관리자 설명서

Sun™ ONE Application Server

버전 7, Update 1

816-6865-10
2003년 3월

목차

설명서 정보

본 설명서는 Sun™ ONE Application Server 7의 구성 및 관리 방법에 대해 설명하고 있으며, 클라이언트 서버 응용 프로그램을 World Wide Web을 통해 더 많은 사용자에게 확대하려는 기업의 정보 기술 관리자를 대상으로 합니다.

이 머리말은 다음 항목으로 구성되어 있습니다.

- 설명서 내용
- 설명서 구성
- 설명서 규칙
- 제품 라인 개요
- 설명서 사용
- 제품 지원

설명서 내용

본 설명서에서는 Sun ONE Application Server를 구성하고 관리하는 방법에 대해 설명합니다. 서버 구성 후에는 이 설명서를 사용하여 서버를 관리하십시오.

설명서 구성

본 설명서는 네 부분과 색인으로 구성되어 있습니다. 1부, “서버 기본 사항 및 글로벌 설정 관리”에서는 제품의 개요를 소개하고 2부, “개별 서버 인스턴스 관리”에서는 Administration Server의 사용 및 모든 서버 인스턴스에 영향을 미치는 다른 서버 기능의 사용에 대해 설명합니다.

이와 같이 Administration Server 사용에 대한 기본 개념을 설명한 후 3부, “HTTP 서버 기능 및 가상 서버 관리”에서는 프로그램 및 구성 스타일 사용에 대한 정보를 제공합니다.

마지막으로 부록에서는 국제화 문제, 서버 확장 및 Sun ONE Application Server 명령줄 인터페이스 설명서를 비롯한 여러 가지 내용을 설명하는 참조 항목을 제공합니다.

1부: 서버 기본 사항 및 글로벌 설정 관리

1부에서는 Sun ONE Application Server를 개괄적으로 설명하며 다음 내용으로 구성되어 있습니다.

- 1장, "Sun ONE Application Server 관리 시작"에서는 Sun ONE Application Server를 개괄적으로 설명합니다.
- 2장, "Administration Server 기본 설정"에서는 Administration Server를 관리하는 방법에 대해 설명합니다.
- 3장, "관리 도메인 구성"에서는 여러 개의 도메인을 사용하는 방법에 대해 설명합니다.

2부: 개별 서버 인스턴스 관리

2부에서는 서버 인스턴스의 구성, 관리 및 사용에 관한 개념과 절차에 대해 자세히 설명하며 다음 내용으로 구성되어 있습니다.

- 4장, "응용 프로그램 서버 인스턴스 사용"에서는 Sun ONE Application Server의 서버 기본 설정 구성 방법에 대해 설명합니다.
- 5장, "로깅 사용"에서는 로깅의 기초와 Sun ONE Application Server의 로깅 기능에 대해 설명합니다.

- 6장, "Sun ONE Application Server 모니터링"에서는 Sun ONE Application Server에서 사용할 수 있는 SNMP (Simple Network Management Protocol) 기능에 대한 정보를 제공합니다.
- 7장, "웹 서버 플러그 인 구성"에서는 Sun ONE Application Server에서 HTTP 요청을 처리하는 방법과 Sun ONE Application Server와 함께 웹 서버 플러그 인을 구성하고 사용하는 방법에 대해 설명합니다.
- 8장, "J2EE 컨테이너 구성"에서는 EJB (Enterprise Java Bean) 및 MDB (Message Driven Bean)와 같은 J2EE 응용 프로그램 구성 요소에 대한 런타임 지원을 제공하는 컨테이너를 구성하고 사용하는 방법에 대해 설명합니다.
- 9장, "트랜잭션 서비스 사용"에서는 데이터베이스 트랜잭션과 그 사용 방법에 대해 설명합니다.
- 10장, "이름 지정 및 자원 구성"에서는 J2EE 자원을 구성하는 방법에 대해 설명합니다.
- 11장, "JMS 서비스 사용"에서는 JMS 공급자인 Sun ONE Message Queue를 통해 기본 제공되는 JMS 서비스를 이해하고 관리하는 데 필요한 정보를 제공합니다.
- 12장, "Corba/IIOP 클라이언트를 지원하도록 서버 구성"에서는 Sun ONE Application Server 환경 내에서 RMI/IIOP 프로토콜을 사용하여 CORBA 기반 클라이언트 지원을 구성하는 방법에 대해 설명합니다.
- 13장, "응용 프로그램 배포"에서는 Sun ONE Application Server에 응용 프로그램을 배포하는 방법에 대해 설명합니다.

3부: HTTP 서버 기능 및 가상 서버 관리

3부에서는 프로그램과 구성 스타일에 대한 관리 인터페이스 사용에 대한 정보를 제공하며 다음 내용으로 구성되어 있습니다.

- 14장, "HTTP 기능 구성"에서는 Sun ONE Application Server의 HTTP 관련 기능에 대한 기본 설정을 구성하는 방법에 대해 설명합니다.
- 15장, "가상 서버 사용"에서는 Sun ONE Application Server를 사용하여 가상 서버를 설정하고 관리하는 방법에 대해 설명합니다.
- 16장, "가상 서버 내용 관리"에서는 서버의 내용을 구성하고 관리하는 방법에 대해 설명합니다.

4부: 부록

4부에서는 여러 가지 부록과 참조 자료를 제공하며 다음 내용으로 구성되어 있습니다.

- 부록 A, "명령줄 인터페이스 사용"에서는 사용자 인터페이스 화면 대신 명령줄 유틸리티를 사용하는 지침을 제공합니다.
- 부록 B, "타사 저작권 고지"에는 추가 저작권 정보가 포함되어 있습니다.

설명서 규칙

이 절에서는 이 설명서 전체에 사용된 규칙에 대해 설명합니다.

- 일반 규칙
- 디렉토리 참조 규칙

일반 규칙

이 설명서에는 다음과 같은 일반적인 규칙이 사용됩니다.

- **파일과 디렉토리 경로**는 슬래시를 사용하여 디렉토리 이름을 구분하는 UNIX® 형식으로 표시됩니다. Windows 버전에서는 동일한 디렉토리 경로를 사용하지만 슬래시 대신 백슬래시를 사용하여 디렉토리를 구분합니다.
- **URL**은 다음 형식으로 표시됩니다.

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

  URL에서 *server*는 응용 프로그램이 실행되는 서버 이름, *domain*은 사용자의 인터넷 도메인 이름, *path*는 서버의 디렉토리 구조이며 *file*은 각 파일의 이름입니다. URL에서 기울임꼴로 표시된 항목은 자리 표시자입니다.
- **글꼴 규칙**
  - 고정 폭 글꼴은 샘플 코드 및 코드 목록, API 및 언어 요소(함수 이름, 클래스 이름 등), 파일 이름, 경로 이름, 디렉토리 이름 및 HTML 태그에 사용됩니다.
  - *기울임꼴*은 코드 변수에 사용됩니다.
  - 책 제목, 강조, 변수 및 자리 표시자, 단어의 문자적 의미를 표시하는 데도 사용됩니다.

- **굵게** 글꼴은 단락 처음 부분이나 단어의 문자적 의미를 나타내는 데 사용됩니다.

- 이 문서에서 **설치 루트 디렉토리는** 대부분의 플랫폼에서 *install_dir*로 표시됩니다. 예외 사항은 21페이지의 "디렉토리 참조 규칙"을 참조하십시오.

  기본적으로 **대부분의** 플랫폼에서 *install_dir*의 위치는 다음과 같습니다.

  - Solaris 8 비패키지 기반 평가판 설치:

    user's home directory`/sun/appserver7`

  - Solaris 번들되지 않은 비평가판 설치:

    `/opt/SUNWappserver7`

  - Windows, 모든 설치:

    `C:\Sun\AppServer7`

  위에 나열된 플랫폼의 경우, *default_config_dir*과 *install_config_dir*은 *install_dir*과 동일합니다. 예외 사항 및 추가 정보는 21페이지의 "디렉토리 참조 규칙"을 참조하십시오.

- **인스턴스 루트 디렉토리**는 이 문서에서 *instance_dir*로 표시되며 그 약어는 다음과 같습니다.

  default_config_dir`/domains/`*domain*`/`*instance*

- **UNIX 특정 사항에 대한 설명**은 별도로 언급된 경우를 제외하면 Linux 운영 체제에도 동일하게 적용됩니다.

디렉토리 참조 규칙

Solaris 8 및 9 패키지 기반 설치와 Solaris 9 번들 설치 시 응용 프로그램 서버 파일은 기본적으로 여러 개의 루트 디렉토리에 분산됩니다. 이 절에서는 이러한 디렉토리에 대해 설명합니다.

- **Solaris 9 번들 설치**의 경우 제공되는 여러 기본 설치 디렉토리에 적용되는 해당 문서 규칙은 다음과 같습니다.

  - *install_dir*은 설치 이미지의 정적인 부분이 있는 `/usr/appserver/`를 참조합니다. 여기에는 응용 프로그램 서버를 구성하는 모든 유틸리티, 실행 파일 및 라이브러리가 있습니다.

  - *default_config_dir*은 생성된 모든 도메인의 기본 위치인 `/var/appserver/domains`를 참조합니다.

- *install_config_dir*은 사용권 및 이 설치에 구성된 관리 도메인의 마스터 목록과 같은 설치 전반에 관련된 구성 정보가 있는 `/etc/appserver/config`를 참조합니다.

- **번들화되지 않은 비평가판 Solaris 8 및 9 패키지 기반 설치**인 경우 여러 기본 설치 디렉토리에 적용되는 해당 문서 규칙은 다음과 같습니다.
  - *install_dir*은 설치 이미지의 정적인 부분을 포함하는 `/opt/SUNWappserver7`을 참조합니다. 여기에는 응용 프로그램 서버를 구성하는 모든 유틸리티, 실행 파일 및 라이브러리가 있습니다.
  - *default_config_dir*은 생성된 모든 도메인의 기본 위치인 `/var/opt/SUNWappserver7/domains`를 참조합니다.
  - *install_config_dir*은 사용권 및 이 설치에 구성된 관리 도메인의 마스터 목록과 같은 설치 전반에 관련된 구성 정보를 포함하는 `/etc/opt/SUNWappserver7/config`를 참조합니다.

제품 라인 개요

Sun ONE Application Server 7은 표준 HTTP 서버 프로그래밍 기능은 물론 새롭게 등장한 Java Web Service 표준을 지원하는 J2EE 1.3 사양 호환 응용 프로그램 서버입니다. 제품 사용 환경과 개발 환경의 다양한 필요에 따라 다음 세 가지의 응용 프로그램 서버가 제공됩니다.

- 플랫폼판
- 표준판
- 엔터프라이즈판

플랫폼판

플랫폼판은 Sun ONE Application Server 7 제품 라인의 핵심을 구성합니다. 이 제품은 생산용으로 무료 제공되며 기본 운영 배포 및 타사 응용 프로그램에 내장에 매우 적합한 고성능, 소용량의 J2EE 1.3 사양 호환 런타임 환경을 제공합니다. 웹 서비스를 갖춘 플랫폼판에는 Sun ONE Web Server 및 Sun ONE Message Queue 제품에 의해 입증된 기본 제공 기술이 있습니다.

플랫폼판 배포는 단일 응용 프로그램 서버 인스턴스, 즉, Java 플랫폼("Java 가상 머신" 또는 "JVM™")의 단일 가상 시스템으로 제한됩니다. 플랫폼판에서는 다중 계층 배포 토폴로지를 지원하지만 웹 서버 계층 프록시에서 로드 균형 조정을 수행하지 않습니다. 플랫폼판의 관리 유틸리티는 로컬 클라이언트로 제한됩니다.

플랫폼판은 Solaris 9에 통합되어 있습니다.

표준판

표준판은 본 *Getting Started Guide*에서 중점적으로 다루어집니다. 표준판은 플랫폼판에 향상된 원격 관리 기능을 추가합니다. 향상된 관리 기능, 원격 명령줄 및 웹 기반 관리가 표준판의 일부 기능으로 포함되어 있으며 웹 서버 계층 프록시를 통한 웹 응용 프로그램 트래픽 분할 기능도 포함됩니다. 표준판에서는 시스템당 여러 개의 응용 프로그램 서버 인스턴스(JVM) 구성을 지원합니다.

엔터프라이즈판

엔터프라이즈판은 가장 복잡한 J2EE 기반 응용 프로그램 배포에 적합한 고가용성, 로드 균형 조정 및 클러스터링 기능으로 핵심 응용 프로그램 서버 플랫폼을 향상시킵니다. 엔터프라이즈판에서는 표준판의 관리 기능을 확대하여 다중 인스턴스 및 다중 시스템 배포가 가능합니다.

클러스터링 지원은 구성이 쉬운 복제된 응용 프로그램 서버 인스턴스 그룹을 포함하여 클라이언트 요청에 대한 로드 균형을 조정할 수 있습니다. 엔터프라이즈판은 외장 로드 균형 조정기 및 로드 균형 조정 웹 계층 기반 프록시를 모두 지원하며, HTTP 세션, Stateful Session Bean 인스턴스 및 JMS (Java Message Service) 자원 장애 복구가 포함되어 있습니다. 특허된 "Always On" 고가용성 데이터베이스 기술은 엔터프라이즈판의 HA 영구 저장소의 기초를 형성합니다.

자세한 제품 정보는 Sun Microsystems 웹 사이트의 Sun ONE Application Server 페이지를 참조하십시오.

설명서 사용

Sun ONE Application Server 설명서는 다음 웹 사이트에서 PDF (Portable Document Format)와 HTML (Hypertext Markup Language) 형식의 온라인 파일로 제공됩니다.

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

다음 표는 Sun ONE Application Server 설명서에 포함된 작업 및 개념의 목록입니다. 왼쪽 열은 작업과 개념, 오른쪽 열은 해당 설명서입니다.

Sun ONE Application Server 설명서 로드맵

정보	설명서
소프트웨어 및 설명서 최신 정보	*릴리스 노트*
지원되는 플랫폼 및 환경	*플랫폼 요약*
새로운 기능, 평가판 설치 정보 및 구조 개요를 비롯한 응용 프로그램 서버 소개	*Getting Started Guide*
Sun ONE Application Server 및 구성 요소(샘플 응용 프로그램, 관리 인터페이스, Sun ONE Message Queue) 설치	*설치 설명서*
응용 프로그램 디자인, 개발자 도구, 보안, 어셈블리, 배포, 디버깅 및 lifecycle 모듈 작성에 대한 일반 정보를 비롯 Sun ONE Application Server 7에서 개방형 Java 표준 모델을 따르는 J2EE 응용 프로그램 작성 및 구현 정보	*Developer's Guide*
Sun ONE Application Server 7에서 웹 응용 프로그램을 위한 개방형 Java 표준 모델을 따르는 J2EE 응용 프로그램 작성 및 구현. 웹 응용 프로그래밍 개념 및 작업 설명 및 샘플 코드, 구현 팁 및 참조 자료 제공	*Developer's Guide to Web Applications*
Sun ONE Application Server 7에서 Enterprise Bean을 위한 개방형 Java 표준 모델을 따르는 J2EE 응용 프로그램 작성 및 구현. Discusses EJB 프로그래밍 개념 및 작업 설명, 샘플 코드, 구현 팁 및 참조 자료 제공	*Developer's Guide to Enterprise JavaBeans Technology*
Creating clients that access J2EE applications on the Sun ONE Application Server 7에서 J2EE 응용 프로그램에 액세스하는 클라이언트 작성	*Developer's Guide to Clients*

Sun ONE Application Server 설명서 로드맵*(계속)*

정보	설명서
웹 서비스 작성	*Developer's Guide to Web Services*
JDBC, JNDI, JTS, JMS, JavaMail, 자원 및 커넥터 등의 J2EE 기능	*Developer's Guide to J2EE Features and Services*
사용자 정의 NSAPI 플러그 인 작성	*Developer's Guide to NSAPI*
다음 관리 작업 수행 • 관리 인터페이스 및 명령줄 인터페이스 사용 • 서버 기본 설정 구성 • 관리 도메인 사용 • 서버 인스턴스 사용 • 서버 활동 모니터링 및 로깅 • 웹 서버 플러그 인 구성 • JMS (Java Message Service) 구성 • J2EE 기능 사용 • CORBA 기반 클라이언트 지원 구성 • 데이터베이스 연결 구성 • 트랜잭션 관리 구성 • 웹 컨테이너 구성 • 응용 프로그램 배포 • 가상 서버 관리	*Administrator's Guide*
서버 구성 파일 편집	*Administrator's Configuration File Reference*
Sun ONE Application Server 7 운영 환경에 대한 보안 구성 및 관리 일반 보안, 인증서, SSL/TLS 암호화 정보 및 HTTP 서버 기반 보안 포함	*Administrator's Guide to Security*
Sun ONE Application Server 7용 J2EE CA 커넥터를 위한 서비스 공급자 구현 구성 및 관리. 관리 도구, DTD 정보 및 XML 파일 샘플 제공	*J2EE CA Service Provider Implementation Administrator's Guide*

Sun ONE Application Server 설명서 로드맵*(계속)*

정보	설명서
Sun ONE Application Server와 함께 제공된 Online Bank 응용 프로그램의 샘플 이전을 포함한 Netscape Application Server 버전 2.1에서 새 Sun ONE Application Server 7 프로그래밍 모델로의 응용 프로그램 이전	*Migrating and Redeploying Server Applications Guide*
Sun ONE Message Queue 사용	Sun ONE Message Queue 설명서 위치: http://docs.sun.com/

제품 지원

시스템에 문제가 발생한 경우 다음 방법 중 하나를 사용하여 고객 지원부에 문의하십시오.

- 온라인 지원 웹 사이트

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

- 유지 관리 계약 관련 전화 번호

문제 해결을 위한 최상의 지원을 받으시려면 고객 지원 부서에 문의하기 전 미리 다음 사항을 준비하시기 바랍니다.

- 문제 발생 상황, 작동에 미친 영향 등 문제에 대한 자세한 설명
- 시스템 유형, 운영 체제 버전, 제품 버전 및 문제에 영향을 미칠 수 있는 패치와 기타 소프트웨어
- 문제를 복제하는 데 사용한 방법의 세부 단계
- 오류 로그 또는 코어 덤프

서버 기본 사항 및 글로벌 설정 관리

1장, “Sun ONE Application Server 관리 시작"

2장, “Administration Server 기본 설정"

3장, “관리 도메인 구성"

Sun ONE Application Server 관리 시작

이 장에서는 Sun ONE Application Server의 관리에 대한 기본적인 내용, 즉 서버에 대한 일반 정보를 찾는 위치, 서버의 사용자 인터페이스에 액세스하는 방법 및 온라인 도움말에 액세스하는 방법에 대해 설명합니다.

이 장은 다음 내용으로 구성되어 있습니다.

- Sun ONE Application Server 정보
- Solaris 번들 버전 구성
- 관리 인터페이스 사용
- 명령줄 인터페이스 사용
- Administration Server 액세스
- 응용 프로그램 서버 인스턴스 액세스
- Sun ONE Studio 사용
- 구성 파일 정보
- 사용권 명령 사용

Sun ONE Application Server 정보

Sun ONE Application Server는 다양한 서버, 클라이언트 및 장치에 전자 상거래 응용 프로그램 서비스의 개발, 배포 및 관리를 위한 강력한 J2EE 플랫폼을 제공합니다. 주요 기능으로는 트랜잭션 관리, 성능, 확장성, 보안 및 엔터프라이즈 응용 프로그램 통합 등이 있습니다.

Sun ONE Application Server는 웹 게시부터 엔터프라이즈급 트랜잭션 처리까지의 서비스를 지원하며 개발자가 JavaServer Pages (JSP™), Java Servlet 및 Enterprise JavaBeans™(EJB™) 기술을 바탕으로 응용 프로그램을 작성할 수 있도록 합니다.

여기에는 관리자를 위한 다음의 기본 도구가 포함됩니다.

- 여러 관리자가 각자의 응용 프로그램 서버 인스턴스를 만들고 관리할 수 있는 여러 관리 도메인
- 관리 기능을 제공하는 Administration Server (도메인 당 하나의 Administration Server)
- 서버 관리를 위한 그래픽 사용자 인터페이스(관리 인터페이스)
- 관리 인터페이스와 같은 작업을 수행할 수 있는 명령줄 인터페이스

이러한 도구를 사용하면 다음을 포함한 모든 관리 작업을 수행할 수 있습니다.

- 도메인 관리
- 서버 인스턴스 관리
- 응용 프로그램 배포
- 서버 모니터링
- 로그 파일 사용
- 자원 관리
- 메시지 대기열 서버 관리
- 트랜잭션 서비스 사용
- Corba/IIOP 클라이언트 사용
- 웹 서버 플러그 인 구성
- J2EE 컨테이너 구성
- HTTP 서버 기능 관리

Sun ONE Application Server 아키텍처 및 기능에 대한 자세한 내용과 Sun ONE Application Server 사용을 시작하기 위한 초기 단계에 대한 내용은 *Sun ONE Application Server Getting Started Guide*를 참조하십시오.

Solaris 번들 버전 구성

이 설명서에서는 Solaris에 대한 두 가지 종류의 Sun ONE Application Server 설치인, Solaris 9 번들 및 비번들에 대해 설명합니다. Sun ONE Application Server를 Solaris 9 설치의 일부로 받은 경우는 Solaris 번들 버전을 받은 것입니다. Sun ONE Application Server의 독립형 버전이 비번들 버전입니다.

주 Sun ONE Application Server의 번들되지 않은 Solaris를 사용하거나 Windows 버전을 사용하는 경우 이 단원을 생략하고 33페이지의 “관리 인터페이스 사용”으로 이동하십시오.

Solaris 9에 번들된 Sun ONE Application Server 버전을 사용하는 경우 서버 사용을 시작하기 전에 몇 가지 추가 구성 단계를 수행해야 합니다.

설치 프로세스의 일부로 Sun ONE Application Server의 비번들 버전을 설치하면 도메인, Administration Server 및 서버 인스턴스가 자동으로 설치됩니다.

Solaris 9 번들 버전을 사용할 때는 다른 단계를 수행하고 이러한 항목을 수동으로 작성해야 합니다. 이러한 초기 단계를 수행한 다음에는 추가 관리 도메인 및 서버 인스턴스를 추가하는 등의 모든 Sun ONE Application 서버 기능의 장점을 활용할 수 있습니다.

이 절에서는 다음 내용을 설명합니다.

- 관리 도메인 작성
- Administration Server 시작
- 응용 프로그램 서버 인스턴스 작성
- 응용 프로그램 배포

관리 도메인 작성

여러 관리 도메인을 사용하여 여러 관리 사용자가 각각의 도메인을 만들어 관리할 수 있습니다. 도메인은 단일 시스템에 설치된 공통 이진 세트를 사용하여 작성된 인스턴스 세트입니다. 각 도메인에는 하나의 Administration Server가 있습니다.

새 도메인을 만들 때는 다음을 지정합니다.

- Administration Server의 포트 번호. 번들 버전을 설치할 때의 기본값은 4848입니다.
- 관리 사용자 이름 및 암호. 관리 인터페이스에 액세스할 때 또는 명령줄 인터페이스를 실행할 때 Administration Server에 액세스하기 위해 필요한 암호입니다.
- 도메인 위치

명령줄 인터페이스의 `asadmin` 유틸리티의 `create-domain` 명령을 사용하여 도메인을 작성해야 합니다. 관리 도메인 작성에 대한 자세한 내용은 3장, "관리 도메인 구성"을 참조하십시오.

Administration Server 시작

관리 도메인을 만들 때는 Administration Server를 만들어야 합니다. Administration Server는 명령줄 인터페이스에 대한 관리 기능을 제공하며 관리 인터페이스를 제공하는 Sun ONE Application Server의 특수한 인스턴스입니다.

관리 인터페이스를 사용하거나 명령줄 인터페이스에서 많은 명령을 사용하려면 Administration Server를 실행해야 합니다. Administration Server 시작에 대한 자세한 내용은 47페이지의 “Administration Server 시작”을 참조하십시오.

응용 프로그램 서버 인스턴스 작성

도메인을 만들고 Administration Server를 시작한 뒤에는 응용 프로그램 서버 인스턴스를 작성해야 합니다. 각 응용 프로그램 서버 인스턴스에는 자체 J2EE 구성, J2EE 자원, 응용 프로그램 배포 영역 및 서버 구성 설정이 있습니다.

관리 인터페이스 또는 명령줄 인터페이스를 통해 응용 프로그램 서버 인스턴스를 작성할 수 있습니다. 서버 인스턴스는 도메인 내의 폴더에 작성됩니다.

비번들 버전에서 설치 시 작성된 서버 인스턴스를 `server1`이라고 합니다. 이 문서의 예에서 `server1`이 사용되는 것을 자주 볼 수 있습니다.

응용 프로그램 서버 인스턴스 작성에 대한 자세한 내용은 78페이지의 “응용 프로그램 서버 인스턴스 추가”를 참조하십시오.

응용 프로그램 배포

도메인을 만들고 Administration Server를 시작한 다음 응용 프로그램 서버 인스턴스를 추가한 뒤에는 해당 인스턴스에 응용 프로그램을 배포할 수 있습니다. 자세한 내용은 13장, "응용 프로그램 배포"를 참조하십시오.

관리 인터페이스 사용

관리 인터페이스를 사용하면 서버의 모든 측면을 구성할 수 있습니다. 이 절에서는 다음 내용을 설명합니다.

- 관리 인터페이스 액세스
- 탭 사용
- 버튼 사용
- 온라인 도움말 액세스
- 관리 인터페이스 종료

주 서버 구성 및 해당 관리 인터페이스의 일부는 아직 변경이 진행 중입니다. 후속 제품 릴리스에서 불안정한 인터페이스가 제거되고 더욱 명확하고 안정적인 버전으로 교체될 수 있습니다. 대부분의 서버 구성 및 관리 인터페이스는 동일하게 유지되거나 호환 가능한 방식으로 변경되지만, 일부 측면은 호환되지 않는 방식으로 변경될 수 있습니다. 이러한 비호환성에 대해서는 후속 릴리스에 대한 제품 문서에서 명확하게 설명합니다.

관리 인터페이스 액세스

Sun ONE Application Server의 관리 인터페이스는 Administration Server라고 하는 HTTP 서버에서 실행됩니다. 비번들 버전을 사용하는 경우 Administration Server는 Sun ONE Application Server를 설치할 때 함께 설치됩니다. 번들 버전을 설치하는 경우 관리 도메인과 Administration Server를 만들어야 합니다. 자세한 내용은 31페이지의 “Solaris 번들 버전 구성”을 참조하십시오.

관리 인터페이스를 사용하려면 Administration Server가 실행 중이어야 합니다. Administration Server 시작에 대한 자세한 내용은 47페이지의 “Administration Server 시작”을 참조하십시오.

Sun ONE Application Server를 설치하거나 도메인을 작성할 때 Administration Server의 포트 번호를 선택하거나 기본 포트인 4848을 사용하게 됩니다. 관리 인터페이스에 액세스하려면 웹 브라우저에서 다음을 입력합니다.

http://*hostname.domain*:*port*/

예:

http://austen.sun.com:4848/

Sun ONE Application Server를 설치한 시스템에서 Administration Server에 액세스하는 경우에는 다음을 사용할 수 있습니다.

http://*localhost*:4848

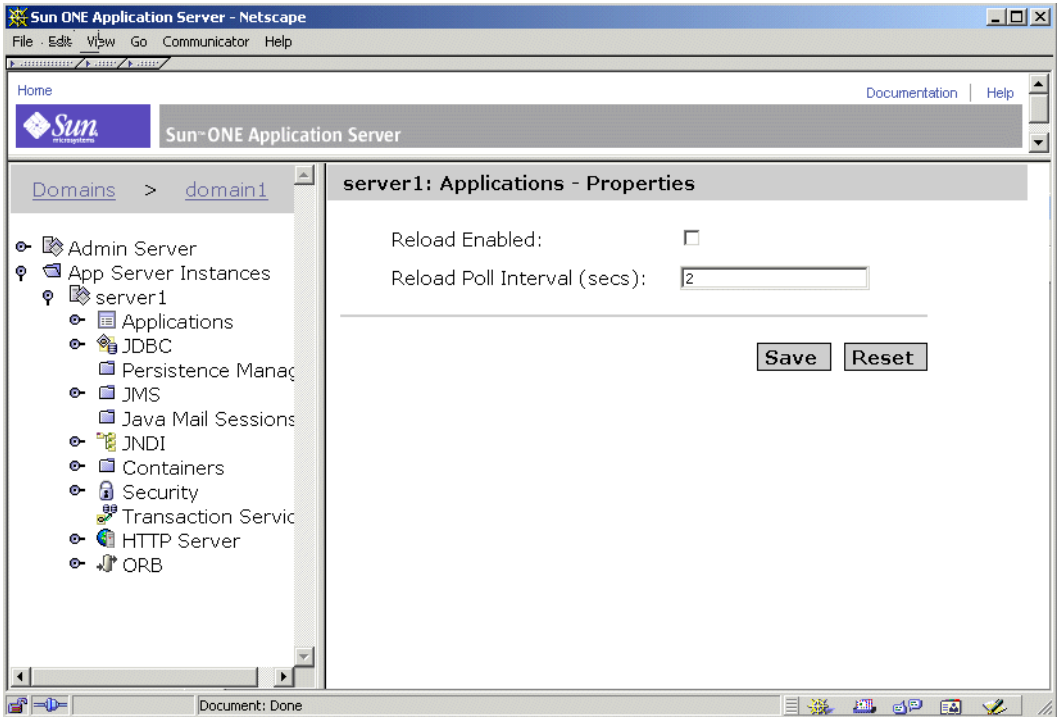
사용자 이름 및 암호를 묻는 메시지가 나타납니다. 이 사용자 이름은 설치 도중 또는 도메인 및 Administration Server를 작성할 때 설정한 사용자입니다.

브라우저에 액세스할 수만 있으면 원격 위치에서 관리 인터페이스에 액세스할 수 있습니다. 네트워크를 통해 서버에 연결할 수 있는 모든 시스템에서 액세스할 수 있습니다.

Windows에서는 "시작", "프로그램", "Sun Microsystems", "Sun ONE Application Server 7", "관리 콘솔 시작"을 선택하여 관리 인터페이스에 액세스할 수 있습니다.

다음 그림은 관리 인터페이스를 보여줍니다.

Sun ONE Application Server 관리 인터페이스



왼쪽 창은 Sun ONE Application Server에서 구성할 수 있는 모든 항목의 트리 보기입니다. 관리 인터페이스를 사용하려면 왼쪽 창에서 항목을 누릅니다. 오른쪽 창에는 해당 항목에 관련된 페이지가 표시됩니다.

왼쪽 창의 항목 옆에 열기/닫기 기호가 있으면 열기/닫기 기호를 눌러서 해당 항목을 하위 항목들로 확장할 수 있습니다. 트리 항목이 확장되지 않은 경우 기호의 모양은 다음과 같습니다.

닫힌 기호

트리 항목이 확장된 경우 기호의 모양은 다음과 같습니다.

열린 기호

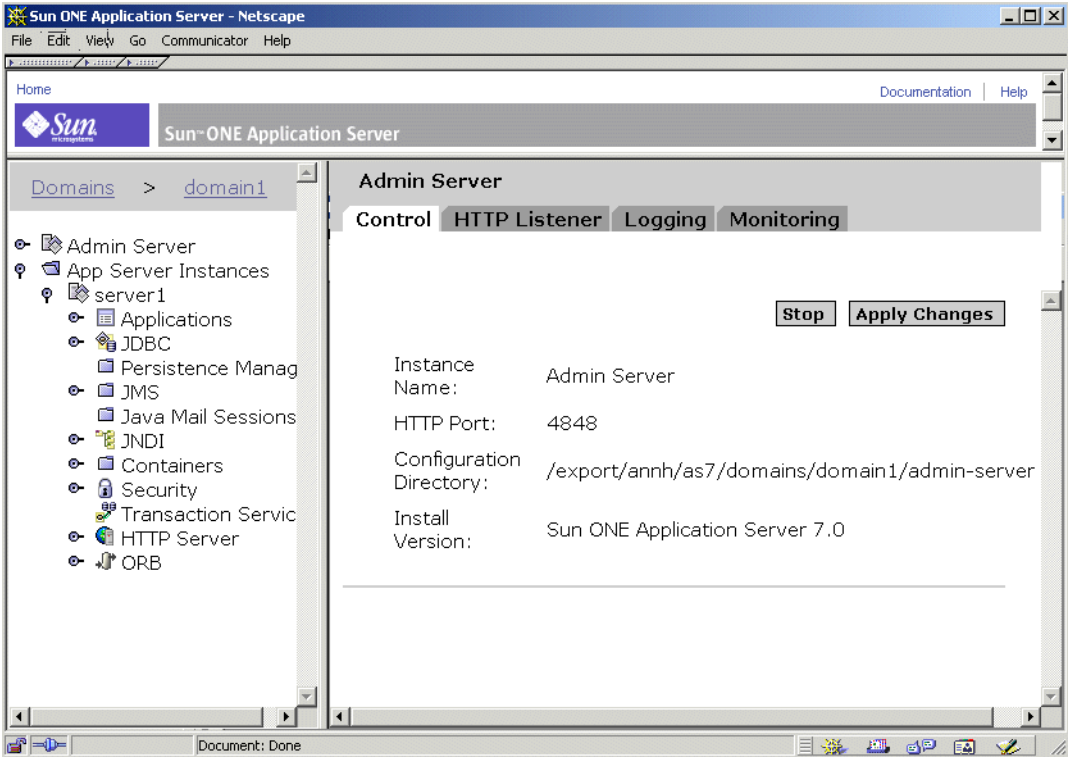
탭 사용

관리 인터페이스의 일부에는 다른 페이지로 탐색할 때 사용하는 탭이 있습니다. 이러한 탭은 오른쪽 창 맨위에 있는 별도 창에 나타납니다.

탭을 사용하려면 탭 이름을 누르십시오. 일부 탭을 누르면 바로 페이지가 나타나며 일부 탭을 누르면 탭 이름 아래에 나타나는 탭에서 사용 가능한 페이지 목록이 나타납니다. 페이지 이름을 누르면 해당 페이지로 이동합니다.

다음 그림은 탭이 있는 관리 인터페이스를 보여줍니다.

탭이 있는 관리 인터페이스



버튼 사용

관리 인터페이스에서는 다음 표준 버튼을 사용할 수 있습니다.

표준 관리 인터페이스 버튼

버튼	수행한 작업
취소	변경 내용을 저장하지 않고 취소한 다음 이전 페이지로 돌아갑니다.
삭제	항목을 삭제합니다. 항목 옆의 "선택"을 눌러서 "삭제"를 눌렀을 때 삭제할 항목을 지정할 수 있습니다.
새로 만들기	새 항목을 만들 수 있는 페이지가 표시됩니다. 예를 들어, "응용 프로그램 서버 인스턴스" 페이지에서 "새로 만들기"를 누르면 "새 인스턴스 작성" 페이지가 표시됩니다.

표준 관리 인터페이스 버튼

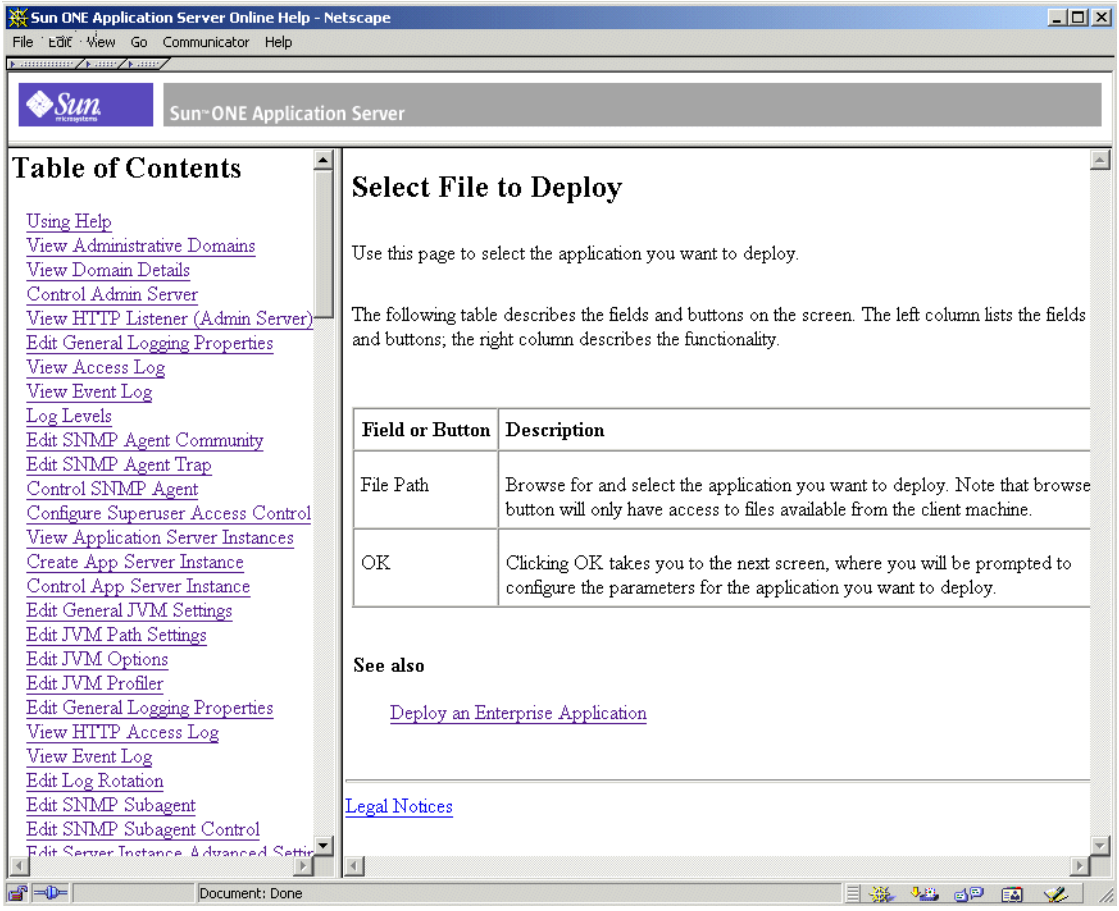
버튼	수행한 작업
확인	항목을 저장합니다. Sun ONE Application Server 인스턴스의 구성을 변경한 경우 변경 내용을 적용해야 합니다. 자세한 내용은 79페이지의 “응용 프로그램 서버 인스턴스에 변경 사항 적용”을 참조하십시오.
재설정	페이지의 값은 기본값으로 재설정합니다.
저장	항목을 저장합니다. Sun ONE Application Server 인스턴스의 구성을 변경한 경우 변경 내용을 적용해야 합니다. 자세한 내용은 79페이지의 “응용 프로그램 서버 인스턴스에 변경 사항 적용”을 참조하십시오.

특정 화면의 필요에 따라 다른 버튼을 사용할 수 있습니다.

온라인 도움말 액세스

관리 인터페이스의 맨 위에 있는 배너에서 "도움말" 버튼을 누르면 관리 인터페이스의 모든 페이지에 대한 도움말에 액세스할 수 있습니다. 온라인 도움말에서는 액세스하고 있는 페이지의 사용법을 설명하고 페이지의 필드에 입력할 내용에 대한 정보를 제공합니다.

온라인 도움말



도움말 창의 왼쪽에 있는 목차를 사용하여 다른 페이지에 대한 도움말을 탐색할 수 있습니다. 도움말 사용 방법에 대한 도움말은 온라인 도움말 목차의 첫 번째 항목인 "도움말 사용"을 참조하십시오.

관리 인터페이스 종료

관리 인터페이스에는 명시적인 종료 또는 로그 아웃 버튼이 없습니다. 종료하려면 관리 인터페이스에 액세스하는 데 사용하고 있는 브라우저를 닫으십시오. 또한 시스템에서 실행 중인 같은 브라우저의 다른 인스턴스도 닫습니다.

명령줄 인터페이스 사용

Sun ONE Application Server에는 명령줄 인터페이스가 포함되어 있습니다. `asadmin` 유틸리티 및 이에 연관된 명령을 사용하여 관리 인터페이스에서 수행할 수 있는 것과 동일한 작업을 수행할 수 있습니다. 예를 들어, 응용 프로그램 서버 인스턴스를 시작 및 중지하고, 서버를 구성하며, 응용 프로그램을 배포할 수 있습니다.

이러한 명령은 셸의 명령 프롬프트에서 사용하거나 다른 스크립트 및 프로그램에서 호출할 수도 있습니다. 이런 명령을 사용하여 반복적인 관리 작업을 자동화할 수 있습니다.

명령줄 인터페이스 사용에 대한 자세한 내용과 명령 목록은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

Administration Server 액세스

Administration Server는 관리 인터페이스를 제공하며 관리 인터페이스 및 명령줄 인터페이스에 대한 관리 기능을 제공하는 Sun ONE Application Server의 특수한 인스턴스입니다. Administration Server에서는 이러한 인터페이스 및 도구에 대한 기능을 구성, 배포 및 모니터링하며 이 서버를 실행해야 해당 인터페이스와 도구가 작동합니다.

Administration Server 등록 정보를 구성하려면 관리 인터페이스에 액세스합니다. 왼쪽 창에서 "Admin Server"를 눌러서 Administration Server의 구성 설정을 표시합니다.

Administration Server에 대한 자세한 내용은 2장, "Administration Server 기본 설정"을 참조하십시오.

응용 프로그램 서버 인스턴스 액세스

여러 개의 Sun ONE Application Server 인스턴스를 실행할 수 있습니다. 각 응용 프로그램 서버 인스턴스마다 다른 응용 프로그램 서버 인스턴스와는 독립적인 구성, 자원 및 응용 프로그램 배포 영역이 있습니다. 따라서 한 응용 프로그램 서버 인스턴스의 구성을 변경하더라도 다른 응용 프로그램 서버 인스턴스의 구성은 변경되지 않습니다. 관리 인터페이스에는 작성한 모든 응용 프로그램 서버 인스턴스에 대한 항목이 포함되어 있습니다. 소프트웨어 번들 버전을 사용하는 경우 설치할 때 하나의 응용 프로그램 서버 인스턴스가 작성됩니다. 원하는 경우 다른 인스턴스를 작성할 수 있습니다.

Solaris 9 번들 버전을 사용하는 경우 응용 프로그램 서버 인스턴스가 자동으로 작성되지 않습니다. 자세한 내용은 31페이지의 “Solaris 번들 버전 구성”을 참조하십시오.

응용 프로그램 서버 인스턴스에 대한 자세한 내용은 4장, "응용 프로그램 서버 인스턴스 사용"을 참조하십시오.

Sun ONE Studio 사용

응용 프로그램을 배포하려면 관리 인터페이스와 명령줄 인터페이스를 사용하는 것 이외에도 Sun ONE Studio 4도 사용할 수 있습니다. Sun ONE Studio에 대한 자세한 내용은 http://docs.sun.com의 *Sun ONE Studio 4, Enterprise Edition for Java with Application Server 7 Tutorial*을 참조하십시오.

구성 파일 정보

관리 인터페이스 또는 명령줄 인터페이스를 통해 서버 설정을 변경하면 Administration Server는 그에 따라 구성 파일을 변경합니다. 이들 파일에는 Administration Server 및 개별 응용 프로그램 서버 인스턴스에 대한 설정이 들어 있습니다.

파일과 각 파일에 포함된 설정에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

사용권 명령 사용

Sun ONE Application Server를 구입하면 설치할 때 적절한 유형의 사용권이 자동으로 설치됩니다. 사용권의 유형과 제한에 대한 자세한 내용은 *Sun ONE Application Server 설치 설명서*를 참조하십시오. ZSun ONE Application Server에는 설치 후 사용권을 관리할 수 있는 명령줄 유틸리티가 포함되어 있습니다.

설치 후 사용권을 업그레이드하려면 Windows 시스템에서 `asadmin` 유틸리티의 `install-license` 명령을 사용할 수 있습니다. 이 명령은 Sun ONE Application Server를 설치한 시스템에서 로컬로 실행해야 합니다. 사용권을 설치하기 전에 실행 중인 모든 응용 프로그램 서버를 중지해야 합니다.

구문은 다음과 같습니다.

```
asadmin install-license
```

번들화되지 않은 Solaris 패키지 기반 설치의 경우 다음 구문을 사용하여 사용권을 설치합니다.

`pkgadd -d` *full_path* SUNWaslco

예:

pkgadd -d /*install_dir*/pkg SUNWaslco

Solaris 9 번들 사용권 설치의 경우 다음 구문을 사용하여 사용권을 설치합니다.

`pkgadd -d` *full_path* SUNWaslc

사용권에 대한 정보를 보려면 다음 구문을 사용하는 `display-license` 명령을 사용합니다.

```
asadmin display-license [--user admin_user] [--password admin_password]
[--passwordfile password_file][--host localhost] [--port admin_port]
[--local=true/false]
```

이 명령은 로컬 옵션의 값에 따라 로컬 또는 원격에서 실행할 수 있습니다. 예를 들어, 호스트와 포트 번호의 기본값을 사용하여 로컬 시스템에서 명령을 실행하려면 다음 구문을 사용합니다.

```
asadmin display-license --local
```

출력에는 현재 설치된 사용권의 유형(예를 들어, 평가판), 만료 날짜(있는 경우), 사용권이 허용하는 Administration Server 당 인스턴스의 수 및 원격 관리의 허용 여부 등이 표시됩니다.

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

Administration Server 기본 설정

Administration Server는 관리 인터페이스를 제공하며 관리 인터페이스 및 명령줄 인터페이스에 대한 관리 기능을 제공하는 Sun ONE Application Server의 특수한 인스턴스입니다. 이 서버는 Sun ONE Application Server에 대한 기능의 구성, 배포 및 모니터링을 관리합니다. 이 장에서는 Administration Server의 구성 방법에 대해 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- Administration Server 정보
- Administration Server 시작
- Administration Server 종료
- Administration Server 설정 액세스
- Administration Server 제어 설정 보기
- Administration Server에 변경 사항 적용
- Administration Server의 HTTP Listener 설정 편집
- SNMP, 로깅 및 보안 기본 설정 구성

Administration Server 정보

Administration Server는 관리 인터페이스에 대한 관리 기능과 명령줄 인터페이스를 제공하는 Sun ONE Application Server의 특수한 인스턴스입니다. 이러한 인터페이스에 대한 기능의 구성, 배포 및 모니터링을 관리합니다. 또한 관리 인터페이스 페이지를 제공합니다. 관리 인터페이스를 사용하여 명령줄 인터페이스에서 대부분의 명령을 실행하려면 이 서버를 실행해야 합니다.

Administration Server는 Sun ONE Application Server (비번들 버전)를 설치하거나 새 도메인을 작성할 때 설치됩니다. Administration Server가 관리하는 여러 개의 Sun ONE Application Server 인스턴스를 작성할 수는 있지만 도메인당 하나의 Administration Server만 둘 수 있습니다. Administration Server를 사용하면 각 응용 프로그램 서버 인스턴스에 대한 구성 설정, 배포된 응용 프로그램 및 기타 서버 기능에 액세스할 수 있습니다.

관리 도메인에 대한 자세한 내용은 3장, "관리 도메인 구성"을 참조하십시오.

비번들 Sun ONE Application Server를 사용하는 경우 Sun ONE Application Server를 선택할 때 Administration Server의 포트 번호를 선택하거나 기본 포트인 4848을 사용하도록 선택하게 됩니다.

Solaris 9 번들 Sun ONE Application Server를 사용하는 경우 도메인과 Administration Server를 수동으로 작성해야 합니다. Solaris 9 번들 버전의 구성에 대한 자세한 내용은 31페이지의 “Solaris 번들 버전 구성”을 참조하십시오.

관리 인터페이스에 액세스하려면 웹 브라우저에 다음을 입력합니다.

http://*hostname.domain*:*port*/

여기서 *domain*은 Sun ONE Application Server 관리 도메인이 아니라 자신의 도메인 이름입니다.

예:

http://austen.sun.com:4848/

Sun ONE Application Server를 설치한 시스템에서 Administration Server에 액세스하는 경우에는 다음을 사용할 수 있습니다.

http://localhost:4848

사용자 이름과 암호를 입력하라는 메시지가 나타납니다.

Administration Server 시작

Administration Server를 시작하거나 다시 시작하려면 다음 항목에서 설명하는 방법 중 하나를 사용하십시오.

- startserv 스크립트 사용
- 명령줄 인터페이스 사용
- 서비스 창 사용(Windows의 경우)
- 시작 메뉴 사용(Windows의 경우)

startserv 스크립트 사용

start 스크립트를 사용하여 Administration Server를 시작하려면, 서버가 1024(UNIX의 경우)보다 작은 포트 번호에서 실행되는 경우 root로 로그인하고 그렇지 않은 경우 root 또는 서버의 사용자 계정으로 로그인합니다. 명령줄 프롬프트에서 다음 디렉토리로 이동합니다.

install_dir/domains/*domain_dir*/admin-server/bin

여기서 *install_dir*은 서버를 설치한 디렉토리이며 *domain_dir*은 관리 도메인 디렉토리입니다.

UNIX의 경우 다음을 입력합니다.

```
./startserv
```

줄의 끝에 선택적 매개 변수인 `-i`를 사용할 수 있습니다. 서버는 일반적으로 백그라운드 프로세스로 실행됩니다. `-i` 옵션을 사용하면 서버가 백그라운드로 실행되지 않습니다. 이 옵션은 `/etc/inittab`를 사용하여 서버를 실행할 때 유용합니다.

Windows의 경우 `startserv.bat` 파일을 실행합니다.

주 서버가 이미 실행 중이면 `startserv` 명령은 실패합니다. 먼저 서버를 중지한 다음 `startserv` 명령을 사용하십시오. 또한 서버 시작이 실패하면 프로세스를 중지시킨 다음 다시 시작해야 합니다.

명령줄 인터페이스 사용

명령줄 인터페이스의 `asadmin` 유틸리티에는 응용 프로그램 서버 및 연관된 모든 Sun ONE Application Server 인스턴스를 시작할 수 있는 `start-domain` 명령이 있습니다. 이 명령은 로컬에서만, 즉 Sun ONE Application Server가 설치된 시스템에서만 실행할 수 있습니다. 이 명령에는 인수가 필요 없습니다.

또한 `start-domain` 명령을 사용하여 관리 도메인 내의 모든 인스턴스를 시작할 수도 있습니다. 구문은 다음과 같습니다.

`asadmin start-domain [--domain` *domain-name*`]`

명령줄 인터페이스 사용에 대한 자세한 내용은 명령줄 인터페이스에 대한 온라인 도움말 및 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

서비스 창 사용(Windows의 경우)

Windows에서 "서비스 제어판"을 사용하여 서버를 시작하려면 다음 단계를 수행합니다.

1. "제어판"에서 "관리 도구"를 누릅니다.
2. "서비스"를 누릅니다.
3. 서비스의 목록을 스크롤하여 Application Server 7.0 Administration Server 서비스를 두 번 누릅니다.

   해당 도메인에 대해 "Administration Server"를 선택합니다. "서비스" 창의 "이름" 열에는 Administration Server의 이름이 Sun App Server Admin Server (*your_domain_name*:admin-server)로 표시됩니다.
4. "시작"을 누릅니다.
5. "확인"을 누릅니다.

그러면 컴퓨터를 시작할 때 Application Server 7.0 Administration Server 서비스가 자동으로 시작됩니다.

시작 메뉴 사용(Windows의 경우)

Windows "시작" 메뉴를 사용하여 서버를 시작하려면 다음 단계를 수행합니다.

1. "시작" 메뉴에서 "프로그램"을 선택합니다.
2. "Sun Microsystems"를 선택합니다.
3. "Sun ONE Application Server 7"을 선택합니다.
4. "응용 프로그램 서버 시작"을 누릅니다.

Administration Server 종료

Administration Server가 시작되면 계속 실행되어 요청을 수신하고 수용합니다. 예를 들어, Administration Server 로깅 기본 설정이나 Administration Server의 HTTP Listener가 수신하는 포트를 변경하는 경우 서버를 중지하였다 다시 시작해야 합니다.

Administration Server를 중지하면 새 연결 수용이 중지됩니다. 그런 다음 아직 해결되지 않은 모든 연결이 완료될 때까지 기다립니다. Administration Server가 중지되어 있으면 관리 인터페이스 또는 명령줄 인터페이스를 사용할 수 없습니다.

다음 항목에서 설명하는 방법 중 하나를 사용하여 서버를 중지할 수 있습니다.

- 관리 인터페이스를 사용하여 종료
- stopserv 스크립트를 사용하여 종료
- 명령줄 인터페이스를 사용하여 종료
- 서비스 창을 사용하여 종료(Windows의 경우)

서버를 종료하는 경우 서버가 종료 과정을 완료하는 데 몇 초가 걸릴 수 있습니다.

관리 인터페이스를 사용하여 종료

관리 인터페이스를 사용하여 Administration Server를 종료하려면 다음 단계를 수행합니다.

1. 왼쪽 창에서 "Admin Server"를 누릅니다.
2. "제어" 탭을 누릅니다.
3. "중지"를 누릅니다.

   이 링크를 누르면 Administration Server가 즉시 종료됩니다. 추가 화면은 표시되지 않습니다.

stopserv 스크립트를 사용하여 종료

Administration Server를 수동으로 중지시키려면 명령 프롬프트에서 다음 디렉토리로 이동합니다.

install_dir/domains/*domain_dir*/admin-server/bin

여기서 *install_dir*은 서버를 설치한 디렉토리이며 *domain_dir*은 도메인 디렉토리입니다.

UNIX의 경우 다음을 입력합니다.

```
./stopserv
```

이 명령은 서버를 실행하고 있는 사용자로 실행해야 합니다.

`*/etc*/inittab` 파일을 사용하여 서버를 다시 시작한 경우 `*/etc*/inittab`에서 server로 시작하는 줄을 제거하고 `kill -1`을 입력한 후 서버를 중지시켜야 합니다. 그렇지 않으면 서버가 중지된 후 자동으로 다시 시작됩니다.

Windows의 경우 `stopserv.bat` 파일을 실행합니다.

명령줄 인터페이스를 사용하여 종료

명령줄 인터페이스 `asadmin` 유틸리티의 `shutdown` 명령을 사용하여 Administration Server를 중지시킬 수 있습니다. 이 명령에는 인수가 필요 없으며 로컬 또는 원격에서 실행할 수 있습니다.

명령줄 인터페이스 `asadmin` 유틸리티의 `stop-appserv` 명령을 사용하여 Administration Server 및 연관된 모든 Sun ONE Application Server 인스턴스를 중지시킬 수 있습니다. 이 명령은 로컬에서만, 즉 Sun ONE Application Server가 설치된 시스템에서만 실행할 수 있습니다. 이 명령에는 인수가 필요 없습니다.

또한 `stop-domain` 명령을 사용하여 도메인을 종료시켜 Administration Server를 종료시킬 수도 있습니다. 이 명령은 기본적으로 Administration Server를 포함하여 도메인의 모든 인스턴스를 종료합니다. 또한 Administration Server를 *제외하고* 도메인의 모든 인스턴스를 종료하도록 구성할 수도 있습니다. 이 명령의 구문은 다음과 같습니다.

`asadmin stop-domain [--user` *admin_user*`] [--password` *admin_password*`] [--host` *admin_host*`] [--port` *admin_port*`] [--local=`*true/false*`] [--domain` *domain_name*`] [--adminserv=`*true/false*`] [--passwordfile` *file_name*`] [--secure | -s]`

`local` 옵션을 사용하면 명령은 로컬에서 실행됩니다. `--adminserv=false` 옵션을 사용하면 명령은 Administration Server를 중지시키지 않습니다. 하지만 `--adminserv`는 기본적으로 `true`로 설정되므로 Administration Server는 기본적으로 중지됩니다.

명령줄 인터페이스 사용에 대한 자세한 내용은 명령줄 인터페이스에 대한 온라인 도움말 및 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

서비스 창을 사용하여 종료(Windows의 경우)

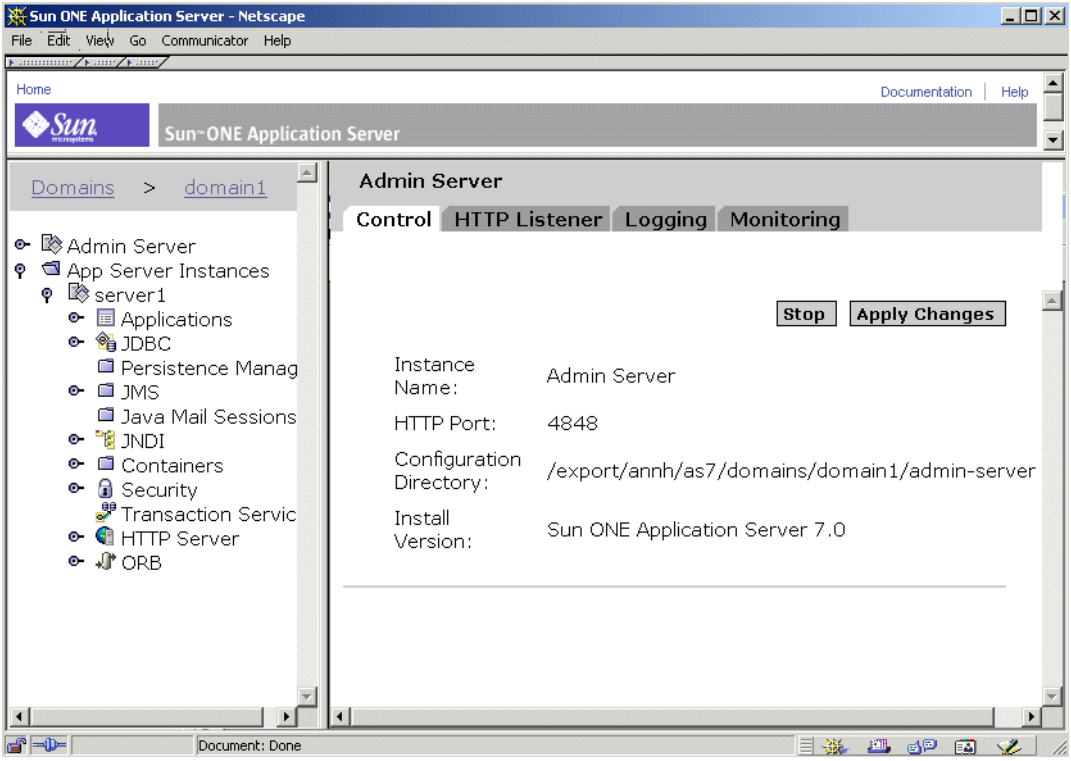
"서비스" 창을 사용하여 Administration Server를 종료하려면 다음 단계를 수행합니다.

1. "제어판"에서 "관리 도구"를 누릅니다.
2. "서비스"를 누릅니다.
3. 서비스 목록을 스크롤하여 Sun Application Server 7 Admin Server 서비스를 두 번 누릅니다.
4. "중지"를 누릅니다.
5. "확인"을 누릅니다.

Administration Server 설정 액세스

Administration Server 설정에 액세스하려면 관리 인터페이스의 왼쪽 창에서 "Admin Server"를 누릅니다. Administration Server의 설정이 몇 개의 탭으로 분류되어 오른쪽 창에 나타납니다.

Administration Server 사용자 인터페이스



탭을 눌러서 특정 기능 영역에 대한 설정에 액세스합니다. 경우에 따라 탭을 누르면 페이지로 직접 이동하거나 선택할 수 있는 페이지 목록이 나타납니다.

이 장에서는 제어 및 HTTP Listener 탭에 대해 설명합니다. 모니터링 및 SNMP 설정에 대한 자세한 내용은 6장, "Sun ONE Application Server 모니터링"을 참조하십시오. 로깅에 대한 자세한 내용은 5장, "로깅 사용"을 참조하십시오.

Administration Server 제어 설정 보기

Administration Server 제어 설정에는 인스턴스 이름(Admin Server), Administration Server가 실행 중인 포트, 구성 파일이 포함된 디렉토리, 실행 중인 Sun ONE Application Server 소프트웨어의 버전이 표시됩니다.

이 설정을 보려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "Admin Server"를 누릅니다.
2. "제어" 탭을 누릅니다.

Administration Server에 변경 사항 적용

관리 인터페이스 또는 명령줄 인터페이스를 사용하여 Administration Server의 구성 정보를 변경한 경우 변경 사항을 적용해야 변경 사항이 실제 적용될 수 있습니다. 이것을 서버 재구성이라고도 합니다. 변경 사항을 적용하면 변경 사항을 마지막으로 적용한 이후의 모든 구성 변경 사항이 적용됩니다.

변경 사항을 적용해야 하는 Administration Server의 구성을 변경한 경우 왼쪽 창의 트리 보기에 있는 Admin Server 옆에 노란색 아이콘이 나타납니다.

경고 아이콘

구성 변경 사항을 적용하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "Admin Server"를 누릅니다.
2. "제어" 탭을 누릅니다.
3. "변경 사항 적용"을 누릅니다.

   변경 사항이 적용되면 화면에 메시지가 표시됩니다.

Administration Server의 HTTP Listener 설정 편집

서버가 요청을 처리할 수 있으려면 먼저 HTTP Listener를 통해 요청을 받아야 합니다.

Sun ONE Application Server의 번들 버전인 Administration Server의 HTTP Listener의 경우 설치할 때 `http-listener-1`이 자동으로 작성됩니다. 이 HTTP Listener는 IP 주소 0.0.0.0과 설치 도중 Administration Server 포트 번호로 지정한 포트 번호를 사용합니다. 기본값은 4848입니다. 기본 HTTP Listener는 삭제할 수 없습니다.

Administration Server 인스턴스(도메인에서)의 경우 HTTP Listener에만 `http-listener-1` ID가 있습니다. 즉, Administration Server 인스턴스를 만드는 경우 하나의 HTTP Listener만 한 번에 하나의 HTTP 또는 HTTPS 프로토콜에서 작동할 수 있습니다. 또한 HTTP 및 HTTPS 연결 모두가 동시에 Administration Server에 연결할 수 없다는 것을 의미하기도 합니다. Solaris 9 번들 버전의 구성에 대한 자세한 내용은 31페이지의 “Solaris 번들 버전 구성”을 참조하십시오.

HTTP Listener는 Administration Server에 대한 SSL/TLS 보안 설정을 활성화하고 구성하는 위치입니다.

HTTP Listener의 억셉터 스레드(억셉트 스레드라고도 함)의 수를 지정합니다. 억셉트 스레드는 연결 대기 스레드입니다. 스레드는 연결을 승인하고 대기열에 위치시킵니다. 그런 다음 작업자 스레드가 여기에서 이 연결을 선택합니다. 새 요청이 들어올 때마다 항상 사용할 수 있도록 충분한 억셉트 스레드가 있는 것이 이상적이지만 너무 많으면 시스템에 부하를 줄 수 있습니다. 기본값은 1입니다. 일반적으로 시스템의 CPU 당 하나의 억셉트 스레드가 있는 것이 좋습니다. 성능 문제가 있는 경우 이 값을 조정할 수 있습니다. 성능에 대한 자세한 내용은 *Sun ONE Application Server Performance Tuning and Sizing Guide*를 참조하십시오.

Administration Server의 HTTP Listener 설정을 편집하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "Admin Server"를 누릅니다.
2. "HTTP Listeners" 탭을 누릅니다.
3. 원하는 대로 변경한 뒤 "확인"을 누릅니다.

HTTP Listener에 대한 자세한 내용은 온라인 도움말을 참조하십시오.

SNMP, 로깅 및 보안 기본 설정 구성

SNMP 설정에 대한 자세한 내용은 6장, "Sun ONE Application Server 모니터링"을 참조하십시오. 로깅에 대한 자세한 내용은 5장, "로깅 사용"을 참조하십시오. 보안 설정에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Guide to Security*를 참조하십시오.

관리 도메인 구성

이 장에서는 Sun ONE Application Server로 관리 도메인을 설정하고 관리하는 방법에 대해 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- 관리 도메인 정보
- 도메인 구성
- 도메인 레지스트리 재작성

관리 도메인 정보

관리 도메인은 여러 관리자가 시스템의 특정 응용 프로그램 서버 그룹(도메인)을 관리할 수 있는 기본 보안 구조를 제공합니다. 응용 프로그램 서버 인스턴스를 이렇게 분리하면 같은 시스템을 여러 조직이 공유하고 각 조직마다 관리자를 둘 수 있습니다.

Sun One Application Server 내의 모든 응용 프로그램 서버 인스턴스는 한 도메인의 구성원입니다. 반드시 두 개 이상의 도메인이 있어야 하는 것은 아니지만 필요한 경우 여러 도메인이 지원되는 것은 유용한 기능입니다.

기본 운영 체제의 보안 메커니즘을 사용하여(즉, 파일 사용 권한을 통해) 로컬 명령에 대해 관리 보안이 설정됩니다. 사용자 이름/암호 쌍을 사용하여 특정 Admin Server와 통신함으로써 원격 명령 보안이 설정됩니다. 관리 도메인은 다른 보안 구조를 사용하지 않습니다.

이 절에서는 다음 내용을 설명합니다.

- 관리 도메인 구현
- 디렉토리 구조
- 프로세스/포트 구조

관리 도메인 구현

도메인은 파일, 운영 체제 프로세스 및 포트를 사용하여 구현됩니다. 각 도메인에는 고유한 이름이 있습니다.

디렉토리 구조

설치 내에서 모든 도메인이 공유하는 파일(구성, 실행 파일 등)이 있습니다. 여기에서 중요한 것은 도메인에 고유한 파일들입니다.

도메인에 고유한 파일은 모두 도메인 디렉토리라고 하는 공용 루트 디렉토리를 공유하며 이 디렉토리의 이름은 도메인의 이름과 같습니다. 도메인 디렉토리 아래에는 인스턴스 별로 인스턴스의 이름과 같은 디렉토리가 하나씩 있으며, 이 인스턴스 디렉토리 아래에는 인스턴스 고유 파일이 있습니다.

도메인 디렉토리는 보안 사용 권한 및 기타 운영 체제 수준 제한에 따라 파일 시스템의 어떠한 곳에도 구성될 수 있습니다. 사용자가 다른 방식을 선택한 경우가 아니면 도메인 디렉토리는 기본 디렉토리 아래에 생성됩니다(도메인 디렉토리). 하지만 사용자가 도메인 디렉토리를 다른 위치에 만들 수 있습니다.

프로세스/포트 구조

도메인은 실행 도중 운영 체제 프로세스와 포트를 사용합니다. 특히 도메인 내에서 실행되는 각 인스턴스(도메인의 Admin Server 포함)마다 프로세스와 포트가 있습니다.

도메인 구성

도메인은 각 용도에 맞는 명령을 사용하여 작성, 삭제, 나열, 시작 및 중지할 수 있습니다.

도메인의 작성, 삭제 및 시작은 로컬에서만 수행할 수 있는 반면 나열 및 중지는 로컬과 원격 모두에서 수행할 수 있습니다.

삭제, 시작 및 중지 명령에는 모두 도메인 이름을 지정해야 합니다. 도메인이 하나뿐인 경우 이 이름은 선택 사항입니다. 여러 개의 도메인이 구성된 상태에서 도메인 이름을 지정하지 않으면 오류가 발생합니다.

이 절에서는 다음 내용을 설명합니다.

- 도메인 작성
- 도메인 삭제
- 도메인 나열
- 도메인 시작
- 도메인 중지

도메인 작성

도메인은 `create-domain` 명령을 사용하여 작성합니다. 이 명령은 로컬에서만 실행할 수 있습니다.

개요:

`asadmin create-domain [--path` *domain_path*`] [--sysuser` *sys_user*`] [--passwordfile` *file_name*`] --adminport` *port_number* `--adminuser` *admin_user* `--adminpassword` *password domain_name*

예: 기본 위치에 도메인 작성

```
$ asadmin create-domain --adminport 123 --adminuser MyAdmin
--adminpassword MyPassword MyDomain
```

이 예는 기본 위치(즉, 도메인 디렉토리)에 MyDomain이라는 도메인을 만듭니다. Administration Server는 포트 123에서 수신하고 관리 사용자 이름은 MyAdmin, 암호는 MyPassword를 사용합니다. 도메인 디렉토리 및 이 디렉토리 아래의 파일은 이 명령을 실행한 운영 체제 사용자가 소유합니다. 또한 운영 체제 프로세스는 이 명령을 실행한 사용자로 실행됩니다.

MyDomain이라는 도메인이 이미 있는 경우 오류 메시지가 반환됩니다.

명령줄에 암호를 사용하면 보안 문제가 발생할 수 있으므로 암호를 파일에 저장한 다음 `--passwordfile` 옵션을 사용하여 이 파일을 통해 암호를 전달할 수 있습니다.

예: 기본 위치가 아닌 곳에 도메인 작성

```
$ asadmin create-domain --path $HOME --adminport 123 --adminuser
MyAdmin --adminpassword MyPassword MyDomain
```

이 예는 첫 번째 예와 비슷하며 도메인 디렉토리가 기본 도메인 디렉토리 아래가 아니라 사용자의 `$HOME` 디렉토리 아래에 위치한다는 점만 다릅니다.

예: 다른 사용자에 대한 도메인 작성(UNIX 전용)

```
# asadmin  create-domain --user AnotherUser --adminport 123
--adminuser MyAdmin --adminpassword MyPassword MyDomain
```

이 예는 첫 번째 예와 비슷하며 도메인과 그 파일을 AnotherUser라는 사용자가 소유한다는 점만 다릅니다. 이 작업은 운영 체제가 처리합니다.

`--sysuser` 옵션을 사용하면 다른 사용자가 관리자인 도메인을 생성할 수 있습니다. 이 옵션을 사용하려면 create-domain 명령을 실행하는 사용자가 root여야 합니다.

UNIX 플랫폼의 사용자 권한

root가 아닌 사용자가 관리 도메인을 작성 및 삭제하려면 사용자 ID를 도메인 구성 파일에 대한 쓰기 권한이 있는 UNIX 그룹에 추가해야 합니다.

1. 설치 전체에 대한 도메인 구성 파일에 적용되는 그룹이 될 UNIX 그룹을 만듭니다. 예를 들어, asadmin이라는 이름의 UNIX 그룹을 만듭니다.
2. /etc/appserver 아래에 있는 설치 전체 도메인 구성 파일을 새로 만든 UNIX 그룹이 소유하도록 설정합니다.

   파일의 이름은 `domains.bin`과 `domains.lck`입니다. 예를 들어, 다음 파일에 할당된 그룹을 변경합니다.

   ```
   -rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.bin
   ```

   ```
   -rw-r--r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
   ```

3. 그런 다음 새로 만든 UNIX 그룹의 이 파일에 대한 쓰기 액세스 권한을 활성화합니다. 이 예에서는, 다음과 같은 사용 권한이 설정됩니다.

   ```
   -rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.bin
   ```

   ```
   -rw-rw-r-- 1 root asadmin 0 Sep 18 14:34 domains.lck
   ```

4. UNIX 그룹에 사용자 ID를 추가합니다.

또는 root 사용자에게 설치 전체 구성 파일에 쓰기 액세스 권한을 부여하지 않으려면 사용자 대신 관리 도메인을 만들 수 있습니다. 새 관리 도메인을 만드는 도중 `--sysuser` 및 `--path` 옵션을 지정하여 도메인의 디렉토리와 파일을 소유할 UNIX 사용자 ID와 관리 도메인이 작성될 위치를 지정합니다. 예를 들어, 60페이지의 "예: 다른 사용자에 대한 도메인 작성(UNIX 전용)"을 참조하십시오.

사용자 ID 아래에 관리 도메인을 만들면 사용자는 새 응용 프로그램 서버 인스턴스를 만들고 응용 프로그램 서버 인스턴스에 대해 다양한 관리 작업을 수행할 수 있게 됩니다. 사용자 ID는 관리 도메인 구성 파일에 대해 쓰기 권한을 갖는 UNIX 그룹에 속할 필요가 없습니다. 관리 도메인을 작성 및 삭제할 경우에만 UNIX 그룹의 멤버십이 필요합니다.

도메인 삭제

도메인은 `delete-domain` 명령을 사용하여 삭제합니다. 도메인을 관리할 수 있는 운영 체제 사용자(또는 root)만 이 명령을 제대로 실행할 수 있습니다. 이 명령은 로컬에서만 실행할 수 있습니다.

개요:

```
asadmin delete-domain [domain_name]
```

예: 도메인 삭제

```
$ asadmin delete-domain MyDomain
```

이 예는 로컬 시스템에서 MyDomain이라는 도메인을 삭제합니다.

도메인 나열

시스템에 작성된 도메인은 `list-domains` 명령을 사용하여 찾아볼 수 있습니다.

이 명령은 로컬과 원격 모두에서 사용할 수 있습니다.

개요:

```
asadmin list-domains [--host host] [--port port] [--password password]
[--user user]
```

예: 로컬 시스템의 도메인 나열

```
$ asadmin list-domains

domain1    [/opt/ias/build/domains/domain1]
```

예: remote 옵션을 사용하여 로컬 시스템의 도메인 나열

```
$ asadmin list-domains --user admin --password password --host
localhost --port 4848

domain1    [/opt/ias/build/domains/domain1]
```

도메인 시작

도메인은 `start-domain` 명령을 사용하여 시작할 수 있습니다. 그러면 도메인의 Admin Server와 도메인에 있는 기타 모든 인스턴스가 시작됩니다.

이 명령은 로컬에서만 실행할 수 있습니다.

개요:

```
asadmin start-domain [--domain domain_name]
```

예: 시스템에 있는 도메인만 시작

```
$ start-domain

Instance domain1:admin-server started

Instance domain1:server1 started

Domain domain1 Started.
```

도메인 중지

도메인은 `stop-domain` 명령을 사용하여 중지시킬 수 있습니다. 사용자는 도메인 내의 모든 인스턴스 또는 Admin Server를 제외한 모든 인스턴스를 중지하여 도메인 원격 관리가 가능하도록 선택할 수 있습니다.

이 명령은 로컬과 원격 모두에서 사용할 수 있습니다.

개요:

```
asadmin stop-domain [--user admin_user] [--password admin_password]
[--host host_name] [--port port_name] [-- local=false] [--domain
domain_name] [--adminserv=true] [--passwordfile file_name] [--secure |
-s]
```

예: Admin Server 인스턴스를 제외한 도메인의 모든 인스턴스 중지

```
$ asadmin asadmin stop-domain --user admin --password password
--host localhost --port 4848 --adminserv=false --domain domain1

DomainStoppedRemotely
```

도메인 레지스트리 재작성

구현 목적상 각 도메인의 세부 사항(이름, 위치, 사용 포트 등)은 도메인 레지스트리 파일에 기록됩니다.

일반적인 작업 조건에서는 도메인 레지스트리를 수정하지 않아야 합니다. 모든 도메인 레지스트리 수정 또는 사용은 시스템 관리에 사용되는 명령으로 캡슐화되기 때문입니다. 하지만 도메인 레지스트리는 파일이기 때문에 손상될 수 있으며(예를 들어, 스크립트가 잘못되었거나 누군가 레지스트리를 실수로 삭제한 경우 등) 이러한 경우 파일을 다시 작성해야 할 수 있습니다.

주 `asadmin` 명령을 사용하여 명령줄 인터페이스를 통해 도메인 레지스트리에 액세스할 수 있습니다.

레지스트리가 손상된 경우 다음 절차를 수행하여 레지스트리를 다시 작성하십시오.

1. 모든 도메인의 목록과 해당 도메인이 위치한 디렉토리(기본값 또는 비기본값)를 얻습니다.
2. 각 디렉토리 이름을 변경합니다. 예를 들어, 각 디렉토리 이름에 접미어 ".bak"를 추가합니다.
3. 포트, 암호 등에 대한 기본값을 사용하여 각 도메인을 다시 원래 위치에 만듭니다.

4. 새 도메인 디렉토리를 삭제하고 원래 디렉토리로 교체합니다.
5. 각 도메인에 대해 `reconfig` 명령을 실행합니다. 그러면 도메인 레지스트리가 기존 도메인의 값으로 업데이트됩니다.

개별 서버 인스턴스 관리

4장, "응용 프로그램 서버 인스턴스 사용"

5장, "로깅 사용"

6장, "Sun ONE Application Server 모니터링"

7장, "웹 서버 플러그 인 구성"

8장, "J2EE 컨테이너 구성"

9장, "트랜잭션 서비스 사용"

10장, "이름 지정 및 자원 구성"

11장, "JMS 서비스 사용"

12장, "Corba/IIOP 클라이언트를 지원하도록 서버 구성"

13장, "응용 프로그램 배포"

응용 프로그램 서버 인스턴스 사용

이 장에서는 Sun ONE Application Server 인스턴스를 만들고 삭제하고 구성하며 시작 및 중지하는 방법에 대해 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- 응용 프로그램 서버 인스턴스 정보
- 응용 프로그램 서버 인스턴스 시작 및 중지
- 디버그 모드에서 응용 프로그램 서버 인스턴스 시작
- 종료 시간 제한 설정
- 응용 프로그램 서버 인스턴스를 자동으로 다시 시작(UNIX의 경우)
- 응용 프로그램 서버 인스턴스를 수동으로 다시 시작(UNIX의 경우)
- 워치독 정보
- 응용 프로그램 서버 인스턴스 추가
- 응용 프로그램 서버 인스턴스 삭제
- 응용 프로그램 서버 인스턴스에 변경 사항 적용
- 응용 프로그램 서버 인스턴스 상태 보기
- JVM 설정 구성
- 로깅 설정 및 모니터링 설정 구성
- 응용 프로그램 서버 인스턴스 고급 설정 변경

응용 프로그램 서버 인스턴스 정보

Sun ONE Application Server는 소프트웨어 비번들 버전을 설치할 때 `server1`이라는 응용 프로그램 서버 인스턴스를 하나 만듭니다. 필요한 경우 `server1` 인스턴스를 삭제하고 다른 이름으로 새 인스턴스를 만들 수 있습니다.

Solaris 9 번들 버전 소프트웨어를 사용하는 경우 고유 서버 인스턴스를 만들어야 합니다. 자세한 내용은 31페이지의 “Solaris 번들 버전 구성”을 참조하십시오.

각 응용 프로그램 서버 인스턴스는 고유한 J2EE 구성, J2EE 자원, 응용 프로그램 배포 영역 및 서버 구성 설정을 갖습니다. 한 응용 프로그램 서버 인스턴스를 변경하는 경우 다른 응용 프로그램 서버 인스턴스는 영향을 받지 않습니다. 한 관리 도메인 내에 많은 응용 프로그램 서버 인스턴스를 가질 수 있습니다. 한 도메인 내의 모든 서버 인스턴스는 같은 Administration Server는 갖습니다. 도메인에 대한 자세한 내용은 3장, "관리 도메인 구성"을 참조하십시오.

대부분의 경우 응용 프로그램 서버 인스턴스는 하나만 있으면 됩니다. 하지만 환경에 따라 하나 이상의 추가 응용 프로그램 서버 인스턴스를 만들어야 할 수도 있습니다. 예를 들어, 개발 환경에서 서로 다른 응용 프로그램 서버 인스턴스를 사용하여 여러 Sun ONE Application Server 구성을 테스트하거나 여러 응용 프로그램 배포를 비교하고 테스트할 수 있습니다. 응용 프로그램 서버 인스턴스는 쉽게 추가하거나 삭제할 수 있으므로 이를 사용하면 개발 중에 테스트를 할 수 있는 임시 "샌드 박스" 영역을 만들 수 있습니다.

또한 각 응용 프로그램 서버 인스턴스에 대해 가상 서버를 만들 수도 있습니다. 단일 설치 응용 프로그램 서버 인스턴스 내에서 회사나 개인 도메인 이름, IP 주소 및 몇 가지 관리 기능을 제공할 수 있습니다. 사용자의 경우 하드웨어와 기본 서버 유지 관리를 제외하면 고유 웹 서버를 가진 것과 거의 같습니다. 이러한 가상 서버는 여러 응용 프로그램 서버 인스턴스에 걸쳐 있을 수 없습니다. 가상 서버에 대한 자세한 내용은 15장, "가상 서버 사용"을 참조하십시오.

운영상 배포할 때 많은 용도로 여러 응용 프로그램 서버 인스턴스 대신 가상 서버를 사용할 수 있습니다. 하지만 가상 서버로는 미흡할 경우 여러 응용 프로그램 서버 인스턴스를 사용할 수도 있습니다.

응용 프로그램 서버 인스턴스 시작 및 중지

Sun ONE Application Server 인스턴스는 자동으로 시작되지 않습니다. 일단 인스턴스를 시작하면 중지할 때까지 인스턴스가 계속 실행됩니다. 응용 프로그램 서버 인스턴스를 중지하면 응용 프로그램 서버 인스턴스는 더 이상 새 연결을 수락하지 않으며 해결되지 않은 모든 연결이 완료될 때까지 대기합니다. 시스템이 충돌하거나 오프라인이 되면 서버가 종료되므로 서버에서 처리 중이던 요청이 손실될 수 있습니다.

다음에서 다루고 있는 여러 방법 중 하나를 사용하여 응용 프로그램 서버 인스턴스를 시작 및 중지할 수 있습니다.

- 관리 인터페이스에서 시작 및 중지 버튼 사용
- start-instance 및 stop-instance 명령 사용
- Windows 서비스 사용(Windows의 경우)
- startserv 및 stopserv 스크립트 사용

주	서버에 보안 모듈이 설치되어 있는 경우 서버를 시작하거나 중지하려면 먼저 해당 암호를 입력해야 합니다.

서버 인증서를 설치한 경우 Sun ONE Application Server가 시작하기 전에 관리자에게 키 데이터베이스 암호를 입력하라는 메시지를 표시합니다. Sun ONE Application Server를 무인으로 다시 시작하려면 암호를 `password.conf` 파일에 저장해야 합니다. 이 파일과 키 데이터베이스가 손상되지 않도록 시스템이 적절하게 보호되어 있는 경우에만 이 방법을 사용하십시오. `password.conf` 만들기 및 사용에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

주	UNIX의 경우 일부 Sun ONE Application Server 설치 시 해당 운영 체제에서 기본적으로 허용하는 것보다 더 많은 메모리 및/또는 파일 설명자에 액세스해야 할 경우가 있습니다. 서버를 시작할 수 없는 경우 `ulimit` 명령을 사용하여 운영 체제에서 부과한 자원 제한을 검사하십시오. 자세한 내용은 운영 체제의 `ulimit` 설명서 페이지를 참조하십시오.

관리 인터페이스에서 시작 및 중지 버튼 사용

관리 인터페이스를 사용하여 서버를 시작 및 중지하려면 다음 작업을 수행합니다.

1. 왼쪽 창의 응용 프로그램 서버 인스턴스에서 시작하거나 중지할 인스턴스의 이름을 누릅니다.
2. 오른쪽 창에서 "시작" 또는 "중지"를 누르거나 "일반" 탭에서 "시작" 또는 "중지"를 누릅니다.
3. 응용 프로그램 서버 인스턴스가 성공적으로 시작하거나 중지되면 메시지가 표시됩니다.

start-instance 및 stop-instance 명령 사용

명령줄 인터페이스 유틸리티 `asadmin`을 사용하면 명령 프롬프트나 스크립트에서 응용 프로그램 서버 인스턴스를 시작하고 중지할 수 있습니다. `start-instance`와 `stop-instance` 명령을 사용하십시오.

이러한 명령 구문은 다음과 같습니다.

`start-instance` [`--user` *admin_user*] [`--password` *admin_password*] [`--host` *admin_host*] [`--port` *admin_port*] [`--local=`*true/false*] [`--domain` *domain_name*] [`--debug=`*true/false*] [`--passwordfile` *file_name*] [`--secure | -s`] *instance_name*

`stop-instance` [`--user` *admin_user*] [`--password` *admin_password*] [`--host` *admin_host*] [`--port` *admin_port*] [`--local=`*true/false*] [`--domain` *domain_name*] [`--passwordfile` *file_name*] [`--secure | -s`] *instance_name*

이러한 명령에는 Administration Server를 통하지 않고 서버를 시작하거나 중지할 수 있는 `local` 옵션이 있습니다. local 옵션을 사용하면 `host`, `port`, `user` 및 `password`(또는 `passwordfile`) 옵션을 지정할 필요가 없습니다.

이러한 명령 구문에 대한 자세한 내용은 `asadmin` 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

Windows 서비스 사용(Windows의 경우)

Windows의 "서비스 제어판"을 사용하여 서버를 시작할 수 있습니다.

다음 단계를 수행합니다.

1. "제어판"에서 "관리 도구"를 누릅니다.
2. "서비스"를 누릅니다.
3. 서비스 목록을 스크롤하여 서버에 맞는 서비스를 두 번 누릅니다.

   Sun Application Server (*domain_name:instance_name*)가 이에 해당합니다. 예를 들어, Sun Application Server (domain1:server1)가 됩니다.
4. "시작"이나 "중지"를 누릅니다.
5. "확인"을 누릅니다.

startserv 및 stopserv 스크립트 사용

`startserv` 및 `stopserv` 스크립트를 사용하려면 명령줄 프롬프트에서 다음 디렉토리로 이동합니다.

instance_dir/bin

여기서 *install_dir*은 서버를 설치한 디렉토리이고 *domain_dir*은 도메인 디렉토리이며 *instance_dir*은 시작할 인스턴스 이름입니다.

UNIX의 경우 다음을 입력합니다.

```
./startserv
```

1024 미만 숫자의 포트에서 서버가 실행 중인 경우 root로 로그인하고 그렇지 않은 경우 root나 서버의 사용자 계정으로 로그인합니다.

줄의 끝에 선택 매개 변수 `-i`를 사용할 수 있습니다. `-i` 옵션은 서버를 `inittab` 모드에서 실행하므로 서버 프로세스가 중지되거나 충돌하는 경우 `inittab`이 서버를 다시 시작합니다. 또한 이 옵션은 서버가 백그라운드 프로세스에서 실행되는 것을 방지합니다.

주	서버가 이미 실행 중이면 `startserv` 명령은 실패합니다. 먼저 서버를 중지한 다음 `startserv` 명령을 사용해야 합니다. 또한 서버 시작에 실패한 경우 프로세스를 다시 시작하기 전에 해당 프로세스를 중지시켜야 합니다.

Windows의 경우 다음을 입력합니다.

```
startserv
```

서버를 수동으로 중지시키려면 명령줄 프롬프트에서 다음 디렉토리로 이동합니다.

instance_dir/bin

여기서 *install_dir*은 서버를 설치한 디렉토리이며 *instance_dir*은 시작할 인스턴스의 이름입니다.

UNIX의 경우 다음을 입력합니다.

```
./stopserv
```

`/etc/inittab` 파일을 사용하여 서버를 다시 시작한 경우 서버를 중지하기 전에 `/etc/inittab`에서 서버를 시작하는 명령줄을 제거하고 `kill -1 1`을 입력해야 합니다. 그렇지 않은 경우 서버는 중지한 후에 자동으로 다시 시작됩니다.

Windows의 경우 다음을 입력합니다.

```
stopserv
```

디버그 모드에서 응용 프로그램 서버 인스턴스 시작

개발자가 J2EE 응용 프로그램을 디버깅하려는 경우 디버그 모드에서 응용 프로그램 서버 인스턴스를 실행할 수 있습니다.

디버그 모드에서 서버를 시작하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 액세스하여 디버그 모드에서 시작할 응용 프로그램 서버 인스턴스의 이름을 누릅니다.
2. "일반" 탭을 누릅니다.
3. "디버그 모드로 실행" 옆에 있는 확인란을 누릅니다.
4. 응용 프로그램 서버 인스턴스를 다시 시작합니다.

디버그 모드는 JVM 설정을 변경합니다. "디버그 사용 가능"이 true로 설정되어 있으면 "디버그 옵션"이 변경됩니다. JVM 디버그 옵션에 대한 자세한 내용은 http://java.sun.com/products/jpda/doc/conninv.html에서 Java Platform Debugger Options 설명서를 참조하십시오.

명령줄 인터페이스에서 디버그 모드로 응용 프로그램 서버 인스턴스를 시작하려면 `debug` 옵션을 true로 설정하고 `asadmin` 유틸리티의 `start-instance` 명령을 사용합니다. 명령 구문에 대한 자세한 내용은 명령줄 인터페이스의 온라인 도움말을 참조하십시오.

종료 시간 제한 설정

응용 프로그램 서버 인스턴스를 중지하면 새 연결 승인도 중지됩니다. 그런 다음 해결되지 않은 모든 연결이 완료될 때까지 응용 프로 그램 서버 인스턴스가 대기합니다. 시간이 초과되기 전에 서버가 대기하는 시간을 *instance_dir*`/config/`에 있는 `init.conf` 파일에서 구성할 수 있습니다. 기본적으로 30초로 설정됩니다. 이 값을 변경하려면 다음 행을 `init.conf`에 추가합니다.

```
TerminateTimeout seconds
```

여기서 *seconds*는 시간이 초과되기 전에 서버가 대기할 시간(초)을 나타냅니다.

이 값을 구성하는 경우의 장점은 연결이 완료될 때까지 서버가 더 오래 기다린다는 것입니다. 하지만 서버는 응답 없는 클라이언트로부터의 연결을 열어 둔 상태로 있는 경우가 있으므로 종료 시간 제한을 늘리면 서버를 종료하는 데 걸리는 시간이 길어질 수 있습니다.

응용 프로그램 서버 인스턴스를 자동으로 다시 시작 (UNIX의 경우)

다음 명령 중 하나를 사용하여 응용 프로그램 서버 인스턴스를 다시 시작할 수 있습니다.

- `/etc/inittab` 파일에서 자동으로 다시 시작합니다.

  System V에서 파생되지 않은 UNIX 버전을 사용 중인 경우 `/etc/inittab` 파일을 사용할 수 없습니다.
- 시스템이 다시 부트할 때 `/etc/rc2.d`의 데몬을 사용하여 자동으로 다시 시작합니다.
- 수동으로 다시 시작합니다. 69페이지의 “응용 프로그램 서버 인스턴스 시작 및 중지”와 78페이지의 “응용 프로그램 서버 인스턴스 삭제”를 참조하십시오.

이 절에서는 다음 항목에 대해 설명합니다.

- 자동으로 다시 시작에 대한 정보
- /etc/inittab을 사용하여 자동으로 다시 시작(UNIX의 경우)
- 시스템 RC 스크립트를 사용하여 자동으로 다시 시작(UNIX의 경우)

자동으로 다시 시작에 대한 정보

설치 스크립트는 `/etc/rc.local` 또는 `/etc/inittab` 파일을 편집할 수 없으므로 텍스트 편집기를 사용하여 이러한 파일을 편집해야 합니다. 이러한 파일의 편집 방법을 모르는 경우 시스템 관리자에게 문의하거나 시스템 설명서를 참조하십시오.

일반적으로 서버는 시작하기 전에 암호를 요구하므로 이러한 파일 중 하나를 사용하여 SSL 사용 서버를 시작할 수 없습니다. 암호를 파일에 일반 텍스트로 저장하면 SSL 사용 서버를 자동으로 시작할 수 있지만 이 방법은 권장하지 *않습니다*.

주의 SSL 사용 서버의 암호를 서버의 startserv 스크립트에 일반 텍스트로 두면 보안상 아주 위험합니다. 파일에 액세스할 수 있는 사람은 SSL 사용 서버의 암호에 액세스할 수 있습니다. SSL 사용 서버의 암호를 일반 텍스트로 보관하기 전에 보안상 위험하다는 점을 고려하십시오.

