドが解消される。さらに、クライアント呼び出しをまたがってアプリケーションの状態を維持できる。
- アプリケーションがリモートシステム上 (Web Server が稼働しているのとは異なるシステム上) に存在できる。
- クライアント認証および入力のフィルタリングを行うアプリケーションが明示的にサポートされるため、アプリケーション機能の柔軟性が高まる。
- FastCGI サーバーに起因するシステムへの影響を、管理者が制限できる。

FastCGI プラグインを使えば、Perl や Python など、広く普及しているサードパーティーの動的コンテンツ生成テクノロジと Web Server との安全な連携動作を、高いスケーラビリティーを維持するかたちで実現できます。

FastCGI の詳細については、http://www.fastcgi.com/devkit/doc/fcgi-spec.html の仕様書を参照してください。

プラグイン関数 (SAF)

FastCGI プラグインが提供するサーバーアプリケーション関数 (SAF) は、次のとおりです。

FastCGI SAF の各種パラメータと「error-reason」文字列については、次の各節で説明します。

auth-fastcgi

`auth-fastcgi` は PatchCheck 関数です。この関数は、「オーソライザ」FastCGI アプリケーションに要求を転送するために使用されます。承認が成功すると、リターンコード 200 が送信されます。それ以外の場合は、「オーソライザ」FastCGI アプリケーションからの応答がユーザーエージェントに送り返されます。

FastCGI のロールの詳細については、
http://www.fastcgi.com/devkit/doc/fcgi-spec.html#S6 を参照してください。

`auth-fastcgi` SAF が受け付けるパラメータについては、244 ページの「FastCGI SAF パラメータ」を参照してください。

次の `obj.conf` コード例は、`auth-fastcgi` の使用方法を示したものです。

```
PathCheck fn="auth-fastcgi" app-path="/usr/bin/perl"
app-args="/fastcgi/apps/auth/SimpleAuth.pl" bind-path="localhost:3432".
```

responder-fastcgi

`responder-fastcgi` は Service 関数です。この関数は、「レスポンダ」として機能する FastCGI アプリケーションに要求を転送するために使用されます。レスポンダアプリケーションからの応答は、ユーザーエージェントに送り返されます。FastCGI のロールの詳細については、http://www.fastcgi.com/devkit/doc/fcgi-spec.html#S6 を参照してください。

`responder-fastcgi` SAF が受け付けるパラメータの一覧については、244 ページの「FastCGI SAF パラメータ」 を参照してください。

次の `obj.conf` コード例は、`responder-fastcgi` の使用方法を示したものです。

```
Service fn="responder-fastcgi"
app-path="/fastcgi-enabled-php-installation/bin/php"
bind-path="localhost:3433" app-env="PHP_FCGI_CHILDREN=8"
app-env="PHP_FCGI_MAX_REQUEST=500".
```

filter-fastcgi

`filter-fastcgi` は Service 関数です。この関数は、「フィルタ」タイプの FastCGI アプリケーションに要求を転送するために使用されます。「フィルタ」アプリケーションは、HTTP 要求に関連付けられた情報と、サーバー上に格納されたファイルからのデータを受け取ります。次に、「フィルタ」アプリケーションは、「フィルタリング」されたバージョンのデータストリームを応答として生成します。これがユーザーエージェントに送り返されます。FastCGI のロールの詳細については、http://www.fastcgi.com/devkit/doc/fcgi-spec.html#S6 を参照してください。

`filter-fastcgi` SAF が受け付けるパラメータの一覧については、244 ページの「FastCGI SAF パラメータ」 を参照してください。

次の `obj.conf` コード例は、`filter-fastcgi` の使用方法を示したものです。

```
Service fn="filter-fastcgi" app-path="/fastcgi/apps/filter/SimpleFilter"
bind-path="localhost:3434"
app-env="LD_LIBRARY_PATH=/fastcgi/fcgi-2.4/libfcgi/.libs" min-procs=2
```

error-fastcgi

`error-fastcgi` は Error 関数です。`error-fastcgi` SAF は、FastCGI プラグインに固有のエラーを処理します。一方、この関数は HTTP エラーを処理しません。エラー時に特定のページを表示するか特定の URL に要求をリダイレクトするように、FastCGI プラグインを構成することができます。

`error-fastcgi` SAF が受け付けるパラメータの一覧については、244 ページの「FastCGI SAF パラメータ」を参照してください。

次の `obj.conf` コード断片は、`error-fastcgi` の使用方法を示したものです。

```
Error fn="error-fastcgi" error-reason="Invalid Parameters"
error-url="http://www.foo.com/errorPage.html"
```

`error-fastcgi` パラメータの詳細については、244 ページの「FastCGI SAF パラメータ」を参照してください。

FastCGI SAF パラメータ

FastCGI プラグインの SAF「auth-fastcgi」、「responder-fastcgi」、および「filter-fastcgi」はすべて、特に明記されていないかぎり、次のパラメータを受け付けます。

パラメータ chroot、user、group、および nice は、UNIX プラットフォームにしか適用できません。Windows プラットフォームでは、それらのパラメータは無視されます。

- app-path - (省略可能) 要求を処理する FastCGI アプリケーションのパス。その機能は次のように、bind-path パラメータの値に依存します。
  1. app-path のみが指定された場合、プラグインは、プラグインによって作成された UNIX ドメインソケットを待機する FastCGI アプリケーションを作成します。このパラメータは、UNIX プラットフォームでしか受け付けられません。Windows では、エラーメッセージがログに記録されます。
  2. app-path と bind-path がどちらも指定された場合、プラグインは、指定された FastCGI アプリケーションのプロセスを起動し、それを指定された bind-path にバインドします。
  3. bind-path のみが指定された場合、FastCGI アプリケーションはリモートで実行されているものとみなされます。このため、プラグインは、FastCGI アプリケーションのプロセスを起動しません。
  4. 「app-path」、「 bind-path」がどちらも指定されなかった場合、プラグインはログにエラーメッセージを記録します。
- app-args — (省略可能) FastCGI アプリケーションプロセスへの引数として渡される値。app-args パラメータは複数指定できます。app-args パラメータを複数指定する場合の形式は、app-args="value" app-args="value" ... です
- bind-path - (省略可能) UNIX ドメインソケット名、形式「host:port」のいずれかになります。「app-path」パラメータの説明のなかで、「bind-path」パラメータの使用方法が説明されています。UNIX ドメインソケット名は UNIX プラットフォームでしか適用できません。Windows プラットフォームでは、bind-path は「host:port 」として指定する必要があります。
- min-procs - (省略可能) 作成すべき FastCGI アプリケーションプロセスの最小数を指定する整数。デフォルトは 1 です。
- max-procs - (省略可能) 同時に作成可能な FastCGI アプリケーションプロセスの最大数を指定する整数。この整数値は、min-procs と等しいかそれより大きくなければいけません。デフォルトは 1 です。
- chroot - (省略可能) FastCGI サーバーアプリケーションプロセスの chroot を実行するときのルートディレクトリを設定するために使用されます。デフォルトは、Web Server のルートディレクトリになります。
- user - (省略可能) FastCGI アプリケーション実行時に使用するユーザー ID を指定します。デフォルトは、Web Server のユーザー ID になります。