서버의 `startserv` 스크립트, 키 쌍 파일 및 키 암호는 이에 대한 읽기 및 쓰기 액세스 권한을 갖는 소유자만 root로 소유해야 합니다. root가 아닌 사용자가 서버를 설치한 경우에는 해당 사용자 계정으로 이를 소유해야 합니다.

/etc/inittab을 사용하여 자동으로 다시 시작(UNIX의 경우)

`inittab`을 사용하여 서버를 다시 시작하려면
`/etc/inittab` 파일의 명령줄에 다음 텍스트를 입력합니다.

```
http:2:respawn:install_dir/path_to_domain_dir/instance_dir/bin/startserv -start
-i
```

여기서 *install_dir*은 서버를 설치한 디렉토리이고 *path_to_domain_dir*은 도메인 경로이며 *instance_dir*은 서버의 디렉토리입니다.

`-i` 옵션은 서버가 백그라운드 프로세스에서 실행되는 것을 방지합니다.

서버를 중지하기 전에 이 명령줄을 제거해야 합니다. 그렇지 않으면 서버가 자동으로 다시 시작됩니다.

시스템 RC 스크립트를 사용하여 자동으로 다시 시작(UNIX의 경우)

`/etc/rc.local`이나 이와 동등한 시스템 위치를 사용하는 경우 다음 행을 `/etc/rc.local`에 두십시오.

install_dir`/`*path_to_domain_dir*`/`*instance_dir*`/bin/startserv`

*install_dir*을 서버를 설치한 디렉토리로 바꾸고 *path_to_domain_dir*을 도메인에 대한 경로로 바꾸고 *instance_dir*을 응용 프로그램 서버 인스턴스의 이름으로 바꾸십시오.

응용 프로그램 서버 인스턴스를 수동으로 다시 시작(UNIX의 경우)

UNIX의 경우 서버 인스턴스를 수동으로 다시 시작할 수 있는 옵션이 있습니다. 서버 인스턴스를 중지한 다음 시작하는 것과는 달리 다시 시작하면 워치독 프로그램이 중지되지 않습니다. 워치독에 대한 자세한 내용은 77페이지의 “워치독 정보”를 참조하십시오.

주 구성 파일을 편집하여 수동으로 변경한 경우 관리 인터페이스의 "변경 사항 적용" 버튼을 사용하거나 `keepmanualchanges` 옵션을 `true`로 설정하고 `asadmin reconfig` 명령을 사용하여 서버를 다시 시작하기 전에 변경 사항을 적용해야 합니다. 변경 사항 적용에 대한 자세한 내용은 79페이지의 “응용 프로그램 서버 인스턴스에 변경 사항 적용”을 참조하십시오.

서버 인스턴스를 다시 시작할 수 있는 방법은 세 가지인데 다음에서 이에 대해 설명합니다.

- 다시 시작 버튼을 사용하여 서버 인스턴스 다시 시작(UNIX의 경우)
- restart-instance 명령을 사용하여 서버 인스턴스 다시 시작(UNIX의 경우)
- restartserv 스크립트를 사용하여 서버 인스턴스 다시 시작(UNIX의 경우)

다시 시작 버튼을 사용하여 서버 인스턴스 다시 시작(UNIX의 경우)

관리 인터페이스를 사용하여 서버 인스턴스를 다시 시작하려면 다음 작업을 수행합니다.

1. 왼쪽 창의 응용 프로그램 서버 인스턴스에서 다시 시작할 인스턴스 이름을 누릅니다.
2. 오른쪽 창에서 "다시 시작"을 누릅니다.
3. 응용 프로그램 서버 인스턴스가 성공적으로 다시 시작되면 메시지가 표시됩니다.

restart-instance 명령을 사용하여 서버 인스턴스 다시 시작(UNIX의 경우)

명령줄 인터페이스 유틸리티 `asadmin`을 사용하면 명령줄이나 스크립트에서 응용 프로그램 서버 인스턴스를 시작 및 중지할 수 있습니다. `restart-instance` 명령을 사용하십시오. 이 명령 구문은 다음과 같습니다.

`restart-instance [--user` *admin_user*`] [--password` *admin_password*`] [--host` *admin_host*`] [--port` *admin_port*`] [--local=`*true/false*`] [--domain` *domain_name*`] [--passwordfile` *file_name*`] [--secure | -s]` *instance_name*

이 명령에는 Administration Server를 통하지 않고 서버 인스턴스를 다시 시작할 때 사용할 수 있는 `local` 옵션이 있습니다.

이러한 명령 구문에 대한 자세한 내용은 `asadmin` 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

restartserv 스크립트를 사용하여 서버 인스턴스 다시 시작(UNIX의 경우)

`restartserv` 스크립트를 사용하려면 명령줄 프롬프트에서 다음 디렉토리로 이동합니다.

instance_dir/bin

여기서 *install_dir*은 서버를 설치한 디렉토리이고 *domain_dir*은 도메인 디렉토리이며 *instance_dir*은 시작할 인스턴스 이름입니다.

다음을 입력합니다.

```
./restartserv
```

1024 미만 숫자의 포트에서 서버가 실행 중인 경우 root로 로그인하고 그렇지 않은 경우 root나 서버의 사용자 계정으로 로그인합니다.

워치독 정보

워치독(UNIX의 경우 `appserv-wdog`, Windows의 경우 `appservd-wdog.exe`)은 Sun ONE Application Server와 함께 제공되는 프로그램으로 다음 작업을 수행합니다.

- 서버를 시작합니다.
- 서버를 중지시킵니다.
- SSL/TLS가 활성화되어 있는 경우 서버가 시작될 때 트러스트 데이터베이스 암호를 입력하라는 메시지를 관리자에게 표시합니다.
- 서버가 중지된 경우 서버를 다시 시작합니다.

워치독은 사용자 간섭없이 백그라운드에서 실행되므로 구성하거나 변경할 필요가 없습니다. Administration Server를 포함하여 각 응용 프로그램 서버 인스턴스에 대해 하나의 워치독이 실행됩니다.

UNIX의 각 워치독은 원시 응용 프로그램 서버(`appservd`) 프로세스에 대한 프로세스를 만들고 이 프로세서는 요청을 수락한 `appservd` 프로세스를 만듭니다. 서버를 시작하기 때문에 워치독 프로세스 ID는 *instance_dir*/logs의 `pid` 로그 파일에 표시됩니다.

주 UNIX 플랫폼의 `appservd` 프로세스: Windows의 경우 각 응용 프로그램 서버 인스턴스에 대해 단일 appservd 프로세스가 시작되지만 UNIX 시스템의 경우 응용 프로그램 서버 인스턴스마다 두 개의 appservd 프로세스가 시작합니다.

UNIX에서 첫 번째 appservd 프로세스를 "원시" 프로세스라 하며 두 번째 appservd 프로세스를 "작업자" 프로세스라 합니다. 원시 프로세스는 중요한 제어기 역할을 하는 반면 작업자 프로세스는 응용 프로그램 요청을 실제로 처리하는 프로세스입니다. 응용 프로그램 서버의 다음 릴리스에서는 각 응용 프로그램 서버 인스턴스에 대해 작업자 프로세스 수를 정의하는 옵션이 제공될 것입니다. 이 제품의 초기 릴리스에서는 응용 프로그램 서버 인스턴스당 하나의 작업자 프로세스만 지원됩니다.

응용 프로그램 서버 인스턴스 추가

관리 인터페이스를 사용하여 응용 프로그램 서버 인스턴스를 추가하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 액세스하여 왼쪽 창에서 "응용 프로그램 서버 인스턴스"를 누릅니다.
2. "일반" 탭을 누릅니다.
3. "응용 프로그램 서버 인스턴스" 페이지에서 "새로 만들기"를 누릅니다.
4. "새 인스턴스 작성" 페이지에서 인스턴스 이름과 포트 번호를 지정합니다.

   인스턴스 이름은 이 Administration Server와 도메인에 대해 고유해야 합니다. 시스템의 다른 프로세스에서 이 포트 번호를 사용하고 있지 않아야 합니다.

   UNIX를 사용 중인 경우 실행할 인스턴스에 대해 UNIX 사용자를 지정할 수도 있습니다.
5. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

명령줄 인터페이스를 사용하여 응용 프로그램 서버 인스턴스를 추가하려면 `asadmin` 유틸리티의 `create-instance` 명령을 사용합니다. 이 명령 구문은 다음과 같습니다.

`asadmin create-instance [--user` *admin_user*`] [--password` *admin_password*`] [--host` *host*`] [--port` *port*`] [--sysuser` *sys_user*`] [--domain` *domain_name*`] [--local=`*true/false*`] [--passwordfile` *file_name*`] [--secure | -s] --instanceport` *instance_port* *instance_name*

이 명령에는 Administration Server를 통하지 않고 서버 인스턴스를 다시 시작할 때 사용할 수 있는 `local` 옵션이 있습니다. `sysuser` 옵션은 UNIX 전용입니다.

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

응용 프로그램 서버 인스턴스 삭제

관리 도메인에서 응용 프로그램 서버 인스턴스를 삭제할 수 있습니다. 삭제 작업은 실행 취소할 수 없으므로 응용 프로그램 서버 인스턴스를 삭제하기 전에 더 이상 필요하지 않은지 확인해야 합니다.

관리 인터페이스를 사용하여 시스템에서 응용 프로그램 서버 인스턴스를 삭제하려면 다음 작업을 수행합니다.

1. 관리 인터페이스에 액세스하여 제거할 응용 프로그램 서버 인스턴스의 이름을 누릅니다.
2. "일반" 탭을 누릅니다.
3. "삭제"를 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

명령줄 인터페이스를 사용하여 시스템에서 응용 프로그램 서버 인스턴스를 삭제하려면 `asadmin` 유틸리티의 `delete-instance` 명령을 사용합니다. 이 명령 구문은 다음과 같습니다.

`asadmin delete-instance [--user` *admin_user*`] [--password` *admin_password*`] [--host` *admin_host*`] [--port` *admin_port*`] [--domain` *domain_name*`] [--local=`*true/false*`] [--passwordfile` *file_name*`] [--secure | -s]` *instance_name*

이 명령에는 Administration Server를 통하지 않고 서버 인스턴스를 삭제할 때 사용할 수 있는 `local` 옵션이 있습니다.

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

응용 프로그램 서버 인스턴스에 변경 사항 적용

관리 인터페이스 또는 명령줄 인터페이스를 사용하여 구성 정보를 변경하면 변경 사항은 바로 적용되지 않고 *server_instance*/config/backup에 있는 특수 파일에 저장됩니다. 관리 인터페이스와 명령줄 인터페이스는 위 디렉토리의 파일에 저장된 구성 값을 표시합니다. 변경 사항이 적용될 때까지는 실제로 바뀌지 않습니다. 변경 사항 적용을 서버 재구성이라고도 합니다. 변경 사항을 적용하면 마지막으로 변경 사항을 적용한 이후에 변경된 내용이 적용됩니다. 인스턴스를 다시 시작하더라도 변경 내용이 자동으로 적용되지는 않습니다.

서버 인스턴스 구성을 변경하여 해당 변경 사항을 적용해야 하는 경우 왼쪽 창의 트리 뷰에서, 서버 인스턴스에 액세스할 때 배너에서, 그리고 서버 인스턴스의 주 페이지에서 응용 프로그램 서버 인스턴스 옆에 노란색 아이콘이 표시됩니다.

경고 아이콘

관리 인터페이스를 사용하여 응용 프로그램 서버 인스턴스에 변경 사항을 적용하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 액세스하여 다시 구성할 응용 프로그램 서버 인스턴스 이름을 누릅니다.
2. "일반" 탭을 누릅니다.
3. "변경 사항 적용"을 누릅니다.

   변경 사항이 적용되면 화면에 메시지가 표시됩니다.

명령줄 인터페이스를 사용하여 응용 프로그램 서버 인스턴스를 사용하려면 `asadmin` 유틸리티의 `reconfig` 명령을 사용합니다. 이 명령 구문은 다음과 같습니다.

`asadmin reconfig --user` *admin_user* `[--password` *admin_password*`] [--host` *admin_host*`] [--port` *admin_port*`] [--passwordfile` *file_name*`] [--secure | -s] [--discardmanualchanges=`*true/false* `| --keepmanualchanges=`*true/false*`]` *instance_name*

구성 파일을 직접 편집하여 수동으로 변경하는 경우 재구성하는 동안 편집 내용을 보관하기 위해 `keepmanualchanges=true` (이 옵션의 기본값은 false)를 사용해야 합니다. `discardmanualchanges=true`로 설정하면 수동으로 변경한 내용은 삭제됩니다. `discardmanulchanges=false` (기본값)로 설정한 내용이 `keepmanualchanges=true`와 같지는 않습니다. 대신 false로 설정하면 `discardmanualchanges` 옵션을 지정하지 않은 것과 같습니다.

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

일부 등록 정보의 경우 변경 사항을 적용하려면 변경한 다음 서버를 다시 시작해야 합니다. 이러한 등록 정보는 `init.conf`와 `obj.conf` 구성 파일의 모든 등록 정보 집합과 `server.xml`의 일부 등록 정보를 포함합니다. 이러한 파일에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

변경 사항으로 인해 다시 시작해야 하는 경우 서버에서는 왼쪽 창의 트리 뷰에서, 서버 인스턴스에 액세스할 경우 배너에서, 그리고 서버 인스턴스의 주 페이지에서 응용 프로그램 서버 인스턴스 옆에 노란색 경고 아이콘을 표시하여 경고합니다. 배너와 페이지에는 다시 시작해야 함을 알리는 메시지가 표시됩니다. 서버 인스턴스를 다시 시작하면 노란색 경고 아이콘이 사라집니다.

다시 시작할 필요가 *없는* `server.xml` 설정에는 다음 내용이 포함됩니다.

- J2EE 응용 프로그램(EAR 파일), EJB 모듈(JAR 파일), 웹 모듈(WAR 파일), 커넥터(RAR 파일) 배포, 배포 해제 및 재배포. 이러한 설정도 "변경 사항 적용"이 필요하지 않습니다.
- J2EE 응용 프로그램(EAR 파일), EJB 모듈(JAR 파일), 웹 모듈(WAR 파일), 커넥터(RAR 파일) 활성화 및 비활성화
- 자원 만들기, 업데이트 및 삭제
- EJB 컨테이너 또는 MDB 컨테이너에 대해 모니터링 사용 가능을 true/false로 설정
- HTTP 및 웹 컨테이너 기능 변경(즉, `server.xml`에서 `http-service`와 `web-container` 및 그 하위 요소에 대한 변경)

응용 프로그램 서버 인스턴스 상태 보기

관리 인터페이스를 사용하면 기본 응용 프로그램 서버 인스턴스뿐만 아니라 서버의 시작 또는 중지 여부를 볼 수 있습니다.

응용 프로그램 서버 인스턴스 상태를 보려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스 이름을 누릅니다.
2. 오른쪽 창에서 "일반" 탭을 누릅니다.

   Sun ONE Application Server 소프트웨어의 호스트 이름, 포트 번호, 설치 디렉토리 및 버전을 비롯하여 서버가 실행 중인지 여부를 볼 수 있습니다.

명령줄 인터페이스를 사용하여 응용 프로그램 서버 인스턴스 상태를 보려면 `asadmin` 유틸리티의 `show-instance-status` 명령을 사용합니다. 상태는 "시작 중", "시작됨", "중지 중" 또는 "중지됨"입니다. 명령 구문은 다음과 같습니다.

`asadmin show-instance-status --user` *admin_user* `[--password` *admin_password*`] [--host` *admin_host*`] [--port` *admin_port*`] [--passwordfile` *file_name*`] [--secure | -s]` *instance_name*

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

JVM 설정 구성

응용 프로그램 서버 인스턴스에 대한 Java 가상 머신(JVM) 설정을 구성할 수 있습니다. 이러한 설정에는 Java 홈의 위치, 컴파일러 옵션, 디버깅 옵션 및 프로파일러 정보가 포함됩니다. 이러한 설정을 구성하는 이유는 성능 향상을 위해서입니다. 성능에 대한 자세한 내용은 *Sun ONE Application Server Performance and Tuning Guide*를 참조하십시오.

이 절에서는 다음 항목에 대해 설명합니다.

- 일반 설정 구성
- 경로 설정 구성
- JVM 옵션 구성
- JVM 프로파일러 구성
- 명령줄 인터페이스를 사용하여 JVM 설정 구성

일반 설정 구성

관리 인터페이스에서 JVM의 일반 옵션을 구성하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스 이름을 누릅니다.
2. 오른쪽 창에서 "JVM" 탭을 누릅니다.
3. "일반"을 누릅니다.
4. Java 홈을 설정합니다.

   Java 홈은 JDK (Java Developer's Kit)가 설치되어 있는 디렉토리의 경로입니다. Sun ONE Application Server는 Sun JDK 1.4.0_02 이상을 지원합니다.
5. 디버깅 활성화 여부를 선택하고 디버그 옵션을 설정합니다.

   디버그 옵션 목록은 http://java.sun.com/products/jpda/doc/conninv.html#Invocation에서 사용 가능합니다.

6. rmic 옵션을 선택합니다.

   rmic 옵션 필드는 응용 프로그램 배포 시 RMI 컴파일러로 전달된 rmic 옵션을 보여줍니다. `-keepgenerated` 옵션은 스텁과 타이에 대해 생성한 소스를 저장합니다. rmic 명령에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guide to Enterprise Java Beans*를 참조하십시오.

7. "저장"을 누릅니다.

경로 설정 구성

관리 인터페이스에서 JVM의 경로 설정을 구성하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스 이름을 누릅니다.
2. 오른쪽 창에서 "JVM" 탭을 누릅니다.
3. "경로 설정"을 누릅니다.
4. 시스템 클래스 경로의 접미어를 선택합니다.
5. 환경 클래스 경로 무시 여부를 선택합니다.

   클래스 경로를 무시하지 않은 경우 CLASSPATH 환경 변수를 읽어 이를 Sun ONE Application Server 클래스 경로에 추가합니다. CLASSPATH 환경 변수는 클래스 경로 접미어 바로 다음에 추가됩니다.

   개발 환경에서 클래스 경로를 사용해야 합니다. 작업 환경에서는 환경 변수의 부작용을 방지하기 위해 이 클래스 경로를 무시해야 합니다.

6. 원시 라이브러리 경로 접두어와 접미어를 설정합니다.

   원시 라이브러리 경로는 Application Server 원시 공유 라이브러리에 대한 설치 상대 경로, 표준 JRE 원시 라이브러리 경로, 셸 환경 설정(UNIX의 LD_LIBRARY_PATH) 및 프로파일러 요소에 지정되어 있는 경로를 자동으로 구성하여 연결한 것입니다. 이렇게 합성되었으므로 원시 라이브러리 경로는 서버 구성에서 명시적으로 표시되지 않습니다.

7. "저장"을 누릅니다.

JVM 옵션 구성

관리 인터페이스에서 JVM 명령줄 옵션을 설정하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스 이름을 누릅니다.
2. 오른쪽 창에서 "JVM" 탭을 누릅니다.
3. JVM 옵션을 누릅니다.
4. JVM 옵션을 추가하려면 화면 맨 위에 있는 텍스트 필드에 옵션을 입력한 다음 "추가"를 누릅니다.
5. JVM 옵션을 삭제하려면 옵션 옆의 확인란을 누르고 "삭제"를 누릅니다.
6. JVM 옵션을 편집하려면 "JVM 옵션" 필드에서 텍스트를 편집하고 "저장"을 누릅니다.

특정 JVM 옵션에 대한 자세한 내용은 http://java.sun.com/docs/hotspot/VMOptions.html을 참조하십시오.

JVM 프로파일러 구성

관리 인터페이스에서 JVM 프로파일러를 구성하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스 이름을 누릅니다.
2. 오른쪽 창에서 "JVM" 탭을 누릅니다.
3. "프로파일러"를 누릅니다.
4. 프로파일러의 이름, 클래스 경로, 원시 라이브러리 경로 및 활성화 여부를 지정합니다.
5. 프로파일러의 JVM 옵션을 추가하려면 화면 맨 위의 텍스트 필드에 옵션을 입력하고 "추가"를 누릅니다.
6. 프로파일러의 JVM 옵션을 삭제하려면 옵션 옆의 확인란을 누르고 "삭제"를 누릅니다.
7. 프로파일러의 JVM 옵션을 편집하려면 "JVM 옵션" 필드에서 텍스트를 편집하고 "저장"을 누릅니다.

프로파일러에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guide*를 참조하십시오.

명령줄 인터페이스를 사용하여 JVM 설정 구성

명령줄 인터페이스의 `asadmin` 유틸리티를 사용하여 JVM 설정을 구성하려면 다음 명령을 사용하십시오.

인스턴스에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

`asadmin> get` *server_instance*`.java-config.*`

server1에서 `classpathprefix`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.java-config.classpathprefix
```

server1에서 `classpathprefix`라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.java-config.classpathprefix=com.sun
```

위의 예에서는 모두 사용자, 암호, 호스트 및 포트를 환경 변수에서 이미 설정했다고 가정합니다. 속성에 대한 전체 목록을 보려면 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

명령줄 인터페이스의 `asadmin` 유틸리티를 사용하여 JVM 옵션을 설정하려면 다음 명령을 사용하십시오.

`asadmin> create-jvm-options --user` *admin_user* `[--password` *admin_password*`]`
`[--host` *host*`] [--port` *port*`] [--secure | -s] [--instance` *instance_name*`]`
`[--profiler=`*true/false*`]`
(*jvm_option_name=jvm_option_value*)`[:`*jvm_option_name=jvm_option_name*`]*`

`asadmin> delete-jvm-options --user` *admin_user* `[--password` *admin_password*`]`
`[--host` *host*`] [--port` *port*`] [--secure | -s] [--instance` *instance_name*`]`
`[--profiler=`*true/false*`]`
(*jvm_option_name=jvm_option_value*)`[:`*jvm_option_name=jvm_option_name*`]*`

주: JVM 옵션을 콜론으로 구분하여 두 개 이상 입력할 수 있습니다. 프로파일러에서 옵션을 사용하는 경우 `--profiler`를 `true`로 설정합니다.

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

로깅 설정 및 모니터링 설정 구성

"로깅" 및 "모니터링" 탭의 설정은 별도의 장에서 설명합니다. 로깅에 대한 자세한 내용은 5장, "로깅 사용"을 참조하십시오. 모니터링 및 SNMP 설정에 대한 자세한 내용은 6장, "Sun ONE Application Server 모니터링"을 참조하십시오.

응용 프로그램 서버 인스턴스 고급 설정 변경

응용 프로그램 서버 인스턴스에는 인스턴스의 로켈(문자 집합 및 언어 같은 설정 결정), 서버 로그 파일의 경로, 배포된 응용 프로그램의 디렉토리에 대한 경로 및 비활성 처리된 Bean 및 지속적 HTTP 세션이 저장되어 있는 세션 저장 디렉토리에 대한 경로를 보여 주는 추가 설정이 있습니다.

또한 응용 프로그램 재로드 기능과 재로드 폴링 간격을 활성화합니다. 동적 응용 프로그램 재로드 기능은 변경할 응용 프로그램을 자동으로 검사하여 변경된 경우 업데이트된 버전을 자동으로 제공합니다. 일반적으로 작업 환경이 아닌 개발 환경에서 동적으로 재로드 기능을 활성화해야 합니다. 폴링 간격은 Application Server에서 업데이트할 응용 프로그램을 검사하는 간격을 지정합니다.

관리 인터페이스를 사용하여 응용 프로그램 서버 인스턴스의 설정을 변경하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스 이름을 누릅니다.
2. 응용 프로그램 서버 인스턴스 페이지에서 "고급" 탭을 누릅니다.
3. 필드에 원하는 값을 입력합니다.
4. "저장"을 누릅니다.

명령줄 인터페이스의 `asadmin` 유틸리티를 사용하여 서버 인스턴스의 고급 설정을 변경하려면 get 및 set 명령을 사용합니다. 서버 인스턴스에 대한 모든 속성을 가져오는 경우

인스턴스에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

`asadmin get` *instance_name*`.*`

예를 들면 다음과 같습니다.

```
asadmin get server1.*
```

`server1`에 대해 `logRoot`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin get server1.logRoot
```

`server1`에 대해 `logRoot`라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin set server1.logRoot=/space/log
```

위의 예에서는 모두 사용자, 암호, 호스트 및 포트를 환경 변수에서 이미 설정했다고 가정합니다. 명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

5장

로깅 사용

이 장에서는 Sun ONE Application Server의 로깅 특징과 기능뿐만 아니라 로깅에 사용되는 구성 요소에 대해 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- 로깅 정보
- UNIX 및 Windows 플랫폼에서 로깅
- 로그 수준 사용
- 가상 서버와 로깅에 대한 정보
- 로거 정보
- 클라이언트측 로깅 정보
- 응용 프로그램 및 서버 로그 출력 리디렉션
- 로그 파일 관리
- 명령줄 인터페이스를 통한 로깅 구성
- 관리 인터페이스를 통한 로깅 구성
- 오류 로깅에 대한 지시문 구성
- 액세스 로그 파일 보기
- 이벤트 로그 파일 보기
- 로그 기본 설정
- 로그 분석기 실행
- 이벤트 보기(Windows 2000 Pro)

로깅 정보

로깅은 응용 프로그램에 사용되는 유용한 디버깅 및 진단 도구입니다. 또한 개발자의 생산성을 높여 줍니다. 응용 프로그램 서버의 고유 로그 출력은 서버 구성 및 배포 문제를 확인하고 진단하는 데 도움을 줍니다.

Sun ONE Application Server의 로깅은 Java 로깅 API를 사용합니다. Sun ONE Application Server는 로깅 정보를 수집하여 `logs` 디렉토리에 있는 두 로그 파일 즉, `access.log`와 `server.log`에 저장합니다. 또한 고유 로그 파일에 로그를 전송할 수 있습니다.

로깅된 메시지는 메시지 자체보다 많은 정보를 제공합니다. 제공되는 추가 정보에는 다음 내용이 포함됩니다.

- 이벤트 날짜와 시간.
- 이벤트의 로그 수준. Appserver에서 지정한 Loglevel ID 또는 이름.
- 프로세스 ID (PID). appserv 프로세스의 PID.
- (선택 사항) 가상 서버 ID (vsid). 메시지를 생성한 vsid.
- 메시지 ID. 하위 시스템 및 네 자리 정수.
- 메시지 데이터.

추가 메시지 정보의 유형과 순서는 로깅에 사용되는 플랫폼과 해당 플랫폼에 사용 가능한 로깅 서비스에 따라 다릅니다. 로깅된 메시지에 대해 가상 서버 ID를 사용하려면 112페이지의 그림 "로그 서비스 구성"을 참조하십시오.

UNIX 및 Windows 플랫폼에서 로깅

이 절에서는 로그 파일을 만드는 방법을 설명합니다. 또한 다음 항목에 대해 설명합니다.

- server.log의 기본 로깅
- syslog를 사용한 로깅
- Windows eventlog를 사용하여 로깅

server.log의 기본 로깅

UNIX와 Windows 플랫폼에서 로그 파일은 `log` 하위 디렉토리의 `server.log`에 만들어집니다. 인스턴스의 모든 서버 구성 요소와 가상 서버에서 전송되는 로그는 이 한 파일에 수집됩니다.

전체 서버에 대해 기본 로그 수준을 설정할 수 있습니다. 그러나 하위 시스템 수준에서 특정한 하위 시스템에 대한 기본 로그 수준을 무시할 수 있습니다. 또한 `stdout` 및 `stderr`을 서버의 이벤트 로그에 리디렉션할 수 있으며, 운영 체제의 시스템 로그에 로그 출력을 전송할 수 있습니다. 또한 `stdout` 및 `stderr` 내용을 서버의 이벤트 로그로 전송할 수 있습니다. 로그 메시지는 기본적으로 지정된 서버 로그 파일뿐만 아니라 `stderr`에 전달됩니다.

또한 로그 메시지와 함께 가상 서버 ID를 기록할 수 있습니다. 이 기능은 여러 가상 서버를 사용하여 동일한 로그 파일에 메시지를 기록하는 경우에 유용합니다. 로그 메시지를 시스템 로그에 기록하도록 선택할 수 있습니다. 이렇게 하면 `server.log` 파일에 로깅이 수행되지 않습니다. 그 대신 UNIX의 `syslog` 로깅 서비스 또는 Windows 플랫폼의 시스템 로깅 서비스를 사용하여 로그를 작성하고 관리합니다.

또한 `server.xml` 속성을 사용하여 이 파일의 내용을 제어할 수 있습니다. `server.xml` 파일에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

server.log의 예

다음은 `server.log`의 예입니다.

타임스탬프, 로그 수준, (PID vsid(선택 사항)): 메시지 ID: 메시지

```
[01/Aug/2002:11:39:31] INFO ( 1224): CORE1116: Sun ONE Application
Server 7.0

[01/Aug/2002:11:39:36] INFO ( 1224): CORE5076: Using [Java
HotSpot(TM) Server VM, Version 1.4.0_02-20020712] from [Sun
Microsystems Inc.]

[01/Aug/2002:11:39:50] INFO ( 1224): JMS5023: JMS service
successfully started. Instance Name = domain1_server1, Home =
[D:\install_7_29\imq\bin].

[01/Aug/2002:11:39:53] INFO ( 1224): CIS0056: Creating TCP
ServerConnection at [EndPoint
[IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]

[01/Aug/2002:11:39:53] INFO ( 1224): CIS0057: Created TCP
ServerConnection at [EndPoint
[IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]
```

```
[01/Aug/2002:11:39:54] INFO ( 1224): CIS0054: Creating TCP
Connection from [-] to [EndPoint

[IIOP_CLEAR_TEXT:192.18.145.66:3700:false]]
```

주

사전 컴파일된 JSP와 관련된 로그 메시지 리디렉션

사전 컴파일된 JSP와 관련된 로그 메시지는 기본적으로 {*domain_root*}/{*domain_name*}/admin-server/logs/server.log에 있는 Administration Server의 로그 파일에 저장됩니다.

모든 메시지는 같은 파일에 기록되므로, 사전 컴파일된 JSP를 사용하여 응용 프로그램을 배포하는 동안 발생하는 예외 또는 오류는 일반 로그 파일의 메시지 볼륨에서 손실될 수 있습니다. 지정된 도메인에서 관할하는 다중 인스턴스에 여러 응용 프로그램을 배포 중인 경우, Admin-Server의 로그 메시지는 특정한 응용 프로그램의 JSP와 관련된 예외가 없는지 주의 깊게 살펴 보아야 합니다. 이로 인해 중복될 수도 있습니다.

따라서 Administration Server의 server.log 파일 대신에 서버 인스턴스의 server.log 파일에서 사전 컴파일된 JSP를 사용하여 배포한 응용 프로그램과 관련된 메시지를 기록하는 것이 좋습니다.

Sun ONE Application Server 인스턴스의 server.log 파일에 로그 메시지를 리디렉션하려면 관리 인터페이스의 로그 파일에 대한 경로를 변경하십시오. 자세한 내용은 로그 서비스 구성을 참조하십시오.

syslog를 사용한 로깅

중앙 집중화된 로깅이 필요한 안정된 운영 환경에서는 syslog가 적절합니다. 로그 출력이 진단 및 디버깅에 자주 필요한 환경의 경우 개별 서버 인스턴스 또는 가상 서버 로그가 더욱 관리하기 수월합니다.

주

- 한 파일의 서버 인스턴스 및 Administration Server에 대해 기록된 모든 데이터는 읽거나 디버깅하기가 어려운 것으로 나타났습니다. 따라서 유연하게 실행 중인 배포된 응용 프로그램에 대해서는 syslog 마스터 로그 파일을 사용하는 것이 좋습니다.
- 로깅된 메시지는 Solaris 데몬 응용 프로그램에서 전송된 다른 모든 로그와 함께 혼합됩니다.

syslogd와 결합된 syslog 로그 파일과 시스템 로그 데몬을 사용하여 다음 작업을 수행하도록 syslog.conf 파일을 구성할 수 있습니다.

- 해당 시스템 로그에 메시지 기록

- 시스템 콘솔에 메시지 기록
- 로깅된 메시지를 사용자 목록에 전달하거나 네트워크를 통해 다른 호스트의 다른 `syslogd`에 전달

주 Sun ONE Application Server를 설치한 후에는 서버에 대한 로그 서비스 요소 속성인 `use-system-logging`을 사용할 수 없습니다. 따라서 기본적으로 로그는 UNIX의 `syslog` 또는 Windows 플랫폼의 Windows Event Log에 전송되지 않습니다. Sun ONE Application Server Configuration File Reference에서 설명한 `server.xml`의 서버 요소에서 이 속성을 사용 가능하게 함으로써 `syslog` 또는 Windows Event Log에 로깅을 전송할 수 있습니다. `use-system-logging`을 설정하기 전에 107페이지의 "로그 파일 관리"를 참조하십시오.

syslog 구성

읽기 및 관리를 더욱 용이하도록 하려면 `/etc` 디렉토리에 있는 `syslog.conf`를 구성하여 심각하지 않은 메시지는 별도의 파일에 전송하여 저장할 수 있습니다.

syslog 구성 방법

1. 심각하지 않은 메시지를 별도의 파일에 전송하여 저장하려면, Solaris의 `syslog.conf` 파일에 다음 명령을 추가합니다.

```
daemon.debug  /var/adm/iasdebug
```

주 로그 메시지가 Windows eventlog에 전송되면 로그 수준이 `INFO`, `WARNING`, `SEVERE`, `ALERT` 또는 `FATAL`인 메시지만 기록됩니다.

2. 접속 끊기 신호를 `syslogd`에 보냅니다. 이 작업은 다음 명령을 사용하여 실행할 수 있습니다.

```
kill -HUP <PID syslogd>
```

3. 관리 인터페이스의 Admin Server로 이동하여 "시스템 로그 옵션에 쓰기"를 선택합니다. 저장한 다음 변경 사항을 적용합니다. 변경 사항이 적용되도록 Admin Server를 다시 시작합니다.

구성된 Solaris `syslog.conf` 파일의 예는 다음과 같습니다.

```
#ident"@(#)syslog.conf1.598/12/14 SMI"/* SunOS 5.0 */
```

```
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (`') names
# that match m4 reserved words.  Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice/dev/sysmsg
*.err;kern.debug;mail.crit/var/adm/messages
daemon.info;daemon.err;daemon.debug;daemon.alert;daemon.crit;daemon
.warning/var/adm/iaslog
daemon.debug/var/adm/iasdebug
#daemon.notice;        /var/adm/iaslognotice
#daemon.warning;       /var/adm/iaslogwarning
#daemon.alert;         /var/adm/iaslogalert
#daemon.err;           /var/adm/iaslogerr

#*.alert;kern.err;daemon.erroperator
#*.alert            root
*.emerg             *

# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.noticeifdef(`LOGHOST', /var/log/authlog, @loghost)

mail.debugifdef(`LOGHOST', /var/log/syslog, @loghost)
#
# non-loghost machines will use the following lines to cause "user"
```

```
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err         /dev/sysmsg
user.err         /var/adm/messages
user.alert       `root, operator'
user.emerg       *
)
```

자세한 내용은 `syslog.conf` 설명서 페이지를 참조하십시오.

`syslog.conf`에 대한 변경 사항이 적용되려면, Sun ONE Application Server를 다시 시작해야 합니다.

`syslog`에 대한 로깅은 모든 Sun ONE Application Server의 로그와 기타 데몬 응용 프로그램의 로그가 동일한 파일에 수집되는 것을 의미하므로 로깅된 메시지는 다음 정보가 보강되어 특정한 서버 또는 가상의 서버 인스턴스에서 전송된 Sun ONE Application Server별 메시지를 확인합니다.

- 고유 메시지 ID
- 타임스탬프
- 인스턴스 이름
- 프로그램 이름(`appservd` 또는 `appserv-wdog`)
- 프로세스 ID (appserv 프로세스의 PID)
- 스레드 ID (선택 사항)
- 서버 ID

`server.xml` 파일에서 서버 인스턴스와 가상 서버 인스턴스에 대해 로그 서비스를 구성할 수 있습니다. 가상 서버 인스턴스에 대한 로그 서비스 구성은 99페이지의 그림 "가상 서버와 로깅에 대한 정보"에서 설명합니다. 서버 인스턴스에 대한 로그 서비스 구성은 112페이지의 그림 "관리 인터페이스를 통한 로깅 구성"에서 설명합니다.

로그 수준은 해당 하위 시스템과 구성 요소에 대한 관리 인터페이스를 통해 구성할 수 있습니다.

UNIX 운영 환경에서 사용되는 `syslog` 로깅 메커니즘에 대한 자세한 내용을 보려면 터미널 프롬프트에서 다음 man 명령을 사용하십시오.

```
man syslog
```

```
man syslogd

man syslog.conf
```

syslog 메시지의 예

다음은 `syslog` 메시지의 예입니다.

타임스탬프, 호스트 이름 [instance_name], [subsystem], [vsid], 메시지 ID, 로그 수준, 메시지 데이터

```
Jul 19 14:33:18 strange /usr/lib/nfs/lockd[164]: [ID 599441
daemon.info] Number of servers not specified. Using default of 20.

Jul 19 14:33:20 strange ntpdate[181]: [ID 558275 daemon.notice]
adjust time server 192.18.56.149 offset 0.06702 6 sec

Jul 19 14:38:13 strange xntpd[248]: [ID 204180 daemon.info]
synchronisation lost

Jul 19 14:38:47 strange server1 appservd[374]: [ID 702911
daemon.info] INFO (  374): CORE1116: Sun ONE Application Server 7.0

Jul 19 14:38:48 strange server1 appservd[374]: [ID 702911
daemon.info] FINE (  374): Collecting statistics for up to 1
processes with 128 threads, 200 listen sockets, and 1000 virtual
servers
```

Windows eventlog를 사용하여 로깅

Windows 운영 환경에서 사용되는 이벤트 로그 메커니즘에 대한 자세한 내용은 키워드 *Event Logging*에 대한 Windows의 도움말 시스템 색인을 참조하십시오.

로그 수준 사용

이 절에서는 로그 수준과 각 Sun ONE Application Server 하위 시스템에 대한 로그 수준 지정 방법에 대해 설명합니다.

다음 항목에 대해 설명합니다.

- 로그 수준 정보
- syslog 구성에 사용된 로그 수준

로그 수준 정보

Sun ONE Application Server는 정보의 선택적 로깅을 위해 표준 JDK 1.4 로그 수준을 사용합니다. Sun ONE Application Server는 표준 JDK 로그 수준뿐만 아니라 `server.log`에 보다 자연스럽게 매핑하고 Solaris와 밀접하게 통합할 수 있도록 설계한 로그 수준을 추가했습니다.

로깅된 메시지는 `server.log`에 라우팅되면 99페이지의 그림 "server.log에 매핑된 Sun ONE Application Server 로그 수준"에 정의된 대로 로그 수준에 매핑됩니다.

주 Admin Server 및 기본 응용 프로그램 서버 인스턴스의 `server.log` 파일 또는 `syslog`에 대한 기본 로그 수준은 INFO입니다. 응용 프로그램 서버 인스턴스에 대해 기본 로그 수준을 사용하면 오류 및 정보 메시지가 기록됩니다. 해당 메시지를 로깅하지 않으려면 `server.xml` 파일 또는 Admin Server 및 서버 인스턴스의 관리 인터페이스에서 로그 수준을 WARNING 또는 SEVERE로 변경하십시오.

서버 전체의 기본 로그 수준은 로그 서비스 요소로 설정할 수 있습니다. 이는 로그 수준이 "기본값"으로 설정된 모든 요소에 영향을 줍니다.

로깅이 가능한 각 Sun ONE Application Server 하위 시스템에 대해 로그 수준을 지정할 수 있습니다. 로그 수준은 런타임 중에 기록되는 메시지 정보의 양을 간소화하는 데 유용합니다. 이러한 로그 수준은 관련 하위 시스템의 `server.xml` 파일에서 지정됩니다. 선택된 하위 시스템의 관리 인터페이스에서 로그 수준을 지정하거나 선택된 하위 시스템에 대해 원하는 로그 수준을 설정하기 위해 `server.xml` 파일을 직접 편집할 수 있습니다.

주의 `server.xml` 파일을 수동으로 편집하면 서버 시작 오류를 가져올 수 있는 구문 오류가 발생할 수 있습니다. 구성 파일을 수동으로 편집하는 지침에 대해서는 *Sun ONE Application Server Administrator's Configuration File Reference*의 "Manually Editing Configuration Files" 섹션을 참조하십시오.

관리 인터페이스를 통한 로그 수준 설정의 예는 그림 102페이지의 그림 "JMS 서비스에 대한 로그 수준"을 참조하십시오. `server.xml` 파일에서 각 하위 시스템 또는 구성 요소에 대한 로그 수준을 직접 설정하려면 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

표 98페이지의 그림 "로그 수준"에서 설명하는 로그 수준은 JDK1.4 로깅 API 사양의 요구 사항과 일치합니다. 그러나 ALERT 및 FATAL과 같은 로그 수준은 Sun ONE Application Server에만 적용되므로 JDK1.4 로깅 API에서는 구현되지 않습니다.

다음 표에서는 심각도가 높아지는 순서로 Sun ONE Application Server의 로그 수준과 메시지를 정의합니다. 왼쪽 열에서는 Sun ONE Application Server의 로그 수준을 나열하고 오른쪽 열에서는 각 로그 수준에 대해 간단히 설명합니다.

로그 수준

로그 수준	설명
FINEST FINER FINE	메시지는 디버그 메시지의 상세 표시 정도를 나타냅니다. FINEST는 가장 상세한 표시를 나타냅니다.
CONFIG	메시지는 다양한 정적 구성 정보를 설명하며, 특정한 구성과 연관될 수 있는 디버깅 문제를 지원합니다.
INFO	메시지는 기본적으로 정보를 제공하며 일반적으로 서버 구성 또는 서버 상태와 관련이 있습니다. 이 메시지는 즉각적인 조치가 필요한 오류를 나타내지 않습니다. 예를 들어, MessageBroker에 새 항목 작성과 같은 구성 변경 알림이 수신되었다는 메시지를 기록할 수 있습니다.
WARNING	메시지는 경고를 나타냅니다. 이 메시지에는 예외가 함께 제공될 수 있습니다.
SEVERE	메시지는 일반 응용 프로그램의 실행을 방해할 수도 있는 상당히 중요한 이벤트를 표시합니다.
ALERT*	메시지는 사용자에게 특정한 조치를 취하라고 경고합니다.
FATAL*	메시지는 치명적인 오류를 표시하며 이 메시지가 기록된 이후에는 서버를 실행하지 않는 것이 좋습니다. 이론상 이 메시지는 서버 충돌 전에 전송되는 마지막 메시지가 됩니다.

* Sun ONE Application Server별 로그 수준

주	INFO보다 낮은 로그 수준(FINEST, FINER, FINE 및 CONFIG)의 모든 메시지는 디버깅과 관련된 문제에 도움이 되는 정보를 제공하며, 기술 지원의 조언에 따라 활성화해야 합니다. INFO보다 낮은 로그 수준의 메시지는 일반적으로 현지화하지 않습니다.

syslog 구성에 사용된 로그 수준

다음 표에는 `syslog`를 사용할 때 Sun ONE Application Server 내에서 구성할 수 있는 로그 수준의 목록이 포함되어 있습니다. 왼쪽 열에는 Sun ONE Application Server의 로그 수준이 나열되어 있고 오른쪽 열에는 `syslog` 기능의 해당 로그 수준이 나열되어 있습니다.

server.log에 매핑된 Sun ONE Application Server 로그 수준

Sun ONE Application Server	syslog 수준
FINEST	LOG_DEBUG
FINER	LOG_DEBUG
FINE	LOG_DEBUG
CONFIG	LOG_INFO
INFO (default)	LOG_INFO
WARNING	LOG_WARNING
SEVERE	LOG_ERR
ALERT	LOG_ALERT
FATAL	LOG_CRIT

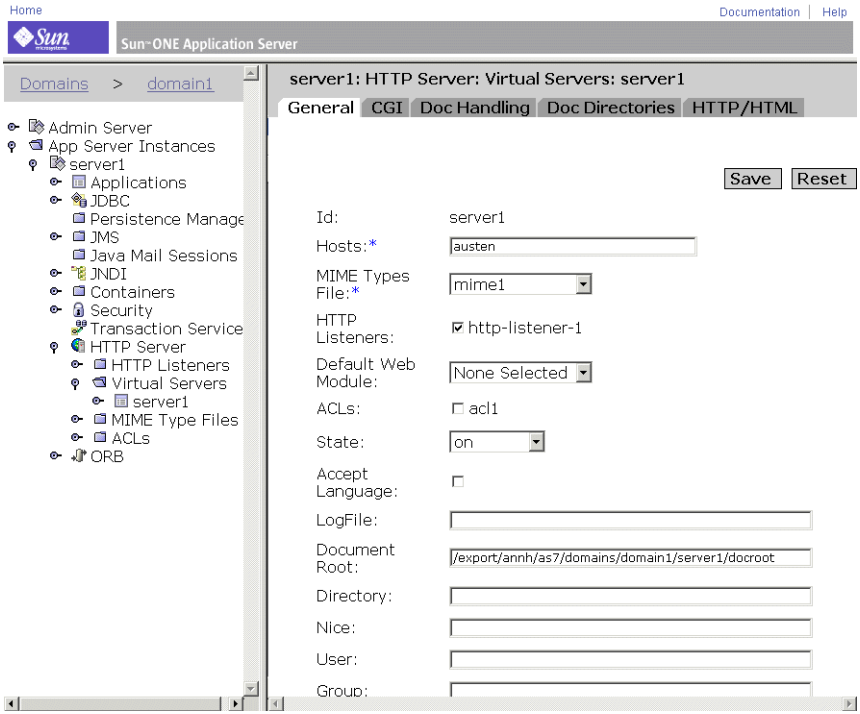
가상 서버와 로깅에 대한 정보

Sun ONE Application Server는 가상 서버 인스턴스를 가질 수 있습니다. Sun ONE Application Server 인스턴스의 각 가상 서버는 고유한 ID를 가지며, 고유 로그 파일을 가질 수 있습니다. 각 가상 서버에 대한 개별 로그 파일을 사용하면 특정 트랜잭션 및 자원에 대한 서버 동작을 추적할 수 있습니다.

관리 인터페이스에서 가상 서버에 대해 로그 파일 이름을 지정하려면, 디렉토리 트리에서 HTTP Server 링크로 이동하여 가상 서버 폴더 아래에 있는 서버 인스턴스 요소를 열어 오른쪽 프레임에서 "일반" 탭을 표시합니다. "로그 파일" 필드에서 이 가상 서버에 대한 로그 파일의 경로와 이름을 입력할 수 있습니다. 그림 100페이지의 그림 "가상 서버 로그 파일 이름 설정"은 이 설정이 있는 위치를 보여줍니다.

주 로깅이 가능하고 응용 프로그램이 실행 중인 경우, 응용 프로그램에서 로깅된 메시지가 가상 서버 ID 없이 기록됩니다.

가상 서버 로그 파일 이름 설정



또한 여러 가상 서버의 로깅된 메시지를 하나의 서버 로그 파일로 전송할 수 있습니다. 이런 경우, `server.xml` 파일의 `log service` 요소에 있는 `log-virtual-server-id`를 활성화할 수 있습니다. 이렇게 하면 사용자는 서로 다른 가상 서버에서 전송된 로그 메시지를 구별할 수 있습니다.

```
<log-service level="FINEST" log-stdout="false" log-stderr="false"
echo-log-messages-to-stderr="false" create-console="false"
log-virtual-server-id="true" use-system-logging="false">

</log-service>

   <http-listener>

      <virtual-server-class>

         <virtual-server id="server1"
http-listeners="http-listener-1" hosts="strange" mime="mime1"
state="on" accept-language="false"/>

         <virtual-server id="server2" hosts="strange"
mime="mime1"/>
```

```
        </virtual-server-class>
    </http-listener>
```

이 예에서 `<log-service log-virtual-server-id="true">`는 모든 로그 메시지에 `virtual_server_id`를 포함하는 역할을 합니다. 이렇게 함으로써 서로 다른 가상 서버에서 보내온 메시지를 구별할 수 있습니다. virtual-server 요소에 "log-file" 속성이 없으면, 모든 가상 서버는 단일 파일에 메시지를 기록합니다.

로거 정보

하위 시스템 수준에서 로깅을 선택적으로 활성화하거나 비활성화할 수 있습니다. 각 하위 시스템에 대한 로깅 제어는 *Sun ONE Application Server Configuration File Reference*에 설명된 대로 `server.xml` 파일에서 지정됩니다. 각 하위 시스템에는 JDK1.4 로깅 API의 요구 사항에 일치하는 고유 로거가 있습니다.

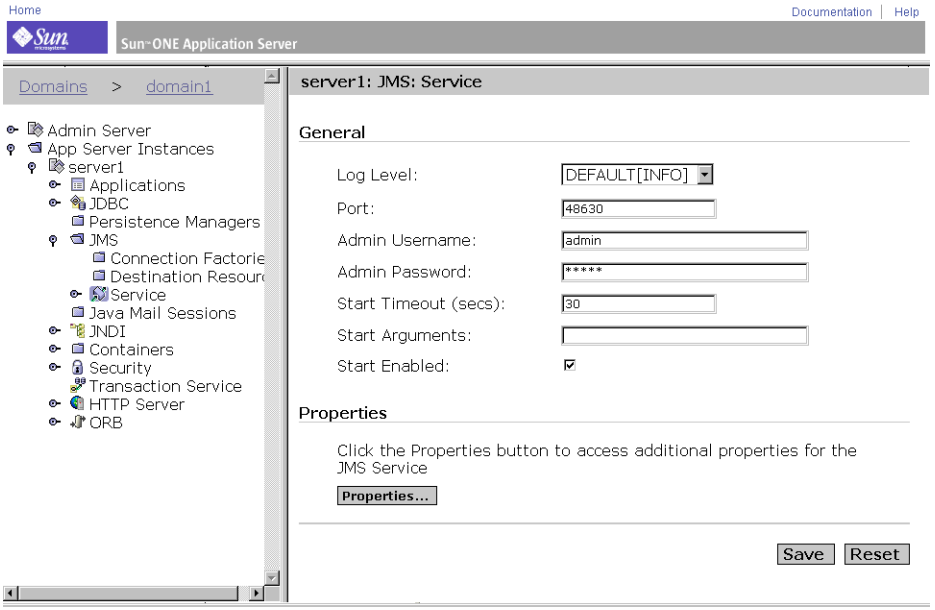
다음 표에서 왼쪽 열에는 하위 시스템이 표시되어 있고 오른쪽 열에는 각 하위 시스템의 `server.xml` 파일의 요소가 표시되어 있습니다.

Sun ONE Application Server의 하위 시스템과 위치

하위 시스템	요소
Administration Server	<admin-service>
EJB 컨테이너	<ejb-container>
웹 컨테이너	<web-container>
MDB 컨테이너	<mdb-container>
Sun ONE Message Queue (JMS 서비스)	<jms-service>
보안 서비스	<security-service>
Java Transaction Service (JTS)	<transaction-service>
Object Request Broker (ORB)	<iiop-service>
기본 처리기[1]	<log-service>

1. 기본 처리기는 유틸리티 클래스와 같이 특정 하위 시스템과 관련이 없는 모든 `server.xml` 항목과 관련된 기본 로거를 말합니다.

JMS 서비스에 대한 로그 수준



주 Windows 플랫폼에서 Windows `server.log`에 로그를 전송할 경우, 로그 수준이 `INFO`, `WARNING`, `SEVERE`, `ALERT` 또는 `FATAL`인 메시지만 Windows Event Log에 기록됩니다.

표 98페이지의 그림 "로그 수준"은 Sun ONE Application Server의 메시지에 대한 심각도가 높아지는 순서에 따라 로그 수준을 정의합니다. 이러한 로그 수준은 JDK1.4 로깅 API 사양의 요구 사항과 일치합니다. 또한 ALERT 및 FATAL과 같은 로그 수준은 Sun ONE Application Server에만 적용되므로 JDK1.4 로깅 API에서는 지원되지 않습니다.

클라이언트측 로깅 정보

Application Client Container (ACC)는 고유한 로그 서비스를 갖고 있으며, 로컬 파일에만 기록할 수 있습니다.

일반적으로 ACC는 응용 프로그램 서버의 다른 호스트에서 고유 프로세스에 따라 실행됩니다. 따라서 ACC에는 고유한 로깅 인프라 및 로그 파일이 있습니다. ACC 구성은 `sun-acc.xml` 파일에 보존됩니다.

ACC에 대한 클라이언트 하위 시스템 로깅 요소는 `log-service`입니다. 다음 표에는 요소 및 속성의 지정된 기본값과 값 범위가 표시되어 있습니다.

ACC 로깅 요소

요소	속성	설명
`log-service`	파일	ACC 로그 파일, 이 파일이 비어 있거나 빠진 경우, `stdout`에 기록
`log-service`	수준	ACC 로그 수준

`sun-acc.xml` 파일의 예는 *Sun ONE Configuration File Reference*를 참조하십시오.

응용 프로그램 및 서버 로그 출력 리디렉션

개발자는 응용 프로그램 구성 요소와 J2EE 응용 프로그램에 대한 단위 테스트를 하는 동안 응용 프로그램 로그 및 서버 로그를 쉽게 사용할 수 있어야 합니다. Windows 플랫폼에서 개발자는 서버 로그 메시지가 데스크탑의 명령 창에 표시되는 것을 선호합니다. UNIX 플래폼의 경우, 개발자는 대체로 서버 인스턴스가 시작되는 단말기 창의 `stderr`로 로그 메시지를 전송하거나, 명령 옵션 `-f`를 사용하여 로그 파일에 기록된 로그 메시지를 보는 것을 선호합니다.

`server.xml` 파일에는 로깅된 메시지를 로그 파일이나 단말기 창 등으로 전송하도록 `stdout`와 `stderr`에 대해 설정할 수 있는 속성이 포함되어 있습니다. `stdout` 및 `stderr`의 사용에 대한 자세한 내용은 *Sun ONE Application Server Configuration File Reference*를 참조하십시오.

로그 서비스에 대한 자세한 내용은 112페이지의 그림 "로그 서비스 구성"을 참조하십시오.

로그 파일 관리

액세스 및 이벤트 로그(`server.log`) 파일이 자동으로 아카이브되도록 설정할 수 있습니다. 일정한 시간에 또는 특정한 간격이 지난 후에 로그는 회전합니다. Sun ONE Application Server는 이전의 로그 파일을 저장하고, 저장 날짜와 시간이 포함된 이름을 저장된 파일에 지정합니다.

주 여러 가상 서버를 만들고 각 가상 서버에 대해 로그 파일을 연결할 수 있지만 개별 가상 서버에 대한 로그 회전 설정은 지원되지 않습니다.

예를 들어, 매시간 액세스 로그 파일이 회전하도록 설정하면, Sun ONE Application Server가 이 파일을 저장하고 "`access.199907152400`"라는 이름을 지정합니다. 이 이름은 로그 파일의 이름과 년, 월, 일, 24시간을 한데 묶어 하나의 문자열로 만든 것입니다. 로그 아카이브 파일의 정확한 형식은 설정한 로그 회전의 유형에 따라 다양합니다.

주 이 기능은 기본적으로 비 Solaris 플랫폼에 제공됩니다.

Solaris에서 이 기능은 기본적으로 사용할 수 없으며, Solaris 9의 `logadm`과 같은 원시 Solaris 운영 체제 로그 관리 기능을 사용해야 합니다. Solaris 8에서는 로그 관리에 대한 기본 설정 유틸리티는 “Solaris cron 유틸리티를 사용하여 logadm 실행 예약”에서 설명한 `cron` 기능입니다.

운영 체제에 따라 로그 회전을 수행할 수 있는 네 가지 방법이 있습니다. 다음 절에서 이 방법에 대해 설명합니다. 다음 항목에 대해 설명합니다.

UNIX와 Windows의 경우

- 내부 데몬 로그 회전
- 스케줄러 기반 로그 회전

Solaris 9의 경우

- Solaris `logadm` 유틸리티 사용 자세한 내용은 106페이지의 그림 "Solaris logadm 유틸리티를 사용하여 회전"을 참조하십시오.

Solaris의 경우(모든 버전)

- Solaris `cron` 유틸리티 사용 자세한 내용은 109페이지의 그림 "Solaris "cron" 유틸리티를 사용하여 회전"을 참조하십시오.

내부 데몬 로그 회전

내부 데몬 로그 회전은 UNIX와 Windows 운영 체제에 모두 사용할 수 있습니다. 내부 데몬 로그 회전은 HTTP 데몬에서 일어나며, 서버 인스턴스 시작 시간에만 구성할 수 있습니다. 이 방법을 사용하여 회전되는 로그는 다음 형식으로 저장됩니다.

`access.<YYYY><MM><DD><HHMM>`

`error. <YYYY><MM><DD><HHMM>`

로그 파일을 회전하고 새 로그 파일을 시작하기 위한 기초로 사용하는 시간을 지정할 수 있습니다. 예를 들어, 회전 시작 시간이 오전 12시이고 회전 간격이 1440분(하루)인 경우 새 로그 파일은 현재 시간과 상관없이 저장 시 즉시 만들어지고 회전 시작 시간까지 정보를 수집합니다. 그리고 로그 파일은 매일 오전 12시에 회전하고 액세스 로그가 오전 12시에 기록되며 `access.199907152400`이라는 이름으로 저장됩니다. 마찬가지로 회전 간격을 240분(4시간)으로 설정하면, 회전은 오전 12시에 시작하고 액세스 로그 파일에는 4시간 간격(오전 12시에서 오전 4시까지, 오전 4시에서 오전 8시까지 등)으로 수집된 정보가 포함됩니다.

로그 회전이 가능하면 서버가 시작할 때 로그 파일 회전이 시작됩니다. 첫 번째로 회전되는 로그 파일은 현재 시간에서 다음 회전 시간까지 정보를 수집합니다. 이전의 예를 사용하여 시작 시간을 오전 12시로, 회전 간격을 240분으로 설정한 상태에서 현재 시간이 오전 6시인 경우 회전되는 첫 번째 로그 파일은 오전 6시에서 오전 8시까지 수집된 정보를 포함하게 되고, 다음 로그 파일은 오전 8시에서 오후 12시(정오)까지 수집된 정보를 포함하는 식으로 진행됩니다.

스케줄러 기반 로그 회전

스케줄러 로그 회전을 통해 로그 파일을 즉시 아카이브하거나 특정한 날 특정한 시간에 서버에서 로그 파일을 아카이브할 수 있습니다. 로그 파일을 즉시 아카이브하려면 관리 인터페이스의 왼쪽 창에서 "Admin Server"를 선택합니다. 그런 다음 오른쪽 페이지의 상단에서 로깅 링크를 누릅니다. 그 다음 "로그 회전"을 누릅니다. 마지막으로 "아카이브"를 누릅니다.

스케줄러 메소드를 사용하여 회전된 로그는 뒷부분에 파일이 회전된 날짜와 시간이 포함된 원래의 파일 이름으로 저장됩니다. 예를 들어, 로그 파일이 오후 4시 30분에 회전하면 `access`는 `access.24Apr-0430PM`이 될 수 있습니다.

서버가 시작할 때 로그 회전이 시작됩니다. 회전이 활성화되면 Sun ONE Application Server는 시간이 표시된 액세스 로그 파일을 만들고 서버가 시작할 때 회전이 시작됩니다.

회전이 시작되면 Sun ONE Application Server는 액세스 또는 오류 로그 파일에 기록해야 하는 요청 또는 오류가 있는 경우 새로운 시간이 표시된 로그 파일을 만들고, 미리 예약된 "다음 회전 시간" 이후에 기록합니다.

주	Windows 플랫폼의 경우와 Solaris에서 `syslog` 이외의 파일에 전송되는 서버 로깅의 경우, 서버 로그를 아카이브해야 합니다.

로그 파일을 아카이브하고 `schedulerd` 제어 메소드의 사용을 지정하려면, 관리 인터페이스의 왼쪽 창에서 "Admin Server"를 선택합니다. 그런 다음 오른쪽 페이지의 상단에서 로깅 링크를 누르고, "스케줄러 기반 로그 회전" 상자를 누릅니다. 마지막으로 "확인"을 누릅니다. 스케줄러의 현재 상태가 나타납니다.

Solaris logadm 유틸리티를 사용하여 회전

Solaris 9 운영 체제에는 로깅된 메시지와 함께 기능 배열을 수행하는 데 사용할 수 있는 `logadm` 유틸리티가 통합되어 있습니다.

Sun ONE Application Server에만 있는 이 유틸리티는 110페이지의 그림 "Solaris cron 유틸리티를 사용하여 logadm 실행 예약"에서 설명한 Solaris cron 유틸리티에서 실행되는 경우 로그 회전 작업을 수행하는 데 유용합니다.

로그 파일과 관련해 다음과 같은 로그 회전 세부 정보를 지정할 수 있습니다.

- 회전되어야 하는 시스템의 모든 로그 파일 이름
- 회전 간격
- 회전을 트리거하는 조건
- 저장할 백업 로그 파일의 수
- 저장할 백업 로그 파일의 명명 규칙

위의 세부 정보는 다음 위치에 있는 `logadm.conf` 파일에 지정됩니다.

```
n /etc/logadm.conf
```

샘플 `logadm.conf` 파일은 다음과 같습니다.

```
# Copyright 2001-2002 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)logadm.conf 1.2 02/02/13 SMI"
```

```
#
# logadm.conf
#
# Default settings for system log file management.
# The -w option to logadm(1M) is the preferred way to write to this
# file,
# but if you do edit it by hand, use "logadm -V" to check it for
# errors.
#
# The format of lines in this file is:
# <logname> <;options>
# For each logname listed here, the default options to logadm
# are given.  Options given on the logadm command line override
# the defaults contained in this file.
# # logadm typically runs early every morning via an entry in
# root's crontab (see crontab(1)).
#
/var/log/syslog -C 8 -P 'Tue Jul 9 10:10:00 2002' -a 'kill -HUP `cat
/var/run/syslog.pid`' /var/adm/messages -C 4 -P 'Tue Jul 30 10:10:00
2002' -a
'kill -HUP `cat /var/run/syslog.pid`' /var/cron/log -c -s 512k -t
/var/cron/olog
/var/lp/logs/lpsched -C 2 -N -t '$file.$N'
#
# The entry below is used by turnacct(1M)
#
/var/adm/pacct -C 0 -N -a '/usr/lib/acct/accton pacct' -g adm -m 664
-o adm -p never
#
# The entry below will rotate SUN One application server's default
logfile
```

```
# every day provided the current logfile size is >= 512k.  It will
compress

# the old log file before archiving it and also delete the old files
after 30

# days. The compression is done with gzip(1) and the resulting log
file has

# the suffix of .gz.

/var/appserver/domains/domain1/server1/logs/server.log -A 30d -s
512k -p 1d -z
```

또는 대화형으로 `logadm` 명령을 호출하여 특정 파일에서 로그 회전을 시작할 수 있습니다.

다음 예에서는 `syslog`를 회전하고 8개 로그 파일을 유지합니다. 이전의 로그 파일은 `/var/log` 대신에 `/var/oldlogs` 디렉토리에 둡니다.

```
% logadm -C8 -t'/var/oldlogs/syslog.$n' /var/log/syslog
```

또한 대화형 명령줄 옵션을 사용하여 `/etc/logadm.conf`로 지정된 파일에서 회전을 호출할 수 있지만 다른 옵션 또는 수정된 옵션을 사용해야 합니다.

`/etc/logadm.conf`와 명령줄에 옵션을 지정한 경우, `/etc/logadm.conf` 파일의 옵션이 먼저 적용됩니다. 따라서 명령줄 옵션은 `/etc/logadm.conf`의 옵션을 무시합니다. 이 예는 다음과 같습니다.

```
% logadm /var/appserver/domains/domain1/server1/logs/server.log -p
now
```

위의 명령은 `/etc/logadm.conf`의 해당 파일에 제공되는 모든 옵션을 사용하여 지정된 파일을 회전합니다.

주 여러 옵션이 동시에 지정된 경우에는 묵시적으로 이들 옵션 간에 AND가 있는 것으로 간주됩니다. 따라서 로그가 회전되기 전에 모든 조건이 충족되어야 합니다.

`logadm` 유틸리티 및 해당 옵션에 대한 자세한 내용은 다음 설명서 페이지를 참조하십시오.

```
% man logadm
```

또는

```
% logadm -h
```

Solaris "cron" 유틸리티를 사용하여 회전

Solaris 8에서 `cron` 유틸리티를 사용하여 응용 프로그램 서버 로그 회전을 수행할 수 있습니다. 이 작업은 다음 명령을 사용하여 실행할 수 있습니다.

```
% crontab -e
```

이 명령은 cron 항목 목록을 제공할 수 있도록 자주 사용하는 편집기(`env.` 변수 $EDITOR로 정의)를 시작합니다.

주 또한 이 명령은 편집기를 종료한 즉시 `/etc/cron.d/logchecker` 스크립트를 호출합니다. 이 스크립트는 cron 데몬에 `changed/new` crontab 항목을 제공합니다. 따라서 이 방법으로 추가된 항목은 즉시 cron 데몬에서 선택되고 로그 회전이 곧바로 시작됩니다.

로그 회전을 활성화하기 위해 cron 데몬을 다시 시작할 필요는 없습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- crontab 항목 형식 정보
- Solaris cron 유틸리티를 사용하여 logadm 실행 예약

crontab 항목 형식 정보

`crontab` 파일은 각각 6개 필드 줄로 구성되어 있습니다. 필드는 공백이나 탭으로 구분됩니다. 첫 번째 5개 필드는 다음을 지정하는 정수 패턴입니다.

- 분(0-59)
- 시간(0-23),
- 일(1-31),
- 월(1-12),
- 요일 (0-6, 0=일요일).

이 형식을 사용하여 일/요일/월의 지정된 시간에 회전하고 회전 반복을 예약하는 액세스 및 이벤트 로그 파일을 지정할 수 있습니다. 예를 들어,

0 0 * * 1-5

```
/opt/SUNWappserver7/appserver/domains/domain1/server1/bin/rotatelogs
```

0 12 * * 1-5

```
/opt/SUNWappserver7/appserver/domains/domain1/server1/bin/rotatelogs
```

0 * * * 1-5

```
/opt/SUNWappserver7/appserver/domains/domain1/mainserver/bin/rotatelogs
```

이 지정에서는 server1의 액세스 및 로그 파일은 월요일부터 금요일까지 매일 자정과 정오에 회전하며, 주 서버의 액세스 및 로그 파일은 월요일부터 금요일까지 매일 매시간 회전합니다.

crontab 파일은 `/var/spool/cron/crontabs/` 아래에 저장됩니다. crontab 파일은 최종 사용자 또는 루트로 만들 수 있습니다. 사용자의 권한에 따라 다음 명령을 사용하여 crontab 항목을 볼 수 있습니다.

`% crontab -l` *username*

Solaris cron 유틸리티를 사용하여 logadm 실행 예약

`cron` 명령은 특정한 날짜와 시간에 명령을 실행하는 프로세스를 시작합니다. `/var/spool/cron/crontabs` 디렉토리의 `crontab` 파일에서 찾을 수 있는 지침에 따라 정기적으로 예약된 명령을 지정할 수 있습니다.

`cron`과 함께 사용되는 정기적으로 예약된 명령의 예로서 `crontab`의 다음 항목은 매일 자정에 `logadm`를 시작합니다.

```
0 0 * * 0-6 logadm
```

사용자는 `crontab(1)` 명령을 사용하여 고유 `crontab` 파일을 제출할 수 있습니다.

`cron`에서 취한 모든 조치를 기록하려면 CRONLOG=YES (기본값)를 `/etc/default/cron` 파일에 지정해야 합니다. `/etc/cron.d/logchecker`는 로그 파일이 시스템 `ulimit`를 초과하는지 여부를 확인하는 스크립트입니다. 초과하는 경우, 로그 파일은 `/var/cron/olog`로 이동합니다.

명령줄 인터페이스를 통한 로깅 구성

서버 인스턴스와 가상 서버 인스턴스에 대해 명령줄에서 로깅 서비스 부분을 구성할 수 있습니다.

주 이 절에서 제공되는 모든 명령의 예는 환경 변수가 설정되어 있음을 전제로 합니다.

서버 인스턴스에 대해 모든 log-service 속성을 가지려면 다음 명령을 사용합니다.

```
asadmin> get instance_name.log-service.*
```

log-service 속성의 정의는 또한 표 114페이지의 그림 "로그 서비스 속성"을 참조하십시오.

이 명령을 지정된 서버 인스턴스 이름과 함께 사용하는 예는 다음과 같습니다.

```
asadmin> get server1.log-service.*
```

server1의 로깅 서비스에 대한 속성 목록이 반환됩니다. `set` 명령을 사용하여 나열된 각 속성을 구성할 수 있습니다.

가상 서버 인스턴스에 대한 가상 서버 ID의 로깅을 활성화하려면 터미널 프롬프트에 다음 명령을 입력합니다.

```
asadmin> get instance_name.LogVirtualServerId
```

LogVirtualServerId의 현재 상태가 반환됩니다. 상태가 false이면 다음과 같이 `set` 명령을 사용하여 LogVirtualServerId를 활성화할 수 있습니다.

```
asadmin> set instance_name.LogVirtualServerId=true
```

가상 서버 인스턴스에 대한 로그 파일 이름을 설정하려면 다음과 같이 `set` 명령을 사용합니다.

```
asadmin> set instance_name.virtual-server.<virtual server
id>.logFile=<log file>
```

한 예로서 다음의 set 로그 파일 명령이 발행됩니다.

```
asadmin> set
instance2.virtual-server.instance2.logFile=/space/IAs7se/appserver7/appse
rv/domains/domain1/instance2/logs/log
```

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오.

`asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

관리 인터페이스를 통한 로깅 구성

이 절에서는 Sun ONE Application Server 관리 인터페이스을 통하여 서버측(전역) 요소, 지시문, 응용 프로그램 구성 요소에 대한 사용 가능한 로깅 서비스 옵션을 구성하는 작업에 대해 설명합니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 로그 서비스 구성
- 응용 프로그램 서버 구성 요소 및 하위 시스템에 대한 로깅 구성
- 오류 로깅에 대한 지시문 구성

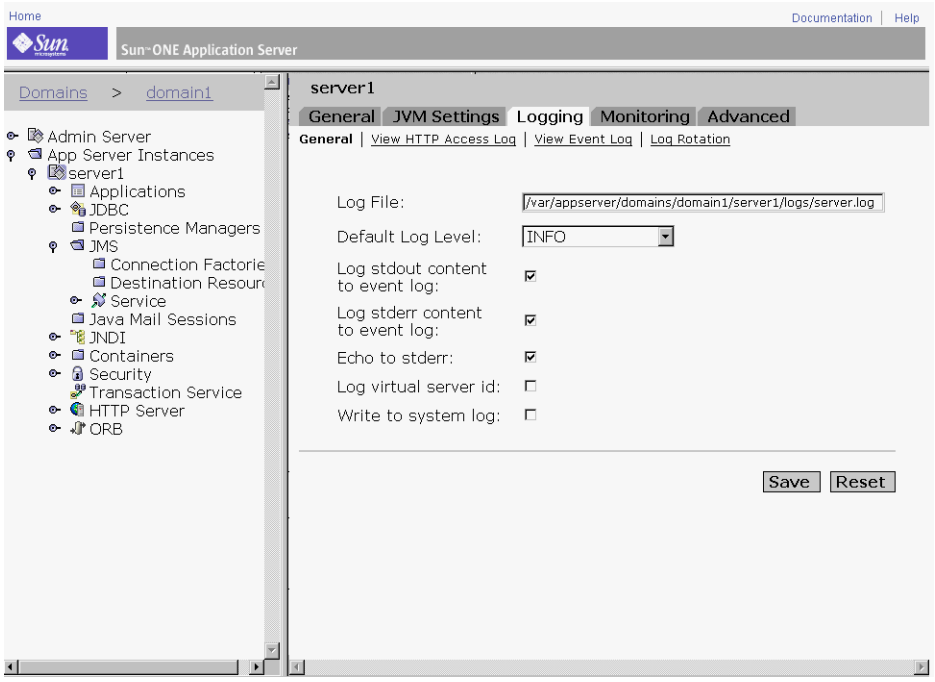
로그 서비스 구성

로그 서비스는 *Sun ONE Application Server Configuration File Reference*에 설명된 대로 `server.xml` 파일의 J2EE 서비스 요소 범주에 있는 요소입니다. 로그 서비스는 시스템 로깅 서비스를 구성하는 데 사용되며, 다음 로그 파일을 포함합니다.

- 서버 로그
- 액세스 로그
- 트랜잭션 로그
- 가상 서버 로그

시스템 로깅 서비스 구성은 로그 서비스 요소의 속성 값 지정을 포함합니다.

서비스 인스턴스에 대한 로그 서비스 관리



113페이지의 그림 "서비스 인스턴스에 대한 로그 서비스 관리"에서처럼 관리 인터페이스를 통해 로그 서비스 요소에 대한 다음 속성을 구성할 수 있습니다.

- 로그 파일
- 기본 로그 수준
- 이벤트 로그에 표준 출력 내용 기록
- 이벤트 로그에 표준 오류 내용 기록
- 표준 오류에 반향
- 콘솔 만들기
- 가상 서버 ID 기록
- 시스템 로그에 쓰기

로그 서비스 링크는 관리 인터페이스의 왼쪽 창에 있는 서버 인스턴스의 확장된 트리 계층에서 액세스할 수 있습니다. 다음 표에서는 허용되는 기본값 및 값 범위와 구성할 수 있는 각 속성에 대해 설명합니다.

로그 서비스 속성

속성	기본값	설명
file	server.log[1]	(선택 사항) 서버 로그의 이름 또는 위치를 무시합니다. 서버 실행에 사용되는 사용자 계정에 상관없이 서버 로그가 유지되는 파일 및 디렉토리를 기록할 수 있어야 합니다.
level	INFO	(선택 사항) 다른 요소에 의해 서버 로그에 로깅되는 메시지의 기본 유형을 제어합니다. 허용된 값은 가장 높은 값에서 가장 낮은 값 순으로 다음과 같습니다. FINEST, FINER, FINE, CONFIG, INFO, WARNING, SEVERE, ALERT, FATAL. 각 값은 보다 낮은 값에 해당되는 모든 메시지를 기록합니다. 예를 들어, FINEST는 모든 메시지를 기록하는 반면, FATAL은 FATAL 메시지만 기록합니다. 기본값은 INFO이며, INFO, WARNING, SEVERE, ALERT 및 FATAL 메시지를 모두 기록합니다.
log-stdout	True	(선택 사항) 값이 true이면 서버 로그에 stdout 출력을 리디렉션합니다. 유효한 값은 on, off, yes, no, 1, 0, true, false입니다.
log-stderr	True	(선택 사항) 값이 true이면, 서버 로그에 stderr 출력을 리디렉션합니다. 유효한 값은 on, off, yes, no, 1, 0, true, false입니다.
echo-log-messages-to-stderr	True	(선택 사항) 값이 true이면, 서버 로그와 stderr에 로그 메시지를 전송합니다. 유효한 값은 on, off, yes, no, 1, 0, true, false입니다.
create-console	False	(선택 사항) 값이 true이면, stderr 출력을 위해 Windows 운영 체제에서 콘솔 창을 만듭니다. 유효한 값은 on, off, yes, no, 1, 0, true, false입니다.
log-virtual-server-id	False	(선택 사항) 값이 true이면, 가상 서버 ID가 가상 서버 로그에 표시됩니다. 이 속성은 여러 virtual-server 요소가 동일한 로그 파일을 공유하는 경우에 유용합니다.
use-system-log	False	값이 true이면, UNIX syslog 서비스 또는 Windows Event Logging을 사용하여 로그를 만들고 관리합니다.

1. 서버 요소의 log-root 속성에 의해 지정된 디렉토리에서

응용 프로그램 서버 구성 요소 및 하위 시스템에 대한 로깅 구성

이 절은 Sun ONE Application Server 구성 요소와 하위 시스템에 대해 로깅을 활성화하고 로그 수준을 선택하는 방법에 대해 설명합니다. Java Transaction Service 구성 요소에는 두 개 이상의 로그 파일이 있습니다. 대부분의 구성 요소와 하위 시스템은 로그 수준 구성과 관련하여 같은 방법으로 처리되므로, 로그 수준을 선택하는 절차는 구성 요소와 하위 시스템의 지정된 그룹에 대해 단 한 번 문서화됩니다.

다음 구성 요소와 하위 시스템은 서버 메시지에 대한 선택적 로깅을 사용할 수 있습니다. 구성 요소와 하위 시스템에 대한 자세한 내용은 지시되어 있는 이 설명서의 다른 항목을 참조하십시오.

- ORB - Corba 기반 클라이언트에 대한 지원 구성
- 웹 컨테이너 - J2EE 서비스 구성
- EJB 컨테이너 - J2EE 서비스 구성
- MDB 컨테이너 - J2EE 서비스 구성(EJB 컨테이너 내부)
- Java Transaction Service - J2EE 서비스 구성
- JMS 서비스 - Java Message Service
- 가상 서버 - 가상 서버 사용

로그 수준 지정

ORB, 웹 컨테이너, EJB 컨테이너, MDB 컨테이너(EJB 컨테이너 내부), Java Transaction Service 및 JMS 서비스에 대한 로그 수준을 지정하려면 다음 절차를 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 Sun ONE Application Server 인스턴스를 확장하여 편집하려는 구성 요소와 하위 시스템을 표시합니다.
2. 원하는 구성 요소 또는 하위 시스템의 링크를 누릅니다.
3. 관리 인터페이스의 오른쪽 페이지의 "로그 수준" 드롭다운 목록에서 다음 로그 수준 매개변수 중 하나를 선택합니다. 로그 수준은 97페이지의 그림 "로그 수준 정보"에서 설명합니다.

로그 파일 지정(가상 서버)

로그 파일을 지정하려면 다음 절차를 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 Sun ONE Application Server 인스턴스를 확장하여 HTTP Server 하위 시스템을 표시합니다.
2. HTTP 서버 링크를 누릅니다.
3. 가상 서버 링크를 누릅니다.
4. 원하는 서버 인스턴스 링크를 누릅니다.
5. 관리 인터페이스의 오른쪽 페이지의 "일반" 탭에서 "로그 파일" 필드에 원하는 디렉토리 경로와 파일 이름을 입력합니다.

트랜잭션 로그 위치 지정(Java Transaction Service)

트랜잭션 로그 위치를 지정하려면 다음 절차를 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 Sun ONE Application Server 인스턴스를 확장하여 Transaction Service 하위 시스템을 표시합니다.
2. Transaction Service 링크를 누릅니다.
3. 관리 인터페이스의 오른쪽 페이지에 있는 "고급 필드 그룹"의 "트랜잭션 로그 위치" 필드에 원하는 디렉토리 경로와 파일 이름을 입력합니다.

오류 로깅에 대한 지시문 구성

Sun ONE Application Server는 `init.conf` 파일에 대한 오류 로깅 지시문을 포함합니다. 여기에 포함되는 지시문은 다음과 같습니다.

- **Error Log Date Format.** `ErrorLogDateFormat` 지시문에서는 서버 로그에서 사용하는 날짜 형식을 지정합니다.
- **Log Flush Interval**. `LogFlushInterval`은 액세스 로그가 메모리에서 `access.log` 파일로 전송되기 전의 최대 시간 간격(초)을 결정합니다.
- **Pid Log.** `PidLog`는 기본 서버 프로세서의 프로세스 ID (pid)를 기록할 파일을 지정합니다. 일부 서버 지원 프로그램은 이 로그가 `logs/pid`의 서버 루트에 있음을 전제로 합니다.

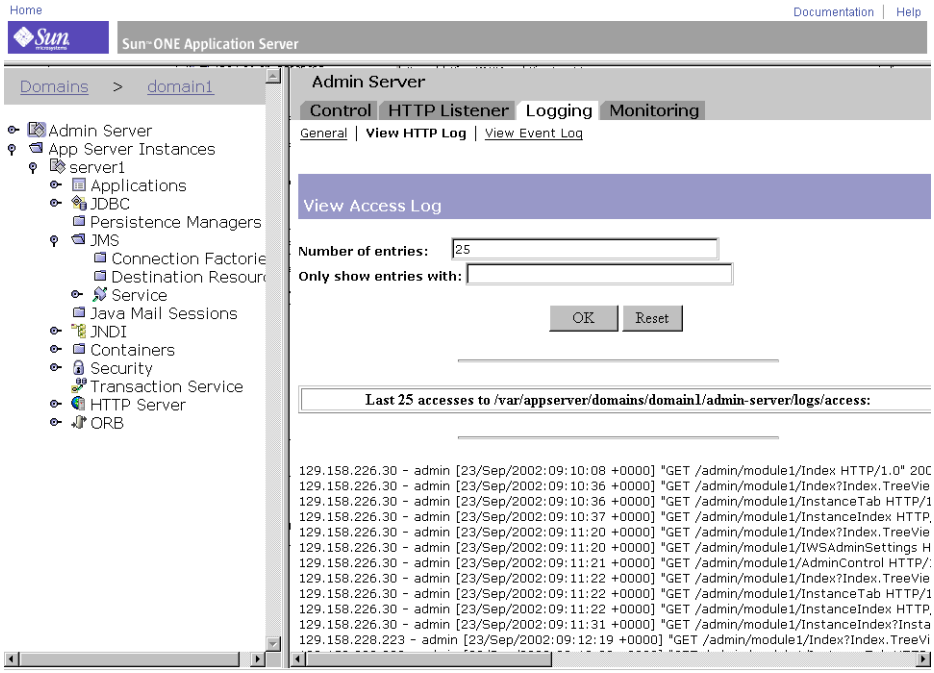
`init.conf`에 대한 모든 지시문에 대한 자세한 내용은 *Sun ONE Application Server Configuration File Reference*를 참조하십시오.

액세스 로그 파일 보기

관리자 서버 및 Sun ONE Application Server 인스턴스의 http 로그 파일을 볼 수 있습니다.

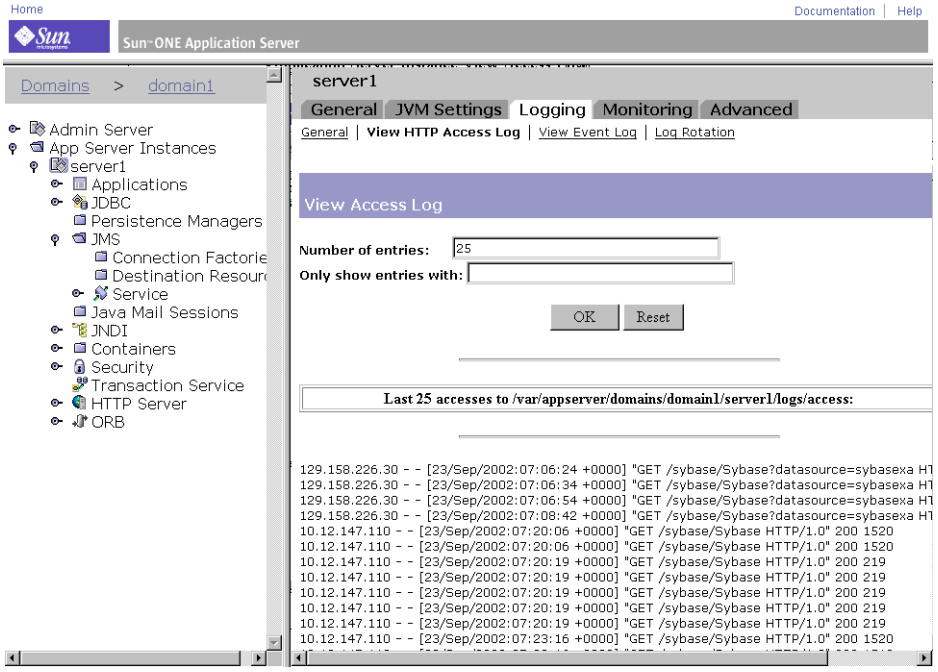
Administration Server의 http 로그를 보려면, 관리 인터페이스의 왼쪽 창에서 "Admin Server"를 선택한 다음, 오른쪽 페이지에서 "로깅" 탭을 선택합니다. "HTTP 액세스 로그 보기" 링크가 나타납니다. 이 링크를 선택하여 구성된 액세스 로그를 봅니다. 이 때 나타나는 로그의 예는 그림 117페이지의 그림 "Admin Server의 HTTP 액세스 로그 보기"와 같습니다.

Admin Server의 HTTP 액세스 로그 보기



응용 프로그램 서버 인스턴스에 대한 액세스 로그를 보려면 관리 인터페이스의 오른쪽 창에서 원하는 서버 인스턴스를 누릅니다. 오른쪽 창에서 "로깅" 탭을 누릅니다. "액세스 로그 보기" 링크를 눌러 해당 서버 인스턴스에 구성된 활성 액세스 로그를 표시합니다. 이에 대한 예는 그림 118페이지의 그림 "응용 프로그램 서버 인스턴스의 액세스 로그 보기"에서 볼 수 있습니다.

응용 프로그램 서버 인스턴스의 액세스 로그 보기

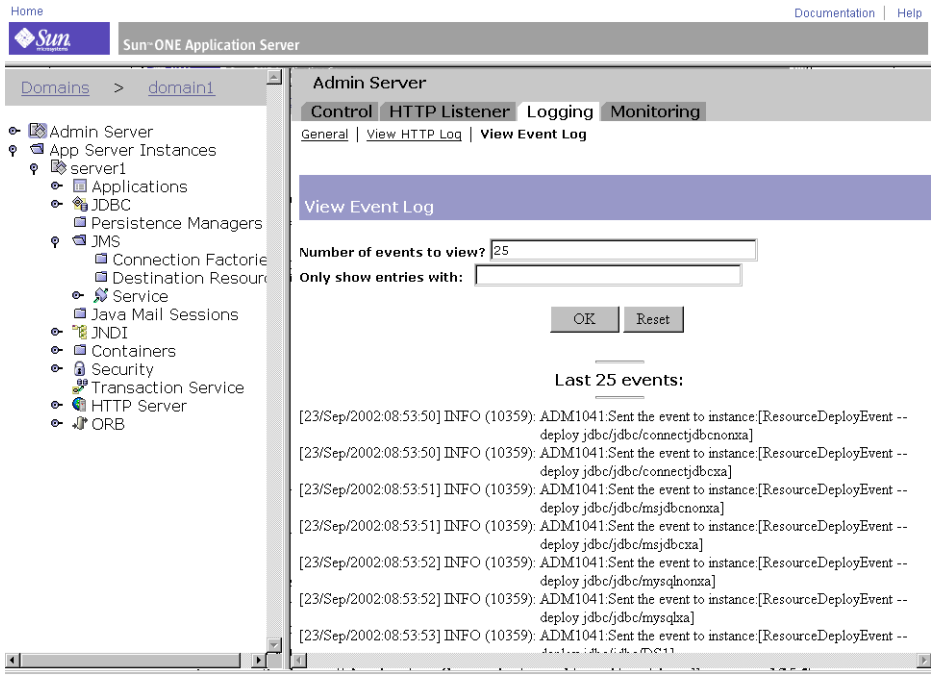


이벤트 로그 파일 보기

관리자 서버 및 Sun ONE Application Server 인스턴스의 활성 이벤트 로그 파일을 모두 볼 수 있습니다.

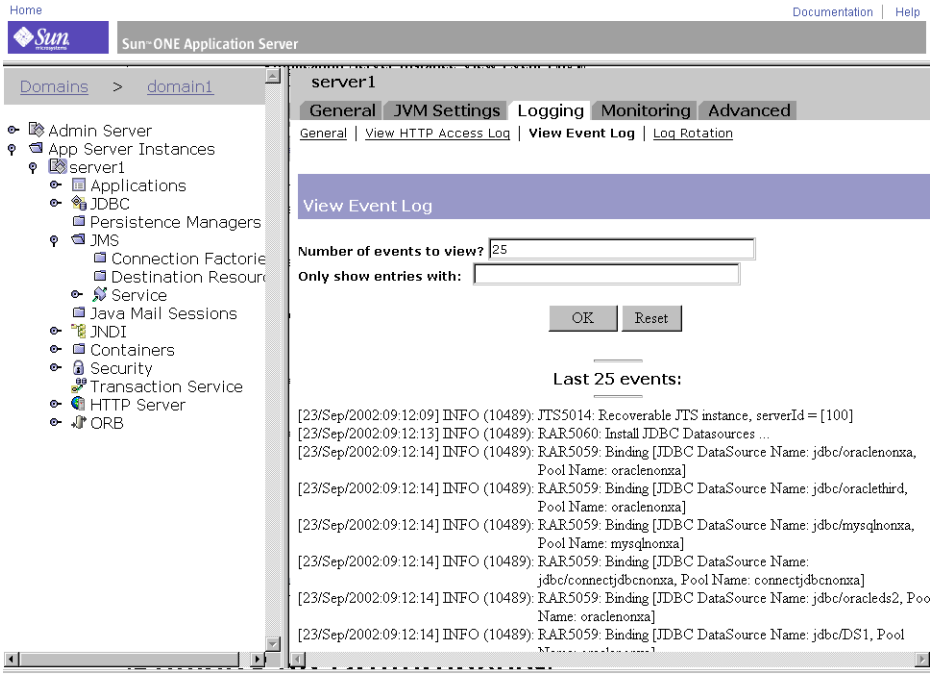
관리자 서버의 이벤트 로그를 보려면, 왼쪽 창에서 "Admin Server"를 선택한 다음 오른쪽 페이지에서 "로깅" 탭을 선택합니다. "이벤트 로그 보기" 링크가 나타납니다. 이 링크를 선택하여 구성된 이벤트 로그를 봅니다. 이 때 나타나는 로그의 예는 그림 119페이지의 그림 "Admin Server의 이벤트 로그 보기"와 같습니다.

Admin Server의 이벤트 로그 보기



응용 프로그램 서버 인스턴스에 대한 이벤트 로그를 보려면, 관리 인터페이스의 왼쪽 창에서 원하는 서버 인스턴스를 누른 다음, 오른쪽 창에서 "로깅" 탭을 선택합니다. 그러면 "이벤트 로그 보기" 링크가 나타납니다. 이 링크를 선택하여 구성된 이벤트 로그를 봅니다. 이 때 나타나는 로그의 예는 120페이지의 그림 "응용 프로그램 서버 인스턴스의 이벤트 로기 보기"와 같습니다.

응용 프로그램 서버 인스턴스의 이벤트 로기 보기



로그 기본 설정

설치 중에 `access`라는 액세스 로그 파일이 서버에 만들어집니다. 액세스 로깅 여부, 로깅에 사용할 형식, 클라이언트가 자원에 액세스할 때 서버에서 클라이언트의 도메인 이름을 찾는 데 시간을 소비할 것인지의 여부를 지정함으로써 자원에 대한 액세스 로깅을 사용자 정의할 수 있습니다.

여러 가상 서버에 하나의 로그 파일을 사용하려면 이벤트 로그에 대한 `server.xml` 파일에서 `LogVsId`를 활성화해야 합니다. 자세한 내용은 *Sun ONE Application Server Configuration File Reference*를 참조하십시오. 또는 관리 인터페이스의 "Admin Server 로깅" 탭에서 `LogVsID`를 활성화할 수 있습니다.

관리 인터페이스에서 "가상 서버 ID 기록"을 활성화하려면 다음 단계를 수행합니다. 변경 사항을 적용하려면 Administration Server를 다시 시작해야 합니다.

1. 관리 인터페이스의 왼쪽 창에서 "Admin Server"를 누릅니다.
2. 오른쪽 페이지에서 "로깅" 탭을 누릅니다.

3. "가상 서버 ID 기록" 확인란을 누릅니다.
4. "저장" 버튼을 눌러 변경 사항을 Sun ONE Application Server에 적용합니다.

이 설정에서는 변경 사항이 적용될 수 있도록 Sun ONE Application Server를 다시 시작해야 합니다.

로그 분석기 실행

`flexanlg`는 로그 파일 보고에 사용되는 로그 분석기 도구입니다. 로그 분석기는 로깅이 `syslog` 이외의 다른 파일에 전송되는 경우에만 사용할 수 있습니다.

로그 분석기를 사용하여 작업 요약, 가장 일반적으로 액세스한 URL, 서버에 대한 액세스 빈도가 가장 높은 하루 중 시간대 등의 기본 서버에 대한 통계를 작성할 수 있습니다. 로그 분석기는 기본 서버가 아닌 가상 서버에 대해서는 통계를 작성할 수 없습니다. 그러나 117페이지의 그림 "액세스 로그 파일 보기"에 설명된 대로 각 가상 서버에 대한 통계는 볼 수 있습니다.

주 로그 분석기를 실행하기 전에 서버 로그를 회전해야 합니다. 자세한 내용은 104페이지의 그림 "로그 파일 관리"를 참조하십시오.

install_dir`/bin/flexanlg` 디렉토리에 있는 도구 `flexanlg`를 실행하여 명령줄에서 로그 분석기 명령을 실행할 수 있습니다.

`flexanlg`를 실행하려면 명령 프롬프트에서 다음 명령과 옵션을 입력하십시오.

```
flexanlg [ -P ] [-n name] [-x] [-r] [-p order] [-i file]* [ -m
metafile ]* [ o file][ c opts] [-t opts] [-l opts] [-h help]
```

명령 옵션(* 표시가 있는 옵션은 반복 가능)

-i *filename*

입력 로그 파일

`-P`

프록시 로그 형식

-n *servername*

서버 이름

`-x`

HTML 출력

-r

호스트 이름에 대한 IP 주소 확인

-p [c, t, l]

출력 순서; 기본 순서는 횟수, 시간 통계, 목록 순

-m *filename*

메타 파일

-o *filename*

출력 로그 파일; 기본값은 stdout

-c [h, n, r, f, e, u, o, k, c, z]

다음 항목을 계산; 기본값: h, n, r, e , u, o, k, c

- h: 총 적중 횟수
- n: 304 Not Modified 상태 코드(로컬 복사 사용)
- r: 302 Found 상태 코드(리디렉션)
- f: 404 Not Found 상태 코드(문서를 찾을 수 없음)
- e: 500 Server Error 상태 코드(잘못된 구성)
- u: 총 고유 URL 수
- o: 총 고유 호스트 수
- k: 전송된 총 KB
- c: 캐시에 저장된 총 KB
- z: 어떤 항목도 계산하지 않음

-t [sx, mx, hx, xx, z]

일반 통계 찾기; 기본값: s5m5h24x10

- s (number): 로그의 최대 (숫자) 초 찾기
- m (number): 로그의 최대 (숫자) 분 찾기
- h (number): 로그의 최대 (숫자) 시간 찾기
- u (number): 로그의 최대 (숫자) 사용자 찾기
- a (number): 로그의 최대 (숫자) 사용자 에이전트 찾기

r (number): 로그의 최대 (숫자) 참조자 찾기

x (number): 기타 키워드에 대한 최대 (숫자) 찾기

z: 어떤 일반 통계도 찾지 않음

-l [cx, hx]

지정된 하위 옵션 목록 만들기; 기본값: c+3h5

c (x, +x): 가장 일반적으로 액세스한 URL

x: x 항목만 나열

+x: x번 이상 액세스 한 경우에만 나열

h (x, +x): 서버에 가장 자주 액세스하는 호스트 또는 IP 주소

x: x 항목만 나열

+x: x번 이상 액세스한 경우에만 나열

z: 어떤 항목도 만들지 않음

예: flexanlg 명령 사용

```
flexanlg -i
/var/opt/SUNQappserver7/domains/domain1/server1/logs/access
```

주 로그 분석기를 실행하기 전에 서버 로그를 아카이브해야 합니다.

이벤트 보기(Windows 2000 Pro)

Sun ONE Application Server는 server.log 파일에 오류를 기록할 뿐만 아니라 이벤트 뷰어에 심각한 시스템 오류를 기록합니다. 이벤트 뷰어는 시스템의 이벤트를 모니터하기 위해 사용합니다. 이러한 이벤트 뷰어를 사용하여 오류 로그가 열리기 전에 발행할 수 있는 기본적인 구성 문제로 인한 오류를 볼 수 있습니다.

이벤트 뷰어를 사용하려면 다음 단계를 수행하십시오.

1. "시작" 메뉴에서 "프로그램"과 "관리 도구"를 차례로 선택합니다. "관리 도구" 프로그램 그룹에서 "이벤트 뷰어"를 선택합니다.
2. "로그" 메뉴에서 "응용 프로그램"을 선택합니다.

응용 프로그램 로그가 이벤트 뷰어에 나타납니다. Sun ONE Application Server의 오류에는 `https-serverid`라는 소스 레이블이 있습니다.

3. 로그에서 이 레이블을 찾으려면 "보기" 메뉴에서 "찾기"를 선택합니다. 업데이트된 로그 항목을 보려면 "보기" 메뉴에서 "갱신"을 선택하십시오.

이벤트 뷰어에 대한 자세한 내용은 시스템 설명서를 참조하십시오.

Sun ONE Application Server 모니터링

이 장에서는 Sun ONE Application Server에서 사용할 수 있는 모니터링 및 SNMP (Simple Network Management Protocol) 기능을 설명합니다.

이 장은 다음 절로 구성되어 있습니다.

- Sun ONE Application Server 모니터링 정보
- CLI를 사용한 모니터링 데이터 추출
- CLI를 사용한 트랜잭션 서비스 관리
- HTTP QOS (Quality of Service) 사용
- SNMP 정보
- SNMP 설정
- SNMP 마스터 에이전트 활성화 및 시작

Sun ONE Application Server 모니터링 정보

사용자는 시스템의 전략적 데이터 포인트에서 작업 통계를 수집하여 Sun ONE Application Server를 모니터할 수 있습니다. 수집한 통계 자료를 통해 서버에서 얼마나 많은 요청을 처리하고 있으며 그러한 요청을 얼마나 잘 처리하고 있는지 알 수 있습니다. 각 가상 서버에 대한 통계와 전체 응용 프로그램 서버 인스턴스에 대한 통계를 볼 수 있습니다. `asadmin` 유틸리티나 SNMP 중 하나를 사용하여 Sun ONE Application Server를 모니터할 수 있습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 통계
- SNMP
- HTTP 서버 모니터링
- 응용 프로그램 구성 요소 및 하위 시스템 모니터링
- QOS (Quality of Service)

통계

HTTP 서버를 비롯한 대부분의 Sun ONE Application Server 응용 프로그램 구성 요소와 하위 시스템에 대해서는 통계 수집이 항상 활성화되어 있기 때문에 별도의 활성화 기능이 필요하지 않습니다. 하지만 특정 하위 시스템에 대한 모니터링이 활성화되거나 관련 기능이 활성화되어 있을 때만 수집되는 통계도 있습니다. 이러한 통계에는 다음과 같은 데이터 포인트가 있습니다.

- EJB 메소드에 대한 통계
- 활성 트랜잭션
- 연결 (QOS(Quality of Service)가 활성화된 경우에만 해당)
- DNS (DNS 캐시가 활성화된 경우만 해당)

127페이지의 "응용 프로그램 구성 요소 및 하위 시스템 모니터링"에 설명된 것처럼 관리 인터페이스에서 응용 프로그램 하위 시스템 또는 구성 요소에 대한 모니터링을 활성화할 수 있습니다.

서버 모니터에서 서버가 많은 양의 요청을 처리 중이라고 보고하는 경우에는 서버 구성이나 시스템의 네트워크 커널을 조정해야 할 수 있습니다. 서버 구성 조정에 대한 자세한 정보는 *Sun ONE Application Server Performance Tuning and Sizing Guide*를 참조하십시오.

SNMP

Sun ONE Application Server는 SNMP (Simple Network Management Protocol)를 사용하여 정보 수집 도구를 통해 네트워크 관리 정보를 제공합니다. SNMP는 네트워크에서 관리 및 모니터링 정보를 교환하는 데 사용하는 프로토콜입니다. SNMP를 사용하여 *에이전트*라는 프로그램으로 네트워크상의 여러 장치(허브, 라우터, 브리지 등)를 모니터합니다. 다른 프로그램에서는 에이전트로부터 데이터를 수집합니다. 모니터링 작업으로 만들어진 데이터베이스를 *관리 정보 베이스*(MIB)라고 합니다. 이 데이터는 네트워크상의 모든 장치가 제대로 작동하는지 확인하는데 사용됩니다.

SNMP를 사용하면 HTTP 서버만 모니터할 수 있지만 명령줄 인터페이스(CLI)를 사용하면 모든 구성 요소와 시스템을 모니터할 수 있습니다.

SNMP에 대한 자세한 내용은 158페이지의 “SNMP 정보” 및 166페이지의 “SNMP 설정”을 참조하십시오.

HTTP 서버 모니터링

HTTP 서버 모니터링은 기본적으로 활성화되어 있으므로 특별히 활성화할 필요가 없습니다. HTTP 서버 모니터링은 XML 파일을 기반으로 이루어지며 모니터할 수 있는 세 가지 속성의 집합으로서 `asadmin` 명령을 사용하여 액세스합니다. 이 XML 파일의 요소, 하위 요소 및 속성은 142페이지의 “모니터할 수 있는 HTTP 서버 요소” 및 143페이지의 “모니터할 수 있는 HTTP 서버 속성”을 참조하십시오.

주 SNMP를 사용하면 HTTP 서버 통계만 수집할 수 있습니다. HTTP 서버를 비롯한 Sun ONE Application Server의 모든 하위 시스템에 대한 통계는 명령줄 인터페이스를 사용하여 수집할 수 있습니다.

`asadmin` 사용에 대한 자세한 내용은 411페이지의 “명령줄 인터페이스 사용”을 참조하십시오.

응용 프로그램 구성 요소 및 하위 시스템 모니터링

Sun ONE Application Server의 일부 하위 시스템 또는 구성 요소는 관련 통계가 항상 수집되기 때문에 모니터링을 활성화할 필요가 없습니다. 예를 들어, 컨테이너와 같은 응용 프로그램 구성 요소에 대한 모니터링은 활성화하거나 비활성화할 수 있습니다. 모니터링을 활성화하면 항상 수집되는 통계 외에 모든 EJB 메소드에 대한 통계가 추가로 수집됩니다. JDBC 연결 풀에 대한 모니터링은 항상 활성화되어 있습니다. 연결 풀은 처음 액세스할 때 초기화되며 그 후에는 언제든지 관련 통계를 모니터할 수 있습니다.

모니터할 수 있는 전체 데이터 포인트 목록은 137페이지의 "모니터할 수 있는 속성 이름"을 참조하십시오.

관리 인터페이스나 명령줄 인터페이스(CLI)에서 선택한 응용 프로그램 구성 요소와 하위 시스템에 대한 모니터링을 활성화할 수 있습니다. 예를 들어, CLI에서 EJB 컨테이너에 대한 모니터링을 활성화하려면 단말기 창에서 다음 명령을 입력합니다.

```
set server1.ejb-container.monitoringEnabled=true
reconfig server1
```

여기서 `server1`은 인스턴스 이름입니다.

컨테이너 노드 아래에 있는 관리 인터페이스에서 같은 기능에 액세스할 수 있습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 컨테이너 하위 시스템 모니터링
- ORB 서비스 모니터링
- 트랜잭션 서비스 모니터링

컨테이너 하위 시스템 모니터링

EJB 컨테이너의 경우 모니터링이 활성화되어 있으면 모든 Entity Bean, Stateless Session Bean을 위한 메소드와 관련된 통계가 수집됩니다. 수집되는 통계 자료는 다음과 같습니다.

- 총 오류 수
- 총 호출 수
- 총 성공 수
- 실행 시간, 밀리초 단위(메소드 최종 호출 시)

그 외의 컨테이너 하위 시스템에 대한 모든 통계는 항상 수집됩니다. 모니터된 데이터 포인트 중 일부는 다음과 관련된 통계를 포함합니다.

- 풀의 초기, 최소 및 최대 Stateless Bean 수
- 캐시의 최소 및 기본 Stateful Bean 및 Entity Bean 수
- 캐시의 최소 및 기본 Stateless Session Bean 수
- 작성 및 삭제된 Bean의 수
- 기타 관련 통계

ORB 서비스 모니터링

ORB 서비스의 경우 모니터된 데이터 포인트에는 ORB 연결 및 ORB 스레드 풀에 대해 수집된 통계가 포함됩니다. ORB 통계는 항상 수집되므로 ORB 서비스에 대한 모니터링을 활성화할 필요가 없습니다.

트랜잭션 서비스 모니터링

JTS (Java Transaction Service) 서비스의 경우 모니터된 데이터 포인트에는 다음 통계가 포함됩니다.

- 완료된 트랜잭션 총 수
- 롤백된 트랜잭션 총 수
- inflight 트랜잭션 총 수
- inflight 트랜잭션 목록

자세한 내용은151페이지의 “CLI를 사용한 트랜잭션 서비스 관리”를 참조하십시오.

QOS (Quality of Service)

*QOS (Quality of Service)*는 서버 인스턴스 가상 서버 클래스 또는 가상 서버에 대해 설정한 성능 제한입니다. 예를 들어, 인터넷 서비스 공급자(ISP)는 제공되는 대역폭에 따라 가상 서버 요금을 다르게 책정할 수 있습니다. 대역폭 용량 및 연결 횟수의 두 영역을 제한할 수 있습니다.

Sun ONE Application Server에서 제공하는 QOS (Quality of Service) 정보는 다음 항목과 관련하여 런타임 동안 서버 효율성을 결정하는 데 사용됩니다.

- 시작 시간
- 서버 트래픽 및 대역폭에 미치는 트래픽의 영향
- 라이브 데이터 대 정적 데이터 분석
- 기타 데이터 요소

자세한 내용은 151페이지의 “CLI를 사용한 트랜잭션 서비스 관리”를 참조하십시오.

CLI를 사용한 모니터링 데이터 추출

명령줄 인터페이스(CLI)에서 `asadmin` 명령을 `list` 및 `get` 명령과 함께 사용하여 모니터된 데이터를 추출할 수 있습니다.

주 `set` 명령은 151페이지의 "CLI를 사용한 트랜잭션 서비스 관리"에 설명된 대로 트랜잭션 서비스에 대한 모니터링을 설정하는 데만 사용됩니다.

이 절에서는 다음 항목에 대해 설명합니다.

- list --monitor 명령
- get --monitor 명령
- CLI 이름 매핑
- 모니터할 수 있는 HTTP 서버 객체

list --monitor 명령

`list` 명령은 지정된 서버 인스턴스 이름에 대해 현재 모니터되고 있는 응용 프로그램 구성 요소 및 하위 시스템에 대한 정보를 제공합니다. 이 명령을 사용하여 서버 인스턴스에 대해 모니터할 수 있는 구성 요소와 하위 구성 요소를 볼 수 있습니다.

예

```
asadmin> list --monitor server1
```

위의 명령은 다음과 같이 모니터링이 활성화되어 있는 응용 프로그램 구성 요소 및 하위 시스템 목록을 반환합니다.

```
iiop-service
transaction-service
application.converter
application.myApp
http-server
```

지정된 서버 인스턴스에서 현재 모니터되는 응용 프로그램을 나열할 수도 있습니다. 이 목록은 응용 프로그램에서 `get` 명령을 사용하여 특정 모니터링 통계를 얻으려고 할 때 유용할 수 있습니다.

예

```
asadmin> list --monitor server1.application
```

위 명령은 다음을 반환합니다.

```
converter
myApp
```

보다 많은 예는 133페이지의 "Petstore 예"를 참조하십시오.

get --monitor 명령

이 명령은 다음과 같은 모니터된 정보를 검색합니다.

- 구성 요소 또는 하위 시스템 내에서 모니터된 모든 속성
- 구성 요소 또는 하위 시스템 내에서 모니터된 특정 속성

특정 구성 요소나 하위 시스템에 존재하지 않는 속성을 요청한 경우에는 오류가 반환됩니다. 마찬가지로 구성 요소나 하위 시스템에 대해 활성화되어 있지 않은 특정 속성을 요청한 경우에도 오류가 반환됩니다.

`get` 명령 사용에 대한 자세한 내용은 132페이지의 "CLI 이름 매핑"을 참조하십시오.

예 1

하위 시스템에서 특정 속성에 대한 모든 속성을 검색하려는 경우

```
asadmin> get --monitor server1.iiop-service.orb.system.orb-connection.*
total-inbound-connections=1
total-outbound-connections=1
```

예 2

J2EE 응용 프로그램에서 모든 속성을 검색하려는 경우

```
asadmin> get --monitor server1.application.converter.*
Attribute name(s) not found.
```

J2EE 응용 프로그램 수준에 모니터할 수 있는 속성이 없기 때문에 이 명령은 실패합니다.

예 3

하위 시스템에서 특정 속성을 검색하려는 경우

```
asadmin> get --monitor server1.transaction-service.inflight-tx
Attribute name = inflight-tx Value = No active transaction found.
```

예 4

하위 시스템 속성 내에서 알 수 없는 속성을 가져오려는 경우

```
asadmin> get --monitor server1.iiop-service.orb.system.orb-connection.bad-name
            Could not get the attribute

Execution failed for the command: get --monitor
server1.iiop-service.orb-connection.bad-name
```

CLI 이름 매핑

Sun ONE Application Server는 트리 구조를 사용하여 모니터할 수 있는 객체를 추적합니다. 트리의 모든 노드에는 이름과 유형이 있습니다. 싱글톤 유형인 경우 모든 부모 노드 아래에는 해당 유형의 단일 노드만 존재합니다. 이 트리의 노드 유형에 대한 자세한 내용은 135페이지의 "모니터할 수 있는 객체 유형"을 참조하십시오.

트리의 루트 객체는 Sun ONE Application Server 인스턴스 이름으로 표시됩니다. 예를 들어 `server1`이라는 이름의 인스턴스에 대한 루트 모니터링 객체는 다음 이름을 사용합니다.

```
server1
```

모든 자식 객체는 점(.) 문자를 분리자로 사용하여 표시됩니다. 자식 노드가 싱글톤 유형인 경우에는 그 객체를 나타내는 데 모니터링 객체 유형만 필요하고 그렇지 않은 경우에는 `type.name` 형태의 이름이 필요합니다.

예를 들어, `http-server`는 유효한 모니터링 가능 객체 유형 중 하나이며 싱글톤입니다. `server1` 인스턴스의 `http-server`를 나타내는 단독 자식 노드를 표시하려면 다음 이름을 사용합니다.

```
server1.http-server
```

다른 예를 들어보면, `application`은 유효한 모니터링 가능 객체 유형이며 싱글톤이 아닙니다. Petstore 응용 프로그램을 나타내는데 싱글톤이 아닌 자식 노드를 표시하려면 다음 이름을 사용합니다.

```
server1.application.petstore
```

CLI 이름을 사용하여 모니터할 수 있는 객체의 특정 속성을 나타낼 수 있습니다. 예를 들어, `http-server`는 모니터할 수 있는 `summary`라는 속성을 가집니다. 다음 이름은 summary 속성을 나타냅니다.

```
server1.http-server.summary
```

모니터링 객체에 의해 노출된 속성 이름을 지정할 때 고정된 규칙이 있는 것은 아닙니다.

사용자는 CLI 사용에 유효한 이름을 알지 못합니다. `list` 명령을 사용하여 모니터할 수 있는 객체를 검사하는 한편 `get` 명령을 와일드카드 매개 변수와 함께 사용하여 모니터할 수 있는 객체의 모든 사용 가능한 속성을 검사할 수 있습니다.

다음 예는 몇 가지 클라이언트 이름 매핑 시나리오를 보여줍니다.

Petstore 예

사용자가 `server1`이라는 이름의 Sun ONE Application Server 인스턴스에 배포된 Petstore 응용 프로그램에서 메소드가 호출된 횟수를 검사하려고 합니다. `list` 명령과 `get` 명령을 조합하여 메소드에 대한 원하는 통계에 액세스합니다.

1. CLI를 다중 모드로 호출합니다.
2. 다음과 같이 몇 가지 유용한 환경 변수를 설정하여 모든 명령에 일일이 입력하지 않도록 합니다.

   ```
   asadmin>export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
   ```

   ```
   asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4848
   ```

3. `server1` 인스턴스에 대해 모니터할 수 있는 구성 요소를 나열합니다.

   ```
   asadmin>list --monitor server1
   ```

   위 명령의 출력 결과는 다음과 같습니다.

   ```
   iiop-service
   transaction-service
   application.CometEJB
   application.ConverterApp
   application.petstore
   http-server
   resources
   ```

   모니터할 수 있는 구성 요소 목록에는 `iiop-service`, `http-server`, `transaction-service`, `resources` 및 배포 및 활성화된 모든 응용 프로그램이 포함됩니다.

4. Petstore 응용 프로그램의 모니터할 수 있는 하위 구성 요소를 나열합니다(`--monitor` 대신 `-m`을 사용할 수 있습니다).

   ```
   asadmin>list -m server1.application.petstore
   ```

   출력 결과는 다음과 같습니다.

   ```
   ejb-module.signon-ejb_jar
   ejb-module.catalog-ejb_jar
   ejb-module.uidgen-ejb_jar
   ```

```
ejb-module.customer-ejb_jar
ejb-module.petstore-ejb_jar
ejb-module.AsyncSenderJAR_jar
ejb-module.cart-ejb_jar
```

5. Petstore 응용 프로그램의 `sigon-ejb_jar` EJB 모듈에 있는 모니터할 수 있는 하위 구성 요소를 나열합니다.

```
asadmin>list -m server1.application.petstore.ejb-module.signon-ejb_jar
```

출력 결과는 다음과 같습니다.

```
entity-bean.UserEJB
stateless-session-bean.SignOnEJB
```

6. Petstore 응용 프로그램의 `sigon-ejb_jar` EJB 모듈에 대한 `UserEJB` Entity Bean에서 모니터할 수 있는 하위 구성 요소를 나열합니다.

```
asadmin>list -m
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB
```

출력 결과는 다음과 같습니다.

```
bean-method.create0
bean-method.findByPrimaryKey1
bean-method.remove2
bean-method.getUserName3
bean-method.setPassword4
bean-method.getPassword5
bean-method.matchPassword6
bean-method.remove7
bean-method.isIdentical8
bean-method.getEJBLocalHome9
bean-method.getPrimaryKey10
bean-pool
bean-cache
```

7. Petstore 응용 프로그램의 `sigon-ejb_jar` EJB 모듈에 있는 `UserEJB` Entity Bean에 대한 `getUserName3` 메소드에서 모니터할 수 있는 하위 구성 요소를 나열합니다.

```
asadmin>list -m
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB.bean-m
ethod.getUserName3
```

출력 결과는 다음과 같습니다.

```
No monitorable entities for element
server1.application.petstore.ejb-module.signon-ejb_jar.entity-bean.UserEJB.bean-m
ethod.getUserName3
```

8. 메소드에 대해 모니터할 수 있는 하위 구성 요소가 없습니다. `getUserName3` 메소드에 대해 모니터할 수 있는 모든 통계를 가져옵니다.

```
asadmin>get -m server1.application.petstore.ejb-module.
signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3.*
method-name = public abstract java.lang.String
com.sun.j2ee.blueprints.signon.user.ejb.UserLocal.getUserName()
total-num-errors = 0
total-num-success = 2
execution-time-millis = 1
total-num-calls = 2
```

9. 실행 시간과 같은 특정 통계도 가져올 수 있습니다.

```
asadmin>get -m server1.application.petstore.ejb-module.
signon-ejb_jar.entity-bean.UserEJB.bean-method.getUserName3.execution-time-millis
execution-time-millis = 1
```

모니터할 수 있는 객체 유형

모니터링에 사용되는 객체 트리는 여러 개의 노드를 포함하고 있습니다. 노드는 객체 트리의 특정 항목으로 고유한 유형, 이름 및 부모 노드에 의해 식별됩니다. 일부 노드 유형은 하나의 부모 노드 아래에 해당 유형의 노드가 하나만 존재할 수 있는 싱글톤입니다. 싱글톤 노드에는 상응하는 이름이 없습니다.

싱글톤이 아닌 유형 노드에는 이름이 필요합니다. 인스턴스 이름 열에서는 가능한 이름 공간에 대해 설명합니다.

다음 표에서는 여러 노드 유형 사이에 사용 가능한 부모-자식 관계 및 일부 노드 유형의 이름 공간에 관한 트리 구조를 설명합니다.

모니터링 객체 유형

노드 유형	싱글톤?	잎?	자식 노드 유형	인스턴스 이름
root	예	아니요	http-server iiop-service resources transaction-service application standalone-ejb-module	
http-server	예	아니요	virtual-server process	
virtual-server	예	예		
process	예	예		

모니터링 객체 유형*(계속)*

노드 유형	싱글톤?	잎?	자식 노드 유형	인스턴스 이름
iiop-service	예	예	orb	
orb	아니요	아니요	orb-connection orb-thread-pool	system은 시스템 ORB에 예약되어 있습니다. 모든 사용자 ORB는 TCP 종점에서 파생된 이름을 얻습니다.
orb-connection	예	예		
orb-thread-pool	예	예		
resources	예	아니요	jdbc-connection-pool	
jdbc-connection-pool	아니요	예		해당 이름은 연결 풀을 만드는 동안 사용자가 지정한 이름과 같습니다.
transaction-service	예	예		
application	아니요	아니요	ejb-module	server.xml에 등록된 응용 프로그램의 이름
ejb-module	아니요	아니요	stateless-session-bean stateful-session-bean entity-bean message-driven-bean	EJB 모듈의 이름. EJB JAR 이름에서 파생됩니다.
standalone-ejb-module	아니요	아니요	stateless-session-bean stateful-session-bean entity-bean message-driven-bean	server.xml에 등록된 독립 실행형 EJB 모듈의 이름
stateless-session-bean	아니요	아니요	bean-pool bean-method	배포 설명자의 Bean 이름
stateful-session-bean	아니요	아니요	bean-cache bean-method	배포 설명자의 Bean 이름
entity-bean	아니요	아니요	bean-cache bean-pool bean-method	배포 설명자의 Bean 이름

모니터링 객체 유형*(계속)*

노드 유형	싱글톤?	잎?	자식 노드 유형	인스턴스 이름
`message-driven-bean`	아니요	아니요	`bean-pool` `bean-method`	배포 설명자의 Bean 이름
`bean-pool`	예	예		
`bean-cache`	예	예		
`bean-method`	아니요	예		Message-driven Bean의 `onMessage` 메소드 이름 뒤에 다른 Enterprise Bean 메소드의 숫자 접미어를 붙입니다. 접미어는 오버로드된 메소드를 명확히 하기 위해 필요합니다.

모니터할 수 있는 속성 이름

모든 모니터할 수 있는 객체가 모니터할 수 있는 속성을 표시할 필요는 없습니다. 일부 객체는 다른 객체를 그룹화하는 데만 사용됩니다. Sun ONE Application Server의 경우 `http-server` 노드 외에는 트리의 잎 노드만 속성을 가집니다. `http-server` 노드 유형은 자식 노드와 속성을 가집니다. 다음 표는 여러 노드에 대해 모니터할 수 있는 속성 이름의 목록입니다.

http-server

속성 이름	데이터 유형	설명
`summary`	문자열(서식 지정됨)	HTTP server 요약. 가상 서버 및 프로세스 포함. 주: 서식이 지정된 문자열로 표시되는 데이터에 대한 자세한 내용은 142페이지의 "모니터할 수 있는 HTTP 서버 객체"를 참조하십시오.

virtual-server

속성 이름	데이터 유형	설명
<vs-id>	문자열(서식 지정됨)	가상 서버 정보. 모든 응용 프로그램 서버 인스턴스에 대해 한 개 이상의 가상 서버가 있을 수 있습니다. 가상 서버 ID 목록은 http-server의 summary 속성에서 얻을 수 있습니다. server1.http-server.virtual-server.<vs-id> 형태의 get 명령 매개 변수를 사용하여 특정 가상 서버에 대한 통계를 조회할 수 있습니다. server1.http-server.virtual-server.* 형태의 get 명령 매개 변수를 사용하여 모든 가상 서버에 대한 통계를 조회할 수 있습니다. 주: 서식이 지정된 문자열로 표시되는 데이터에 대한 자세한 내용은 142페이지의 "모니터할 수 있는 HTTP 서버 객체"를 참조하십시오.

process

속성 이름	데이터 유형	설명
<pid>	문자열(서식 지정됨)	프로세스 정보. 모든 응용 프로그램 서버 인스턴스에는 하나의 프로세스가 있습니다. 프로세스 ID는 http-server의 summary 속성에서 얻을 수 있습니다. 프로세스에 대한 통계는 get 명령에 server1.http-server.process.<pid> 형태의 매개 변수를 사용하여 얻을 수 있습니다. 주: 서식이 지정된 문자열로 표시되는 데이터에 대한 자세한 내용은 142페이지의 "모니터할 수 있는 HTTP 서버 객체"를 참조하십시오.

orb-connection

속성 이름	데이터 유형	설명
total-inbound-connections	정수	ORB로의 총 인바운드 연결 수
total-outbound-connections	정수	ORB로부터의 총 아웃바운드 연결 수

orb-thread-pool

속성 이름	데이터 유형	설명
thread-pool-size	정수	ORB 스레드 풀의 총 스레드 수
waiting-thread-count	정수	도착할 작업을 대기 중인 스레드 풀의 스레드 수

jdbc-connection-pool

속성 이름	데이터 유형	설명
total-threads-waiting	정수	JDBC 연결 대기 중인 총 스레드 수
total-outbound-connections	정수	총 JDBC 연결 검증 실패 수
total-connections-timed-out	정수	시간 초과된 총 연결 요청 수

transaction-service

속성 이름	데이터 유형	설명
total-tx-completed	정수	완료된 총 트랜잭션 수
total-tx-rolled-back	정수	롤백된 총 트랜잭션 수
total-tx-inflight	정수	총 inflight (활성) 트랜잭션 수
isFrozen	문자열	트랜잭션 시스템 동결 여부(true 또는 false)
inflight-tx	문자열(서식 지정됨)	inflight 트랜잭션 목록

bean-pool

속성 이름	데이터 유형	설명
max-pool-size	정수	풀에 있는 최대 Bean 인스턴스 수
steady-pool-size	정수	일반적으로 Bean 인스턴스 수는 풀에서 관리됩니다. 풀은 처음 만들어질 때 steady-pool-size와 같은 크기로 채워집니다. 풀에서 인스턴스가 삭제되면 비동기적으로 다시 채워지므로 풀 크기는 steady-pool-size와 같거나 그 이상이 됩니다.

bean-pool*(계속)*

속성 이름	데이터 유형	설명
pool-resize-quantity	정수	풀의 크기가 max-pool-size까지 커지거나 steady-pool-size로 줄어드는 증분
idle-timeout-in-seconds	정수	풀 정리 스레드가 실행되는 비율을 정의합니다. 현재 풀 크기가 고정 풀 크기보다 큰 지 확인하고 pool-resize-quantity 요소를 제거합니다. 현재 크기가 steady-pool-size 보다 작으면 pool-resize-quantity만큼 증가됩니다. 증분의 최대 크기는 min (current-pool-size+pool - resize-quantity, max-pool-size)으로 제한됩니다. pool-idle-timeout-in-seconds 보다 긴 시간 동안 액세스되지 않은 객체는 제거 대상이 됩니다.
num-beans-in-pool	정수	풀에서 사용할 수 있는 Bean의 수
num-threads-waiting	정수	사용 가능한 Bean을 기다리는 스레드의 수
total-beans-created	정수	현재까지 만들어진 Bean의 수
total-beans-destroyed	정수	현재까지 삭제된 Bean의 수
jms-max-messages-load	정수	Message-driven Bean을 위해 JMS 세션에 한 번에 로드하는 최대 메시지 수. 기본값은 1이며, Message-driven Bean의 풀에만 적용됩니다.

bean-cache

속성 이름	데이터 유형	설명
cache-resize-quantity (resize-quantity)	정수	캐시의 Bean 수가 max-cache-size와 같을 때(캐시 오버플로 발생 시) 캐시 크기가 축소되는 양
cache-misses	정수	사용자 요청 시 Bean을 캐시에서 찾지 못한 횟수
idle-timeout-in-seconds	정수	캐시 정리 스레드가 예약되는 비율. 이 정리 스레드는 캐시에 있는 모든 Bean을 확인하고 cache-idle-timeout-in-seconds 동안 액세스되지 않은 Bean을 비활성화 처리합니다.
cache-hits	정수	사용자 요청 시 캐시에서 항목을 찾은 횟수

bean-cache*(계속)*

속성 이름	데이터 유형	설명
total-beans-in-cache	정수	캐시에 있는 Bean의 수. 이 값은 캐시의 현재 크기입니다.
max-beans-in-cache	정수	캐시에 저장할 수 있는 Bean의 최대 수로 이 값을 초과하면 오버플로가 발생합니다.
num-passivations	정수	비활성화 처리 횟수. Stateful Session Bean에만 적용됩니다.
num-passivation-errors	정수	비활성화 처리를 수행하는 동안 발생한 오류 횟수. Stateful Session Bean에만 적용됩니다.
num-expired-sessions-removed	정수	정리 스레드로 제거된 만료된 세션의 수. Stateful Session Bean에만 적용됩니다.
num-passivation-success	정수	비활성화 처리가 성공적으로 완료된 횟수. Stateful Session Bean에만 적용됩니다.

bean-method

속성 이름	데이터 유형	설명
method-name	문자열	메소드의 완전 수식 이름
total-num-calls	정수	메소드가 호출된 횟수. 이 통계는 EJB 컨테이너에 대한 모니터링이 활성화된 경우에는 Stateful Session Bean, Stateless Session Bean 및 Entity Bean에 대해, Message-driven Bean 컨테이너에 대한 모니터링이 활성화된 경우에는 Message-driven Bean에 대해 수집됩니다.
total-num-errors	정수	메소드 실행 결과 예외가 발생한 횟수. 이 통계는 EJB 설정 하에 모니터링이 활성화된 경우에는 Stateful Session Bean, Stateless Session Bean, Entity Bean에 대해, MDB 설정 하에 모니터링이 활성화된 경우에는 Message-driven Bean에 대해 수집됩니다.
total-num-success	정수	메소드가 성공적으로 실행된 횟수. 이 통계는 EJB 컨테이너에 대한 모니터링이 활성화된 경우에는 Stateful Session Bean, Stateless Session Bean 및 Entity Bean에 대해, Message-driven Bean 컨테이너에 대한 모니터링이 활성화된 경우에는 Message-driven Bean에 대해 수집됩니다.

bean-method(*계속*)

속성 이름	데이터 유형	설명
`execution-time-millis`	Long	이 메소드가 성공적으로 최종 실행 시 걸린 시간. 이 통계는 EJB 컨테이너에 대한 모니터링이 활성화된 경우에는 Stateful Session Bean, Stateless Session Bean 및 Entity Bean에 대해, Message-driven Bean 컨테이너에 대한 모니터링이 활성화된 경우에는 Message-driven Bean에 대해 수집됩니다.

모니터할 수 있는 HTTP 서버 객체

모니터할 수 있는 HTTP 서버 속성 이름 `summary`는 각 하위 요소의 수와 속성 값을 비롯하여 `Server` 요소의 속성 값과 하위 요소에 대한 요약을 인쇄합니다. HTTP 서버 `virtual-server` 속성은 `VirtualServer` 요소의 속성 값과 각 하위 요소의 세부 사항을 인쇄합니다. `process` 속성은 `Process` 요소의 속성 값과 각 하위 요소의 세부 사항을 인쇄합니다.

NSAPI 성능 프로필을 활성화하고 `Profile` 및 `ProfileBucket` 요소에 대한 통계를 얻으려면 *Sun ONE Application Server Developer's Guide to NSAPI*를 참조하십시오.

성능 조정을 위한 모니터링 통계 사용 방법에 대한 내용은 *Sun ONE Application ServerPerformance and Tuning Guide*를 참조하십시오.

모니터할 수 있는 HTTP 서버 요소

다음 표는 모니터할 수 있는 HTTP 서버 요소 목록입니다.

모니터할 수 있는 HTTP 서버 요소

요소 이름	하위 요소	설명

`Server`	`ConnectionQueue`
	`ThreadPool`
	`Profile`
	`Process`
	`VirtualServer`
		서버 인스턴스

`ConnectionQueue`	없음	요청이 처리되기 전에 대기하는 대기열. Sun ONE Application Server 7에는 한 개의 연결 대기열만 있습니다.
`ThreadPool`	없음	`init.conf` 파일에 정의된 스레드 풀

모니터할 수 있는 HTTP 서버 요소*(계속)*

요소 이름	하위 요소	설명
`Profile`	없음	`init.conf` 파일에 정의된 NSAPI 성능 프로필 버킷

`Process`	`ConnectionQueueBucket`
	`ThreadPoolBucket`
	`DnsBucket`
	`DnsBucket`
	`KeepaliveBucket`
	`CacheBucket`
	`Thread`
		서버 인스턴스 내의 단일 서버 프로세스

`ConnectionQueueBucket`	없음	특정 `ConnectionQueue`와 관련된 통계 추적
`ThreadPoolBucket`	`ThreadPoolBucket`	특정 `ThreadPool`과 관련된 통계 추적
`DnsBucket`	없음	DNS 통계 추적
`KeepaliveBucket`	없음	연결 유지(지속적인 연결) 통계 추적
`CacheBucket`	없음	파일 캐시(NSFC) 통계 추적
`Thread`	`RequestBucket` `ProfileBucket`	요청 처리 스레드 설명
`VirtualServer`	`RequestBucket` `ProfileBucket`	가상 서버 설명
`RequestBucket`	없음	요청 관련 통계 추적
`ProfileBucket`		`Profile` 요소와 관련된 통계 추적

모니터할 수 있는 HTTP 서버 속성

다음 표는 모니터할 수 있는 HTTP 속성 목록입니다.

서버

속성 이름	값	설명
`Id`		서버 인스턴스 ID (예: `server1`)
`VersionServer`		Sun ONE Application Server 버전을 포함하는 문자열
`TimeStarted`	GMT	이 서버 인스턴스가 시작된 시간

서버*(계속)*

속성 이름	값	설명
SecondsRunning		이 서버 인스턴스가 시작된 후 경과된 시간(초)
TicksPerSecond		초당 눈금 수. 이 값은 시스템에 따라 다릅니다.
MaxProcs		최대 프로세스 수
MaxThreads		최대 처리 스레드 수
MaxVirtualServers		추적된 최대 가상 서버 수
FlagProfilingEnabled	0 (off), 1 (on)	NSAPI 성능 프로필의 활성화(on) 여부를 나타냅니다.
FlagVirtualServerOverflow	0 (no), 1 (yes)	MaxVirtualServers 이상의 것이 구성되어 있는지(yes) 여부를 나타냅니다. 이 속성이 1로 설정되면 모든 가상 서버에 대한 통계가 추적되지 않습니다.
LoadMinuteAverage		1분 동안의 평균 로드
Load5MinuteAverage		5분 동안의 평균 로드
Load15MinuteAverage		15분 동안의 평균 로드
RateBytesTransmitted	초당 바이트	서버에 정의된 시간 간격을 두고 데이터가 전송되는 속도 또는 이 정보가 없는 경우에는 0
RateBytesReceived	초당 바이트	서버에 정의된 시간 간격을 두고 데이터가 수신되는 속도 또는 이 정보가 없는 경우에는 0

ConnectionQueue

속성 이름	값	설명
Id		연결 대기열 ID

ThreadPool

속성 이름	값	설명
Id		스레드 풀 ID

ThreadPool*(계속)*

속성 이름	값	설명
Name		스레드 풀의 심볼릭 이름

Profile

속성 이름	값	설명
Id		NSAPI 성능 프로필 버킷 ID
Name		NSAPI 성능 프로필 버킷의 심볼릭 이름
Description		NSAPI 성능 프로필 버킷에 대한 설명

Process

속성 이름	값	설명
Pid		이 프로세스를 고유하게 식별하는 운영 체제 프로세스 ID
Mode	알 수 없음 활성	이 프로세스가 활성화되어 있을 때는 활성이라고 표시됩니다.
TimeStarted	GMT	이 프로세스가 시작된 시간
CountConfigurations		하나의 구성이 로드된 횟수 또는 이 정보가 없는 경우에는 0
SizeVirtual	킬로바이트 (KB)	이 프로세스가 사용하는 가상 메모리의 크기
SizeResident	킬로바이트 (KB)	이 프로세스가 사용하는 상주 메모리의 크기
FractionSystemMemoryUsage		이 프로세스가 사용하는 시스템 메모리의 비율

ConnectionQueueBucket

속성 이름	값	설명
ConnectionQueue		ConnectionQueue 요소의 ID
CountTotalConnection		승인된 새로운 연결의 총 수
CountQueued		현재 대기열에 있는 연결의 수
PeakQueued		대기열에 동시에 존재한 최대 연결 수
MaxQueued		대기열에 있을 수 있는 최대 연결 수
CountOverflow		대기열이 가득 차서 연결을 수용할 수 없었던 횟수
CountTotalQueued		대기열에 있던 연결의 총 수 특정 연결이 여러 번 대기열에 있을 수 있으므로 CountTotalQueued 값은 CountTotalConnections보다 크거나 같을 수 있습니다.
TicksTotalQueued		연결이 대기열에서 보낸 시간의 총 눈금 수. 눈금은 시스템 종속 시간 단위입니다. 참조:TicksPerSecond

ThreadPoolBucket

속성 이름	값	설명
Thread-pool		ThreadPool 요소의 ID
CountThreadsIdle		현재 유휴 상태인 요청 처리 스레드의 수
CountThreads		요청 처리 스레드의 수
MaxThreads		동시에 존재할 수 있는 요청 처리 스레드의 최대 수
CountQueued		이 스레드 풀로 처리하기 위해 대기열에 있는 요청 수

ThreadPoolBucket*(계속)*

속성 이름	값	설명
PeakQueued		대기열에 동시에 존재한 최대 요청 수
MaxQueued		대기열에 있을 수 있는 최대 요청 수

DnsBucket

속성 이름	값	설명
FlagCacheEnabled	0 (off), 1 (on)	DNS 캐시의 활성화(on) 여부를 나타냅니다.
CountCacheEntries		현재 캐시에 있는 DNS 항목의 수
MaxCacheEntries		캐시에 수용할 수 있는 최대 DNS 항목 수
CountCacheHits		DNS 캐시 조회가 성공한 횟수
CountCacheMisses		DNS 캐시 조회가 실패한 횟수
FlagAsyncEnabled	0 (off), 1 (on)	비동기 DNS 조회 활성화(on) 여부를 나타냅니다.
CountAsyncNameLookups		수행된 비동기 DNS 이름 조회 총 수
CountAsyncAddrLookups		수행된 비동기 DNS 주소 조회 총 수
CountAsyncLookupsInProgress		현재 처리 중인 비동기 DNS 조회 총 수

KeepaliveBucket

속성 이름	값	설명
CountConnections		현재 연결 유지 모드에 있는 연결 수
MaxConnections		최대 동시 연결 유지 연결 수

KeepaliveBucket*(계속)*

속성 이름	값	설명
CountHits		후속적으로 유효한 요청이 이루어진 연결 유지 모드의 총 연결 횟수
CountFlushes		서버가 연결 유지 연결을 닫은 횟수
CountTimeouts		연결 유지 연결의 시간이 초과된 횟수
SecondsTimeouts		서버가 유휴 연결 유지 연결을 닫기 전의 시간(초)
CountRefusals		서버가 연결 유지 연결을 거부한 횟수

CacheBucket

속성 이름	값	설명
FlagEnabled	0 (off), 1 (on)	파일 캐시 활성화(on) 여부를 나타냅니다.
SecondsMaxAge	시간(초)	파일 캐시 항목의 최대 사용 기간
CountEntries		현재 파일 캐시에 있는 항목 수
MaxEntries		파일 캐시에서 동시에 수용할 수 있는 최대 캐시 항목 수
CountOpenEntries		열려 있는 파일과 관련된 항목 수
MaxOpenEntries		파일 캐시에서 동시에 수용할 수 있는 열린 파일과 관련된 최대 캐시 항목 수
SizeHeapCache	바이트 수	캐시된 파일 내용이 사용한 힙의 양
MaxHeapCacheSize	바이트 수	파일 캐시에서 캐시된 파일 내용에 사용하는 최대 힙의 양
SizeMmapCache	바이트 수	메모리 매핑된 파일 내용이 사용하는 주소 공간의 양
MaxMmapCacheSize	바이트 수	파일 캐시에서 메모리 매핑된 파일 내용에 사용하는 최대 주소 공간 양
CountHits		캐시 항목 조회가 성공한 횟수

CacheBucket*(계속)*

속성 이름	값	설명
CountMisses		캐시 항목 조회가 실패한 횟수
CountInfoHits		파일 정보 조회가 성공한 횟수
CountInfoMisses		파일 정보 조회가 실패한 횟수
CountContentHits		내용 조회가 성공한 횟수
CountContentMisses		내용 조회가 실패한 횟수

Thread

속성 이름	값	설명
Mode	알 수 없음, 유휴, DNS, 요청, 처리, 응답, 업데이트 중	마지막으로 사용한 스레드의 상태
TimeStarted	GMT	이 스레드가 시작된 시간
ConnectionQueue		스레드에서 서비스하고 있는 ConnectionQueue의 ID

VirtualServer

속성 이름	값	설명
Id		가상 서버 ID
Mode	알 수 없음, 활성	이 가상 서버가 활성화 상태이면 활성이라고 표시됩니다.
Hosts		이 가상 서버가 서비스하는 소프트웨어 가상 서버 호스트 이름 (예: www.foo.com foo.com foo.isp.com).
Interfaces		가상 서버가 구성된 인터페이스(Listener) (예: 192.168.1.2:80 192.168.1.2:443)

RequestBucket

속성 이름	값	설명
CountRequests		서비스된 요청 수
CountBytesReceived		수신된 바이트 수 또는 이 정보가 없는 경우에는 0
CountBytesTransmitted		전송된 바이트 수 또는 이 정보가 없는 경우에는 0
RateBytesTransmitted	초당 바이트	서버에서 정의한 시간 간격 동안 데이터가 전송된 속도 또는 이 정보가 없는 경우에는 0
MaxByteTransmissionRate		사용자가 정의한 시간 간격 동안 데이터가 전송된 최대 속도 또는 이 정보가 없는 경우에는 0
CountOpenConnections		열려 있는 연결의 수 또는 이 정보가 없는 경우에는 0
MaxOpenConnections		열려 있는 연결의 최대 수 또는 이 정보가 없는 경우에는 0
Count2xx		전송된 200 수준 응답 수
Count3xx		전송된 300 수준 응답 수
Count4xx		전송된 400 수준 응답 수
Count5xx		전송된 500 수준 응답 수
CountOther		200, 300, 400 또는 500 수준 이외에 전송된 응답의 수
Count200		전송된 200 수준 응답 수
Count302		전송된 302 수준 응답 수
Count304		전송된 304 수준 응답 수
Count400		전송된 400 수준 응답 수
Count401		전송된 401 수준 응답 수
Count403		전송된 403 수준 응답 수
Count404		전송된 404 수준 응답 수
Count503		전송된 503 수준 응답 수

ProfileBucket

속성 이름	값	설명
Profile		Profile 요소의 ID
Countcalls		NSAPI SAF에 대한 호출 수
CountRequests		처리된 요청 수
TicksDispatch		요청을 디스패치하는 데 사용된 눈금 수. 눈금은 시스템 종속 시간 단위입니다. 참조: TicksPerSecond
TicksFunction		NSAPI SAF에 사용된 눈금 수. 눈금은 시스템 종속 시간 단위입니다. 참조: TicksPerSecond

CLI를 사용한 트랜잭션 서비스 관리

set 명령을 사용하여 JTS에 대해 모니터하려는 통계를 관리할 수 있습니다.

예 1

롤백 목록에 트랜잭션을 추가하려면(그 결과 롤백 또는 특정 트랜잭션 발생) 다음과 같이 set 명령을 사용합니다.

```
set --monitor server1.transaction-service.rollback-list=txnid1
```

예 2

트랜잭션 서비스를 동결하려면 다음과 같이 set 명령을 사용합니다.

```
set --monitor server1.transaction-service.freeze=true
```

다음 표에서는 JTS에 대한 통계를 수집하기 위해 모니터할 수 있는 속성을 설명합니다. 이 속성들은 132페이지의 “CLI 이름 매핑”에 설명된 규칙에 따라 명령줄에서 설정할 수 있습니다.

Java Transaction Service에 대한 자세한 내용은 9장, "트랜잭션 서비스 사용"을 참조하십시오.

HTTP QOS (Quality of Service) 사용

다음 설정에 따라 트래픽 측정 방법과 대역폭 재계산 빈도가 결정됩니다.

- 재계산 간격—대역폭 계산 빈도(밀리초)를 나타냅니다.
- 메트릭 간격—트래픽 계산에 데이터가 사용되는 시간을 나타냅니다.

관리 인터페이스에서 서버 인스턴스 또는 가상 서버 클래스에 대한 이러한 서버 또는 클래스 수준 설정을 활성화할 수 있습니다. 그러나 개별 가상 서버에 대한 설정을 대체할 수 있습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- QOS (Quality of Service)의 예
- QOS (Quality of Service)의 구성
- obj.conf 파일에 필요한 변경 사항
- QOS (Quality of Service)에 사용되는 제한

QOS (Quality of Service)의 예

다음은 QOS (Quality of Service) 정보가 수집되고 계산되는 방법에 대한 예입니다.

- 서버의 메트릭 간격은 30초입니다.
- 서버는 0초에 시작합니다.
- 1초에는 HTTP 연결에서 서버에서 나가고 서버로 들어오는 5000바이트의 트래픽을 생성합니다.
- 그 후에는 더 이상 연결이 이루어지지 않습니다. 30초에는 지난 30초 동안의 총 트래픽이 5000바이트입니다.
- 32초에는 1초에서의 트래픽 샘플이 메트릭 간격인 30초보다 오래 되었으므로 취소됩니다. 지난 30초 간의 총 트래픽은 이제 0입니다.

재계산 간격도 이와 비슷하게 작동합니다. 서버의 재계산 간격은 100밀리초입니다.

같은 예를 가지고 계속 설명하자면 대역폭은 매 100밀리초마다 정기적으로 재계산됩니다. 이 계산은 메트릭 간격과 트래픽 양을 기반으로 이루어집니다.

- 0초에 대역폭이 처음으로 계산됩니다. 총 트래픽은 0이며 이것을 30초의 메트릭 간격으로 나누면 대역폭은 0이 됩니다.
- 1초에 1/10 시간(1000밀리초/100밀리초)에 대한 대역폭이 계산됩니다. 총 트래픽은 5000바이트이고 이것을 30초로 나누면 대역폭은 5000/30 = 166바이트/초 입니다.
- 30초에 1/300 시간에 대한 대역폭이 계산됩니다. 총 트래픽 5000바이트를 30초로 나누면 대역폭은 5000/30 = 166바이트/초 입니다.
- 32초에 1/320 시간에 대해 대역폭이 다시 계산됩니다. 트래픽을 생성했던 연결이 오래되어 취소되었으므로 이 때의 트래픽은 0이며 이를 30으로 나누면 대역폭은 0바이트/초가 됩니다.

QOS (Quality of Service)의 구성

서버 인스턴스 또는 가상 서버의 클래스를 위한 QOS (Quality of Service)는 관리 인터페이스를 통해 구성됩니다.

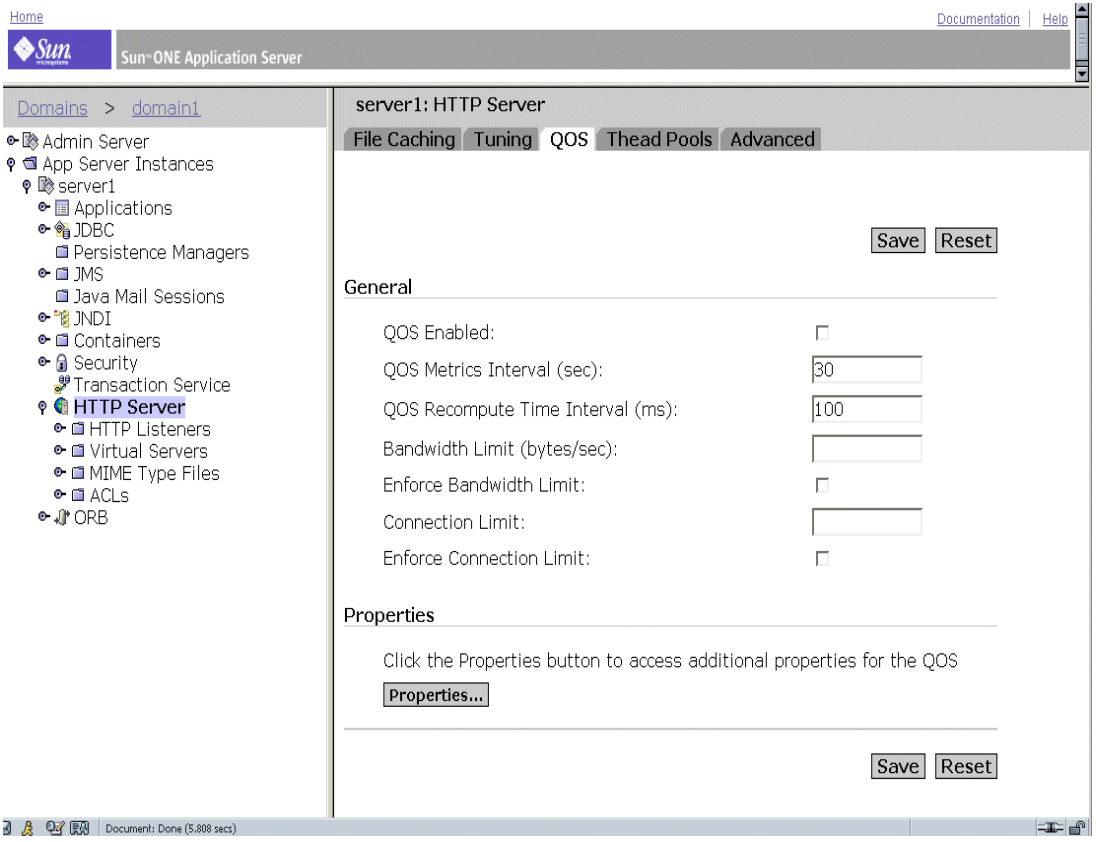
주 QOS (Quality of Service) 설정을 적용하려면 156페이지의 “obj.conf 파일에 필요한 변경 사항”에 설명된 대로 `obj.conf` 파일에 SAF (Server Application Function)를 설정해야 합니다.

QOS (Quality of Service)를 구성하려면 다음 단계를 수행합니다.

1. 왼쪽 창에서 "응용 프로그램 서버 인스턴스" 노드를 선택합니다.
2. 서버 인스턴스 노드를 확장하여 "HTTP 서버" 노드를 표시합니다.
3. "HTTP 서버" 노드를 눌러 "QOS" 탭을 표시합니다.
4. "QOS" 탭을 누릅니다.

   다음 페이지에는 QOS (Quality of Service)에 대한 일반 설정이 등록 정보 버튼과 함께 표시되어 있습니다.

가상 서버 인스턴스 QOS 탭



5. 이 HTTP 서버에 QOS (Quality of Service)를 사용하려면 "QOS 사용"을 누릅니다.

   주: 기본적으로 "QOS 사용"은 비활성화되어 있습니다. "QOS 사용"을 활성화하면 서버 오버헤드가 약간 증가합니다.

6. "QOS 메트릭 간격"을 지정합니다.

   메트릭 간격은 서버 트래픽을 계산하는 동안의 데이터 샘플 선택 시간을 초 단위로 표시한 것입니다. 기본값은 30초입니다.

대형 파일 전송이 많은 사이트인 경우에는 이 필드에 큰 값(몇 분 이상)을 사용하십시오. 대형 파일 전송에는 짧은 메트릭 간격 동안 허용된 대역폭이 모두 사용되므로 최대 대역폭 설정을 적용하면 연결이 거부됩니다. 대역폭은 메트릭 간격으로 평균화되므로 간격이 길면 대형 파일로 생긴 급격한 상승이 완화됩니다.

대역폭 제한이 사용할 수 있는 대역폭보다 훨씬 낮으면(예: 대역폭 제한은 1MB/초이지만 백본에 대한 연결은 1GB/초인 경우) 메트릭 간격을 단축해야 합니다.

주: 대형 정적 파일이 전송되는 경우나 대역폭 제한이 사용할 수 있는 대역폭보다 훨씬 낮은 경우에는 서로 반대되는 해결책이 요구되기 때문에 어떤 상황에 맞춰 조정할 것인지 결정해야 합니다.

7. "QOS 재계산 시간 간격"을 지정합니다.

재계산 시간 간격은 모든 서버, 클래스 및 가상 서버의 대역폭 계산 간의 시간(밀리초)입니다. 기본값은 100밀리초입니다.

8. "대역폭 제한"을 지정합니다.

이 값은 서버 인스턴스에 대한 최대 대역폭 제한을 초당 바이트로 나타낸 것으로 "QOS 메트릭 간격"과 어느 정도 상호 의존성을 가집니다.

9. 최대 대역폭 설정을 적용할 것인지 여부를 선택합니다.

최대 대역폭을 적용하기로 선택하는 경우에는 서버가 대역폭 제한에 도달할 때 추가 연결이 거부됩니다.

최대 대역폭을 적용하지 않기로 선택하는 경우에는 최대값이 초과될 때 서버가 이벤트 로그에 메시지를 기록합니다.

10. "연결 제한"을 지정합니다.

이 값은 동시 요청 처리의 수입니다.

11. 연결 제한 설정 적용 여부를 선택합니다.

최대 연결을 적용하기로 선택하는 경우에는 서버가 그 제한에 도달할 때 추가 연결이 거부됩니다. 최대 연결을 적용하지 않기로 선택하는 경우에는 최대값이 초과될 때 서버가 이벤트 로그에 메시지를 기록합니다.

12. (선택 사항) QOS (Quality of Service)에 대한 추가 이름-값 쌍 속성을 지정하려면 "등록 정보" 버튼을 누릅니다.

    QOS (Quality of Service) 등록 정보에 허용되는 이름-값 목록은 온라인 도움말을 참조하십시오.

13. "저장"을 눌러 변경 사항을 서버 인스턴스에 완결합니다.

14. 왼쪽 창에서 "응용 프로그램 서버 인스턴스"와 서버 인스턴스에 액세스한 다음 "변경 사항 적용"을 누릅니다.

obj.conf 파일에 필요한 변경 사항

QOS (Quality of Service)을 적용하려면 `obj.conf` 파일에 다음 SAF (Server Application Function)를 호출하는 지시문을 포함시켜야 합니다.

- AuthTrans `qos-handler`
- Error `qos-error`

제대로 적용하려면 `qos-handler` AuthTrans 지시문이 기본 객체에 구성되는 첫 번째 `AuthTrans`어야 합니다. QOS (Quality of Service) 처리기의 역할은 가상 서버, 가상 서버 클래스 및 전역 서버에 대한 현재 통계를 조사하고 오류를 반환하여 제한을 적용하는 것입니다. Sun ONE Application Server에는 `qos-handler`라는 기본 제공 샘플 QOS (Quality of Service) 처리기 SAF가 포함되어 있습니다. 이 SAF는 제한에 도달할 때 이를 기록하고 503 `Server busy` 오류를 서버에 반환하여 NSAP에서 이를 처리하도록 합니다.

Sun ONE Application Server에는 또한 `qos-error`라는 기본 제공 샘플 오류 SAF도 포함되어 있습니다. 이 SAF는 503 오류를 초래한 제한과 그 제한을 트리거한 통계 값을 명시한 오류 페이지를 반환합니다.

이러한 SAF와 그 사용 방법에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guide to NSAPI*를 참조하십시오.

QOS (Quality of Service)에 사용되는 제한

QOS (Quality of Service) 기능 사용 시 다음과 같은 제한 사항이 있습니다.

- QOS (Quality of Service) 기능은 응용 프로그램 수준에서의 HTTP 대역폭만 측정합니다. HTTP 대역폭은 다음과 같은 여러 가지 이유 때문에 실제 TCP 네트워크 대역폭과 다를 수 있습니다.
  - SSL이 활성화되어 있는 경우에는 핸드셰이크 및 클라이언트 인증서 교환이 트래픽에 추가되지만 측정되지 않습니다.
  - 청크 인코딩이 어느 한 방향 또는 양방향으로 활성화되어 있는 경우에는 청크 계층이 청크 헤더를 제거하며 트래픽에 계산되지 않습니다. 다른 헤더나 프로토콜 항목은 계산됩니다.
- QOS (Quality of Service) 기능은 `PR_TransmitFile` 호출에서 트래픽을 정확히 측정할 수 없습니다. `PR_Send()/net_write` 또는 `PR_Recv()/net_read`와 같은 기본 입출력 작업의 경우, 한 시스템 호출에 전송된 바이트 수는 일반적으로 버퍼의 크기이고 입출력 호출은 빨리 반환하므로 전송된 데이터는 대역폭 관리자에 의해 신속하게 계산될 수 있습니다. 이는 동적 내용 응용 프로그램의 순간 대역폭을 측정하는 데 매우 유용합니다. 그러나 `PR_TransmitFile`에서 전송되는 데이터의 양은 전송 과정의 마지막에 가서야 알 수 있기 때문에 전송이 완료되기 전에 측정할 수 없습니다.

  `PR_TransmitFile`이 짧은 경우에는 QOS (Quality of Service) 기능이 적절하게 수행됩니다. 그러나, `PR_TransmitFile`이 길면 전화 접속 사용자가 긴 파일을 다운로드할 경우 전송된 전체 데이터의 양은 전송 완료 시에 계산됩니다. 대역폭 관리자가 다음 재계산 간격 기간이 시작된 후 대역폭을 재계산하면 계산된 대역폭은 최근의 크기가 큰 `PR_TransmitFile`로 인해 크게 증가됩니다. 이 경우 대역폭 관리자가 너무 오래된 파일 전송 작업을 "만료"하여 대역폭 값이 다시 감소하는 다음 메트릭 간격까지 서버에서 모든 요청을 거부하게 될 수 있습니다. 매우 긴 정적 파일 다운로드가 자주 이루어지는 사이트인 경우에는 메트릭 간격을 기본값인 30초보다 길게 설정해야 합니다.
- 계산된 대역폭은 즉시 측정되지 않지만 정기적 간격을 두고 일정 기간마다 재계산되기 때문에 항상 근사값입니다. 예를 들어, 메트릭 간격이 기본값인 30초이고 서버가 29초 동안 유휴 상태로 있을 경우 다음 1초에 클라이언트는 1초의 대역폭 제한의 30배를 사용할 가능성이 있습니다.

- QOS (Quality of Service) 대역폭 통계는 서버가 동적으로 재구성될 때마다 없어집니다. 또한 대역폭 관리자 스레드는 활성 구성에 대한 대역폭 통계만 계산하기 때문에 더 오래된 비활성 구성에 연결된 스레드에서는 QOS (Quality of Service) 제한이 적용되지 않습니다. 오랫동안 소켓을 닫지 않고 활성 상태로 남아 있어서 서버에서 시간 초과로 처리되지 않는 클라이언트에는 동적 서버 재구성 후 QOS (Quality of Service) 제한이 적용되지 않을 가능성이 있습니다.
- 동시 연결은 가상 서버 클래스와 전역 서버 인스턴스에 대한 경우와 가상 서버에 대한 경우 서로 다른 단위로 계산됩니다. 개별 가상 서버에 대한 연결 카운터는 요청을 구문 분석하여 가상 서버로 경로를 지정한 후 즉시 극소량 증가되고 그 요청에 대한 응답 처리 과정의 마지막에 미세하게 감소됩니다. 이것은 가상 서버 연결 통계가 항상 정확하다는 것을 의미합니다.

  그러나 가상 서버 클래스와 전역 서버 인스턴스에 대한 연결 통계는 즉시 업데이트되지 않습니다. 이 통계는 대역폭 관리자 스레드에 의해 매 재계산 간격마다 업데이트됩니다. 가상 서버 클래스의 연결 횟수는 해당 클래스의 모든 가상 서버 연결의 합이고, 전역 서버 인스턴스 연결 횟수는 모든 가상 서버 클래스 연결의 합입니다.

  이러한 값의 계산 방법 때문에 가상 서버에 대한 연결 횟수는 항상 정확하며 연결 횟수에 제한을 적용하는 경우에는 그 제한보다 많아지는 일은 없습니다. 가상 서버 클래스와 서버 인스턴스 값은 정해진 간격에만 계산되기 때문에 아주 정확하지는 않습니다.

SNMP 정보

SNMP (Simple Network Management Protocol)는 네트워크 간에 관리 및 모니터링 정보를 교환하는데 사용됩니다. SNMP를 사용하면 데이터가 관리 대상 장치와 네트워크 관리 스테이션(NMS) 사이를 이동합니다. *관리 대상 장치*는 호스트, 라우터, HTTP 서버 및 기타 네트워크 상의 서버 등 SNMP를 실행하는 모든 장치를 말합니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 네트워크 관리 스테이션(NMS)
- 관리 정보 베이스(MIB) 객체

- SNMP 메시지
- SNMP 트랩 대상
- SNMP 에이전트 커뮤니티

네트워크 관리 스테이션(NMS)

네트워크 관리 스테이션(NMS)은 특정 네트워크를 원격 관리하는 데 사용하는 시스템입니다. 일반적으로 NMS 소프트웨어는 수집된 데이터를 표시하는 그래프를 제공하거나 그 데이터를 사용하여 서버가 특정 허용 한도 내에서 작동되도록 합니다.

NMS는 일반적으로 한 개 이상의 네트워크 관리 응용 프로그램이 설치된 고성능 워크스테이션입니다. HP OpenView와 같은 네트워크 관리 응용 프로그램은 HTTP 서버와 같은 관리 대상 장치에 대한 정보를 그래픽으로 표시합니다. 예를 들어, 네트워크상의 어떤 서버가 실행 중이고 어떤 서버가 다운되어 있는지, 오류 메시지의 수와 유형은 어떤 것인지 등을 표시할 수 있습니다. SNMP를 Sun ONE Application Server와 함께 사용하면 이 정보는 하위 에이전트와 마스터 에이전트 두 종류의 에이전트를 사용하여 NMS와 서버 간에 전송됩니다.

하위 에이전트는 여러 도메인에서 실행되는 서버 인스턴스에 대한 정보를 수집하여 마스터 에이전트로 전달합니다. Sun ONE Application Server의 모든 설치에는 마스터 에이전트와 하위 에이전트가 하나씩 있습니다.

주 SNMP 구성을 변경한 후에는 "적용" 버튼을 누른 다음 SNMP 하위 에이전트를 다시 시작해야 합니다.

---

*마스터 에이전트*는 여러 하위 에이전트와 NMS 간에 정보를 교환합니다. 마스터 에이전트는 Sun ONE Application Server와 함께 설치됩니다.

하나의 호스트 컴퓨터에 여러 개의 하위 에이전트를 설치할 수 있지만 마스터 에이전트는 하나만 설치할 수 있습니다. 예를 들어, Sun ONE Directory Server, Sun ONE Application Server 및 Sun ONE Messaging Server가 같은 호스트에 설치된 경우 각 서버의 하위 에이전트는 같은 마스터 에이전트와 통신합니다.

NMS는 서버에서 정보를 요청하거나 서버의 MIB에서 변수 저장 값을 변경합니다. 예를 들면 다음과 같습니다.

1. NMS는 Admin Server 마스터 에이전트에 메시지를 보냅니다. 메시지는 데이터 요청(GET 메시지)이거나 MIB에 변수를 설정하기 위한 지침(SET 메시지)일 수 있습니다.

2. 마스터 에이전트는 메시지를 해당 하위 에이전트로 전달합니다.
3. 하위 에이전트는 데이터를 검색하거나 MIB에서 변수를 변경합니다.
4. 하위 에이전트는 데이터나 상태를 마스터 에이전트에게 보고하고 마스터 에이전트는 메시지(`GET` 메시지)를 NMS로 다시 전달합니다.
5. NMS는 데이터를 네트워크 관리 응용 프로그램을 통해 텍스트나 그래픽으로 표시합니다.

관리 정보 베이스(MIB) 객체

Sun ONE Application Server는 네트워크를 통해 관리 및 모니터링 정보와 관련된 변수를 저장합니다. 마스터 에이전트가 액세스할 수 있는 변수를 관리 대상 객체라고 합니다. 이러한 객체는 *관리 정보 베이스*(MIB)라 불리는 트리와 비슷한 구조로 정의됩니다. MIB은 HTTP 서버의 네트워크 구성, 상태 및 통계에 대한 액세스를 제공합니다. SNMP를 사용하여 네트워크 관리 스테이션(NMS)에서 이 정보를 볼 수 있습니다.

MIB 트리의 최상위 수준은 인터넷 객체 ID가 다음과 같은 하위 트리를 가지고 있다는 것을 보여줍니다.

- directory (1)
- mgmt (2)
- experimental (3)
- private (4)

private (4) 하위 트리는 enterpises (1) 노드를 포함합니다. enterprises (1) 노드의 각 하위 트리는 고유한 특정 MIB 확장명을 등록한 조직인 개별 enterprise에 할당됩니다. enterprise는 그 하위 트리 아래에 제품별 하위 트리를 만들 수 있습니다. 회사에서 만든 MIB은 enterprises (1) 노드 아래에 위치합니다.

각 Sun ONE Application Server 하위 에이전트는 SNMP 통신에 사용할 MIB을 제공합니다. 서버는 이러한 변수를 포함한 메시지나 트랩을 보냄으로써 중요한 이벤트를 NMS에 보고합니다. NMS는 서버의 MIB에 데이터에 대한 쿼리를 수행할 수 있습니다.

각 Sun ONE Application Server에는 *install_dir*`/lib`에 고유한 MIB를 가집니다.

Sun ONE Application Server의 MIB은 `appserv.mib`라는 파일입니다. 이 MIB에는 Sun ONE Application Server의 네트워크 관리와 관련된 여러 변수의 정의가 포함되어 있습니다.

Sun ONE Application Server MIB의 객체 ID는

`appserver 1 (as appserver7 OBJECT IDENTIFIER ::= {appserver 1 })`이며 *install_dir*`/lib` 디렉토리에 있습니다.

Sun ONE Application Server MIB을 사용하여 Sun ONE Application Server에 대한 관리 정보를 보고 서버를 실시간으로 모니터할 수 있습니다. 다음 표에서는 `appserv.mib` 파일에 저장된 관리 대상 객체 목록과 그에 대한 설명을 제공합니다.

`appserv.mib` 관리 대상 객체 및 설명

관리 대상 객체	설명
`iwsCpuID`	CPU ID
`iwsCpuIdleTime`	유휴 CPU 시간
`iwsCpuKernelTime`	CPU 커널 시간
`iwsCpuTable`	Sun ONE Application Server CPU
`iwsCpuUserTime`	CPU 사용자 시간
`iwsInstanceTable`	Sun ONE Application Server 인스턴스
`iwsInstanceId`	서버 인스턴스 ID
`iwsInstanceVersion`	`SunONE-ApplicationServer-Enterprise/7 BB1-01/24/2001 17:15 (SunOS DOMESTIC)` 등의 문자열
`iwsInstanceDescription`	서버 인스턴스에 대한 설명
`iwsInstanceOrganization`	서버 인스턴스 담당 조직
`iwsInstanceContact`	서버 인스턴스 담당자 연락처 정보
`iwsInstanceLocation`	서버 위치
`iwsInstanceStatus`	서버 인스턴스의 상태
`iwsInstanceUptime`	서버 실행 기간
`iwsInstanceDeathCount`	서버 인스턴스 프로세스 고장 횟수
`iwsInstanceRequests`	서버 인스턴스에서 처리한 요청의 수
`iwsInstanceInOctets`	서버 인스턴스에 수신된 8진수의 수. 이 정보가 없는 경우 0이 표시됩니다.
`iwsInstanceOutOctets`	서버 인스턴스에서 전송한 8진수의 수. 이 정보가 없는 경우 0이 표시됩니다.

`appserv.mib` 관리 대상 객체 및 설명*(계속)*

관리 대상 객체	설명
`iwsInstanceCount2xx`	서버 인스턴스에서 발생된 200 수준(성공) 응답의 수
`iwsInstanceCount3xx`	서버 인스턴스에서 발생된 300 수준(리디렉션) 응답의 수
`iwsInstanceCount4xx`	서버 인스턴스에서 발생된 400 수준(클라이언트 오류) 응답의 수
`iwsInstanceCount5xx`	서버 인스턴스에서 발생된 500 수준(서버 오류) 응답의 수
`iwsInstanceCountOther`	서버 인스턴스에서 발생된 2xx, 3xx, 4xx 또는 5xx이 아닌 기타 응답의 수
`iwsInstanceCount200`	서버 인스턴스에서 발생된 200(확인) 응답의 수
`iwsInstanceCount302`	서버 인스턴스에서 발생된 302(임시 이동) 응답의 수
`iwsInstanceCount304`	서버 인스턴스에서 발생된 304(수정되지 않음) 응답의 수
`iwsInstanceCount400`	서버 인스턴스에서 발생된 400(잘못된 요청) 응답의 수
`iwsInstanceCount401`	서버 인스턴스에서 발생된 401(허가되지 않음) 응답의 수
`iwsInstanceCount403`	서버 인스턴스에서 발생된 403(금지) 응답의 수
`iwsInstanceCount404`	서버 인스턴스에서 발생된 404(찾을 수 없음) 응답의 수
`iwsInstanceLoad1MinuteAverage`	서버 인스턴스를 실행하는 시스템의 1분 로드 평균
`iwsInstanceLoad5MinuteAverage`	서버 인스턴스를 실행하는 시스템의 5분 로드 평균
`iwsInstanceLoad15MinuteAverage`	서버 인스턴스를 실행하는 시스템의 15분 로드 평균
`iwsInstanceNetworkInOctets`	초당 네트워크에서 전송된 8진수의 수
`iwsInstanceNetworkOutOctets`	초당 네트워크에 수신된 8진수의 수
`iwsVsTable`	서버 가상 서버
`iwsVsId`	가상 서버 ID

appserv.mib 관리 대상 객체 및 설명*(계속)*

관리 대상 객체	설명
iwsVsRequests	가상 서버에서 처리한 요청의 수
iwsVsInOctets	가상 서버로 수신된 8진수의 수
iwsVsOutOctets	가상 서버로 전송된 8진수의 수
iwsVsCount2xx	가상 서버에서 발생된 200 수준(성공) 응답의 수
iwsVsCount3xx	가상 서버에서 발생된 300 수준(리디렉션) 응답의 수
iwsVsCount4xx	가상 서버에서 발생된 400 수준(클라이언트 오류) 응답의 수
iwsVsCount5xx	가상 서버에서 발생된 500 수준(서버 오류) 응답의 수
iwsVsCountOther	가상 서버에서 발생된 2xx, 3xx, 4xx 또는 5xx이 아닌 응답의 수
iwsVsCount200	가상 서버에서 발생된 200(확인) 응답의 수
iwsVsCount302	가상 서버에서 발생된 302(임시 이동) 응답의 수
iwsVsCount304	가상 서버에서 발생된 304(수정되지 않음) 응답의 수
iwsVsCount400	가상 서버에서 발생된 400(잘못된 요청) 응답의 수
iwsVsCount401	가상 서버에서 발생된 401(허가되지 않음) 응답의 수
iwsVsCount403	가상 서버에서 발생된 403(금지) 응답의 수
iwsVsCount404	가상 서버에서 발생된 404(찾을 수 없음) 응답의 수
iwsProcessTable	Sun ONE Application Server 프로세스
iwsProcessId	운영 체제 프로세스 ID
iwsProcessThreadCount	요청 처리 스레드의 수
iwsProcessThreadIdle	현재 유휴 상태인 요청 처리 스레드의 수
iwsProcessConnectionQueueCount	현재 연결 대기열에 있는 연결의 수

appserv.mib 관리 대상 객체 및 설명*(계속)*

관리 대상 객체	설명
iwsProcessConnectionQueuePeak	동시에 대기열에 있던 최대 연결 수
iwsProcessConnectionQueueMax	연결 대기열에 허용된 최대 연결 수
iwsProcessConnectionQueueTotal	승인된 연결의 수
iwsProcessConnectionQueueOverflows	연결 대기열 오버플로로 인해 거부된 연결의 수
iwsProcessKeepaliveCount	현재 연결 유지 대기열에 있는 연결의 수
iwsProcessKeepaliveMax	연결 유지 대기열에 허용된 최대 연결 수
iwsProcessSizeVirtual	프로세스 크기(KB)
iwsProcessSizeResident	프로세스 상주 크기(KB)
iwsProcessFractionSystemMemoryUsage	시스템 메모리에서 프로세스 메모리의 비율
iwsListenTable	Sun ONE Application Server 수신 소켓
iwsListenId	수신 소켓 ID
iwsListenAddress	소켓 수신 주소
iwsListenPort	소켓 수신 포트
iwsListenSecurity	암호화 지원
iwsThreadPoolCount	거부된 요청 수
iwsThreadPoolMax	대기열에 허용된 최대 요청 수
iwsThreadPoolPeak	동시에 대기열에 있던 최대 요청 수
iwsThreadPoolTable	Sun ONE Application Server 스레드 풀
iwsVsCount503	발생된 503(사용 불가능) 응답의 수
iwsInstanceCount503	발생된 503(사용 불가능) 응답의 수

SNMP 메시지

GET 및 SET는 SNMP로 정의된 두 가지 유형의 메시지입니다.

각 객체에는 MIB 내의 고유한 ID가 할당됩니다. SNMP 관리자는 객체의 고유 ID를 지정하는 GET 및 GETNEXT 명령을 발생하여 객체에 액세스할 수 있습니다. 프록시 에이전트는 지정된 객체의 값을 얻어 SNMP 관리자에게 전송합니다. 로그에 추가된 이벤트는 트랩 필터 조건을 만족할 경우 SNMP 트랩을 생성할 수 있습니다. 트랩을 생성하지 않는 이벤트는 유지 관리 로그 테이블에 항목으로 기록만 되고 정상적인 GET 및 GETNEXT 명령을 통해 SNMP 관리자에서 액세스합니다.

GET 메시지와 SET 메시지는 네트워크 관리 스테이션(NMS)에 의해 마스터 에이전트로 전송됩니다. 관리 인터페이스를 통해 두 메시지 중 어느 하나를 사용하거나 두 메시지를 모두 사용할 수 있습니다.

SNMP는 네트워크 정보를 프로토콜 데이터 단위(PDU) 형태로 교환합니다. 이 단위는 HTTP 서버와 같은 관리 대상 장치에 저장된 변수에 대한 정보를 포함합니다. 관리 대상 객체로도 알려진 이 변수는 필요에 따라 NMS로 보고되는 값과 이름을 가집니다. 서버에서 NMS로 전송된 프로토콜 데이터 단위를 *트랩*이라고도 합니다. GET, SET 및 트랩 메시지의 사용에 대해서는 다음 절에 보다 자세히 설명되어 있습니다.

SNMP 트랩 대상

*SNMP 트랩*은 SNMP 에이전트에서 네트워크 관리 스테이션(NMS)으로 보내는 메시지입니다. 예를 들어, SNMP 에이전트는 시작부터 종료까지 인터페이스의 상태가 변경될 때 트랩을 보냅니다. SNMP 에이전트는 트랩을 보낼 NMS의 주소를 알아야 합니다.

Sun ONE Application Server 관리 인터페이스에서 SNMP 마스터 에이전트에 대한 트랩 대상을 구성할 수 있습니다. 또한 이미 구성된 트랩 대상을 보고, 편집 및 제거할 수 있습니다. 관리 인터페이스를 사용하여 구성 트랩 대상을 구성할 때 실제로는 CONFIG 파일을 편집하는 것입니다.

서버 하위 에이전트는 중요한 이벤트가 발생했을 때 NMS에 메시지 또는 트랩을 보냅니다. 예를 들면 다음과 같습니다.

1. 하위 에이전트는 서버가 중단되었다는 것을 마스터 에이전트에게 알립니다.
2. 마스터 에이전트는 이 이벤트를 NMS에 보고하는 메시지 또는 트랩을 보냅니다.
3. NMS는 해당 네트워크 관리 응용 프로그램을 통해 이 정보를 텍스트나 그래픽으로 표시합니다.

SNMP 트랩 포트 설정에 대한 지침은 170페이지의 “SNMP 마스터 에이전트 설치”를 참조하십시오.

SNMP 에이전트 커뮤니티

*SNMP 에이전트 커뮤니티*는 지정된 커뮤니티에 할당된 커뮤니티 문자열과 작업으로 구성됩니다. 커뮤니티 문자열은 SNMP 에이전트가 승인에 사용하는 네트워크 관리 스테이션(NMS) 이름에 대한 텍스트 문자열입니다. 이것은 NMS에서 에이전트에 보내는 각 메시지와 함께 커뮤니티 문자열을 보낸다는 것을 의미합니다.

할당된 작업은 `get` 및/또는 `set`입니다. SNMP 에이전트는 NMS가 데이터 교환을 위해 `get`, `set` 또는 `get` 및 `set` 작업 두 가지 모두 수행하도록 승인받았는지 여부를 확인할 수 있습니다. 커뮤니티 문자열은 SNMP 패킷으로 보낼 경우 숨겨지지 않고 ASCII 텍스트로 전송됩니다.

커뮤니티 문자열과 각 지정된 커뮤니티에 대해 허용된 작업은 관리 인터페이스에서 구성하고 관리할 수 있습니다. SNMP 에이전트 커뮤니티 설정에 대한 지침은 170페이지의 “SNMP 마스터 에이전트 설치”를 참조하십시오.

SNMP 설정

일반적으로 SNMP를 사용하려면 하나의 마스터 에이전트와 최소한 하나의 하위 에이전트가 시스템에 설치 및 실행되어야 합니다. 마스터 에이전트를 설치해야 하위 에이전트를 사용할 수 있습니다. 170페이지의 “SNMP 마스터 에이전트 설치”를 참조하십시오.

SNMP 설정 절차는 시스템에 따라 다릅니다. 다음 표에서는 여러 상황에서 따라야 할 절차에 대한 개요를 제공합니다. 실제 절차에 대해서는 이 장의 뒷부분에 자세히 설명되어 있습니다.

서버 조건	수행 절차 (다음 절에 자세히 설명됨)
현재 실행 중인 원시 에이전트가 없는 경우	1. 마스터 에이전트를 시작합니다. 2. 시스템에 설치된 각 서버의 하위 에이전트를 활성화합니다.

서버 조건	수행 절차 (다음 절에 자세히 설명됨)
• 현재 원시 에이전트를 실행 중인 경우 • SMUX가 없는 경우 • 원시 에이전트를 계속 사용할 필요가 없는 경우	1. Administration Server에 대한 마스터 에이전트 설치 시 원시 에이전트를 중단합니다. 2. 마스터 에이전트를 시작합니다. 3. 각 서버 인스턴스에 대한 SNMP 하위 에이전트를 구성합니다.
• 현재 원시 에이전트를 실행 중인 경우 • SMUX가 없는 경우 • 원시 에이전트를 계속 사용해야 하는 경우	1. 프록시 SNMP 에이전트를 설치합니다. 2. 프록시 SNMP 에이전트를 시작합니다. 3. 마스터 에이전트 포트 번호 이외의 포트 번호를 사용하여 원시 에이전트를 다시 시작합니다. 4. 마스터 에이전트를 시작합니다. 5. 시스템에 설치된 각 서버에 대한 하위 에이전트를 활성화합니다.

시작하기 전에 다음 두 가지를 확인합니다.

- 시스템에서 SNMP 에이전트(운영 체제의 원시 에이전트)를 이미 실행하고 있습니까?
- 그렇다면 원시 SNMP 에이전트에서 SMUX 통신을 지원합니까?

이 정보를 확인하는 방법에 대해서는 해당 시스템 설명서를 참조하십시오.

주 Admin Server에서 SNMP 설정을 변경하고 새 서버를 설치하거나 또는 기존 서버를 삭제한 후에는 다음 단계를 수행해야 합니다.

- (Windows 2000) Windows SNMP 서비스를 다시 시작하거나 시스템을 재부트합니다.
- (Unix의 경우) Admin Server를 사용하여 SNMP 마스터 에이전트와 SNMP 하위 에이전트를 다시 시작합니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 프록시 SNMP 에이전트(UNIX/Linux) 사용
- SNMP 마스터 에이전트 설치

프록시 SNMP 에이전트(UNIX/Linux) 사용

이미 원시 에이전트가 실행 중이고 이를 Sun ONE Application Server 마스터 에이전트와 함께 동시에 사용하려는 경우 프록시 SNMP 에이전트를 사용합니다. 시작하기 전에 원시 마스터 에이전트를 중단하십시오. 자세한 내용은 해당 시스템 설명서를 참조하십시오.

주	프록시 에이전트를 사용하려면 먼저 설치한 다음 시작해야 합니다. 또한 Sun ONE Application Server 마스터 에이전트가 실행되고 있는 포트가 아닌 포트를 사용하여 원시 SNMP 마스터 에이전트를 다시 시작해야 합니다.

이 절에서는 다음 내용을 설명합니다.

- 프록시 SNMP 에이전트 설치
- 프록시 SNMP 에이전트 시작
- 원시 SNMP 데몬 재시작

프록시 SNMP 에이전트 설치

SNMP 에이전트가 시스템에 실행 중이며 원시 SNMP 데몬을 계속 사용하려는 경우에는 다음 절에 설명된 단계를 수행합니다.

1. SNMP 마스터 에이전트 설치를 설치합니다. 170페이지의 “SNMP 마스터 에이전트 설치”를 참조하십시오.
2. 프록시 SNMP 에이전트를 설치 및 시작하고 원시 SNMP 데몬을 다시 시작합니다. 168페이지의 “프록시 SNMP 에이전트(UNIX/Linux) 사용”을 참조하십시오.
3. SNMP 마스터 에이전트를 시작합니다. 172페이지의 “SNMP 마스터 에이전트 활성화 및 시작”을 참조하십시오.
4. 하위 에이전트를 활성화합니다. 179페이지의 “하위 에이전트 활성화하기”를 참조하십시오.

SNMP 프록시 에이전트를 설치하려면 서버 루트 디렉토리의 *install_dir*/lib/snmp/sagt에 있는 CONFIG 파일을 편집합니다. 이 파일에는 다른 이름을 지정할 수 있습니다. 프록시 에이전트는 또한 MIB 트리와 프록시 SNMP 에이전트가 전달할 트랩을 포함해야 합니다.

다음은 CONFIG 파일의 예입니다.

```
AGENT AT PORT 1161 WITH COMMUNITY public
SUBTREES 1.3.6.1.2.1.1,
       1.3.6.1.2.1.2,
       1.3.6.1.2.1.3,
       1.3.6.1.2.1.4,
       1.3.6.1.2.1.5,
       1.3.6.1.2.1.6,
       1.3.6.1.2.1.7,
       1.3.6.1.2.1.8
FORWARD ALL TRAPS;
```

프록시 SNMP 에이전트 시작

프록시 SNMP 에이전트를 시작하려면 명령 프롬프트에 다음을 입력합니다.

```
# sagt -c CONFIG&
```

원시 SNMP 데몬 재시작

프록시 SNMP 에이전트를 시작한 후에 CONFIG 파일에 지정된 포트에서 원시 SNMP 데몬을 다시 시작합니다.

원시 SNMP 데몬을 다시 시작하려면 명령 프롬프트에 다음을 입력합니다.

```
# snmpd -P port_number
```

여기서 *port_number*는 CONFIG 파일에 지정된 포트 번호입니다. 예를 들어, 앞서 언급한 CONFIG 파일 예에 있는 포트를 사용하는 Solaris 플랫폼에는 다음과 같이 입력합니다.

```
# snmpd -P 1161
```

SNMP 마스터 에이전트 설치

주 서버가 `root`로 실행되고 있지 않으면 관리 인터페이스를 사용하여 마스터 SNMP 에이전트를 설치 및 시작할 수 없습니다.

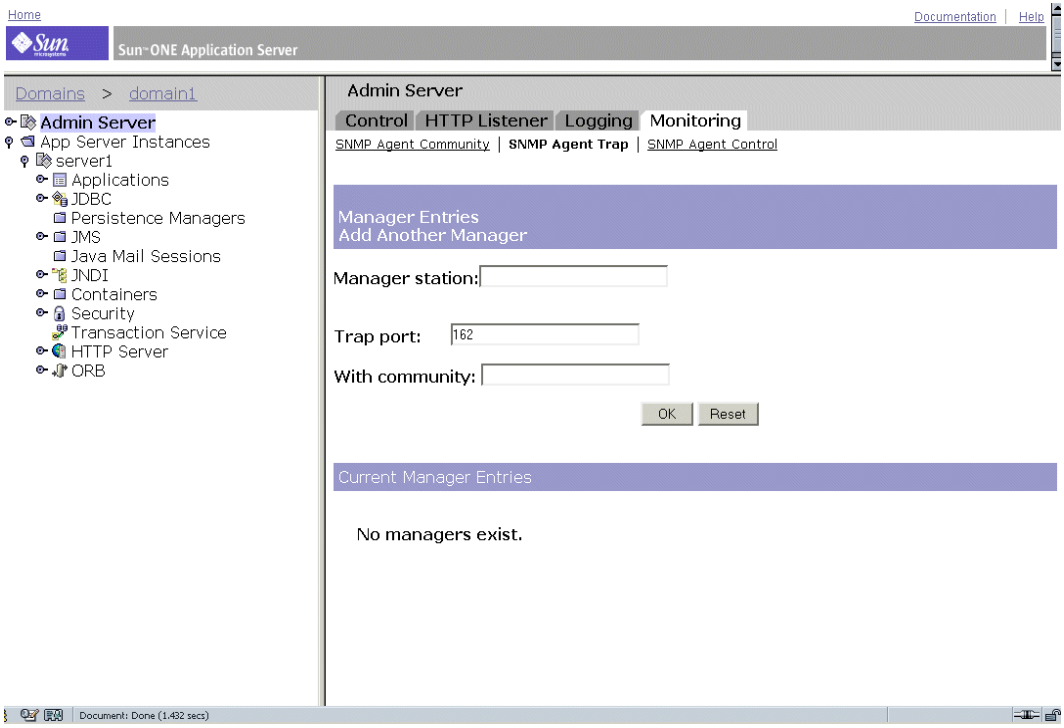
마스터 SNMP 에이전트를 설치하려면 다음 작업을 수행합니다.