- `group` - (省略可能) 指定されたグループの下で FastCGI アプリケーションが実行されます。デフォルトは、Web Server のグループになります。
- `nice` - (省略可能) FastCGI アプリケーションプロセスの `nice/priority` 値を指定します。
- `listen-queue` - (省略可能) ソケットの待機キューサイズを指定する整数。このパラメータのデフォルト値は 256 です。
- `app-env` - (省略可能) FastCGI アプリケーションプロセスに環境変数として渡される値ペア。「`app-env`」パラメータは複数指定できます。`app-env` パラメータを複数指定する場合の形式は、`app-env="name=value" app-env="name=value"...` です。
- `reuse-connection` - (省略可能) FastCGI アプリケーションへの接続が再利用されるかどうかを決定するブール値。`False` (`0`、`false`、`no`) は、要求が終了するたびに FastCGI アプリケーションへの接続が閉じられることを示します。`True` (`1`、`true`、`yes`) は、既存の接続が新しい要求で再利用されることを示します。デフォルトは `false` です。`connection-timeout` も参照してください。
- `connection-timeout` - (省略可能) 「`reuse-connection`」が `True` に設定されている場合に、この値はプール内の接続のタイムアウト値を秒単位で指定します。この指定された期間の間、ある接続がアイドル状態になっていると、プラグインはその接続を閉じます。このパラメータのデフォルト値は 5 秒です。`reuse-connection` も参照してください。
- `resp-timeout` - (省略可能) FastCGI サーバー応答タイムアウトを秒単位で表す整数。指定された期間の間、FastCGI アプリケーションから応答がなかった場合、その要求は破棄されます。このパラメータのデフォルト値は 5 分です。
- `restart-interval` - (省略可能) FastCGI アプリケーションが次に再起動されるまでの時間間隔 (分) を表す整数。このパラメータのデフォルト値は 60 分 (1 時間) です。このパラメータの値をゼロに設定すると、FastCGI アプリケーションが強制的に再起動されることがなくなります。
- `req-retry` - (省略可能) FastCGI アプリケーションが要求を拒否した場合にプラグインが要求を再送信すべき回数を表す整数。このパラメータのデフォルト値はゼロです。

`error-fastcgi` サーバーアプリケーション関数 (SAF) は次のパラメータを受け付けます。

- `error-url` - 障害またはエラーが発生したときに表示するページの URI または URL を指定します。このパラメータの値は、絶対パス、ドキュメントルートからの相対パス、URL、URI のいずれかになります。
- `error-reason` - (省略可能) FastCGI プロトコルのエラーを表す文字列。この文字列は、プラグインエラー発生時に表示するエラー URL を区別するために使用されます。

error-fastcgi SAF のエラー理由文字列

この節では、有効なすべての「error-reason」文字列とその説明の一覧を示します。

- 「Missing or Invalid Config Parameters」: app-path と bind-path が指定されていない場合に常に使用。
- 「Stub Start Error」: Fastcgistub プロセスの起動に失敗した。
- 「Stub Connection Failure」: Fastcgistub に接続できない。
- 「No Permission」: FastCGI アプリケーションまたは Fastcgistub が実行権を持っていない。
- 「Stub Request Handling Error」: 要求をスタブに送信できない、スタブから要求に対する無効な応答を受信したり応答を受信しなかった、など。
- 「Set Parameter Failure」: ユーザー、グループ、ディレクトリ変更、優先順位などの設定に失敗した場合。
- 「Invalid user and/or group」: ユーザーまたはグループが無効である場合。
- 「Server Process Creation Failure」: FastCGI アプリケーションの実行に失敗したか、指定されたアドレスに FastCGI アプリケーションをバインドできない。
- 「Fastcgi Protocol Error」: 無効な FastCGI バージョンまたはロールを含むヘッダーが、FastCGI アプリケーションに含まれている。
- 「Internal Error」: フィルタアプリケーションに送信するファイルを開くことができない、あるいはその他の未知のエラーが発生した。

Web Server での FastCGI プラグインの構成

FastCGI プラグインは Web Server 7.0 にバンドルされています。このプラグインのインストール場所は次のとおりです。

32 ビットの FastCGI プラグインバイナリは、*<install_dir>*`/plugins/fastcgi` ディレクトリにインストールされます。
64 ビットの Solaris SPARC FastCGI プラグインバイナリは、
<install_dir>`/lib/plugins/fastcgi/64` ディレクトリにインストールされます。

インストールされる FastCGI バイナリは次のとおりです。

`libfastcgi.so` (Solaris/Linux 用)
`fastcgi.dll` (Windows 用)
`Fastcgistub.exe` (Windows 用)
`libfastcgi.sl` (HP-UX 用)
`Fastcgistub` (実行可能ファイル)

FastCGI プラグインを構成するには、<instance-dir>/config ディレクトリに格納されている Web Server 構成ファイルを使用します。FastCGI プラグインを構成するには、次の手順を実行します。

- 247 ページの「magnus.conf の変更」
- 247 ページの「MIME タイプの変更 (省略可能)」
- 248 ページの「obj.conf の変更」
- 249 ページの「FastCGI プラグインのトラブルシューティング」
- 250 ページの「FastCGI アプリケーションの開発」

magnus.conf の変更

「load-modules」Init 関数を使って FastCGI プラグインの共有ライブラリを読み込みます。

```
Init fn=flex-init access="access" format.access="%Ses->client.ip%
- %Req->vars.auth-user% [%SYSDATE%] \"%Req->reqpb.clf-request%\"
%Req->srvhdrs.clf-status% %Req->srvhdrs.content-length%"

Init fn="load-modules" shlib="libJava EEplugin.so" shlib_flags="(global|now)"

Init fn="load-modules" shlib="libfastcgi.so" shlib_flags="(global|now)"
```

MIME タイプの変更 (省略可能)

mime.types ファイルを編集して MIME マッピングを指定します。MIME タイプのマッピングの変更は、省略可能な手順です。

次に例を示します。

```
#--Sun Microsystems Inc. MIME Information

# Do not delete the above line. It is used to identify the file type.

#

# Copyright 2006 Sun Microsystems, Inc. All rights reserved.

# Use is subject to license terms.

#


type=application/octet-stream exts=bin
```

```
type=application/astound exts=asd,asn

...

...

type=magnus-internal/fastcgi exts=php

...

...
```

obj.conf の変更

obj.conf ファイルを編集し、これまでの節で説明したプラグイン SAF を使って FastCGI に固有の要求を構成します。

変更後の obj.conf ファイルの例を、次に示します。

```
#

# Copyright 2006 Sun Microsystems, Inc. All rights reserved.

# Use is subject to license terms.

#


# You can edit this file, but comments and formatting changes

# might be lost when you use the administration GUI or CLI.



<object name = "default">

        AuthTrans fn="match-browser" browser="*MSIE*"
                        ssl-unclean-shutdown="true"
        NameTrans fn="ntrans-Java EE" name="Java EE"
        NameTrans fn="pfx2dir" from="/mc-icons"
                        dir="/ws7/lib/icons" name="es-internal"
        NameTrans fn="assign-name" from="/fcgi/*" name="fcgi.config"

</object>

<Object name="fcgi.config">
```

```
        AuthTrans fn="auth-fastcgi" app-path="/fastcgi/apps/c/simpleAuth"
                bind-path="localhost:2111"
        Service fn="responder-fastcgi"
                        app-path="/fastcgi_enabled_php_installation_dir/bin/php"
                app-env="name1=abc"

</object>
...
```