1. root로 로그인합니다.
2. SNMP 데몬(`snmpd`)이 포트 161에서 실행 중인지의 여부를 확인합니다.

   SNMP 데몬이 실행 중이 아닌 경우 4단계로 이동합니다.

   SNMP 데몬이 실행 중인 경우에는 재시작 방법 및 지원되는 MIB 트리를 확인합니다.
3. SNMP 데몬이 실행 중인 경우 해당 프로세스를 종료합니다.
4. 관리 인터페이스 왼쪽 창에서 Admin Server 노드를 선택합니다.
5. "모니터링" 탭을 선택하여 다음 그림에 표시된 "SNMP 에이전트 트랩" 페이지를 표시합니다.

SNMP 에이전트 트랩 페이지

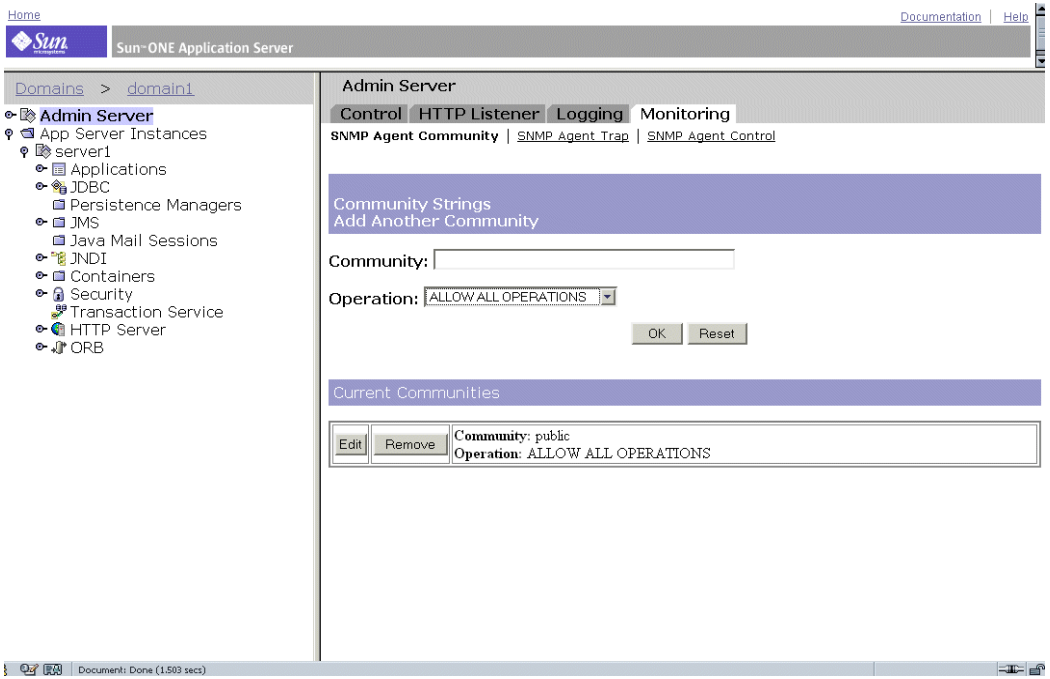


이 페이지에는 "관리자 항목" 정보가 표시됩니다.

6. 네트워크 관리 소프트웨어가 실행 중인 시스템의 이름을 입력합니다.
7. 네트워크 관리 시스템에서 트랩을 수신하는 트랩 포트 번호를 입력합니다. 자주 사용되는 포트는 162입니다. 트랩에 대한 자세한 내용은 165페이지의 “SNMP 트랩 대상”을 참조하십시오.
8. 트랩에 사용하려는 커뮤니티 문자열을 입력합니다. 커뮤니티 문자열에 대한 자세한 내용은 166페이지의 “SNMP 에이전트 커뮤니티”를 참조하십시오.
9. "확인"을 누릅니다.
10. "모니터링" 탭에 있는 "SNMP 에이전트 커뮤니티" 링크를 누릅니다.

    다음 그림과 같이 커뮤니티 문자열 정보가 표시됩니다.

SNMP 에이전트 커뮤니티 페이지



11. 마스터 에이전트의 커뮤니티 문자열을 입력합니다.
12. 커뮤니티의 작업 수준을 선택합니다.

    커뮤니티를 설정한 후 설정 사항을 편집하거나 이 페이지의 "현재 커뮤니티" 머리글 내에 표시된 버튼에서 커뮤니티를 제거할 수 있습니다.
13. "확인"을 누릅니다.
14. 왼쪽 창에서 "응용 프로그램 서버 인스턴스"와 서버 인스턴스에 액세스한 다음 "변경 사항 적용"을 누릅니다.

SNMP 마스터 에이전트 활성화 및 시작

마스터 에이전트 작업은 `CONFIG`라는 에이전트 구성 파일에 정의되며 이 파일을 수동으로 편집할 수 있습니다. SNMP 하위 에이전트를 활성화하기 전에 마스터 SNMP 에이전트를 설치해야 합니다.

주 마스터 에이전트를 다시 시작할 때 `System Error: Could not bind to port`와 비슷한 바인드 오류가 발생하면 `ps -ef | grep snmp`를 사용하여 `magt`가 실행 중인지 확인합니다. magt가 실행 중인 경우에는 `kill -9` *pid* 명령을 사용하여 그 프로세스를 종료합니다. SNMP의 CGI 작동이 다시 시작할 것입니다.

이 절에서는 다음 내용을 설명합니다.

- 다른 포트에서 마스터 에이전트 시작하기
- 수동으로 SNMP 마스터 에이전트 구성하기
- 마스터 에이전트 CONFIG 파일 편집하기
- sysContact 및 sysLocation 변수 정의하기
- SNMP 마스터 에이전트 구성하기
- SNMP 마스터 에이전트 시작하기
- 하위 에이전트 활성화하기

다른 포트에서 마스터 에이전트 시작하기

관리 인터페이스는 161 포트 이외의 다른 포트에서 SNMP 마스터 에이전트를 시작하지 않습니다. 하지만 다음 단계를 수행하면 마스터 에이전트를 다른 포트에서 수동으로 시작할 수 있습니다.

1. *install_dir*`/lib/snmp/magt/CONFIG`를 편집하여 원하는 포트를 지정합니다.
2. 다음과 같이 시작 스크립트를 실행합니다.

```
cd instance_root/admin-server ./start -shell
install_dir/lib/snmp/magt/magt
install_dir/lib/snmp/magt/CONFIG
install_dir/lib/snmp/magt/INIT
```

마스터 에이전트가 원하는 포트에서 시작됩니다. 그러나, 관리 인터페이스에서는 마스터 에이전트가 실행 중이라는 것을 감지할 수 있습니다.

수동으로 SNMP 마스터 에이전트 구성하기

수동으로 마스터 SNMP 에이전트를 구성하려면 다음 작업을 수행합니다.

1. root로 로그인합니다.
2. 포트 161에 실행 중인 SNMP 데몬(`snmpd`)이 있는지 확인합니다.

   SNMP 데몬이 실행 중이면 재시작 방법과 지원되는 MIB 트리를 확인한 다음 그 프로세스를 종료합니다.
3. 서버 루트 디렉토리의 `lib/snmp/magt`에 있는 `CONFIG` 파일을 편집합니다.
4. (선택 사항) 174페이지의 "sysContact 및 sysLocation 변수 정의하기"에 설명된 대로 `sysContact` 및 `sysLocation` 변수를 `CONFIG` 파일에 정의합니다.

마스터 에이전트 CONFIG 파일 편집하기

`CONFIG` 파일에서 마스터 에이전트와 함께 사용할 커뮤니티와 관리자를 정의합니다. 관리자 값은 유효한 시스템 이름이거나 IP 주소여야 합니다.

다음은 기본 `CONFIG` 파일의 예입니다.

```
COMMUNITY          public
                   ALLOW ALL OPERATIONS

MANAGER            manager_station_name
                   SEND ALL TRAPS TO PORT 162
                    WITH COMMUNITY public
```

sysContact 및 sysLocation 변수 정의하기

`CONFIG` 파일을 편집하여 `sysContact` 및 `sysLocation` MIB-II 변수를 지정하는 `sysContact` 및 `sysLocation` 변수의 초기 값을 추가할 수 있습니다. 다음 예에서 `sysContact`와 `sysLocation`의 문자열은 큰따옴표로 묶여 있습니다. 공간, 줄바꿈, 탭 등이 포함된 모든 문자열은 큰따옴표 안에 사용해야 합니다. 값을 16진수로 지정할 수도 있습니다.

다음은 `sysContact` 및 `sysLocation` 변수가 정의된 `CONFIG` 파일의 예입니다.

```
COMMUNITY          public
                   ALLOW ALL OPERATIONS

MANAGER            nms2
                   SEND ALL TRAPS TO PORT 162
                   WITH COMMUNITY public

INITIAL            sysLocation "Server room
901 San Antonio Road
Palo Alto CA 94303
USA"

INITIAL            sysContact "John Doe
email: jdoe@sun.com"
```

SNMP 하위 에이전트 구성하기

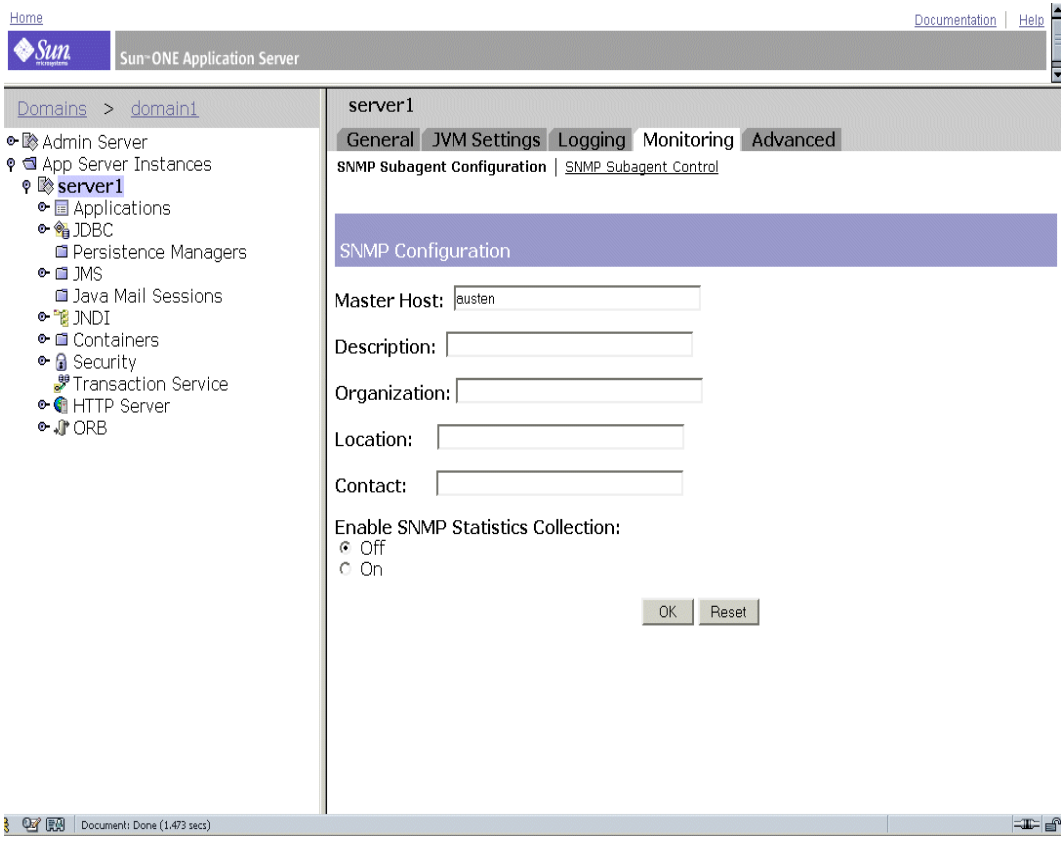
SNMP 하위 에이전트를 구성하려면 다음을 수행합니다.

1. 왼쪽 창의 "Admin Server"에서 "서버 인스턴스" 노드를 선택합니다.
2. 오른쪽 창에서 "모니터링" 탭을 선택합니다.

3. "SNMP 하위 에이전트 구성" 링크를 선택합니다.

   다음 페이지가 표시됩니다.

SNMP 하위 에이전트 구성 페이지



4. (UNIX만 해당) "마스터 호스트" 필드에 서버의 이름과 도메인을 입력합니다.
5. 운영 체제 정보를 비롯한 서버에 대한 "설명"을 입력합니다.
6. 서버를 담당하는 "조직"을 입력합니다.
7. 서버 인스턴스의 위치를 입력합니다.
8. "연락처" 필드에 서버를 담당하는 사람의 이름과 연락처 정보를 입력합니다.
9. "SNMP 통계 수집 활성화"에서 "설정"을 선택합니다.

10. "확인"을 누릅니다.

11. 왼쪽 창에서 "응용 프로그램 서버 인스턴스"와 서버 인스턴스에 액세스한 다음 "변경 사항 적용"을 누릅니다.

SNMP 마스터 에이전트 시작하기

일단 SNMP 마스터 에이전트를 설치한 후에는 이를 수동으로 시작하거나 관리 인터페이스에서 Admin Server를 사용하여 시작할 수 있습니다.

수동으로 SNMP 마스터 에이전트 시작하기

수동으로 마스터 에이전트를 시작하려면 명령 프롬프트에 다음과 같이 입력합니다.

```
# magt CONFIG INIT&
```

`INIT` 파일은 MIB-II 시스템 그룹의 정보를 포함하는 비휘발성 파일입니다. `INIT`이 아직 없는 경우에는 마스터 에이전트를 처음으로 시작할 때 생성됩니다.

주 `CONFIG` 파일에 잘못된 관리자 이름이 있으면 마스터 에이전트가 시작되지 않습니다.

기본 포트가 아닌 포트에서 마스터 에이전트를 시작하려면 다음 중 한 가지 방법을 사용합니다.

방법 1: `CONFIG` 파일에 마스터 에이전트가 관리자로부터 SNMP 요청을 수신하는 각 인터페이스의 전송 매핑을 지정합니다. 전송 매핑을 통해 마스터 에이전트는 기본 포트와 기본 포트가 아닌 포트에서 연결을 받아들일 수 있습니다. 또한 마스터 에이전트는 기본 포트가 아닌 포트에서 SNMP 트래픽을 받아들일 수 있습니다. 최대 동시 SNMP 수는 열려 있는 소켓 수나 프로세스당 파일 설명자 수에 대한 대상 시스템의 제한으로 제한됩니다. 다음은 전송 매핑 항목의 예입니다.

```
TRANSPORT          extraordinary   SNMP
                   OVER UDP SOCKET
                    AT PORT 11161
```

`CONFIG` 파일을 수동으로 편집한 후에는 명령 프롬프트에 다음을 입력하여 마스터 에이전트를 수동으로 시작합니다.

```
# magt CONFIG INIT&
```

방법 2: `/etc/services` 파일을 편집하여 마스터 에이전트가 기본 포트 및 기본 포트가 아닌 포트에서 연결을 받아들일 수 있습니다.

Admin Server를 사용하여 SNMP 마스터 에이전트 시작하기

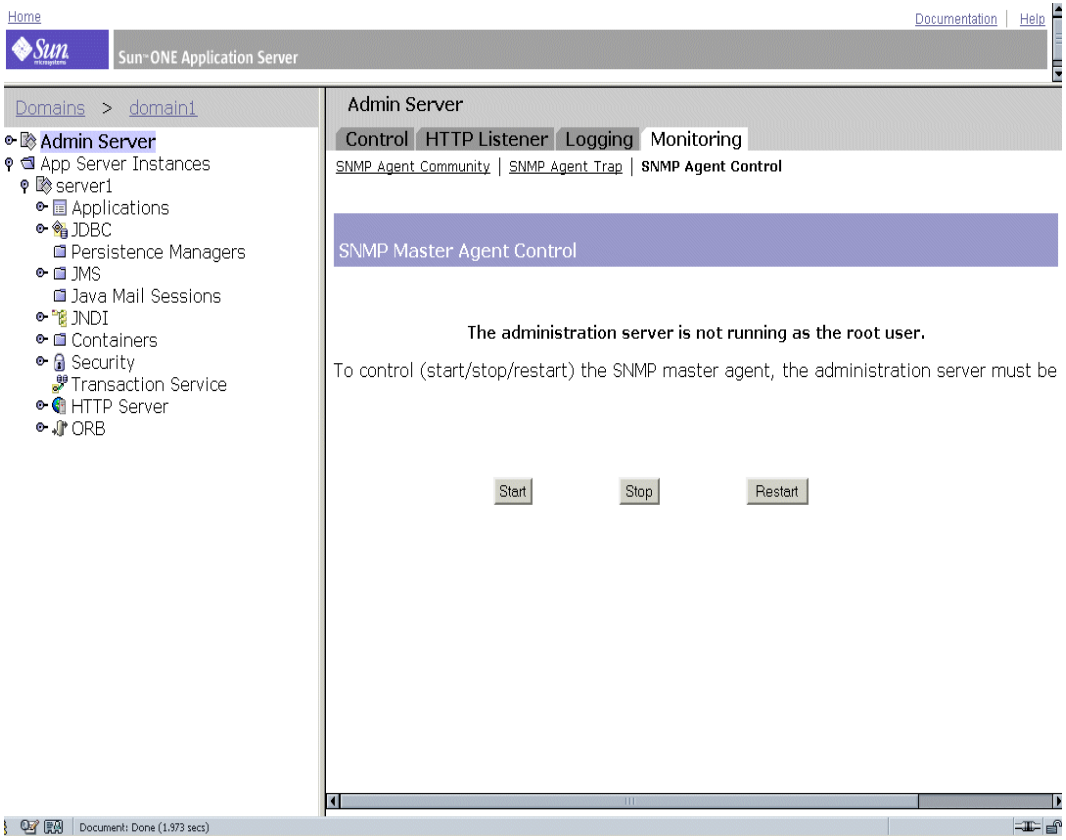
Admin Server를 사용하여 SNMP 마스터 에이전트를 시작하려면 다음을 수행합니다.

주 SNMP 마스터 에이전트를 시작하려면 Sun ONE Application Server에 root로 로그인해야 합니다.

1. Admin Server에 로그인합니다.
2. 왼쪽 창의 "Admin Server" 노드에서 "모니터링" 탭을 선택합니다.
3. 오른쪽 창 상단에 있는 "SNMP 에이전트 제어" 링크를 선택합니다.

   다음 페이지가 표시됩니다.

SNMP 에이전트 제어 페이지



4. "시작"을 누릅니다.

   "SNMP 에이전트 제어" 페이지에서 SNMP 마스터 에이전트를 중단하고 다시 시작할 수도 있습니다.

하위 에이전트 활성화하기

Administration Server와 함께 제공되는 마스터 에이전트를 설치한 후에는 시작하기 전에 서버 인스턴스의 하위 에이전트를 활성화해야 합니다. 마스터 에이전트 설치에 대한 내용은 170페이지의 “SNMP 마스터 에이전트 설치”를 참조하십시오.

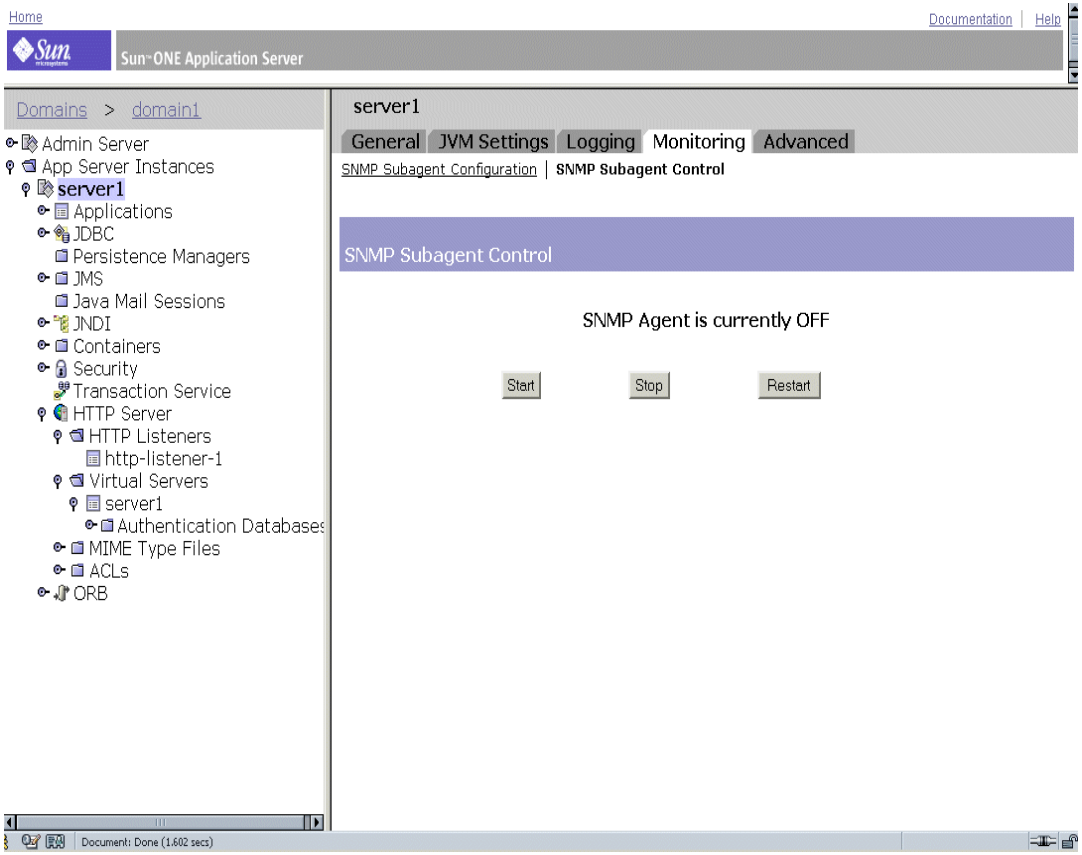
하위 에이전트를 사용하여 UNIX/Linux 플랫폼의 SNMP 기능을 중단할 수 있습니다. 먼저 하위 에이전트를 중단한 후 마스터 에이전트를 중단해야 합니다. 마스터 에이전트를 먼저 중단하면 하위 에이전트를 중단할 수 없을 수도 있습니다. 그런 경우에는 마스터 에이전트를 다시 시작하고 하위 에이전트를 중단한 다음 마스터 에이전트를 중단합니다.

SNMP 하위 에이전트를 활성화하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "응용 프로그램 서버 인스턴스" 노드를 확장합니다.
2. 서버 인스턴스를 선택한 다음 "모니터링" 탭을 누릅니다.

3. "SNMP 하위 에이전트 제어" 옵션을 선택하여 다음 그림과 같은 페이지를 표시합니다.

SNMP 하위 에이전트 제어 페이지



이 페이지에서 SNMP 하위 에이전트를 시작, 중단 또는 재시작할 수 있습니다. 하위 에이전트의 상태는 제어 버튼 바로 위에 표시됩니다.

Windows 플랫폼에서는 Sun ONE Application Server 모니터링에 Windows SNMP 서비스가 사용됩니다. 이 서비스는 "제어판"/"관리 도구"/"서비스" 옵션에서 제어할 수 있습니다.

주 SNMP 구성을 변경한 후 "확인"을 누르고 "SNMP 하위 에이전트 제어" 페이지에서 SNMP 하위 에이전트를 다시 시작해야 합니다.

웹 서버 플러그 인 구성

이 장에서는 Sun ONE Application Server가 HTTP (HyperText Transfer Protocol) 요청을 처리하는 방법과 Sun ONE Application Server에 웹 서버 플러그 인을 구성 및 사용하는 방법에 대해 설명합니다. 또한 Microsoft IIS 및 Apache 웹 서버에 웹 서버 플러그 인을 사용하고 구성하는 방법에 대해서도 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- 웹 서버 플러그 인 정보
- 클라이언트 요청 처리
- 웹 서버 플러그 인 구성
- 웹 서버 플러그 인 SAF 참조
- 웹 서버 플러그 인 사용
- 웹 서버 플러그 인을 사용하도록 Microsoft IIS 구성
- 웹 서버 플러그 인을 Apache 서버에 사용하도록 구성

웹 서버 플러그 인 정보

웹 서버 플러그 인은 Sun ONE Web Server 또는 Sun ONE Application Server가 특정 HTTP 요청을 다른 서버에 전달하도록 지시할 수 있도록 하는 HTTP 역 프록시 플러그 인입니다. 예를 들어, 인터넷에 연결된 웹 서버가 특정 웹 응용 프로그램의 요청을 회사 방화벽 뒤에 있는 응용 프로그램 서버로 전달하도록 구성할 수 있습니다.

Sun ONE Application Server 내에서 웹 서버 플러그 인을 통해 하나의 서버 인스턴스가 다른 서버 인스턴스로 HTTP (웹) 요청을 전달할 수 있습니다.

웹 서버 플러그 인은 다음 기능을 수행합니다.

- 가능한 프록시 서버의 연결을 다시 사용합니다. 이를 통해 받는 요청을 처리하기 위해 새 연결을 열 필요가 없어집니다.
- 웹 서버 플러그 인은 수신할 때마다 요청과 응답의 스트리밍을 시작합니다. 즉, 플러그 인은 원격 서버에 전달하기 전에 요청이나 응답이 완전히 수집될 때까지 기다리지 않습니다.
- 웹 서버 플러그 인은 동일한 원격 서버에 대한 여러 아웃바운드 HTTP 연결을 필요에 따라 관리합니다. 웹 서버 플러그 인이 전달한 요청에 대해 형성된 연결을 아웃바운드 HTTP 연결이라고 합니다.

웹 서버 플러그 인의 작동 방식을 이해하려면 HTTP 요청의 기본을 이해하고, 특히 HTTP 요청을 처리하기 위해 Sun ONE Application Server가 사용하는 방법에 대해 알고 있어야 합니다.

클라이언트 요청 처리

Sun ONE Application Server는 HTTP 요청을 직접 수용하고 응답할 수 있는 응용 프로그램 서버입니다. 이 절에서는 HTTP 기본에 대해 설명하고 Sun ONE Application Server가 요청을 처리하는 방식에 대해 설명합니다. 이 절에서는 다음 항목에 대해 설명합니다.

- HTTP 기본
- 요청 처리 프로세스의 단계

HTTP 기본

요약하자면 HTTP/1.1 프로토콜은 다음과 같이 작동합니다.

- 클라이언트(일반적으로 브라우저)가 서버에 대한 연결을 열고 요청을 보냅니다.
- 서버가 요청을 처리하고, 응답을 생성한 다음 `Connection: Close` 헤더가 있으면 연결을 닫습니다.

요청은 요청된 자원을 나타내는 URI (Universal Resource Identifier)와 공백으로 구분된 HTTP 프로토콜 버전인 `GET` 또는 `POST`와 같은 메소드를 나타내는 행으로 구성됩니다.

이 메소드 뒤에는 일반적으로 헤더, 헤더의 끝을 나타내는 빈 줄 및 본DD문 데이터가 나옵니다. 헤더는 요청 또는 클라이언트 본문 데이터에 대한 다양한 정보를 제공합니다. 헤더는 일반적으로 `POST` 및 `PUT` 메소드의 경우에만 전송됩니다.

아래의 예의 요청은 브라우저가 `foo.com` 서버에게 */index.html*의 자원을 전송하라는 요청을 전송합니다. 이 예에서는 메소드가 `GET` (요청의 시점이D 데이터를 보내는 것이 아니라 데이터를 받기 위한 것이므로)이기 때문에 본문 데이터가 전송되지 않습니다.

```
GET /index.html HTTP/1.0
User-agent: Mozilla
Accept: text/html, text/plain, image/jpeg, image/gif, */*
Host: foo.com
```

서버는 요청을 받아서 처리합니다. 서버는 여러 요청을 동시에 처리할 수 있지만 각 요청을 개별적으로 처리합니다. 각 요청은 요청 처리 프로세스를 구성하는 일련의 단계로 분리됩니다.

서버는 HTTP 프로토콜 버전, HTTP 상태 코드 및 공백으로 분리된 응답 구문이 포함된 응답을 생성합니다. 그 뒤에는 일반적으로 헤더가 나옵니다. 헤더의 끝은 빈 줄로 표시됩니다. 그 뒤에 응답의 본문 데이터가 있습니다. 일반적은 HTTP 응답은 다음과 같습니다.

```
HTTP/1.0 200 OK
Server: Standard/7.0
Content-type: text/html
Content-length: 83

<HTML>
<HEAD><TITLE>Hello World</Title></HEAD>
<BODY>Hello World</BODY>
</HTML>
```

상태 코드 및 응답 구문은 클라이언트에게 서버가 요청을 처리한 방식을 알립니다. 일반적으로 상태 코드 200이 반환되는 것은 요청이 성공적으로 처리되었으며 본문 데이터에 요청된 항목이 포함되어 있음을 의미합니다. 다른 결과 코드는 다른 서버나 브라우저 캐시로의 리디렉션, "404 Not Found"와 같은 여러 HTTP 오류 유형을 나타냅니다.

요청 처리 프로세스의 단계

Sun ONE Application Server가 처음 시작되면 특정 초기화 작업을 수행한 다음 클라이언트(예: 브라우저)의 HTTP 요청을 대기합니다. 요청을 받으면 먼저 가상 서버를 선택합니다.

가상 서버가 선택되면 가상 서버의 `obj.conf` 파일은 다음 단계를 통해 요청 처리 방식을 지정합니다.

1. **AuthTrans** (인증 변환)

   요청에 전송된 인증 정보(예: 이름 및 암호)를 확인합니다.

2. **NameTrans** (이름 변환)

   논리적 URI를 논리적 파일 시스템 경로로 변환합니다.

3. **PathCheck** (경로 확인)

   로컬 파일 시스템의 유효성을 확인하고 요청자가 파일 시스템의 요청된 자원에 액세스 권한을 갖고 있는지 확인합니다.

4. **ObjectType** (객체 유형 지정)

   요청된 자원의 MIME (Multi-purpose Internet Mail Encoding) 유형(예: `text/html`, `image/gif` 등)을 결정합니다.

5. **Service** (응답 생성)

   응답을 생성하여 클라이언트에 반환합니다.

6. **AddLog** (로그 항목 추가)

   로그 파일에 항목을 추가합니다.

7. **Error** (서비스)

   이 단계는 이전 단계에서 오류가 발생한 경우에만 실행됩니다. 오류가 발생하면 서버는 오류 메시지를 기록하고 프로세스를 취소합니다.

웹 서버 플러그 인 구성

웹 서버 플러그 인의 구성 및 동작은 구성 파일 집합에 의해 결정됩니다. Sun ONE Application Server는 클라이언트의 요청을 처리할 때마다 이 파일에 정의된 구성을 찾습니다. 구성 파일의 이름은 `obj.conf` 및 `init.conf`입니다. `obj.conf` 파일에는 `server1-obj.conf`와 같이 가상 서버의 이름이 접두어로 붙습니다. 자세한 내용은 371페이지의 "obj.conf 파일"을 참조하십시오.

Sun ONE Application Server의 각 인스턴스에는 시작할 때 서버가 참조하는 자체 `init.conf` 파일이 있습니다.

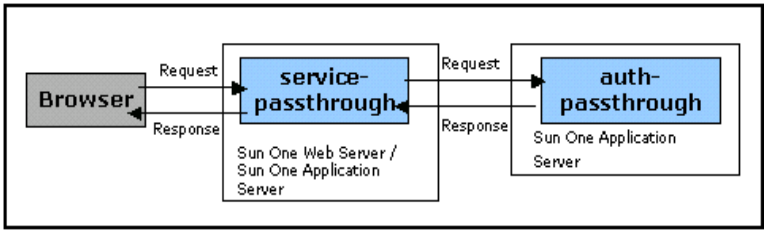
이전 항목에서 설명했듯이 `obj.conf` 구성 파일에는 Sun ONE Application Server에게 클라이언트 요청 및 응답 프로세스의 각 단계에서 무엇을 수행할지 지시하는 지시(지시문)가 포함되어 있습니다. 각 지시문은 SAF (Server Application Function)를 호출합니다.

`obj.conf` 파일은 Sun ONE Application Server의 운영에 필수적인 파일입니다. 관리 인터페이스를 통해 서버를 변경하면 시스템은 `obj.conf`를 자동으로 업데이트합니다.

`init.conf` 구성 파일은 초기화 도중 서버를 구성하는 변수의 값을 설정합니다. 서버는 서버 시작 도중 이 파일에 지정된 구성 매개 변수를 실행합니다. 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

다음 그림에서는 웹 브라우저, 프런트엔드 웹 서버, 백엔드 응용 프로그램 서버 및 웹 서버 플러그 인의 `service-passthrough`와 `auth-passthrough` SAF 사이의 관계를 설명합니다.

웹 브라우저 , 웹 서버 , 응용 프로그램 서버 , 웹 서버 플러그 인 SAF 사이의 관계



웹 서버 플러그 인 SAF 참조

이 절에서는 다음 SAF (Server Application Functions)의 기능과 동작에 대해 설명합니다.

- init-passthrough
- auth-passthrough
- service-passthrough
- check-passthrough

init-passthrough

`init-passthrough` 기능은 웹 서버 플러그 인을 초기화합니다. 이 기능을 호출해야 웹 서버 플러그 인을 사용할 수 있습니다.

예:

```
Init fn="load-modules" shlib="c:/plugins/passthrough.dll"
funcs="init-passthrough,auth-passthrough,check-passthrough,service-
passthrough" NativeThread="no"
Init fn="init-passthrough"
```

auth-passthrough

`auth-passthrough` SAF는 `AuthTrans-class` 지시문에서 사용할 수 있습니다.

auth-passthrough 기능은 받는 HTTP (웹) 요청에서 중개 서버에서 실행 중인 service-passthrough가 인코딩한 클라이언트 정보를 검사합니다. 클라이언트 정보에는 다음이 포함됩니다.

- 요청을 보낸 수신자의 IP 주소
- 보낸 클라이언트가 사용하는 SSL 키 크기
- 보낸 클라이언트가 제시하는 SSL 클라이언트 인증서

auth-passthrough가 인코딩된 클라이언트 정보를 감지하면 요청을 `service-passthrough`가 실행 중인 매개 서버가 전달한 요청이 아니라 보낸 클라이언트의 직접 요청으로 간주합니다.

다음과 같은 2계층 배포 시나리오에서 유용합니다.

- Sun ONE Application Server 인스턴스가 회사 방화벽 뒤의 두 번째 방화벽에 의해 숨겨진 경우
- S1AS 인스턴스에 직접 허용된 클라이언트 연결이 없는 경우

이러한 네트워크 아키텍처에서 클라이언트는 항상 프록시 플러그 인이 실행 중인 프런트엔드 웹 서버에 연결합니다. 이 웹 서버는 요청을 Sun ONE Application Server로 전달하는 서버입니다. 이것은 Sun ONE Application Server가 프록시 호스트(이 경우 웹 서버)의 요청만 받을 수 있고 클라이언트 호스트에서 직접 받을 수 없습니다. 즉, 두 개의 방화벽 뒤에 있는 Sun ONE Application Server 인스턴스에 배포된 응용 프로그램이 클라이언트의 IP 주소와 같은 클라이언트 정보를 쿼리하면 응용 프로그램은 프록시 호스트 IP를 받게 됩니다. 이것이 전달된 요청의 실질적인 보낸 호스트이기 때문입니다. `auth-passthrough` SAF를 사용하면 이 정보 대신 원격(프록시를 거친) 클라이언트 정보가 나타나도록 이 동작을 수정할 수 있습니다.

`auth-passthrough`를 사용하면 인증에 사용될 수 있는 정보(예를 들어, 요청을 보낸 측의 IP 주소)를 무시하는 것이 가능하기 때문에 `auth-passthrough`가 실행 중인 서버에는 신뢰할 수 있는 클라이언트나 서버만 연결할 수 있도록 하는 것이 중요합니다. 더욱 주의하는 의미에서 회사 방화벽 뒤의 서버에서만 `auth-passthrough`를 실행하는 것이 좋습니다. 인터넷을 통해 액세스할 수 있는 서버에서는 `auth-passthrough` SAF를 실행하지 않는 것이 좋습니다. `auth-passthrough` SAF는 보낸 클라이언트에 대한 관련 정보가 요청된 경우에만 사용해야 합니다.

위에서 설명한 시나리오의 경우 SSL 클라이언트 인증을 웹 서버에 대해서만 켜고 응용 프로그램 서버의 경우에는 항상 꺼서 구성이 제대로 작동하도록 할 수 있습니다.

명령 예:

```
AuthTrans fn="auth-passthrough"
```

service-passthrough

`service-passthrough` SAF는 Service-class 지시문에서 사용할 수 있습니다.

`service-passthrough` SAF는 한 서버의 요청을 다른 서버로 전달하여 처리할 수 있습니다. 원래 요청이 수신된 연결 유형에 독립적으로, 원격 서버에 대해 SSL 또는 비 SSL (HTTPS 또는 HTTP) 연결을 사용하도록 `service-passthrough` SAF를 구성할 수 있습니다. `service-passthrough` SAF는 원격 서버에서 실행 중인 `auth-passthrough` 함수에 의해 디코딩될 수 있는 보낸 클라이언트에 대한 정보를 인코딩합니다.

service-passthrough 지시문은 다음과 같이 obj.conf 구성 파일에서 다른 지시문과 결합하여 사용합니다.

```
<Object name="passthrough">
ObjectType fn="force-type" type="magnus-internal/passthrough"
Error reason="Bad Gateway" fn="send-error"
uri="$docroot/badgateway.html"
</Object>
<Object name="default">
....
NameTrans fn="assign-name" from="(/webapp1|/webapp1/*)" name="passthrough"
...
</Object>
```

백엔드 응용 프로그램 서버가 다운되면 사용자에게는 로컬 HTML 파일인 badgateway.html이 대신 표시됩니다. service-passthrough SAF를 실행 중인 서버가 액세스 권한을 갖고 있는 파일을 서비스해야 하고 거부된 요청만 백엔드 응용 프로그램 서버에 전달해야 하는 경우 ObjectType 행을 다음과 같이 변경합니다.

```
ObjectType fn="check-passthrough" type="magnus-internal/passthrough"
```

check-passthrough

check-passthrough SAF는 ObjectType-class 지시문에서 사용할 수 있습니다.

check-passthrough 함수는 요청된 자원(예: HTML 문서 또는 GIF 이미지)을 로컬 서버에서 사용할 수 있는지 확인합니다. 요청된 자원이 로컬에 존재하지 않으면 check-passthrough SAF는 요청을 다른 서버로 전달하여 service-passthrough SAF가 처리하도록 지시하는 유형을 설정합니다.

매개 변수:

type - (선택 사항) 요청 자원이 존재하지 않을 경우 설정할 유형입니다. 기본값은 "magnus-internal/passthrough"입니다.

예

```
ObjectType fn="check-passthrough"
```

웹 서버 플러그 인 사용

Sun ONE Web Server나 Sun ONE Application Server 인스턴스에 플러그 인을 추가하려면 다음 단계를 수행합니다.

1. `libpassthrough.so` (UNIX의 경우)가 *install_dir*`/plugins/passthrough/bin` 디렉토리에 있고 `passthrough.dll` 파일(Windows의 경우)이 *install_dir*`/bin` 디렉토리에 있는지 확인합니다. 플러그 인을 Sun ONE Web Server 6.0에 추가할 때는 플러그 인을 Sun ONE Application Server 7 설치에서 복사해야 합니다.
2. 다음 행을 *install_dir*`/config/init.conf` (Sun ONE Application Server 7의 경우) 또는 *install_dir*`/config/magnus.conf` (Sun ONE Web Server 6.0의 경우) 구성 파일에 추가합니다. 각 행은 `Init`로 시작되며 모두 한 줄에 들어갑니다.

   Windows:

   ```
   Init fn="load-modules" shlib="c:/install_dir/bin/passthrough.dll"
   funcs="init-passthrough,auth-passthrough,check-passthrough,
   service-passthrough" NativeThread="no"

   Init fn="init-passthrough"
   ```

   UNIX:

   ```
   Init fn="load-modules"
   shlib="install_dir/plugins/passthrough/bin/libpassthrough.so"
   funcs="init-passthrough,auth-passthrough,check-passthrough,servi
   ce-passthrough" NativeThread="no"

   Init fn="init-passthrough"
   ```

3. `config/`*appserver-server-instance*`-obj.conf` 구성 파일의 맨 위에 다음 행을 추가하여 passthrough `<Object>`를 `obj.conf`에 추가합니다. Sun ONE Application Server에서는 `config/<`*appserver-server-instance*`>-obj.conf` 파일에 대해 이 단계를 수행해야 합니다.

   ```
   <Object name="passthrough">
   ObjectType fn="force-type" type="magnus-internal/passthrough"
   Service type="magnus-internal/passthrough"
   fn="service-passthrough" servers="server"
   Error reason="Bad Gateway" fn="send-error"
   uri="$docroot/badgateway.html"
   </Object>
   ```

   여기서 서버는 다음 형식의 URL입니다.

   `http://`*servername*`:port`

4. *install_dir*/config/obj.conf 구성 파일의 기본 객체 맨 위에 다음과 같은 행을 추가하여 전달할 URI를 구성합니다.

   ```
   NameTrans fn="assign-name" from="(/uri|/uri/*)"
   name="passthrough"
   ```

   uri는 원격 서버에 배포된 웹 응용 프로그램의 컨텍스트 루트이며 passthrough는 3단계에서 할당된 obj.conf의 <Object> 이름입니다.

   예:

   ```
   <Object name="default">
   ...
   NameTrans fn="assign-name" from="(/webapp1|/webapp1/*)"
   name="passthrough"
   ...
   </Object>
   ```

5. 서버를 다시 시작합니다.

주 Solaris 및 Linux의 경우 플러그 인 라이브러리의 이름은 libpassthrough.so입니다. Windows에서 플러그 인 라이브러리의 이름은 passthrough.dll 입니다.

웹 서버 플러그 인을 사용하도록 Microsoft IIS 구성

Microsoft Internet Information Services가 웹 서버 플러그 인을 사용하도록 구성하려면 Microsoft IIS에서 사용할 수 있도록 웹 서버 플러그 인을 구성하고, 웹 서버 플러그 인을 사용하도록 Microsoft IIS를 구성해야 합니다.

또한 서버 풀이 여러 서버에서 실행되는 여러 응용 프로그램을 처리하도록 구성할 수도 있습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- IIS에서 사용할 수 있도록 웹 서버 플러그 인 구성
- IIS가 웹 서버 플러그 인을 사용하도록 구성
- 여러 서버 풀 구성
- 샘플 sun-passthrough.properties 파일

IIS에서 사용할 수 있도록 웹 서버 플러그 인 구성

웹 서버 플러그 인을 IIS에서 사용할 수 있도록 구성하려면 다음 작업을 수행하십시오.

1. C:\ 명령줄 프롬프트에서 다음 명령을 입력하여 IIS `wwwroot` 디렉토리 아래에 웹 서버 플러그 인용 디렉토리를 만듭니다.

   ```
   md \Inetpub\wwwroot\sun-passthrough
   ```

2. 플러그 인 파일을 `C:\Inetpub\wwwroot\sun-passthrough` 디렉토리에 복사합니다.

3. 텍스트 편집기를 사용하여 Sun ONE Application Server가 설치된 시스템의 URL을 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` 파일에 추가합니다.

   텍스트 편집기를 통해 다음 정보를 추가해야 합니다.

   `server=http://`*appservername*`:`*port*

   여기서 *appservername*은 Sun ONE Application Server가 설치된 시스템의 호스트 이름 또는 IP 주소이며 `port`는 이것이 수신하는 포트의 번호입니다(이 값은 일반적으로 80으로 설정됨).

4. Sun ONE Application Server가 서비스할 컨텍스트 루트를 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` 파일에 나열합니다.

   이러한 컨텍스트 루트는 Sun ONE Application Server에 배포된 응용 프로그램의 컨텍스트 루트에 해당해야 합니다. 이들 컨텍스트 루트에 대한 요청은 Sun ONE Application Server이 서비스해야 하며 다른 요청은 IIS 웹 서버에서 처리합니다. 요청을 웹 응용 프로그램에 전달할 명령줄은 다음과 같습니다.

   `passthrough=/`*webapplication*

   여기서 `/`*webapplication*은 웹 응용 프로그램의 컨텍스트 루트입니다. 모든 요청을 Sun ONE Application Server에 전달하려면 다음 행을 추가하십시오.

   ```
   passthrough=/
   ```

이제 Microsoft IIS 루트 디렉토리에 웹 서버 플러그 인을 구성했습니다. 이제 작업을 완료하려면 Microsoft IIS가 웹 서버 플러그 인을 사용하도록 구성해야 합니다.

IIS가 웹 서버 플러그 인을 사용하도록 구성

IIS가 웹 서버 플러그 인을 사용하도록 구성하려면 Windows 인터넷 서비스 관리자를 열어야 합니다. 인터넷 서비스 관리자는 "제어판" 폴더의 "관리 도구" 폴더에 있습니다.

인터넷 서비스 관리자를 열고 다음 작업을 수행합니다.

1. 플러그 인을 활성화할 웹 사이트를 선택합니다. 이 웹 사이트의 이름은 일반적으로 "기본 웹 사이트"입니다.
2. 웹 사이트를 마우스 오른쪽 버튼으로 누른 다음 "속성"을 선택하여 "속성" 노트북을 엽니다.
3. "ISAPI 필터" 탭을 열고 "추가" 버튼을 누른 다음 아래 단계에 따라 새 ISAPI 필터를 추가합니다.
   a. "필터 이름" 필드에 Sun ONE Application Server를 입력합니다.
   b. "실행 파일" 필드에 C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll을 입력합니다.
   c. "확인"을 누르고 "속성" 노트북을 닫습니다.
4. 이제 새 가상 디렉토리를 만들어서 구성해야 합니다. 다음 단계에 따라 새 가상 디렉토리를 만들고 구성하십시오.
   a. 기본 웹 사이트를 마우스 오른쪽 버튼으로 누른 다음 "가상 디렉토리"를 선택합니다. "가상 디렉토리 작성 마법사"가 열립니다.
   b. "별칭" 필드에 `sun-passthrough`를 입력합니다.
   c. "디렉토리" 필드에 `C:\Inetpub\wwwroot\sun-passthrough`를 입력합니다.
   d. "실행 권한" 확인란이 선택되어 있고 다른 모든 사용 권한 확인란은 선택되어 있지 않아야 합니다.
   e. "마침"을 누릅니다.

5. 웹 서버를 중지했다 시작해야 새 설정이 적용됩니다. 웹 서버를 중지하려면 웹 사이트를 오른쪽 버튼으로 누르고 "중지"를 선택합니다. 웹 서버를 시작하려면 웹 사이트를 오른쪽 버튼으로 누르고 "시작"을 선택합니다.

   그 후 웹 브라우저에 다음 사항을 입력하여 웹 응용 프로그램 컨텍스트 루트에 액세스합니다.

   `http://`*`webservername`*`/`*`webapplication`*

   여기서 *webservername*은 웹 서버의 호스트 이름 또는 IP 주소이며, `/`*webapplication*은 웹 서버, 웹 서버 플러그 인 및 Sun ONE Application Server가 제대로 작동하는지 확인하기 위해 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` 파일에 나열된 컨텍스트 루트입니다.

여러 서버 풀 구성

`sun-passthrough.properties` 파일에 서버 풀을 구성하여 웹 응용 프로그램을 여러 응용 프로그램 서버로 분할할 수 있습니다. 즉, 일부 응용 프로그램은 한 서버 집합에서 실행하고 다른 응용 프로그램은 다른 서버 집합에서 실행할 수 있습니다. 각 서버 풀마다 문자와 숫자로 구성된 고유한 이름을 선택합니다. 190페이지의 “웹 서버 플러그 인을 사용하도록 Microsoft IIS 구성” 절의 설명대로 Microsoft IIS에 대한 웹 서버 플러그 인을 설치 및 구성하는 단계를 마쳤으면 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` 파일을 편집하고 서버 풀에 대해 선택한 고유 이름에 관련 서버 및 passthrough 등록 정보 행을 접두어로 추가합니다. 서버 풀 이름 뒤에는 점(.)을 입력합니다.

예를 들어, `sun-passthrough.properties` 파일의 다음 행은 두 개의 서버 풀을 정의합니다.
첫 번째 서버 풀은 `server-a`와 컨텍스트 루트 `/app1`에 대한 서비스 요청으로 구성됩니다.
두 번째 서버 풀은 `server-b`와 `/app2` 및 `/app3` 컨텍스트 루트에 대한 서비스 요청으로 구성됩니다.

```
server=http://server-a

passthrough=/app1

serverpool2.server=http://server-b

serverpool2.passthrough=/app2

serverpool2.passthrough=/app
```

샘플 sun-passthrough.properties 파일

```
# Sun ONE Application Server web server plugin for IIS
#
# This file is used to configure the Sun ONE Application Server web server
# plugin for IIS.  Lines beginning with a '#' are ignored.
# server
#
# The server property specifies the URL of an application server.  If multiple
# server properties are given, the plugin will distribute load across the
# specified application servers.
#
server=http://localhost:8080
# passthrough
#
# The passthrough property specifies the context root (virtual directory) of a
# web application.  Requests for the given context root will be passed to the
# application server for processing.  If 'passthrough=/' is specified, all
# requests will be passed to the application server for processing.
#
# passthrough properties should be ordered from most to least specific.  For
# example, 'passthrough=/apps/app1' should appear before 'passthrough=/apps'.
#
# Multiple passthrough properties are allowed.
#
#passthrough=/webapp
#passthrough=/servlets
#passthrough=*.jsp
passthrough=/
# prefix
#
```

```
# The prefix property specifies the IIS virtual directory that contains the
# plugin DLL, sun-passthrough.dll.
#
prefix=/sun-passthrough
# error-url
#
# The error-url property specifies the URL of a page to redirect the client to
# when the application server is unavailable.
#
#error-url=/badgateway.htm
# It is possible to configure multiple server pools by prefixing the server
# and passthrough property names with a pool name followed by a period ('.').
# Pool names can be any sequence of letters and numbers.
#
# For example, the following properties define two server pools.  One server
# pool will service the web applications at '/app1' and the other will service
# the web applications at '/app2' and '/app3':
#
#serverpool1.server=http://server-a
#serverpool1.passthrough=/app1
#
#serverpool2.server=http://server-b
#serverpool2.passthrough=/app2
#serverpool2.passthrough=/app3
```

웹 서버 플러그 인을 Apache 서버에 사용하도록 구성

이 절에서는 Apache 소스 코드를 컴파일하고 Apache 웹 서버의 설치를 구성하는 방법에 대해 설명합니다.

1. 최신 Apache 배포판은 `www.apache.org`에서 다운로드합니다.

   소스 배포판의 압축을 풉니다. 소스 배포판은 압축되어 있습니다. Apache 1.3.22를 설치하는 경우 소스 배포판의 이름은 `apache_1.3.22.tar.gz`입니다.

2. 다음 명령을 사용하여 압축 파일의 압축을 풉니다.

   ```
   $ tar -zxvf apache_1.3.26.tar.gz
   ```

   이 명령은 현재 작업 디렉토리에 `apache_1.3.26`이라는 디렉토리를 만듭니다.

3. 이제 Apache 소스 코드를 컴파일하도록 환경을 구성해야 합니다. 소스 배포판에는 환경에 Apache를 제대로 컴파일하기 위해 필요한 지원 파일(예: 헤더, 공유 라이브러리 및 유틸리티 프로그램)이 있는지 검사하는 `configure`라는 스크립트가 함께 제공됩니다.

   환경을 구성하려면 Apache 소스 디렉토리로 이동하여 다음 단계를 계속합니다.

   a. Apache를 Solaris나 Linux에 설치할 때 다음 경로가 있는지 확인합니다.

      - `CC=/usr/dist/share/devpro/5.x-sparc/bin/cc`
      - PATH에 있어야 하며 이 명령은 다음 경로에서 찾을 수 있습니다. `/usr/ccs/bin`
      - 파일을 작성합니다. 이 명령은 다음 경로에서 찾을 수 있습니다. `/usr/ccs/bin/make`

   b. 다음 명령을 실행합니다.

      ```
      $ ./configure &ndash;enable-module=proxy
      --prefix=/usr/local/apache
      ```

   위의 명령에 지정된 디렉토리는 변수입니다. Apache를 설치한 경로를 지정할 수 있습니다. prefix 인수는 Apache를 설치할 위치를 나타냅니다. 이 명령을 실행하면 화면에 몇 줄의 출력이 나타납니다. 이 명령은 시스템 구성에 따라 빌드할 Makefile을 작성합니다. `configure`에 오류가 있으면 다음을 진행하기 전에 설치해야 하는 헤더 파일이나 유틸리티 프로그램이 손실될 수 있습니다.

4. `configure` 스크립트가 성공적으로 실행된 후 다음과 같이 `make` 명령을 사용하여 Apache를 컴파일할 수 있습니다.

   $ make

   이 명령을 실행하면 프로세스가 Apache 소스 코드를 컴파일하고 Apache를 링크하고 있음을 나타내는 몇 줄의 출력이 화면에 표시됩니다. 이 프로세스는 대부분 오류 없이 종료됩니다. 하지만 오류가 발생하면 Apache의 모든 라이브러리 파일 및 유틸리티 프로그램이 제대로 다운로드되었는지 확인하십시오.

5. 이제 Apache를 설치해야 합니다. Apache는 `/usr/local/apache` 디렉토리(또는 사용자가 지정한 다른 디렉토리)에 설치됩니다. Apache를 설치하려면 다음 명령을 실행하십시오.

   ```
   $ make install
   ```

   이 명령이 성공적으로 실행되면 시스템에 Apache가 설치됩니다. 그러면 다음 디렉토리에서 Apache의 설치 파일을 볼 수 있습니다.

   ```
   /usr/local/apache
   ```

   기본 구성 파일인 `httpd.conf`는 `/usr/local/apache/` 디렉토리에 설치됩니다.

6. 다음 명령을 실행하여 Apache를 구성합니다.

   ```
   $ Configure Apache
   ```

   Apache는 `httpd.conf` 파일을 통해 구성됩니다. 이 파일은 Apache 서버의 여러 동작 매개 변수를 결정하는 다양한 Apache 지시문으로 구성됩니다. Apache 단순 설치의 경우 다음 몇 개의 지시문을 수정해야 합니다.

   - Server `Root &ldquo;` (Server Root는 Apache를 설치한 디렉토리. 예: Server Root &/usr/local/apache or /space/apache).
   - `Port &ldquo; 5000&rdquo;` (사용자 선택)`&rdquo;`

그러면 Apache가 기본 동작 및 웹 서비스에 맞게 구성됩니다. 설정하고자 하는 다른 매개 변수가 있는 경우 구성 파일을 참조하십시오.

7. Apache에는 Apache를 시작, 중지 및 다시 시작하는 `apachectl`이라는 스크립트가 번들되어 있습니다. Apache를 시작하려면 다음 명령을 실행합니다.

```
$ /usr/local/apache/bin/apachectl start
```

Apache를 중지하려면 다음 명령을 실행합니다.

```
use /usr/local/apache/bin/apachectl stop
```

시작한 뒤에는 Apache의 설치를 테스트할 수 있습니다. Apache가 실행되면 웹 브라우저에 다음 명령을 입력하십시오. `http://localhost/.` 제대로 설치가 완료되어 Apache가 실행된 경우 메시지가 표시되는 테스트 페이지가 나타납니다.

8장

J2EE 컨테이너 구성

Sun ONE Application Server는 J2EE 1.3 사양에 따른 다양한 J2EE 컨테이너를 제공합니다. 컨테이너는 EJB (Enterprise Java Bean) 및 MDB (Message Driven Bean)와 같은 J2EE 응용 프로그램 구성 요소에 대한 런타임 지원을 제공합니다. MDB와 EJB는 다른 J2EE 응용 프로그램 구성 요소와 직접 상호 작용하지 않습니다. 이들 구성 요소는 EJB 컨테이너의 프로토콜과 메소드를 사용하여 서로 상호 작용하거나 Java Transaction Service와 같은 플랫폼 서비스와 상호 작용합니다. 컨테이너는 응용 프로그램 구성 요소와 J2EE 서비스 간의 상호 작용을 지원합니다. 따라서 컨테이너는 구성 요소 배포 설명자로 정의되는, 선언적 트랜잭션 관리, 보안 검사, 자원 풀링, 상태 관리 등의 서비스를 투명하게 제공할 수 있습니다.

Sun One Application Server는 웹 컨테이너와 EJB 컨테이너를 통합합니다.

이 장에서는 다음 내용을 설명합니다.

- 웹 컨테이너에 대한 정보
- EJB 컨테이너에 대한 정보

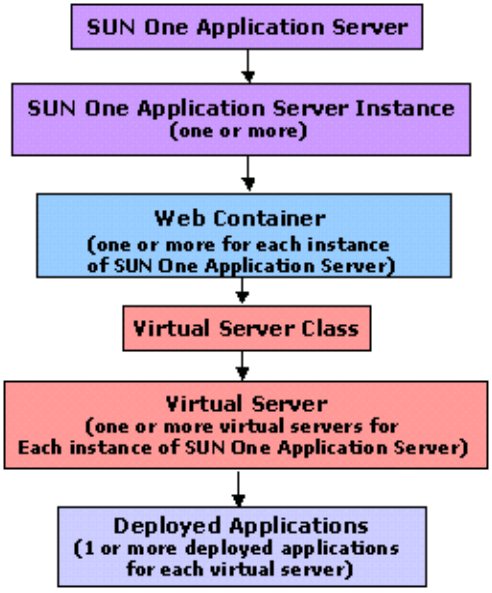
웹 컨테이너에 대한 정보

웹 컨테이너는 웹 응용 프로그램을 호스트하는 J2EE 컨테이너입니다. 웹 컨테이너는 개발자에게 서블릿과 Java Server Page (JSP)를 실행하는 환경을 제공함으로써 웹 서버 기능을 확장합니다. 서블릿은 CGI (Common Gateway Interface) 프로그램의 성능 제한 없이 웹 기반 응용 프로그램을 구축하기 위한 구성 요소 기반 및 플랫폼 독립적인 방법을 제공합니다. JSP 기술은 서블릿 기술을 확장한 것으로, HTML 및 XML 페이지 작성을 지원하기 위해 만들어졌습니다. 웹 컨테이너에 포함된 서블릿이나 JSP는 EJB (Enterprise Java Bean) 컨테이너에서 Bean 메소드를 호출할 수 있습니다. Bean 메소드는 ORB (Object Request Broker)를 사용하여 로컬 호출 또는 원격 호출을 통해 호출할 수 있습니다.

웹 컨테이너는 JNDI (Java Naming Directory Interface)를 사용하여 찾은 로컬 EJB에 웹 응용 프로그램에 대한 액세스를 제공합니다.

그림 "Sun ONE Application Server 구조 내부의 웹 컨테이너"는 Sun ONE Application Server 구조에서 웹 컨테이너의 역할과 위치를 설명합니다.

Sun ONE Application Server 구조 내부의 웹 컨테이너



이 절에서는 다음 항목에 대해 설명합니다.

- 웹 컨테이너의 역할 이해
- 웹 응용 프로그램 구성
- 웹 응용 프로그램 배포
- 단일 사인 온(SSO) 기능
- 웹 컨테이너 로깅

웹 컨테이너의 역할 이해

웹 컨테이너의 주된 역할은 웹 응용 프로그램을 위한 런타임 환경을 제공하고, 해당 컨테이너에 호스트하는 웹 응용 프로그램에 데이터베이스 액세스, 보안, 다중 스레딩과 같은 서비스를 제공하는 것입니다. 웹 응용 프로그램은 서블릿과, HTML 페이지, 클래스 및 Sun ONE Application Server에서 완전한 응용 프로그램을 구성하는 기타 자원의 집합입니다.

웹 응용 프로그램의 요소는 다음과 같습니다.

- 서블릿
- JSP 페이지
- 유틸리티 클래스
- 정적 문서(html, 이미지, 사운드 파일 등)
- 클라이언트측 Java 애플릿, Bean, 클래스
- 위의 모든 요소를 함께 결합하는 설명적 메타 정보

웹 응용 프로그램은 Sun ONE Application Server에서 실행 중인 웹 컨테이너에 배포될 수 있습니다.

Sun ONE Application Server와 함께 웹 서버 플러그 인을 구성하고 사용하는 방법에 대한 자세한 내용은 181페이지의 “웹 서버 플러그 인 구성”을 참조하십시오.

웹 응용 프로그램 구성

또한 웹 컨테이너는 가상 서버 내부에 웹 응용 프로그램을 배포하도록 구성할 수 있습니다. 이 때 웹 컨테이너는 둘 이상의 가상 서버를 포함하도록 구성할 수 있으며, 각 가상 서버는 웹 응용 프로그램을 호스트하도록 구성할 수 있습니다. 웹 응용 프로그램의 배포 범위는 가상 서버의 컨텍스트 내부입니다. 가상 서버에 대한 자세한 내용은 15장 "가상 서버 사용"을 참조하십시오.

이 절에서는 다음 항목에 대해 설명합니다.

- 가상 서버 속성
- Web-module 속성

가상 서버 속성

가상 서버에 구성할 수 있는 특정 속성에 대한 값을 지정할 수 있습니다. 가상 서버는 관련 웹 응용 프로그램을 둘 이상 가질 수 있습니다. 사용자는 웹 응용 프로그램에 *사인 온*해야 합니다.

단일 사인 온(SSO)에 대한 속성인 `sso-enabled`가 `server.xml` 파일에서 기본값인 true로 설정되어 있는 경우, 사용자는 특정 가상 서버와 관련된 모든 웹 응용 프로그램에 사인 온할 수 있습니다. 이 때 사용자의 ID는 동일한 가상 서버에서 실행 중인 다른 모든 응용 프로그램에 의해 인식됩니다. 그러나 `sso-enabled` 값이 false로 설정되어 있을 경우에는 이 가상 서버의 모든 웹 응용 프로그램에 대해 단일 사인 온(SSO)이 비활성화됩니다.

`sso-enabled` 속성은 동적으로 구성할 수 있으며, 서버를 다시 시작하지 않아도 변경 사항이 적용됩니다.

단일 사인 온(SSO)에 대한 자세한 내용은 204페이지의 "단일 사인 온(SSO) 기능"을 참조하십시오.

Web-module 속성

Sun ONE Application Server 고유의 배포 설명자는 `sun-web.xml`라는 파일에 지정되어 있으며, 이 파일은 해당 웹 응용 프로그램의 `WEB-INF` 디렉토리에 들어 있습니다.

일반적으로 각 웹 응용 프로그램에 대해 `sun-web.xml` 파일이 구성되어 있습니다. 그러나 웹 컨테이너의 모든 웹 응용 프로그램이 `sun-web.xml` 파일을 가질 필요는 없습니다. `sun-web.xml` 파일이 없을 경우, 웹 컨테이너는 Sun ONE Application Server 고유의 모든 속성에 대해 기본값을 사용합니다.

context-root 속성

이 속성은 웹 응용 프로그램이 설치된 컨텍스트 루트를 정의합니다. 이 속성이 비어 있는 문자열인 경우, 이 웹 응용 프로그램은 가상 서버에 대한 기본 웹 응용 프로그램으로 지정됩니다. 가상 서버에 대한 기본 웹 응용 프로그램은 가상 서버에 배포된 다른 웹 응용 프로그램에서 처리할 수 없는 모든 요청에 응답합니다. 모든 가상 서버는 기본 웹 응용 프로그램을 갖습니다.

기본 웹 응용 프로그램에서 이 필드의 값은 빈 문자열 `" "`이어야 합니다.

location 속성

이 속성을 입력할 때는 기본 웹 응용 프로그램의 위치를 나타내는 유효한 디렉토리 경로를 입력해야 합니다. 설치 과정 중에 기본 웹 응용 프로그램의 위치는 `modules/`*`default-web-app`*`/` 디렉토리로 설정됩니다.

`location` 속성은 필수 항목이며, WAR (Web ARchive) 파일 내용의 압축을 푼 디렉토리에 대한 정규화된 경로 또는 상대 경로가 될 수 있습니다. 지정된 경로가 상대 경로인 경우, 가상 서버 수준에서 정의된 *응용 프로그램 루트 디렉토리*에 대해 상대 경로이어야 합니다.

예를 들면 다음과 같습니다.

`location="`*applications/<ear name>/<war-module name>/*`"`

`location="`*modules/<war-module name>*`"`

`location="`*/u/myapps/<war-module name>*`"`

`location="`*/u/myapps/<ear-name>/<war-module name>*`"`

enabled 속성

이 속성의 기본값은 웹 응용 프로그램이 요청을 처리할 수 있음을 나타내는 `true`입니다. enabled 속성 값이 `false`로 설정되면, 웹 응용 프로그램이 임시로 요청을 서비스할 수 없게 됩니다. 그러나 하드 디스크에 저장된 웹 응용 프로그램의 내용은 제거되지 않습니다.

웹 응용 프로그램 배포

웹 컨테이너는 WAR (Web ARchive) 파일이나 WAR 파일의 구조 정보(`WEB-INF/lib`, `WEB-INF/`클래스 등)를 포함하는 디렉토리에서 웹 응용 프로그램을 배포합니다. 응용 프로그램을 배포하기 위하여 서버를 다시 시작할 필요는 없습니다.

웹 컨테이너는 각 가상 서버에 "기본" 웹 응용 프로그램을 배포합니다. 기본 위치(디렉토리)는 가상 서버의 `app root` 디렉토리 아래 `modules`*/default-web-app/* 하위 디렉토리에 있습니다. 이 기본 웹 응용 프로그램은 가상 서버에 배포된 다른 웹 응용 프로그램에서 처리할 수 없는 모든 요청에 응답합니다. 이러한 웹 응용 프로그램은 JSP 페이지를 표시할 수 있도록 `/servlet/*`에 대한 요청을 처리하는 호출자 서블릿과 JSP 페이지를 처리하는 JSP 서블릿으로 구성됩니다. 기본 웹 응용 프로그램은 사용자가 `web.xml`과 `sun-web.xml` 파일에서 EJB 참조를 표시하면 EJB에 액세스할 수 있습니다.

기본 웹 응용 프로그램은 가상 서버의 `server.xml`에서 정의되며 다음과 같습니다.

```
<web-module context-root="" location="modules/default-web-app/">
```

동적 재배포와 핫 배포

동적 재배포는 서버를 다시 시작하지 않고 기존 응용 프로그램을 재배포하는 기능입니다. 동적 재배포는 응용 프로그램의 구성(`xml` 파일의 내용)과 특정한 클래스가 변경되었을 때 발생합니다. 동적 재배포를 수행하면 전체 응용 프로그램 클래스가 동적으로 다시 로드되는 것과 같은 동작이 발생합니다. 또한 응용 프로그램 컨텍스트(web 및 ejb)가 새로 만들어지고 이전 응용 프로그램 컨텍스트는 제거됩니다. 따라서 동적 재배포 결과, 새로운 응용 프로그램(기존 세션 데이터 제외) 인스턴스가 만들어지게 됩니다. 이 기능은 개발 모드에서만 지원되며, 동적 재로드 시와 유사한 예외를 발생시킵니다. 또한 구성 변경 사항은 서버를 다시 시작해야만 적용됩니다. 동적 재로드는 중앙 구성에서 이 기능이 지정되어 있는 응용 프로그램과 비공유 독립 실행형 모듈에서만 활성화됩니다.

웹 응용 프로그램이 다시 로드되는 경우, 기존의 모든 세션 정보는 영구 메커니즘이 이러한 세션 관리자에 구성되어 있는지의 여부와 상관없이 자동으로 저장되고 복원됩니다.

핫 배포는 서버를 다시 시작할 필요 없이 서버 런타임에 응용 프로그램을 배포하는 기능입니다. 이 기능은 동적 재배포 작업과 동일한 인프라를 사용합니다. 그러나 이전 구현에서 어떠한 상태도 남아 있지 않기 때문에 이 기능은 실제 작업 시에 지원됩니다.

단일 사인 온(SSO) 기능

사용자가 특정 가상 서버의 웹 응용 프로그램에 있는 보호되지 않은 자원에만 액세스하는 경우에는 인증 절차가 필요 없습니다.

그러나 사용자가 특정 가상 서버와 관련된 웹 응용 프로그램의 보호되는 자원에 액세스하는 경우에는 현재 액세스하는 웹 응용 프로그램에 대해 정의된 로그인 방법을 사용하여 자신을 인증하는 절차를 거쳐야 합니다.

일단 인증을 통과하면 사용자의 역할은 관련된 모든 웹 응용 프로그램에 걸쳐 액세스 제어를 결정하는 데 사용됩니다. 그러면 사용자는 각 웹 응용 프로그램에 대해 개별적으로 인증을 받지 않아도 됩니다.

사용자가 웹 응용 프로그램에서 로그아웃하게 되면, 모든 웹 응용 프로그램의 사용자 세션은 무효화됩니다. 따라서 사용자가 응용 프로그램의 보호되는 자원에 다시 액세스를 하려면 재인증을 받아야 합니다.

단일 사인 온(SSO) 기능은 HTTP 쿠키를 사용하여 각 요청을 저장된 사용자 ID에 연결하는 토큰을 전송합니다. 따라서 단일 사인 온(SSO) 기능은 쿠키를 지원하는 클라이언트 환경에서만 사용할 수 있습니다.

웹 컨테이너 로깅

서로 다른 *로그 수준*을 설정하여 웹 컨테이너와 가상 서버에 호스트되는 응용 프로그램의 기본 로깅 동작을 제어할 수 있습니다. 로깅 동작은 응용 프로그램의 고유한 로깅에 영향을 주지 않아야 합니다.

로그 수준 지정은 기록되는 메시지의 유형을 제어합니다. 예를 들어, 로그 수준 FATAL을 가진 메시지만 기록되도록 지정하면, 이 값보다 "높은" 로그 수준의 메시지는 자연스럽게 무시됩니다. 그리고 명시적 로그 수준으로 기록되는 메시지만 이 값과 비교됩니다.

명시적 로그 수준을 갖지 않은 메시지는 무제한으로 기록됩니다. 기본 동작은 모든 경고, 오류 및 치명적 메시지를 기록하는 것입니다.

웹 컨테이너에 대한 로그 수준을 설정하려면, 다음 작업을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 Sun ONE Application Server 인스턴스 트리를 확장하여 수정할 웹 컨테이너 구성을 찾습니다.
2. "컨테이너" 탭을 확장하여 J2EE 컨테이너 목록이 나타나면 이 목록에서 "웹 컨테이너"를 선택합니다. 관리 인터페이스의 오른쪽 창에서 "웹 컨테이너 로깅"이라고 표시된 페이지가 나타납니다.

웹 컨테이너 로깅

3. "로그 수준" 드롭다운 목록에서 원하는 로그 수준을 선택합니다. 모든 로그 수준과 정의 목록을 보려면, 5장 "로깅 사용"을 참조하십시오.
4. "저장"을 눌러 해당 설정을 저장합니다.

웹 컨테이너에 대한 추가 등록 정보를 만들려면 "등록 정보" 버튼을 누릅니다.

EJB 컨테이너에 대한 정보

Enterprise Java Bean 컨테이너는 Enterprise Bean을 제어하고 이 Bean에 중요한 시스템 수준 서비스를 제공하는 런타임 환경입니다. EJB는 EJB 컨테이너에서 실행되는 구성 요소로서, EJB 서버 내부에서 차례로 실행됩니다. Bean에 제공되는 시스템 수준 서비스는 다음과 같습니다.

- 트랜잭션 관리
- 보안
- lifecycle 관리
- 원격 연결
- 데이터베이스 연결 풀링
- 이름 지정 서비스

Enterprise Bean은 비즈니스 논리를 포함하는, Java로 작성된 서버 구성 요소입니다. EJB 컨테이너는 Bean에 원격 액세스를 제공합니다. EJB는 항상 컨테이너의 컨텍스트 내에서 동작하며, EJB와 이를 호스트하는 서버 간의 링크의 역할을 합니다. EJB 컨테이너는 고유의 구성 요소와 다른 공급업체의 구성 요소를 사용하여 분산 응용 프로그램을 구축할 수 있습니다.

EJB 컨테이너를 통해 Sun ONE Application Server는 높은 수준의 트랜잭션, 상태 관리, 다중 스레딩 및 자원 풀링 래퍼를 제공하므로, 낮은 수준의 API 세부 정보를 알 필요가 없도록 보호해 줍니다. 이 컨테이너는 2.0 EJB Specification에서 가능한 모든 기본 컨테이너 서비스뿐 아니라 Sun ONE Application Server에 특정한 추가 서비스도 제공합니다.

또한 컨테이너는 비활성화와 활성화 프로세스를 사용하여 확장성을 보장할 있도록 Bean 동작을 관리합니다.

이 절에서는 다음 항목에 대해 설명합니다.

- EJB 컨테이너 역할의 이해
- EJB 컨테이너 구성

EJB 컨테이너 역할의 이해

EJB 컨테이너는 다음 기본 서비스를 제공합니다.

- 비활성화

  EJB를 메모리에서 보조 저장 장치로 전달하는 프로세스입니다. 비활성화를 사용하면 Bean을 삭제하지 않고 Bean의 자원을 해제할 수 있습니다. 이 방법을 사용하여 Bean은 지속적인 상태로 만들 수 있고, 인스턴스화의 오버헤드 없이 재호출될 수 있습니다.

- 활성화

  EJB를 보조 저장 장치에서 메모리로 전달하는 프로세스입니다. 컨테이너 계약은 EJB와 해당 컨테이너 간의 관계를 설정하며 클라이언트에 대해 완전하게 투명합니다. 이러한 관계는 다음과 같습니다.

  - lifecycle

    Session Bean의 경우, lifecycle에는 `javax.ejb.SessionBean` 및 `javax.ejb.SessionSynchronization` 인터페이스 구현이 포함됩니다. Entity Bean의 경우, lifecycle에는 `javax.ejb.EntityBean` 인터페이스 구현이 포함됩니다. Message-Driven Bean의 경우, lifecycle에는 `javax.ejb.MessageDriven` 인터페이스 구현이 포함됩니다.

  - 세션 컨텍스트

    컨테이너는 Bean 인스턴스가 만들어질 때 Session Bean 인스턴스로 서비스와 정보를 전달할 수 있도록 `javax.ejb.SessionContext` 인터페이스를 구현합니다.

  - 엔티티 컨텍스트

    컨테이너는 Bean 인스턴스가 만들어질 때 Entity Bean으로 서비스와 정보를 전달할 수 있도록 `javax.ejb.EntityContext` 인터페이스를 구현합니다.

  - 메시지 컨텍스트

    컨테이너는 Bean 인스턴스가 만들어질 때 Message-Driven Bean으로 서비스와 정보를 전달할 수 있도록 `javax.ejb.MDBContext` 인터페이스를 구현합니다.

- 환경

  컨테이너는 `java.util.Properties`를 구현하고 이러한 등록 정보를 EJB에 대해 사용할 수 있게 합니다.

- 서비스 정보

  컨테이너는 EJB에서 사용할 수 있는 서비스를 만듭니다.

Sun ONE Application Server 서비스로는 원격 액세스, 이름 지정, 보안, 동시성, 트랜잭션 제어, 데이터베이스 액세스 등이 있습니다.

EJB 컨테이너의 기능은 다음과 같습니다.

- 원격 연결을 허용하는 구현 객체(`EJBObject`) 만들기
- `EJBObject` 작성을 허용하는 홈 구현 객체 만들기
- 클라이언트가 홈 객체를 찾을 수 있도록 홈 구현 객체를 이름 지정 서비스에 바인딩
- 권한이 있는 클라이언트만이 `EJBObject` 통해 Bean 메소드를 호출
- 비즈니스 메소드를 해당 트랜잭션에서 호출
- Bean의 lifecycle 관리 Bean의 lifecycle 관리에는 다음이 포함됩니다.
  - Bean 풀링
  - 적절한 콜백 메소드 호출(예: `ejbActivate/ejbPassivate`)
  - 응용 프로그램이 연결을 더욱 효과적으로 사용 및 재사용할 수 있도록 데이터베이스 연결 풀 관리

실제적인 구현 세부 정보는 컨테이너와 EJB 간의 표준 규정 인터페이스를 기반으로 하는 컨테이너의 일부입니다. 따라서 플랫폼별 구현 세부 정보를 이해하거나 처리할 필요가 없습니다. 그 대신 EJB 표준을 지원하는 공급업체의 제품과 함께 사용할 수 있도록 일반적이고 작업 중심적인 EJB를 만들 수 있습니다.

Sun ONE Application Server에서 사용되는 EJB의 유형을 알아 두면 도움이 됩니다.

Enterprise Java Bean의 유형

EJB는 다음 중 하나를 나타내는 객체입니다.

- 다중 클라이언트 호출 메소드에 대해 자동으로 상태를 유지하는 특정한 클라이언트의 세션
- 다중 클라이언트 간에 공유될 수 있는 지속적인 엔티티 객체
- 메시지 처리와 같은 상태 없는 서비스

Java Database Connectivity (JDBC) API를 사용하여 데이터 액세스를 처리하는 데 기본적으로 사용되는 Entity Bean입니다. 한편, Session Bean은 임시 응용 프로그램 객체를 제공하고 개별 비즈니스 작업을 수행합니다. EJB에는 다음과 같은 세 가지 종류가 있습니다.

- Session Bean에 대한 정보
- Entity Bean에 대한 정보
- Message-Driven Bean에 대한 정보

Session Bean에 대한 정보

Session Bean은 특정한 클라이언트 요청에 대한 비즈니스 규칙이나 논리를 구현합니다.

Session Bean은 개별 클라이언트를 위한 단일 데이터베이스 기록, 편집을 위한 문서 복사 또는 전문화된 비즈니스 객체와 같이 임시 객체와 프로세스를 나타내기 위해 고안되었습니다. 다시 말해, Session Bean은 이 Bean을 만든 클라이언트에 의해서만 사용되는 개인 자원입니다. 이 객체는 한 클라이언트에서만 사용할 수 있으므로, Session Bean은 *대화* 상태라고 하는 클라이언트 고유의 세션 정보를 유지할 수 있습니다.

예를 들어, 전자 장바구니를 생성하기 위해 EJB를 만들 수 있습니다. 그러면 사용자가 응용 프로그램에 로그인할 때마다 응용 프로그램은 이 사용자의 구매를 위한 장바구니 Session Bean을 만들게 됩니다. 사용자가 로그아웃하거나 쇼핑을 끝내면 이 Session Bean은 해제됩니다.

세션은 다음과 같은 특징을 갖습니다.

- Session Bean은 단일 클라이언트와 관련하여 실행됩니다.
- Session Bean은 비교적 수명이 짧습니다.
- Session Bean은 서버 충돌에서 살아 남지 못할 수도 있습니다.
- EJB 컨테이너가 충돌하는 경우, Session Bean은 제거됩니다.
- Session Bean은 등록 정보 설정에 따라 트랜잭션 관리를 처리합니다. 이것은 선택 사항입니다.

- Session Bean은 기본 데이터베이스에서 공유 데이터를 업데이트합니다. 이것은 선택 사항입니다.
- Stateless Session Bean이거나 Stateful Session Bean일 수 있습니다.

Stateless Session Bean. Stateless Session Bean은 제한된 시간 범위 내에서 특정한 고객에 필요한 비즈니스 논리의 임시 부분을 캡슐화합니다. Stateless Session Bean은 대화 상태를 유지하지 않습니다.

Stateful Session Bean. StatefulSession Bean은 임시적이나 클라이언트 호출 간에 내용과 값에 대한 정보를 보존할 수 있도록 대화 상태를 유지합니다. 대화 상태를 통하여 Bean의 컨테이너는 Session Bean 상태에 대한 정보를 유지하고 필요한 경우, 나중에 프로그램 실행에서 상태를 다시 만들 수 있습니다.

Entity Bean에 대한 정보

Entity Bean은 일반적으로 데이터베이스에서 직접 유지되거나 객체로서 Enterprise Information System (EIS) 응용 프로그램을 통해 액세스되는 지속적인 데이터를 나타냅니다. EJB를 호스트하는 서버와 EJB 컨테이너는 동시에 활성화되는 엔티티 EJB에 대한 확장 가능한 런타임 환경을 제공합니다.

Entity Bean의 간단한 예로는 데이터베이스 테이블의 한 행을 나타내기 위해 정의되는 Entity Bean이 있습니다. 이 때 각 Bean 인스턴스는 특정한 행을 나타냅니다. 더 복잡한 예로는 데이터베이스의 결합된 테이블의 복잡한 보기를 나타내기 위해 설계된 Entity Bean이 있습니다. 여기서 각 Bean 인스턴스는 단일 장바구니의 내용을 나타냅니다.

Entity Bean은 다음과 같은 특징을 갖습니다.

- Entity Bean은 EIS 자원(일반적으로 데이터베이스)에서 데이터의 객체 보기를 제공합니다.
- Entity Bean은 모든 사용자가 액세스할 수 있습니다.
- Entity Bean은 서버 충돌에서 투명하게 보존됩니다.
- Entity Bean은 컨테이너 관리 트랜잭션 또는 Bean 관리 트랜잭션을 사용합니다.

Entity Bean은 컨테이너 관리 지속성(CMP) 또는 Bean 관리 지속성(BMP)으로 지속성 데이터를 나타냅니다. Bean 자체 또는 컨테이너에서 Entity Bean의 지속성을 관리할 수 있습니다.

Bean 관리 지속성. Entity Bean이 고유 지속성을 관리하는 경우 Bean 개발자는 EJB 클래스 메소드에서 직접 지속적인 코드(예: JDBC 호출)를 구현합니다. 소유 인터페이스를 사용하는 경우, 이식성이 떨어지거나 특정한 데이터베이스에 Bean을 연결시킬 때 위험성이 있다는 단점이 있습니다.

컨테이너 관리 지속성. Entity Bean이 컨테이너에서 관리되는 경우 컨테이너는 지속성 상태를 투명하게 관리하기 때문에 Bean 메소드에서 데이터 액세스 코드를 구현할 필요가 없습니다. 이 메소드는 더욱 쉽게 구현할 수 있을 뿐 아니라 특정한 데이터베이스에 연결하지 않아도 Bean을 완전 이식할 수 있습니다.

컨테이너 관리 지속성을 사용하는 Entity Bean은 Bean 관리 지속성을 사용하는 Entity Bean의 (컨테이너에 의한) 자동 생성 버전이어야 합니다.

Entity Bean의 구축과 사용에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guide to Enterprise JavaBeans Technology*를 참조하십시오.

Message-Driven Bean에 대한 정보

Message-Driven Bean은 J2EE 응용 프로그램이 비동기적으로 메시지를 처리할 수 있도록 해주는 EJB입니다. Message-Driven Bean은 Java Message Service 메시지가 도착하면 구동됩니다.

Message-Driven Bean 인스턴스는 생성되어 삭제될 때까지 Message-Driven Bean 컨테이너에서 상주합니다. 컨테이너는 보안, 트랜잭션, 메시지 동시 처리, Message-Driven Bean 인스턴스의 lifecycle 관리 및 Message-Driven Bean에 대한 기타 서비스를 제공합니다. EJB와 EJB 컨테이너를 호스트하는 서버는 동시 활성화된 Message-Driven Bean에 대한 확장 가능한 런타임 환경을 제공합니다.

J2EE 1.3 플랫폼의 Java Message Service API는 다음 내용을 지정합니다.

- 응용 프로그램 클라이언트, EJB 구성 요소 및 웹 구성 요소는 Java Message Service 메시지를 전송하거나 비동기적으로 수신할 수 있습니다. 또한 응용 프로그램 클라이언트는 Java Message Service 메시지를 비동기적으로 사용할 수 있습니다.
- Message-Driven Bean은 메시지의 비동기적인 소모가 가능합니다. Java Message Service 제공자는 Message-Driven Bean에 의해 메시지의 동시 처리를 선택적으로 구현할 수도 있습니다.

Message-Driven Bean은 상태 없는 서비스를 나타냅니다. 따라서 기본적으로 완전 익명이거나 클라이언트 표시 ID가 없는 비동기 메시지 사용자입니다. Message-Driven Bean에는 홈과 구성 요소 인터페이스가 없습니다. 클라이언트는 Message-Driven Bean 클래스가 MessageListener가 되는 Java Message Service 대상(대기열 또는 항목)에 메시지를 전송하여 Java Message Service를 통하여 Message-Driven Bean에 액세스합니다.

Message-Driven Bean만이 비동기적으로 메시지를 수신할 수 있습니다. 세션 또는 Entity Bean은 Java Message Service `MessageListener`가 될 수 없습니다.

Message-Driven Bean은 다음과 같은 특징을 갖습니다.

- 단 하나의 클라이언트 메시지를 받아 실행됩니다.
- 비동기적으로 호출됩니다.
- 비교적 수명이 짧습니다.
- 데이터베이스에서 공유 데이터를 직접 나타내지는 않지만, 이 데이터를 액세스하거나 업데이트할 수 있습니다.
- EJB 서버 충돌 시에는 제거됩니다.
- 상태가 없습니다.
- 선택적으로 트랜잭션을 인식합니다.

EJB 컨테이너 구성

EJB 컨테이너에 대한 로그 수준을 구성할 수 있으며, 모니터링도 가능합니다. EJB 컨테이너는 EJB와 MDB를 처리합니다. 관리 인터페이스를 사용하여 컨테이너가 관리하는 EJB와 MDB에 대한 설정을 구성할 수 있습니다. 이 절에서는 다음 항목에 대해 설명합니다.

- 일반 구성 수행
- EJB 설정 구성
- MDB 풀 설정 구성

일반 구성 수행

EJB 컨테이너의 다음 부분을 구성할 수 있습니다.

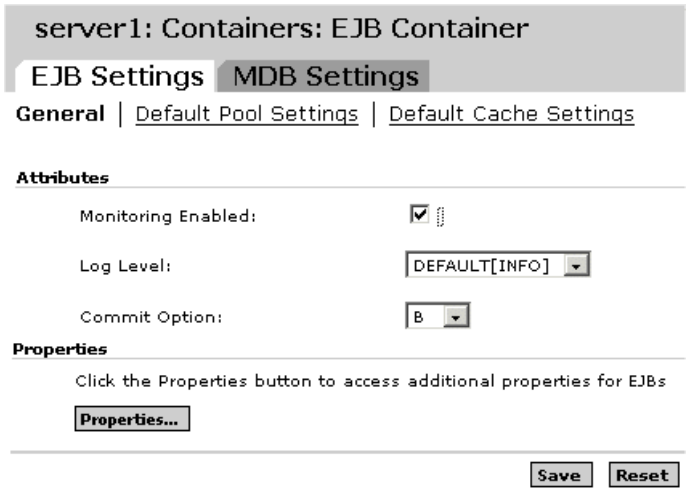
- 로깅
- 모니터링
- 트랜잭션 속성

모니터링이 가능하고 트랜잭션 속성을 설정할 수 있도록 EJB 컨테이너에 대한 로그 수준을 설정하려면 다음 작업을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 수정하려는 EJB 컨테이너 구성에 대한 Sun ONE Application Server 인스턴스 트리를 엽니다.

2. "컨테이너" 탭을 확장하여 J2EE 컨테이너 목록이 나타나면 이 목록에서 "EJB 컨테이너"를 선택합니다. 관리 인터페이스의 오른쪽 창에서 "EJB 컨테이너 - 일반 구성"이라고 표시된 창이 나타납니다.

EJB 컨테이너 - 일반 구성



3. EJB 컨테이너를 모니터링하려면 "모니터링 사용 가능"에 해당하는 확인란을 선택하십시오. 이제 이 특정한 Sun ONE Application Server 인스턴스의 EJB에 대한 모니터링이 가능합니다. 모니터가 가능한 EJB 컨테이너 부분에 대한 목록을 보려면 표 "EJB 컨테이너에 대한 모니터링 통계"를 참조하십시오.

4. "로그 수준" 드롭다운 목록에서 원하는 로그 수준을 선택합니다. 모든 로그 수준과 정의에 대한 목록은 5장 "로깅 사용"을 참조하십시오. 로그 수준 지정은 기록되는 메시지의 유형을 제어합니다. 예를 들어, 로그 수준 FATAL을 가진 메시지만 기록되도록 지정하면, 이 값보다 "높은" 로그 수준의 메시지는 자연스럽게 무시됩니다. 그리고 명시적 로그 수준으로 기록되는 메시지만 이 값과 비교됩니다.

   명시적 로그 수준을 갖지 않은 메시지는 무제한으로 기록됩니다. 기본 동작은 모든 경고, 오류 및 치명적 메시지를 로깅하는 것입니다.

5. "완결 옵션" 드롭다운 목록에서 EJB 컨테이너에 대해 사용할 "완결 옵션"을 선택합니다.

   트랜잭션은 두 가지 방법으로 끝낼 수 있습니다. 즉, 완결을 사용하거나 롤백을 사용할 수 있습니다. 트랜잭션을 완결하면 문을 사용한 데이터 수정이 저장됩니다. Enterprise Bean을 설계할 때는 완결이 컨테이너 관리 트랜잭션인지, 아니면 Bean 관리 트랜잭션인지를 결정합니다. 따라서 UI의 옵션은 Bean 관리 완결의 경우 B이며, 컨테이너 관리 완결의 경우 C입니다.

6. "등록 정보" 버튼을 눌러 EJB 컨테이너에 대한 새 등록 정보를 만듭니다.

7. 확인을 눌러 설정을 저장합니다.

다음 표는 모니터할 수 있는 EJB 컨테이너에 대한 속성을 나타냅니다.

EJB 컨테이너에 대한 모니터링 통계

통계적 이름	데이터 유형 및 단위	값 범위	설명
minBeansInPool	정수	0-MAXINT	적정한 풀의 최소 Bean 수(Stateless Session Bean에 적용)
initialBeansInPool	정수	0-MAXINT	풀의 초기 Bean 수(Stateless Session Bean에 적용)
maxBeansInPool	정수	0-MAXINT	풀의 최대 Bean 수 (Stateless Session Bean에 적용)
beanIdleTimeoutInSeconds	정수	0-MAXLONG	Bean이 삭제되는 유휴 시간 초과(초)
numBeansCreated	정수	0-MAXINT	지금까지 만든 Bean의 수
numBeansDestroyed	정수	0-MAXINT	지금까지 삭제된 Bean의 수
numThreadsWaitaing	정수	0-MAXINT	사용 가능한 Bean을 기다리는 스레드 수
numBeansInPool	정수	0-MAXINT	풀에서 사용 가능한 Bean의 수(이 수가 0보다 크면 ThreadsWaitaing은 0이 되어야 함)
maxBeansInCache	정수	0-MAXINT	캐시의 최대 Bean 수(Entity Bean 및 Stateful Bean에 적용)
minBeansInCache	정수	0-MAXINT	캐시의 적정한 최소 Bean 수(Entity Bean 및 Stateful Bean에 적용)
cacheFaultsPercentage	Double		그 결과 백업 저장소에서 활성화되는 캐시 실패의 수

EJB 설정 구성

관리 인터페이스를 사용하여 EJB 컨테이너에서 관리되는 EJB에 대한 기본 풀과 Bean 캐시 설정을 구성할 수 있습니다. 이와 관련한 내용은 다음 항목에서 설명합니다.