FastCGI の SAF は、URL パターンごとに異なるオブジェクトを定義したり、SAF を異なる MIME タイプにマップしたりするなど、さまざまな方法で呼び出せます。

obj.conf の構成や構文の詳細については、『*Administration Configuration File Reference Guide*』を参照してください。

FastCGI プラグインのトラブルシューティング

Fastcgistub は、FastCGI アプリケーションプロセスのライフサイクルを管理するプロセスマネージャーです。Fastcgistub は、Web Server の一時ディレクトリの下の `Fastcgistub.log` ファイルに、自身のメッセージを記録します。エラーが発生した場合、このファイルをチェックすると問題のデバッグが容易になる可能性があります。

問題: FastCGI の要求が処理されない

可能性のある原因とその解決法は、次のとおりです。

1. FastCGI プラグインが読み込まれているかチェックします。Web Server の起動時に次のメッセージが表示されれば、このプラグインは正常に読み込まれています。そうでない場合は、`magnus.conf` でプラグインライブラリへのパスをチェックします。 `FCGI1000: Sun Java System Web Server 7.0 FastCGI NSAPI Plugin` <ビルド情報>
2. `obj.conf` で要求のマッピングが正しく指定されているかチェックします。`obj.conf` ファイルの詳細については、『Sun Java System Web Server *Administrator's Configuration Reference File*』を参照してください。
3. エラーログにエラーメッセージが含まれていないかチェックします。
4. スタブバイナリと FastCGI アプリケーションのアクセス権をチェックします。十分なアクセス権が与えられていない場合、プラグインはスタブまたはアプリケーションの起動に失敗します。
5. `Fastcgistub.log` ファイルにスタブ側のエラーが含まれていないかチェックします。
6. 可能であれば、FastCGI アプリケーションをスタンドアロンモードで実行し、アプリケーションが問題なく実行されるかチェックします。

ライブラリの依存関係に関するエラーがスローされた場合には、LD_LIBRARY_PATH=<依存関係のあるライブラリのパス> を値に持つ app-env パラメータとして、obj.conf 内で LD_LIBRARY_PATH を指定します。

問題: FastCGI アプリケーションが起動されない。

可能性のある原因とその解決法は、次のとおりです。

Fastcgistub.log ファイルに次のログメッセージが含まれていないかチェックします。

```
..
<pid> process startup failure, trying to restart
...
Even after trying <n> time(s), <application path> process failed to start...no more retries
```

起動が失敗する原因の 1 つとして、依存関係のあるライブラリの読み込みに失敗していることが考えられます。obj.conf ファイル内で構成されている FastCGI アプリケーションへの app-env パラメータの値として、適切なライブラリパスを指定すれば、この問題を解決できます。次に例を示します。

```
Service fn="responder_fastcgi" app-path="/fastcgi/c/tux-app" bind-path="localhost:2112"
app-env="LD_LIBRARY_PATH=/tuxedo/lib"
```

FastCGI アプリケーションの開発

FastCGI アプリケーションの開発は、Perl、PHP、C、および Java を使って行えます。次の各節では、広く普及しているいくつかのプログラミング言語を使ってアプリケーションを開発する手順について、簡単に説明します。

▼ FastCGI アプリケーションの実行

1. **Web Server** を停止します。
2. **Web Server** を再起動します。
3. 「**fcgi**」をアプリケーションルートに持つアプリケーションにアクセスします。

   例: http://localhost/fcgi/ListDir.php

FastCGI アプリケーションの構造

典型的な FastCGI アプリケーションのコードは、次の構造を持ちます。

```
初期化コード

応答ループの開始
        応答ループの本体
応答ループの終了
```

初期化コードが実行されるのは、アプリケーションの初期化時に 1 回だけです。初期化コードは通常、データベースのオープンや、テーブルまたはビットマップの値の計算など、時間のかかる処理を実行します。CGI プログラムを FastCGI プログラムに変換する場合の主なタスクは、初期化コードと、要求ごとに実行する必要のあるコードとを分離することです。

応答ループは継続的に実行され、クライアント要求が到着するのを待ちます。このループはまず、FastCGI ライブラリのルーチン `FCGI_Accept` を呼び出します。`FCGI_Accept` ルーチンは、クライアントが FastCGI アプリケーションを要求するまで、プログラムの実行をブロックします。クライアント要求が到着すると、`FCGI_Accept` はブロックを解除し、応答ループの本体を 1 回実行したあと再度ブロックし、別のクライアント要求の到着を待ちます。このループが終了するのは、システム管理者または Web Server が FastCGI アプリケーションを終了した場合だけです。

Perl の使用

最新の FCGI モジュールを CPAN からダウンロードしてインストールします。ActivePerl については、`http://aspn.activestate.com/ASPN/Downloads/ActivePerl/PPM/Zips` からモジュールをダウンロードできます。

Perl を使って FastCGI アプリケーションを記述する方法の詳細については、`http://www.fastcgi.com/devkit/fastcgi-prog-guide/ch3perl.htm#3659` を参照してください

PHP の使用

PHP 4.3.0 以降、FastCGI が PHP エンジンのサポートされた構成の 1 つになりました。PHP 4.3.x 以上のエンジンを FastCGI サポートを有効にしてコンパイルするには、次のように、構成スイッチ `--enable-fastcgi` をビルド処理の一部として含めます。

```
./configure <other-options> --enable-fastcgi
gmake
```

コンパイルが完了すると、`php` バイナリが FastCGI 対応になります。

PHP バージョン 5.1.2 以前 (PHP 4.x を含む) を使用する場合は、host:port 形式の bind-path を使って FastCGI プラグインを構成するようにしてください。たとえば、bind-path = “localhost:3333” のようにします。

PHP バージョン 5.1.3 以降の場合、bind-path は省略可能になります。指定する場合は、「host:port」形式を使用しないようにしてください。文字列で表します。たとえば、bind-path = “myphpbindpath” のようにします。

C/Java の使用

FastCGI 開発キットは、FastCGI C/Java アプリケーションを開発するための API を提供します。このキットは http://www.fastcgi.com/devkit/doc/fcgi-devel-kit.htm からダウンロードできます。

ダウンロードした FastCGI 開発キットをビルドするには、次の手順を実行します。

1. tar ファイルを展開します。このアクションにより、`fcgi-devel-kit` という名前の新しいディレクトリが作成されます
2. `fcgi-devel-kit` ディレクトリ内で次の一連のコマンドを実行します。
   a. ./configure
   b. make

C を使って FastCGI アプリケーションを記述する方法の詳細については、http://www.fastcgi.com/devkit/doc/fcgi-devel-kit.htm#S3 を参照してください

Java を使って FastCGI アプリケーションを記述する方法の詳細については、http://www.fastcgi.com/devkit/doc/fcgi-java.htm を参照してください

サンプル FastCGI アプリケーション

この節には、PHP、Perl、および C を使って記述されたサンプル FastCGI アプリケーションが含まれています。

- 252 ページの「PHP で記述されたレスポンダアプリケーション (ListDir.php)」
- 253 ページの「Perl で記述されたオーソライザアプリケーション (SimpleAuth.pl)」
- 254 ページの「C で記述されたフィルタアプリケーション (SimpleFilter.c)」

PHP で記述されたレスポンダアプリケーション (ListDir.php)

```
<?php
        $dir = "/tmp/";
```

```
        // ある既知のディレクトリを開き、続いてその内容を読み取ります
        if (is_dir($dir)) {
            if ($dh = opendir($dir)) {
                 while (($file = readdir($dh)) !== false) {
                     echo "filename: $file : filetype: " . filetype($dir . $file) . "\n";
                 }
                closedir($dh);
            }

        }
?>
```

この例に対する obj.conf の一部を次に示します。

```
<Object name="default">
        NameTrans fn="assign-name" from="/fcgi/*" name="responder.fcgi"
</Object>
<Object name="responder.fcgi">
        Service fn="responder-fastcgi" app-path="/foo/fastcgi-enabled-php-installation/bin/php"
            bind-path="localhost:3431"  min-procs=3
</Object>
```

Perl で記述されたオーソライザアプリケーション (SimpleAuth.pl)

```
#!/usr/bin/perl

use FCGI;

while (FCGI::accept >= 0) {
        if( $ENV{'HTTP_AUTHORIZATION'} ) {
        # この値をさらに復号化すれば、実際のユーザー名とパスワードが得られるので、
        # 何らかのユーザー検証を実行できます。このプログラムでは、この環境パラメータ
        # の存在チェックしか行っておらず、その値を実際に処理しているわけではありません

        print( "Status: 200\r\n" );
        print( "\r\n" );

    } else {

        print( "Status: 401\r\n" );
        print( "WWW-Authenticate: basic realm=\"foo\"\r\n" );
        print( "\r\n" );

   }

}
```

この例に対する obj.conf の一部を次に示します。

```
<Object name="responder.fcgi">
        AuthTrans fn="auth-fastcgi" app-path="/fastcgi/apps/auth/SimpleAuth.pl"
        bind-path="localhost:3432"
        Service fn="responder-fastcgi" app-path="/foo/fastcgi-enabled-php-installation/bin/php"
        bind-path="localhost:3433" app-env="PHP_FCGI_CHILDREN=8" min-procs=1
</Object>
```

`http://localhost/fcgi/php/ListDir.php` への要求が初めて発生すると、ブラウザによって認証ダイアログボックスが表示されます。ユーザー名とパスワードの入力が完了すると、「/tmp」ディレクトリの内容が一覧表示されます。

C で記述されたフィルタアプリケーション (SimpleFilter.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcgi_stdio.h>

        void main(void) {
            size_t PageSize = 1024 * 3;
            char *page;
            FCGX_Stream *in, *out, *err;
            FCGX_ParamArray envp;

             int count=0;
            page = (char *)malloc(PageSize);

             if (page == NULL) {

                printf("Content-type: text/x-server-parsed-html\r\n");
                printf("<title>malloc failure</title>");
                printf("<h1>Cannot allocate memory to run filter. exiting</h1>");
                printf("\r\n\r\n");
                exit(2);
            }

            while(FCGI_Accept() >= 0) {

             char *tmp;
             char *execcgi;
             char *dataLenStr = NULL;
             int numchars = 0;
             int stdinDataSize = 0;
```

```
 int filterDataLen = 0;
 int dataToBeRead = 0;
 int x = 0;
 int loopCount = 0;


      count++;
 dataLenStr = getenv("FCGI_DATA_LENGTH");

 if(dataLenStr)
     filterDataLen = atoi(dataLenStr);

    /* stdin のクリア */
 while (EOF != getc(stdin)) {
     stdinDataSize++;
 }

    dataToBeRead = filterDataLen;
FCGI_StartFilterData();
tmp = page; /** fread や fwrite が我々のポインタを動かす場合にそなえて **/


    // 応答を開始します
    printf("Content-type: text/plain\r\n");
    printf("\r\n"); /** send a new line at the beginning **/
    printf("<title>SIMPLE FILTER</title>");
    printf(<h1>This page was Filtered by SimpleFilter FastCGI filter</h1>");
    printf("file size=%d<br>", filterDatalen);
    printf("stdin size=%d<br>, stdinDataSize);


    while(dataToBeRead > 0 ) {
       x = 0;
       page = tmp;

       if(dataToBeRead > PageSize)
          x = PageSize;
       else
           x = dataToBeRead;
        numchars = fread((void *)(page), 1, x, stdin);

        if( numchars == 0 )
            continue;
        /** この時点で、データは page ポインタに格納されています。したがって、そのデータをサーバー

        dataToBeRead -= numchars;
        loopCount++;
        printf("loop count = %d ... so far read %d bytes <br>", loopCount,
```

```
                    (filterDatalen - dataToBeRead));
                }
                printf("\r\n\r\n"); /** 転送終了時に改行を送信します **/

                fflush(stdout);

                page = tmp; /** ページポインタを復元します **/
                memset(page,NULL,numchars);
        }

        free(page);
}
```

この例に対する obj.conf の設定例を次に示します。

この FastCGI アプリケーションが、Web Server が稼働しているのと同じマシン上で利用可能になっている場合は、次のようにします。

```
<Object name=<"filter.fcgi">
                    Service fn="filter-fastcgi" app-path="/fastcgi/apps/filter/SimpleFilter.exe"
            bind-path="localhost:3434" app-env="LD_LIBRARY_PATH=/fastcgi/fcgi-2.4/libfcgi/.libs"
</Object>
```

このアプリケーションがリモートマシン上で実行されている場合には、obj.conf ファイル内に次のコード行を含める必要があります。

```
<Object name="filter.fcgi">
            Service fn="filter-fastcgi" bind-path="<remote-host>:<remote-port>"
</Object>
```

Web Server インスタンスのドキュメントルートディレクトリの下にある fcgi ディレクトリに格納されたサイズ「26868」バイトの「FilterThisFile」が、フィルタリング対象のファイルである場合、「http://localhost/fcgi/filter/FilterThisFile」への要求によって次の出力が生成されます。

```
This page was Filtered by SimpleFilter FastCGI filter


file size = 26868
stdin size = 0
loop count = 1... so far read 3072 bytes
loop count = 2... so far read 6144 bytes
loop count = 3... so far read 9216 bytes
loop count = 4... so far read 12288 bytes
loop count = 5... so far read 15360 bytes
loop count = 6... so far read 18432 bytes
loop count = 7... so far read 21504 bytes
loop count = 8... so far read 24576 bytes
```

```
loop count = 9... so far read 26868 bytes
```

◆ ◆ ◆ 付録 C

Web サービス

Sun Java System Web Server 7.0 で Web サービスを実行する場合、追加の構成を行う必要はありません。JWSDP はサーバーに統合されているため、どのような JWSDP Web アプリケーションでも、Web アプリケーションとして配備すれば実行できるはずです。

Web アプリケーションの配備方法の詳細については、174 ページの「Web アプリケーションの追加」を参照してください。

Web Server 7.0 での JWSDP 2.0 サンプルの実行

JWSDP 2.0 に含まれる Web アプリケーションサンプルを Web Server 7.0 に配備するには、まずそれらの構成ファイルを変更する必要があります。具体的には、`jaxws` サンプルの構成ファイルを編集し、Web Server 7.0 に配備できるようにします。手順は次のとおりです。

▼ JWSDP 2.0 サンプルの実行

1. **JWSDP 2.0** をダウンロードします。
2. **$JWSDP_HOME/jwsdp-shared/bin** に、**Web Server** 固有の `sjsws.props` を作成します。
   サンプルの `sjws.props` を次に示します。すべてのフィールドが必須です。