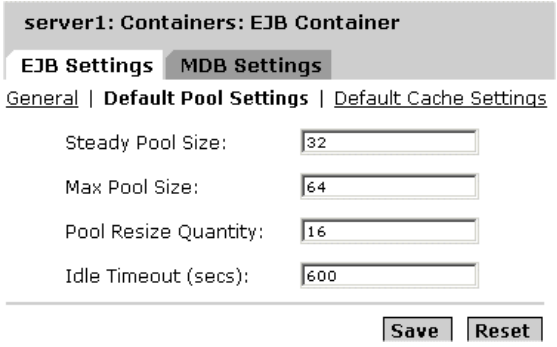
- EJB 풀 설정 구성
- EJB 캐시 설정 구성

EJB 풀 설정 구성

EJB 풀 설정을 구성하려면 다음 작업을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 수정할 EJB 설정의 Sun ONE Application Server 인스턴스 트리를 엽니다.
2. "컨테이너" 탭을 확장하여 J2EE 컨테이너 목록이 나타나면 이 목록에서 "EJB 컨테이너"를 선택합니다. 관리 인터페이스의 오른쪽 창에서 "EJB 풀 설정 구성"이라고 표시된 창이 나타납니다.

EJB 풀 설정 구성



3. "고정 풀 크기" 필드에서 풀의 최소 Bean 수를 지정합니다. 이 수는 Stateless Session Bean에 적용됩니다.
4. "최대 풀 크기" 드롭다운 목록에서 지정된 시간 내에 원하는 풀의 최대 Bean 수를 지정합니다. 이 설정은 Stateless Session Bean에 적용됩니다.
5. idle-timeout-in-seconds 태그에서 지정한 시간을 초과해서 Bean이 계속 유휴 상태로 있는 경우, 풀에서 제거할 Bean의 수를 "풀 크기 조정 개수" 필드에 지정합니다.

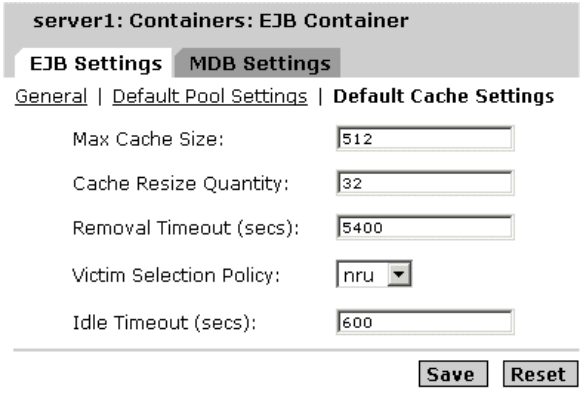
6. "유휴 시간 초과(초)" 필드에서 Bean이 유휴 상태가 될 수 있는 시간(초)을 지정합니다. 유휴 시간 초과 기간이 경과해도 여전히 Bean이 유휴이면 삭제됩니다.
7. "저장"을 눌러 변경 사항을 저장합니다.

EJB 캐시 설정 구성

EJB 캐시 설정을 구성하려면 다음 작업을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 수정할 EJB 설정의 Sun ONE Application Server 인스턴스 트리를 엽니다.
2. "컨테이너" 탭을 확장하여 J2EE 컨테이너 목록이 나타나면 이 목록에서 "EJB 컨테이너"를 선택합니다. 관리 인터페이스의 오른쪽 창에서 "EJB 풀 설정 구성"이라고 표시된 창이 나타납니다.

EJB 캐시 설정 구성



3. `Max Cache Size` 필드에서, 캐시에서 유지하려는 최대 Bean의 수를 지정합니다. 이 속성에 대한 기본값은 `idle-timeout-seconds` 속성에서 지정한 것과 같습니다.
4. `Cache Resize Quantity` 필드에서, 풀의 Bean 수가 `Max Cache Size` 속성에서 지정한 개수를 초과할 경우 삭제할 Bean의 개수를 지정합니다.

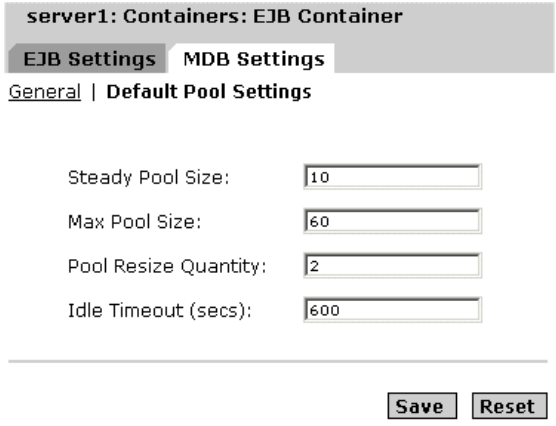
5. 백업 저장소에서 유휴인 Bean이 비활성화 상태를 유지할 수 있는 시간을 `Removal Timeout` (secs) 필드에 지정합니다. `removal-timeout-in-seconds` 속성에서 지정한 값을 넘어 클라이언트가 Bean에 액세스하지 않는 경우, Bean은 백업 저장소에서 제거되므로 클라이언트에서 액세스할 수 없게 됩니다.
6. `Victim Selection Policy` 드롭다운 목록에서, 풀에서 제거할 희생자 Bean을 선택하기 위해 사용해야 하는 희생자 선택 알고리즘을 선택합니다.
7. `Idle Timeout` (secs) 필드에서 캐시에서 유휴하도록 허용된 Bean에 대한 시간을 지정합니다. 이 시간이 경과되면 Bean이 비활성화됩니다. Bean이 비활성화 상태에 있는 시간(유휴 백업 저장소 내)은 `removal-timeout-in-seconds` 매개 변수에 의해 제어됩니다.
8. "저장"을 눌러 변경 사항을 저장합니다.

MDB 풀 설정 구성

관리 인터페이스를 사용하여 EJB 컨테이너에서 관리되는 MDB에 대한 기본 풀 설정을 구성할 수 있습니다. MDB에 대한 기본 풀 설정을 구성하려면, 다음 작업을 수행하십시오.

1. 관리 인터페이스의 왼쪽 창에서 수정하려는 MDB 컨테이너 구성의 Sun ONE Application Server 인스턴스 트리를 엽니다.
2. "컨테이너" 탭을 확장하여 J2EE 컨테이너 목록이 나타나면 이 목록에서 "EJB 컨테이너"를 선택합니다. 관리 인터페이스의 오른쪽 창에서 "MDB 풀 설정 구성"이라고 표시된 창이 나타납니다.

MDB 풀 설정 구성



3. MDB 설정을 누릅니다. `Steady Pool Size` 텍스트 필드에서 풀의 최대 Bean 수를 지정합니다. 이 수는 Stateless Session Bean에 적용됩니다.
4. `Max Pool Size` 필드에서 지정된 시간 내에 원하는 풀의 최대 Bean 수를 지정합니다.
5. `idle-timeout-in-seconds` 태그에서 지정한 시간을 초과해서 Bean이 계속 유휴 상태로 있는 경우 풀에서 제거할 Bean의 수를 "풀 크기 조정 개수" 필드에 지정합니다.
6. `Idle Timeout` (secs) 필드에서 Bean이 유휴 상태를 유지할 수 있는 시간(초)을 지정합니다. 유휴 시간 초과 기간이 경과해도 Bean이 여전히 유휴 상태에 있으면 삭제됩니다.
7. "저장"을 눌러 해당 설정을 저장합니다.

9장

트랜잭션 서비스 사용

트랜잭션은 비즈니스의 통합된 일부입니다. 일반 비즈니스 트랜잭션은 두 개 이상의 관련 부분 사이 자산 이동과 관련되어 있습니다. 정확성을 요하는 기록은 보통 하나 이상의 데이터베이스에 저장됩니다. 이 정보는 비즈니스 작업에 꼭 필요하기 때문에 항상 유효하고, 현재성을 가지고, 신뢰할 수 있어야 합니다. 초보 프로그래머에게는 트랜잭션 처리가 어려울 수 있습니다. J2EE 플랫폼은 종속 가능한 트랜잭션 처리 응용 프로그램 전개를 쉽게 해주는 여러 추상적 개념을 제공합니다. 이 장에서는 Sun ONE Application Server의 J2EE 트랜잭션 및 트랜잭션 지원을 설명합니다.

이 장에서는 일반적으로 Java 트랜잭션, 구체적으로는 Sun ONE Application Server에 포함되어 있는 트랜잭션 지원에 대해 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- 트랜잭션 정보
- J2EE 트랜잭션
- 트랜잭션 자원 관리자
- 로컬 및 분산 트랜잭션
- 컨테이너 관리 트랜잭션
- Bean 관리 트랜잭션
- 트랜잭션 서비스 관리

트랜잭션 정보

비즈니스 트랜잭션을 에뮬레이트하려면 프로그램이 여러 단계를 수행해야 할 수 있습니다. 예를 들어, 재무 프로그램이 다음 의사 코드에 나열된 단계를 수행하면 자금을 수표 계좌에서 예금 계좌로 이체할 수 있습니다.

```
begin transaction
debit checking account
credit savings account
update history log
commit transaction
```

앞의 의사 코드에서 `begin` 및 `commit` 명령은 트랜잭션 경계를 표시합니다. 트랜잭션을 완료하려면 세 단계가 모두 완료되어야 합니다. 세 단계가 모두 완료되지 못하면 데이터 통합이 손상될 수 있습니다.

이 보장을 원자성이라고 합니다. 트랜잭션은 `commit` 또는 `rollback` 명령으로 끝납니다. 트랜잭션을 완결하면 트랜잭션 경계 내의 명령으로 수행한 모든 수정이 저장 및 지속됩니다. 변경은 *영구적*이며 이후 시스템 장애가 일어나도 그대로 유지됩니다. 트랜잭션 내의 명령 중 하나라도 실패하면 트랜잭션 전체가 롤백하며, 그 때까지 트랜잭션 내에서 실행된 모든 명령은 취소됩니다. 예를 들어, 의사 코드의 credit 단계 중 디스크 드라이브에 오류가 발생하면, 트랜잭션은 롤백되고 debit 명령으로 수행되었던 데이터 변경도 모두 취소됩니다.

트랜잭션이 실패해도 트랜잭션 계정 잔액은 여전히 유지되기 때문에 데이터 통합이 손상되지 않습니다. 트랜잭션 작동의 이런 측면을 트랜잭션 *일관성*이라 합니다.

트랜잭션 서비스는 *고립화*도 제공합니다. 이것은 트랜잭션이 완결 또는 롤백되기 전까지는 트랜잭션 내 구문을 다른 응용 프로그램에서 관찰할 수 없음을 의미합니다. 일단 트랜잭션이 완결되면 이 완결된 트랜잭션은 응용 프로그램 및 스레드가 안전하게 관찰할 수 있습니다.

J2EE 트랜잭션

J2EE의 트랜잭션 처리는 트랜잭션 관리자, 응용 프로그램 서버, 자원 관리자, 자원 어댑터 및 사용자 응용 프로그램의 다섯 참가자에 의해 수행됩니다. 각 엔티티는 다음에서 설명하는 여러 API 및 기능을 구현하여 신뢰할 수 있는 방법으로 트랜잭션을 처리합니다.

- 트랜잭션 관리자는 트랜잭션 구분, 트랜잭션 자원 관리, 동기화 및 트랜잭션 컨텍스트 전파 지원에 필요한 서비스 및 관리 기능을 제공합니다.
- 응용 프로그램 서버는 트랜잭션 상태 관리 등의 응용 프로그램 런타임 환경 지원에 필요한 기반 구조를 제공합니다.
- 자원 관리자(자원 어댑터를 통한)는 응용 프로그램에게 자원에 대한 액세스를 제공합니다. 자원 관리자는 트랜잭션 관리자가 트랜잭션 연결, 트랜잭션 완료 및 복구 작업과 통신할 때 사용하는 트랜잭션 자원 인터페이스를 구현하여 분산된 트랜잭션에 참여합니다. 이런 자원 관리자의 예로 관계형 데이터베이스 서버를 들 수 있습니다.
- 자원 어댑터는 응용 프로그램 서버 또는 클라이언트가 자원 관리자와의 연결에 사용하는 시스템 수준 소프트웨어 라이브러리입니다. 자원 어댑터는 일반적으로 자원 관리자마다 따로 지정됩니다. 자원 어댑터는 라이브러리로 사용 가능하며 이것을 사용하는 클라이언트 주소 공간 내에서 사용됩니다. 이런 자원 어댑터의 예로는 JDBC 드라이버가 있습니다.
- J2EE 응용 프로그램 서버 환경에서 작업하기 위해 개발된 트랜잭션 사용자 응용 프로그램은 JNDI를 사용하여 트랜잭션 데이터 소스 및 선택적으로 트랜잭션 관리자를 조회합니다. EJB의 선언적 트랜잭션 속성 설정 또는 명시적인 프로그램형 트랜잭션 구분을 사용할 수 있습니다.

자원 관리자와 자원 어댑터 엔티티는 서로 밀접하게 연관되어 있어서, 자원 관리자라는 용어가 자원 어댑터와 혼용되는 경우도 많습니다.

트랜잭션 자원 관리자

다음 트랜잭션 자원 관리자는 J2EE 트랜잭션 내에서 지원됩니다.

- 데이터베이스
- JMS 공급자
- J2EE 커넥터

데이터베이스

가장 자주 접하게 되는 J2EE 응용 프로그램의 트랜잭션 자원 관리자는 데이터베이스입니다. JDBC는 J2EE 구성 요소가 데이터베이스 액세스에 사용하는 API입니다. 데이터베이스 자원은 JDBC 자원으로 구성됩니다. JDBC 자원은 자원 관리자 또는 JDBC 드라이버가 관리합니다. JDBC 드라이버는 로컬 트랜잭션 또는 전역 트랜잭션 및 때로 로컬 및 전역 트랜잭션 모두에 대한 지원을 제공할 수 있습니다.

Sun ONE Application Server는 여러 J2EE 구성 요소의 JDBC 및 트랜잭션 사용을 지원합니다. JDBC 자원의 등록 및 구성 방법에 대한 자세한 내용은 263페이지의 “JDBC 자원 정보”를 참조하십시오. 응용 프로그램 서버는 트랜잭션 연속성 제공(즉 여러 응용 프로그램 구성 요소로부터 트랜잭션 초기화 및 데이터베이스 액세스 수행)을 담당합니다. 예를 들어, 서블릿이 트랜잭션을 시작하고, 데이터베이스를 액세스하고, 동일한 트랜잭션의 일부로 동일한 데이터베이스를 액세스하는 Enterprise Bean을 불러낸 후, 마지막으로 트랜잭션을 완결할 수 있습니다.

JMS 공급자

JMS는 Java Message Service의 약자입니다. JMS 공급자는 메시지 브로커 서비스를 의미하는 J2EE 용어입니다. JMS API는 응용 프로그램 사이에 신뢰할 수 있는 트랜잭션 방식의 교환을 제공합니다. 트랜잭션 JMS 데이터 소스 지원은 J2EE의 필수 기능입니다. JMS 자원 및 JDBC 자원이 함께 동일한 트랜잭션에 사용될 수 있습니다.

Sun ONE Application Server는 Sun ONE 메시지 대기열, 완전한 기능의 JMS 공급자 및 해당 트랜잭션 자원 관리자에 통합됩니다. 이렇게 하면 Sun ONE Application Server가 서블릿, JSP 페이지 및 Enterprise Bean에서 트랜잭션 JMS에 액세스할 수 있습니다. Sun ONE Application Server에 타사 JMS 공급자를 사용할 수도 있습니다. 자세한 내용은 11장, "JMS 서비스 사용"을 참조하십시오.

J2EE 커넥터

Sun ONE Application Server는 트랜잭션 자원 관리자로 XATransaction 모드를 사용하는 자원 어댑터를 지원합니다. 플랫폼에서는 서블릿, JSP 페이지 및 Enterprise Bean에서 자원 어댑터로 트랜잭션 액세스할 수 있어야 합니다. 단일 트랜잭션 내의 여러 응용 프로그램 구성 요소에서 자원 어댑터에 액세스할 수 있습니다. 예를 들어, 서블릿이 트랜잭션을 시작하고, 자원 어댑터를 액세스하고, 동일한 트랜잭션의 일부로 자원 어댑터를 액세스하는 Enterprise Bean을 불러낸 후, 마지막으로 트랜잭션을 완결할 수 있습니다.

로컬 및 분산 트랜잭션

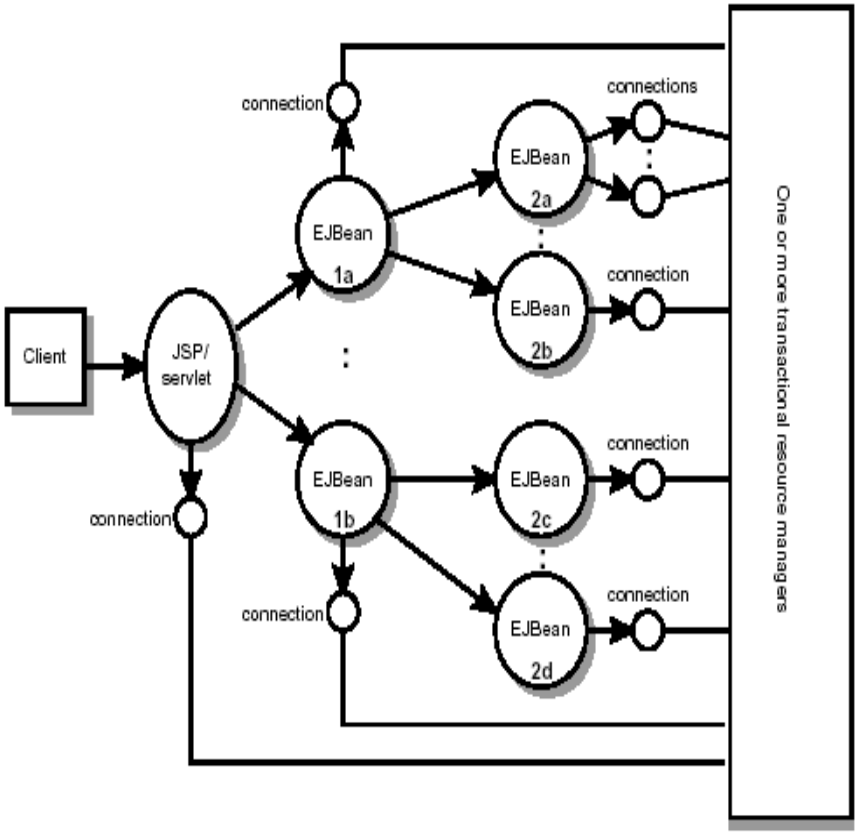
한 자원과만 관련 있는 트랜잭션은 로컬 트랜잭션을 사용하여 완료될 수 있습니다. 또한 로컬 트랜잭션은 모든 참여 응용 프로그램 구성 요소가 하나의 프로세스 내에서 실행되어야 합니다. 두 개 이상의 자원 또는 여러 참여 프로세스가 관련된 트랜잭션은 분산 또는 전역 트랜잭션이 될 수 있습니다. 로컬 트랜잭션 최적화는 최적화 전용 자원 관리자를 사용하며, 이것은 J2EE 응용 프로그램에 투명합니다.

트랜잭션 유형은 관련 자원 관리자에 구현된 인터페이스로 판단할 수 있습니다. 예를 들어, `javax.sql.DataSource` 인터페이스를 구현하는 JDBC 데이터 소스는 로컬 트랜잭션에 참여할 수 있습니다. `javax.sql.XADataSource`를 구현하는 데이터 소스는 전역 트랜잭션의 일부를 취할 수 있습니다. 일부 JDBC 자원은 두 인터페이스를 모두 구현하며, 이런 JDBC 자원이 Sun ONE Application Server에 등록되어 있으면 Sun ONE Application Server의 구성에 해당 자원의 선호 기능을 표시하는 구성 정보를 추가해야 할 수 있습니다.

로컬 트랜잭션이 더 간단하며 기본적으로 전역 트랜잭션보다 효율적입니다. 변형해야 할 데이터가 여러 데이터 소스에 있을 경우에는 로컬 트랜잭션이 적절하지 않습니다. 때로 몇 개의 데이터 소스가 트랜잭션에 나열되어야 하는지 예측할 수 없는 경우도 있습니다. 이 때문에 실제로는 전역 트랜잭션을 더 많이 사용합니다. 전역 트랜잭션에 사용할 수 있는 몇 가지 성능 개선 최적화가 있습니다.

J2EE는 트랜잭션 내의 여러 Enterprise Bean에 액세스하는 서블릿이나 JSP의 임의 조합을 모두 포함하는 트랜잭션 응용 프로그램을 지원합니다. 각 구성 요소는 하나 이상의 트랜잭션 자원 관리자에 대해 하나 이상의 연결을 가질 수 있습니다. 다음 그림에서, 호출 트리는 여러 Enterprise Bean을 액세스하는 서블릿 또는 JSP에서 시작하며, 이후 다른 Enterprise Bean을 액세스할 수 있습니다. 구성 요소는 연결을 통해 자원 관리자에 액세스합니다.

트랜잭션의 자원에 액세스하는 J2EE 구성 요소



예를 들어, 응용 프로그램은 위 그림의 모든 구성 요소가 단일 트랜잭션의 일부로 자원에 액세스해야 할 수도 있습니다. 응용 프로그램 서버 공급자는 이런 시나리오를 지원하는 트랜잭션 기능을 제공해야 합니다.

J2EE 트랜잭션 관리는 플랫 트랜잭션을 지원합니다. 플랫 트랜잭션은 자식(중첩된) 트랜잭션을 가질 수 없습니다.

트랜잭션 복구는 분산 트랜잭션에 있어 중요한 측면입니다. 크리티컬 포인트 동안 자원을 사용할 수 없거나 기타 복구할 수 없는 오류가 발생하면, 분산 트랜잭션의 상태를 모르게 될 수 있습니다. 꼬이거나 완료되지 못한 트랜잭션을 자동 및 수동으로 복구하는 것은 Sun ONE Application Server의 중요한 기능입니다. 자동 트랜잭션 복구는 관리 인터페이스를 사용하여 수행할 수 있습니다. 트랜잭션 복구 제어 방법의 자세한 내용은 233페이지의 “트랜잭션 서비스 관리”를 참조하십시오.

연결(여기에서는 자원과 동의어로 사용됨)은 공유 가능 여부가 표시될 수 있습니다. 공유 불가능한 방법으로 연결을 사용하려는 J2EE 응용 프로그램 구성 요소는 영향을 받는 측에 배포 정보를 제공하여 컨테이너가 연결을 공유하지 못하도록 해야 합니다. 이것이 필요할 수 있는 경우의 예로는 보안 속성, 고립화 수준, 문자 집합 및 로컬화 구성을 변경하는 경우가 포함됩니다.

컨테이너는 공유 불가능으로 표시된 연결을 공유하지 말아야 합니다. 연결이 공유 불가능으로 표시되어 있지 않으면, 이것은 연결이 실제 공유되어 있는지 여부와 상관없이 응용 프로그램에 투명해야 합니다.

J2EE 응용 프로그램 구성 요소는 선택적 전개 설명자 요소 `res-sharing-scope`를 사용하여 자원 관리자에 대한 연결의 공유 가능 여부를 표시할 수 있습니다. 컨테이너는 전개 관련 정보가 제공되지 않으면 공유 가능한 연결이라고 간주합니다. J2EE 응용 프로그램 구성 요소는 연결 객체를 캐시하여 여러 트랜잭션에서 다시 사용할 수 있습니다. 공유 연결을 제공하는 컨테이너는 이런 캐시된 연결 객체를 투명하게 전환하여(디스패치할 때) 올바른 트랜잭션 범위를 가진 적절한 공유 연결을 가리키도록 해야 합니다.

Enterprise Bean 응용 프로그램을 설계하는 경우, 개발자는 경계 지정 방법을 결정해야 합니다.

컨테이너 관리 트랜잭션

컨테이너 관리 트랜잭션이 있는 Enterprise Bean의 경우에는 EJB 컨테이너가 트랜잭션 경계를 설정합니다. Enterprise Bean 유형으로 Session, Entity 또는 Message-Driven을 가진 컨테이너 관리 트랜잭션을 사용할 수 있습니다. 컨테이너 관리 트랜잭션은 Enterprise Bean 코드가 트랜잭션의 경계를 명시적으로 표시하지 않기 때문에 간단하게 배포할 수 있습니다. 이 코드에는 트랜잭션 시작 및 종료 명령이 없습니다.

일반적으로 컨테이너는 Enterprise Bean 메소드가 시작하기 직전에 트랜잭션을 시작하고, 메소드가 종료되기 직전 트랜잭션을 완결합니다. 각 메소드는 단일 트랜잭션에 관련될 수 있습니다. 한 메소드 내에 중첩된 트랜잭션 또는 여러 트랜잭션을 허용하지 않습니다.

컨테이너 관리 트랜잭션이 트랜잭션에 연결될 때 모든 메소드가 필요하지는 않습니다. Bean을 배포할 때에는 트랜잭션 속성을 설정하여 트랜잭션과 관련된 Bean의 메소드를 지정합니다.

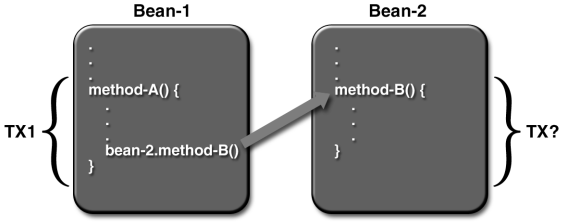
이 절에서는 다음 내용을 설명합니다.

- 트랜잭션 속성
- 트랜잭션 속성 설정
- 컨테이너 관리 트랜잭션 롤백
- Session Bean의 인스턴스 변수 동기화
- 컨테이너 관리 트랜잭션에 허용되지 않는 메소드

트랜잭션 속성

트랜잭션 속성은 트랜잭션 범위를 제어합니다. 다음 그림은 범위 제어가 왜 중요한지 보여주고 있습니다. 다이어그램에서, 메소드 A는 트랜잭션을 시작하고 Bean 2의 메소드 B는 트랜잭션을 불러냅니다. 메소드 B가 실행되면 메소드 A가 시작한 트랜잭션 범위 내에서 실행될까요? 아니면 새 트랜잭션으로 실행될까요? 답은 메소드 B의 트랜잭션 속성에 따라 달라집니다.

트랜잭션 속성



트랜잭션 속성은 다음 값 중 하나를 가질 수 있습니다.

- Required
- RequiresNew
- Mandatory
- NotSupported
- Supports
- Never

Required

클라이언트가 트랜잭션 내에서 실행 중일 때 Enterprise Bean의 메소드를 불러내면, 이 메소드는 클라이언트 트랜잭션 내에서 실행됩니다. 클라이언트가 트랜잭션과 관련되어 있지 않으면, 메소드가 실행되기 전 먼저 컨테이너가 새 트랜잭션을 시작합니다.

`Required` 속성은 대부분의 트랜잭션에 사용됩니다. 이 때문에 최소한 개발 초기에는 이것을 기본값으로 사용하는 것이 좋습니다. 트랜잭션 속성은 선언적이기 때문에 나중에 쉽게 변경할 수 있습니다.

RequiresNew

클라이언트가 트랜잭션 내에서 실행 중일 때 Enterprise Bean의 메소드를 불러내면, 컨테이너는 다음 단계를 수행합니다.

- 클라이언트의 트랜잭션 일시 중단
- 새 트랜잭션 시작
- 메소드에게 호출 위임
- 메소드 실행 완료 후 클라이언트의 트랜잭션 다시 시작

클라이언트가 트랜잭션과 관련되어 있지 않으면, 메소드가 실행되기 전 먼저 컨테이너가 새 트랜잭션을 시작합니다.

메소드가 새 트랜잭션 내에서 항상 실행되도록 하려면 `RequiresNew` 속성을 사용해야 합니다.

Mandatory

클라이언트가 트랜잭션 내에서 실행 중일 때 Enterprise Bean의 메소드를 불러내면, 이 메소드는 클라이언트 트랜잭션 내에서 실행됩니다. 클라이언트가 트랜잭션과 관련되어 있지 않으면 `TransactionRequiredException`이 발생합니다.

Enterprise Bean의 메소드가 클라이언트 트랜잭션을 사용해야 하는 경우 `Mandatory` 속성을 사용합니다.

NotSupported

클라이언트가 트랜잭션 내에서 실행 중일 때 Enterprise Bean의 메소드를 불러내면, 컨테이너는 이 메소드를 불러내기 전 먼저 클라이언트 트랜잭션을 일시 중단합니다. 메소드가 완료된 다음 컨테이너는 클라이언트 트랜잭션을 다시 시작합니다.

클라이언트가 트랜잭션과 관련되어 있지 않으면, 컨테이너는 메소드가 실행되기 전 새 트랜잭션을 시작하지 않습니다.

트랜잭션이 필요하지 않은 메소드에는 `NotSupported` 속성을 사용합니다. 트랜잭션 불러내기는 오버헤드가 있기 때문에 이 속성은 성능을 개선할 수 있습니다.

Supports

클라이언트가 트랜잭션 내에서 실행 중일 때 Enterprise Bean의 메소드를 불러내면, 이 메소드는 클라이언트 트랜잭션 내에서 실행됩니다. 클라이언트가 트랜잭션과 관련되어 있지 않으면, 컨테이너는 메소드가 실행되기 전 새 트랜잭션을 시작하지 않습니다.

메소드의 트랜잭션 작업은 다양하기 때문에 `Supports` 속성은 주의해서 사용해야 합니다.

Never

클라이언트가 트랜잭션 내에서 실행 중일 때 Enterprise Bean의 메소드를 불러내면, 컨테이너는 `RemoteException`를 발생합니다. 클라이언트가 트랜잭션과 관련되어 있지 않으면, 컨테이너는 메소드가 실행되기 전 새 트랜잭션을 시작하지 않습니다.

속성 요약

다음 표는 트랜잭션 속성의 영향을 요약합니다. T1 및 T2 트랜잭션은 모두 컨테이너로 제어됩니다. T1 트랜잭션은 Enterprise Bean의 메소드를 호출하는 클라이언트와 관련되어 있습니다. 대부분의 경우 클라이언트도 별도의 Enterprise Bean입니다. T2 트랜잭션은 메소드가 실행되기 전 컨테이너에 의해 시작됩니다.

마지막 열의 '없음'이라는 단어는 비즈니스 메소드가 컨테이너가 제어하는 트랜잭션 내에서 실행되지 않음을 표시합니다. 하지만 이런 비즈니스 메소드에 호출된 데이터베이스는 DBMS의 트랜잭션 관리자가 제어할 수 있습니다.

트랜잭션 속성

트랜잭션 속성	클라이언트 트랜잭션	비즈니스 메소드의 트랜잭션
Required	없음	T2
	T1	T1
RequiresNew	없음	T2
	T1	T2
Mandatory	없음	오류
	T1	T1

트랜잭션 속성		
NotSupported	없음	없음
	T1	없음
Supports	없음	없음
	T1	T1

트랜잭션 속성 설정

트랜잭션 속성은 배포 설명자에 저장되기 때문에 Enterprise Bean 작성, 응용 프로그램 어셈블리 및 배포 등 J2EE 응용 프로그램 전개자의 여러 단계 도중 변경될 수 있습니다. 하지만 Bean 작성시 속성을 지정하는 것은 개발자가 해야 합니다. 속성은 구성 요소를 더 큰 응용 프로그램으로 어셈블리하는 응용 프로그램 개발자만 수정할 수 있어야 합니다. J2EE 응용 프로그램을 배포하는 사람은 트랜잭션 속성을 지정할 책임이 없습니다.

전체 Enterprise Bean 또는 각 메소드에 대해 트랜잭션 속성을 지정할 수 있습니다. 한 속성은 메소드에 대해 지정하고 다른 속성은 Bean에 대해 지정한 경우에는 메소드에 대해 지정한 속성이 우선권을 갖습니다. 각 메소드에 대해 속성을 지정하는 경우 필요한 사항은 Bean 유형에 따라 달라집니다. Session Bean에는 비즈니스 메소드에 정의된 속성이 필요하지만, 생성 메소드에 대해서는 허용하지 않습니다. Entity Bean에는 비즈니스, 생성, 제거 및 검색 메소드에 대한 트랜잭션 속성이 필요합니다. Message-Driven Bean에는 `onMessage` 메소드에 대한 트랜잭션 속성(`Required` 또는 `NotSupported` 중 하나)이 필요합니다.

컨테이너 관리 트랜잭션 롤백

컨테이너 관리 트랜잭션의 롤백에는 두 가지 방법이 있습니다. 먼저 시스템 예외가 발생하면 컨테이너는 자동으로 트랜잭션을 롤백합니다. 두 번째로 `EJBContext` 인터페이스의 `setRollbackOnly` 메소드를 불러내서, Bean 메소드가 컨테이너에게 트랜잭션 롤백을 지시할 수 있습니다. Bean이 응용 프로그램 예외를 발생시키면 롤백이 자동으로 실행되지 않아도 `setRollbackOnly`를 호출하여 초기화할 수 있습니다.

다음 예에서 `BankEJB` 예의 `transferToSaving` 메소드는 `setRollbackOnly` 메소드를 나타내고 있습니다. 수표 계정의 잔액이 마이너스가 되면, `transferToSaving`은 `setRollBackOnly`를 불러내고 응용 프로그램 예외(`InsufficientBalanceException`)를 발생시킵니다. `updateChecking` 및 `updateSaving` 메소드는 데이터베이스 테이블을 업데이트합니다. 업데이트에 실패하면 메소드는 `SQLException`을 발생시키고 `transferToSaving` 메소드는 `EJBException`을 발생시킵니다.

EJBException은 시스템 예외기 때문에, 컨테이너는 자동으로 트랜잭션을 롤백합니다. 다음은 transferToSaving 메소드의 코드입니다.

```
public void transferToSaving(double amount) throws
   InsufficientBalanceException  {

   checkingBalance -= amount;
   savingBalance += amount;

if (checkingBalance < 0.00) {
context.setRollbackOnly();

throw new InsufficientBalanceException();
}
   try {
      updateChecking(checkingBalance);

   updateSaving(savingBalance);
   } catch (SQLException ex) {
      throw new EJBException
         ("Transaction failed due to SQLException: "
         + ex.getMessage());
   }
}
```

컨테이너가 트랜잭션을 롤백하는 경우, 이것은 항상 트랜잭션 내의 SQL 호출로 발생한 데이터 변경을 모두 실행 취소합니다. 하지만 컨테이너는 Entity Bean에서만 인스턴스 변수에 대한 변경을 취소합니다(이 작업은 Entity Bean의 ejbLoad 메소드를 자동으로 호출하여 수행하며, 데이터베이스에서 인스턴스 변수가 로드됨). 롤백이 발생하면 Session Bean은 트랜잭션 내에 변경된 인스턴스 변수를 명시적으로 다시 설정합니다. Session Bean 인스턴스 변수를 다시 설정하는 가장 쉬운 방법은 SessionSynchronization 인터페이스를 구현하는 것입니다.

트랜잭션 ID를 명령줄 인터페이스를 통해 전달하여 트랜잭션을 롤백할 수도 있습니다. 자세한 내용은 236페이지의 “명령줄 인터페이스를 사용한 트랜잭션 관리”를 참조하십시오.

Session Bean의 인스턴스 변수 동기화

SessionSynchronization 인스턴스(선택적)를 사용하면 인스턴스 변수를 데이터베이스 내의 대응하는 값과 동기화할 수 있습니다. 컨테이너는 트랜잭션의 각 주요 단계에서 SessionSynchronization 메소드(afterBegin, beforeCompletion 및 afterCompletion)를 불러냅니다.

afterBegin 메소드는 새 트랜잭션이 시작되었음을 인스턴스에게 알립니다. 컨테이너는 처음으로 트랜잭션 내의 비즈니스 메소드를 불러내기 전에 먼저 afterBegin을 불러냅니다. afterBegin 메소드는 데이터베이스에서 인스턴스 변수를 로드하기에 좋은 위치입니다. 예를 들어, BankBean 클래스는 afterBegin 메소드 내에 checkingBalance 및 savingBalance 변수를 로드합니다.

```
public void afterBegin() {

    System.out.println("afterBegin()");
    try {
        checkingBalance = selectChecking();
        savingBalance = selectSaving();
    } catch (SQLException ex) {
        throw new EJBException("afterBegin Exception: " +
            ex.getMessage());
    }
}
```

컨테이너는 비즈니스 메소드는 완료되었지만 트랜잭션이 완결되기 전에 beforeCompletion 메소드를 불러냅니다. beforeCompletion 메소드는 Session Bean이 트랜잭션을 롤백할 수 있는(setRollbackOnly를 호출하여) 마지막 장소입니다. 아직 인스턴스의 변수 값을 데이터베이스에 업데이트하지 않았다면, Session Bean은 이 작업을 beforeCompletion 메소드에서 수행합니다.

`afterCompletion` 메소드는 트랜잭션이 완료되었음을 표시합니다. 여기에는 boolean 값의 매개 변수 하나가 있으며, 트랜잭션이 완결되면 true, 트랜잭션이 롤백되면 false 값을 갖습니다. 롤백이 발생하면 Session Bean은 `afterCompletion` 메소드에서 데이터베이스의 인스턴스 변수를 새로 고칠 수 있습니다.

```
public void afterCompletion(boolean committed) {

   System.out.println("afterCompletion: " + committed);
   if (committed == false) {
      try {
         checkingBalance = selectChecking();
         savingBalance = selectSaving();
} catch (SQLException ex) {
         throw new EJBException("afterCompletion SQLException:
         " + ex.getMessage());
}
}
}
```

컨테이너 관리 트랜잭션에 허용되지 않는 메소드

컨테이너가 설정한 트랜잭션 경계를 간섭할 수 있는 메소드는 모두 불러낼 수 없습니다. 금지되는 메소드 목록은 다음과 같습니다.

- `java.sql.Connection`의 완결, `setAutoCommit` 및 롤백 메소드.
- `javax.ejb.EJBContext`의 `getUserTransaction` 메소드.
- `javax.transaction.UserTransaction`의 모든 메소드.

하지만 Bean 관리 트랜잭션에는 이 메소드들을 사용하여 경계를 설정할 수 있습니다.

Bean 관리 트랜잭션

Bean 관리 트랜잭션의 경우, Session 또는 Message-Driven Bean 내의 코드에서 명시적으로 트랜잭션 경계를 표시합니다. Entity Bean은 Bean 관리 트랜잭션을 가질 수 없으며 대신 컨테이너 관리 트랜잭션을 사용해야 합니다. 컨테이너 관리 트랜잭션을 가진 Bean은 필요한 코드가 줄어들지만 한 가지 제한을 갖습니다. 메소드가 실행되면 Bean은 단일 트랜잭션에 관련되거나 아무런 트랜잭션에도 관련되지 않을 수 있습니다. 이 제한 때문에 Bean 코딩이 어려워지는 경우, Bean 관리 트랜잭션 사용을 고려해야 합니다.

다음 의사 코드는 Bean 관리 트랜잭션으로 구할 수 있는 세분화된 제어 종류를 표시합니다. 의사 코드는 여러 조건을 확인하여 비즈니스 메소드 내에서 여러 트랜잭션의 시작 또는 중지 여부를 결정합니다.

```
begin transaction
...
update table-a
...
if (condition-x)
    commit transaction
else if (condition-y)
    update table-b
    commit transaction
else
    rollback transaction
    begin transaction
    update table-c
    commit transaction
```

트랜잭션 서비스 관리

관리 인터페이스 또는 명령줄 인터페이스를 사용하여 트랜잭션을 관리할 수 있습니다.

이 절에는 다음 주제가 포함됩니다.

- 관리 인터페이스를 사용한 트랜잭션 관리
- 명령줄 인터페이스를 사용한 트랜잭션 관리

관리 인터페이스를 사용한 트랜잭션 관리

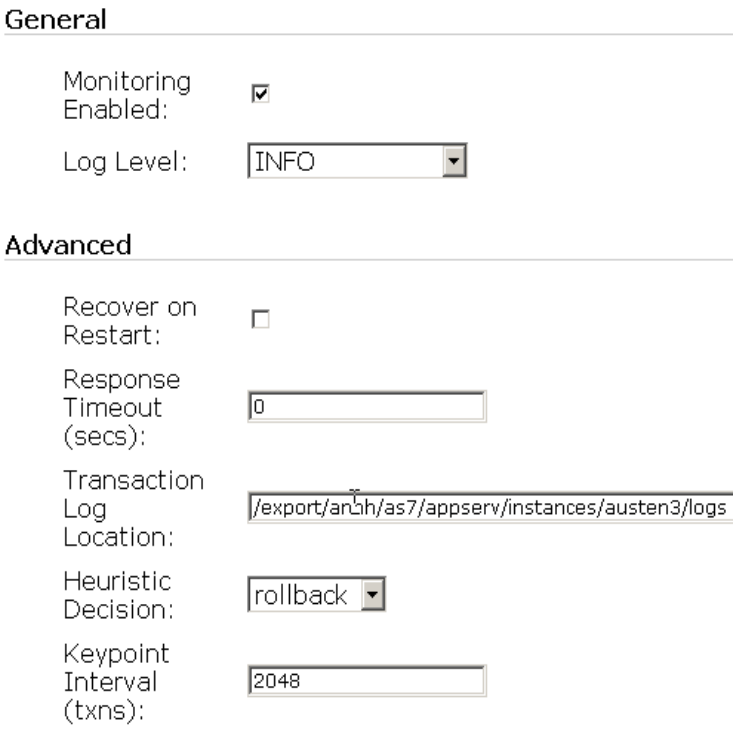
관리 인터페이스를 사용하여 트랜잭션 모니터링, 로그 수준 설정, 고급 옵션 지정 등을 수행할 수 있습니다.

복구 정책 및 시간 초과와 같은 인스턴스 범위의 트랜잭션 서비스 속성을 제어할 수 있습니다. 여기에서 지정하는 등록 정보 및 구성은 `server.xml` 파일에 저장됩니다.

트랜잭션 서비스 옵션을 구성하려면 다음 작업을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 트랜잭션 구성을 수정할 Sun ONE Application Server 인스턴스 트리를 엽니다.
2. 배포된 J2EE 서비스에서 트랜잭션 서비스를 선택합니다. 관리 인터페이스 오른쪽 창의 그림 "트랜잭션 서비스 구성 옵션"에 표시된 다음 창이 나타납니다.

트랜잭션 서비스 구성 옵션



3. 트랜잭션에 대한 모니터링을 활성화하려면 "모니터링 활성화" 확인란을 표시합니다. 다음 표는 모니터링될 수 있는 Java Transaction Service 기능을 나열합니다.

Java Transaction Service의 모니터링 가능한 속성

등록 정보	Type	설명
transactionsCompleted	int	모니터링이 활성화된 후 완료된 트랜잭션 개수
transactionsRolledBack	int	모니터링이 활성화된 후 롤백된 트랜잭션 개수
transactionsRecovered	int	모니터링이 활성화된 후 복구된 트랜잭션 개수
transactionsInFlight	int	현재 처리 중인 트랜잭션 개수
timeStamp	long	통계가 작성된 타임스탬프(밀리초 단위). 이것은 모두 `System.getCurrentTimeInMillis()`에서 보고합니다.

4. "로그 수준" 드롭다운 목록에서 트랜잭션에 설정할 로그 수준을 선택합니다. 로그 수준 및 통합 방법의 자세한 내용은 5장, "로깅 사용"을 참조하십시오.
5. 서버 재시작시 실패한 트랜잭션을 자동으로 복구하려면 "재시작시 복구" 확인란을 표시합니다. 트랜잭션 완결 프로토콜의 크리티컬 포인트 동안 자원을 사용할 수 없으면, 트랜잭션은 완료되지 못하고 트랜잭션 로그 파일에 남을 수 있습니다. 이 확인란이 표시되어 있으면 서버는 서버 재시작시 오류가 발생한 트랜잭션 복구를 시도합니다. 관련 자원에 여전히 접근할 수 없으면 서버 재시작 시간이 지연될 수 있습니다. 이 확인란은 기본적으로 표시되지 않습니다.
6. 컨테이너 관리 트랜잭션이 있는 Enterprise Bean의 경우, 트랜잭션 시간 초과(초) 등록 정보의 값을 설정하여 트랜잭션 시간 초과 간격을 제어할 수 있습니다.

   이 등록 정보의 값이 0으로 설정되면 트랜잭션은 시간 초과되지 않습니다.

   "트랜잭션 시간 초과(초)" 필드에 트랜잭션 시간 초과 간격을 지정합니다. 트랜잭션이 지정된 시간 내에 완료되지 않으면 트랜잭션은 롤백됩니다. 이 속성의 값이 0으로 설정되면 트랜잭션은 시간 초과되지 않습니다.

7. "트랜잭션 로그 위치" 필드에 로그 파일을 저장할 디렉토리의 절대 경로를 지정합니다. 새로 지정한 트랜잭션 로그 디렉토리를 적용하려면 서버를 재시작해야 합니다.
8. "발견적 결정" 드롭다운 상자에서 트랜잭션에 적용할 발견적 결정을 선택합니다. 명백한 결과를 결정할 수 없는 경우 표시된 옵션에서 완결 또는 롤백을 선택하여 응용 프로그램 서버에서 복구 도중 의심스러운 트랜잭션의 결과를 결정하는 방법을 지정합니다. 발견적 결정을 롤백으로 설정하면 트랜잭션을 롤백합니다. 이런 트랜잭션을 완결하는 것이 허용되는 경우도 있습니다.
9. "키 포인트 간격(트랜잭션)" 필드에 로그의 키 포인트 작업 사이의 트랜잭션 개수를 지정합니다. 키 포인트 작업은 완료된 트랜잭션의 항목을 제거하고 파일을 압축하여 트랜잭션 로그 파일의 크기를 줄입니다. 이 속성 값이 커지면 트랜잭션 로그 파일이 커지지만, 키 포인트 작업이 줄어들어 성능이 좋아질 수 있습니다. 이 값이 작아지면(예를 들어, 100) 로그 파일이 작아지지만, 키 포인트 작업 빈도가 높아져서 성능이 약간 낮아질 수 있습니다.

명령줄 인터페이스를 사용한 트랜잭션 관리

다음 절에서 설명한 것처럼, CLI (Command Line Interface)를 사용하여 데이터베이스 트랜잭션을 관리 및 모니터링할 수 있습니다.

- 실행 중인 트랜잭션 나열
- 트랜잭션 관리
- 트랜잭션 서비스 고정
- 트랜잭션 모니터링

이 절은 명령줄 인터페이스를 사용한 트랜잭션 관리 및 모니터링 방법을 설명합니다.

실행 중인 트랜잭션 나열

다음 명령은 실행 중인 트랜잭션 데이터를 구할 때 사용됩니다(multimode에 있고 이미 사용자 이름 및 암호를 설정했다고 가정합니다).

```
- asadmin> get --monitor
<instanceName>.transaction-service.inflight-tx
```

다중 행 출력은 다음과 같습니다.

```
Transaction Id State Elapsed Time (ms)

txnid1 Prepared 20

txnid2 Active 100

txnid3 Active 120

  ... ... ...
```

트랜잭션 관리

실행 중인 트랜잭션 나열의 예에서는 트랜잭션 ID `txn-ids`, `txnid2` 및 `txnid3`로 트랜잭션을 롤백한다고 가정합니다. 선택한 트랜잭션을 롤백하는 샘플 명령은 다음의 예와 같습니다.

```
asadmin> set --monitor
<instanceName>.transaction-service.rollback-list=txnid2,txnid3
```

트랜잭션 서비스 고정

트랜잭션 서비스를 고정하려면 다음 명령을 실행합니다.

```
asadmin> set --monitor
<instanceName>.transaction-service.freeze=true
```

트랜잭션 서비스가 고정되면 응용 프로그램 서비스 내의 트랜잭션 관리자는 모든 실행 중인 트랜잭션이 일시 중단됩니다. 고정은 제품 배포 시스템에서는 권장하지 않습니다.

트랜잭션 서비스를 고정 해제하려면 다음 명령을 실행합니다.

```
asadmin> set --monitor
<instanceName>.transaction-service.freeze=false
```

트랜잭션 서비스가 다시 동작 중으로 설정되면 시스템은 중단되었던 지점부터 다시 계속합니다. 활성 시스템을 너무 오래 고정 상태로 두면 일부 데이터베이스 연결이 시간 초과되어 트랜잭션이 롤백될 수 있습니다.

트랜잭션 모니터링

실행 중인 트랜잭션 데이터를 포함하여 트랜잭션 데이터를 모니터링하려면 다음 명령을 실행합니다.

```
asadmin> get --monitor <instanceName>.transaction-service.*
```

명령을 실행했을 때 활성 트랜잭션이 없으면 다음 출력을 구할 수 있습니다.

```
total-tx-completed = 5
total-tx-rolledback = 2
total-tx-inflight = 0
isFrozen = false
tx-inflight = No active transactions found.
```

명령을 실행했을 때 활성 트랜잭션이 있으면 다음 출력을 구할 수 있습니다.

```
total-tx-completed = 5
total-tx-rolledback = 2
total-tx-inflight = 2
isFrozen = false
tx-inflight =
Transaction Id State Elapsed Time(ms)
txnid1 Prepared 500
txnid2 Active 360
```

이름 지정 및 자원 구성

이 장에서는 Sun ONE Application Server에서 사용하는 J2EE 자원에 대해 설명하고 이러한 자원을 만들고 관리하기 위해 사용하는 방법에 대해서도 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- J2EE 이름 지정 서비스 및 자원 정보
- JNDI (Java Naming and Directory Interface) 정보
- 지속성 관리자 자원 정보
- JDBC 자원 정보
- JavaMail 자원 정보

J2EE 이름 지정 서비스 및 자원 정보

EJB를 포함한 J2EE 응용 프로그램, 웹 응용 프로그램 구성 요소 및 응용 프로그램 클라이언트는 자원 관리자, 데이터 소스(예: SQL 데이터 소스), 연결 팩토리, 우편 세션, Java Message Service 대상 객체, URL 연결 팩토리 같은 다양한 자원에 액세스합니다. J2EE 플랫폼은 JNDI (Java Naming and Directory) 이름 지정 서비스를 통해 이러한 자원을 응용 프로그램에 제공합니다.

Sun ONE Application Server를 사용하면 다음 J2EE 자원을 만들고 관리할 수 있습니다.

- JDBC 데이터 소스
- JavaMail 세션
- JMS 대상

JDBC 데이터 소스

JDBC 데이터 소스는 Sun ONE Application Server를 사용하여 만들고 관리할 수 있는 J2EE 자원입니다.

JDBC API는 관계형 데이터베이스 시스템과의 연결을 위한 API입니다. JDBC API는 두 부분으로 구성되어 있습니다.

- 데이터베이스에 액세스하기 위해 응용 프로그램 구성 요소에서 사용하는 응용 프로그램 수준 인터페이스
- JDBC 드라이버를 J2EE 플랫폼에 연결하기 위한 서비스 공급자 인터페이스

JDBC `DataSource` 객체는 Java 프로그래밍 언어로 데이터 소스를 표현하는 것입니다. 기본적으로 데이터 소스란 데이터를 저장하는 기능을 의미합니다. 데이터 소스는 대기업에서 사용하는 복잡한 데이터베이스처럼 복잡할 수도 있고 행과 열로 구성된 파일처럼 간단할 수도 있습니다. JDBC 데이터 소스는 Sun ONE Application Server를 통해 만들고 관리할 수 있는 J2EE 자원입니다.

JDBC 데이터 소스에 대한 자세한 내용은 263페이지의 "JDBC 자원 정보"를 참조하십시오.

JavaMail 세션

JMS 대상은 Sun ONE Application Server를 통해 만들고 관리할 수 있는 J2EE 자원입니다.

많은 인터넷 응용 프로그램에서 전자 우편 알림을 보낼 수 있는 기능을 필요로 하므로 J2EE 플랫폼은 JavaMail API와 함께 JavaMail 서비스 공급자를 포함하여 응용 프로그램 구성 요소가 인터넷 우편을 보낼 수 있도록 합니다. JavaMail API는 두 부분으로 구성되어 있습니다.

- 우편을 보내기 위해 응용 프로그램 구성 요소에서 사용하는 응용 프로그램 수준 인터페이스
- J2EE SPI 수준에서 사용하는 서비스 공급자 인터페이스

JavaMail 세션은 Sun ONE Application Server를 통해 만들고 관리할 수 있는 J2EE 자원입니다.

JavaMail 세션에 대한 자세한 내용은 285페이지의 "JavaMail 자원 정보"를 참조하십시오.

JMS 대상

JMS (Java Messaging Service)는 게시-가입 모델과 신뢰할 수 있는 지점간 메시징을 지원하는 메시징용 표준 API입니다. 이 사양에는 지점간 메시징과 게시-가입 메시징을 모두 구현하는 JMS 공급자가 필요합니다.

JMS는 관리 대상 객체의 두 가지 일반 유형, 연결 팩토리와 대상을 제공합니다. 두 가지 모두 공급자 관련 정보를 캡슐화하지만 JMS 클라이언트 내에서 아주 다르게 사용됩니다. 연결 팩토리는 메시지 서버를 연결하는 데 사용되며 대상 객체는 JMS 메시징 서비스에서 사용하는 물리적 대상을 식별하는 데 사용됩니다.

JNDI (Java Naming and Directory Interface) 정보

이 절에서는 JNDI (Java Naming and Directory Interface)에 대해 설명합니다. JNDI는 다양한 종류의 이름 지정 및 디렉토리 서비스를 액세스하기 위한 응용 프로그램 프로그래밍 인터페이스(API)입니다. J2EE 구성 요소는 JNDI 조회 메소드를 호출하여 객체를 찾습니다.

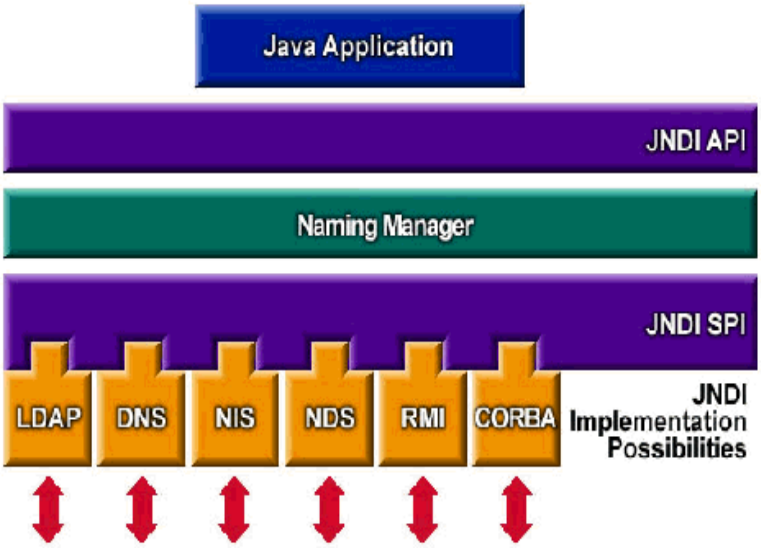
이 절에서는 다음 내용을 설명합니다.

- JNDI 구조
- J2EE 이름 지정 서비스
- 이름 지정 참조 및 바인딩 정보
- J2EE 표준 배포 설명자의 이름 지정 참조
- JNDI 연결 팩토리

JNDI 구조

JNDI 구조는 API (Application Programmer's Interface)와 SPI (Service Provider Interface)로 구성됩니다. Java 응용 프로그램은 JNDI API를 사용하여 다양한 이름 지정 및 디렉토리 서비스를 액세스합니다. SPI는 다양한 이름 지정 및 디렉토리 서비스가 투명하게 플러그 인되게 함으로써 Java 응용 프로그램이 JNDI API를 사용하여 해당 서비스에 액세스할 수 있게 해줍니다. 다음 그림 "JNDI 구조의 개요"는 JNDI API를 통해 액세스할 수 있는 서비스를 보여줍니다.

JNDI 구조의 개요



J2EE 이름 지정 서비스

JNDI 이름은 사람들에게 친숙한 객체 이름입니다. 이러한 이름은 J2EE 서버에서 제공하는 이름 지정 및 디렉토리 서비스에 의해 객체에 바인딩됩니다. J2EE 구성 요소는 JNDI API를 통해 이 서비스에 액세스하므로 일반적으로 사람들에게 친숙한 객체 이름을 JNDI 이름이라고 합니다. Pointbase 데이터베이스의 JNDI 이름은 `jdbc/Pointbase`입니다. Sun ONE Application Server는 시작할 때 구성 파일로부터 정보를 읽어 이름 공간에 JNDI 데이터베이스 이름을 자동으로 추가합니다.

J2EE 응용 프로그램 클라이언트, Enterprise Bean 및 웹 구성 요소에는 JNDI 이름 지정 환경에 대한 액세스 권한이 필요합니다.

응용 프로그램 구성 요소의 이름 지정 환경은 배포나 조립 중에 응용 프로그램 구성 요소 비즈니스 논리의 사용자 정의를 허용하는 메커니즘입니다. 응용 프로그램 구성 요소 환경을 사용하면 응용 프로그램 구성 요소의 소스 코드를 액세스하거나 변경할 필요 없이 응용 프로그램 구성 요소를 사용자 정의할 수 있습니다.

J2EE 컨테이너는 응용 프로그램 구성 요소의 환경을 구현하며 이러한 환경을 응용 프로그램 구성 요소 인스턴스에 JNDI 이름 지정 컨텍스트로 제공합니다. 응용 프로그램 구성 요소의 환경은 다음과 같이 사용됩니다.

- 응용 프로그램 구성 요소의 비즈니스 메소드는 JNDI 인터페이스를 사용하여 환경에 액세스합니다. 응용 프로그램 구성 요소 공급자는 해당 응용 프로그램 구성 요소가 런타임에 자체 환경 내에서 필요로 하는 모든 환경 항목을 배포 설명자에 선언합니다.
- 컨테이너는 응용 프로그램 구성 요소 환경을 저장하는 JNDI 이름 지정 컨텍스트 구현을 제공합니다. 컨테이너는 배포자가 각 응용 프로그램 구성 요소의 환경을 만들고 관리할 수 있게 해주는 도구도 제공합니다.
- 배포자는 컨테이너에서 제공하는 도구를 사용하여 응용 프로그램 구성 요소 배포 설명자에 선언되어 있는 환경 항목을 초기화합니다. 배포자는 환경 항목의 값을 설정하고 수정합니다.
- 컨테이너는 런타임에 환경 이름 지정 컨텍스트를 응용 프로그램 구성 요소 인스턴스에서 사용할 수 있게 합니다. 응용 프로그램 구성 요소의 인스턴스는 JNDI 인터페이스를 사용하여 환경 항목의 값을 가져옵니다.

각 응용 프로그램 구성 요소는 고유 환경 항목 집합을 정의합니다. 같은 컨테이너 내의 모든 응용 프로그램 구성 요소 인스턴스는 같은 환경 항목을 공유합니다. 응용 프로그램 구성 요소 인스턴스는 런타임에 환경을 수정할 수 없습니다. 웹 컨테이너 및 EJB 컨테이너 같은 J2EE 컨테이너가 JNDI 이름 지정 서비스를 사용하여 객체를 조회하는 방법에 대한 자세한 내용은 199페이지의 “J2EE 컨테이너 구성”을 참조하십시오.

이름 지정 참조 및 바인딩 정보

자원 참조는 자원에 대한 구성 요소의 코드화된 이름을 식별하는 배포 설명자의 요소입니다. 즉, 코드화된 이름은 자원의 연결 팩토리를 참조합니다. 다음 절의 예에서 자원 참조 이름은 `jdbc/SavingsAccountDB`입니다.

자원의 JNDI 이름과 자원 참조의 이름은 같지 않습니다. 이 방법으로 이름을 지정하려면 배포 전에 두 이름을 매핑해야 하지만 자원으로부터 구성 요소를 분리하기도 합니다. 이러한 분리 기능으로 인해 나중에 구성 요소가 다른 자원에 액세스해야 하는 경우 코드에서 이름을 변경하지 않아도 됩니다. 또한 이러한 융통성으로 인해 기존의 구성 요소로부터 J2EE 응용 프로그램을 어셈블 하기가 쉽습니다.

다음 표, "JNDI 조회 및 관련 참조"는 Sun ONE Application Server가 사용하는 J2EE 자원에 대한 JNDI 조회 및 관련 참조를 나열합니다.

JNDI 조회 및 관련 참조

JNDI 조회 이름	관련 참조
`java:comp/env`	응용 프로그램 환경 항목
`java:comp/env/jdbc`	JDBC 데이터 소스 자원 관리자 연결 팩토리
`java:comp/env/ejb`	EJB 참조
`java:comp/UserTransaction`	UserTransaction 참조
`java:comp/env/mail`	JavaMail 세션 연결 팩토리
`java:comp/env/url`	URL 연결 팩토리
`java:comp/env/jms`	JMS 연결 팩토리 및 대상
`java:comp/ORB`	응용 프로그램 구성 요소 간에 공유되는 ORB 인스턴스

J2EE 표준 배포 설명자의 이름 지정 참조

이름 지정 참조는 주어진 이름 지정 컨텍스트에서 객체를 조회하기 위해 응용 프로그램에서 사용하는 문자열입니다. 각 J2EE 응용 프로그램에 대해 이름 지정 컨텍스트가 있고 표준 구성 요소 배포 설명자에 참조가 구성되어 있습니다. 이 절에서는 Sun ONE Application Server에서 사용하는 표준 배포 설명자에 대해 설명합니다. 이 절은 다음 내용으로 구성되어 있습니다.

- 응용 프로그램 환경 항목
- EJB 참조
- 자원 관리자 연결 팩토리에 대한 참조
- 자원 환경 참조
- UserTransaction 참조
- COSNaming 서비스

응용 프로그램 환경 항목

<env-entry>를 사용하여 정의하는 환경 항목은 J2EE 응용 프로그램에 배포 시간 매개 변수를 지정할 수 있는 방법을 제공합니다. 웹 응용 프로그램의 경우 <context-param>을 사용하여 서블릿 컨텍스트 초기화 매개 변수를 정의할 수 있지만 응용 프로그램 배포자는 응용 프로그램 매개 변수의 이름, 유형 및 값을 명시적으로 지정하여 이러한 매개 변수를 구성하므로 <env-entry>가 선호되는 방법입니다.

다음 예에서는 J2EE 표준 배포 설명자에 지정되어 있는 <env-entry>의 구문을 설명합니다.

```
<env-entry>
<description> Send pincode by mail </description>
<env-entry-name> mailPincode </env-entry-name>
<env-entry-value> false </env-entry-value>
<env-entry-type> java.lang.Boolean </env-entry-type>
</env-entry>
```

<env-entry-type> 태그는 항목에 대한 전체 클래스 이름을 지정합니다. 다음은 응용 프로그램 구성 요소(서블릿/JSP, Entity Bean 또는 IIOP 응용 프로그램 클라이언트를 가리키는 용어)에서 JNDI를 사용하여 <env-entry>를 조회할 수 있는 코드 조각입니다.

```
Context initContext = new InitialContext();
Boolean mailPincode = (Boolean)
initContext.lookup("java:comp/env/mailPincode");

// one could use relative names into the sub-context
Context envContext = initContext.lookup("java:comp/env");
Boolean mailPincode = (Boolean)
envContext.lookup("mailPincode");
```

EJB 참조

다음 예에서 설명하는 것처럼 배포 설명자 지원 기능과는 별도로 JNDI 이름 지정 서비스는 응용 프로그램에서 "논리" 이름(EJB 참조)을 사용하여 Enterprise Bean의 홈 인터페이스에 매핑하게 해줍니다.

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<ejb-ref-type> Entity </ejb-ref-type>
<home> com.wombat.empl.EmployeeRecordHome </home>
<remote> com.wombat.empl.EmployeeRecord </remote>
<ejb-link> EmployeeEJB </ejb-link>
</ejb-ref>
```

JSP와 같은 응용 프로그램 구성 요소는 다음 예에서 설명하는 것처럼 JNDI를 사용하여 EJB 홈 객체에 액세스할 수 있습니다.

```
Context initContext = new InitialContext();
Context envContext = initContext.lookup("java:comp/env");
Object result = envContext.lookup("ejb/EmplRecord");
EmployeeRecordHome emplRecordHome = (EmployeeRecordHome)
javax.rmi.PortableRemoteObject.narrow(result,
EmployeeRecordHome.class);
```

`ejb-ref-name` 요소는 위의 예에서와 같이 응용 프로그램 코드에 사용되는 문자열을 정의합니다. `ejb-link` 요소는 `ejb-jar.xml`에서 정의된 Entity Bean의 `ejb-name` 요소를 사용하여 정의되는 대상 Enterprise Bean에 이러한 참조를 연결합니다. 응용 프로그램 배포 설명자나 Enterprise Bean 설명자 중 하나를 수정하지 않고 연결을 설정할 수 있습니다.

자원 관리자 연결 팩토리에 대한 참조

팩토리는 요청 시 다른 객체를 만드는 객체입니다. 자원 팩토리는 데이터베이스 연결이나 메시지 서비스 연결 같은 자원 객체를 만들며 이러한 객체는 표준 배포 설명자의 `<resource-ref>` 요소를 사용하여 구성됩니다.

다음 예에서는 팩토리 사용을 설명합니다.

예 A:

`javax.sql.DataSource` 유형의 객체를 반환하는 JDBC 연결 팩토리에 대한 참조 선언

```
<resource-ref>
<description> Primary database </description>
<res-ref-name> jdbc/primaryDB </res-ref-name>
<res-type> javax.sql.DataSource </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

예 B:

다음은 JavaMail 세션 자원 팩토리를 참조하는 예입니다.

```
<resource-ref>
<description> mail Session </description>
<res-ref-name> mail/Session </res-ref-name>
<res-type> javax.mail.Session </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

<res-type>은 자원 팩토리의 전체 클래스 이름입니다. <res-auth> 변수를 Container나 Application에 값으로 할당할 수 있습니다. JavaMail 세션 자원 팩토리 구성에 대한 자세한 내용은 285페이지의 "JavaMail 자원 정보"를 참조하십시오.

Container가 지정되어 있는 경우 웹 컨테이너는 자원 팩토리를 JNDI 조회 레지스트리에 바인딩하기 전에 인증을 처리합니다. Application이 지정되어 있으면 서블릿이 프로그래밍 방식으로 인증을 처리해야 합니다. 다음과 같이 자원 유형을 설명하는 별도의 하위 컨텍스트에서 다양한 자원 팩토리를 조회할 수 있습니다.

- JDBC `javax.sql.DataSource` 팩토리의 경우 `jdbc/`
- `javax.jms.QueueConnectionFactory` 또는 `JMS javax.jms.TopicConnectionFactory`의 경우 `jms/`
- `JavaMail javax.mail.Session` 팩토리의 경우 `mail/`
- `java.net.URL factory`의 경우 `url/`

다음은 응용 프로그램 구성 요소의 JDBC 연결을 인증을 처리하는 컨테이너와 연결하기 위한 코드 조각입니다.

```
InitialContext initContext = new InitialContext();
DataSource source =
(DataSource) initContext.lookup("java:comp/env/jdbc/primaryDB");
Connection conn = source.getConnection();
```

이러한 자원 참조가 제대로 작동하려면 `res-ref-name`이 런타임에 유효한 자원 팩토리에 매핑되어야 합니다.

자원 환경 참조

자원 환경 참조는 JNDI 조회를 통해 자원과 관련된 관리 대상 객체에 액세스하는 방법을 제공합니다. 예를 들어, 응용 프로그램에서 JMS 대상 객체에 액세스해야 할 경우가 있습니다. 표준 배포 설명자에 정의되어 있는 <resource-env-ref> 요소를 사용하면 응용 프로그램이 자원 요구 사항을 선언할 수 있습니다.

<resource-env-ref>와 <resource-ref> 요소의 주요한 차이점은 특정 자원 인증 요구 사항이 없다는 것입니다. 두 가지 요소 모두 자원 팩토리 설명자가 백업해야 합니다.

예:

```
<resource-env-ref>
<description> My Topic </description>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<res-env-ref-type> javax.jms.Topic </res-type>
</resource-env-ref>
```

다음 코드를 사용하면 JMS 항목 객체에 액세스할 수 있습니다.

```
InitialContext initContext = new InitialContext();
javax.jms.Topic myTopic =
(javax.jms.Topic) initContext.lookup("java:comp/env/jms/MyTopic");
```

이러한 `resource-env-ref` 변수가 올바로 작동하려면 관리자가 런타임에 대상 자원 팩토리를 사용할 수 있게 만들어야 합니다. JMS 항목 및 대기열 대상에 대한 자세한 내용은 11장, "JMS 서비스 사용"을 참조하십시오.

UserTransaction 참조

J2EE는 JNDI 이름 `java:comp/UserTransaction`에서 컨테이너가 `UserTransaction` 객체 구현을 제공할 것을 요구합니다. 응용 프로그램은 `UserTransaction` 객체를 사용하여 트랜잭션을 시작, 완결 및 중단할 수 있습니다.

프로그래밍 방식으로 트랜잭션을 시작하고 수행하기 위해 구성 요소는 `java:comp/UserTransaction`에 대해 JNDI 조회를 하여 컨테이너의 기본 트랜잭션 조정자에 대한 참조를 얻습니다. 반환된 객체는 `javax.transaction.UserTransaction` 인터페이스를 구현하며 프로그램에서 트랜잭션의 시작, 완결, 롤백 및 상태 쿼리를 위해 사용할 수 있습니다. Sun ONE Application Server의 JNDI 구현은 이러한 트랜잭션 조정자 조회를 지원합니다. `javax.transaction.UserTransaction` 인터페이스에 대한 자세한 내용은 219 페이지의 “트랜잭션 서비스 사용”을 참조하십시오.

초기 이름 지정 컨텍스트

Sun ONE Application Server에서 지원하는 이름 지정 기능은 기본적으로 J2EE 1.3을 기반으로 하며 몇 가지 향상된 기능이 추가되었습니다. 응용 프로그램 구성 요소가 `InitialContext()`를 통해 초기 컨텍스트를 만들면 Sun ONE Application Server는 응용 프로그램의 이름 지정 환경에 대한 핸들 역할을 하는 객체를 반환합니다. 그러면 이 객체는 `java:comp/env` 이름 공간에 대한 하위 컨텍스트를 제공합니다. 각 응용 프로그램은 고유한 이름 공간을 갖습니다. 즉, 각 응용 프로그램별로 `java:comp/env` 이름 공간이 존재하므로 한 응용 프로그램 이름 공간에 바인딩된 객체는 다른 응용 프로그램에 바인딩된 객체와 충돌하지 않습니다.

COSNaming 서비스

EJB 상호 운용성 프로토콜은 JNDI API를 사용하여 EJB 객체를 조회할 때 COSNaming 프로토콜을 사용할 것을 요구합니다.

CORBA `CosNaming` 서비스에서 `EJBHome` 객체 참조를 게시하려면 EJB 컨테이너가 필요합니다. `CosNaming` 서비스는 정의되어 있는 `CosNaming` 모듈에서 IDL 인터페이스를 구현해야 하며 클라이언트가 IIOP를 통해 확인 및 목록 작업을 호출할 수 있도록 합니다.

`CosNaming` 서비스는 루트 `NamingContext` 객체의 호스트, 포트 및 객체 키를 제공하기 위해 CORBA Interoperable Name Service 사양의 요구 사항을 따라야 합니다. `CosNaming` 서비스는 광고 호스트, 포트 및 객체 키에서 루트 `NamingContext`의 IIOP 호출을 처리할 수 있어야 합니다.

클라이언트 컨테이너(즉, EJB, 웹 또는 응용 프로그램 클라이언트 컨테이너)는 표준 `CosNaming` API를 사용하여 서버의 `CosNaming` 서비스에 연결하고 `EJBHome` 객체를 확인하기 위해 Interoperable Name Service 사양에 정의되어 있는 메커니즘을 사용하는 JNDI `CosNaming` 서비스 공급자를 포함해야 합니다. JNDI `CosNaming` 서비스 공급자는 JNDI SPI 구조를 사용할 수도 있고 사용하지 않을 수도 있습니다. JNDI `CosNaming` 서비스 공급자는 다음 URL로부터 객체 참조를 만들어 서버 `CosNaming` 서비스의 루트 `NamingContext`에 액세스해야 합니다.

`corbaloc:iiop:1.2@<host>:<port>/<objectkey>` (여기서 `<host>, <port>, <objectkey>`는 서버의 `CosNaming` 서비스 또는 이와 동등한 메커니즘을 사용하여 광고한 루트 `NamingContext`에 해당하는 값임)

배포 시 클라이언트 컨테이너의 개발자는 서버의 `CosNaming` 서비스의 호스트, 포트 및 객체 키와 클라이언트 구성 요소 배포 설명자의 각 `ejb-ref` 요소에 대해 서버의 이름 공간을 검색하는 것과 같은 작업을 수행하여 서버 `EJBHome` 객체의`CosNaming` 이름을 얻어야 합니다. JNDI 조회 호출 시 클라이언트 코드에서 사용하는 `ejb-ref-name`은 `EJBHome` 객체의 `CosNaming` 이름에 연결되어야 합니다. 런타임에 클라이언트 구성 요소의 JNDI 조회 호출은 `CosNaming` 서비스 공급자를 사용합니다. 이 서비스 공급자는 서버의 `CosNaming` 서비스에 연결하여 `CosNaming` 이름을 결정한 다음 `EJBHome` 객체 참조를 클라이언트 구성 요소에 반환합니다.

`EJBHome` 객체의 이름은 제공된 호스트와 포트에서 액세스할 수 있는 `CosNaming` 서비스의 이름 공간 내에 있으므로 클라이언트와 서버 컨테이너의 이름 공간을 결합할 필요가 없습니다.

`CosNaming` 사용의 장점은 비 J2EE CORBA 클라이언트 및 서버와의 상호 운용성을 비롯하여 상호 운용성에 필요한 IIOP 인프라와 더 잘 통합된다는 점입니다. `CosNaming`은 CORBA 객체만 저장하기 때문에 공급업체는 다른 자원을 저장하는 데 다른 엔터프라이즈 디렉토리 서비스를 사용할 가능성이 높습니다.

Sun ONE Application Server는 J2EE 1.3 사양에 따라 JNDI의 모든 이름 지정 자원을 통합합니다.

CosNaming 공급자. 전역 JNDI 이름 공간(IIOP 응용 프로그램 클라이언트 액세스 가능)을 지원하기 위해 Sun ONE Application Server는 CORBA 참조(원격 EJB 참조) 바인딩을 지원하는 J2EE 기반 CosNaming 공급자를 포함합니다. IIOP 클라이언트에 반환되는 `InitialContext`는 CosNaming 공급자입니다. Sun ONE Application Server 서버의 인스턴스는 IIOP 클라이언트가 조회하고 바인딩하는 Entity Bean을 등록합니다.

Sun ONE Application Server에서는 CosNaming에 저장되어 있는 객체를 처리하며 로컬 JNDI 이름 지정 환경은 일시적인 것임에 유의하십시오. 즉, 응용 프로그램을 재로드하거나 서버를 시작할 때마다 모든 관련 객체는 이름 공간에 다시 바인딩됩니다. CORBA/IIOP 클라이언트 구성 지원에 대한 자세한 내용은 327페이지의 “Corba/IIOP 클라이언트를 지원하도록 서버 구성”을 참조하십시오.

JNDI 연결 팩토리

J2EE 웹 응용 프로그램의 경우 `web.xml` 파일의 배포 설명자는 응용 프로그램 환경 항목, 자원 관리자(예: SQL 데이터 소스) 연결 팩토리 또는 EJB에 대한 참조를 정의하기 위한 자리 표시자입니다. 응용 프로그램은 J2EE 컨테이너가 제공하는 JNDI `InitialNamingContext`를 사용하여 이러한 참조를 조회합니다. 따라서 응용 프로그램의 소스 코드를 액세스하거나 수정하지 않고 배포 설명자만 변경하여 응용 프로그램을 다양한 응용 프로그램 서버 환경에 이식할 수 있습니다. 마찬가지로 J2EE는 Entity Bean (`ejb-jar.xml`)과 IIOP 응용 프로그램 클라이언트 (`application-client.xml`)의 배포 설명자를 이러한 JNDI 이름 지정 참조를 위한 주요 방법으로 사용합니다.

연결 팩토리는 J2EE 구성 요소가 자원에 액세스할 수 있게 해주는 연결 객체를 만드는 객체입니다. 데이터베이스의 연결 팩토리는 `javax.sql.DataSource` 객체이며 이 객체는 `java.sql.Connection` 객체를 만듭니다.

Sun ONE Application Server에서 다음 자원 및 자원 팩토리에 액세스하기 위한 방법을 구성할 수 있습니다.

- JDBC 연결 팩토리
- MQ를 기반으로 하는 JMS 연결 팩토리
- JavaMail 세션 연결 팩토리
- JCA 커넥터 팩토리

- 일반, 사용자 정의 사용자 작성 자원 객체 팩토리
- LDAP와 같은 외부 자원 리포지토리에 대한 지원

모든 Sun ONE Application Server 자원 팩토리는 `server.xml`의 `<resources> </resources>` 태그 안에 지정되어 있으며 `jndi-name` 속성을 사용하여 지정한 JNDI 이름을 갖습니다. 이 속성은 서버 차원의 이름 공간에서 팩토리를 등록할 때 사용됩니다. 배포자는 `resource-ref-mapping` 요소를 사용하여 사용자 지정, 응용 프로그램별 자원 참조 이름(`resource-ref` 또는 `resource-env-ref` 요소 내에 선언됨)을 서버 차원의 자원 팩토리에 매핑할 수 있습니다. 따라서 주어진 응용 프로그램에 대해 어떤 JDBC 드라이버 및 기타 자원 팩토리를 사용할지와 관련하여 배포 시간을 결정할 수 있습니다.

사용자 정의 자원은 로컬 JNDI 리포지토리에 액세스하고 외부 자원은 외부 JNDI 리포지토리에 액세스합니다. 두 가지 유형의 자원 모두 사용자 지정 팩토리 클래스 요소, JNDI 이름 속성 등을 필요로 합니다. 이 절에서는 J2EE 자원에 대해 JNDI 연결 팩토리 자원을 구성하는 방법과 이러한 자원을 액세스하는 방법을 설명합니다.

이 절은 다음 내용으로 구성되어 있습니다.

- 사용자 정의 자원 만들기
- 외부 JNDI 자원 만들기
- 외부 JNDI 리포지토리 액세스
- 응용 프로그램 자원 참조 매핑
- URL 연결 팩토리 자원 정보
- 응용 프로그램 자원 환경 참조 매핑
- EJB 참조 매핑

사용자 정의 자원 만들기

`server.xml`에 정의되어 있는 `custom-resource` 요소는 사용자 정의 서버 차원의 자원 객체 팩토리를 지정할 수 있는 방법을 제공합니다. 이러한 객체 팩토리는 `javax.naming.spi.ObjectFactory` 인터페이스를 구현합니다. 이 요소는 JNDI 이름(다른 Sun ONE Application Server 자원 같은 jndi-name 하위 요소를 통해 지정)을 연결하여 서버 차원 이름 공간, 유형, 자원 팩토리 클래스의 이름 및 같은 내용을 인스턴스화하는 데 사용되는 표준 등록 정보 집합에서 사용되도록 합니다.

다음 예에서는 `javax.naming.spi.ObjectFactory` 인터페이스의 구현을 설명합니다.

```
<resources> <custom-resource jndi-name="test/myBean"
res-type="test.MyBean"factory-class="test.MyBeanFactory"
enabled="true">

<property name="foo" value="test custom bean prop" />
</custom-resource>
</resources>
```

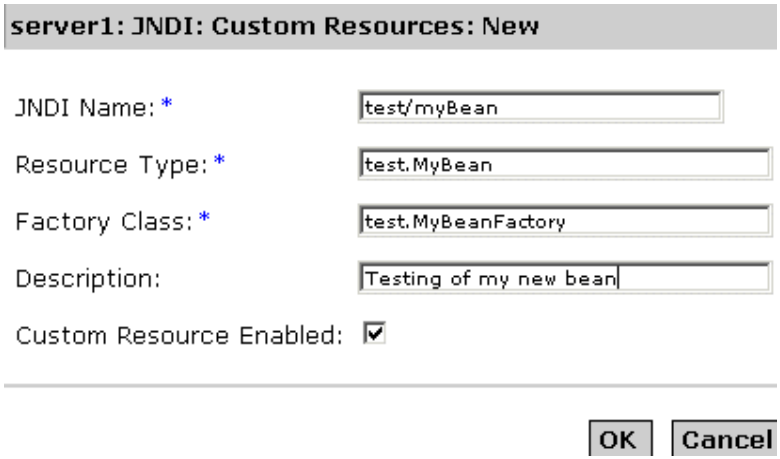
자원 참조의 환경 참조와 EJB 참조가 `server.xml`의 `custom-resource`와 `external-jndi-resource` 태그를 사용하여 정의한 구성 서버 차원 자원에 연결되도록 해야 합니다. 응용 프로그램 구성 요소를 동적으로 재배포하는 것은 JNDI 이름 지정 환경과 관련된

새로운 참조를 새로 설치한 응용 프로그램의 이름 지정 컨텍스트에 다시 바인딩합니다.

관리 인터페이스를 사용하여 사용자 정의 자원을 만들려면 다음을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 JNDI 구성을 수정할 Sun ONE Application Server 인스턴스를 엽니다.
2. "JNDI" 탭을 열고 "사용자 정의 자원"을 누릅니다. 사용자 정의 자원이 이미 만들어져 있으면 오른쪽 창에 나열됩니다. 새 사용자 정의 자원을 만들려면 "새로 만들기"를 누릅니다. 관리 인터페이스의 오른쪽 창에 "JNDI 사용자 정의 자원 페이지"가 표시됩니다.

JNDI 사용자 정의 자원 페이지



3. 자원을 액세스할 때 사용할 이름을 "JNDI 이름" 필드에 입력합니다. 이 이름은 JNDI 이름 지정 서비스에 등록됩니다.

4. 위의 예에서와 같이 "자원 유형" 필드에 전체 유형 정의를 입력합니다. 자원 유형 정의는 `xxx.xxx` 형식을 따라야 합니다.
5. 만들고 있는 사용자 정의 자원에 대한 팩토리 클래스 이름을 "팩토리 클래스" 필드에 입력합니다. 팩토리 클래스는 팩토리 클래스에 대한 사용자 지정 이름입니다. 이 클래스는 `javax.naming.spi.ObjectFactory` 인터페이스를 구현합니다.
6. 만들고 있는 자원에 대한 설명을 "설명" 필드에 입력합니다. 이 설명은 문자열 값이므로 최대 250자까지 입력할 수 있습니다.
7. 사용자 정의 자원을 활성화하려면 "사용자 정의 자원 사용" 확인란을 선택합니다.
8. 사용자 정의 자원을 저장하려면 "확인"을 누릅니다.

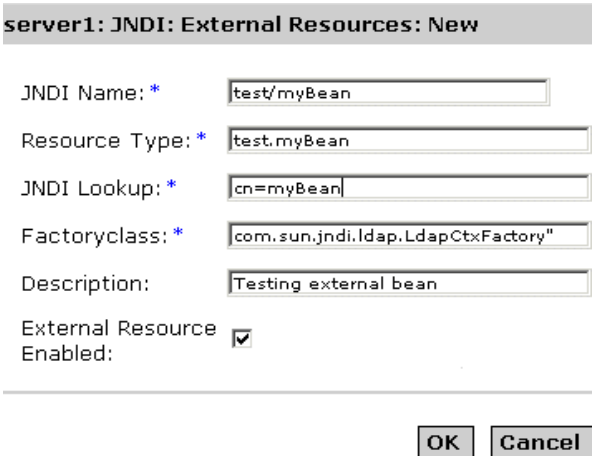
외부 JNDI 자원 만들기

관리 인터페이스를 사용하여 외부 자원을 만들려면 다음을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 JNDI 구성을 수정할 Sun ONE Application Server 인스턴스를 엽니다.
2. JNDI를 열고 "외부 자원"을 선택합니다. 외부 자원이 이미 만들어져 있으면 오른쪽 창에 나열됩니다. 새 외부 자원을 만들려면 "새로 만들기"를 누릅니다.

관리 인터페이스 오른쪽 창에 그림 "JNDI 외부 자원 페이지"에서와 같이 다음 창이 표시될 것입니다.

JNDI 외부 자원 페이지



3. 자원을 액세스하는 데 사용할 이름을 "JNDI 이름" 필드에 입력합니다. 이 이름은 JNDI 이름 지정 서비스에 등록됩니다.
4. 위의 예에서와 같이 "자원 유형" 필드에 전체 유형 정의를 입력합니다. 자원 유형 정의는 `xxx.xxx` 형식을 따라야 합니다.
5. 외부 리포지토리에서 조회할 JNDI 값을 "JNDI 조회" 필드에 입력합니다. 예를 들어, Bean 클래스를 테스트하기 위해 외부 자원을 만들어 외부 리포지토리에 연결하는 경우 JNDI 조회는 `cn=`*testmybean*을 읽을 수 있습니다.
6. `com.sun.jndi.ldap`와 같은 JNDI 팩토리 클래스 외부 리포지토리를 "팩토리 클래스" 필드에 입력합니다. 이 클래스는 javax.naming.spi.`ObjectFactory` 인터페이스를 구현합니다.
7. 만들고 있는 자원에 대한 설명을 "설명" 필드에 입력합니다. 이 설명은 문자열 값이므로 최대 250자까지 입력할 수 있습니다.
8. 외부 자원을 활성화하려면 "외부 자원 사용" 확인란을 선택합니다.
9. 사용자 정의 자원을 저장하려면 "확인"을 누릅니다.

외부 JNDI 리포지토리 액세스

Sun ONE Application Server에서 실행되는 응용 프로그램의 경우 외부 JNDI 리포지토리에 저장된 자원에 액세스해야 하는 경우가 있습니다. 예를 들어, LDAP 서버에 일반 Java 객체를 Java 스키마별로 저장할 수 있습니다. 사용자는 외부 JNDI 자원 요소를 사용하여 이러한 외부 자원 리포지토리를 구성할 수 있습니다. 외부 JNDI 팩토리는 `javax.naming.spi.InitialContextFactory` 인터페이스를 구현해야 합니다.

예:

```
<resources>
<!-- external-jndi-resource element specifies how to access J2EE resources
-- stored in an external JNDI repository. The following example
-- illustrates how to access a java object stored in LDAP.
-- factory-class element specifies the JNDI InitialContext factory that
-- needs to be used to access the resource factory. property element
-- corresponds to the environment applicable to the external JNDI context
-- and jndi-lookup-name refers to the JNDI name to lookup to fetch the
-- designated (in this case the java) object.
-->
<external-jndi-resource jndi-name="test/myBean"
jndi-lookup-name="cn=myBean"
res-type="test.myBean"
factory-class="com.sun.jndi.ldap.LdapCtxFactory">

<property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
<property name="SECURITY_AUTHENTICATION" value="simple" />
<property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
<property name="SECURITY_CREDENTIALS" value="changeit" />
</external-jndi-resource>
</resources>
```

응용 프로그램 자원 참조 매핑

응용 프로그램 관련 자원 참조는 미리 정의된 서버 차원 자원 팩토리에 매핑해야 합니다. 이때 Sun ONE Application Server 관련 자원 참조 매핑 요소를 사용할 수 있습니다.

다음 예에서는 자원 참조가 JDBC 데이터 소스에 지정되어 있는 웹 응용 프로그램의 배포 설명자인 `web.xml`을 살펴볼 것입니다.

```
<resource-ref>
<res-ref-name> jdbc/EstoreDataSource </res-ref-name>
<res-type> javax.sql.DataSource </res-type>
<res-auth> Container </res-auth>
</resource-ref>
```

원하는 `res-ref-name`을 다음과 같이 컨테이너 차원 Orcale JDBC 연결 자원 팩토리에 매핑할 수도 있습니다.

```
<resource-ref>
<res-ref-name> jdbc/EstoreDataSource </resource-ref-name>
<jndi-name> jdbc/estore/InventoryDB </jndi-name>
</resource-ref>
```

URL 연결 팩토리 자원 정보

URL 연결 팩토리의 경우 server.xml에 정의되어 있는 자원이 필요하지 않습니다. 해당 Sun ONE Application Server 응용 프로그램(웹 또는 ejb) 배포 설명자의 jndi-name 요소가 대상 URL을 지정합니다.

예를 들어, 웹 응용 프로그램의 배포 설명자 `web.xml`이 `java.net.URL` 자원 참조를 지정하고 `sun-web.xml`에 있는 URL `http://www.`*sun.com/index.html*에 매핑된다고 가정합니다.

매핑은 다음과 같을 것입니다.