```
ADMIN_USER=admin
ADMIN_PORT=8800
ADMIN_HOST=localhost
ADMIN_PASSWORD_FILE=/tmp/admin.passwd
CONFIG=jwsdp
VS=jwsdp
WS_HOME=/export/ws7.0
```

```
WS_PORT=5555
WS_HOST=localhost
```

注 – admin.passwd ファイルには管理者のサーバーパスワードが格納されています。このエントリの例: wadm_password=adminadmin

---

3 構成ファイルを変更します。

実行する予定のサンプルの build.xml および etc/deploy-targets.xml ファイルを変更します。deploy-targets.xml で必要な変更はサンプルに依存しません。マスターコピーを使用し、それを実行する予定のアプリケーションの etc ディレクトリにコピーできるようにしてください。

build.xml の変更。

build.xml の先頭にある Application Server の lib.home 定義をコメントにし、Web Server の lib の場所を追加します。変更後の build.xml の一部を、次に示します。

```
<!--
   **                                                    **
   ** Application Server の lib.home 宣言をコメントにします **
   **                                                    **
    <property file="../../../jwsdp-shared/bin/sjsas.props"/>
      <condition property="lib.home" value="${DOMAIN_DIR}/../lib">
        <available file="../../../jwsdp-shared/bin/sjsas.props"/>
      </condition>
    <condition property="lib.home" value="${env.JAXWS_HOME}/lib">
      <not>
        <available file="../../../jwsdp-shared/bin/sjsas.props"/>
      </not>
    </condition>
-->
<!--
   ** Web Server のライブラリの場所を追加します **
-->
    <property name="lib.home" value="${WS_HOME}/lib" />
```

deploy-targets.xml **changes**.

etc/deploy-targets.xml を Web サーバー固有の deploy-targets.xml で置き換えます。この変更により、Web アプリケーションが Web Server に配備されるようになります。deploy-targets.xml ファイルの一部を、次に示します。

```
<property environment="env"/>
<!-- Web Server のプロパティーの読み込み -->
<property environment="env"/>
<property file="../../../jwsdp-shared/bin/sjsws.props"/>
<property name="ws.home" value="${WS_HOME}"/>
<property name="ws.admin" value="${ws.home}/bin/wadm"/>
```

```
<property name="lib.sample.home" value="${basedir}/../lib"/>
<property name="build.home" value="${basedir}/build"/>
<property name="build.classes.home" value="${build.home}/classes"/>
<property name="build.war.home" value="${build.home}/war"/>
<property name="config" value="${CONFIG}"/>

<target name="deploy">
    <exec executable="${ws.admin}" vmlauncher="true">
      <arg value="add-webapp" />
      <arg value="--user=${ADMIN_USER}" />
      <arg value="--password-file=${ADMIN_PASSWORD_FILE}" />
      <arg value="--host=${ADMIN_HOST}" />
      <arg value="--port=${ADMIN_PORT}" />
      <arg value="--config=${CONFIG}" />
      <arg value="--vs=${VS}" />
      <arg value="--uri=/jaxws-${ant.project.name}" />
      <arg value="${build.war.home}/jaxws-${ant.project.name}.war" />
    </exec>