```
<resource-ref>
<res-ref-name>myURL</res-ref-name>
<res-type>java.net.URL</res-type>
<res-auth>Container</res-auth>
</resource-ref>
```

```
<sun-web-app>
<resource-ref>
<res-ref-name>myURL</res-ref-name>
<jndi-name> http://www.sun.com/index.html </jndi-name>
</resource-ref>
</sun-web-app>
```

응용 프로그램 자원 환경 참조 매핑

응용 프로그램 관련 자원 환경 참조 선언은 응용 프로그램 서버의 런타임 환경에서 사용 가능한 대상 자원 객체에 매핑되어야 합니다. Sun ONE Application Server 관련 구성 파일에 정의되어 있는 자원 환경 매핑 요소를 사용하여 배포자는 다음과 같이 매핑할 수 있습니다.

예:

```
<resource-env-ref>
<description> My Topic </description>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<res-env-ref-type> javax.jms.Topic </res-type>
</resource-env-ref>
```

이 참조는 server.xml에 정의되어 있는 jms/iMQ/Topics/Stocks/SUNW 항목에 매핑됩니다. 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

```
<resource-env-ref-mapping>
<res-env-ref-name> jms/MyTopic </res-ref-name>
<jndi-name> jms/iMQ/Topics/Stocks/SUNW </jndi-name>
</resource-env-ref-mapping>
```

EJB 참조 매핑

대상 Enterprise Bean에 사용된 ejb-name에서 응용 프로그램 코드에 사용된 실제 ejb-name을 분리할 수도 있습니다. 웹 응용 프로그램 배포 설명자, web.xml을 수정하지 않고 Enterprise Bean 배포 설명자의 ejb-name을 사용하려고 할 때 이 방법이 특히 유용합니다. Sun ONE Application Server 관련 구성을 사용하면 Sun ONE Application Server 관련 배포 설명자의 ejb-ref-mapping 요소를 사용하지 않고 ejb-ref-name 요소를 대상 Bean의 ejb-name에 매핑할 수 있습니다.

예:

```
<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<ejb-ref-type> Entity </ejb-ref-type>
<home> com.wombat.empl.EmployeeRecordHome </home>
<remote> com.wombat.empl.EmployeeRecord </remote>
</ejb-ref>

<ejb-ref>
<ejb-ref-name> ejb/EmplRecord </ejb-ref-name>
<jndi-name> AccountEJB </jndi-name>
</ejb-ref-mapping>
```

지속성 관리자 자원 정보

이 모듈에서는 지속성에 대해 설명하고 Sun ONE Application Server에서 지원하는 플러그형 지속성 관리자 사용을 위한 프레임워크를 설정합니다.

이 모듈에서는 다음 내용을 설명합니다.

- 지속성이란?
- 지속성 관리자의 역할

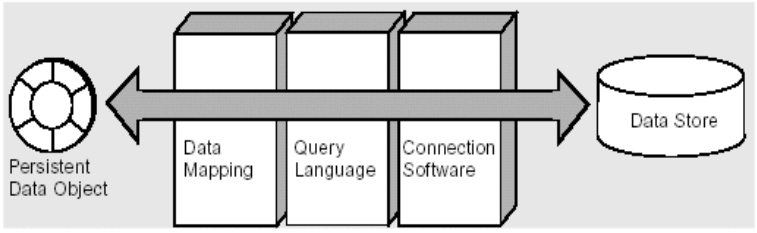
- 사전 배포 Bean 구성
- 새 지속성 관리자 만들기

지속성이란?

대부분의 비즈니스 응용 프로그램의 주요 측면은 지속성 데이터(응용 프로그램의 외부에 저장되어 있는 오래된 데이터)를 프로그래밍 방식으로 조작하는 것입니다. 지속성 데이터를 사용하거나 수정하려고 할 때는 임시 메모리로 읽어들이지만 오랫 동안 저장하기 위해서는 관계형 데이터베이스나 플랫 파일 시스템에 기록합니다.

객체 지향 프로그래밍 시스템에서 지속성 데이터는 응용 프로그램 코드에 의해 조작되는 하나 이상의 데이터 객체로 메모리에 표현됩니다. 일반적으로 데이터 저장소의 지속성 데이터와 메모리에 표현된 지속성 데이터 객체 간의 대응은 다음 그림 "기본 지속성 체계"에 표시된 대로 많은 소프트웨어 계층을 통해 이루어집니다.

기본 지속성 체계



각 데이터 저장소에는 데이터 저장소와 응용 프로그램 간의 연결을 설정하고 유지하는 데 사용되는 드라이버 소프트웨어를 통해 외부 영역과 연결되는 인터페이스가 있습니다. 이 연결이 설정되면 질의어를 사용하여 데이터 저장소로부터 정보를 검색하여 응용 프로그램으로 읽어들이거나 반대로 응용 프로그램에서 데이터 저장소로 데이터를 기록할 수 있습니다. 다른 계층은 메모리의 데이터 객체와 데이터 저장소의 정보 간의 매핑을 제공합니다.

이러한 일반 체계를 통해 프로그래머는 지속성 데이터를 응용 프로그램에서 사용하고 조작할 수 있는 런타임 객체로 표현할 수 있습니다. 이러한 체계는 CRUD로 축약되기도 하는 모든 기본 지속성 작업을 지원합니다.

- C - 지속성 데이터 만들기(데이터 저장소에 삽입)
- R - 지속성 데이터 검색(데이터 저장소에서 선택)

- U - 지속성 데이터 업데이트
- D - 지속성 데이터 삭제

지속성 관리자의 역할

지속성 관리자(PM)는 EJB 컨테이너에서 컨테이너 관리 지속성과 함께 Entity Bean의 지속성을 관리합니다. Entity Bean 공급자는 Entity Bean 클래스를 추상 클래스로 제공해야 합니다. 지속성 관리자 공급자의 도구를 통해 이를 구체적으로 구현해야 합니다. 이러한 작업을 수행하려면 추상 Entity Bean과 관련 클래스를 하위 클래스로 나눈 다음 이를 구체적으로 구현하거나 캡슐화와 위임을 사용해야 합니다.

지속성 관리자 도구가 제공하는 클래스는 Entity Bean 사이의 관계를 관리하고 지속성 상태에 대한 액세스를 관리할 책임이 있습니다. PM 도구는 컨테이너 관리 지속성(CMR)을 유지하는 데 사용되는 `java.util.Collection` 클래스 구현을 제공할 책임도 있습니다.

사전 배포 Bean 구성

Enterprise Java Bean 표준은 Entity Bean에 대한 두 가지 지속성, 컨테이너 관리 지속성(CMP)과 Bean 관리 지속성(BMP)을 제공합니다. EJB 2.0 사양은 EJB 서버와 지속성 관리자 사이의 표준 API를 정의하지 않습니다.

이 절에서는 배포 및 코드 생성 시의 통합 요구 사항을 설명합니다. 배포라는 용어는 여러 의미로 사용될 수 있습니다. 일반적으로 배포 과정에는 구성, 코드 생성, 설치라는 세 가지 뚜렷한 단계가 포함된다고 할 수 있습니다.

사용된 지속성 메커니즘, 지속성 공급업체, 사용 중인 버전, 지속성 메커니즘에 필요한 추가 정보 등을 비롯하여 Bean에 대해 많은 등록 정보를 지정해야 합니다. 대부분의 지속성 공급업체는 모든 관련 Bean과 그 종속 클래스를 단일 단위로 배포할 수 있는 프로젝트의 개념을 갖고 있습니다. 프로젝트별로 공급업체 관련 xml 파일이 있을 수 있습니다.

배포 목적으로 지원되는 세 가지 표준 파일에는 `ejb-jar.xml`, `sun-ejb-jar.xml` 및 `sun-cmp-mappings.xml`이 포함됩니다. `sun-ejb-jar.xml`에 CMP Bean과 함께 있는 각 EJB 모듈에는 다섯 개 이상의 속성을 지정하는 한 개 이상의 `<pm-descriptor>` 요소와 함께 `<pm-descriptors>`가 있어야 합니다. 이러한 다섯 가지 속성은 `pm-identifier`, `pm-version`, `pm-config`, `pm-class-generator`, `pm-mapping-factory`입니다.

Sun ONE Application Server 관련 설명자(`sunEjb_jar_2_0.DTD`에서와 마찬가지)는 지속성 관리자 관련 태그를 정의합니다. 샘플 CMP 설명자는 Sun ONE Application Server DTD에 정의된 것처럼 아래와 비슷할 것입니다.

```
 PM descriptors contain one or more pm descriptors, but only of them
must be in use at any given time

-->

<!ELEMENT pm-descriptors ( pm-descriptor+, pm-inuse)>

<!--

 pm-descriptor describes the properties for the persistence manager

 associated with entity bean

-->

<!ELEMENT pm-descriptor ( pm-identifier, pm-version, pm-config?,
pm-class-generator?,

pm-mapping-factory?)>

<!--
```

이 요소는 PM 구현을 제공한 공급업체를 설명합니다. 예를 들어, Sun ONE Application Server Transparent Persistence, `TopLink`, `Versant` 또는 `CocoBase`가 될 수 있습니다.

```
-->

<!ELEMENT pm-identifier (#PCDATA)>

<!--

 pm-version further specifies which version of PM vendor product to
be used

-->

<!ELEMENT pm-version (#PCDATA)>

<!--

 pm-config specifies the vendor specific config file to be used

-->
```

```
<!ELEMENT pm-config (#PCDATA)>

<!--

 pm-class-generator specifies the vendor specific concrete class
generator

 This is the name of the class specific to a vendor:

-->

<!ELEMENT pm-class-generator (#PCDATA)>

<!--

 pm-mapping-factory specifies the vendor specific mapping factory

 This is the name of the class specific to a vendor:

-->

<!ELEMENT pm-mapping-factory (#PCDATA)>
```

새 지속성 관리자 만들기

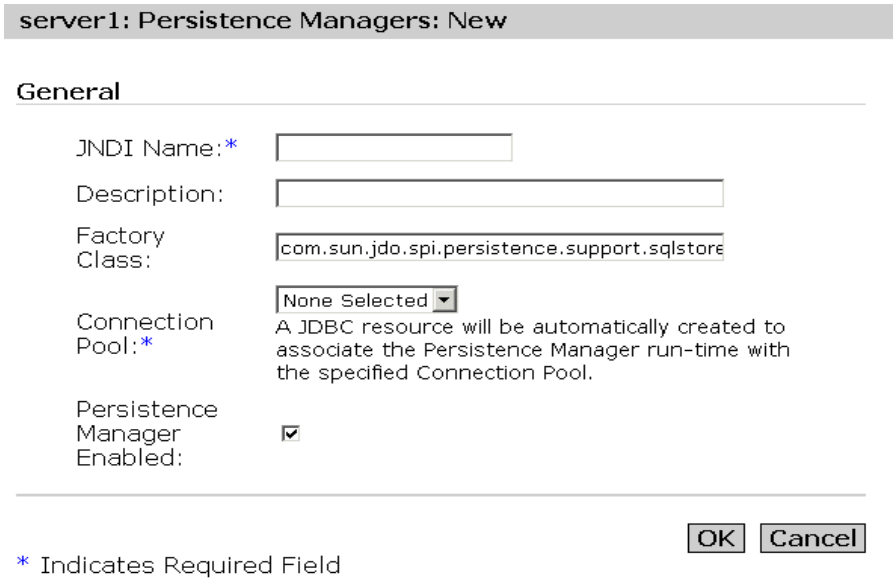
관리 인터페이스를 사용하여 새 지속성 관리자 인스턴스를 만들 수 있습니다. 새 지속성 관리자 인스턴스를 만들려면 다음을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 새 지속성 관리자를 만들 Sun ONE Application Server 인스턴스를 엽니다. 표시된 서버 구성 요소 목록에서 "지속성 관리자"를 누릅니다.

   Sun ONE Application Server의 해당 인스턴스에 대해 지속성 관리자가 만들어져 있으면 관리 인터페이스의 오른쪽 창에 목록이 표시될 것입니다.

2. 새 지속성 관리자를 만들려면 "새로 만들기"를 누릅니다. 그림 "새 지속성 관리자 만들기"에 표시된 다음 창이 표시될 것입니다.

새 지속성 관리자 만들기



3. 응용 프로그램을 대신하여 특정 지속성 관리자를 찾기 위해 응용 프로그램 서버가 런타임에 사용하는 JNDI 이름입니다. 이 이름은 Sun 관련 배포 설명자의 Entity Bean cmp 자원 요소에 정의된 것과 같아야 합니다.
4. "설명" 필드에 새 지속성 관리자에 대한 설명을 입력합니다. 이 필드의 값은 문자열이므로 250자까지 입력할 수 있습니다.
5. 지속성 관리자의 팩토리 클래스 연결을 "팩토리 클래스" 필드에 입력합니다. `setEntityContext`는 JNDI 이름 조회를 통해 이 연결 팩토리를 조회합니다. 팩토리 클래스 이름은 지속성 관리자 인스턴스를 만드는 지속성 관리자 팩토리의 클래스 이름입니다. 기본적으로 이 이름은 Sun ONE Application Server의 내부 지속성 관리자 팩토리 클래스로 설정됩니다. 대체 구현을 사용하는 경우 이 클래스를 서버 클래스 경로에서 사용할 수 있도록 해야 합니다.

6. "연결 풀" 드롭다운 목록에서 새 지속성 관리자를 풀링할 데이터베이스 연결 풀을 선택합니다. 연결 풀링을 사용하면 Entity Bean은 단일 연결을 요청하고 이를 사용하여 여러 클라이언트 스레드에 대해 명령문을 동시에 실행할 수 있습니다. 다른 데이터베이스 액세스와 마찬가지로 지속성 관리자는 연결 풀링을 사용하여 성능 및 확장성을 향상시킵니다. 기존 연결 풀을 선택하거나, 아직 만들어진 풀이 없는 경우 "선택된 대상이 없습니다."를 선택합니다.

   주: JDBC 자원이 자동으로 만들어져 PM 런타임이 JNDI를 사용하여 연결 풀에 바인딩하게 해줍니다. JDBC 자원의 JNDI 이름은 "PM"이란 접두어를 갖는 PM JNDI 이름과 같을 것입니다. 지속성 관리자를 삭제하면 관련 JDBC 자원도 삭제됩니다.

7. 지속성 관리자를 활성화하려면 "지속성 관리자 사용" 확인란을 선택합니다. 이제 지정된 연결 팩토리에 대해 지속성 관리자가 활성화됩니다.

8. 변경 사항을 저장하려면 "확인"을 누릅니다.

JDBC 자원 정보

이 모듈에서는 일반적인 JDBC API, JDBC 자원 및 구현과 Sun ONE Application Server에서의 사용에 대해 구체적으로 설명합니다.

이 모듈은 다음 절로 구성되어 있습니다.

- JDBC API 정보
- 데이터베이스 액세스 모델 정보
- JDBC 데이터 소스 정보
- JDBC 연결 정보
- JDBC 트랜잭션 정보

JDBC API 정보

JDBC API는 실제로 모든 유형의 테이블 형식 데이터에 액세스하기 위한 Java API입니다. 흥미롭게도 JDBC는 머리 글자어가 아닌 상표 이름입니다. 그럼에도 불구하고 JDBC를 "Java Database Connectivity"의 약어로 생각하는 경우가 종종 있습니다. JDBC API는 도구/데이터베이스 개발자를 위한 표준 API를 제공하고 모든 Java API를 사용하여 데이터베이스 응용 프로그램을 작성할 수 있게 해주는 Java 프로그래밍 언어로 작성된 일련의 클래스와 인터페이스로 구성됩니다.

JDBC API는 관계형 데이터베이스 시스템으로 SQL 문을 쉽게 보낼 수 있게 해주며 SQL의 모든 언어를 지원합니다. 하지만 JDBC 3.0 API는 SQL 이상을 포함하므로 데이터베이스 외부의 파일을 비롯한 다른 종류의 데이터 소스와도 상호 작용할 수 있게 해줍니다.

JDBC API의 이점은 응용 프로그램이 실제로 모든 데이터 소스에 액세스하며 Java 가상 머신을 사용하는 모든 플랫폼에서 실행될 수 있다는 것입니다. 즉, JDBC API를 사용하면 Sybase 데이터베이스, Oracle 데이터베이스, IBM DB2 데이터베이스 각각을 액세스하기 위해 별도의 프로그램을 작성할 필요가 없다는 것입니다. JDBC API를 사용하여 단일 프로그램을 작성하면 이 프로그램에서 SQL이나 기타 명령문을 적절한 데이터 소스로 보낼 수 있습니다. 또한 Java 프로그래밍 언어로 작성된 응용 프로그램을 사용하면 여러 플랫폼에서 실행하기 위해 여러 응용 프로그램을 작성할 필요가 없습니다. 프로그래머는 Java 플랫폼과 JDBC API를 조합하여 코드를 한 번만 작성해 놓으면 어느 플랫폼에서나 이 코드를 실행할 수 있습니다.

JDBC API를 사용한 작업

JDBC 기술 기반 드라이버(JDBC 드라이버)를 사용하면 다음 세 가지 작업을 할 수 있습니다.

- 데이터 소스와의 연결 설정
- 쿼리를 전송하여 데이터 소스에 대한 문 업데이트
- 결과 처리

다음 코드 단편은 이러한 세 단계에 대한 간단한 예를 제공합니다.

```
Context ctx = new InitialContext();

DataSource ds = (DataSource)ctx.lookup("jdbc/AcmeDB");

Connection con = ds.getConnection("myLogin", "myPassword");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");
```

```
while (rs.next()) {
          int x = rs.getInt("a");
          String s = rs.getString("b");
          float f = rs.getFloat("c");
   }
```

데이터베이스 액세스 모델 정보

JDBC API는 데이터베이스 액세스를 위해 2계층과 3계층 모델을 모두 지원합니다. Sun ONE Application Server는 좀 더 일반적인 2계층 데이터베이스 액세스 모델을 통합합니다.

이 절에서는 다음 내용을 설명합니다.

- 2계층 데이터베이스 액세스 모델
- 3계층 데이터베이스 액세스 모델

2계층 데이터베이스 액세스 모델

2계층 데이터베이스 액세스 모델에서 Java 애플릿이나 응용 프로그램은 DBMS 소유 프로토콜을 사용하여 데이터 소스와 직접 통신합니다. 이 액세스 모델에는 액세스할 특정 데이터 소스와 통신할 수 있는 JDBC 드라이버가 필요합니다. 사용자의 명령은 데이터베이스나 다른 데이터 소스로 전달되며 이러한 문의 결과는 다시 사용자에게 보내집니다. 데이터 소스는 네트워크를 통해 연결된 다른 시스템에 있을 수도 있습니다. 이러한 구성을 클라이언트/서버 구성이라고 하며 사용자 컴퓨터는 클라이언트로, 데이터 소스를 관리하는 컴퓨터는 서버의 역할을 합니다. 네트워크는 기업 내에서 직원을 연결하는 인트라넷이 되거나 인터넷이 될 수 있습니다.

3계층 데이터베이스 액세스 모델

3계층 데이터베이스 액세스 모델에서 Java 애플릿이나 응용 프로그램은 서비스의 "중간 계층"으로 명령을 보내고 이 중간 계층은 데이터 소스로 명령을 보냅니다. 클라이언트 응용 프로그램은 HTTP, RM, CORBA 또는 기타 호출을 통해 중간 계층과 통신합니다. 중간 계층은 DBMS 소유 프로토콜을 통해 데이터 저장소와 통신합니다. 데이터 소스는 명령을 처리하여 결과를 다시 중간 계층에 보내고 중간 계층은 결과를 사용자에게 다시 보냅니다. MIS 담당자는 중간 계층을 사용하여 기업 데이터에 대한 여러 가지 업데이트 및 액세스를 제어할 수 있으므로 3계층 모델을 선호할 것입니다. 그밖의 장점으로는 응용 프로그램 배포 과정이 단순화된다는 것입니다. 마지막으로 3계층 구조는 성능상의 이점을 제공합니다.

JDBC 데이터 소스 정보

DataSource 객체는 데이터 소스를 Java 프로그래밍 언어로 나타낸 것입니다. 기본적으로 데이터 소스는 데이터를 저장하기 위한 기능을 의미합니다. 데이터 소스는 대기업의 복잡한 데이터베이스처럼 복잡할 수도 있고 행과 열로 구성된 파일처럼 간단할 수도 있습니다. 데이터 소스는 원격 서버에 있을 수도 있고 로컬 데스크탑 컴퓨터에 있을 수도 있습니다. 응용 프로그램은 연결을 사용하여 데이터 소스에 액세스하므로 DataSource 객체는 DataSource 인스턴스가 나타내는 특정 데이터 소스 연결을 위한 팩토리로 생각할 수 있습니다. 데이터 소스 인터페이스는 데이터 소스와 연결을 설정하기 위한 두 가지 방법을 제공합니다.

DataSource 객체에는 자신이 나타내는 데이터 소스를 식별하고 설명하는 등록 정보가 있습니다. 또한 DataSource 객체는 JNDI 이름 지정 서비스와 함께 작동하며 해당 객체를 사용하는 응용 프로그램과는 별도로 만들어지고 배포되며 관리됩니다. 드라이버 공급업체는 DataSource 인터페이스의 기본 구현인 클래스를 JDBC 2.0 또는 3.0 드라이버 제품의 일부로 제공합니다.

이 절에서는 다음 내용을 설명합니다.

- 데이터 소스 객체의 등록 정보
- JDBC 자원 등록

데이터 소스 객체의 등록 정보

DataSource 객체에는 자신이 나타내는 실제 데이터 소스를 식별하고 설명하는 등록 정보가 있습니다. 이러한 등록 정보에는 데이터베이스 서버 위치, 데이터베이스 이름, 서버와 통신하는 데 사용되는 네트워크 프로토콜 등의 정보가 포함됩니다. DataSource 등록 정보는 JavaBeans 디자인 패턴을 따르며 대개 DataSource 객체가 배포될 때 설정됩니다.

여러 공급업체의 DataSource 구현 간에 일관성을 유지하기 위해 JDBC 2.0 API는 표준 등록 정보 집합과 각 등록 정보에 대한 표준 이름을 지정합니다.

DataSource 인터페이스를 구현하는 클래스의 인스턴스는 한 개의 특정 데이터 소스를 나타냅니다. 해당 인스턴스에 의해 설정되는 각 연결은 같은 데이터 소스를 참조합니다. 기본 DataSource 구현 시 DataSource.getConnection 메소드를 호출하면 DriverManager 기능에서 반환하는 연결 객체와 같은, 데이터 소스에 대한 물리적 연결 객체를 반환합니다.

JNDI는 응용 프로그램이 네트워크를 통해 원격 서비스를 찾아 액세스할 수 있는 일관된 방법을 제공합니다. 원격 서비스에는 메시징 서비스나 응용 프로그램 서비스를 비롯한 모든 엔터프라이즈 서비스가 포함되지만 JDBC 응용 프로그램은 주로 데이터베이스 서비스만 처리합니다. DataSource 객체를 만들어 JNDI 이름 지정 서비스에 등록하면 응용 프로그램은 JNDI API를 사용하여 해당 DataSource 객체에 액세스할 수 있습니다. 이 데이터 소스 객체는 자신이 나타내는 데이터 소스에 연결하는 데 사용될 수 있습니다.

마찬가지로 연결 풀링을 구현하는 DataSource 객체는 DataSource 클래스가 나타내는 특정 데이터 소스에 대한 연결을 생성합니다. 하지만 DataSource.getConnection 메소드가 반환하는 연결 객체는 물리적 연결이 아닌 PooledConnection 객체에 대한 핸들입니다. 응용 프로그램은 평소대로 연결 객체를 사용하며 일반적으로 이 연결 객체가 어떠한 방식으로건 달라졌다는 점을 인식하지 못합니다. 연결 풀링은 모든 연결과 마찬가지로 풀링된 연결이 항상 명시적으로 닫혀 있어야 한다는 점을 예외로 하면 응용 프로그램 코드에 전혀 영향을 미치지 않습니다. 응용 프로그램이 풀링된 연결을 닫으면 해당 연결은 재사용 가능한 연결 풀로 돌아갑니다. 다음에 DataSource.getConnection을 호출하면 이 풀링된 연결 중 하나가 사용 가능한 경우 이에 대한 핸들이 반환됩니다. 연결 풀링을 사용하면 연결이 요청될 때마다 물리적 연결을 새로 만들지 않아도 되므로 응용 프로그램을 훨씬 더 빠르게 실행할 수 있습니다.

분산 트랜잭션 환경에서 작동하도록 DataSource 클래스를 구현할 수 있습니다. 예를 들어, EJB 서버는 분산 트랜잭션을 지원하고 자신과 상호 작용하도록 구현되는 DataSource 클래스를 필요로 합니다. 이 경우 DataSource.getConnection 메소드는 분산 트랜잭션에서 사용할 수 있는 연결 객체를 반환합니다. 일반적으로 EJB 서버는 분산 트랜잭션과 연결 풀링에 대한 지원 기능을 제공합니다. 연결 풀링과 마찬가지로 트랜잭션 관리도 내부적으로 처리되므로 분산 트랜잭션을 사용하기 쉽습니다. 트랜잭션을 분산하는 경우(두 개 이상의 데이터 소스와 관련되는 경우) 응용 프로그램에서 commit이나 rollback 같은 트랜잭션 메소드를 호출할 수 없다는 점이 유일한 제한 사항입니다. 또한 연결을 auto-commit 모드에 둘 수도 없습니다. 이렇게 제한을 두는 이유는 트랜잭션 관리자가 분산 트랜잭션을 모르게 시작하고 끝내기 때문에 트랜잭션이 시작하거나 끝나는 시간에 영향을 미치는 작업을 응용 프로그램에서 할 수 없게 됩니다. Java 트랜잭션에 대한 자세한 내용은 9장, "트랜잭션 서비스 사용"을 참조하십시오.

JDBC 자원 등록

Sun ONE Application Server관리 인터페이스나 명령줄 인터페이스 중 하나를 사용하여 JDBC 자원을 등록할 수 있습니다.

이 절에서는 다음 내용을 설명합니다.

- 명령줄을 사용하여 자원 등록
- 관리 인터페이스를 사용하여 자원 등록

명령줄을 사용하여 자원 등록

명령줄 인터페이스를 사용하여 JDBC 자원을 등록하려면 다음 명령을 실행합니다.

```
./asadmin create-jdbc-resource
```

JDBC 자원을 등록하려면 XML 코드 단편에서 아래(`sun-server_7_0.dtd`에서 발췌)와 같이 몇 가지 속성을 지정해야 합니다.

```
<!-- JDBC javax.sql.DataSource resource definition -->

<!ELEMENT jdbc-resource (description?, property*)>

<!ATTLIST jdbc-resource  jndi-name CDATA     #REQUIRED

pool-name CDATA     #REQUIRED

enabled %boolean; 'true'>
```

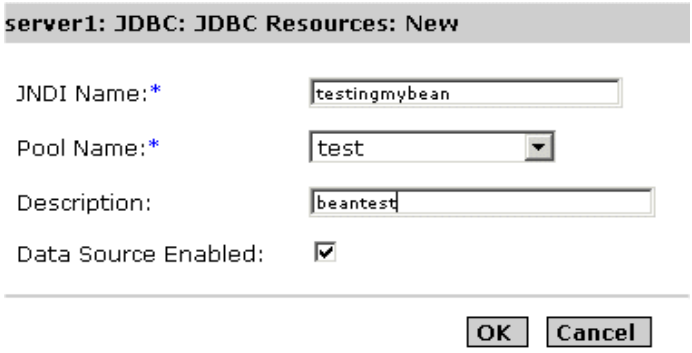
여기에서 지정하는 모든 내용은 응용 프로그램이 J2EE 응용 프로그램 내부에서 이 데이터 소스를 참조할 때 사용하는 심볼릭 이름입니다. `pool-name` 속성은 데이터베이스 연결의 모든 측면을 지정하는, 이름이 지정된 풀 정의를 가리킵니다. 관리자는 활성 속성을 사용하여 일부 자원을 해제할 수 있습니다.

관리 인터페이스를 사용하여 자원 등록

관리 인터페이스를 사용하여 데이터 소스를 등록하려면 다음 작업을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 JDBC 자원을 등록할 Sun ONE Application Server 인스턴스를 엽니다.
2. JDBC를 엽니다.
3. JDBC에서 "JDBC 자원"을 누릅니다.
4. 오른쪽 창에서 "새로 만들기"를 누릅니다. 그림 "새 JDBC 자원 만들기"에 표시된 것과 같이 새 JDBC 자원을 만들 수 있는 페이지가 오른쪽 창에 표시됩니다.

새 JDBC 자원 만들기



5. 만들고 있는 자원에 대한 JNDI 이름을 입력합니다.

   JDBC 자원은 JNDI 리포지토리에 저장되며 JNDI 이름을 사용하여 액세스합니다. JNDI 이름의 루트는 `Java:comp:env/`에 명시되어 있기 때문에 이름의 해당 부분을 지정할 필요가 없습니다. JDBC 자원(데이터 소스)은 '`jdbc`' 하위 컨텍스트에 저장하는 것이 좋으므로 JNDI 이름은 `jdbc/EmployeeDB_DS`와 비슷할 것입니다.

6. "풀 이름" 드롭다운 목록의 풀 이름 목록에서 새 데이터 소스에 대한 풀 이름을 선택합니다. 등록된 모든 연결 풀이 이 드롭다운 목록에 표시됩니다. 선택한 풀 이름은 데이터베이스 연결의 모든 측면을 지정하는, 이름이 지정된 풀 정의를 가리킵니다. 두 개 이상의 JDBC 자원이 하나의 풀 정의를 사용할 수 있습니다. JDBC 연결 풀 구성 방법에 대한 자세한 내용은 273페이지의 “관리 인터페이스를 사용하여 새 JDBC 연결 풀 만들기”를 참조하십시오.

7. 데이터 소스의 용도를 설명하는 간단한 설명을 "설명" 필드에 입력합니다. 설명은 250자를 초과할 수 없습니다.

8. "사용 가능" 확인란을 선택하여 데이터 소스를 활성화하거나 비활성화합니다. 데이터 소스가 활성화되어 있지 않으면 데이터베이스를 연결할 때 데이터 소스를 사용할 수 없습니다.

9. "확인"을 눌러 새 데이터 소스를 등록하거나 "취소"를 눌러 새 데이터 소스를 취소합니다. "취소"를 누르면 새 데이터 소스를 다시 만들 수 있는 주 "JDBC 자원" 페이지로 돌아갑니다.

JDBC 연결 정보

연결 객체는 데이터베이스와의 연결을 나타냅니다. 연결 세션에는 실행된 SQL 문과 연결을 통해 반환되는 결과가 포함됩니다. 단일 응용 프로그램은 단일 데이터베이스와 한 개 이상의 연결을 가질 수 있거나 많은 데이터베이스와 연결을 가질 수 있습니다.

사용자는 `Connection.getMetaData` 메소드를 호출하여 연결 객체의 데이터베이스에 대한 정보를 얻을 수 있습니다. 이 메소드는 데이터베이스 테이블, 지원하는 SQL 문법, 저장 프로시저, 이러한 연결의 기능 등의 정보가 있는 `DatabaseMetaData` 객체를 반환합니다.

응용 프로그램은 데이터 소스 객체가 만든 연결 객체를 사용합니다. 예외가 발생하더라도 연결을 확실히 닫을 수 있도록 응용 프로그램은 항상 "finally" 블록을 포함해야 합니다. 이는 유효한 연결이 사용 가능한 연결 풀에 항상 다시 반환되도록 하므로 연결 객체가 풀링된 연결인 경우 특히 중요합니다. 다음 코드 단편은 연결이 유효한 경우 연결을 닫는 finally 블록의 예이며 여기서 con은 연결 객체입니다.

```
finally{
        if (con != null) con.close();
}
```

다음 예에서와 같이 `finally` 블록은 try/catch 블록 다음에 표시되며 여기서 `ds`는 데이터 소스 객체입니다.

```
try {
        Connection con = ds.getConnection("user", "secret");
        // . . . code to do the application's work
} catch {
       // . . . code to handle an SQLException
} finally {
        if (con != null) con.close();
}
```

이 절에서는 다음 내용을 설명합니다.

- JDBC URL 정보
- JDBC 연결 풀 구성
- 연결 풀링 정보

- JDBC 연결 풀링 모니터링
- 연결 공유 정보

JDBC URL 정보

URL (Uniform Resource Locator)은 인터넷에서 자원을 찾는 데 필요한 정보를 제공합니다. URL은 인터넷상의 주소라고 할 수 있습니다.

JDBC URL은 적절한 드라이버가 데이터 소스를 인식하여 데이터 소스와의 연결을 설정할 수 있도록 데이터 소스를 식별하는 방법을 제공합니다. 특정 드라이버를 식별하는 JDBC URL을 실제로 결정하는 사람은 드라이버 작성자입니다. 사용자는 JDBC URL 구성 방법을 신경 쓸 필요가 없고 사용하고 있는 드라이버에서 제공하는 URL을 사용하기만 하면 됩니다. JDBC의 역할은 드라이버 작성자가 JDBC URL을 구성할 때 따라야 할 몇 가지 규칙을 권장하는 것입니다.

JDBC URL은 다양한 드라이버와 함께 사용되기 때문에 규칙도 매우 유연해야 합니다. 먼저, JDBC URL은 다양한 드라이버가 데이터베이스 이름을 지정할 때 다양한 체계를 사용하는 것을 허용합니다. 예를 들어, ODBC 하위 프로토콜은 URL이 속성 값을 갖는 것을 허용합니다. 하지만 속성 값을 요구하지는 않습니다.

두 번째로, JDBC URL은 드라이버 작성자가 JDBC URL 안에 필요한 모든 연결 정보를 인코딩할 수 있도록 허용합니다. 따라서 사용자가 시스템 관리 작업을 하지 않더라도 주어진 데이터베이스와 통신하려는 애플릿이 데이터베이스 연결을 열 수 있습니다.

세 번째로, JDBC URL은 주소 간접도를 허용합니다. 이는 JDBC URL이 네트워크 이름 지정 시스템에 의해 실제 이름으로 동적 변환되는 로컬 호스트나 데이터베이스 이름을 참조할 것임을 의미합니다. 따라서 시스템 관리자는 특정 호스트를 JDBC 이름의 일부로 지정하지 않아도 됩니다. 다양한 네트워크 이름 서비스가 많이 있으며 사용할 수 있는 서비스에 대한 제한은 없습니다.

JDBC URL의 표준 구문은 다음과 같습니다. 세 부분으로 되어 있으며 콜론으로 구분합니다.

`jdbc:<`*subprotocol*`>:<`*subname*`>`

JDBC URL의 세 부분은 다음과 같이 구성됩니다.

- jdbc-the protocol:

  JDBC URL의 프로토콜은 항상 jdbc입니다.
- `<subprotocol>`

드라이버의 이름 또는 하나 이상의 드라이버에서 지원하는 데이터베이스 연결 메커니즘의 이름입니다. 하위 프로토콜 이름의 두드러진 예는 ODBC로 ODBC 스타일 데이터 소스 이름을 지정하는 URL을 위해 예약되어 있습니다. 예를 들어, JDBC-ODBC 브리지를 통해 데이터베이스에 액세스하려면 `jdbc:odbc:fred` 같은 URL을 사용해야 합니다.

이 예에서 하위 프로토콜은 ODBC이며 하위 이름 fred는 로컬 ODBC 데이터 소스입니다.

네트워크 이름 서비스를 사용하려는 경우(따라서 JDBC URL의 데이터베이스 이름이 실제 이름을 가질 필요가 없는 경우), 이름 지정 서비스가 하위 프로토콜이 될 수 있습니다. 따라서 URL이 다음과 같을 수 있습니다.

```
jdbc:dcenaming:accounts-payable
```

이 예에서 URL은 로컬 DCE 이름 지정 서비스가 계정에 적합한 데이터베이스 이름을 실제 데이터베이스에 연결할 때 사용할 수 있는 좀 더 구체적인 이름으로 변환할 것을 지정합니다.

- `<subname>`:

  데이터 소스를 식별할 수 있는 방법입니다. 하위 프로토콜에 따라 하위 이름은 다양할 수 있으며 `sub-subname`을 비롯하여 드라이버 작성자가 선택할 수 있는 내부 구문을 갖습니다. `subname`의 핵심은 데이터 소스를 찾는 데 충분한 정보를 제공하는 것입니다. 이전 예의 경우 ODBC가 정보의 나머지 부분을 제공하므로 fred만 지정해도 충분합니다. 하지만 원격 서버의 데이터 소스는 좀 더 자세한 정보를 필요로 합니다. 예를 들어, 인터넷을 통해 데이터 소스에 액세스하는 경우 네트워크 주소가 `subname`의 일부로 JDBC URL에 포함되어야 하며 다음 표준 URL 명명 규칙을 따라야 합니다.

  //hostname`:`*port*`/`*subsubname*

  `dbnet`이 인터넷에서 호스트에 연결하기 위한 프로토콜이라고 가정하면 JDBC URL은 다음과 같을 것입니다.

  `jdbc:`*dbnet*`://`*wombat:356/fred*

JDBC 연결 풀 구성

Sun ONE Application Server에서 사용자는 이름이 지정된 JDBC 연결 풀을 만들 수 있습니다. JDBC 연결 풀은 연결 풀을 만드는 데 사용되는 등록 정보를 정의합니다. 풀 정의의 이름이 지정되며 여러 JDBC 자원을 구성하기 위해 정의를 다시 사용할 수 있습니다. 이름이 지정된 각 풀 정의는 서버 시작 시 물리적 풀을 인스턴스화합니다. 두 개 이상의 JDBC 자원이 같은 풀 정의를 가리키는 경우 이러한 자원은 런타임에 같은 연결 풀을 사용하게 됩니다.

다음 절에서 설명하는 대로 관리 인터페이스와 명령줄 인터페이스를 사용하여 JDBC 연결 풀을 만들고 구성할 수 있습니다.

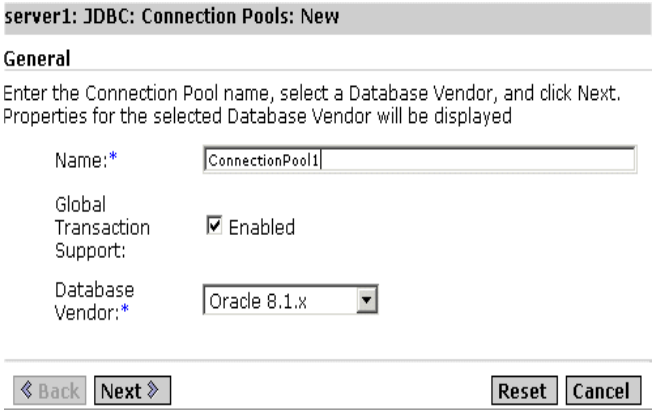
- 관리 인터페이스를 사용하여 새 JDBC 연결 풀 만들기
- 명령줄 인터페이스를 사용하여 새 JDBC 연결 풀 만들기
- 명령줄 인터페이스를 사용하여 JDBC 연결 풀 관리

관리 인터페이스를 사용하여 새 JDBC 연결 풀 만들기

관리 인터페이스를 사용하여 새 JDBC 연결 풀을 만들려면 다음을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 JDBC 연결 풀을 만들 Sun ONE Application Server 인스턴스를 엽니다.
2. Sun ONE Application Server에 나열된 J2EE 서비스 목록에서 "JDBC"를 선택하여 그 아래의 `ConnectionPools` 탭을 엽니다. 그림 “새 JDBC 연결 풀 만들기”와 같은 화면이 관리 인터페이스의 오른쪽 창에 표시됩니다.

새 JDBC 연결 풀 만들기



3. 만들고 있는 연결 풀의 JNDI 이름을 "이름" 필드에 입력합니다.
4. 새 연결 풀에 대해 전역 트랜잭션 지원을 활성화하려면 "전역 트랜잭션 지원 사용" 확인란을 선택합니다. 전역 트랜잭션에 참여할 수 있는 연결 풀을 XA 가능 연결 풀이라고 합니다.

5. "데이터베이스 공급업체" 드롭다운 목록에서 데이터베이스 공급업체를 선택하고 "다음"을 누릅니다. 다음과 같은 화면에서 연결 풀 설정을 구성해야 합니다.

연결 풀 설정 구성

풀 설정을 구성하려면 273페이지의 "관리 인터페이스를 사용하여 새 JDBC 연결 풀 만들기"의 1 단계-5 단계를 수행합니다. 5 단계에 설명한 대로 "다음"을 누르면 관리 인터페이스의 오른쪽 창에 새 페이지가 표시됩니다. 여기에는 다음과 같은 절이 포함되어 있습니다.

- 일반
- 등록 정보
- 풀 설정
- 연결 검증
- 트랜잭션 격리

다음 표의 지침에 따라 제공된 매개 변수의 값을 이 페이지의 "일반" 섹션에서 지정합니다.

일반 설정

매개 변수	설명
이름	연결 풀의 이름입니다.
데이터 소스 클래스 이름	데이터 소스 및/또는 XADataSource API를 구현하는 공급업체별 클래스 이름입니다.
설명	연결 풀에 대한 설명입니다.

이 페이지의 "등록 정보" 섹션에서 표준 및 독점 JDBC 연결 풀 등록 정보를 지정합니다. 이러한 등록 정보 중 많은 부분은 선택 사항입니다. 기본적으로 모든 표준 등록 정보의 이름이 제공됩니다. 어떤 표준 등록 정보 및 공급업체 관련 등록 정보가 필요한지 확인하려면 데이터베이스 공급업체의 설명서를 참조해야 합니다.

다음 표의 지침에 따라 제공된 매개 변수의 값을 이 창의 "풀 설정" 섹션에서 지정합니다.

연결 풀 설정

매개 변수	설명
고정 풀 크기	풀에서 유지해야 하는 최소 연결 수를 지정합니다. 요청된 스레드에 연결이 할당되면 해당 스레드가 풀에서 제거되므로 현재 풀 크기가 줄어듭니다. 고정 풀 크기는 서버 시작 시 풀에 추가될 항목 수도 참조합니다.
최대 풀 크기	지정된 시점에서 풀에서 사용할 수 있는 최대 연결 수를 지정합니다.
풀 크기 조정 개수	풀이 고정 풀 크기로 축소되면 일괄 처리의 크기가 조정됩니다. 이 값은 일괄 처리의 크기를 결정합니다. 이 값을 너무 크게 하면 연결 재순환이 너무 지체되며 너무 적게 하면 효율성이 떨어집니다. 풀 용량은 한 번에 연결을 한 개만 증가시키므로 이 필드는 풀 용량 증가에는 영향을 미치지 않습니다.
유휴 시간 초과(초)	풀에서 연결이 유휴 상태로 있을 수 있는 최대 시간(초)입니다. 이 시간이 지나면 풀 구현에서 해당 연결을 닫을 수 있습니다.
최대 대기 시간	연결 시간이 초과될 때까지 호출자가 대기해야 하는 시간입니다. 기본 대기 시간은 long이므로 호출자는 오랫 동안 대기할 수 있습니다.

이 창의 "연결 검증" 및 "트랜잭션 격리"에서 다음 표의 지침에 따라 연결 풀의 검증 방법과 트랜잭션 격리 방법을 선택합니다.

연결 검증 및 트랜잭션 격리

매개 변수	설명
연결 검증 필요	이 필드를 선택하면 연결이 응용 프로그램으로 전달되기 전에 검증됩니다. 따라서 네트워크 실패나 데이터베이스 서버 크래시로 인해 데이터베이스를 사용할 수 없는 경우 응용 프로그램 서버가 데이터베이스 연결을 자동으로 다시 설정합니다. 연결 검증을 수행하면 추가 오버헤드가 발생하며 성능이 약간 저하됩니다.

연결 검증 및 트랜잭션 격리

매개 변수	설명
검증 방법	데이터베이스 연결을 검증하기 위해 응용 프로그램 서버에서 사용할 수 있는 방법으로는 세 가지가 있습니다. 적절한 방법을 결정하려면 사용 중인 데이터베이스의 기능에 대해 잘 알고 있어야 합니다. 세 가지 검증 방법은 다음과 같습니다. • `auto-commit, meta-data` - con.getAutoCommit() 및 con.getMetaData() 메소드는 일반적으로 연결을 검증하는 데 사용됩니다. 안타깝게도 많은 JDBC 드라이버는 이러한 호출 결과를 캐시하므로 항상 신뢰할만한 검증 결과를 제공하지는 않습니다. 이러한 호출의 캐시 여부를 결정하려면 공급업체에 확인해야 합니다. • `table`: 이 메소드를 사용하려면 응용 프로그램 서버가 사용자 지정 테이블에 대해 쿼리를 수행해야 합니다. 실제 쿼리는 "select (count *) from <table-name>"입니다. 행은 없어도 되지만 테이블은 반드시 있어야 하고 액세스할 수 있어야 합니다. 많은 수의 행이 있는 기존 테이블이나 이미 자주 액세스하는 테이블은 사용하지 마십시오.
테이블 이름	"검증 방법" 드롭다운 목록에서 마지막 검증 옵션, `table`을 선택하는 경우 여기에서 테이블 이름을 지정합니다.
모든 연결 끊기	풀의 모든 연결을 끊으려면 이 확인란을 선택하고 한 연결만 끊으려면 연결을 다시 설정하십시오. 이 확인란을 선택하지 않으면 연결은 사용될 때만 개별적으로 다시 설정됩니다.
트랜잭션 격리	이 연결의 트랜잭션 격리 수준을 선택할 수 있습니다. 이 확인란을 선택하지 않으면 풀은 JDBC 드라이버에서 제공하는 기본 격리 수준으로 실행됩니다.
격리 수준 보장	격리 수준이 지정된 경우에만 적용할 수 있습니다. 따라서 풀에서 가져온 연결은 항상 동일한 격리 수준을 갖게 됩니다. 예를 들어, 마지막으로 사용했을 때 연결의 격리 수준을 프로그래밍 방식으로 변경한 경우 (예: `con.setTransactionIsolation`) 이 메커니즘은 연결의 격리 수준을 지정된 격리 수준으로 다시 변경합니다.

명령줄 인터페이스를 사용하여 새 JDBC 연결 풀 만들기

이 절에서는 예를 통해 명령줄 인터페이스를 사용하여 JDBC 연결 풀을 만드는 방법을 설명합니다.

다음 표는 연결 풀을 만드는 데 필요한 모든 옵션(예: 서버 이름, 암호)을 나열합니다. 다음 표에는 샘플 값이 사용되었습니다. 이 절에서 설명한 명령을 실행하기 전에 Sun ONE Application Server 설치와 관련된 매개 변수를 준비하는 것이 좋습니다.

명령줄 인터페이스를 사용하여 JDBC 연결 풀을 만드는 데 필요한 옵션

필요한 옵션 설명	**샘플 값**
응용 프로그램 서버 관리 사용자 이름	*admin*
응용 프로그램 서버 관리 암호	*adminadmin*
응용 프로그램 서버 관리 포트	*8888*
응용 프로그램 서버 컴퓨터 이름	*sas.sun.com*
응용 프로그램 서버 인스턴스 이름	*server1*
연결 풀의 데이터 소스 클래스 이름	*oracle.jdbc.xa.client.OracleXADataSource* 주: 연결 풀을 만들 데이터베이스의 데이터 소스 클래스 이름을 사용합니다. 이 예에서 사용되는 데이터베이스는 Oracle입니다.
Jdbc 자원 설명 샘플	*Jdbc 자원*
연결 풀 설명 샘플	*Jdbc 연결 풀*
Jdbc 자원 이름	*jdbc/SampleJdbcResource*
연결 풀 이름	*SampleJdbcConnectionPool*
데이터베이스 사용자 이름	*oracle*
데이터베이스 이름	*oracle*
Jdbc 연결 URL	*jdbc:oracle:thin:@oracleserver.sun.com:1521:ORA*

다음 예에서는 "명령줄 인터페이스를 사용하여 JDBC 연결 풀을 만드는 데 필요한 옵션" 표에 나열된 변수를 사용합니다.

예 1:

이 예에서는 `SampleJdbcConnectionPool`이라는 JDBC 연결 풀을 만듭니다. 이 예에서는 JDBC 연결 풀을 만들기 위해 다음과 같이 두 단계 과정을 사용합니다.

- 1단계 - 연결 풀 만들기
- 2단계 - 인스턴스에 변경 사항 적용

1단계 - 연결 풀 만들기

다음은 JDBC 연결 풀을 만들기 위한 명령줄 인터페이스 구문입니다.

`asadmin create-jdbc-connection-pool --user` *admin_user* `[--password` *admin_password*`] [--host` *localhost*`] [--port` *4848*`] [--secure | -s] [--instance` *instancename*`] --datasourceclassname` *classname* `[--restype` *res_type*`] [--steadypoolsize` *8*`] [--maxpoolsize` *32*`] [--maxwait` *60000*`] [--poolresize` *2*`] [--idletimeout` *300*`] [--isolationlevel` *isolation_level*`] [--isisolationguaranteed] [--isconnectvalidatereq=false] [--validationmethod` *auto-commit*`] [--validationtable` *tablename*`] [--failconnection=false] [--description` *text*`] [--property (name=value)[:name=value]*]` *connectionpool_id*

예를 들어, 다음 명령은 `SampleJdbcConnectionPool`이라는 연결 풀을 만듭니다.

asadmin create-jdbc-connection-pool --user admin --password adminadmin --host sas.sun.com --port 8888 --instance server1 --restype javax.sql.XADataSource --datasourceclassname oracle.jdbc.xa.client.OracleXADataSource --description "Sample Jdbc Connection Pool" --property User="oracle":Password="oracle":URL="jdbc\:oracle\:thin\:@oracleserver.sun.com\:1521\:ORA" SampleJdbcConnectionPool

---

주 새 연결 풀에 대해 "전역 트랜잭션 지원"을 활성화하려면 `--restype javax.sql.XADataSource`를 설정합니다. URL 등록 정보에서 콜론(`:`)을 (`\:`)으로 바꾸십시오.

JDBC 연결 풀이 성공적으로 만들어지면 다음 메시지가 표시됩니다.

```
Created the JDBC connection pool resource with id =
SampleJdbcConnectionPool
```

2단계 - 인스턴스에 변경 사항 적용

이제 JDBC 연결 풀을 성공적으로 만들었으므로 이러한 변경 사항을 Sun ONE Application Server의 현재 인스턴스에 적용해야 합니다.

다음은 Sun ONE Application Server의 인스턴스에 변경 사항을 적용하기 위한 구문입니다.

`asadmin reconfig --user` *admin_user* `[--password` *admin_password*`] [--host` *localhost*`] [--port` *adminport*`] [--secure | -s] [--discardmanualchanges=false|--keepmanualchanges=false]` *instancename*

예를 들어, 다음 명령은 Sun ONE Application Server의 인스턴스인 *server1*에 변경 사항을 적용합니다.

`asadmin reconfig --user` *admin* `--password` *adminadmin* `--host` *sas.sun.com* `--port` *8888* *server1*

Sun ONE Application Server의 인스턴스에 변경 사항이 적용되면 다음 메시지가 표시됩니다.

```
Successfully reconfigured
```

명령줄 인터페이스를 사용하여 JDBC 연결 풀 관리

이 절에서 설명하는 대로 명령줄 인터페이스를 사용하여 JDBC 연결 풀과 해당 등록 정보를 관리할 수 있습니다.

연결 풀 나열. 다음 명령은 2단계에서 사용한 Sun ONE Application Server의 인스턴스인 *server1*에 대해 만든 모든 연결 풀을 나열합니다.

`asadmin list-jdbc-connection-pools --user` *admin* `--password` *adminadmin* `--host` *sas.sun.com* `--port` *8888* *server1*

JDBC 연결 풀 등록 정보 변경. 다음과 같이 `maxPoolSize` 등록 정보 같은 JDBC 연결 풀 등록 정보를 변경할 수 있습니다.

1. JDBC 연결 풀 속성 `maxPoolSize`에 지정한 값을 얻으려면 다음 명령을 실행합니다.

   ```
   asadmin get -u admin -w adminadmin -H sas.sun.com -p 8888
   server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSize
   ```

   이 명령을 실행하면 다음과 같은 결과가 표시됩니다.

   ```
   server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSiz
   e = 32
   ```

   다음 명령을 실행하여 MaxPoolSize 값을 80으로 변경합니다.

   ```
   asadmin set -u admin -w adminadmin -H sas.sun.com -p 8888
   server1.jdbc-connection-pool.SampleJdbcConnectionPool.maxPoolSize="80"
   ```

   표시된 대로 값을 지정하면 다음 메시지가 표시됩니다.

   ```
   Attribute maxPoolSize set to 80
   ```

2. 다음 명령을 사용하여 Sun ONE Application Server의 인스턴스에 변경 사항을 적용합니다.

   ```
   asadmin reconfig --user admin --password adminadmin --host sas.sun.com
   --port 8888 server1
   ```

사용자 등록 정보 변경. 다음 샘플 코드에서 "`User`" 등록 정보를 `oracle`에서 `System`으로 변경할 수 있습니다.

```
asadmin create-jdbc-connection-pool --user admin --password adminadmin
--host sas.sun.com --port 8888 --instance server1 --restype
javax.sql.XADataSource --datasourceclassname oracle.jdbc.xa.client.OracleXADataSource
--description "Sample Jdbc Connection Pool" --property
User="oracle":Password="oracle":URL="jdbc\:oracle\:thin\:@oracleserver.sun.com\:1521\:
ORA" SampleJdbcConnectionPool
```

1. 사용자 등록 정보를 변경하려면 다음 명령을 실행합니다.

   ```
   asadmin set -u admin -w adminadmin -H sas.sun.com -p 8888
   server1.jdbc-connection-pool.SampleJdbcConnectionPool.property.User="System"
   ```

   사용자 이름이 *Oracle*에서 *System*으로 변경됩니다.

2. 사용자 이름을 변경하고 다음 명령을 실행하여 변경 사항을 적용합니다.

   ```
   asadmin reconfig --user admin --password adminadmin --host
   sas.sun.com --port 8888 server1
   ```

SampleJdbcResource라는 JDBC 자원 만들기. 아래에 설명한 대로 JDBC 자원을 만들 수 있습니다. 다음은 JDBC 자원을 만들기 위한 구문입니다.

```
asadmin create-jdbc-resource --user admin_user [--password
admin_password] [--host localhost] [--port 4848] [--secure | -s]
[--instance instancename] --connectionpoolid id [--enabled=true]
[--description text] [--property (name=value)[:name=value]*] jndiname
```

1. SampleJdbcResource라는 JDBC 자원을 만들려면 다음 명령을 실행합니다.

   ```
   asadmin create-jdbc-resource --user admin --password adminadmin
   --host sas.sun.com --port 8888 --instance server1 --description "Sample
   Jdbc Resource" --connectionpoolid SampleJdbcConnectionPool
   jdbc/SampleJdbcResource
   ```

   이 명령을 실행하면 JDBC 자원이 만들어지고 다음 메시지가 표시됩니다.

   ```
   Created the external JDBC resource with jndiname =
   jdbc/SampleJdbcResource
   ```

2. 다음 명령을 실행하여 Sun ONE Application Server의 인스턴스에 변경 사항을 적용합니다.

   ```
   asadmin reconfig --user admin --password adminadmin --host sas.sun.com
   --port 8888 server1
   ```

3. 다음 명령을 실행하여 *server1* 인스턴스의 JDBC 자원을 모두 나열합니다.

   ```
   asadmin list-jdbc-resources --user admin --password adminadmin
   --host sas.sun.com --port 8888 server1
   ```

연결 풀링 정보

응용 프로그램은 JNDI를 사용하여 데이터 소스를 먼저 조회하여 연결을 얻을 수 있습니다. 이 작업을 위한 샘플 코드 단편이 아래에 나와 있습니다.

```
InitialContext ctx  = new InitialContext();

DataSource ds = (DataSource)
ctx.lookup("java:comp/env/jdbc/employee_ds");
```

DataSource를 얻은 다음 응용 프로그램 구성 요소는 J2EE 배포 설명자의 <res-auth> 요소에 설정된 값에 따라 두 가지 방법으로 연결을 얻을 수 있습니다. 이 요소의 값이 Container인 경우 응용 프로그램은 ds.getConnection() 메소드를 사용하여(즉, 사인 온 정보를 지정하지 않음) 연결을 얻을 수 있습니다. 그렇지 않은 경우 응용 프로그램은 사인 온 정보를 입력해야만 ds.getConnecion(*userName, password*)같은 자원 관리자로부터 연결을 얻을 수 있습니다.

getConnection()에 대한 모든 요청은 풀에서 처리됩니다. server.xml에서 설명된 일련의 매개 변수에 따라 JDBC 연결 풀이 만들어집니다. 풀이 만들어지면 풀에서 처음에 사용할 수 있는 연결 개수가 지정됩니다. 따라서 ds.getConnection() 요청은 풀에서 현재 사용할 수 있는 연결에 따라 처리될 수 있습니다. 다음 요청은 풀이 비어 있음을 발견하고(이전 연결 중에서 풀로 반환된 연결이 없는 경우) 풀에 지정된 최대 연결 개수 제한에 따라 연결 수를 늘리게 됩니다. 풀 구현은 만들어진 연결 개수를 추적합니다. getConnection() 요청이 풀이 비어 있음을 발견하거나 만든 연결 수가 풀의 최대 연결 수와 같은 경우 현재 요청이 차단됩니다. 연결 공유가 가능하지 않은 경우에만 차단되며 연결이 풀에 반환될 때까지 계속됩니다.

데이터베이스가 크래시되어 시작되더라도 연결 풀이 계속 실행되는 한 연결 풀은 계속 제대로 작동합니다. 274페이지의 “연결 풀 설정 구성”에 설명한 대로 연결 검증을 활성화한 경우에만 작동합니다.

"검증 유형" 드롭다운 목록에서 선택한 값에 따라 다음 매개 변수가 풀 구현 프로그램에 의해 실행됩니다.

- auto-commit을 연결 검증 유형으로 선택한 경우 시스템에서는 conn.getAutoCommit() 메소드를 수행하여 연결이 유효한지 여부를 검사합니다. 메소드가 SQLException을 발생시키지 않으면 연결은 유효한 것으로 간주됩니다. auto-commit은 이 매개 변수의 기본 옵션입니다.
- meta-data를 연결 검증 유형으로 선택한 경우 conn.getMetaData() 메소드를 사용하여 연결의 메타데이터를 검사합니다. 메소드가 SQLException을 발생시키지 않으면 연결은 유효한 것으로 간주됩니다.
- table을 연결 검증 유형으로 선택한 경우 "Select * From <table-name>" 쿼리가 실행됩니다. 이 호출이 SQLException을 발생시키지 않으면 연결은 유효한 것으로 간주됩니다.

fail-all-connections 등록 정보를 활성화한 경우 풀의 연결 중 어느 하나라도 유효하지 않은 것으로 확인되면 모든 연결이 닫히고 다시 설정됩니다. 그렇지 않은 경우 무효 및 재설정이 지연되었다가 개별 연결을 사용할 때 발생합니다.

또한 풀 구현은 풀에서 사용 가능한 연결을 모두 재순환할 수 있는 기능을 제공합니다. 연결이 지정된 유휴 기간을 초과하여 유휴 상태로 있는 경우 연결이 닫히며 풀 크기는 고정 풀 크기로 바뀝니다. 풀이 과도하게 유휴 상태에 있는 경우 컨테이너가 비활성화된 연결을 다시 설정하려고 시도할 수 있으므로 항상 풀을 사용 가능한 고정 개수로 채워 놓아야 합니다. 최대 풀 크기에 상대적으로 고정 풀 크기를 설정할 것인지 여부를 결정할 때에는 이 내용을 유념해야 합니다.

JDBC 연결 풀링 모니터링

풀 크기 구성이 효율적으로 작동하는지 확인하기 위해 풀 작업을 주기적으로 모니터링할 수 있습니다. 다음 표는 모니터링할 수 있는 모든 JDBC 연결 풀링 매개 변수를 나열합니다.

모니터링을 활성화하는 방법과 모니터링할 수 있는 속성은 다음 릴리스에서 개선될 것입니다.

모니터링을 위한 JDBC 연결 풀 매개 변수

속성 이름	데이터 유형	설명
`total-threads-waiting`	정수	JDBC 연결을 대기하는 총 스레드 수
`total-outbound-connections`	정수	총 JDBC 연결 검증 실패 횟수
`total-connections-timed-out`	정수	시간이 초과된 총 연결 요청 수

연결 공유 정보

J2EE 응용 프로그램에서 얻은 여러 연결이 동일한 자원 관리자를 사용하는 경우 풀 구현 시 같은 트랜잭션 범위 내에서 연결 공유를 제공합니다. 트랜잭션 범위라는 용어를 이해하려면 다음 예를 참조하십시오.

Bean_A는 트랜잭션(Tx1)을 시작하여 연결을 얻습니다. 그런 다음 Bean_A는 같은 트랜잭션(Tx1)의 Bean_B에서 메소드를 호출합니다. Bean_B는 같은 `DataSource`로부터 같은 사인 온 정보를 사용하여 연결을 얻으므로 Bean_A가 트랜잭션을 완료한 경우에만 같은 연결을 공유할 수 있습니다. 또한 자원 공유 범위가 J2ee 배포 설명자에서 `Shareable`로 설정된 경우에만 연결을 공유할 수 있습니다.

연결 공유를 원하지 않는 경우 배포 설명자에서 자원 공유 범위를 `Unshareable`로 설정해야 합니다. Sun ONE Application Server는 성능 향상을 위해 연결 공유를 제공합니다.

JDBC 트랜잭션 정보

트랜잭션은 실행되고 완료된 다음 완결되거나 롤백된 하나 이상의 문으로 구성됩니다. `commit`이나 `rollback` 메소드가 호출되면 현재 트랜잭션이 끝나고 다른 트랜잭션이 시작됩니다.

일반적으로 새 연결 객체는 기본적으로 auto-commit 모드에 배치되며 이는 문이 완료되면 해당 문에 대해 `commit` 메소드가 자동으로 호출될 것임을 의미합니다. 이 경우 각 문은 개별적으로 완결되므로 한 트랜잭션은 한 문으로 구성됩니다. auto-commit 모드가 비활성화되어 있는 경우 `commit`이나 `rollback` 메소드가 명시적으로 호출될 때까지 트랜잭션은 종료되지 않습니다. 따라서 commit이나 rollback를 마지막으로 호출한 이후에 실행된 모든 문을 포함하게 됩니다. 두 번째의 경우 트랜잭션의 모든 문은 그룹으로 완결되거나 롤백됩니다.

`commit` 메소드는 SQL 문이 데이터베이스에 변경한 내용을 영구적으로 반영하며 트랜잭션에서 설정한 잠금을 해제합니다. `rollback` 메소드는 이러한 변경 사항을 무시합니다.

한 트랜잭션에 두 개의 업데이트가 있는 경우 두 번째 업데이트도 함께 변경되지 않는다면 한 업데이트에 대해서만 변경 사항을 적용하고 싶지 않을 수 있습니다. 이렇게 하려면 auto-commit을 비활성화하고 두 업데이트를 한 트랜잭션으로 그룹화하면 됩니다. 두 가지 업데이트가 모두 성공한 경우 `commit` 메소드가 호출되어 두 가지 업데이트가 모두 영구적으로 반영됩니다. 한 업데이트나 두 가지 업데이트가 모두 실패하면 `rollback` 메소드가 호출되어 업데이트가 실행되기 이전의 값을 복구합니다. 대부분의 JDBC 드라이버는 트랜잭션을 지원합니다.

`javax.sql` 패키지의 클래스와 인터페이스는 연결 객체가 분산 트랜잭션(두 개 이상의 DBMS 서버에 대한 연결과 관련된 트랜잭션)의 일부가 될 수 있도록 합니다. 분산 트랜잭션에 참여하려면 중간 계층 서버의 분산 트랜잭션 인프라와 작업할 수 있도록 구현된 `DataSource` 객체가 연결 객체를 만들어야 합니다. `DriverManager`가 만든 연결 객체와는 달리 `DataSource` 객체가 만든 연결 객체는 기본적으로 auto-commit 모드가 비활성화되어 있습니다. 반면 `DataSource` 객체의 표준 구현은 `DriverManager` 클래스가 만드는 것과 똑같은 연결 객체를 만듭니다.

연결 객체가 분산 트랜잭션의 일부인 경우 트랜잭션 관리자는 commit 또는 rollback 메소드가 호출된 시점을 확인합니다. 따라서 연결 객체가 분산 트랜잭션에 참여하고 있는 경우 응용 프로그램은 `Connection.commit`이나 `Connection.rollback` 메소드를 호출하거나 연결의 auto-commit 모드를 설정하는 등 연결이 시작되거나 끝나는 시기에 영향을 미치는 작업을 하면 안 됩니다. 이러한 작업은 트랜잭션 관리자가 분산 트랜잭션을 처리하는 데 방해가 됩니다.

JavaMail 자원 정보

JavaMail API를 사용하면 메시지 저장소에 포함된 전자 우편 메시지에 액세스할 수 있으며 메시지 전송을 사용하여 전자 우편 메시지를 만들고 전송할 수 있습니다. 인터넷 표준 MIME 메시지에 대한 지원 기능도 포함되어 있습니다. 메시지 저장소와 전송에 대한 액세스는 특정 저장소와 전송 프로토콜을 지원하는 프로토콜 공급자를 통해 이루어집니다. JavaMail API 사양에서는 특정 프로토콜 공급자를 요구하지는 않지만 JavaMail에는 IMAP 메시지 저장소 공급자와 SMTP 메시지 전송 공급자가 포함되어 있습니다.

JavaMail API는 우편 시스템을 구성하는 객체를 정의하는 추상 클래스 집합을 제공합니다. API는 메시지, 저장소, 전송 등과 같은 클래스를 정의합니다. 새 프로토콜을 제공하고 필요한 경우 기능을 추가하기 위해 API를 확장하고 하위 클래스로 나눌 수 있습니다. 또한 API는 추상 클래스의 구체적인 하위 클래스를 제공합니다. `MimeMessage`와 `MimeBodyPart` 같은 하위 클래스는 널리 사용되는 인터넷 우편 프로토콜을 구현합니다.

JavaMail API는 IMAP, MAPI, CMC, c-client 및 기타 전자 우편 메시징 시스템 API와 많은 상호 작용을 합니다. JavaMail API는 다양한 메시지 저장소와 다양한 메시지 형식 및 다양한 메시지 전송 같은 여러 메시징 시스템 구현을 지원합니다. JavaMail API는 클라이언트 응용 프로그램의 API를 정의하는 기본 클래스와 인터페이스 집합을 제공합니다. 개발자는 IMAP, POP3, SMTP 같은 특정 메시징 시스템 구현을 위해 JavaMail 클래스를 하위 클래스로 나눌 수 있습니다.

이 절에서는 다음 내용을 설명합니다.

- JavaMail 메시지 처리 과정 정보
- JavaMail의 구조적 구성 요소 정보
- JAF (JavaBeans Activation Framework) 정보
- JavaMail 구성 매개 변수 정보
- JavaMail 세션 참조용 J2EE 배포 설명자

- Sun ONE Application Server 배포 설명자의 항목
- 새 JavaMail 세션 만들기
- 고급 자원 등록 정보 구성

JavaMail 메시지 처리 과정 정보

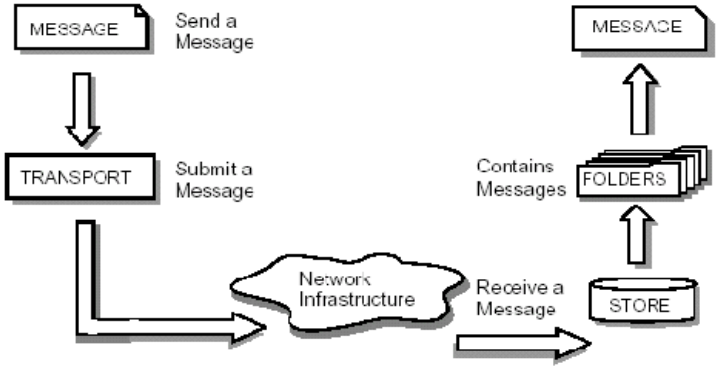
JavaMail API는 일반 클라이언트 응용 프로그램을 위한 표준 우편 처리 과정을 구성하는 다음 기능을 수행합니다.

- Content-Type 헤더 필드에 지정된 몇 가지 알려진 데이터 유형의 데이터 블록과 헤더 속성 집합으로 이루어진 우편 메시지를 만듭니다. JavaMail은 Part 인터페이스와 메시지 클래스를 사용하여 우편 메시지를 정의합니다. 그리고 JAF 정의 DataHandler 객체를 사용하여 메시지에 있는 데이터를 포함합니다.
- 사용자를 인증하고 메시지 저장소와 전송에 대한 액세스를 제어하는 세션 객체를 만듭니다.
- 메시지를 받는 사람 목록으로 보냅니다.
- 메시지 저장소로부터 메시지를 검색합니다.
- 검색된 메시지에서 고급 명령을 실행합니다. view와 print 같은 고급 명령은 JAF 인식 JavaBeans를 통해 구현되도록 만들어졌습니다.

주 현재 JavaMail 프레임워크는 메시지 전달, 보안, 연결 해제 작업, 디렉토리 서비스 또는 필터 기능을 지원하는 메커니즘은 정의하지 않습니다.

다음 그림은 JavaMail API가 메시지를 처리하는 과정을 보여줍니다.

JavaMail API 의 메시지 처리 과정



JavaMail API는 `javax.mail.Session`을 만드는 정적 팩토리 메소드를 사용하여 구성되었습니다. Sun ONE Application Server는 JNDI를 사용하여 세션 객체를 요청하고 `resource-ref` 요소를 사용하여 배포 설명자에 세션 객체에 대한 요청을 나열합니다. JavaMail API 세션 객체는 자원 팩토리로 간주됩니다.

메시지 전송은 `javax.mail.internet.InternetAddress` 유형의 주소와 `javax.mail.internet.MimeMessage` 유형의 메시지를 처리할 수 있어야 합니다. `javax.mail.Transport` 클래스의 send 메소드를 사용하여 이러한 메시지를 전송할 수 있도록 기본 메시지 전송을 올바르게 구성해야 합니다.

JavaMail API의 추상 계층은 모든 우편 시스템이 지원하는 우편 처리 기능을 지원하기 위해 클래스, 인터페이스 및 추상 메소드를 선언합니다. 추상 계층을 구성하는 API 요소는 표준 데이터 유형을 지원하고 필요한 경우 메시지 액세스와 메시지 전송 프로토콜과 상호 작용하기 위해 하위 클래스로 나뉘고 확장되도록 설계되었습니다.

인터넷 구현 계층은 인터넷 표준 - `RFC822`와 `MIME`을 사용하여 추상 계층의 일부를 구현합니다.

JavaMail의 구조적 구성 요소 정보

이 절에서는 다음 항목을 통해 JavaMail 구조를 구성하는 주요 구성 요소에 대해 설명합니다.

- 메시지 클래스
- 메시지 저장 및 검색
- 메시지 작성 및 전송

메시지 클래스

메시지 클래스는 우편 메시지 내용과 속성 집합을 정의하는 추상 클래스입니다. 메시지 클래스의 속성은 주소 지정 정보를 지정하고 내용 유형을 비롯한 내용 구조를 정의합니다. 내용은 실제 데이터를 둘러싸는 DataHandler 객체로 표현됩니다.

메시지 클래스는 Part 인터페이스를 구현합니다. Part 인터페이스는 메시지 객체가 전달하는 데이터 내용을 정의하고 형식을 지정하며 우편 시스템과 성공적으로 인터페이스하는 데 필요한 속성을 정의합니다. 메시지 클래스는 메시지 전송 시스템을 통해 메시지를 라우팅할 때 필요한 보낸 사람, 받는 사람, 제목, 회신 및 기타 속성을 추가합니다. 메시지 객체가 폴더에 포함되어 있는 경우 해당 객체는 자신과 관련된 일련의 태그를 갖습니다. JavaMail은 특정 메시징 구현을 지원하는 메시지 하위 클래스를 제공합니다.

메시지 내용은 메시지 객체 내에 캡슐화된 바이트 모음이거나 바이트 모음에 대한 참조입니다. JavaMail은 메시지 내용의 데이터 유형이나 형식을 알지 못합니다. 메시지 객체는 중간 계층 - JAF (JavaBeans Activation Framework)를 통해 해당 내용과 상호 작용합니다. 이렇게 분리되어 있기 때문에 메시지 객체는 임의의 내용을 처리한 다음 같은 API 메소드를 호출하고 적절한 전송 프로토콜을 사용하여 내용을 전송할 수 있습니다. 메시지 받는 사람은 일반적으로 해당 내용의 데이터 유형 및 형식과 그 처리 방법을 알고 있습니다.

또한 JavaMail API는 각 Bodypart가 자신의 고유 속성 및 내용 집합을 정의하는 multipart 메시지 객체를 지원합니다.

메시지 저장 및 검색

메시지는 폴더 객체에 저장됩니다. 폴더 객체는 메시지와 하위 폴더를 포함할 수 있으므로 트리형 폴더 계층을 제공합니다. 폴더 클래스는 메시지를 불러오고 추가하며 복사하고 삭제하는 메소드를 선언합니다. 또한 폴더 객체는 Event Listener로 등록된 구성 요소로 이벤트를 보낼 수 있습니다.

저장소 클래스

저장소 클래스는 메시지와 함께 폴더 계층을 보관하는 데이터베이스를 정의합니다. 또한 저장소 클래스는 폴더에 액세스하여 폴더에 저장된 메시지를 검색하는 액세스 프로토콜을 지정합니다. 또한 폴더를 불러오고 연결을 닫기 위해 데이터베이스에 대한 연결을 설정하는 메소드를 제공합니다. 메시지 액세스 프로토콜(IMAP, POP3 등)을 구현하는 서비스 공급자는 저장소 클래스를 하위 클래스로 나누는 작업부터 시작합니다. 사용자는 일반적으로 특정 저장소 구현에 연결하여 우편 시스템과의 세션을 시작합니다.

메시지 작성 및 전송

클라이언트는 적절한 메시지 하위 클래스를 인스턴스화하여 새 메시지를 만듭니다. 받는 사람 주소와 제목 같은 속성을 설정하고 내용을 메시지 객체에 삽입합니다. 마지막으로 Transport.send 메소드를 호출하여 메시지를 보냅니다. 전송 클래스는 메시지를 대상 주소로 라우팅하는 전송 에이전트를 모델화합니다. 이 클래스는 메시지를 받는 사람 목록으로 보내는 메소드를 제공합니다. 메시지 객체와 함께 Transport.send 메소드를 호출하면 대상 주소에 따라 적절한 전송 방법을 식별합니다.

세션 클래스

세션 클래스는 우편 활성 클라이언트와 네트워크 사이의 인터페이스를 정의하는 전역 및 사용자별 우편 관련 등록 정보를 정의합니다.

JavaMail 시스템은 세션 객체를 사용하여 특정 등록 정보를 설정하고 얻습니다. 또한 세션 클래스는 데스크탑 응용 프로그램이 공유할 수 있는 기본 인증 세션 객체를 제공합니다. 세션 클래스는 구체적인 마지막 클래스이며 하위 클래스로 나눌 수 없습니다. 세션 클래스는 또한 특정 액세스 및 전송 프로토콜을 구현하는 저장소 및 전송 객체에 대한 팩토리 역할을 합니다. 세션 객체에 대해 적절한 팩토리 메소드를 호출하면 클라이언트는 특정 프로토콜을 지원하는 저장소 및 전송 객체를 얻을 수 있습니다.

JAF (JavaBeans Activation Framework) 정보

JavaMail은 메시지 데이터를 캡슐화하고 이 데이터와 상호 작용할 명령을 처리하기 위해 JAF (JavaBeans Activation Framework)를 사용합니다. 메시지와의 상호 작용은 JavaMail API가 제공하지 않는 JAF 인식 JavaBeans를 통해 이루어져야 합니다.

JavaBeans Activation Framework 표준 확장을 통해 Java 기술을 사용하는 개발자는 표준 서비스를 이용하여 임의 데이터 부분의 유형을 결정하고 이에 대한 액세스를 캡슐화하여 사용 가능한 작업을 알아내며 이러한 작업을 수행하기 위해 적절한 Bean을 인스턴스화할 수 있습니다. 예를 들어, 브라우저에서 JPEG 이미지를 가져온 경우 이 프레임워크는 브라우저가 해당 데이터 스트림을 해당 유형의 JPEG 이미지로 식별할 수 있게 해주며 브라우저는 그 이미지를 조작하거나 볼 수 있는 객체를 찾아 인스턴스화할 수 있습니다.

JavaBeans Activation Framework API는 다양한 MIME 데이터 유형을 지원합니다. JavaMail API는 다음 표에서 설명하는 Java 프로그래밍 언어 유형에 해당하는 다음 MIME 데이터 유형에 대한 `javax.activation.DataContentHandlers`를 포함해야 합니다.

JavaMail API MIME 데이터 유형 대 Java 유형 매핑

MIME 유형	Java 유형
Text/Plain	java.lang.String
Multipart/	javax.mail.internet.MIME.Multipart
Message/rfc822	javax.mail.internet.MIME.Message

JavaBeans Activation Framework는 MIME 데이터 유형에 대한 지원을 Java 플랫폼에 통합합니다. MIME 바이트 스트림은 javax.activation.DataContentHandlerobjects를 사용하여 Java 프로그래밍 언어 객체와 상호 변환할 수 있습니다. JavaBeans 구성 요소에 대해 데이터 보기나 편집 같은 MIME 데이터에 대한 작업을 지정할 수 있습니다. 또한 JavaBeans Activation Framework는 파일 이름 확장자를 MIME 유형에 매핑할 수 있는 메커니즘을 제공합니다. JavaMail API는 JavaBeans Activation Framework를 사용하여 전자 우편 메시지에 포함된 데이터를 처리합니다. 전자 우편을 복잡하게 사용하는 응용 프로그램의 경우 JavaBeans Activation Framework를 직접 사용해야 할 수도 있지만 일반 J2EE 응용 프로그램은 JavaBeans Activation Framework를 직접 사용할 필요가 없습니다.

JavaMail 구성 매개 변수 정보

다음 구성 매개 변수는 Sun ONE Application Server의 JavaMail 자원에서 사용됩니다. 이러한 구성 매개 변수는 이름, 값 쌍으로 이루어지며 `server.xml` 파일의 `mail-resource` 요소에서 읽을 수 있습니다.

- JNDI 이름

  JNDI 이름은 이 우편 자원이 J2EE 응용 프로그램에서 참조하는 데 사용하는 이름을 지정합니다.

- Enabled

  `enabled` 구성 매개 변수는 이 우편 자원을 JNDI 트리에 게시하여 참조할 수 있는지 여부를 지정합니다. J2EE 응용 프로그램이 비활성화된 자원을 참조하면 `NameNotFoundException` 예외를 받게 됩니다.

- store-protocol

  기본 메시지 액세스 프로토콜을 지정합니다. `Session.getStore()` 메소드는 이 프로토콜을 구현하는 저장소 객체를 반환합니다. 클라이언트는 이 등록 정보를 무시하고 `Session.getStore (String protocol)` 메소드를 사용하여 프로토콜을 명시적으로 지정할 수 있습니다.

- store-protocol 클래스

  위에 지정된 저장소 프로토콜을 구현하는 클래스 이름을 지정합니다. 이 클래스의 기본값은 `com.sun.mail.imap.IMAPStore`입니다.

- transport-protocol

  기본 전송 프로토콜을 지정합니다. `Session.getTransport()` 메소드는 이 프로토콜을 구현하는 전송 객체를 반환합니다. 클라이언트는 이 등록 정보를 무시하고 `Session.getTransport (String protocol)` 메소드를 사용하여 프로토콜을 명시적으로 지정할 수 있습니다.

- transport-protocol 클래스

  위에서 지정한 전송 프로토콜을 구현하는 클래스 이름을 지정합니다. 이 클래스의 기본값은 `com.sun.mail.smtp.SMTPTransport`입니다.

- host

  기본 메일 서버를 지정합니다. 프로토콜 관련 호스트 등록 정보가 없는 경우 저장소 및 전송 객체의 연결 메소드는 이 등록 정보를 사용하여 대상 호스트를 찾습니다.

- user

  메일 서버에 연결할 때 입력할 사용자 이름을 지정합니다. 프로토콜 관련 사용자 이름 등록 정보가 없는 경우 저장소 및 전송 객체의 연결 메소드는 이 등록 정보를 사용하여 사용자 이름을 얻습니다.

- from

  현재 사용자의 반송 주소를 지정합니다. 현재 사용자의 전자 우편 주소를 지정하기 위해 `InternetAddress.getLocalAddress` 메소드에서 사용합니다.

- debug

  초기 디버그 모드를 지정합니다. 이 등록 정보를 true로 설정하면 디버그 모드가 설정되며 false로 설정하면 디버그 모드가 해제됩니다.

- mail-<protocol>-host

  프로토콜 관련 기본 메일 서버를 지정합니다. 따라서 `mail.host` 등록 정보는 무시됩니다. store-protocol 속성의 값에 따라 이 등록 정보를 설정할 수 있습니다. store-protocol의 Imap 또는 POP 값에 대해서는 `mail.imap.host`나 `mail.pop3.host`란 이름을 각각 사용하여 등록 정보를 추가해야 합니다. 특정 등록 정보의 값은 우편 시스템 구성에 적합하게 설정되어야 합니다. 예를 들어, store-protocol을 IMAP로 설정하면 등록 정보 이름 `mail-imap-host`는 `spaceduck.acme.com`이라는 값을 가질 것입니다.

- mail-<protocol>-user

  메일 서버에 연결하기 위한 프로토콜 관련 기본 사용자 이름을 지정합니다. 따라서 `mail.user` 등록 정보는 무시됩니다. 선택한 store-protocol 속성에 따라 이 등록 정보는 `mail.imap.user`나 `mail.pop3.user`가 될 수 있습니다. 예를 들어, store-protocol을 IMAP로 설정하면 등록 정보 이름 `mail-imap-user`는 `fredbloggs`라는 값을 가질 것입니다.

JavaMail 세션 참조용 J2EE 배포 설명자

JavaMail 자원을 서버에 등록하면 JNDI 조회를 사용하여 모든 J2EE 응용 프로그램 구성 요소에서 해당 자원을 참조할 수 있습니다. 자원 관리자 연결 팩토리를 참조하는 응용 프로그램을 배포하려면 구성 요소 공급자가 표준 J2EE 1.3 배포 설명자에서 모든 자원 관리자 연결 팩토리 참조를 선언해야 합니다.

JavaMail 참조에 필요한 모든 J2EE1.3 설명자 요소를 아래에서 설명합니다.

```
<resource-ref>
    <description>
        우편을 보내는 데 사용하는 JavaMail 자원
     </description>
     <res-ref-name>mail/MyMailSession</res-ref-name>
     <res-type>javax.mail.Session</res-type>
     <res-auth>Container</res-auth>
     <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

Sun ONE Application Server 배포 설명자의 항목

우편 자원을 참조하는 배포된 각 구성 요소에 대해 배포자는 구성 요소에서 사용되는 자원 이름을 데이터 소스가 이름 지정 서비스에 등록한 실제 jndi 이름에 매핑해야 합니다. 배포 도구는 배포자가 이 매핑을 쉽게 할 수 있도록 도움을 주어야 합니다. 이 매핑은 Sun ONE Application Server 관련 xml에 등록되어 있습니다. 매핑을 포함하는 Sun ONE Application Server의 특정 XML 단편이 아래에 나와 있습니다.

```
<resource-ref>
   <res-ref-name>mail/MyMailSession</res-ref-name>
   <jndi-name>mail/Session</jndi-name>
</resource-ref>
```

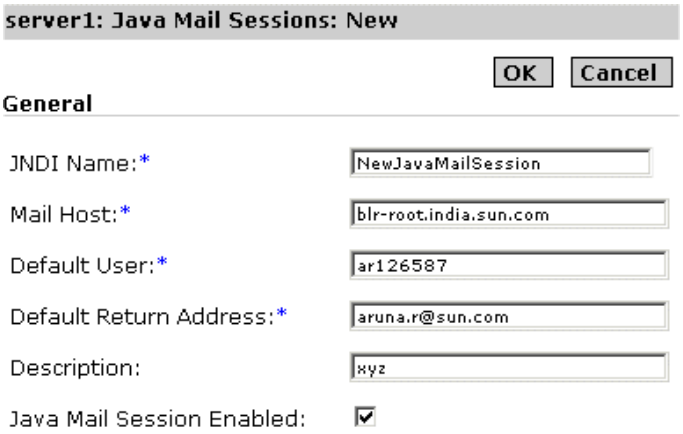
새 JavaMail 세션 만들기

관리 인터페이스를 사용하여 JavaMail 세션을 구성할 수 있습니다. 새 JavaMail 세션을 만들고 구성하려면 다음을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 새 JavaMail 세션을 만들 Sun ONE Application Server 인스턴스를 확장합니다.

2. "JavaMail 세션"을 누릅니다. 관리 인터페이스의 오른쪽 창에 있는 그림 “JavaMail 세션 구성”에 표시된 다음 창이 표시됩니다.

JavaMail 세션 구성



3. 만들고 있는 JavaMail 세션에 대한 JNDI 이름을 "JNDI 이름" 텍스트 필드에 입력합니다. JavaMail 자원을 서버에 등록하면 JNDI 조회를 사용하여 모든 J2EE 응용 프로그램 구성 요소에서 해당 자원을 참조할 수 있습니다.
4. 기본 메일 서버의 DNS 이름을 "메일 호스트" 텍스트 필드에 지정합니다. 프로토콜 관련 호스트 등록 정보가 없는 경우 저장소 및 전송 객체의 연결 메소드는 이 등록 정보를 사용하여 대상 호스트를 찾습니다.
5. 메일 서버에 연결할 때 제공할 사용자 이름을 "기본 사용자" 텍스트 필드에 지정합니다. 프로토콜 관련 사용자 이름 등록 정보가 없는 경우 저장소 및 전송 객체의 연결 메소드는 이 등록 정보를 사용하여 사용자 이름을 가져옵니다.
6. 현재 사용자의 기본 반송 주소를 "기본 반송 주소"에 지정합니다. 기본 주소의 형식은 username@host이어야 합니다.
7. "설명" 필드에 이 JavaMail 세션에 대한 설명을 입력합니다.
8. 만든 JavaMail 세션을 활성화하려면 "JavaMail 세션 사용" 확인란을 선택합니다.
9. 구성한 새 JavaMail 세션을 저장하려면 "확인"을 누릅니다.

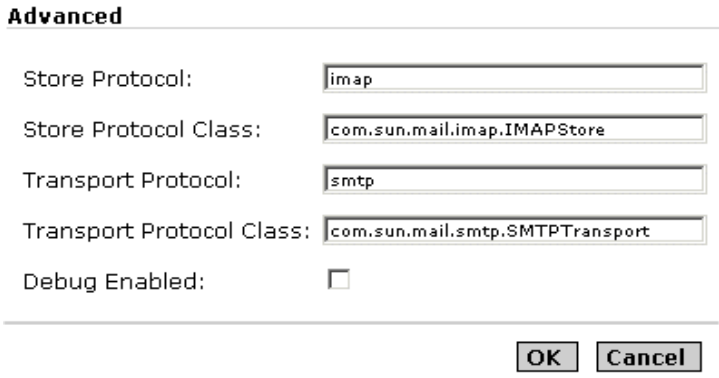
고급 자원 등록 정보 구성

관리 인터페이스를 사용하여 새 JavaMail 세션에 대한 몇 가지 추가 등록 정보를 구성할 수 있습니다. 등록 정보 이름과 값 쌍은 사용 중인 우편 프로토콜에 따라 달라집니다. 이러한 등록 정보를 server.xml 파일에서 직접 지정할 수도 있습니다.

추가 등록 정보를 구성하려면 다음을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 JavaMail 세션을 수정할 Sun ONE Application Server 인스턴스를 확장합니다.
2. "JavaMail 세션"을 누릅니다. “새 JavaMail 세션 만들기”에서 설명한 주요 구성 섹션 아래에 있는 관리 인터페이스의 오른쪽 창에 그림 “JavaMail 세션의 추가 자원 구성”에 있는 대로 다음 창이 표시됩니다.

JavaMail 세션의 추가 자원 구성



3. 이러한 특정 JavaMail 세션에 사용할 저장소 프로토콜(예: POP3, IMAP)을 "저장소 프로토콜" 텍스트 필드에 지정합니다.
4. 표시한 저장소 프로토콜의 클래스 이름을 예에 표시된 대로 "저장소 프로토콜 클래스" 텍스트 필드에 지정합니다.
5. 이러한 JavaMail의 특정 세션에 사용할 전송 프로토콜(예: SMTP)을 "전송 프로토콜" 텍스트 필드에 입력합니다.
6. 이 세션에 대해 표시한 전송 프로토콜의 클래스 이름을 위의 예에서 표시한 대로 "전송 프로토콜 클래스" 텍스트 필드에 지정합니다.

7. 이러한 JavaMail의 특정 세션 디버깅을 활성화하려면 "디버그 사용" 확인란을 선택합니다. 이 확인란을 활성화하면 디버그 모드가 설정됩니다.
8. 추가 등록 정보 구성을 저장하려면 "확인"을 누릅니다.

우편 자원 구성에 대한 전체 예가 아래에 나와 있습니다.

```
<mail-resource
    jndi-name = "mail/Session"
    enabled = "true"
    store-protocol = "imap"
    store-protocol-class = "com.sun.mail.imap.IMAPStore"
    transport-protocol = "smtp"
    transport-protocol-class = "com.sun.mail.smtp.SMTPTransport"
    host = "gopostal.acme.com"
    user = "kingkong"
    from = "kingkong@acme.com"
    debug = "false">
    <property name = "mail-imap-host" value = "spaceduck.acme.com"/>
    <property name = "mail-imap-user" value = "fredbloggs"/>
</mail-resource>
```

JMS 서비스 사용

Sun ONE Application Server는 메시징 작업에 대한 JMS (Java Message Service) API (Application Programming Interface)를 지원합니다. JMS는 Java 응용 프로그램의 분산 환경 내 메시지 작성, 보내기, 수신 및 읽기를 위한 일반적인 방법을 제공하는 프로그래밍 인터페이스 모음입니다.

특히 JMS는 J2EE (Java 2 Enterprise Edition) 응용 프로그램에서 비동기 메시징을 수행하는 표준 기반 방법입니다. 따라서 J2EE 구성 요소(웹 구성 요소 또는 EJB (Enterprise JavaBeans))는 JMS API를 사용하여 MDB (Message-Driven Bean)라는 특별한 EJB가 비동기적으로 사용하는 메시지를 보낼 수 있습니다.

Sun ONE Application Server는 일반적으로 JMS 메시징, 특히 MDB를 지원하며, JMS 사양을 구현하는 메시징 미들웨어인 JMS 공급자를 필요로 합니다. Sun ONE Application Server는 원시 JMS 공급자로 Sun ONE Message Queue (MQ) 버전 3.01을 사용합니다.

MQ는 투명한 JMS 메시징 지원을 제공하는 Sun ONE Application Server에 완벽하게 통합되어 있습니다. 이 지원(Sun ONE Application Server에서는 *JMS 서비스*로 통함)에는 최소한의 관리만 필요합니다.

이 장에서는 Sun ONE Message Queue를 통해 제공되는 기본 제공 JMS 서비스 이해 및 관리에 필요한 정보에 대해 설명합니다. 여기에서는 다음 항목에 대해 설명합니다.

- JMS 정보
- 기본 제공 JMS 서비스
- 기본 제공 JMS 서비스 관리

JMS 정보

JMS 사양은 분산 엔터프라이즈 메시징을 지원하는 프로그램 인터페이스 모음을 설명합니다. 엔터프라이즈 메시징 시스템을 사용하면 독립 분산 구성 요소 또는 응용 프로그램이 메시지를 통해 통신할 수 있습니다. 동일한 시스템, 동일한 네트워크 또는 인터넷을 통해 느슨하게 연결된 구성 요소는 메시징을 사용하여 데이터를 전달하고 각각의 기능을 조정합니다.

JMS는 신뢰할 수 있는 비동기 메시지 전달을 제공하여 엔터프라이즈 규모의 메시징을 지원합니다.

신뢰할 수 있는 전달. 한 구성 요소에서 다른 구성 요소로 전달되는 메시지가 네트워크 또는 시스템 장애로 인해 손실되면 안됩니다. 즉, 시스템은 메시지 전달을 보장할 수 있어야 합니다.

비동기 전달. 많은 수의 구성 요소가 동시에 메시지를 교환할 수 있게 하고 고밀도 처리량을 지원하려면, 메시지 전송 여부가 사용자의 즉시 수신 준비 상태에 좌우되면 안됩니다. 사용자가 다른 작업을 수행 중이거나 오프라인 상태일 경우, 시스템은 메시지를 일단 전송한 후 사용자가 준비되었을 때 수신할 수 있도록 해야 합니다. 이것을 비동기 메시지 전달이라고 하며, 보통 축적 전송(store-and-forward) 메시징이라고 합니다.

다음 항목은 JMS 개념 및 용어에 대한 간단한 개요입니다.

- 기본 메시징 시스템 개념
- JMS 사양
- Message-Driven Bean

JMS의 전체 설명은 다음

> `http://java.sun.com/products/jms/docs.html`의 `JMS 1.0.2` 사양을 참조하십시오.

기본 메시징 시스템 개념

몇 가지 기본 개념, 특히 JMS는 엔터프라이즈 메시징 시스템의 기초가 됩니다. 여기에서는 다음 주제로 설명합니다.

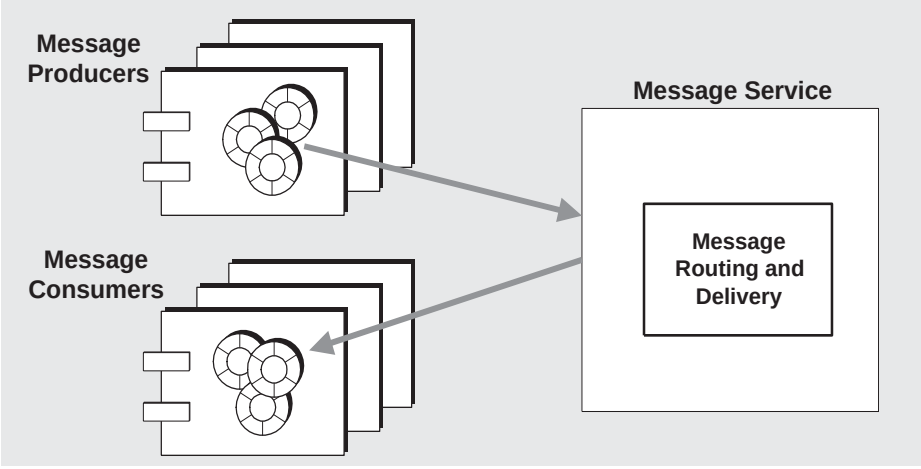
메시지

메시지는 특정 형식의 데이터(메시지 본문), 대상, 수명과 같은 메시지의 특성이나 등록정보 또는 메시징 시스템이 결정한 기타 특성을 설명하는 메타데이터(메시지 헤더)로 구성되어 있습니다.

메시지 서비스 구조

메시징 시스템의 기본 구조는 그림 "메시지 서비스 구조에 표시되어 있습니다. 이것은 일반 메시지 서비스를 사용하여 메시지를 교환하는 메시지 생성자와 메시지 사용자로 구성되어 있습니다. 일반적으로 메시지 생성자 및 사용자를 몇 명이든 메시징 구성 요소에 저장할 수 있습니다. 메시지 생성자는 메시징 서비스에 메시지를 보냅니다. 그러면 메시지 서비스는 메시지 라우팅 및 전달 구성 요소를 사용하여 메시지를 메시지에 인터레스트를 등록한 한 명 이상의 메시지 사용자에게 전달합니다. 메시지 라우팅 및 전달 구성 요소는 메시지가 모든 해당 사용자에게 전달되도록 보장할 책임이 있습니다.

메시지 서비스 구조



메시지 전달 모델

생성자 및 사용자 사이에는 일대일, 일대다, 다대다 관계 등 많은 관계가 존재합니다. 예를 들어, 다음과 같이 메시지를 전달할 수 있습니다.

- 한 생성자가 한 사용자에게
- 한 생성자가 많은 사용자에게
- 많은 생성자가 한 사용자에게
- 많은 생성자가 많은 사용자에게

이 관계들은 *점대점* 및 *게시/가입* 메시징의 두 메시지 전달 모델로 압축되는 경우가 많습니다. 점대점 전달 모델의 핵심은 특정 생성자가 보내 특정 사용자가 수신하는 메시지에 있습니다. 게시/가입 전달 모델의 핵심은 여러 명의 생성자가 보내 여러 명의 사용자가 수신하는 메시지에 있습니다. 이 메시지 전달 모델은 중복할 수 있습니다.

역사적으로 메시징 시스템은 이 두 전달 모델을 다양하게 결합하여 지원해 왔습니다. JMS API는 점대점 및 게시/가입 전달 모델을 모두 지원하는 일반 프로그래밍 접근 방법을 사용하도록 설계되었습니다.

JMS 사양

JMS는 메시지 구조, 프로그래밍 모델, 제공된 규칙 및 의미 모음을 지정합니다.

JMS 메시지 구조

JMS 사양에 따라, 메시지는 헤더, 등록 정보(헤더의 확장자에 따라 달라질 수 있음) 및 본문으로 구성됩니다.

헤더. 헤더는 메시지의 JMS 특성인 메시지 대상, 지속성 여부, 유지 시간 및 우선순위를 지정합니다. 이 특성은 메시징 시스템의 메시지 전달 방법을 관리합니다.

등록 정보. 등록 정보는 선택적이어서, 응용 프로그램이 다양한 선택 기준에 따라 메시지 필터에 사용할 값을 제공합니다. 등록 정보는 선택적입니다.

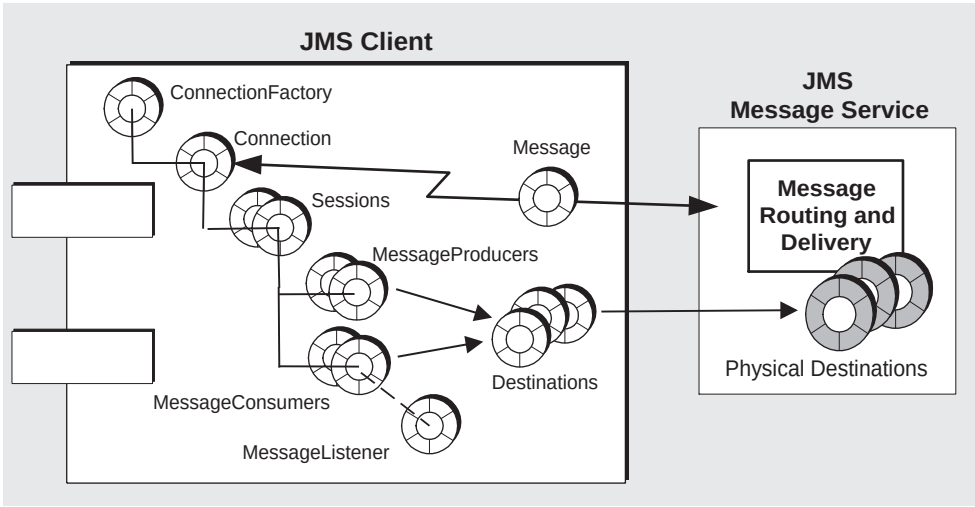
메시지 본문. 메시지 본문에는 교환될 실제 데이터가 포함됩니다. JMS는 여섯 가지 본문 유형을 지원합니다.

JMS 프로그래밍 모델

JMS 프로그래밍 모델에서 JMS 클라이언트(구성 요소 또는 응용 프로그래밍)는 JMS 메시지 서비스를 사용하여 메시지를 교환합니다. 메시지 생성자는 메시지 서비스에 메시지를 보내고, 메시지 사용자는 그 서비스에서 메시지를 받습니다. 이 메시징 작업은 JMS API를 구현하는 객체 모음(JMS 공급자가 제공)을 사용하여 수행됩니다. 그림 "JMS 프로그래밍 객체"은 메시지 전달 프로그램에 사용되는 JMS 객체를 표시합니다.

JMS 프로그래밍 모델의 경우, JMS 클라이언트는 `ConnectionFactory` 객체를 사용하여 JMS 메시지 서비스에 송수신되는 메시지와의 연결을 작성합니다. 연결은 JMS 클라이언트의 메시지 서비스에 대한 활성화 연결입니다. 클라이언트의 통신 자원 및 인증은 모두 연결이 작성될 때 할당됩니다.

JMS 프로그래밍 객체



연결은 세션 작성에 사용됩니다. 세션은 메시지 생성 및 사용의 단일 스레드 컨텍스트입니다. 메시지를 송수신할 메시지 생성자 및 사용자 작성에 사용됩니다. 세션은 수많은 확인 옵션 또는 트랜잭션(배포된 트랜잭션 관리자가 관리할 수 있는)을 통해 신뢰할 수 있는 전달을 제공합니다.

JMS 클라이언트는 `MessageProducer`를 사용하여 API에 대상 객체로 표시된 지정된 물리적 대상에 메시지를 보냅니다. 메시지 생성자는 생성자가 물리적 대상에 전송한 모든 메시지를 관리하는 기본 전달 모드(지속성 대 비지속성 메시지), 우선 순위 및 지속 시간을 지정할 수 있습니다.

마찬가지로, JMS 클라이언트는 `MessageConsumer`를 사용하여 API를 사용하여 대상 객체로 다시 제공된 지정된 물리적 대상에서 메시지를 수신합니다. 메시지 사용자는 메시지의 동기 또는 비동기 사용을 지원할 수 있습니다. 비동기 사용은 MessageListener를 사용자로 등록하여 수행합니다. 클라이언트는 세션 스레드가 `MessageListener` 객체의 `onMessage()` 메소드를 불러낼 때 메시지를 사용합니다.

관리 객체: 공급자 독립성

그림 301페이지의 표 "JMS 프로그래밍 모델"에는 JMS 공급자의 JMS 메시지 서비스 구현 방법의 세부 정보에 따라 두 개의 객체가 표시되어 있음. 연결 팩토리 객체는 공급자가 메시지를 전달할 때 사용하는 기반 프로토콜 및 메커니즘, 특정 명명 규칙 및 공급자가 사용한 물리적 대상의 용량에 따라 달라집니다.

일반적으로 이 공급자별 특성은 JMS API 구현의 세부 정보에 따라 JMS 클라이언트 코드를 작성합니다. 하지만 공급자에 독립적인 JMS 클라이언트 코드를 작성하려면, JMS 사양은 클라이언트 코드로 직접 인스턴스를 작성하지 않고 표준 방법으로 액세스되는 공급자별 객체(관리 객체라고 함)가 필요합니다.

관리 객체는 공급자별 구현 및 구성 정보를 캡슐화합니다. 관리 객체는 관리자가 작성 및 구성하고, 이름 서비스에 저장되며, 클라이언트 응용 프로그램이 표준 JNDI 조회 코드를 통해 액세스합니다. 이런 방법으로 관리 객체를 사용하여 공급자에 독립적인 JMS 클라이언트 코드를 작성합니다.

JMS는 연결 팩토리 및 대상이라는 두 가지 관리 객체 일반 유형을 제공합니다. 두 가지 모두 공급자별 정보를 캡슐화하지만, JMS 클라이언트 내에서 서로 매우 다르게 사용됩니다. 연결 팩토리는 메시지 서버에 연결을 작성할 때 사용되는 반면, 대상 객체는 JMS 메시지 서비스를 사용하여 물리적 대상을 식별할 때 사용됩니다.

주 Sun ONE Application Server의 컨텍스트에서, JMS 관리 객체는 응용 프로그램 서버 자원과 마찬가지로 JMS 자원으로 간주됩니다.

Message-Driven Bean

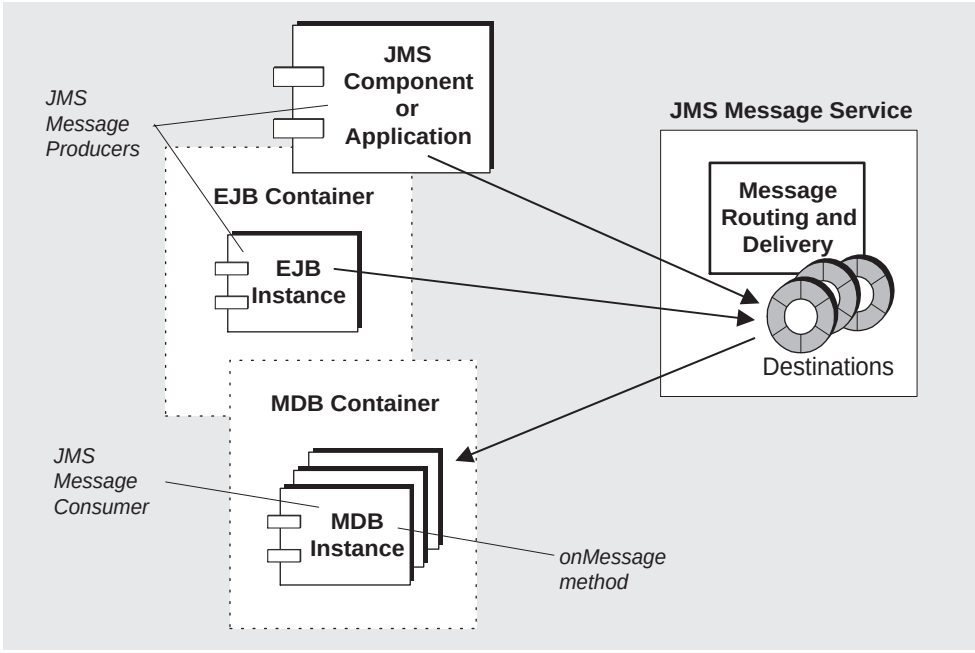
그림 301페이지의 표 "JMS 프로그래밍 모델"에 소개된 일반 JMS 클라이언트 프로그래밍 모델과 함께, J2EE 응용 프로그램의 컨텍스트에 사용되는 JMS API의 추가 특별 적용이 있습니다. 이 특별한 JMS 클라이언트를 *Message-Driven Bean*라고 부르며, EJB 2.0 사양에 지정된 EJB 구성 요소 패밀리 중 하나입니다(`http://java.sun.com/products/ejb/docs.html`).

다른 EJB 구성 요소(Session Bean 및 Entity Bean)는 동기적으로만 호출될 수 있기 때문에 Message-Driven Bean이 필요해 집니다. 이 때문에 이런 Bean의 메소드를 호출하면 메소드가 완료될 때까지 자원이 차단됩니다. 이 EJB 구성 요소의 경우 표준 EJB 인터페이스를 통해서만 액세스되기 때문에 메시지를 비동기적으로 받는 메커니즘을 가지고 있지 않습니다.

하지만 비동기 메시징은 많은 엔터프라이즈 응용 프로그램에 필요합니다. 이 때문에 메시지 생성과 밀접하게 연관되어 있지 않은 채 메시지를 수신하고 사용할 수 있는 EJB 구성 요소가 필요해지고 있습니다.

MDB는 특수 EJB 컨테이너(이것이 지원하는 구성 요소의 배포 서비스를 제공하는 소프트웨어 환경)가 지원하는 특수 EJB 구성 요소입니다.

MDB 메시지 사용자



Message-driven Bean. MDB는 JMS `MessageListener` 인터페이스를 구현하는 JMS 메시지 사용자입니다. MDB의 `onMessage` 메소드(MDB 개발자가 작성합니다)는 MDB 컨테이너가 메시지를 수신할 때 호출됩니다. `onMessage` 메소드는 JMS MessageListener 객체의 `onMessage` 메소드처럼 메시지를 사용합니다. MDB는 단일 대상에서 온 메시지를 사용할 수 있습니다. 메시지는 "MDB 메시지 사용자"의 그림에서 표시한 것처럼 단독 JMS 클라이언트 응용 프로그램, 웹 구성 요소 또는 기타 EJB 구성 요소가 생성할 수 있습니다.

MDB 컨테이너. MDB는 MDB 인스턴스 작성 및 메시지의 비동기 사용 설정을 담당한 특수 EJB 컨테이너에 의해 지원됩니다. 이것은 메시지 서비스(인증 포함)와의 연결 설정, 주어진 대상과 관련된 세션 풀 작성 및 메시지가 세션 풀 및 관련 MDB 인스턴스의 풀 사이에 수신될 때의 메시지 배포 관리와 관련되어 있습니다. 컨테이너는 MDB 인스턴스의 lifecycle을 제어하기 때문에 MDB 인스턴스 풀을 수신 메시지 로드에 맞추어 관리합니다.

MDB와 관련된 것으로, 메시지 사용 설정시 컨테이너가 사용하는 관리 객체의 JNDI 조회 이름을 지정하는 배포 설명자가 있습니다. 연결 팩토리 및 대상. 배포 설명자도 배포 도구가 컨테이너 구성에 사용할 수 있는 다른 정보를 포함할 수 있습니다. 이런 컨테이너는 각각 단일 MDB의 인스턴스를 지원합니다.

Sun ONE Application Server에 대한 EJB 컨테이너의 MDB 설정 구성 정보는 211페이지의 표 “Message-Driven Bean에 대한 정보”를 참조하십시오.

기본 제공 JMS 서비스

일반적으로 JMS 메시징 지원, 특히 MDB 지원은 Sun ONE Application Server에 구축되어 있습니다. 이 지원은 Sun ONE Message Queue와 Sun ONE Application Server 원시로 제공하는 기본 제공 JMS 서비스의 밀접한 통합을 통해 지원됩니다.

여기에서는 기본 제공 JMS 서비스 이해에 필수적인 다음 항목들을 설명합니다.

- Sun ONE Message Queue (MQ) 정보
- MQ와 통합 Sun ONE Application Server

기본 제공 JMS 서비스 관리에 대한 정보는 312페이지의 표 “기본 제공 JMS 서비스 관리”를 참조하십시오.

Sun ONE Message Queue (MQ) 정보

Sun ONE Message Queue (MQ)는 JMS 열기 표준을 구현하는 엔터프라이즈 메시징 시스템입니다. 이것이 JMS 공급자입니다.

MQ 제품에는 신뢰할 수 있는 비동기 메시징에 필요한 JMS 사양의 최소 요구 사항을 넘어서는 기능이 있습니다. 이 기능 중 일부(중앙 집중식 관리, 조정 가능한 성능, 여러 메시징 전송 지원, 사용자 인증 및 권한 부여)는 Sun ONE Application Server에 통합된 MQ Platform Edition에서 사용할 수 있습니다. MQ Enterprise Edition으로 업그레이드하면 추가 기능(확장 가능한 메시지 서버 및 보안 메시징)을 사용할 수 있습니다.

그림 306페이지의 표 "MQ 시스템 구조"에 표시된 것처럼, MQ 메시징 시스템은 신뢰할 수 있는 메시지 전달을 위해 공동 작업하는 여러 부분으로 구성되어 있습니다.

MQ 메시징 시스템의 주요 부분은 다음과 같습니다.

- MQ 메시지 서버
- MQ 클라이언트 런타임
- MQ 관리 객체
- MQ 관리 도구

이것은 다음 항목을 통해 간략히 소개됩니다. MQ 메시징 시스템의 자세한 전체 설명은

http://docs.sun.com/의 MQ *Administrator's Guide*를 참조하십시오.

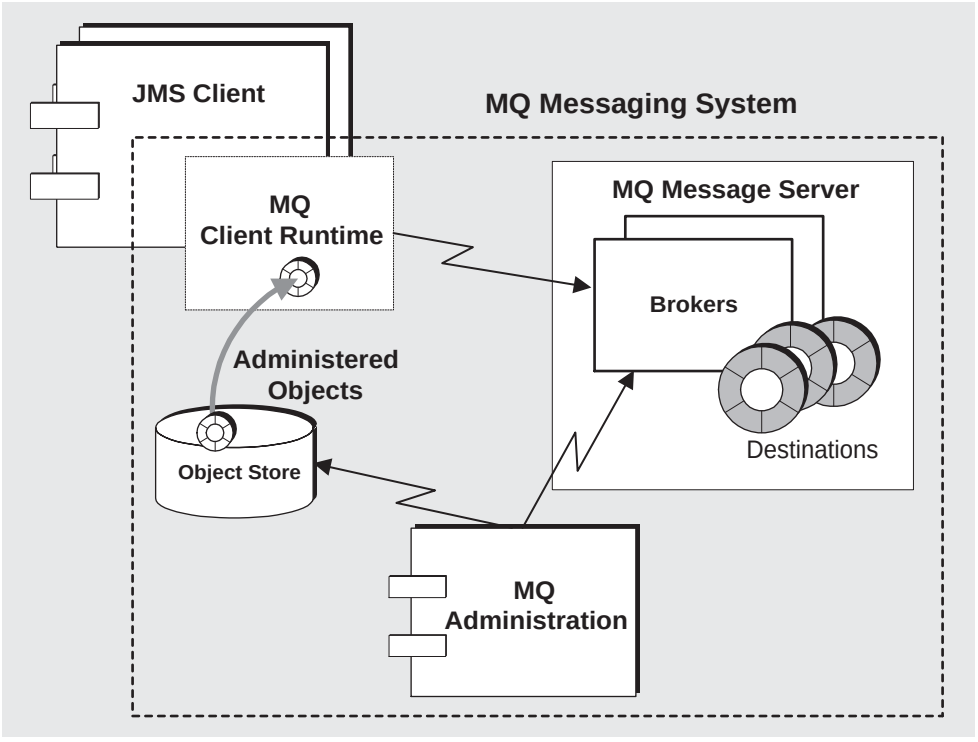
MQ 메시지 서버

그림 306페이지의 표 "MQ 시스템 구조"에 표시된 것처럼, MQ 메시지 서버의 주요 부분은 브로커 및 물리적 대상입니다.

브로커. 브로커는 MQ 메시지 시스템에 전달 서비스를 제공합니다. 메시지 전달은 연결 서비스, 메시지 라우팅 및 전달, 지속성, 보안 및 로깅을 처리하는 수많은 지원 구성 요소를 사용하여 수행됩니다. 메시지 서버는 하나 이상의 브로커를 사용하여 확장할 수 있습니다.

물리적 대상. 메시지 전달은 브로커가 관리하는 클라이언트에서 물리적 대상으로의 전달과, 대상에서 하나 이상의 사용 클라이언트로의 전달 등 두 단계 프로세스로 구성됩니다. 물리적 대상은 브로커의 물리적 메모리 및/또는 지속성 저장소(자세한 정보는 307페이지의 표 "물리적 대상" 참조) 내의 위치를 나타냅니다.

MQ 시스템 구조



브로커

생성 클라이언트에서 대상으로, 이후 대상에서 하나 이상의 사용 클라이언트로의 MQ 메시징 시스템의 메시지 전달은 브로커가 수행합니다(또는 MQ 3.01 Enterprise Edition의 경우 브로커 클러스터가 함께 작업을 수행합니다). 브로커가 메시지를 전달하려면, 클라이언트와의 통신 채널 설정, 인증 및 권한부여 수행, 적절하게 메시지 라우트, 신뢰할 수 있는 전달 보장 및 모니터링 시스템 성능을 위한 데이터 제공 등을 수행해야 합니다.

브로커는 복잡한 기능을 수행하기 위해 많은 구성 요소를 사용하며, 각각은 전달 프로세스에서 특정 역할을 수행합니다. 이 내부 구성 요소들을 로드 조건, 응용 프로그램 복잡성 등에 따라 적절하게 구성하면 브로커 성능을 최적화할 수 있습니다. 자세한 정보는 MQ *Administrator's Guide*를 참조하십시오.

물리적 대상

MQ 메시징은 두 단계 메시지 전달로 기술됩니다. 먼저 생성자 클라이언트에서 브로커에 있는 대상으로의 전달이 수행되고, 다음으로 브로커의 대상에서 하나 이상의 사용자 클라이언트로의 전달이 수행됩니다. 전달에는 대기열(점대점 전달 모델) 및 주제(게시/가입 전달 모델)의 두 가지 유형이 있습니다. 이런 대상들은 수신 메시지가 사용자 클라이언트에게 라우트되기 전 마샬링되는 브로커의 물리적 메모리 내 위치를 표시합니다.

일반적으로 관리 도구를 사용하여 물리적 대상을 작성하지만, 브로커가 메시지를 수신했을 때 자동으로 작성되기도 합니다.

대기열 대상. 대기열 대상은 점대점 메시징에 사용되며, 여기에서 메시지는 대상에 인터레스트로 등록된 여러 사용자 중 최종적으로 하나에만 전달된 것을 의미합니다. 메시지는 생성자 클라이언트에서 출발한 다음 사용자 클라이언트의 대기열에 쌓이고 전달됩니다.

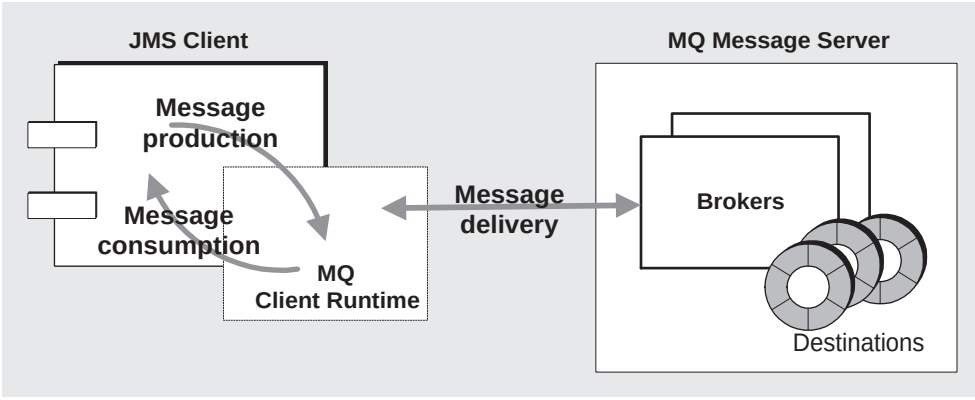
주제 대상. 주제 대상은 게시/가입 메시징에 사용되며, 여기에서 메시지는 대상에 인터레스트로 등록된 모든 사용자에 최종적으로 전달된 것을 의미합니다. 메시지는 생성자에서 출발한 다음 해당 주제에 가입한 모든 사용자에게 라우트됩니다. 사용자가 해당 주제에 대해 영구 가입되어 있으면 메시지가 해당 주제에 전달되었을 때 활성화하지 않고, 브로커가 메시지를 일단 저장한 다음 사용자가 다시 활성화되었을 때 전달합니다.

MQ 클라이언트 런타임

MQ 클라이언트 런타임은 MQ 메시지 서버로의 인터페이스를 가진 JMS 클라이언트(단독 응용프로그램, 웹 구성 요소 또는 EJB 구성 요소)를 제공합니다. 이것은 JMS 클라이언트에게 클라이언트가 대상에 메시지를 보내고 이런 대상에서 메시지를 수신할 때 필요한 모든 프로그래밍 인터페이스 구현을 제공합니다.

그림 308페이지의 표 “메시징 작업”은 JMS 클라이언트 및 MQ 클라이언트 런타임 사이의 통신과 관련된 메시지 생성 및 사용 방법을 표시하며, 여기에서 메시지 전달은 MQ 클라이언트 런타임 및 MQ 메시지 서버 사이의 통신과 관련됩니다.

메시징 작업



MQ 관리 객체

MQ 관리 객체를 사용하면 JMS 클라이언트 코드가 공급자에 영향을 받지 않을 수 있습니다 (302페이지의 표 "관리 객체: 공급자 독립성" 참조). MQ 관리 객체는 공급자별 구현 및 구성 정보를 객체 안에 저장하여 이후 공급자에 독립적인 방법으로 클라이언트 응용 프로그램이 사용할 수 있도록 하는 캡슐화 방법을 사용합니다. MQ 관리 객체는 관리자가 작성 및 구성하며, 표준 JNDI 조회 코드를 통해 JMS 클라이언트가 액세스합니다.

연결 팩토리 관리 객체. 연결 팩토리 객체는 JMS 클라이언트(단독 응용 프로그램, 웹 구성 요소 또는 EJB 구성 요소)와 MQ 메시지 서버 사이의 물리적 연결 작성에 사용됩니다. 연결 팩토리 객체는 브로커 내에 물리적으로 표현되지 않으며, 그저 JMS 클라이언트를 활성화하여 브로커에 연결을 구축합니다. 또한 연결 팩토리 객체를 사용하여 연결이 브로커 액세스에 사용하는 연결 및 클라이언트 런타임의 동작을 지정합니다. 이 때문에 MQ 연결 팩토리에는 MQ 시스템의 성능을 조정할 수 있는 구성 가능한 속성이 아주 많습니다.

대상 관리 객체. 대상 관리 객체(대기열 또는 주제)는 브로커 내의 이름이 공개된 대상 관리 객체에 해당하는 물리적 대상(물리적 대기열 또는 물리적 주제)을 표현합니다. 대상 관리 객체를 작성하면 JMS 클라이언트(메시지 사용자 및/또는 메시지 생성자)가 해당 물리적 대상을 액세스할 수 있습니다.

MQ 관리 도구

MQ 관리 도구는 명령줄 유틸리티 및 GUI (Graphical User Interface) 관리 콘솔의 두 가지 기준으로 나뉘어집니다.

관리 콘솔. 관리 콘솔을 사용하여 브로커 연결 및 관리, 브로커에 물리적 대상 작성, 객체 저장소에 연결 및 객체 저장소에서 관리 객체 추가, 업데이트 및 삭제 작업을 수행할 수 있습니다. 관리 콘솔을 사용하여 수행할 수 없는 작업에는 주로 브로커 시작, 브로커 클러스터 작성, 브로커의 특수한 추가 등록 정보 구성 및 사용자 데이터베이스 관리 등이 있습니다.

명령줄 유틸리티. MQ 유틸리티를 사용하면 관리 콘솔을 사용하여 수행할 수 있는 모든 작업과 함께, 브로커 시작 및 관리, 브로커의 특수한 추가 등록 정보 구성 및 MQ 사용자 데이터베이스 관리 작업도 수행할 수 있습니다.

MQ와 통합 Sun ONE Application Server

MQ Platform Edition은 Sun ONE Application Server 설치 프로세스 중 일부로 자동 설치됩니다. 자세한 정보는 *Sun ONE Application Server 설치 설명서*를 참조하십시오.

이것을 설치하면 많은 수의 Sun ONE Application Server 인스턴스를 지원하는 Sun ONE Application Server JMS 메시징 시스템이 제공됩니다. 기본적으로 각 서버 인스턴스에는 인스턴스에서 실행 중인 모든 JMS 클라이언트를 지원하는 관련 기본 제공 JMS 서비스가 들어 있습니다.

이 항목은 다음 내용으로 구성되어 있습니다.

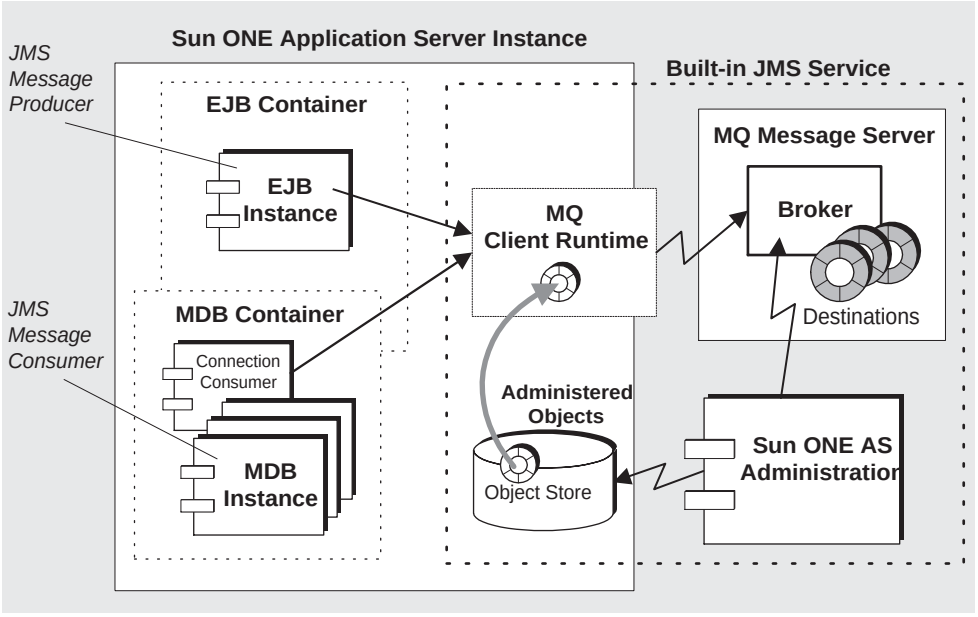
- 기본 제공 JMS 서비스의 구조
- 기본 제공 JMS 서비스 비활성화

기본 제공 JMS 서비스는 Sun ONE Application Server 관리 도구(312페이지의 표 “기본 제공 JMS 서비스 관리” 참조)를 사용하여 관리할 수 있습니다.

기본 제공 JMS 서비스의 구조

그림 310페이지의 표 “기본 제공 MQ 메시징 시스템”에 표시된 기본 제공 JMS 서비스는 보통 MQ 메시징 시스템(그림 306페이지의 표 “MQ 시스템 구조” 참조)과 유사하지만, 다음에서 설명하는 많은 자격 증명을 가진 점이 다릅니다.

기본 제공 MQ 메시징 시스템



MQ 메시지 서버. 각 Sun ONE Application Server 인스턴스는 고유 JMS 서비스와 관련되어 있습니다. 기본 제공 JMS 서비스는 브로커 메시지 서버 하나를 사용합니다. 브로커는 그림 "기본 제공 MQ 메시징 시스템"에 표시된 Sun ONE Application Server 인스턴스 외부에서 별도 프로세스로 실행됩니다. 기본적으로 브로커 인스턴스(기본 제공 JMS 서비스)는 서버 인스턴스가 종료되어 관련된 서버 인스턴스가 시작하고 종료될 때 시작됩니다. 서버 인스턴스의 기본 제공 JMS 서비스에 대한 구성 정보는 Sun ONE Application Server 구성 저장소(`server.xml` 파일 참조)에 기록되며, 313페이지의 표 "JMS 서비스 구성" 내에 표시된 대로 수정될 수 있습니다.

MQ 클라이언트 런타임. JMS 서비스의 클라이언트 런타임 부분은 JMS API가 지원하는 라이브러리 모음입니다. 서버 인스턴스 내에서 실행 중인 JMS 클라이언트(JMS 클라이언트 구성 요소, MDB 포함)는 어느 것이든 이 라이브러리에 액세스할 수 있습니다.

MQ 관리 객체. 기본 제공 JMS 서비스는 Sun ONE Application Server이 제공하는 객체 저장소를 사용합니다. 각 서버 인스턴스는 자신의 객체 저장소를 가지고 있습니다. JMS 서비스는 객체 저장소 내에 관리 객체(연결 팩토리 및 대상 자원)를 저장합니다. 319페이지의 표 "관리 객체 자원 관리"에서 설명한 것과 같은 이 관리 객체 자원을 작성하고, JNDI 조회 코드를 사용하여 JMS 클라이언트로 액세스합니다.

Sun ONE Application Server 관리. Sun ONE Application Server 관리 인터페이스 및 명령줄 유틸리티는 MQ 관리 기능의 제한된 일부만을 구현합니다. 관리 인터페이스 및 명령줄을 사용하면 기본 제공 JMS 서비스 구성, 물리적 대상 작성 및 삭제, JMS 클라이언트가 JMS 메시징 작업을 수행할 때 필요한 관리 객체 자원 작성 및 삭제를 수행할 수 있습니다. 하지만 이 관리 도구로는 브로커 등록 정보 설정, MQ 클라이언트 런타임 조정, MQ 사용자 리포지토리 수정, MQ 보안 관리와 같은 좀 더 복잡한 관리 작업은 수행할 수 없습니다(또는 이용할 수 없습니다). 기본 제공 JMS 서비스의 이런 관리 작업을 수행하려면 MQ와 함께 설치되어 있고 MQ *Administrator's Guide*에 설명되어 있는 관리도구를 사용해야 합니다. MQ 및 Sun ONE Application Server 관리 기능 비교는313페이지의 표 "Sun ONE Message Queue와 Sun ONE Application Server 관리 기능의 비교"를 참조하십시오.

기본 제공 JMS 서비스 비활성화

기본적으로 기본 제공 JMS 서비스는 관련된 Sun ONE Application Server인스턴스가 시작할 때 시작됩니다(즉, MQ 브로커가 시작됩니다). 그러나 서버 인스턴스가 JMS 메시징을 지원할 필요가 없거나 서버 인스턴스가 *외부 JMS 서비스*를 사용하기 때문에, 서버 인스턴스를 시작할 때 JMS 서비스가 자동으로 시작하지 않도록 하고 싶을 수 있습니다. 기본 제공 JMS 서비스를 사용하지 않으려면 313페이지의 표 "JMS 서비스 구성"을 참조하십시오.

외부 JMS 서비스는 Sun ONE Application Server 내에서 제어되지 않는 메시징 시스템입니다. MQ (*원시* JMS 공급자)의 경우 이것은 단순히 MQ 관리 도구로 MQ 메시지 서버를 독립적으로 시작 및 관리한다는 의미입니다. 여러 서버 인스턴스에서 실행 중인 JMS 클라이언트도 여전히 MQ 관리 객체를 사용하여 MQ 메시지 서버에 접근할 수 있습니다. 관리 객체는 각 응용 프로그램 서버 인스턴스와 관련된 객체 저장소에 저장되거나 각 객체를 별도로 MQ 관리 도구로 관리하는 독립적인 객체 저장소에 저장할 수 있습니다(필요하면 여러 서버 인스턴스가 저장소를 공유할 수 있음).

서버 인스턴스의 외부 JMS 서비스 사용 시나리오에는 여러 가지가 있습니다. 가장 일반적인 것은 상이한 서버 인스턴스의 JMS 클라이언트가 동일한 물리적 대상에 접근해야 하는 경우입니다. 이 경우 서버 인스턴스 모두는 동일한 메시지 서버에 접근해야 합니다. 이를 위해서는 모든 서버 인스턴스의 기본 제공 JMS 서비스를 비활성화하고, 모든 JMS 클라이언트가 외부 JMS 서비스에 해당 JNDI 조회를 수행하도록 구성합니다. 또한 외부 JMS 서비스 공급자의 관리 도구를 사용하여 외부 JMS 서비스(메시지 서버 관리, 물리적 대상 작성 및 필요한 모든 관리 객체 작성)를 독립적으로 관리합니다.

여러 응용 프로그램 서버 인스턴스가 하나의 MQ 브로커 인스턴스를 공유하도록 구성하려면 다음 작업을 수행합니다.

1. 모든 서버 인스턴스의 JMS 서비스를 비활성화합니다.
2. 서버 인스턴스와 관계없이 공유 MQ 브로커를 관리합니다. 즉 외부 서비스를 관리하는 관리 도구를 사용하여 브로커를 시작 및 종료해야 합니다. 또한 Sun ONE 응용 프로그램 서버와 관계없이 물리적 대상을 관리해야 합니다.
3. 예를 들어, 각 서버 인스턴스의 연결 팩토리 JMS 자원이 외부 MQ 브로커를 지정하도록 구성합니다. `imqBrokerHostName` 및 `imqBrokerHostPort` 등록 정보를 적절하게 설정해야 합니다.
4. Sun ONE 응용 프로그램 서버에 JMS 응용 프로그램을 배포할 때 연결 팩토리 자원을 사용합니다.

외부 JMS 서비스와 기본 제공 JMS 서비스를 동시에 실행시킬 수 있습니다. 서버 인스턴스의 JMS 클라이언트는 필요하면 어느 JMS 서비스에든 접근할 수 있습니다.

많은 서버 인스턴스가 동일한 기본 제공 JMS 서비스를 공유하게 만드는 것은 좋지 *않습니다* (하나만 활성화되고 나머지는 비활성화됨). 관련된 서버 인스턴스가 실행될 때에만 활성화된 JMS 서비스가 실행되기 때문에 이렇게 하면 상황을 관리하기 어렵게 되어 좋지 않습니다.

기본 제공 JMS 서비스를 비활성화하면 기본 제공 JMS 서비스와 관련된 관리 작업도 수행할 수 없습니다. 외부 JMS 서비스 지원에 필요한 모든 관리 작업은 외부 서비스 관리용 관리 도구를 사용해서 수행되어야 합니다.

기본 제공 JMS 서비스 관리

이 주제는 기본 제공 JMS 서비스의 관리에 중점을 둡니다. *관리는 서버 인스턴스별로 수행됩니다*.

기본 제공 JMS 서비스 관리에는 다음 작업도 포함됩니다.

- JMS 서비스 구성
- 물리적 대상 관리
- 관리 객체 자원 관리
- 명령줄 인터페이스를 사용한 기본 제공 JMS 서비스 관리

관리는 Sun ONE Application Server의 관리 인터페이스 또는 명령줄 유틸리티를 사용하여 수행될 수 있습니다. 표 "Sun ONE Message Queue와 Sun ONE Application Server 관리 기능의 비교"에는 이 관리 도구와 MQ 관리 도구와의 비교가 표시되어 있습니다.

Sun ONE Message Queue와 Sun ONE Application Server 관리 기능의 비교

기능	Sun ONE MQ 관리 도구	Sun ONE AS 관리 인터페이스	Sun ONE AS 관리 명령줄
MQ 브로커 상태 관리	예	시작/중지	시작/중지
MQ 브로커 구성	예	아니요	아니요
MQ 브로커 사용자 리포지토리 관리	예	아니요	아니요
다중 브로커 클러스터	예	아니요	아니요
보안 관리	예	아니요	아니요
물리적 대상 관리	예	작성/삭제	작성/삭제
영구 가입 및 트랜잭션 관리	예	아니요	아니요
관리 객체 자원 관리	예	작성/삭제/구성	예

다음 절에서는 Sun ONE Application Server 관리 인터페이스를 사용하는 JMS 서비스 관리 작업 수행 방법에 대해 설명합니다.

JMS 서비스 구성

많은 JMS 서비스 등록 정보는 설치시 기본 제공 JMS 서비스가 함께 설치됩니다. 이 등록 정보의 기본 값은 JMS 서비스를 구성하여 변경할 수 있습니다.

JMS 서비스 등록 정보는 표 "JMS 서비스 등록 정보"에 설명되어 있습니다.

JMS 서비스 등록 정보

등록 정보	설명	기본 값
로그 수준	Sun ONE 응용 프로그램 서버 로그 파일에 기록할 로깅 정보 수준. 자세한 정보는 5장, "로깅 사용"을 참조하십시오.	`DEBUG_HIGH`
포트	기본 제공 JMS 서비스가 제공하는 브로커 인스턴스의 기본 포트 번호. 기본적으로 기본 제공 JMS 서비스는 기본으로 제공된 기본 포트 번호를 사용합니다. 하지만 포트가 다른 소프트웨어와 충돌하거나 두 개 이상의 Sun ONE Application Server 인스턴스를 시작하는 경우에는 각 인스턴스에 대해 고유 기본 포트 번호를 지정해야 합니다. 설치시 JMS 서비스에 지정된 포트 번호를 이후 다른 서비스에서 사용하면 포트가 충돌할 수 있음에 주의하십시오. 이 경우 JMS 서비스에 다른 포트 번호를 부여해야 합니다.	`7676`
관리자의 사용자 이름/암호	물리적 대상 관리(316페이지의 표 "물리적 대상 관리" 참조)와 같은 브로커 관리 작업 수행에 필요한 사용자 이름/암호. 관리자가 브로커 인스턴스에 보안이 유지된 상태로 액세스하려면(기본값으로 어떤 사용자도 액세스할 수 있습니다), 먼저 브로커의 사용자 리포지토리에 적절한 항목을 작성한 다음(MQ Administrator's Guide에서 설명) 관리자의 사용자 이름 및 암호에 해당 값을 입력해야 합니다.	`admin/admin`
시작 시간 초과	서버 인스턴스가 JMS 서비스 시작을 기다리는 시간(초 단위)을 지정합니다. 시간이 초과되면 서버 인스턴스 시작이 중단됩니다.	`60`
시작 인수	JMS 서비스 시작에 사용될 인수를 지정합니다. `imqbroker` 명령의 시작 인수 및 지정 방법은 MQ *Administrator's Guide*에서 찾을 수 있습니다 (`-name` 및 `-port` 인수는 제공해도 무시).	

JMS 서비스 등록 정보*(계속)*

등록 정보	설명	기본 값
시작 활성화	서버 인스턴스 시작시 기본 제공 JMS 서비스가 시작되는지 여부를 지정합니다. JMS 메시징을 지원하지 않거나 외부 JMS 메시지 서비스를 사용하는 경우에는 이 등록 정보를 FALSE로 설정하십시오.	TRUE

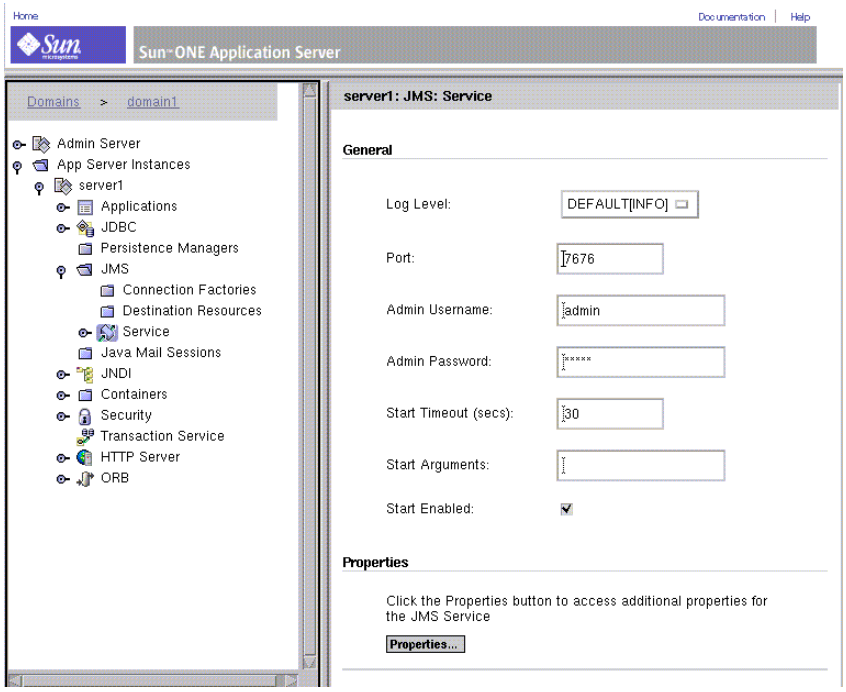
기본 제공 JMS 서비스는 해당 서버 인스턴스 시작 전에 미리 구성될 수 있습니다. 하지만 서버 인스턴스가 이미 실행 중인 경우에는 구성 변경 사항이 서버 인스턴스가 중지되고 이후 다시 시작되어야 적용됩니다.

기본 제공 JMS 서비스를 구성하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 엽니다.
2. 왼쪽 분할 창에서 서버 인스턴스를 엽니다.
3. JMS 폴더를 엽니다.
4. "서비스" 링크를 선택합니다.

   "JMS 서비스 구성" 화면이 오른쪽 분할 창에 표시됩니다.

JMS 서비스 구성 화면



5. 필요한 등록 정보를 수정합니다(314페이지의 표 “JMS 서비스 등록 정보” 참조).
6. "저장" 버튼을 누릅니다.

   "JMS 서비스" 화면이 새로 고쳐집니다.

물리적 대상 관리

JMS 메시징의 경우, JMS 공급자는 JMS 사용자로 디스패치된 메시지 서비스의 물리적 대상에 메시지를 보냅니다.

기본 제공 JMS 서비스의 경우, 이 물리적 대상을 명시적으로 작성할 수도 있고 메시지를 수신했을 때 JMS 서비스(MQ 브로커)가 자동으로 작성할 수도 있습니다. 일반적으로 메시징 응용 프로그램에 필요한 물리적 대상을 명시적으로 작성하면 메시징 시스템 및 자원에 대해 더 많은 제어권을 가질 수 있습니다. 이 대상이 더 이상 필요하지 않으면 삭제할 수 있습니다.

기본 제공 JMS 서비스의 물리적 대상을 작성하거나 삭제하려면, JMS 서비스가 실행 중이고 관리자의 사용자 이름 및 암호(기본 제공 JMS 서비스 구성시 지정)가 브로커의 사용자 리포지토리 내의 유효한 항목에 대응되어야 합니다(314페이지의 표 “JMS 서비스 등록 정보” 참조).

관리 인터페이스를 사용하는 경우에는 기본 제공 JMS 서비스의 물리적 대상에 다음 관리 작업을 수행할 수 있습니다.

- 대기열 또는 주제 대상 작성
- 물리적 대상 목록
- 물리적 대상 삭제

대기열 또는 주제 대상 작성

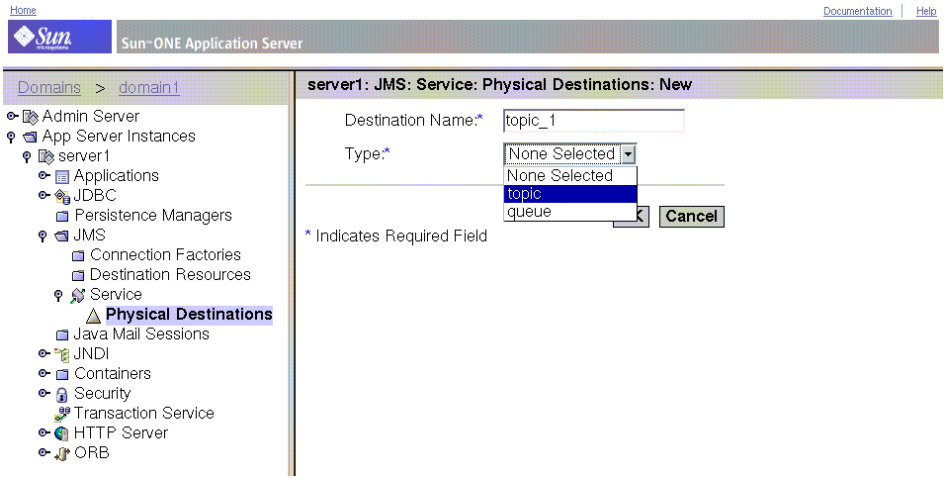
대기열 또는 주제 대상을 작성하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 엽니다.
2. 왼쪽 분할 창에서 서버 인스턴스를 엽니다.
3. JMS 폴더를 엽니다.
4. "서비스 링크"를 엽니다.
5. "물리적 대상" 링크를 선택합니다.

   "물리적 대상" 화면이 오른쪽 분할 창에 표시됩니다.
6. "새로 만들기" 버튼을 누릅니다.

   물리적 대상은 다음과 같습니다. "새로 만들기" 화면은 오른쪽 분할 창에 표시됩니다.

새 JMS 물리적 대상 화면



7. 물리적 대상 이름을 입력합니다.
8. "유형" 풀 다운에서 대기열 또는 주제를 선택합니다.
9. "확인" 버튼을 누릅니다.

   오른쪽 분할 창이 새로 고쳐지고 기존 대기열 및 주제 대상 목록에 새 대기열 또는 주제 대상이 표시됩니다.

물리적 대상 목록

기존 대기열 및 주제 대상을 나열하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 엽니다.
2. 왼쪽 분할 창에서 서버 인스턴스를 엽니다.
3. JMS 폴더를 엽니다.
4. "서비스 링크"를 엽니다.
5. "물리적 대상" 링크를 선택합니다.

   오른쪽 분할 창에 현재 물리적 대상이 표시됩니다.

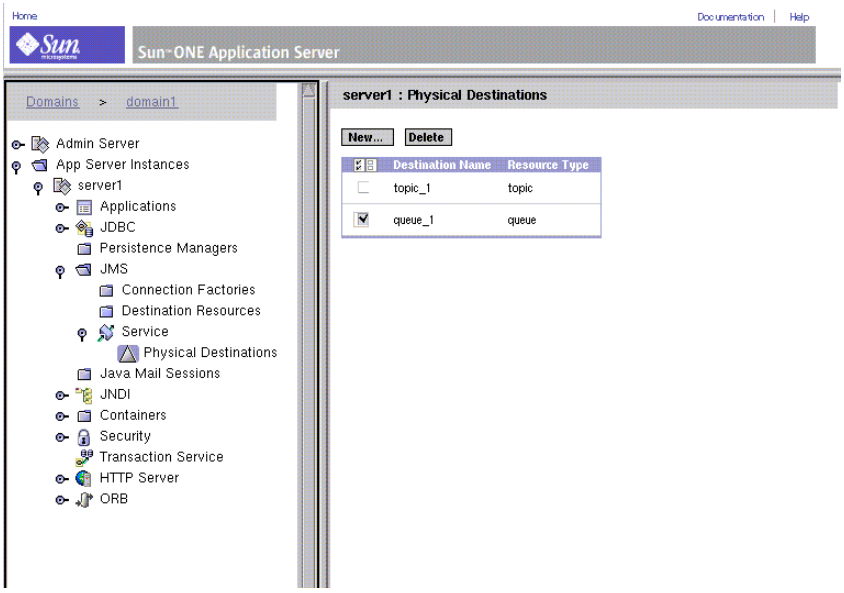
물리적 대상 삭제

필요하면 대기열 또는 주제 대상을 삭제할 수 있습니다.

물리적 대상을 삭제하려면 다음 작업을 수행합니다.

1. 기존 대상을 나열합니다(318페이지의 표 “물리적 대상 목록” 참조).
2. 삭제할 각 대상의 "선택" 상자를 누릅니다.

JMS 물리적 대상 화면



3. "삭제" 버튼을 눌러 선택한 대상을 제거합니다.

   목록이 새로 고쳐져 남은 대상을 표시합니다.

관리 객체 자원 관리

Sun ONE 응용 프로그램 서버에서 *JMS 자원*이 되는 MQ 관리 객체는 JMS 클라이언트가 JMS 서비스(기본 제공 또는 외부) 액세스에 사용합니다.

J2EE 구성 요소는 연결 팩토리 및 대상의 두 관리 객체 자원을 사용하여 JMS 서비스에 대한 연결을 구한 다음, 서비스(308페이지의 표 "MQ 관리 객체" 참조)의 물리적 대상과 메시지를 주고 받습니다.

관리 객체 자원 작성은 JMS 서비스와 직접적인 관련이 없기 때문에 JMS 서비스를 활성화할 필요가 없으며, 서비스 인스턴스의 관리 객체 자원 작성에 유효한 사용자 이름 및 암호(314페이지의 표 "JMS 서비스 등록 정보" 참조)가 필요하지 않습니다.

관리 객체 속성

JMS 메시징을 지원하려면 서버 인스턴스에 실행 중인 모든 JMS 클라이언트에 대해 필요한 관리 객체 자원을 작성해야 합니다. 최소한 각 관리 객체 자원의 JNDI 조회 이름, 유형(연결 팩토리, 대기열 또는 주제), 설명(선택적) 및 자원 활성화 여부는 반드시 지정해야 합니다. 기타 속성은 다음 절에서 설명합니다.

대상(대기열 또는 주제)

대기열 또는 주제 관리 객체의 경우, 해당하는 물리적 대상 이름도 지정해야 합니다.

연결 팩토리

연결 팩토리 관리 객체의 경우, 관리 인터페이스는 기본적으로 기본 제공 JMS 서비스(즉 호스트 이름이 로컬 호스트이고 포트 번호를 JMS 서비스 구성 때 설정한 브로커 인스턴스. 314페이지의 표 "JMS 서비스 등록 정보" 참조)를 지칭하는 연결 객체를 작성합니다.

하지만 특정 서버 인스턴스의 JMS 서비스를 비활성화하면 이 서버 인스턴스가 지원하는 JMS 클라이언트는 외부 JMS 서비스를 사용해야 합니다. 외부 JMS 서비스에 대한 연결 작성에 사용할 연결 팩토리를 작성하는 경우, 사용자는 적절한 브로커 인스턴스의 호스트 이름 및 포트 번호를 지정하는 속성을 설정해야 합니다.

연결 팩토리 관리 객체에는 서버 인스턴스의 MQ 클라이언트 런타임을 조정할 때 사용하는 추가 속성이 있습니다. 이것은 MQ *Developer's Guide*에 문서화되어 있습니다.

관리 인터페이스를 사용하여 작성된 연결 팩토리 관리 객체는 배포된 트랜잭션 관리자를 지원합니다.

관리 객체 자원 관리 작업

관리 인터페이스에서는 관리 객체 자원에 대한 다음 관리 작업을 수행할 수 있습니다.

- 대기열 또는 주제 관리 객체 작성(대상 자원)
- ConnectionFactory 관리 객체 작성
- 관리 객체 자원 목록
- 관리 객체 자원 삭제

대기열 또는 주제 관리 객체 작성(대상 자원)

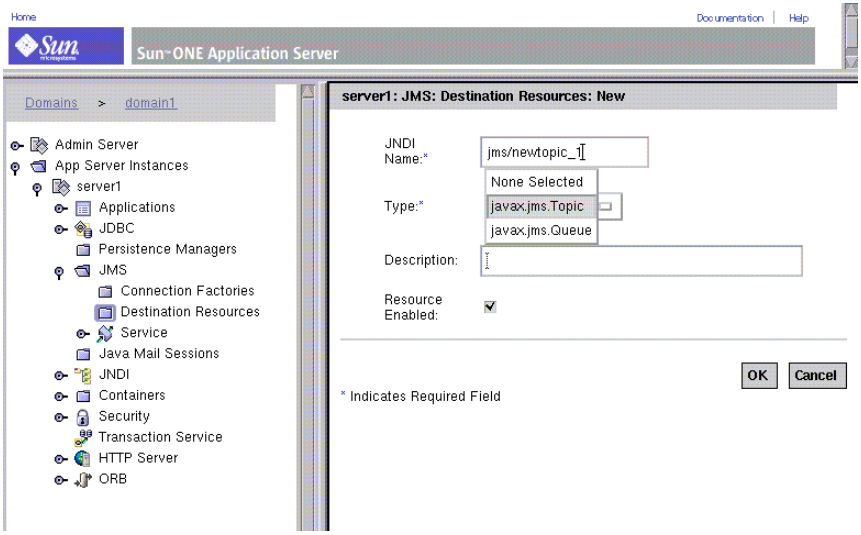
대기열 또는 주제 관리 객체를 작성하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 엽니다.
2. 왼쪽 분할 창에서 서버 인스턴스를 엽니다.
3. JMS 폴더를 엽니다.
4. "대상 자원" 링크를 선택합니다.

   "대상 자원" 화면이 오른쪽 분할 창에 표시됩니다.
5. "새로 만들기" 버튼을 누릅니다.

   "대상 자원: 새로 만들기" 화면이 나타납니다.

새 대상 관리 객체 화면



6. 이 대상 관리 객체와 관련된 JNDI 조회 이름을 입력합니다.
7. 풀 다운 목록에서 "대기열" 또는 "주제"를 선택합니다.
8. "확인" 버튼을 누릅니다.

   오른쪽 분할 창에 대상 자원이 다시 표시됩니다. 새 창이 나타납니다.

또한 `imqDestinationName` 등록정보를 지정하여 객체의 대상 이름을 지정해야 합니다. 이 등록 정보의 값은 물리적 대상 이름과 일치해야 합니다.

이 등록 정보의 값을 변경하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 엽니다.
2. 왼쪽 분할 창에서 서버 인스턴스를 엽니다.
3. JMS 폴더를 엽니다.
4. 대상 자원 폴더를 엽니다.
5. 편집할 대상 자원을 선택합니다.

   오른쪽 분할 창에 대상 자원 창이 표시됩니다.
6. 오른쪽 분할 창에서 "등록 정보"를 누릅니다.

   "등록정보 편집" 창이 나타납니다.
7. "이름" 필드에 imqDestinationName을 입력합니다.
8. "값" 필드에 물리적 대상 이름을 입력합니다.
9. "확인"을 누릅니다.

   오른쪽 분할 창에 대상 자원 창이 다시 표시됩니다.

ConnectionFactory 관리 객체 작성

대기열 연결 팩토리 또는 주제 연결 팩토리 관리 객체를 작성하려면 다음 작업을 수행합니다.

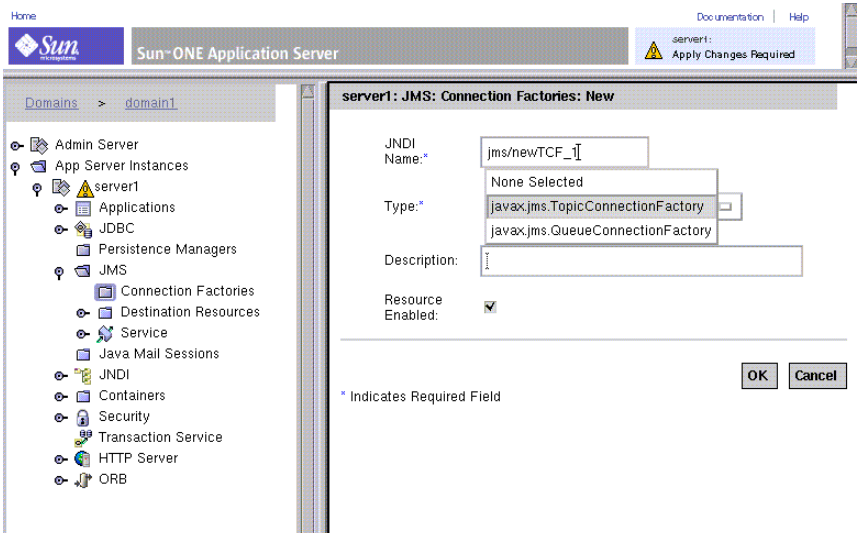
1. 관리 인터페이스를 엽니다.
2. 왼쪽 분할 창에서 서버 인스턴스를 엽니다.
3. JMS 폴더를 엽니다.
4. "연결 팩토리" 링크를 선택합니다.

   "연결 팩토리 자원" 화면이 오른쪽 분할 창에 표시됩니다.

5. "새로 만들기" 버튼을 누릅니다.

   "연결 팩토리 자원: 새로 만들기" 화면이 나타납니다.

새 ConnectionFactory 관리 객체 화면



6. 이 연결 팩토리 관리 객체와 관련된 JNDI 조회 이름을 입력합니다.
7. 풀 다운 목록에서 연결 팩토리 객체 유형을 선택합니다.
8. "확인" 버튼을 누릅니다.

   오른쪽 분할 창에 목록에 새로 작성된 연결 팩토리 객체가 있는 연결 팩토리 자원 창이 다시 표시됩니다.

연결 팩토리가 기본 제공 JMS 서비스의 브로커가 아닌 다른 브로커에 연결을 작성한 경우에는 `imqBrokerHostName` 및 `imqBrokerHostPort` 등록 정보의 값을 설정해야 합니다.

이 등록 정보의 값을 변경하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 엽니다.
2. 왼쪽 분할 창에서 서버 인스턴스를 엽니다.
3. JMS 폴더를 엽니다.
4. 연결 팩토리 폴더를 엽니다.

5. 편집할 연결 팩토리 자원을 선택합니다.

   오른쪽 분할 창에 연결 팩토리 창이 표시됩니다.

6. 오른쪽 분할 창에서 "등록 정보"를 누릅니다.

   "등록 정보 편집" 창이 나타납니다.

7. "이름" 필드에 `imqBrokerHostName`을 입력합니다.
8. "값" 필드에 해당 등록 정보의 값을 입력합니다.
9. "이름" 필드에 `imqBrokerHostPort`을 입력합니다.
10. "값" 필드에 해당 등록 정보의 값을 입력합니다.
11. "확인"을 누릅니다.

    오른쪽 분할 창에 연결 팩토리 창이 다시 표시됩니다.

관리 객체 자원 목록

기존 관리 객체를 나열하려면 다음 작업을 수행합니다.

1. 관리 인터페이스를 엽니다.
2. 왼쪽 분할 창에서 서버 인스턴스를 엽니다.
3. JMS 폴더를 엽니다.
4. "대상 자원" 또는 "연결 팩토리 자원" 링크를 선택합니다.

   오른쪽 분할 창에 현재 대상 또는 연결 팩토리 관리 객체가 표시됩니다.

관리 객체 자원 삭제

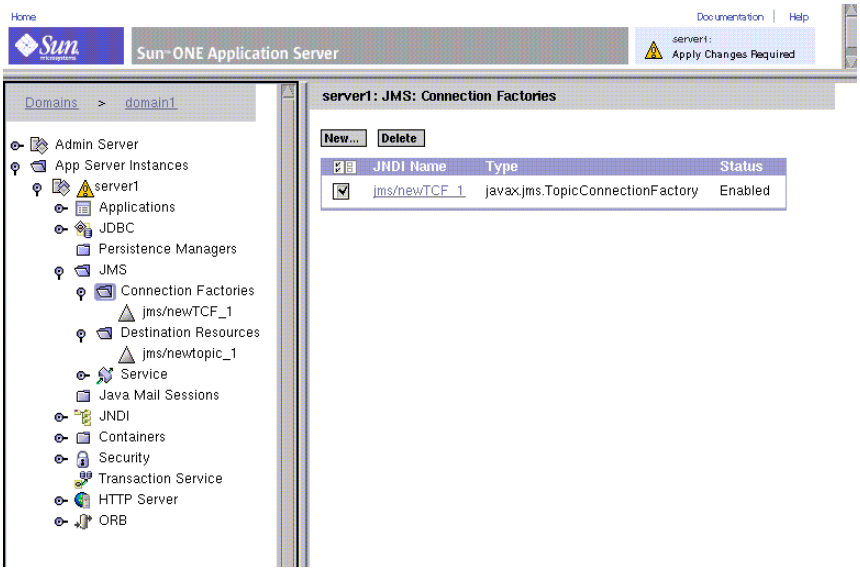
관리 객체 자원을 삭제하려면 다음 작업을 수행합니다.

1. 기존 관리 객체 자원을 나열합니다(324페이지의 표 “관리 객체 자원 목록” 참조).

   오른쪽 분할 창에 기존 관리 객체 자원이 표시됩니다.

2. 삭제할 각 객체의 "선택" 상자를 누릅니다.

JMS 연결 팩토리 화면, 새로 고쳐짐.



3. "삭제" 버튼을 눌러 각 선택 객체를 제거합니다.

   화면의 목록이 새로 고쳐져 남은 관리 객체 자원을 표시합니다.

명령줄 인터페이스를 사용한 기본 제공 JMS 서비스 관리

Sun ONE 응용 프로그램 서버에는 명령줄 유틸리티 `asadmin`이 있어서, 관리 인터페이스로 수행할 수 있는 작업과 동일한 모든 작업 수행에 사용할 수 있습니다.

asadmin 명령을 사용하여 기본 제공 JMS 서비스를 구성 및 관리합니다.

기본 제공 JMS 서비스 관리에 사용하는 asadmin 명령

명령	사용
add-resources	하나 이상의 jdbc, jms 또는 javamail 유형 자원을 추가합니다.
create-jmsdest	JMS 물리적 대상을 작성합니다.

기본 제공 JMS 서비스 관리에 사용하는 asadmin 명령

명령	사용
create-jms-resource	JMS 자원을 작성합니다.
delete-jmsdest	JMS 물리적 대상을 삭제합니다.
delete-jms-resource	JMS 자원을 삭제합니다.
jms-ping	JMS 공급자에게 ping하여 실행 중인지 확인합니다.
list-jmsdest	서버 인스턴스의 JMS 물리적 대상을 나열합니다.
list-jms-resources	서버 인스턴스의 JMS 자원을 나열합니다.
get and set jms-service	JMS 서비스의 속성을 가져와서 설정합니다.
get and set jms-resource	JMS 자원의 속성 값을 구하거나 설정합니다.

이 명령의 구문에 대한 정보는 asadmin의 온라인 도움말을 참조하십시오. jms-service 및 jms-resource의 속성 목록과 함께 asadmin에 대한 추가 정보는 부록 A, “명령줄 인터페이스 사용을 참조하십시오.

Corba/IIOP 클라이언트를 지원하도록 서버 구성

이 장에서는 Sun ONE Application Server 환경 내에서 RMI/IIOP 프로토콜을 사용하여 CORBA/IIOP 클라이언트에 대한 지원을 구성하는 방법에 대해 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- CORBA/IIOP 클라이언트 지원 정보
- ORB 구성

CORBA/IIOP 클라이언트 지원 정보

J2EE 플랫폼은 상호 운용성 요구 사항을 통해 다양한 클라이언트 유형, 여러 하드웨어 플랫폼 및 다수의 소프트웨어 응용 프로그램에 대한 간접적인 지원을 제공합니다. J2EE 호환 제품인 Sun ONE Application Server는 상호 운용성을 위한 표준 프로토콜 및 형식 집합을 지원합니다.

CORBA (Common Object Request Broker Architecture) 모델은 원격 메소드 요청의 형태로 객체에 요청을 발행하여 잘 정의된 인터페이스를 통해 분산 객체나 서버에서 서비스를 요청하는 클라이언트를 기반으로 합니다. 원격 메소드 요청은 서비스 제공자와 실제 매개 변수(있는 경우)의 객체 이름(객체 참조라고 함)을 포함하여 수행되어야 하는 작업에 대한 정보를 전송합니다. CORBA는 객체 등록, 객체 위치 지정, 객체 활성화, 요청 멀티플렉싱 해제, 오류 처리, 마샬 및 작업 디스패치 등과 같은 많은 네트워킹 프로그래밍 작업을 자동으로 처리합니다.

이 절에서는 다음 내용을 설명합니다.

- 상호 운용성 정보
- ORB 정보
- RMI/IIOP 기능 정보
- 인증 과정 정보

상호 운용성 정보

상호 운용성은 엔터프라이즈 환경에서 여러 언어로 작성된 응용 프로그램을 함께 사용할 수 있는 능력을 의미합니다. 이러한 기존 응용 프로그램 중 하나 이상은 개인용 컴퓨터 플랫폼에서 실행되고 다른 일부는 UNIX에서 실행될 수 있습니다. 또한 이러한 엔터프라이즈 환경은 J2EE 플랫폼이 직접 지원하지 않는 독립형 Java 기술 기반 응용 프로그램을 지원할 수도 있습니다.

J2EE는 CORBA IIOP (Internet Inter-Orb Protocol) 프로토콜에 대한 지원을 제공해야 합니다. CORBA는 네트워크에 분산되어 있는 객체 간의 상호 운용성을 사용자에게 투명하게 지정하는 모델을 정의합니다. CORBA는 구현에 독립적인 방식을 통해, 외부에서 볼 수 있는 분산 객체의 특성을 지정하는 방법을 정의함으로써 이런 기능을 수행합니다.

ORB 정보

Object Request Broker (약자: ORB)는 CORBA의 핵심 구성 요소입니다. ORB는 객체를 식별하거나 찾고, 연결 관리를 처리하고, 데이터를 전송하며, 통신을 요청하기 위해 필요한 인프라를 제공합니다.

CORBA 객체는 서로 직접 통신하지 않습니다. 대신 객체는 로컬 시스템에서 실행 중인 ORB에 대한 원격 스텁을 통해 요청을 합니다. 그러면 로컬 ORB는 IIOP (Internet Inter-Orb Protocol)를 사용하여 다른 시스템의 ORB에 이 요청을 전달합니다. 그런 다음 원격 ORB는 요청을 처리하는 적절한 객체(서번트)를 찾아서 결과를 반환합니다. IIOP는 RMI-IIOP 기술을 사용하여 JAVA 응용 프로그램이나 객체에 의해 RMI (Remote Method Invocation)로 사용될 수 있습니다.

RMI/IIOP 기능 정보

CORBA는 응용 프로그램이 위치에 관계 없이 서로 통신할 수 있도록 해주는 ORB를 지정합니다. 이러한 상호 운용성은 IIOP를 통해 제공되며 일반적으로 인트라넷 설정에서 찾아볼 수 있습니다. IIOP를 통해 RMI에서 얻을 수 있는 기능은 다음과 같습니다.

- 다른 언어로 작성된 언어와의 상호 운용성
- 트랜잭션 및 보안 컨텍스트를 전파할 수 있는 기능
- ORB 서비스에 대한 플러그 앤 플레이 환경
- EJB와의 상호 운용성
- COSNaming 서비스, IIOP 기반 이름 지정 서비스의 사용. EJB 상호 운용성 프로토콜을 사용하려면 JNDI (Java Naming Directory Interface) API를 사용하여 EJB 객체를 찾기 위해 COSNaming을 사용해야 합니다.

Sun ONE Application Server와 함께 제공되는 JAVA ORB는 다음 기능을 지원합니다.

- CSIv2 (Common Secure Interoperabillity version 2)의 레벨 0 호환
- IDL 인터페이스를 구현하며 EJB 컨테이너가 EJBHome 참조를 게시할 수 있도록 돕는 완전하게 호환되는 COSNaming 서비스
- IIOP/GIOP Ver 1.2. CORBA는 응용 프로그램이 위치에 관계 없이 서로 통신할 수 있도록 해주는 ORB를 지정합니다. 이러한 상호 운용성은 IIOP를 통해 제공됩니다.

인증 과정 정보

인증은 신원을 확인하는 과정입니다. 네트워크 상호 작용의 측면에서 인증은 한쪽에서 다른 한쪽을 식별하는 것입니다. 인증서는 인증을 지원하는 한 가지 방법입니다.

다음 두 가지 종류의 인증을 사용할 수 있습니다.

서버 인증. 서버 인증은 클라이언트에 의한 서버의 인증입니다. 즉, 특정 네트워크 주소의 서버에 대한 책임을 가지는 것으로 가정되는 조직의 신원입니다.

클라이언트 인증. 클라이언트 인증은 서버에 의한 클라이언트 인증입니다. 즉, 클라이언트 소프트웨어를 사용하고 있는 것으로 가정되는 사람의 신원입니다.

사람이 몇 개의 신분증을 가질 수 있는 것처럼 클라이언트에도 여러 개의 인증서가 있을 수 있습니다.

ORB 구성

Sun ONE Application Server의 각 인스턴스에 대해 여러 개의 IIOP-listener를 구성할 수 있습니다. 기본적으로 하나의 IIOP listener가 구성됩니다. ORB에 대해 IIOP Listener 등록 정보를 구성하여 Listener를 더 추가할 수 있습니다.

또한 ORB의 모니터링을 활성화하고, 메시지를 기록할 로그 레벨을 지정하고, 스레드 풀 설정을 지정하며, IIOP 경로에 대한 IIOP Listener와 SSl 구성을 구성할 수 있습니다. 이 절에서는 Sun ONE Application Server의 인스턴스에 대해 ORB 지원을 구성하는 방법에 대해 설명합니다.

이 절에서는 다음 내용을 설명합니다.

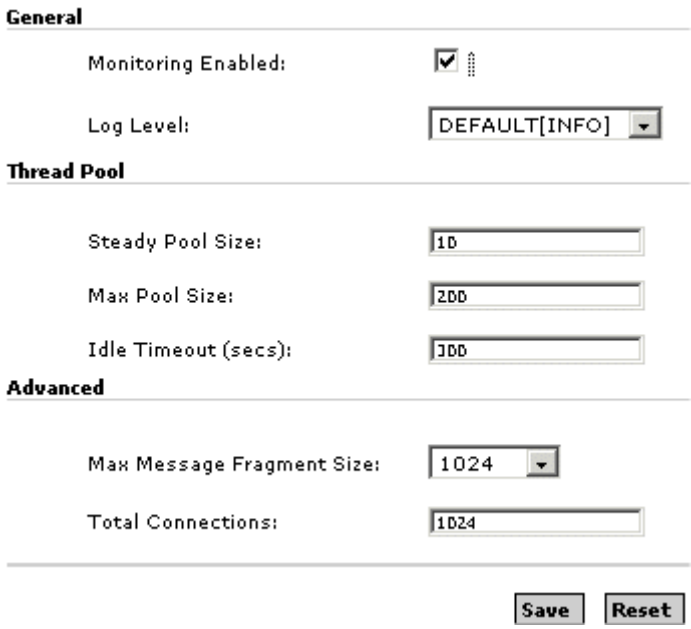
- 일반 ORB 구성 작업
- ORB에 대해 IIOP Listener 구성

일반 ORB 구성 작업

관리 인터페이스를 사용하면 모니터링을 활성화하고, 로그 레벨을 설정하고, 스레드 풀에 대한 풀 설정을 구성할 수 있습니다. 이러한 일반 ORB 구성 작업을 수행하려면 다음을 수행하십시오.

1. 관리 인터페이스의 왼쪽 창에서 ORB 설정을 구성할 Sun ONE Application Server 인스턴스를 확장합니다.
2. ORB 탭을 누릅니다. 관리 인터페이스의 오른쪽 창에 “일반 ORB 구성” 그림이 표시됩니다.

일반 ORB 구성



3. 이 창의 "일반" 섹션에서는 모니터링을 활성화하고 ORB에 대한 로그 수준을 설정할 수 있습니다.

   a. ORB 모니터링을 활성화하려면 "모니터링 사용" 확인란을 선택하십시오.

   b. "로그 수준" 드롭다운 목록에서 원하는 로그 수준을 선택합니다. 서버의 기본 로그 수준은 일반적으로 "정보"로 설정됩니다. ORB의 기본 수준은 서버에 대한 기본값을 사용하기 위한 것입니다. 따라서 로그 수준은 드롭다운 목록에 "기본값(정보)"가 표시됩니다.

      로그 수준은 "최고"부터 "치명적 오류"까지 심각도 범위의 메시지를 기록하기 위해 제공됩니다. 로그 수준을 설정하면 로그에 메시지를 어느 정도 자세하게 표시할지 선택할 수 있습니다. "경고"의 경우 "경고", "경보", "심각한 오류" 및 "치명적 오류" 메시지를 표시합니다. 일반적으로 서버 전반에 걸친 수준으로 설정해야 하지만 이 설정을 사용하여 Sun ONE Application Server ORB에서 표시되는 메시지를 제어할 수 있습니다.

4. 이 창의 "스레드 풀" 섹션에서는 ORB에서 사용하는 요청 스레드의 풀 설정을 지정할 수 있습니다.

   요청 스레드는 응용 프로그램 구성 요소에 대한 사용자 요청을 처리합니다. Sun ONE Application Server는 요청을 받으면 요청을 스레드 풀의 여유 스레드에 할당합니다. 그러면 스레드는 사용자의 요청을 실행하여 결과를 반환합니다. 예를 들어, 요청이 현재 작업 중인 시스템 자원을 사용해야 하는 경우 스레드는 자원의 작업이 끝날 때까지 기다린 후 요청이 해당 자원을 사용할 수 있도록 합니다.

   응용 프로그램에서 요청에 예약되는 스레드의 최소 수와 최대 수를 지정할 수 있습니다. 스레드 풀은 이러한 두 값 사이에서 동적으로 조절됩니다. 지정한 최소 스레드 풀 크기는 응용 프로그램에 대한 예약에 적어도 그 수만큼의 스레드를 할당하라는 신호를 ORB에게 보냅니다. 이 수는 지정한 최대 스레드 풀 크기만큼 증가됩니다.

   프로세스가 사용할 수 있는 스레드의 수를 늘리면 프로세스는 더 많은 응용 프로그램 요청에 동시에 응답할 수 있습니다.

   a. "고정 풀 크기" 필드에서 풀에 있는 스레드의 최소 수를 지정합니다. 스레드가 "유휴 시간 초과(초)" 필드에 지정된 기간 동안 유휴 상태이면 풀은 다시 이 수만큼 줄어듭니다.

   b. "최대 풀 크기" 필드에서는 스레드 풀이 늘어날 수 있는 최대 스레드 수를 지정합니다.

   c. "유휴 시간 초과(초)" 필드에서는 스레드 풀의 유휴 스레드를 정리할 초과 시간을 지정합니다.

5. 이 창의 "고급" 섹션에서는 ORB에 대한 다음 고급 옵션을 구성할 수 있습니다.

   a. "메시지 조각 크기" 필드에서는 조각을 지원하기 위한 최대 GIOP 1.2 메시지 크기를 지정합니다. 기본 조각 크기는 1024입니다.

   b. "전체 연결 수" 필드에서는 ORB 서버 프로세스가 허용하는 들어오는 원격 IIOP 연결의 최대 수를 지정합니다.

6. "저장"을 눌러서 설정을 저장합니다. 최근 변경 내용을 저장하지 않고 이전 설정을 되돌리려면 "되돌리기"를 누릅니다.

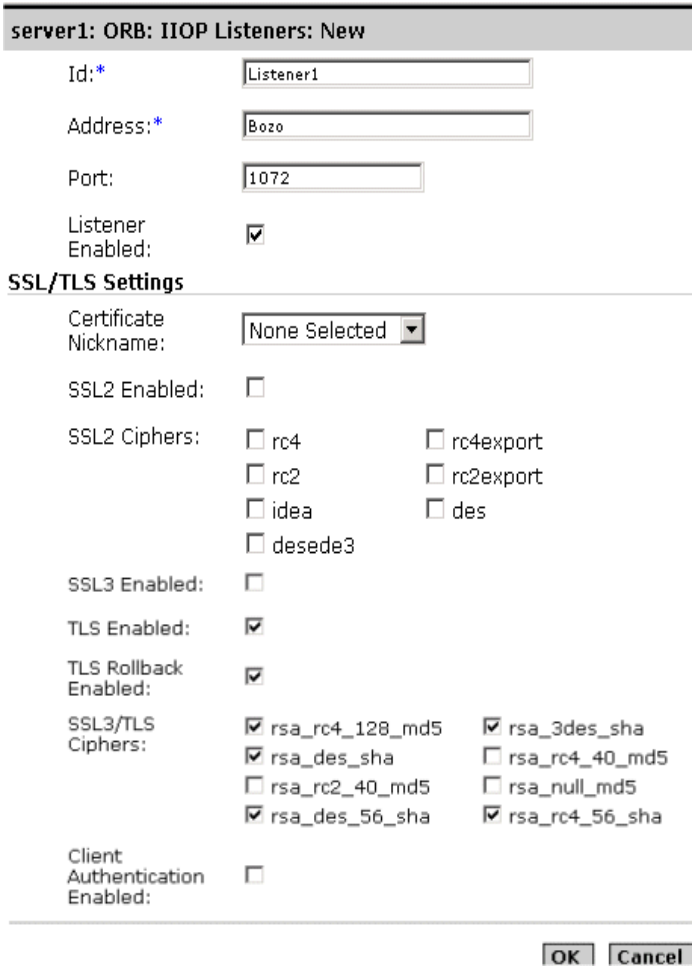
ORB에 대해 IIOP Listener 구성

Sun ONE Application Server의 각 인스턴스에는 미리 구성된 IIOP Listener가 포함된 기본 ORB 구성이 함께 제공됩니다. IIOP Listener는 특정 포트에서 수신하여 CORBA 기반 클라이언트 응용 프로그램에서 들어오는 연결을 받아들이는 수신 소켓입니다. 하나의 Sun ONE Application Server 인스턴스에 원하는 수의 IIOP Listener를 구성할 수 있습니다.

새 IIOP listener를 만들거나 IIOP Listener 등록 정보를 구성하려면 다음 작업을 수행하십시오.

1. 관리 인터페이스의 왼쪽 창에서 ORB 등록 정보를 구성할 Sun ONE Application Server 인스턴스를 확장합니다.
2. ORB를 누른 후 아래의 "IIOP Listener" 탭을 엽니다. 특정 Sun ONE Application Server 인스턴스에 대해 구성된 모든 IIOP Listener의 목록이 나타납니다.
3. 새 IIOP IIOP Listener를 만들려면 "새로 만들기"를 누릅니다. 기존 IIOP Listener를 편집하는 경우 Listener를 열고 다음 단계의 작업을 수행하십시오. "새로 만들기"를 누르거나 기존 IIOP Listener를 열면 “새 IIOP Listener 작성”이 나타납니다.

새 IIOP Listener 작성



4. IIOP Listener에 대한 일반 매개 변수를 다음과 같이 구성할 수 있습니다.

   a. "Id" 텍스트 필드에 Listener를 식별할 이름을 지정합니다. *ORB_Listener1*, *ORB_Listener2* 등과 같은 원하는 ID를 사용할 수 있습니다.

   b. "주소" 텍스트 필드에 Sun ONE Application Server를 설치한 시스템의 주소를 입력합니다 시스템 주소는 제공된 예와 같이 `machinename.domainname` 형식으로 지정하거나 시스템의 IP 주소를 지정할 수 있습니다.

c. "포트" 텍스트 필드에 새 IIOP Listener의 고유 포트 번호를 입력합니다. 기본 IIOP Listener에는 기본 포트 번호가 함께 제공됩니다. 이 포트 번호를 변경할 수 있습니다. 하지만 포트 번호를 변경하기 전에 자신이 지정한 새 포트 번호가 다른 기존 소프트웨어 응용 프로그램이나 프로세스에서 사용하고 있지 않은지 확인해야 합니다.

d. Listener를 활성화하려면 "Listener 사용" 확인란을 선택합니다.

5. 이 페이지의 "SSL/TLS 설정" 섹션에서 IIOP Listener에 대한 보안을 설정할 수 있습니다. 모든 암호화를 포함하여 Secure Sockets Layer (SSL) 및 Transport Layer Security (TLS)에 연결된 적절한 상자를 확인합니다. SSL2 또는 SSL3/TLS 소켓을 선택할 수 있습니다. 다음과 같이 Listener에 맞는 SSL/TLS 설정을 구성할 수 있습니다.

a. "인증서 별명" 필드에서 서버가 SSL 핸드셰이크 동안 클라이언트에게 제시할 인증서의 별명을 지정합니다. 이전에 인증서를 설치해야 해당 인증서의 별명이 이 목록에 나타납니다.

b. Listener 경로에 대해 SSL2 보안 옵션을 사용하려면 "SSL2 사용" 필드를 선택합니다.

c. SSL2 보안에 사용할 SSL2 암호화를 선택합니다. 필요한 암호화의 확인란을 선택합니다. 특정 암호 제품군을 사용하면 안 되는 특별한 이유가 없는 한 모두 허용해야 합니다.

d. Listener 경로에 대해 SSL3 보안 옵션을 사용하려면 "SSL3 사용" 필드를 선택합니다.

e. "TLS 사용" 필드를 선택하여 TLS를 활성화합니다. 서버에 액세스하려는 브라우저에서 TLS를 활성화해야 합니다. Netscape Navigator 6.0의 경우에는 TLS와 SSL3를 모두 선택합니다.

f. "TLS 롤백 사용" 필드를 선택합니다. TLS 롤백을 활성화하려면 TLS를 먼저 활성화해야 합니다. 또한 이 옵션을 활성화한 경우 SSL3과 SSL2를 비활성화했는지 확인하십시오. Microsoft Internet Explorer 5.0 및 5.5의 경우에는 TLS 롤백 옵션을 사용하십시오.

g. SSL3과 TLS에 사용할 SSL3/TLS 암호화를 선택합니다. 이 암호화는 SSL3 또는 TLS를 활성화한 경우에만 선택하십시오. 특정 암호 제품군을 사용하면 안 되는 특별한 이유가 없는 한 모두 허용해야 합니다.

h. "클라이언트 인증 사용" 확인란을 선택하여 클라이언트 인증이 있는 SSL IIOP 연결에 대한 ORB Listener의 사용 여부를 지정합니다. 클라이언트 인증은 신뢰할 수 있는 CA 목록의 CA에 대해 인증서 서명과 인증서 체인을 암호로 확인함으로써 클라이언트 인증서를 인증하는 과정입니다.

6. "확인"을 눌러서 IIOP Listener 설정을 저장합니다.

주

- Sun ONE Application Server를 설치하면 기본 서버 인스턴스에 대해 IIOP Listener가 생성됩니다. 기본 IIOP Listener 포트에 대한 기본 포트 번호는 3700입니다.
- 각 IIOP Listener의 포트 번호는 서로 달라야 합니다. 또한 "주소" 텍스트 필드에 지정한 시스템 주소는 Sun ONE Application Server가 설치된 시스템의 주소여야 합니다.
- listener 경로를 위한 SSL 설정에 대한 자세한 내용과 Sun ONE Application Server의 보안에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Guide to Security*를 참조하십시오.

응용 프로그램 배포

이 장에서는 다양한 Sun ONE Application Server 모듈 및 응용 프로그램을 배포하는 방법에 대해 설명합니다.

Sun ONE Application Server 모듈 및 응용 프로그램에는 J2EE 표준 요소 및 Sun ONE Application Server 고유 요소가 포함되어 있습니다. 이 장에서는 Sun ONE Application Server 고유 요소만 설명합니다.

배포를 위해 모듈 및 응용 프로그램을 패키지하고 어셈블하는 방법에 대해서는 *Sun ONE Application Server Developer's Guide*를 참조하십시오.

이 장에서는 다음 내용을 설명합니다.

- J2EE 모듈 정보
- J2EE 응용 프로그램 정보
- J2EE 표준 설명자
- Sun ONE Application Server 설명자
- 이름 지정 표준
- 디렉토리 구조 배포
- 런타임 환경
- 클래스 로더 정보
- 모듈 및 응용 프로그램 배포
- 응용 프로그램 배포 설명자 파일

J2EE 모듈 정보

J2EE 모듈은 해당 유형의 배포 설명자가 있는 동일한 컨테이너 유형의 하나 이상의 J2EE 구성 요소 모음입니다. 두 설명자 중 하나는 J2EE 표준이고 다른 하나는 Sun ONE Application Server 고유 설명자입니다. J2EE 모듈의 유형은 다음과 같습니다.

- **WAR (Web Application Archive):** 웹 응용 프로그램은 번들화하여 여러 J2EE 응용 프로그램 서버로 배포할 수 있는 서블릿, HTML 페이지, 클래스 및 기타 자원의 모음입니다. WAR 파일은 서블릿, JSP, JSP 태그 라이브러리, 유틸리티 클래스, 정적 페이지, 클라이언트측 애플릿, Bean, Bean 클래스 및 배포 설명자(`web.xml`과 선택적 `sun-web.xml`)로 구성될 수 있습니다.
- **EJB JAR 파일:** EJB JAR 파일은 Enterprise Bean을 어셈블하는 표준입니다. 이 파일에는 Bean 클래스(home, remote, local 및 implementation), 모든 유틸리티 클래스 및 배포 설명자(`ejb-jar.xml`과 선택적 `sun-ejb-jar.xml`)가 포함됩니다. EJB가 컨테이너 관리 영속성을 갖는 Entity Bean인 경우 CMP 배포 설명자인 `sun-cmp-mapping.xml`도 포함될 수 있습니다.
- **응용 프로그램(RMI/IIOP) 클라이언트 JAR 파일:** RMI/IIOP 클라이언트는 J2EE 클라이언트의 Sun ONE Application Server 고유 유형입니다. RMI/IIOP 클라이언트는 표준 J2EE Application Client 사양을 지원하며 Sun ONE Application Server에 대한 직접 액세스도 지원합니다. 이것의 배포 설명자는 `application-client.xml`과 선택적 `sun-application-client.xml`입니다.
- **자원 RAR 파일:** J2EE CA 커넥터에 적용되는 RAR 파일입니다. 커넥터 모듈은 장치 드라이버와 비슷합니다. 이것을 사용하면 EJB가 외부 엔터프라이즈 시스템에 이식 가능한 방식으로 액세스할 수 있습니다. 각 Sun ONE Application Server 커넥터에는 J2EE XML 파일인 `ra.xml`이 있습니다. 커넥터에는 Sun ONE Application Server 배포 설명자인 `sun-ra.xml`도 있어야 합니다.

모든 모듈의 소스 코드에는 모듈이 배포된 후 클래스 로더가 클래스를 제대로 찾을 수 있도록 패키지 정의를 사용해야 합니다.

배포 설명자의 정보는 선언적이기 때문에 소스 코드를 수정하지 않고 변경할 수 있습니다. 런타임에서 J2EE 서버는 이 정보를 읽고 그에 따라 작동합니다.

EJB JAR 및 웹 모듈은 다음 그림에서처럼, 응용 프로그램 외부에서 별도의 `.jar` 또는 `.war` 파일로 어셈블하고 별도로 배포할 수 있습니다.

J2EE 응용 프로그램 정보

J2EE 응용 프로그램은 응용 프로그램 배포 설명자에 의해 함께 결합된 하나 이상의 J2EE 모듈의 논리적 모음입니다. 구성 요소는 모듈 또는 응용 프로그램 수준에서 어셈블할 수 있습니다. 또한 구성 요소를 모듈 또는 응용 프로그램 수준에서 배포할 수도 있습니다.

구성 요소는 모듈로 어셈블된 다음 배포할 준비가 된 Sun ONE Application Server 응용 프로그램 `.ear` 파일로 어셈블됩니다.

각 모듈에는 Sun ONE Application Server 배포 설명자와 J2EE 배포 설명자가 있습니다. Sun ONE Application Server 관리 인터페이스에서는 배포 설명자를 사용하여 응용 프로그램 구성 요소를 배포하고 Sun ONE Application Server에 자원을 등록합니다.