    <antcall target="commit-config" />
</target>

<target name="commit-config">
    <exec executable="${ws.admin}" vmlauncher="true">
      <arg value="deploy-config" />
      <arg value="--user=${ADMIN_USER}" />
      <arg value="--password-file=${ADMIN_PASSWORD_FILE}" />
      <arg value="--host=${ADMIN_HOST}" />
      <arg value="--port=${ADMIN_PORT}" />
      <arg value="--force=true" />
      <arg value="${CONFIG}" />
    </exec>
</target>
```

用語集

`admpw`	Enterprise Administrator Server スーパーユーザーのユーザー名とパスワードのファイル。
CGI	Common Gateway Interface (共通ゲートウェイインタフェース)。外部プログラムが HTTP サーバーと通信するために使用するインタフェース。CGI を使用するプログラムは、CGI プログラムまたは CGI スクリプトと呼ばれる。通常サーバーでは処理されないフォームや解析されない出力を、CGI プログラムが処理したり解析したりする。
chroot (chroot)	サーバーを特定のディレクトリに制限するために作成可能な追加ルートディレクトリ。この機能を使用して保護されていないサーバーを守ることができます。
ciphertext (暗号化テキスト)	暗号化によって隠蔽される情報。あらかじめ決められた受信者のみが復号化できます。
client auth	クライアント認証。
DHCP	Dynamic Host Configuration Protocol (動的ホスト構成プロトコル)。インターネット提案標準プロトコル。システムが、ネットワーク内の個々のコンピュータに IP を動的に割り当てることができるようにします。
DNS エイリアス (DNS alias)	DNS サーバーが管理している、別のホストを参照するためのホスト名。具体的には DNS の CNAME レコード。マシンの実際の名前は 1 つだけですが、1 つまたは複数のエイリアスを持つことができます。たとえば、`www.`*yourdomain.domain* というエイリアスが、現在サーバーが置かれている `realthing.`*yourdomain.domain* という実際のマシンを指すようにすることができます。
DNS (ドメインネームシステム)	Domain Name System (ドメインネームシステム)。ネットワーク上のマシンが、198.93.93.10 などのような 標準の IP アドレス を、`www.sun.com` などのようなホスト名 に関連付けるための仕組み。各マシンは通常、この変換済みの情報を DNS サーバーから取得するか、またはそれぞれのシステム上で維持されているテーブル内からその情報を検索します。
Expires ヘッダー	リモートサーバーによって指定される、返されたドキュメントの有効期限。
FORTEZZA	重要ではあっても機密扱いされない情報を管理するために、米国政府機関によって使用される暗号化システム。
FTP (ファイル転送プロトコル)	File Transfer Protocol (ファイル転送プロトコル)。ネットワーク上で 1 つのコンピュータから別のコンピュータへのファイル転送を実現するインターネットプロトコルの 1 つ。

GIF
(グラフィックス交換形式)
Graphics Interchange Format (グラフィックス交換形式)。CompuServe によって開発されたクロスプラットフォームのイメージ形式。GIF ファイルは通常、ほかのグラフィックファイルタイプ (BMP、TIFF) よりもサイズがかなり小さくなっています。GIF は、広く利用される一般的な交換形式です。GIF イメージは、UNIX、Microsoft Windows、および Apple Macintosh の各システムでそのまま表示できます。

HTML
(ハイパーテキストマークアップ言語)
Hypertext Markup Language (ハイパーテキストマークアップ言語)。World Wide Web 上のドキュメントで使用される書式設定言語。HTML ファイルは、テキストの表示方法、グラフィックスやフォーム項目の配置方法、およびほかのページへのリンクの表示方法を、Netscape Navigator などのブラウザに指示するための書式設定コードを含む、プレーンテキストファイルです。

HTTP
HyperText Transfer Protocol (ハイパーテキスト転送プロトコル)。HTTP サーバーと HTTP クライアントとの間で情報を交換するための手段。

HTTP-NG
次世代 HTTP。

HTTPD
(ハイパーテキスト転送プロトコルデーモン)
HTTP デーモンまたはサービスの略。これは、HTTP プロトコルを使用して情報を提供するプログラムです。

HTTPS
セキュリティー機能を備えた HTTP。Secure Sockets Layer (SSL) を使って実装されます。

imagemap
イメージ領域をアクティブにするプロセス。ユーザーは、イメージの異なる領域をマウスでクリックして情報を参照および取得できます。imagemap は、「imagemap」という名前の CGI プログラムを指すこともあります。これは、ほかの HTTPD 実装内で imagemap 機能を処理するために使用されます。

inittab
(UNIX)
何らかの理由で停止された場合に再起動する必要のあるプログラムのリストを含む UNIX ファイル。これは、プログラムが連続して実行されることを保証します。これは、格納されている位置に基づいて、`/etc/inittab` とも呼ばれます。UNIX システムでも、このファイルを使用できない場合があります。

IP アドレス
Internet Protocol (インターネットプロトコル) アドレス。インターネット上でのマシンの実際の場所を指定する、ドットで区切られた一連の数字 (たとえば 198.93.93.10 など)。

ISDN
サービス総合デジタル網。

ISINDEX
クライアントでの検索を有効にする HTML タグ。ドキュメントは、ネットワークナビゲータの機能を使用して検索文字列を受け入れ、それをサーバーに送信して、フォームを使用せずに検索可能インデックスにアクセスできます。<ISINDEX> を使用するには、クエリーハンドラを作成する必要があります。

ISMAP
ISMAP は、名前付きイメージが imagemap であることをサーバーに通知するために HTML ドキュメントで使用される、IMG SRC タグの拡張機能です。

ISP
Internet Service Provider (インターネットサービスプロバイダ)。インターネットへの接続を提供する組織。

Java
Sun Microsystems によって作成されたオブジェクト指向プログラミング言語。アプレットと呼ばれるリアルタイム対話型プログラムを作成するために使用されます。

JavaScript クライアントおよびサーバーのインターネットアプリケーションを開発する際に使用する、コンパクトなオブジェクトベーススクリプト言語。

JavaServer Pages インスタンス化、初期化、破棄、ほかのコンポーネントからのアクセス、および設定管理など、すべての JavaServer ページのメタ関数を可能にする拡張機能。JavaServer Pages は、Web ブラウザ内ではなく、Web サーバー上で実行される再利用可能な Java アプリケーションです。

Java サーブレット (Java Servlets) インスタンス化、初期化、破棄、ほかのコンポーネントからのアクセス、および設定管理を含む、すべての Java サーブレットのメタ関数を可能にする拡張機能。Java サーブレットは、Web ブラウザ内ではなく、Web サーバー上で実行される再利用可能な Java アプリケーションです。

Last-Modified ヘッダー HTTP 応答でサーバーから返されたドキュメントファイルの最終変更時刻。

LDAP データベース (LDAP database) 認証で使用するユーザーとグループのリストを保管するデータベース。

`magnus.conf` Web Server の主要構成ファイル。このファイルには、ポートやセキュリティーなど、グローバルのサーバー構成情報が含まれています。このファイルでは、初期化時にサーバーを構成するための変数の値が設定されています。Enterprise Sever は起動時にこのファイルを読み取り、変数の設定を実行します。サーバーを再起動しないかぎり、サーバーがこのファイルを再度読み取ることはありません。したがって、このファイルを変更するたびにサーバーを再起動する必要があります。

MD5 RSA Data Security によるメッセージダイジェストアルゴリズム。MD5 を使用すると、高い確率で一意になる短い形式のダイジェストデータを生成できます。同一のメッセージダイジェスト電子メールを生成するデータを生成することは、数学的に極めて困難です。

MD5 シグニチャー (MD5 signature) MD5 アルゴリズムによって生成されるメッセージダイジェスト。

MIB Management Information Base (管理情報ベース)。

MIME Multi-Purpose Internet Mail Extensions。マルチメディア電子メールとメッセージの新たな標準。

mime.types MIME (Multi-purpose Internet Mail Extension) タイプの構成ファイル。このファイルは、ファイル拡張子を MIME タイプにマッピングし、要求されているコンテンツのタイプをサーバーが判別できるようにします。たとえば、拡張子が `.html` のリソース要求は、クライアントが HTML ファイルを要求していることを示し、拡張子が `.gif` のリソース要求は、クライアントが GIF 形式のイメージファイルを要求していることを示します。

modutil 外部暗号化またはハードウェアアクセラレータデバイスの PKCS#11 モジュールをインストールする場合に必要なソフトウェアユーティリティー。

MTA (メッセージ転送エージェント) Message Transfer Agent (メッセージ転送エージェント)。サーバーの MTA ホストを定義し、サーバーでエージェントサービスを使用する必要があります。

NIS (UNIX) Network Information Service (ネットワーク情報サービス)。コンピュータネットワーク全域のマシン、ユーザー、ファイルシステム、およびネットワークパラメータに関する特定の情報を、収集、照合、および共有するために、UNIX システムで使用されるプログラムとデータファイルのシステム。

NNTP (ネットワークニュース転送プロトコル) ニュースグループ用の Network News Transfer Protocol (ネットワークニュース転送プロトコル)。ニュースサーバーホストを定義し、サーバーでエージェントサービスを使用する必要があります。

obj.conf サーバーのオブジェクト設定ファイル。このファイルには、補足的な初期化情報、サーバーのカスタマイズ設定、およびブラウザなどのクライアントからの要求をサーバーが処理する場合に使用する命令が記述されています。Sun Java System Web Server は、クライアント要求を処理するたびにこのファイルを読み取ります。

pk12util 内部マシンから証明書と鍵データベースをエクスポートし、外部 PKCS#11 モジュールにそれらの情報をインポートする場合に必要なソフトウェアユーティリティー。

RAM (ランダムアクセスメモリー) Random Access Memory (ランダムアクセスメモリー)。コンピュータ内の物理的な半導体ベースメモリー。

rc.2.d (UNIX) マシンの起動時に実行されるプログラムについて記述した、UNIX マシンのファイル。このファイルは、格納されている位置に基づいて、/etc/rc.2.d とも呼ばれます。

RFC Request For Comments。通常、インターネットコミュニティーに提示される、手法または標準仕様に関する文書を指します。テクノロジが標準仕様として受け入れられる前に、それらのテクノロジに関するコメントを誰でも送ることができます。

root (UNIX) UNIX マシン上でもっとも大きな特権を持つユーザー。root ユーザーは、マシンのすべてのファイルへのアクセス権限を持っています。

SNMP Simple Network Management Protocol (簡易ネットワーク管理プロトコル)。

SOCKS ファイアウォールソフトウェアまたはハードウェア (ルーター設定など) によって直接接続を回避しているときに、ファイアウォールの内側から外側に向かって接続を確立するファイアウォールソフトウェア。

SSL Secure Sockets Layer。クライアントとサーバーの間にセキュリティー保護された接続を確立するソフトウェアライブラリ。HTTP のセキュリティー保護されたバージョンである HTTPS を実装するために使用されます。

SSL 認証 (SSL authentication) 本人であることの証明としてクライアント証明書の情報を使用して、または LDAP ディレクトリで発行されたクライアント証明書を検証して、セキュリティー証明書によるユーザーのアイデンティティー確認を行います。

strftime 日付と時刻を文字列に変換する関数。これは、トレーラを追加するときにサーバーによって使用されます。strftime には、日付と時刻に使用する特殊な形式の言語があり、サーバーはその言語をトレーラで使用して、ファイルの最終変更日付を表示できます。

Sun Java System Web Server 管理コンソール
企業ネットワーク内のどこか 1 つの中央場所からすべての Sun Java System Web Server を管理するためのグラフィカルインタフェースを、サーバー管理者に提供する Java アプリケーション。Sun Java System Web Server 管理コンソールの任意のインストール済みインスタンスから、アクセス権限を許可されている企業ネットワーク上にあるすべての Sun Java System サーバーを表示およびアクセスできます。

Sym-link (UNIX)
symbolic link (シンボリックリンク) の略。これは、UNIX オペレーティングシステムが使用するリダイレクションの一種です。Sym-link を使えば、ファイルシステムのある部分からその別の部分に存在する既存のファイルまたはディレクトリへのポインタを作成できます。

TCP/IP
Transmission Control Protocol/Internet Protocol。インターネットおよびエンタープライズ (企業) ネットワークの主要なネットワークプロトコル。

telnet
ネットワーク上の 2 つのマシンを相互接続し、両者間でリモートログイン用の端末エミュレーションをサポートするプロトコル。

TLS
Secure Sockets Layer。クライアントとサーバーの間にセキュリティー保護された接続を確立するソフトウェアライブラリ。HTTP のセキュリティー保護されたバージョンである HTTPS を実装するために使用されます。

top (UNIX)
システムリソースの現在の使用状態を表示する、一部の UNIX システムに搭載されたプログラム。

uid (UNIX)
UNIX システム上で各ユーザーに関連付けられる一意の番号。

URI
Uniform Resource Identifier。省略形の URL を使用することで追加のセキュリティー層を提供するファイル識別子。URL の最初の部分が URL マッピングで置換されるため、ファイルの完全な物理パス名がユーザーからわからないようになります。「URL マッピング」も参照してください。

URL
Uniform Resource Locator。ドキュメントを要求するためにサーバーとクライアントが使用するアドレス指定方式。URL はロケーションとも呼ばれる。URL の形式は *protocol://machine:port/document*。

たとえば、`http://www.sun.com/index.html` は URL の一例。

URL データベース修復 **(URL database repair)**
ソフトウェアの障害、システムのクラッシュ、ディスクの動作停止、または満杯のファイルシステムによって損傷した URL データベースを修復および更新するプロセス。

URL マッピング **(URL mapping)**
ドキュメントディレクトリの物理パス名をユーザー定義エイリアスにマッピングするプロセス。これにより、ディレクトリ内部のファイルが、ファイルの完全物理パス名の代わりにディレクトリのエイリアスを参照するだけで済むようになります。したがって、ファイルを `usr/sun/servers/docs/index.html` として指定する代わりに、`/myDocs/index.html` として指定できます。これにより、サーバーファイルの物理的な位置をユーザーが知る必要がなくなるので、サーバーのセキュリティーが高まります。

Web アプリケーション (web application)	サーブレット、JavaServer Pages、HTML ドキュメント、およびその他の Web リソースの集合。これには、イメージファイルや圧縮されたアーカイブなどのデータが含まれることがあります。Web アプリケーションは、アーカイブ (WAR ファイル) にパッケージ化されている場合や、オープンディレクトリ構造に配備されている場合があります。
Web アプリケーションアーカイブ (WAR)	完全な Web アプリケーションを圧縮された形式で含むアーカイブファイル。
Windows CGI (Windows)	Visual Basic などの Windows ベースプログラミング言語で記述された CGI プログラム。
アクセス制御エントリ **(ACE)**	受信したアクセス要求を評価するために Web サーバーが使用する規則の階層。
アクセス制御リスト **(ACL)**	ACE の集合。ACL は、サーバーへのアクセス権を持つユーザーを定義する仕組みです。特定のファイルやディレクトリに固有の ACL 規則を定義して、1 つ以上のユーザーまたはグループへのアクセスを付与または拒否できます。
暗号化	あらかじめ決められた受信者以外のユーザーが情報を復号化したり読んだりできないように情報を変換するプロセス。
暗号化方式 **(cipher)**	暗号化方式は、暗号化と復号化に使用される暗号化アルゴリズム (数学関数) です。
インテリジェントエージェント **(intelligent agent)**	ユーザーのために、HTTP、NNTP、SMTP、FTP などのさまざまな要求を実行するサーバー内部のオブジェクト。インテリジェントエージェントは、ある意味でサーバーのクライアントとして動作し、サーバーが実行する要求を発行します。
エージェント	ルーター、ホスト、あるいは X 端末などのネットワークデバイス上でネットワーク管理ソフトウェアを実行するソフトウェア。「インテリジェントエージェント」も参照してください。
エクストラネット **(extranet)**	企業のイントラネットをインターネット領域にまで延長したネットワーク。顧客、サプライヤ、およびリモート作業者がデータにアクセスできます。
仮想サーバー **(virtual server)**	仮想サーバーは、1 つのインストールサーバーで、複数のドメイン名、IP アドレス、およびサーバー監視機能を設定するための方法です。
仮想サーバークラス **(virtual server class)**	obj.conf ファイルの同じ基本設定情報を共有する仮想サーバーの集合。
簡易インデックス **(simple index)**	凝ったインデックスの反対。このタイプのディレクトリ一覧表示では、グラフィカル要素をまったく伴わずに、ファイルの名前だけが表示されます。
管理サーバー **(Administration Server)**	すべての Sun Java System Web Server の構成に使用するフォームを備えた、Web ベースのサーバー。
危殆化鍵リスト **(CKL)**	危殆化鍵を持つユーザーに関する鍵情報のリスト。このリストも CA が提供します。

キャッシュ **(cache)**	ローカルに格納された、元のデータのコピー。データがキャッシュされていれば、要求があったときに再びリモートサーバーから取得する必要がなくなります。
共通ログファイル形式 **(Common LogFile Format)**	サーバーがアクセスログに情報を入力するために使用する形式。この形式は、Sun Java System Web Server を含むすべての主要なサーバー間で同じです。
クライアント	Netscape Navigator など、World Wide Web の情報を要求および表示するために使用するソフトウェア。
クラスタ	「マスター」管理サーバーに追加され、「マスター」管理サーバーによって制御される、リモート「スレーブ」管理サーバーのグループ。クラスタ内のすべてのサーバーは、同じプラットフォームで構築し、同じユーザー ID とパスワードを設定する必要があります。
公開鍵	公開鍵暗号方式で使用される暗号化鍵。
公開情報ディレクトリ **(UNIX)**	ドキュメントルート内に存在せず、ある UNIX ユーザーのホームディレクトリ内に存在しているディレクトリ、またはそのユーザーの制御下にあるディレクトリ。
凝ったインデックス **(fancy indexing)**	簡易インデックスよりも多くの情報を提供するインデックス化の手法。凝ったインデックスは、ファイルサイズ、最後の変更日付、およびファイルタイプを反映するアイコンと一緒に、名前ごとのコンテンツリストを表示します。このため、凝ったインデックスは、クライアントがロードするときに、簡易インデックスの場合より時間がかかります。
コレクション **(collection)**	単語のリストやファイルのプロパティーなど、ドキュメントに関する情報を含むデータベース。検索機能では、コレクションを使用して、指定された検索条件に一致するドキュメントを検索します。
サーバーデーモン **(server daemon)**	いっったん稼働中になると、クライアントからの要求を待機し、受け付けるプロセス。
サーバープラグイン API **(Server Plug-in API)**	Sun Java System Web Server の主要機能の拡張またはカスタマイズあるいはその両方を可能にするとともに、HTTP サーバーとバックエンドアプリケーションの間にインタフェースを構築する、スケーラブルで有効なメカニズムを提供する拡張機能。NSAPI とも呼ばれます。
サーバールート **(server root)**	サーバープログラム、設定、保守、および情報ファイルを保管するための、専用のディレクトリサーバーマシン上のディレクトリです。
サービス品質 **(QoS)**	サーバーインスタンス、仮想サーバークラス、または仮想サーバーに対して設定されるパフォーマンス制限。
証明書 **(certificate)**	移転や偽造の不可能なデジタルファイルで、通信する両者によって信頼済みの第三者機関から発行されたもの。
証明書失効リスト **(CRL)**	CA によって提供される、失効されたすべての証明書の CA リスト。

スーパーユーザー **(UNIX)**
UNIX マシン上で利用可能な、もっとも大きな特権を持つユーザー (root とも呼ばれる)。スーパーユーザーは、マシンのすべてのファイルへのアクセス権限を持っています。

ストップワード **(stop word)**
検索機能によって検索対象外の単語として識別される単語。これには通常、the、a、an、and などの単語が含まれます。「ドロップワード」とも呼ばれます。

ソフト再起動 **(soft restart)**
サーバーを内部で再起動させる、つまり設定ファイルを再読み込みする、サーバーの再起動方法。ソフト再起動では、プロセスに HUP シグナル (シグナル番号 1) を送信します。プロセス自体は終了せず、ハード再起動で終了します。

待機ソケット **(listen socket)**
ポート番号と IP アドレスの組み合わせ。サーバーとクライアントの間の接続は、待機ソケット上で確立されます。

ダイジェスト認証 **(digest authentication)**
ユーザー名とパスワードを平文として送信せずにユーザーの認証を可能にします。ブラウザは、MD5 アルゴリズムを使用してダイジェスト値を作成します。サーバーはダイジェスト認証プラグインを使用して、クライアントから送られてきたダイジェスト値を比較します。

タイムアウト **(timeout)**
サーバーが、ハングしたように見えるサービスルーチンの終了処理を中止するまでの指定された時間。

デーモン **(UNIX)**
ある特定のシステムタスクを担当するバックグラウンドプロセス。

ドキュメントルート **(document root)**
サーバーにアクセスするユーザーに対して表示するファイル、イメージ、データを格納した、サーバーマシン上のディレクトリ。

トップレベルドメイン権限 **(top-level domain authority)**
ホスト名の分類における最上位のカテゴリ。通常は、ドメインの組織のタイプ (たとえば、.com は企業、.edu は教育機関) またはドメインの元となる国 (たとえば、.us はアメリカ合衆国、.jp は日本、.au はオーストラリア、.fi はフィンランド) を意味します。

ドロップワード **(drop word)**
「ストップワード」を参照してください。

認証
自身のアイデンティティーをサーバーに対して検証できるようにします。基本認証またはデフォルト認証では、ユーザー名とパスワードを入力しないと、Web サーバーまたは Web サイトにアクセスできません。それは、LDAP データベース内にユーザーとグループのリストを必要とします。「ダイジェスト認証」と「SSL 認証」も参照してください。

サーバー全体またはサーバー上の特定のファイルやディレクトリへのアクセスを許可すること。承認は、ホスト名や IP アドレスなどの条件を使って制限できます。

認証局 **(CA)**
暗号化トランザクションで使用されるデジタルファイルを発行する、内部またはサードパーティーの組織。

ネットワーク管理ステーション **(NMS)**
ユーザーがネットワークのリモート管理用として使用できるマシン。管理対象デバイスは、ホスト、ルーター、Web サーバーなど、SNMP が稼動するすべてのデバイスです。NMS は通常、1 つ以上のネットワーク管理アプリケーションがインストールされた強力なワークステーションです。

ハード再起動 **(hard restart)**
プロセスまたはサービスの終了と、その後の再起動。「ソフト再起動」も参照してください。

パスワードファイル **(UNIX)**
UNIX ユーザーのログイン名、パスワード、およびユーザー ID 番号が格納された、UNIX マシン上のファイル。これは、その格納場所にちなんで `/etc/passwd` とも呼ばれます。

非公開鍵
公開鍵暗号方式で使用される復号化鍵。

ファイアウォール
組織内部のネットワーク接続されたコンピュータを、通常はハードウェアとソフトウェアの両方により、外部からのアクセスから保護するネットワーク構成。一般に、ファイアウォールは物理的な建物または組織のサイト内にある、ネットワークの電子メールやデータファイルなどの情報を保護するために使用されます。

ファイル拡張子 **(file extension)**
ファイル名の最後の部分。この部分は通常、ファイルのタイプを定義します。たとえば、ファイル名が `index.html` の場合、ファイル拡張子は `html` です。

ファイルタイプ **(file type)**
ある特定のファイルの形式。たとえば、グラフィックファイルのファイルタイプは、テキストファイルのファイルタイプとは異なります。ファイルタイプは通常、ファイル拡張子 (`.gif` や `.html` など) によって識別されます。

フレキシブルログ形式 **(flexible log format)**
アクセスログに情報を入力する場合にサーバーが使用する記録形式。

プロトコル
ネットワークのデバイスが情報を交換する方法について記述した一連の規則。

ホームページ **(home page)**
サーバーに置かれ、サーバーのコンテンツのカタログまたはエントリポイントとして動作するドキュメント。このドキュメントの位置は、サーバーの設定ファイル内で定義されます。

ホスト名 **(hostname)**
machine.domain.dom の形式で指定するマシンの名前。これは IP アドレスに変換されます。たとえば、`www.sun.com` は、`com` ドメイン内の `sun` サブドメインにある `www` というマシンを示します。

リソース
サーバーがアクセスして要求元のクライアントに送信できる、任意のドキュメント (URL)、ディレクトリ、またはプログラム。

リダイレクション **(redirection)**
特定の URL にアクセスしているクライアントを、同じサーバーまたは別のサーバーにある異なる位置に送信する際のシステム。このシステムは、リソースが移動されているときに、クライアントに新しい位置を透過的に使用させたい場合に便利です。またこれを使用して、後続のスラッシュを付けずにディレクトリへのアクセスがあった場合に、相対リンクの整合性を保持します。

索引

S

T

U

W

X