응용 프로그램은 하나 이상의 모듈, 선택적 Sun ONE Application Server 배포 설명자 및 필수 J2EE 응용 프로그램 배포 설명자로 구성됩니다. 모든 항목은 Java ARchive(`.jar`) 파일 형식을 사용하여 확장명이 `.ear`인 하나의 파일로 어셈블됩니다.

J2EE 표준 설명자

J2EE 플랫폼은 어셈블리 및 배포 기능을 제공합니다. 이러한 기능은 JAR 파일을 구성 요소 및 응용 프로그램에 대한 표준 패키지로 사용하고 XML 기반 배포 설명자를 매개 변수 사용자 정의에 사용합니다. J2EE 어셈블리 및 배포 프로세스에 대한 자세한 내용은 *Developing Enterprise Applications with the J2EE*, v 1.0, 7장을 참조하십시오.

J2EE 표준 배포 설명자는 J2EE 사양, v1.3에서 설명합니다.

배포 전에 이러한 배포 설명자가 맞는지 확인하려면 *Sun ONE Application Server Developer's Guide*에서 배포 설명자 검증자에 대한 정보를 참조하십시오.

다음 표 "J2EE 표준 설명자"에서는 J2EE 표준 배포 설명자에 대한 정보를 볼 수 있는 위치를 설명합니다. 왼쪽 열은 배포 설명자이고 오른쪽 열은 해당 설명자에 대한 자세한 내용을 찾을 수 있는 참조 위치입니다.

J2EE 표준 설명자

배포 설명자	세부 정보 참조 위치
`application.xml`	Java 2 Platform Enterprise Edition Specification, v1.3, 8장, "Application Assembly and Deployment - J2EE:application XML DTD"
`web.xml`	Java Servlet Specification, v2.3 13장, "Deployment Descriptor", and JavaServer Pages Specification, v1.2, 7장, "JSP Pages as XML Documents", 및 5장, "Tag Extensions"
`ejb-jar.xml`	Enterprise JavaBeans Specification, v2.0, 16장, "Deployment Descriptor"
`application-client.xml`	Java 2 Platform Enterprise Edition Specification, v1.3, 9장, "Application Clients - J2EE:application-client XML DTD"
`ra.xml`	Java 2 Enterprise Edition, J2EE Connector Architecture Specification, v1.0, 10장, "Packaging and Deployment."

다음 위치에서 사양을 볼 수 있습니다.

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

Sun ONE Application Server 설명자

Sun ONE Application Server에서는 추가 배포 설명자를 사용하여 Sun ONE Application Server에 한정된 기능을 구성합니다. 이러한 기능은 커넥터 모듈에 필수인 `sun-ra.xml` 파일을 제외하면 선택 사항입니다.

배포 전에 이러한 배포 설명자가 맞는지 확인하려면 *Sun ONE Application Server Developer's Guide*에서 배포 설명자 검증자에 대한 정보를 참조하십시오.

다음 표 "Sun ONE Application Server 설명자"에서는 Sun ONE Application Server 배포 설명자에 대한 자세한 내용을 볼 수 있는 위치를 설명합니다. 왼쪽 열은 배포 설명자이고 오른쪽 열은 해당 설명자에 대한 자세한 내용을 찾을 수 있는 참조 위치입니다.

Sun ONE Application Server 설명자

배포 설명자	세부 정보 참조 위치
`sun-application.xml`	356페이지의 "응용 프로그램 배포 설명자 파일".
`sun-web.xml`	*Sun ONE Application Server Developer's Guide to Web Applications*
`sun-ejb-jar.xml` 및 `sun-cmp-mapping.xml`	*Sun ONE Application Server Developer's Guide to Enterprise Java Beans*
`sun-application-client.xml` 및 `sun-acc.xml`	*Sun ONE Application Server Developer's Guide to Clients?*
`sun-ra.xml`	*Sun ONE J2EE CA Service Provider Implementation Administrator's Guide*

주 UNIX에서 Sun ONE Application Server 배포 설명자는 600 수준의 액세스 권한을 갖고 있어야 합니다.

모든 Sun ONE Application Server 배포 설명자에 대한 DTD 스키마 파일은 *install_dir*`/appserv/lib/dtds` 디렉토리에 있습니다.

이름 지정 표준

응용 프로그램 및 개별적으로 배포된 EJB JAR, WAR, 커넥터 RAR 모듈의 이름은 (`server.xml` 파일의 `name` 속성에 지정) Sun ONE Application Server에서 고유해야 합니다. 이름을 명시적으로 지정하지 않으면 파일 이름의 첫 번째 부분(`.war` 또는 `.jar` 확장명 제외)이 기본 이름이 됩니다. `server.xml`에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

응용 프로그램이 배포되면 개별 모듈이 있는 디렉토리의 이름에는 `_jar`, `_war` 및 `_rar` 접미사가 추가되므로 유형이 다른 모듈도 응용 프로그램 내에서 이름이 같을 수 있습니다. 응용 프로그램 내에서 유형이 같은 모듈은 이름이 고유해야 합니다. 또한 데이터베이스 스키마 파일 이름은 응용 프로그램 내에서 고유해야 합니다.

`ejb-jar.xml` 파일의 `<module-name>` 부분에서 볼 수 있는 모듈 파일 이름, EAR 파일 이름, 모듈 이름 및 `ejb-jar.xml` 파일의 `<ejb-name>` 부분에서 볼 수 있는 EJB 이름에는 Java 패키지와 같은 이름 지정 스키마를 사용하는 것이 좋습니다. 이렇게 패키지와 비슷한 이름 지정 스키마를 사용하면 이름 충돌을 방지할 수 있습니다. 이러한 이름 지정의 이점은 Sun ONE Application Server뿐만 아니라 다른 J2EE 응용 프로그램 서버에도 적용됩니다.

EJB에 대한 JNDI 조회 이름 역시 고유해야 합니다. 이 경우에도 일관된 명명 규칙을 사용하는 것이 좋습니다. 예를 들어, EJB 이름에 응용 프로그램 이름과 모듈 이름을 추가하는 것도 고유한 이름을 유지하는 한 가지 방법입니다. 이 경우 `mycompany.pkging.pkgingEJB.MyEJB`는 응용 프로그램 `pkging.ear`에 패키지된 `pkgingEJB.jar` 모듈의 EJB에 대한 JNDI 이름이 됩니다.

운영 체제에서 사용할 수 없는 공백이나 문자를 패키지 및 파일 이름에 사용하지 않도록 하십시오.

디렉토리 구조 배포

응용 프로그램을 배포할 때 개별 모듈이 있는 디렉토리 이름에는 `_jar`, `_war` 및 `_rar` 접미어가 붙습니다. `asadmin deploydir` 명령을 사용하여 EAR 파일 대신 디렉토리를 배포하는 경우 디렉토리 구조는 다음 규칙을 따라야 합니다.

모듈 및 응용 프로그램 디렉토리 구조는 J2EE 사양에 정의된 구조를 따라야 합니다.

다음은 웹 모듈, EJB 모듈 및 클라이언트 모듈이 포함된 간단한 응용 프로그램의 디렉토리 구조의 예입니다.

```
+ converter_1/
|--- converterClient.jar
|--+ META-INF/
|  |--- MANIFEST.MF
|  |--- application.xml
|  '--- sun-application.xml
|--+ war-ic_war/
|  |--- index.jsp
|  |--+ META-INF/
|  |  |--- MANIFEST.MF
|  '--+ WEB-INF/
|     |--- web.xml
|     '--- sun-web.xml
|--+ ejb-jar-ic_jar/
|  |--- Converter.class
|  |--- ConverterBean.class
|  |--- ConverterHome.class
|  '--+ META-INF/
|     |--- MANIFEST.MF
|     |--- ejb-jar.xml
|     '--- sun-ejb-jar.xml
'--+ app-client-ic_jar/
   |--- ConverterClient.class
   '--+ META-INF/
      |--- MANIFEST.MF
      |--- application-client.xml
      '--- sun-application-client.xml
```

다음은 개별적으로 배포된 커넥터 모듈의 디렉토리 구조의 예입니다.

```
+ MyConnector/
|--- readme.html
|--- ra.jar
|--- client.jar
|--- win.dll
|--- solaris.so
'--+ META-INF/
   |--- MANIFEST.MF
   |--- ra.xml
   '--- sun-ra.xml
```

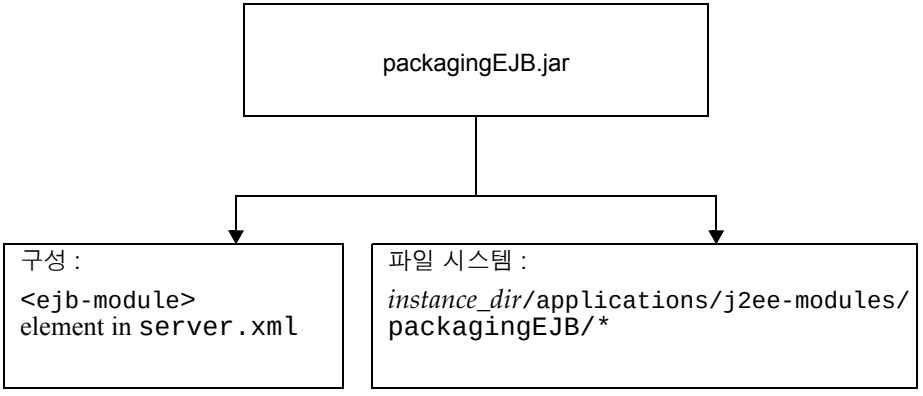
런타임 환경

구성 요소를 개별적으로 배포된 모듈로 배포하든지 응용 프로그램으로 배포하든지 상관없이 배포는 파일 시스템과 서버 구성에 영향을 줍니다. "모듈 런타임 환경" 및 "응용 프로그램 런타임 환경" 그림을 참조하십시오.

모듈 런타임 환경

다음 그림 "모듈 런타임 환경"은 개별적으로 배포된 모듈 기반 배포의 환경입니다.

모듈 런타임 환경



파일 시스템 항목의 경우 모듈은 다음과 같이 추출됩니다.

instance_dir/applications/j2ee-modules/*module_name*
instance_dir/generated/ejb/j2ee-modules/*module_name*
instance_dir/generated/jsp/j2ee-modules/*module_name*

generated/ejb 디렉토리에는 스텁과 타이가 포함되며 generated/jsp 디렉토리에는 컴파일된 JSP가 포함됩니다.

Lifecycle 모듈은 다음과 같이 추출됩니다.

instance_dir/applications/lifecycle-modules/*module_name*

구성 항목은 다음과 같이 server.xml에 추가됩니다.

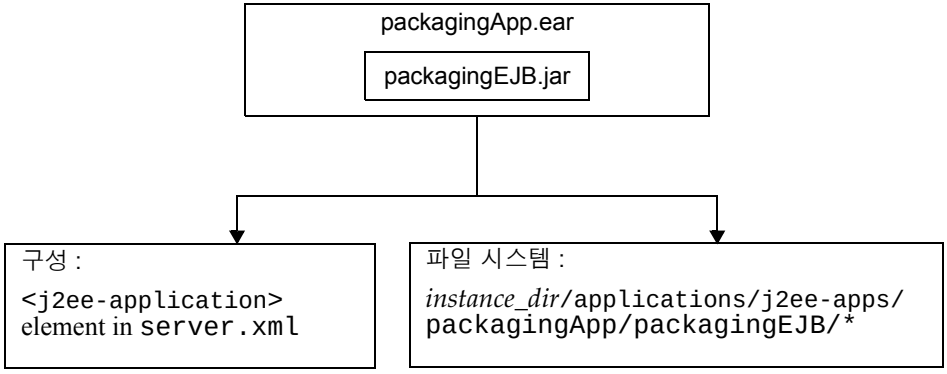
```
<server>
   <applications>
      <type-module>
         ...module configuration...
      </type-module>
   </applications>
</server>
```

server.xml에 있는 모듈의 *type*은 lifecycle, ejb, web 또는 connector일 수 있습니다. server.xml에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

응용 프로그램 런타임 환경

다음 그림 "응용 프로그램 런타임 환경"은 응용 프로그램 기반 배포를 위한 환경입니다.

응용 프로그램 런타임 환경



파일 시스템 항목의 경우 응용 프로그램은 다음과 같이 추출됩니다.

instance_dir/applications/j2ee-apps/*app_name*
instance_dir/generated/ejb/j2ee-apps/*app_name*
instance_dir/generated/jsp/j2ee-apps/*app_name*

generated/ejb 디렉토리에는 스텁과 타이가 포함되며 generated/jsp 디렉토리에는 컴파일된 JSP가 포함됩니다.

구성 항목은 다음과 같이 server.xml에 추가됩니다.

```
<server>
   <applications>
      <j2ee-application>
         ...application configuration...
      </j2ee-application>
   </applications>
</server>
```

server.xml에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

FastJavac 컴파일러를 사용하도록 server.xml 구성

기본적으로 Sun ONE Application Server는 내장 JDK 컴파일러를 사용하여 배포 도중 응용 프로그램을 컴파일합니다. 배포 도중 컴파일 속도가 더 빠른 Sun One Studio의 FastJavac 컴파일러를 사용할 수도 있습니다.

번들 Solaris 설치에서 FastJavac 컴파일러의 위치는 투명하지 않습니다. FastJavac 컴파일러를 사용하려면 Administration Server의 server.xml을 다음과 같이 컴파일러 경로로 구성해야 합니다.

server.xml의 java-config 요소에 다음 jvm-option을 추가합니다.

```
<java-config java-home="/<install-dir>/jdk" server-classpath="....." >

jvm-options>-Dcom.sun.aas.deployment.java.compiler=/<install-dir>/s
tudio4/bin/fastjavac/fastjavac.sun</jvm-options>

<property name="com.sun.aas.deployment.java.compiler.options"

value="-jdk /<install-dir>/jdk" />

</java-config>
```

클래스 로더 정보

Sun ONE Application Server 클래스 로더에 대해 이해하고 있으면 모듈 및 응용 프로그램에 대한 지원 JAR 및 자원 파일을 위치시키는 방법 및 위치를 결정하는 데 도움이 됩니다.

Java 가상 머신(JVM)에서 클래스 로더는 종속성을 해결하는 데 필요한 특정 Java 클래스 파일을 동적으로 로드합니다. 예를 들어, `java.util.Enumeration`의 인스턴스를 작성해야 하는 경우 클래스 로더 중 하나가 관련 클래스를 환경으로 로드합니다. 클래스 로더에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guide*를 참조하십시오.

모듈 및 응용 프로그램 배포

이 절에서는 J2EE 응용 프로그램 및 모듈을 Sun ONE Application Server에 배포하는 여러 방법에 대해 설명합니다. 이 절에서는 다음 항목에 대해 설명합니다.

- 배포 이름 및 오류
- 배포 lifecycle
- 모듈이나 응용 프로그램의 배포
- WAR 모듈 배포
- EJB JAR 모듈 배포
- Lifecycle 모듈 배포
- RMI/IIOP 클라이언트 배포
- J2EE CA 자원 어댑터 배포
- 정적 내용 배포
- 공유 프레임워크에 액세스

배포 이름 및 오류

응용 프로그램이나 모듈을 배포할 때 `server.xml` 파일에 고유 이름이 생성됩니다. 이 이름을 변경하지 마십시오. 배포 도중 서버는 모든 이름 충돌을 감지하여 고유하지 않은 이름이 있는 응용 프로그램이나 모듈은 로드하지 않습니다. 이 경우 서버 로그에 메시지가 전송됩니다. 이름 지정에 대한 자세한 내용은 341페이지의 "이름 지정 표준"을 참조하십시오.

배포 도중 오류가 발생하면 응용 프로그램이나 모듈이 배포되지 않습니다. 응용 프로그램 내의 모듈에 오류가 있으면 전체 응용 프로그램이 배포되지 않습니다.

`server.xml`에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

배포 lifecycle

응용 프로그램을 처음 배포한 뒤 수정, 재로드, 다시 배포, 비활성화, 다시 활성화, 배포 해제(서버에서 제거)할 수 있습니다. 이 절에서는 이러한 배포 lifecycle에 관련된 다음 항목에 대해 설명합니다.

- 동적 배포
- 배포된 응용 프로그램 또는 모듈 비활성화
- 동적 재로드

동적 배포

서버를 다시 시작하지 않고 응용 프로그램이나 모듈을 배포, 다시 배포 및 배포 해제할 수 있습니다. 이것을 동적 배포라고 합니다.

주로 개발자를 위한 것이지만 동적 배포는 운영 환경에서 서버를 다시 시작하지 않고 응용 프로그램 및 모듈을 온라인시키는 데 사용할 수 있습니다. 재배포가 완료될 때마다 전송 시간의 세션은 무효화됩니다. 클라이언트는 세션을 다시 시작해야 합니다.

배포된 응용 프로그램 또는 모듈 비활성화

배포된 응용 프로그램이나 모듈을 서버에서 제거하지 않고 비활성화할 수 있습니다. 각 응용 프로그램과 모듈에는 `server.xml` 파일에 `enabled` 속성이 있으며 관리 인터페이스에 해당 인터페이스가 있으므로 이를 수정할 수 있습니다. `server.xml`에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

동적 재로드

동적 재로드를 활성화하면 코드를 변경한 경우 응용 프로그램이나 모듈을 다시 배포할 필요가 없습니다. 변경된 클래스 파일을 응용 프로그램이나 모듈의 배포 디렉토리에 복사하기만 하면 됩니다. 서버에서 변경 사항을 주기적으로 확인하여 변경 사항이 적용된 응용 프로그램을 자동 및 동적으로 다시 배포합니다.

코드 변경을 빠르게 테스트할 수 있기 때문에 개발 환경에서 유용합니다. 하지만 동적 재로드는 성능이 저하되기 때문에 작업 환경에는 적합하지 않습니다. 또한 재로드가 완료될 때마다 해당 전송 시간의 세션이 무효화됩니다. 클라이언트는 세션을 다시 시작해야 합니다.

동적 재로드를 활성화하려면 다음 중 하나를 수행할 수 있습니다.

- 관리 인터페이스를 사용합니다.
  a. 서버 인스턴스 아래에서 "응용 프로그램" 구성 요소를 엽니다.
  b. "응용 프로그램" 페이지로 이동합니다.
  c. "재로드 사용 가능" 확인란을 선택하여 동적 재로드를 활성화합니다.
  d. "재로드 폴링 간격" 필드에 초 단위 시간을 입력하여 응용 프로그램과 모듈의 코드 변경을 확인하고 동적으로 로드할 간격을 설정합니다.
  e. "저장" 버튼을 누릅니다.
  f. 서버 인스턴스 페이지로 이동하여 "변경 사항 적용" 버튼을 선택합니다.
- `server.xml` 파일의 `applications` 요소의 다음 속성을 편집합니다.
  - `dynamic-reload-enabled="true"`는 동적 재로드를 활성화합니다.
  - `dynamic-reload-poll-interval-in-seconds`는 응용 프로그램과 모듈의 코드 변경을 확인하고 동적으로 다시 로드할 간격을 설정합니다.

  `server.xml`에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

또한 새 서블릿 파일을 로드하거나, EJB 관련 변경 사항을 다시 로드하거나, 배포 설명자 변경 사항을 다시 로드하려면 다음을 수행해야 합니다.

1. 배포된 응용 프로그램의 루트에 `.reload`라는 이름의 빈 파일을 만듭니다.

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

또는 개별적으로 배포된 모듈의 경우 다음을 만듭니다.

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

2. 위의 변경 작업을 할 때마다 `.reload` 파일의 타임스탬프(UNIX에서 `touch .reload`)를 명시적으로 업데이트합니다.

JSP의 경우 `sun-web.xml` 파일에서 `jsp-config` 요소의 `reload-interval` 속성에 설정된 빈도에 따라 변경 사항이 자동으로 다시 로드됩니다. JSP의 동적 재로드를 비활성화하려면 `reload-interval="-1"`을 설정합니다.

배포 도구

이 절에서는 모듈과 응용 프로그램 배포에 사용할 수 있는 여러 도구에 대해 설명합니다. 배포 도구에는 다음 항목이 있습니다.

- asadmin 유틸리티
- 관리 인터페이스
- Sun ONE Studio

asadmin 유틸리티

`asadmin` 유틸리티를 사용하면 응용 프로그램 및 개별적으로 배포된 모듈을 로컬 서버에 배포하거나 배포 해제할 수 있습니다. 여러 시스템에 동시 배포는 지원되지 않습니다. 이 절에서는 `asadmin` 유틸리티에 대해 간략하게 설명합니다.

lifecycle 모듈을 배포하려면 353페이지의 “Lifecycle 모듈 배포”를 참조하십시오.

asadmin 배포

`asadmin deploy` 명령은 WAR, JAR, RAR 또는 EAR 파일을 배포합니다. 응용 프로그램을 배포하려면 명령에 `--type application`을 지정합니다. 개별 모듈을 배포하려면 `--type ejb`, `web`, `connector`를 지정합니다. 구문은 다음과 같습니다(선택적 매개 변수에는 기본값 사용).

```
asadmin deploy --user admin_user [--password admin_password] [--host
localhost] [-port 4848] [--secure | -s] [--virtualservers
virtual_servers] [--type application|ejb|web|connector] [--contextroot
contextroot] [--force=true] [--precompilejsp=false] [--name
component_name] [--upload=true] [--retrieve local_dirpath] [--instance
instance_name] filepath
```

예를 들어, 다음 명령은 개별 EJB 모듈을 배포합니다.

```
asadmin deploy --user jadams --password secret --host localhost
--port 4848 --type ejb --instance server1 packagingEJB.jar
```

asadmin deploydir

`asadmin deploydir` 명령은 공개 디렉토리 구조에 응용 프로그램이나 모듈을 배포합니다. 구조는 342페이지의 “디렉토리 구조 배포”에 지정된 것과 같아야 합니다.

instance_dir`/applications/j2ee-apps` 또는 *instance_dir*`/applications/j2ee-modules` 아래의 *dirpath* 위치를 보면 응용 프로그램인지 개별적으로 배포된 모듈인지 판별할 수 있습니다. 구문은 다음과 같습니다(선택적 매개 변수에는 기본값 사용).

`asadmin deploydir --user` *admin_user* `[--password` *admin_password*`] [--host localhost] [-port 4848] [--secure | -s] [--virtualservers` *virtual_servers*`] [--type application|ejb|web|connector] [--contextroot` *contextroot*`] [--force=true] [--precompilejsp=false] [--name` *component_name*`] [--instance` *instance_name*`]` *dirpath*

예를 들어, 다음 명령은 개별 EJB 모듈을 배포합니다.

```
asadmin deploydir --user jadams --password secret --host localhost
--port 4848 --type ejb --instance server1 packagingEJB
```

asadmin undeploy

`asadmin undeploy` 명령은 응용 프로그램이나 모듈의 배포를 해제합니다. 응용 프로그램 배포를 해제하려면 명령에 `--type app`를 지정합니다. 모듈 배포를 해제하려면 `--type ejb`, `web` 또는 `connector`를 지정합니다. 구문은 다음과 같습니다(선택적 매개 변수에는 기본값 사용).

`asadmin undeploy --user` *admin_user* `[--password` *admin_password*`] [--host localhost] [-port 4848] [--secure | -s] [--type application|ejb|web|connector] [--instance` *instance_name*`]` *component_name*

예를 들어, 다음 명령은 개별 EJB 모듈을 배포 해제합니다.

```
asadmin undeploy --user jadams --password secret --host localhost
--port 4848 --type ejb --instance server1 packagingEJB
```

관리 인터페이스

관리 인터페이스를 사용하여 모듈과 응용 프로그램을 로컬 및 원격 Sun ONE Application Server 사이트에 배포할 수 있습니다. 이 도구를 사용하려면 다음 단계를 수행합니다.

1. 서버 인스턴스 아래에서 응용 프로그램 구성 요소를 엽니다.
2. "엔터프라이즈 응용 프로그램", "웹 응용 프로그램", "커넥터 모듈" 또는 "EJB 모듈" 페이지로 이동합니다.
3. "배포" 버튼을 누릅니다.
4. 모듈이나 응용 프로그램에 대한 전체 경로를 입력한 다음(또는 "찾아보기"를 눌러서 찾은 다음) "확인" 버튼을 누릅니다.
5. 모듈 또는 응용 프로그램 이름을 입력합니다.

   해당 확인란을 선택하여 이미 존재하는 모듈이나 응용 프로그램도 다시 배포할 수 있습니다. 이 방법은 선택 사항입니다.
6. 가상 서버 이름 옆의 확인란을 선택하여 응용 프로그램이나 모듈을 하나 이상의 가상 서버에 지정합니다.
7. "확인" 버튼을 누릅니다.

lifecycle 모듈을 배포하려면 353페이지의 “Lifecycle 모듈 배포”를 참조하십시오.

Sun ONE Studio

Sun ONE Studio 4를 사용하여 J2EE 응용 프로그램과 모듈을 배포할 수 있습니다. Sun ONE Studio의 사용에 대한 자세한 내용은 *Sun ONE Studio 4, Enterprise Edition Tutorial*을 참조하십시오.

주 Sun ONE Studio에서는 모듈이나 응용 프로그램을 배포하는 것을 *실행한다*라고 합니다. 실행에는 서버가 실행되고 올바른 URL을 표시하여 모듈이나 응용 프로그램을 활성화하는 것도 포함됩니다.

모듈이나 응용 프로그램의 배포

응용 프로그램에 독립적인 응용 프로그램 또는 개별 모듈을 배포할 수 있습니다. 응용 프로그램 기반 또는 개별 모듈 기반 모듈의 런타임 및 파일 시스템 관계에 대해서는 344페이지의 “런타임 환경”에서 설명합니다.

다음 항목이 구성 요소에 액세스해야 하는 경우 일반적으로 개별 모듈 기반 배포가 많이 사용됩니다.

- 기타 모듈
- J2EE 응용 프로그램
- RMI/IIOP 클라이언트(모듈 기반 배포를 사용하여 RMI/IIOP 클라이언트, 서블릿 또는 EJB에서 Bean에 공유 액세스를 할 수 있음).

모듈은 EAR 파일과 결합되어 단일 모듈로 배포될 수 있습니다. 이것은 EAR의 모듈을 독립적으로 배포하는 것과 비슷합니다.

WAR 모듈 배포

WAR 모듈은 350페이지의 “배포 도구”에서 설명하는 방법 중 하나를 사용하여 배포합니다.

`-keepgenerated` 속성을 `sun-web.xml`의 `jsp-config` 속성에 추가하여 JSP에 대해 생성된 소스를 유지할 수 있습니다. WAR 모듈을 배포할 때 이 속성을 포함시키면 생성된 소스는, 응용 프로그램인 경우 *instance_dir*`/generated/jsp/j2ee-apps/`*app_name*`/`*module_name*에, 개별적으로 배포된 웹 모듈의 경우에는 *instance_dir*`/generated/jsp/j2ee-modules/`*module_name*에 보관됩니다. `-keepgenerated` 등록 정보에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guide to Web Applications*를 참조하십시오.

EJB JAR 모듈 배포

EJB JAR 모듈은 350페이지의 "배포 도구"에서 설명하는 방법 중 하나를 사용하여 배포합니다.

`server.xml`에서 `java-config` 요소의 `rmic-options` 속성에 `-keepgenerated` 플래그를 추가하여 스텁 및 타이에 대해 생성된 소스를 유지할 수 있습니다. EJB JAR 모듈을 배포할 때 이 플래그를 포함시키면 생성된 소스는, 응용 프로그램의 경우 *instance_dir*`/generated/ejb/j2ee-apps/`*app_name*`/`*module_name*에, 개별적으로 배포된 EJB JAR 모듈의 경우 *instance_dir*`/generated/ejb/j2ee-modules/`*module_name*에 유지됩니다. `-keepgenerated` 플래그에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

Lifecycle 모듈 배포

lifecycle 모듈에 대한 일반 정보는 *Sun ONE Application Server Developer's Guide*를 참조하십시오.

다음 도구를 사용하여 lifecycle 모듈을 배포할 수 있습니다.

- asadmin 유틸리티
- 관리 인터페이스

asadmin 유틸리티

lifecycle 모듈을 배포하려면 `asadmin create-lifecycle-module` 명령을 사용하십시오. 구문은 다음과 같습니다(선택적 매개 변수에는 기본값 사용).

`asadmin create-lifecycle-module --user` *admin_user* `[--password` *admin_password*`] [--host localhost] [-port 4848] [--secure | -s] [--instance` *instance_name*`] --classname` *classname* `[--classpath` *classpath*`] [--loadorder` *load_order_number*`] [--failurefatal=false] [--enabled=true] [--description` *text_description*`] [--property (`*name*`=`*value*`)[:`*name*`=`*value*`]*]` *modulename*

예:

```
asadmin create-lifecycle-module --user jadams --password secret
--host localhost --port 4848 --instance server1 --classname
RMIServer MyRMIServer
```

lifecycle 모듈의 배포를 해제하려면 `asadmin delete-lifecycle-module` 명령을 사용하십시오. 구문은 다음과 같습니다(선택적 매개 변수에는 기본값 사용).

`asadmin delete-lifecycle-module --user` *admin_user* `[--password` *admin_password*`] [--host localhost] [-port 4848] [--secure | -s] [--instance` *instance_name*`]` *module_name*

예:

```
asadmin delete-lifecycle-module --user jadams --password secret
--host localhost --port 4848 --instance server1 MyRMIServer
```

서버 인스턴스에 배포된 lifecycle 모듈을 나열하려면 `asadmin list-lifecycle-modules` 명령을 사용합니다. 구문은 다음과 같습니다(선택적 매개 변수에는 기본값 사용).

`asadmin list-lifecycle-modules --user` *admin_user* `[--password` *admin_password*`] [--host localhost] [-port 4848]` *instance_name*

예:

```
asadmin list-lifecycle-module --user jadams --password secret --host
localhost --port 4848 server1
```

관리 인터페이스

관리 인터페이스를 사용하여 lifecycle 모듈을 배포할 수도 있습니다. 다음 단계를 수행합니다.

1. 서버 인스턴스 아래에서 응용 프로그램 구성 요소를 엽니다.
2. "Life Cycle 모듈" 페이지로 이동합니다.
3. "배포" 버튼을 누릅니다.
4. 다음 정보를 입력합니다.
   - 이름(필수) - life cycle 모듈의 이름
   - 클래스 이름(필수) - life cycle 모듈의 클래스 파일의 정규화된 이름
   - 클래스 경로(선택) - life cycle 모듈의 클래스 경로. 모듈의 위치를 지정합니다. 기본 위치는 응용 프로그램 루트 디렉토리 아래입니다.
   - 로드 순서(선택) - 시작할 때 life cycle 모듈을 로드할 순서를 결정합니다. 정수 값이 작은 모듈이 먼저 로드됩니다. 값의 범위는 `101`부터 운영 체제의 `MAXINT`까지입니다. `1`부터 `100`까지의 값은 예약되어 있습니다.
   - 치명적 오류(선택) - life cycle 모듈이 실패한 경우 서버를 종료할지 결정합니다. 기본값은 `false`입니다.

- 활성화(선택) - life cycle 모듈의 활성화 여부를 결정합니다. 기본값은 `true`입니다.

5. "확인" 버튼을 누릅니다.

RMI/IIOP 클라이언트 배포

EJB와 통신하는 클라이언트에만 배포가 필요합니다. RMI/IIOP 클라이언트 배포는 세 단계로 수행됩니다.

1. RMI/IIOP 클라이언트가 액세스할 EAR 또는 EJB JAR을 배포합니다.
2. 필요한 클라이언트 파일을 어셈블하고 클라이언트를 배포합니다.
3. 배포 후 클라이언트 JAR 파일이 응용 프로그램의 다음 위치에 생성됩니다.

   instance_dir`/applications/j2ee-apps/`*app_name*`/`*app_name*`Client.jar`

   개별적으로 배포된 모듈의 경우 다음 위치에 생성됩니다.

   instance_dir`/applications/j2ee-modules/`*module_name*`/`*module_name*`Client.jar`

   클라이언트 JAR에는 타이 및 RMI/IIOP 클라이언트에 필요한 클래스가 포함되어 있습니다. 이 파일은 클라이언트 시스템에 복사하고 클라이언트의 APPCPATH 환경 변수가 이 JAR를 가리키도록 설정합니다.

이제 클라이언트 실행 준비가 끝났습니다. 자세한 내용은 *Sun ONE Application Server Developer's Guide to Clients*를 참조하십시오.

J2EE CA 자원 어댑터 배포

커넥터 모듈은 350페이지의 "배포 도구"에서 설명하는 방법 중 하나를 사용하여 배포합니다.

정적 내용 배포

정적 내용(HTML, 이미지 등)은 웹 서버와 Sun ONE Application Server에서 호스트할 수 있습니다. 하지만 WAR 파일이 등록되면 정적 내용이 응용 프로그램 서버에 배포됩니다. Sun ONE Application Server와 함께 제공되는 모든 샘플은 응용 프로그램 서버에서 정적 내용을 호스트합니다.

예를 들어, 응용 프로그램의 정적 파일 `index.html`에 액세스하려면 다음을 사용합니다.

`http://server:port/NASApp/&ltcontext_root/index.html`

공유 프레임워크에 액세스

J2EE 응용 프로그램 및 모듈이 공유 프레임워크 클래스(예: 구성 요소 및 라이브러리)를 사용하면 클래스를 응용 프로그램이나 모듈이 아니라 시스템 클래스 또는 공용 클래스 로드의 경로에 배포할 수 있습니다. 크기가 큰 공유 라이브러리를 해당 라이브러리를 사용하는 모든 모듈에 어셈블하면 파일 크기가 너무 커져서 서버 등록 시간이 오래 걸립니다. 또한 서로 다른 클래스 로드에 동일한 클래스의 여러 버전이 존재할 수 있으므로 자원을 낭비하게 됩니다.

시스템 클래스 로드에 대한 자세한 내용은 347페이지의 “클래스 로더 정보”를 참조하십시오.

응용 프로그램 배포 설명자 파일

Sun ONE Application Server 응용 프로그램에는 다음 두 개의 배포 설명자 파일이 있습니다.

- Java Servlet Specification, v2.3, 13장, “Deployment Descriptors”에 설명되어 있는 J2EE 표준 파일(`application.xml`)
- 이 절에서 설명한 선택적 Sun ONE Application Server 고유 파일 (`sun-application.xml`)

응용 프로그램 배포 설명자 파일에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guide*를 참조하십시오.

HTTP 서버 기능 및 가상 서버 관리

14장, “HTTP 기능 구성”

15장, “가상 서버 사용”

16장, “가상 서버 내용 관리”

HTTP 기능 구성

이 장에서는 Sun ONE Application Server의 HTTP 관련 기능에 대한 기본 설정을 구성하는 방법에 대해 설명합니다. 가상 서버 및 HTTP Listener에 관련된 기본 설정에 대한 자세한 내용은 15장, "가상 서버 사용"을 참조하십시오.

이 장에서는 다음 내용을 설명합니다.

- HTTP 기능에 대한 정보
- 파일 캐시 구성
- 서버 성능 조정
- HTTP 서비스 품질(QOS) 구성
- 스레드 풀 추가 및 사용
- 파일 캐시 구성고급 설정 편집
- MIME 유형 구성

HTTP 기능에 대한 정보

Sun ONE Application Server HTTP 기능에는 응용 프로그램 서버 인스턴스에 대한 성능 수준 설정, 성능 조정 관련 매개 변수 설정 및 파일 캐시를 사용한 성능 향상 등이 있습니다. 이러한 설정은 `init.conf`와 `server.xml`의 두 구성 파일에 저장됩니다. `init.conf` 설정은 "고급 설정" 페이지에서 편집합니다. 자세한 내용은 363페이지의 “고급 설정 편집”을 참조하십시오.

사용자가 편집하는 다른 등록 정보는 `server.xml` 파일의 `http-service` 요소에 저장됩니다. `init.conf` 파일 및 `server.xml` 파일에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

파일 캐시 구성

Sun ONE Application Server는 파일 캐시를 사용하여 정적 정보를 더 빠르게 서비스합니다. 파일 캐시에는 파일 및 정적 파일 내용에 대한 정보가 들어 있습니다. 또한 파일 캐시는 서버 구문 분석 HTML의 처리 속도를 높이는 데 사용되기도 합니다.

파일 캐시는 기본적으로 활성화됩니다. 파일 캐시 설정은 `nsfc.conf`라는 파일에 있습니다. 파일 캐시 매개 변수를 기본값이 아닌 값으로 변경한 경우에만 이 파일이 생성됩니다. `nsfc.conf`에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

파일 캐시를 설정하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "HTTP 서버"를 누릅니다.
2. "파일 캐싱" 탭을 누릅니다.
3. 필드에 원하는 값을 입력합니다.
4. "확인"을 누릅니다.

파일 캐시를 사용하여 성능을 향상시키는 방법에 대한 자세한 내용은 *Sun ONE Application Server Performance Tuning and Sizing Guide*를 참조하십시오.

서버 성능 조정

"성능 조정" 페이지에서는 처리 가능한 요청의 수, 활동이 없는 요청을 시간 초과 전까지 열어 둘 시간, DNS를 사용한 클라이언트의 IP 역조회 수행 여부를 제어하여 Sun ONE Application Server의 성능을 제어하는 설정을 구성할 수 있습니다. 또한 DNS를 사용하는 경우 비동기 DNS 및 DNS 캐시 설정의 사용 여부 등과 같은 성능 관련 기능도 설정할 수 있습니다.

조정에 대한 자세한 내용은 *Sun ONE Application Server Performance Tuning Guide*를 참조하십시오.

성능 조정 설정을 구성하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "HTTP 서버"를 누릅니다.
2. "조정" 탭을 누릅니다.
3. 필드에 원하는 값을 입력합니다.
4. "확인"을 누릅니다.

관리 인터페이스를 통해 조정할 수 있는 설정에 대한 추가 정보는 온라인 도움말을 참조하십시오.

HTTP 서비스 품질(QOS) 구성

서비스 품질(QOS)은 서버에 대해 설정하는 성능 제한을 가리킵니다. 예를 들어, ISP에서는 각 가상 서버에 허용된 대역폭의 양에 따라 서로 다른 요금을 부과할 수 있습니다.

특정 가상 서버에 대한 서비스 품질(QOS)을 사용하려면 서버 인스턴스에 대해 서비스 품질(QOS)을 활성화하고 몇 가지 값을 설정해야 합니다.

서버 인스턴스에 대한 서비스 품질(QOS) 설정을 구성하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "HTTP 서버"를 누릅니다.
2. "QOS" 탭을 누릅니다.
3. 서비스 품질(QOS) 전체를 활성화하려면 "활성화"를 누릅니다.

   기본적으로 서비스 품질은 비활성화되어 있습니다. 서비스 품질을 향상시키면 서버 오버헤드가 약간 증가합니다.
4. "재계산 간격"을 선택합니다.

   재계산 간격은 대역폭을 계산하는 간격을 밀리초 단위로 나타낸 것입니다. 기본값은 100밀리초입니다.
5. "메트릭 간격"을 선택합니다.

   메트릭 간격은 트래픽이 측정되는 초 단위 간격입니다. 기본값은 30초입니다. 이 시간 동안 측정되는 모든 대역폭의 평균값이 초당 바이트 수입니다.

   사이트에 큰 파일 전송량이 많은 경우 큰 값(몇 분 이상) 또는 이 필드를 사용하십시오. 큰 파일을 전송하면 짧은 메트릭 간격에 대해 허용된 모든 대역폭을 차지하게 되어 최대 대역폭 설정을 적용한 경우 연결이 거부될 수 있습니다. 대역폭은 메트릭 간격으로 평균값이 구해지므로 간격이 길어지면 큰 파일로 인한 성능 저하를 줄일 수 있습니다.

   대역폭 한계가 사용 가능한 대역폭보다 많이 낮은 경우(예를 들어, 초당 1MB의 대역폭 한계가 있지만 백본에 초당 1GB 연결이 있는 경우) 메트릭 간격이 더 짧아야 합니다.

큰 정적 파일을 전송하는데 대역폭 한계가 사용 가능한 대역폭보다 많이 낮은 경우 반대의 해결책이 필요할 수 있으므로 어떤 상황을 조정해야 할지 결정해야 합니다.

6. 서버에 대한 대역폭 한계를 초당 바이트 단위로 설정합니다.
7. 대역폭 한계 설정의 적용 여부를 선택합니다.

   대역폭 한계를 적용하기로 선택한 경우, 서버가 대역폭 한계에 도달하면 추가 연결이 거부됩니다.

   대역폭 한계를 적용하지 않은 경우에는 한계가 초과되면 서버에서 오류 로그에 메시지를 기록합니다.
8. 서버에 대해 허용된 최대 연결 수를 선택합니다.

   이 수는 처리된 동시 요청 수입니다.
9. 연결 한계 설정의 적용 여부를 선택합니다.

   연결 한계를 적용하기로 선택한 경우, 서버가 한계에 도달하면 추가 연결이 거부됩니다.

   연결 한계를 적용하지 않은 경우에는 한계가 초과되면 서버에서 오류 로그에 메시지를 기록합니다.
10. 추가 이름/값 쌍을 지정하려면 "등록 정보" 버튼을 누릅니다.
11. "확인"을 누릅니다.

명령줄 인터페이스의 `asadmin` 유틸리티를 사용하여 서비스 품질(QOS)을 구성하려면 다음 명령을 사용합니다.

- `create-http-qos`
- `delete-http-qos`

이들 명령에는 다음 구문을 사용합니다.

`asadmin create-http-qos --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`] [--virtualserver` *virtual_server_id*`] [--bwlimit` *bandwidth_limit*`] [--enforcebwlimit` *enforce_bandwidth_limit*`] [--connlimit` *connection_limit*`] [--enforceconnlimit` *enforce_connection_limit*`]` *instancename*

`asadmin delete-http-qos --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`][--virtualserver` *virtual_server_id*`]` *instancename*

가상 서버를 지정하면 이러한 명령은 해당 가상 서버에 대한 서비스 품질(QOS) 정보를 만들거나 삭제합니다. 가상 서버를 지정하지 않으면 명령은 서버 인스턴스에 영향을 줍니다.

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

서비스 품질 기능의 한계에 대한 자세한 내용은 151페이지의 “CLI를 사용한 트랜잭션 서비스 관리”를 참조하십시오.

스레드 풀 추가 및 사용

스레드 풀을 사용하여 특정 서비스에 일정한 수의 스레드를 지정하여 사용자가 원하는 스레드의 수를 초과하지 않도록 할 수 있습니다. 스레드 풀의 다른 용도는 스레드에 안전하지 않은 플러그 인을 실행하는 것입니다. 스레드의 최대 수가 1로 설정된 풀을 정의하면 지정한 서비스 기능에는 하나의 요청만 허용됩니다.

스레드 풀을 추가할 때 지정하는 정보에는 스레드의 최소 및 최대 개수, 스택 크기 및 대기열 크기가 포함됩니다.

스레드 풀을 추가하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "HTTP 서버"를 누릅니다.
2. "스레드 풀"을 누릅니다.
3. 필드에 원하는 값을 입력합니다.
4. "확인"을 누릅니다.

페이지 아래쪽에 스레드 풀이 표시됩니다. 스레드 풀을 편집하거나 삭제하려면 옆에 있는 "편집" 또는 "삭제" 버튼을 누릅니다.

스레드 풀을 설정한 뒤에 해당 스레드 풀을 특정 서비스의 스레드 풀로 지정하여 사용합니다.

스레드 풀을 사용하여 성능을 향상시키는 방법에 대한 자세한 내용은 *Performance Tuning and Sizing Guide*를 참조하십시오.

고급 설정 편집

Sun ONE Application Server는 시작할 때 *instance_dir*`/config/` 디렉토리의 `init.conf` 파일을 찾아서 서버의 동작 및 구성에 적용되는 전역 변수 설정 집합을 설정합니다. Sun ONE Application Server는 `init.conf`에 정의된 모든 지시문을 실행합니다.

이러한 설정은 "고급 설정" 페이지에 표시됩니다. `init.conf` 파일에서 다음 영역에 적용되는 특정 설정을 편집할 수 있습니다.

- DNS
- SSL
- 성능
- CGI
- 연결 유지
- 로깅

`init.conf` 파일에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

고급 설정을 편집하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 "HTTP 서버"를 누릅니다.
2. "고급" 탭을 누릅니다.
3. 변경할 설정의 유형(DNS, SSL 등)을 누릅니다.
4. 설정을 원하는 대로 변경한 후 "확인"을 누릅니다.

각 설정 유형에 대한 자세한 내용은 온라인 도움말을 참조하십시오.

MIME 유형 구성

"Mime 유형" 페이지에서는 서버의 MIME 파일을 편집할 수 있습니다. MIME (Multi-purpose Internet Mail Extension) 유형은 시스템에서 지원하는 멀티미디어 파일의 유형을 조절할 수 있습니다. 또한 MIME 유형은 CGI 프로그램이 어떤 파일인지 지정하는 것과 같이, 특정 서버 파일 유형에 속하는 파일 확장자를 지정합니다.

MIME 유형 파일은 원하는 수대로 만들 수 있으며 응용 프로그램 서버 인스턴스 또는 가상 서버와 연결할 수 있습니다. 서버에는 `mime.types`라는 하나의 MIME 유형 파일이 있는데 이 파일은 삭제할 수 없습니다.

새 MIME 유형 파일 작성

1. 왼쪽 창에서 "HTTP 서버"를 누르고 "MIME 유형 파일"을 누릅니다.
2. 오른쪽 창에서 "새로 만들기"를 누릅니다.

3. MIME 파일의 ID와 파일 이름을 입력합니다.
4. "확인"을 누릅니다.

MIME 파일의 정의를 편집하려면 다음 작업을 수행합니다.

1. 왼쪽 창의 "HTTP 서버" 아래에서 "MIME 유형 파일" 옆의 아이콘을 눌러서 보기를 확장합니다.
2. 편집할 MIME 파일의 ID를 누릅니다.
3. 이 페이지에서 ID에 연결된 MIME 파일 이름을 편집합니다.
4. MIME 파일의 파일 확장자를 편집하려면 "MIME 파일 편집"을 누릅니다.
5. 기존 항목을 편집하려면 해당 항목 옆의 "편집"을 누릅니다.
6. 나타나는 페이지에서 변경을 한 다음 "MIME 유형 변경"을 누릅니다.
7. MIME 유형을 삭제하려면 해당 항목 옆의 "제거"를 누릅니다.
8. 새 MIME 유형을 추가하려면 필드에 범주, 내용 유형, 파일 접미어를 입력한 다음 "새 유형"을 누릅니다.

명령줄 인터페이스의 asadmin 유틸리티를 사용하여 MIME 유형을 구성하려면 다음 명령을 사용합니다.

- `create-mime`
- `delete-mime`
- `list-mimes`

이 명령에는 다음 구문을 사용합니다.

`asadmin create-mime --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`] [--instance` *instancename*`] --mimefile` *filename mime_id*

`asadmin delete-mime --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`] [--instance` *instancename*`]` *mime_id*

`asadmin list-mimes --user` *admin_user* `[--password` *password*`] [--host` *hostname*`] [--port` *admin_port*`] [--secure | -s] [--passwordfile` *file_name*`]` *instancename*

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

가상 서버에 MIME 유형을 사용하는 방법에 대한 자세한 내용은 온라인 도움말 및 15장, "가상 서버 사용"을 참조하십시오.

15장

가상 서버 사용

이 장에서는 Sun ONE Application Server를 사용하여 가상 서버를 설정 및 관리하는 방법에 대해 설명합니다. 가상 서버 내용에 대한 설정을 구성하는 방법에 대한 자세한 내용은 16장, “가상 서버 내용 관리”를 참조하십시오.

이 장에서는 다음 내용을 설명합니다.

- 가상 서버 개요
- 가상 서버에서 Sun ONE Application Server 기능 사용
- HTTP Listener 만들기 및 구성
- 가상 서버 만들기 및 구성
- 가상 서버 배포

가상 서버 개요

가상 서버를 사용하는 경우, 설치된 단일 서버에 회사 또는 개인 도메인 이름과 IP 주소 및 서버 모니터링 기능을 제공할 수 있습니다. 사용자의 경우 하드웨어를 갖추고 가상 서버를 유지 관리하긴 하지만 고유 웹 서버를 가진 것과 거의 같습니다.

Sun ONE Application Server의 비번들 버전을 설치하면 응용 프로그램 인스턴스에 대한 기본 가상 서버가 만들어집니다. 즉, 기본 응용 프로그램 서버 인스턴스 `server1`에 대해 `server1`이라는 가상 서버가 또한 만들어집니다. Solaris 9 번들 버전을 사용하는 경우에는 서버 인스턴스를 만들어야 합니다. 서버 인스턴스를 만들면 이름이 같은 가상 서버가 또한 만들어집니다. 가상 서버는 이렇게 만든 각 추가 응용 프로그램 서버 인스턴스에 대해 만들어집니다. 가상 서버 만들기 및 구성에 대한 자세한 내용은 378페이지의 “가상 서버 만들기 및 구성”을 참조하십시오. 가상 서버 배포에 대한 자세한 내용은 "가상 서버 배포"를 참조하십시오.

가상 서버는 개별 가상 서버에서 사용 가능한 Sun ONE Application Server의 HTTP 기능을 제어합니다. 가상 서버를 여러 개 사용하지 않을 경우에도 응용 프로그램 서버 인스턴스를 사용하여 만든 기본 가상 서버를 구성하여 응용 프로그램 서버 인스턴스에 대한 특정한 등록 정보를 구성합니다.

가상 서버에 대한 설정은 *instance_dir*/`config` 디렉토리에 있는 `server.xml` 파일의 `virtual-server` 요소에 저장됩니다. 이 파일에 대한 자세한 내용은 *Sun ONE Application Server Administratorís Configuration File Reference*를 참조하십시오.

가상 서버에 대한 일부 정보는 `obj.conf` 파일에 저장됩니다. 각 가상 서버에는 별도의 `obj.conf` 파일이 있습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- HTTP Listener
- 가상 서버
- obj.conf 파일
- 요청 프로세스에 대한 가상 서버 선택
- 문서 루트
- 액세스 로그 파일과 서버 로그 파일 사용

HTTP Listener

서버와 클라이언트 간 연결은 수신 소켓이라고도 하는 HTTP Listener에서 이루어집니다. 사용자가 만든 각 HTTP Listener에는 IP 주소와 포트 번호, 서버 이름 반환 및 기본 가상 서버가 있습니다. HTTP Listener를 시스템의 해당 포트에 구성된 모든 IP 주소에서 수신하려면 IP 주소에 대해 0.0.0.0, any, ANY 또는 INADDR_ANY를 사용하십시오. HTTP Listener 만들기 및 구성에 대한 자세한 내용은 375페이지의 “HTTP Listener 만들기 및 구성”을 참조하십시오.

Sun ONE Application Server의 비번들 버전을 설치하면, 하나의 HTTP Listener (`http-listener-1`)가 자동으로 만들어집니다. 이 HTTP Listener는 IP 주소 0.0.0.0과 설치 중에 HTTP 서버 포트 번호로 지정한 포트 번호를 사용합니다. UNIX에서 기본 포트 번호는 루트로 설치하지 않는 경우 80 또는 1024입니다. 기본 HTTP Listener는 삭제할 수 없습니다. 여러 가상 서버를 사용 중인 경우, 모든 가상 서버에 대해 기본 HTTP Listener를 사용하거나 여러 HTTP Listener를 만들 수 있습니다.

Solaris 9 번들형 Sun ONE Application Server를 사용하는 경우, 서버 인스턴스를 만들 때 HTTP Listener가 생성됩니다. 이 HTTP Listener는 IP address 0.0.0.0과 인스턴스를 만들 때 지정한 포트 번호를 갖습니다.

HTTP Listener는 IP 주소와 포트 번호의 조합이므로, 여러 HTTP Listener의 IP 주소는 같고 포트 번호는 다를 수 있습니다 또는 IP 주소는 다르고 포트 번호는 같을 수 있습니다. 예를 들어, HTTP Listener는 1.1.1.1:81 및 1.1.1.1:82가 될 수 있으며, 1.1.1.1:81 및 1.2.3.4:81가 될 수도 있습니다. 이는 시스템이 해당 주소에 모두 응답할 수 있도록 구성된 경우에만 가능합니다. 그러나 한 포트에 지정된 모든 IP 주소에서 수신할 수 있는 0.0.0.0 IP 주소를 사용하는 경우, 특정한 IP 주소와 동일한 포트에서 수신할 수 있는 추가 IP 주소를 HTTP Listener에 설정할 수 없습니다. 예를 들어, 0.0.0.0:80(포트 80에 지정된 모든 IP 주소)을 사용하는 HTTP Listener가 있는 경우 1.2.3.4:80을 사용하는 HTTP Listener는 만들 수 없습니다.

또한 각 HTTP Listener에는 요청에서 지정한 가상 서버에 연결되지 않을 때 해당 요청을 라우팅하는 서버인 기본 가상 서버가 있습니다.

또한 HTTP Listener의 억셉터 스레드(억셉트 스레드라고도 함) 수를 지정합니다. 억셉트 스레드는 연결을 기다리는 스레드입니다. 스레드는 연결을 승인하고 이 연결을 작업자 스레드에서 선택할 수 있도록 대기열에 둡니다. 억셉트 스레드는 충분히 있으면 새로운 요청이 있을 때 항상 사용할 수 있으나 시스템에 많은 부담을 줄 수 있습니다. 기본값은 1이며, 시스템의 CPU당 하나의 억셉트 스레드를 갖는 것이 좋습니다. 성능에 문제가 있는 경우 이 값을 조정할 수 있습니다.

또한 HTTP Listener에 대한 보안의 사용 여부와 사용 중인 보안의 종류(예: SSL 종류 및 암호화 종류)를 지정할 수 있습니다.

가상 서버

가상 서버를 만들려면, 먼저 원하는 가상 서버의 종류를 결정해야 합니다. 사용자는 IP 주소 기반 가상 서버 또는 URL 호스트 기반 가상 서버를 사용하고 있을 수 있습니다. 가상 서버를 만들려면 하나의 가상 서버 ID, 하나 이상의 HTTP Listener 및 하나 이상의 URL 호스트를 지정해야 합니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 가상 서버의 유형
- IP 주소 기반 가상 서버
- URL 호스트 기반 가상 서버
- 기본 가상 서버

가상 서버의 유형

모든 가상 서버에는 지정된 URL 호스트가 있습니다. 그러나 가상 서버를 HTTP Listener에 기반한 IP 주소에 연결할 수도 있습니다. 가상 서버의 HTTP Listener가 특정한 IP 주소에서 수신하는 경우, 이 가상 서버를 IP 주소 기반 가상 서버라고 합니다.

여러 가상 서버가 동일한 IP 주소에서 수신하는 경우, 해당 가상 서버는 URL 호스트에 의해 구별되므로 URL 호스트 기반 가상 서버라고 합니다.

새로운 요청이 있을 경우, 서버에서 IP 주소 또는 Host 헤더 값을 기반으로 하여 해당 요청을 전달할 서버의 종류를 결정합니다. 서버에서는 IP 주소를 먼저 평가합니다. 자세한 내용은 372 페이지의 "요청 프로세스에 대한 가상 서버 선택"을 참조하십시오.

IP 주소 기반 가상 서버

한 컴퓨터에서 여러 IP 주소를 사용하려면 운영 체제를 통해 주소를 매핑하거나 추가 카드를 제공해야 합니다. 운영 체제를 통해 여러 IP 주소를 설정하려면 네트워크 제어판(Windows의 경우) 또는 `ifconfig` 유틸리티(UNIX의 경우)를 사용하십시오. `ifconfig`의 사용 방법은 플랫폼에 따라 다양합니다. 자세한 내용은 운영 체제 설명서를 참조하십시오.

특정한 IP 주소에서 수신하는 HTTP Listener를 만들어 IP 주소 기반 가상 서버를 만듭니다. 그런 다음 가상 서버(HTTP Listener에 대한 기본 가상 서버)를 연결합니다. 가상 서버 배포 방법에 대한 자세한 내용은 385페이지의 "가상 서버 배포"를 참조하십시오.

URL 호스트 기반 가상 서버

가상 서버에 고유한 URL 호스트를 지정하여 URL 호스트 기반 가상 서버를 설정할 수 있습니다. 호스트 요청 헤더의 내용을 보면 올바른 가상 서버를 알 수 있습니다.

예를 들어, 고객 (*aaa*, *bbb* 및 *ccc*)에 대한 가상 서버를 설정하여 각 고객이 개별 도메인 이름을 가질 수 있도록 하려면, 먼저 DNS를 구성하여 각 고객의 URL (`www.aaa.com`, `www.bbb.com`, `www.ccc.com`)이 사용 중인 HTTP Listener의 IP 주소를 인식하도록 해야 합니다. 그런 다음 각 가상 서버에 대한 URL 호스트를 정확하게 설정합니다(예: `www.aaa.com`). `/etc/hosts` 파일의 IP 주소에 호스트를 매핑해야 합니다.

HTTP Listener에 연결된 이러한 URL 호스트 기반 가상 서버의 수에는 제한이 없습니다.

URL 호스트 기반 가상 서버에서는 호스트 요청 헤더를 사용하여 사용자가 정확한 페이지를 찾을 수 있도록 하므로, 일부 클라이언트 소프트웨어는 이 서버에서 작동하지 않습니다. HTTP Host 헤더를 지원하지 않는 이전의 클라이언트 소프트웨어는 작동하지 않습니다. 이러한 클라이언트는 HTTP Listener에 대한 기본 가상 서버를 수신합니다.

기본 가상 서버

URL 호스트 기반 가상 서버는 호스트 요청 헤더를 사용하여 선택됩니다. 최종 사용자의 브라우저에서 Host 헤더를 전송하지 않거나 서버에서 지정된 Host 헤더를 찾을 수 없는 경우, HTTP Listener에 대한 기본 가상 서버에서 요청을 처리합니다.

또한 IP 주소 기반 가상 서버의 경우, Sun ONE Application Server에서 지정된 IP 주소를 찾을 수 없을 경우 HTTP Listener의 기본 가상 서버에서 요청을 처리합니다. 오류 메시지를 전달하거나 특정한 문서 루트에서 서버 페이지를 전송하도록 기본 가상 서버를 구성할 수 있습니다.

주 HTTP Listener의 기본 가상 서버와 서버를 설치할 때 만들어지는 기본 가상 서버를 서로 혼동하지 마십시오. 기본 가상 서버는 기본 응용 프로그램 서버 인스턴스에 대한 가상 서버입니다. HTTP Listener의 기본 가상 서버는 사용자가 기본값으로 지정한 가상 서버입니다.

HTTP Listener를 만들 때 기본 가상 서버를 지정합니다. 언제든지 기본 가상 서버를 바꿀 수 있습니다.

obj.conf 파일

기본적으로 각 가상 서버에는 가상 서버 설정을 저장하는 별도의 `obj.conf` 파일이 있습니다. 관리 인터페이스 또는 명령줄 인터페이스를 통해 설정을 변경하면, 이 변경 사항은 가상 서버의 `obj.conf` 파일을 포함하여 구성 파일에 자동으로 적용됩니다. 모든 `obj.conf` 파일은 *instance_dir*/config 디렉토리에 있습니다. 이 설명서에서 언급하는 "obj.conf 파일"은 모든 `obj.conf` 파일 또는 설명 중인 가상 서버의 `obj.conf` 파일을 가리킵니다.

접두어가 없는 `obj.conf`라는 이름의 파일은 Sun ONE Application Server에서 각 가상 서버에 대한 `obj.conf` 파일을 만드는 데 사용하는 템플리트입니다. 이 파일은 편집해도 기존의 가상 서버에 영향을 주지 않지만, 이후에 만들어지는 가상 서버에는 영향을 줍니다. `obj.conf` 파일을 직접 편집하는 방법에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

기본적으로 각 활성 `obj.conf` 파일의 이름은 *virtual_server_name*`-obj.conf`입니다. 서버 인스턴스에 대한 기본 가상 서버는 인스턴스의 이름을 따서 이름이 지정되기 때문에 서버 인스턴스를 처음 만들 때 해당 `obj.conf` 파일의 이름은 *instance_name*`-obj.conf`가 됩니다. 이 파일 중 하나를 직접 편집하거나 관리 인터페이스를 통해 편집하면, 가상 서버의 구성이 변경됩니다.

요청 프로세스에 대한 가상 서버 선택

서버에서 요청을 처리하려면 HTTP Listener를 통해 요청을 수락한 다음, 올바른 가상 서버로 요청을 전송해야 합니다. 이 절에서는 가상 서버를 결정하는 방법에 대해 설명합니다.

- HTTP Listener가 기본 가상 서버에 대해서만 구성되는 경우, 해당 가상 서버가 선택됩니다.
- HTTP Listener에 둘 이상의 가상 서버가 구성되어 있는 경우, 요청 `Host` 헤더는 가상 서버의 `hosts` 속성과 일치합니다. `Host` 헤더가 없거나 `hosts` 속성이 일치하지 않은 경우, HTTP Listener의 기본 가상 서버가 선택됩니다.

SSL HTTP Listener에 가상 서버를 구성하는 경우 서버를 시작할 때 `hosts` 속성이 인증서의 주제 패턴과 일치하는지 검사합니다. 일치하지 않을 경우 경고가 발생되고 서버 로그에 기록됩니다.

가상 서버가 결정된 후 Sun ONE Application Server는 가상 서버의 `obj.conf` 파일을 실행합니다. 서버에서 `obj.conf`에서 실행할 지시문을 결정하는 방법에 대한 자세한 내용은 *Sun ONE Application Server Administratorís Configuration File Reference*를 참조하십시오.

문서 루트

문서 루트(기본 문서 디렉토리라고도 함)는 원격 클라이언트에서 사용 가능한 가상 서버의 모든 파일을 포함하는 중앙 디렉토리입니다.

문서 루트 디렉토리를 사용하면 가상 서버에서 파일에 대한 액세스를 간단한 방법으로 제한할 수 있습니다. 또한 URL에 지정된 경로가 기본 문서 디렉토리와 연관되어 있으므로, URL을 변경하지 않고 간편하게 문서를 다른 디스크의 새 디렉토리로 이동할 수도 있습니다.

예를 들어, 문서 디렉토리가 *install_dir*/docs인 경우 http://www.sun.com/products/info.html과 같은 요청은 *install_dir*/docs/info.html에서 파일을 찾으라고 서버에 지시합니다. 문서 루트를 변경하는 경우, 다시 말해 모든 파일과 하위 디렉토리를 이동시키는 경우, 모든 URL을 새 디렉토리에 매핑하거나 클라이언트에게 새 디렉토리에서 찾으라고 지시하는 대신에 가상 서버에서 사용하는 문서 루트를 변경하기만 하면 됩니다.

기본 Sun ONE Application Server 인스턴스(server1)에 대한 문서 루트는 server1 응용 프로그램 서버 인스턴스 내에서 만든 가상 서버의 문서 루트가 됩니다. 만든 각 가상 서버에 대한 해당 디렉토리를 무시할 수 있습니다.

가상 서버에서 Sun ONE Application Server 기능 사용

Sun ONE Application Server는 가상 서버에서 SSL 및 액세스 제어와 같은 많은 기능을 사용할 수 있습니다. 다음 여러 절에서는 기능에 대해 설명하고 기능에 대한 자세한 내용을 참조할 수 있는 위치에 대한 정보를 제공합니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 가상 서버에서 SSL 사용
- 액세스 로그 파일과 서버 로그 파일 사용
- 가상 서버에서 액세스 제어 사용
- 가상 서버에서 CGI 사용

가상 서버에서 SSL 사용

가상 서버에서 SSL를 사용하려면 일반적으로 IP 주소 기반 가상 서버를 사용합니다. 일반적인 포트는 443입니다. Sun ONE Application Server는 요청을 전송할 URL 호스트를 결정하기 전에 요청을 읽어야 하므로, URL 호스트 기반 가상 서버에서 SSL를 사용하기는 어렵습니다. 서버에서 요청을 읽은 경우 보안 정보가 교환되는 초기 핸드셰이크가 이미 발생한 상태입니다.

유일한 예외로는 URL 호스트 기반 가상 서버가 모두 "와일드 카드 인증서"를 사용하며, 동일한 서버 인증서를 포함한 동일한 SSL 구성을 가진 경우입니다. 자세한 내용은 *Sun ONE Application Server Administratorís Guide to Security*를 참조하십시오.

가상 서버에서 SSL를 구현하는 한 가지 방법은 두 개의 HTTP Listener를 구성하여 하나는 SSL를 사용하고 포트 443에서 수신하고 다른 하나는 SSL을 사용하지 않는 것입니다. 일반적으로 사용자는 비 SSL HTTP Listener를 통해 가상 서버에 액세스합니다. 보안 트랜잭션을 수행해야 하는 경우 사용자는 웹 페이지의 버튼을 눌러 보안 트랜잭션을 시작할 수 있습니다. 그러면 보안 HTTP Listener를 통해 요청을 수행할 수 있습니다.

SSL 트랜잭션은 비 SSL 트랜잭션보다 훨씬 느리기 때문에 이러한 설계는 SSL 트랜잭션이 필요한 경우에만 사용하는 것이 좋습니다. 그 밖의 경우에는 속도가 빠른 비 SSL 연결을 사용합니다.

가상 서버에서 Sun ONE Application Server를 통해 보안을 설정 및 사용하는 방법에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Guide to Security*를 참조하십시오. 가상 서버에서의 샘플 SSL 구성에 대한 다이어그램은 387페이지의 "예 2: 보안 서버"를 참조하십시오.

액세스 로그 파일과 서버 로그 파일 사용

액세스 로그 파일은 가상 서버에 대한 HTTP 액세스가 기록되는 파일입니다. 가상 서버를 새로 만들면, 기본적으로 액세스 로그 파일은 응용 프로그램 서버 인스턴스와 동일한 로그 파일이 됩니다. 일반적으로 각 가상 서버에 고유한 로그 파일을 둘 수 있습니다. 이렇게 설정하려면 각 가상 서버에 대한 로그 경로를 변경할 수 있습니다. 모든 가상 서버 액세스를 동일한 액세스로그 파일에 기록하려면, 가상 서버 ID가 로그 파일에 포함되도록 서버 인스턴스에 대한 로깅 설정을 변경할 수 있습니다. 응용 프로그램 서버 인스턴스의 로깅 변경에 대한 자세한 내용은 5장, "로깅 사용"을 참조하십시오.

서버 로그 파일은 정보 메시지와 오류가 기록되는 파일입니다. 가상 서버를 새로 만들면, 기본적으로 해당 로그 파일은 응용 프로그램 서버 인스턴스에 대한 로그 파일과 동일합니다. 각 가상 서버에 대한 로그 파일은 변경할 수 있습니다.

가상 서버에서 액세스 제어 사용

가상 서버별로 액세스 제어를 설정할 수 있습니다. 또한 각 가상 서버에서 LDAP 데이터베이스를 사용하여 사용자 및 그룹 인증을 할 수 있도록 액세스 제어를 구성할 수도 있습니다. 자세한 내용은 *Sun ONE Application Server Administratorís Guide to Security*를 참조하십시오.

가상 서버에서 CGI 사용

가상 서버에서 CGI를 사용할 수 있습니다. 각 가상 서버에서 CGI를 저장하는 디렉토리를 설정하고 CGI에 대한 파일 유형을 설정해야 합니다. CGI에 대한 자세한 내용은 *Sun ONE Application Server Developerís Guide to Web Applications*를 참조하십시오.

HTTP Listener 만들기 및 구성

서버에서 요청을 처리하려면 HTTP Listener를 통해 요청을 수락한 다음 올바른 가상 서버에 요청을 전송해야 합니다. 서버 인스턴스가 생성될 때(설치 중 또는 설치 이후) 하나의 HTTP Listener (`http-listener-1`)가 자동으로 만들어집니다. 이 HTTP Listener는 IP주소 0.0.0.0과 응용 프로그램 서버 포트 번호로 지정한 포트 번호를 사용합니다. 기본 HTTP Listener는 삭제할 수 없습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- HTTP Listener 만들기
- HTTP Listener 설정 편집
- HTTP Listener 삭제

HTTP Listener 만들기

관리 인터페이스를 사용하여 HTTP Listener를 만들려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. HTTP Listener를 누릅니다.
3. "새로 만들기"를 누릅니다.

4. 필드에 입력합니다.

   HTTP Listener의 포트 번호와 IP 주소의 조합은 고유해야 합니다. IPV4 또는 IPV6 주소를 사용할 수 있습니다. IP 주소 기반 가상 서버에 대한 HTTP Listener를 만들려면 HTTP Listener에 대해 특정한 IP 주소를 지정하십시오.

   "서버 이름 반환" 필드는 서버에서 클라이언트에 전송하는 URL의 호스트 이름을 지정합니다. 이는 서버에서 자동으로 생성하는 URL에 영향을 주지만, 서버에 저장된 디렉토리 및 파일의 URL에는 영향을 주지 않습니다. 서버에서 별칭을 사용하는 경우, 이 이름도 별칭이어야 합니다.

   기본 가상 서버는 다른 가상 서버가 먼저 발견되지 않는 경우 HTTP Listener에 대한 요청에 응답하는 가상 서버입니다. 자세한 내용은 372페이지의 “요청 프로세스에 대한 가상 서버 선택”을 참조하십시오.

   요청을 수락하려면 HTTP Listener를 활성화해야 합니다.

   또한 이 HTTP Listener에 대한 보안을 활성화하고 고급 등록 정보를 구성할 수 있습니다. IPV6를 지정하려면 "패밀리" 필드에서 값 inet6을 사용하십시오. 이 값이 inet6이면 IPv4 주소의 접두어는 서버 로그에서 ::ffff:가 됩니다.

5. "확인"을 누릅니다.

HTTP Listener를 만들 때 "기본 가상 서버" 필드에 기존 가상 서버를 입력해야 합니다. 서버 인스턴스를 통해 만든 가상 서버를 사용할 수 있습니다. 그런 다음 필요한 경우 추가 가상 서버를 만든 후 되돌아가서 가상 서버를 변경할 수 있습니다.

명령줄 인터페이스를 사용하여 HTTP Listener를 만들려면 `asadmin` 유틸리티의 `create-http-listener` 명령을 사용합니다. 만든 모든 HTTP Listener의 목록을 작성하려면 `list-http-listeners` 명령을 사용합니다.

HTTP Listener를 만들려면, 다음 구문을 사용합니다.

`asadmin create-http-listener --user` *username* `[--password` *password*`] [--host` *hostname*`] [--port` *adminport*`] [--secure | -s] [--passwordfile` *file_name*`] --address` *address* `[--instance` *instancename*`] --listenerport` *listener_port* `--defaultvs` *virtual_server* `--servername` *server_name* `[--family` *family*`] [--acceptorthreads` *acceptor_threads*`] [--blockingenabled` *blocking_enabled*`] [--securityenabled` *security_enabled*`] [--enabled` *enabled*`]` *listener_id*

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, “명령줄 인터페이스 사용”을 참조하십시오.

HTTP Listener 설정 편집

관리 인터페이스를 사용하여 HTTP Listener 설정을 편집하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "HTTP Listeners"를 엽니다.
3. 편집할 HTTP Listener를 누릅니다.
4. 원하는 대로 변경하고 "저장"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

또한 명령줄 인터페이스에서 `asadmin` 유틸리티를 사용하여 HTTP Listener를 편집할 수 있습니다. `get` 명령을 사용하여 현재 설정을 가져와서 `set` 명령을 사용하여 현재 설정에 새 값을 지정하십시오.

HTTP Listener의 모든 속성값을 표시하려면 다음 작업을 수행합니다.

`asadmin> get` *server_instance*`.http-listener.`*http_listener_name*`.*`

예를 들어, 기본 HTTP Listener에 대한 값을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.http-listener.http-listener-1.*
```

속성값을 설정하려면 다음 작업을 수행합니다.

`asadmin> set` *server_instance*`.http-listener.`*http_listener_name*`.`*attribute_name*`=`*value*

예를 들어, `defaultVirtualServer` 속성을 `http-listener-1`에 대한 `server2`에 설정하려면 다음 작업을 수행합니다.

```
asadmin> set
server1.http-listener.http-listener-1.defaultVirtualServer=server2
```

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, “명령줄 인터페이스 사용”을 참조하십시오.

HTTP Listener 삭제

관리 인터페이스를 사용하여 HTTP를 삭제하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "HTTP Listeners"를 누릅니다..
3. 삭제하려는 HTTP Listener의 옆에 있는 확인란을 누릅니다.
4. "삭제"를 누릅니다.

명령줄 인터페이스를 사용하여 HTTP Listener를 삭제하려면 `asadmin` 유틸리티의 `delete-http-listener` 명령을 사용합니다. 이 때 사용되는 구문은 다음과 같습니다.

`asadmin delete-http-listener ---user` *username* `[--password` *password*`] [--host` *hostname*`] [--port` *adminport*`] [--secure | -s] [--passwordfile` *file_name*`] --instance` *instance httplistener_id*

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

가상 서버 만들기 및 구성

HTTP Listener를 설정하고 나면 가상 서버를 만들어 사용할 수 있습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 가상 서버 만들기
- 가상 서버 설정 편집
- 가상 서버 삭제

가상 서버 만들기

관리 인터페이스를 사용하여 가상 서버를 만들려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 누릅니다.
3. "새로 만들기"를 누릅니다.
4. 필수 필드와 선택 필드에 입력합니다.
5. "저장"을 누릅니다.

명령줄 인터페이스를 사용하여 가상 서버를 만들려면 `asadmin` 유틸리티의 `create-virtual-server` 명령을 사용합니다. 이 때 사용되는 구문은 다음과 같습니다.

`asadmin create-virtual-server --user` *username* `---user` *username* `[--password` *password*`] [--host` *hostname*`] [--port` *adminport*`] [--secure | -s] [--passwordfile` *file_name*`] [--instance` *instancename*`] --hosts` *hosts* `--mime` *mime_types_file* `[--httplisteners` *http-listeners*`] [--defaultwebmodule` *default_web_module*`] [--configfile` *config_file*`] [--defaultobj` *default_object*`] [--state` *state*`] [--acls` *acls*`] [--acceptlang` *accept_language*`] [--logfile` *logfile*`] [--property (`*name*`=`*value*`)[:`*name*`=`*value*`]*]` *virtual_server_id*

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

가상 서버를 만들 때 다음 유형의 설정을 입력할 수 있습니다.

- 필수 설정
- 선택적 일반 설정
- 웹 응용 프로그램 설정
- CGI 설정
- HTTP 서비스 품질(QOS) 설정

필수 설정

가상 서버의 필수 설정에는 이름(ID)과 URL 호스트가 포함됩니다.

또한 MIME 유형 파일을 지정해야 합니다. MIME 유형 파일의 경우 파일 유형에 파일 확장자를 매핑해야 됩니다. 예를 들어, MIME 유형 파일의 경우 .cgi로 끝나는 모든 파일은 CGI 파일로 처리되도록 지정할 수 있습니다.

각 가상 서버에 개별 MIME 유형 파일을 만들 필요는 없습니다. 그 대신 MIME 유형 파일을 필요한 수만큼 만들어 가상 서버와 연결합니다. 기본 MIME 유형 파일은 `mime1`이며, 파일 이름은 `mime.types`입니다.

MIME 유형 파일에 대한 자세한 내용은 364페이지의 "MIME 유형 구성"을 참조하십시오.

선택적 일반 설정

필수 필드뿐만 아니라 선택 필드도 설정할 수 있습니다.

HTTP Listener

HTTP Listener는 가상 서버에 대한 연결을 처리합니다. 원격 클라이언트에서 가상 서버에 액세스할 수 있도록 HTTP Listener 하나를 지정해야 합니다.

ACL

가상 서버에 적용된 액세스 제어 목록 (ACL)입니다. 자세한 내용은 *Sun ONE Application Server Administratorís Guide to Security*를 참조하십시오.

사용 가능한 언어 헤더

클라이언트에서 HTTP 1.1을 사용하여 서버에 연결하면 클라이언트에서 사용 가능한 언어를 설명하는 헤더 정보를 전송할 수 있습니다. 또한 이 언어 정보를 구문 분석할 수 있도록 서버를 구성할 수 있습니다.

예를 들어, 일본어와 영어로 작성한 문서를 저장하는 경우 사용 가능한 언어 헤더를 구문 분석하도록 선택할 수 있습니다. 사용 가능한 언어 헤더가 일본어인 클라이언트에서 서버에 연결하면 해당 페이지의 일본어 버전이 표시됩니다. 사용 가능한 언어 헤더가 영어인 클라이언트에서 서버에 연결하면 영어 버전이 표시됩니다.

여러 언어를 지원하지 않는 경우에는 사용 가능한 언어 헤더를 구문 분석해서는 안 됩니다.

상태

이 상태는 가상 서버의 상태를 말하며, 응용 프로그램 인스턴스의 설정 또는 해제와는 상관이 없습니다. 이 페이지에 표시된 가상 서버의 상태가 설정이면, 응용 프로그램 서버 인스턴스가 설정인 경우에만 가상 서버에서 요청을 수락합니다.

이는 기본 응용 프로그램 서버 인스턴스에 대한 기본 가상 서버의 경우에도 마찬가지입니다. 따라서 응용 프로그램 인스턴스를 비활성화하면 기본 가상 서버가 설정 상태이더라도 연결을 수락하지 않습니다.

유효한 상태로는 설정, 해제 또는 사용 불가능이 있습니다. 가상 서버가 설정 상태이면 연결을 수락할 수 있습니다.

응용 프로그램 서버 인스턴스에 대한 기본 가상 서버는 해제 또는 사용 불가능 상태로 설정할 수 없습니다.

로그 파일

로그 파일(서버 로그 파일이라고도 함)은 정보 메시지와 오류가 기록되는 파일입니다. 액세스 로그 파일은 가상 서버에 대한 HTTP 액세스가 기록되는 파일입니다.

문서 루트

문서 루트(기본 문서 디렉토리라고도 함)는 원격 클라이언트에서 사용 가능한 가상 서버의 모든 파일을 포함하는 중앙 디렉토리입니다. 자세한 내용은 372페이지의 “문서 루트”를 참조하십시오.

웹 응용 프로그램 설정

웹 응용 프로그램은 서블릿, JavaServer 페이지, HTML 문서 및 기타 웹 자원(이미지 파일, 압축 아카이브 및 기타 데이터 포함)의 모음입니다. 웹 응용 프로그램은 아카이브(WAR 파일)에 패키지화하거나 열린 디렉토리 구조에 있을 수 있습니다.

Sun ONE Application Server 7은 Servlet 2.3 API 사양을 지원하므로 서블릿과 JSP가 웹 응응 프로그램에 포함될 수 있습니다. 또한 Sun ONE Application Server 7은 비 J2EE 응용 프로그램 구성 요소인 SHTML과 CGI를 지원합니다.

가상 서버를 만들 때는 가상 서버에 대한 기본 웹 모듈을 지정합니다. 기본 웹 모듈은 가상 서버에 배포되는 다른 웹 모듈에서 확인할 수 없는 모든 요청에 응답합니다. 기본 웹 모듈을 지정하지 않으면 컨텍스트 루트가 비어 있는 웹 모듈이 사용됩니다. 컨텍스트 루트가 비어 있는 웹 모듈이 없을 때는 시스템 기본 웹 모듈이 생성되어 사용됩니다.

웹 응용 프로그램을 배포할 때는 가상 서버를 지정합니다. 웹 응용 프로그램을 배포하면 가상 서버에 대한 기본 웹 모듈로 선택할 수 있도록 사용 가능한 웹 모듈 목록에 웹 응용 프로그램이 나타납니다. 가상 서버에 대한 기본 웹 모듈로서 웹 모듈을 지정하는 경우, 해당 가상 서버는 자동으로 가상 서버의 웹 응용 프로그램 목록에 추가됩니다.

CGI 설정

가상 서버를 만들 때 설정하는 CGI 설정은 CGI 프로그램을 실행하는 사용자 및 그룹을 제어하며 CGI가 실행되기 전 변경할 디렉토리(chroot) 및 chroot 다음에 변경할 디렉토리를 제어합니다.

UNIX의 경우, 서버와 관련해 CGI 프로그램의 우선 순위를 결정하는 증분값인 우선 순위(nice)를 설정할 수 있습니다. 일반적으로 서버는 우선 순위 값 0으로 실행되며, 우선 순위(nice) 증분값은 0 (CGI 프로그램이 실행되는 우선 순위가 서버와 같음)과 19 (CGI 프로그램이 실행되는 우선 순위가 서버보다 훨씬 낮음) 사이입니다.

HTTP 서비스 품질(QOS) 설정

서비스 품질(QOS)은 가상 서버에 설정한 성능 제한을 말합니다. 예를 들어, ISP는 허용되는 대역폭의 범위에 따라 가상 서버에 대한 요금을 서로 다르게 부과할 수 있습니다. 지정된 대역폭과 최대 연결 수만 허용되도록 이 설정을 적용되거나 적용하지 않을 수 있습니다. 설정을 적용하지 않으면 메시지는 제한이 초과될 때 로그 파일에 기록됩니다. 자세한 내용은 151페이지의 “CLI를 사용한 트랜잭션 서비스 관리”를 참조하십시오.

관리 인터페이스를 통해 이 설정을 바꿀 수 있을 뿐만 아니라 명령줄 인터페이스의 `asadmin` 유틸리티를 사용할 수도 있습니다. 명령줄 인터페이스의 `asadmin` 유틸리티를 사용하여 서비스 품질(QOS)을 구성하려면 다음 명령을 사용합니다.

- `create-http-qos`
- `delete-http-qos`

이 명령은 다음 구문을 사용합니다:

`asadmin create-http-qos --user` *username* `[--password` *password*`] [--host` *hostname*`] [--port` *adminport*`] [--secure | -s] [--passwordfile` *file_name*`] [--virtualserver` *virtual_server_id*`] [--bwlimit` *bandwidth_limit*`] [--enforcebwlimit` *enforce_bandwidth_limit*`] [--connlimit` *connection_limit*`] [--enforceconnlimit` *enforce_connection_limit*`]` *instance_name*

`asadmin delete-http-qos --user` *username* `[--password` *password*`] [--host` *hostname*`] [--port` *adminport*`] [--secure | -s] [--passwordfile` *file_name*`] [--virtualserver` *virtual_server_id*`]` *instance_name*

가상 서버를 지정하는 경우, 이 명령은 해당 가상 서버에 대한 서비스 품질(QOS) 정보를 만들거나 삭제합니다. 가상 서버를 지정하지 않으면 이 명령은 서버 인스턴스에 영향을 줍니다.

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, "명령줄 인터페이스 사용"을 참조하십시오.

가상 서버 설정 편집

가상 서버를 설정하고 나면 편집이 가능합니다. 가상 서버 설정 편집에 대한 자세한 내용은 다음을 참조하십시오.

- 관리 인터페이스를 사용하여 일반 설정 편집
- 명령줄 인터페이스를 사용하여 일반 설정 편집
- CGI 설정 편집
- 문서 처리 설정, 문서 디렉토리 설정 및 HTTP/HTML 설정 편집

관리 인터페이스를 사용하여 일반 설정 편집

가상 서버의 일반 설정은 가상 서버를 만들 때 설정할 수 있습니다. 이 설정을 변경하려면 다음 단계를 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 "가상 서버"를 누릅니다.
4. 원하는 대로 변경합니다.

   변경할 수 있는 영역으로는 서비스 품질(QOS) 설정, ACL 추가, 문서 루트 및 사용 가능한 언어 헤더와 같은 내용 관련 설정, 사용자, 그룹, 우선 순위(nice) 및 chroot 설정과 같은 CGI 설정, 기본 웹 모듈 등이 있습니다.
5. "저장"을 누릅니다.

이 설정 중 일부에 대한 자세한 내용은 378페이지의 “가상 서버 만들기 및 구성”을 참조하십시오. 또한 온라인 도움말도 참조하십시오.

명령줄 인터페이스를 사용하여 일반 설정 편집

명령줄 인터페이스에서 `asadmin` 유틸리티를 사용하여 이 설정을 편집할 수도 있습니다. `get` 명령을 사용하여 현재 설정을 표시하고 `set` 명령을 사용하여 현재 설정에 새 값을 설정하십시오.

가상 서버의 모든 속성을 표시하려면 다음 구문을 사용합니다.

```
asadmin> get instance_name.virtual-server.vserver_id.*
```

예를 들면 다음과 같습니다.

```
asadmin> get server1.virtual-server.vs1.*
```

응용 프로그램 서버 인스턴스 `server1`에 대한 모든 속성을 표시하려면 다음 구문을 사용합니다.

```
asadmin> get server1.virtual-server.server1.*
```

예를 들어, 사용 가능한 언어 헤더 속성을 설정하려면 다음 구문을 사용합니다.

```
asadmin> set
server1.virtual-server.server1.virtualserver.acceptLanguage=false
```

주 명령줄 인터페이스를 사용하여 "일반" 페이지에 있는 모든 필드 값을 설정할 수 있습니다. 그러나 다른 탭의 페이지, 예를 들어 "CGI" 탭 페이지의 필드 값을 설정하기 위해 명령줄 인터페이스를 사용할 수는 없습니다.

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin` 사용에 대한 자세한 내용은 부록 A, “명령줄 인터페이스 사용”을 참조하십시오.

CGI 설정 편집

CGI 편집에 대한 자세한 내용은 *Sun ONE Application Server Developers Guide to Web Applications*를 참조하십시오.

문서 처리 설정, 문서 디렉토리 설정 및 HTTP/HTML 설정 편집

이러한 설정 변경에 대한 자세한 내용은 16장, “가상 서버 내용 관리”를 참조하십시오.

가상 서버 삭제

가상 서버를 삭제하려면 다음 작업을 수행합니다.

1. 관리 인터페이스의 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 누릅니다.
3. 삭제할 가상 서버 옆에 있는 확인란을 누릅니다.
4. "삭제"를 누릅니다.

   관리 인터페이스를 사용하여 일부 가상 서버를 삭제할 수 없습니다.

명령줄 인터페이스를 사용하여 가상 서버를 삭제하려면 `asadmin` 유틸리티의 `delete-virtual-server` 명령을 사용합니다.

사용법은 다음과 같습니다.

`asadmin delete-virtual-server --user` *username* `[--password` *password*`]` `[--host` *hostname*`]` `[--port` *adminport*`]` `[--secure | -s]` `[--passwordfile` *file_name*`]` `--instance` *instance virtualserver_id*

명령 구문에 대한 자세한 내용은 명령줄 인터페이스 도움말을 참조하십시오. `asadmin`에 대한 자세한 내용은 부록 A, “명령줄 인터페이스 사용”을 참조하십시오.

가상 서버 배포

Sun ONE Application Server의 가상 서버 구조는 매우 유연합니다. 응용 프로그램 인스턴스의 보안 또는 비보안 HTTP Listener의 수에는 제한이 없습니다. 이러한 HTTP Listener에 연결할 수 있는 가상 서버의 수에는 제한이 없습니다. 또한 IP 주소 기반 및 URL 호스트 기반 가상 서버를 모두 가질 수 있습니다.

모든 가상 서버는 고유한 ACL 목록, `mime.types` 파일 및 고유한 Java 웹 응용 프로그램 집합을 가질 수 있으나 반드시 가질 필요는 없습니다.

이러한 설계로 인해 사용자는 다양한 응용 프로그램에 대해 서버를 구성할 수 있는 최대한의 유연성을 가질 수 있습니다. 다음은 Sun ONE Application Server에 사용 가능한 구성의 일부 예입니다.

- 예 1: 기본 구성
- 예 2: 보안 서버
- 예 3: 인트라넷 호스팅
- 예 4: 다중 호스팅

예 1: 기본 구성

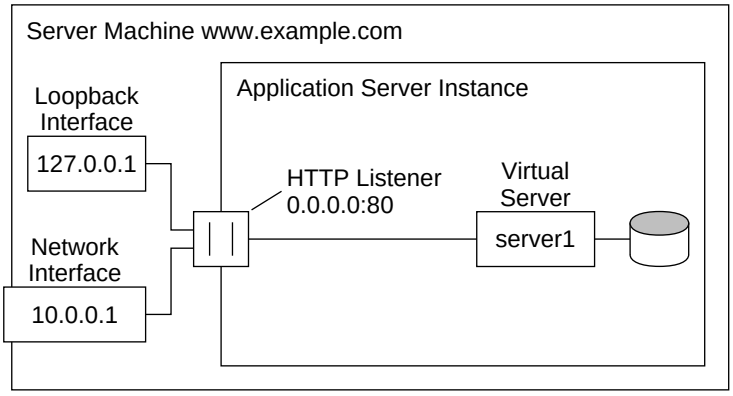
기본 구성에는 응용 프로그램 서버 인스턴스가 하나 있습니다. 이 응용 프로그램 서버 인스턴스에는 포트 80, 1024 또는 컴퓨터에 구성된 IP 주소에서 사용자가 선택한 포트를 수신하는 HTTP Listener가 하나만 있습니다.

로컬 네트워크의 일부 메커니즘에서는 컴퓨터에 구성된 각 주소에 대해 이름과 주소가 매핑합니다. 다음 예에서 컴퓨터는 두 개의 네트워크 인터페이스를 갖습니다. 하나는 주소 127.0.0.1의 루프백 인터페이스(네트워크 카드가 없어도 존재하는 인터페이스)이고 다른 하나는 주소 10.0.0.1의 이더넷 인터페이스입니다.

example.com이 DNS를 통해 10.0.0.1에 매핑합니다. HTTP Listener는 해당 시스템에 구성된 한 주소의 포트 80에서 수신하도록 구성됩니다("0.0.0.0:80").

기본 구성에 IP 주소 기반 가상 서버가 없으므로 HTTP Listener만이 기본 구성입니다. 모든 연결은 가상 서버 server1을 통합니다.

기본 구성



DNS

www.example.com	10.0.0.1

이 구성에서 가상 서버 VS1에 의해 다음 항목과 서버가 연결됩니다.

- http://127.0.0.1/ (example.com에서 시작)
- http://localhost/ (example.com에서 시작)
- http://example.com/
- http://10.0.0.1/

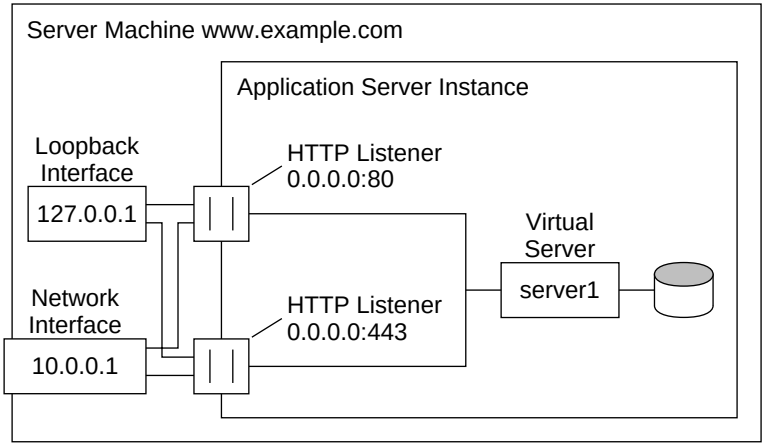
전통적인 HTTP 서버 사용 시 이 구성을 사용합니다. 추가 가상 서버나 HTTP Listener가 필요 없습니다. server1에 대한 설정을 변경하여 서버 설정을 구성하십시오.

예 2: 보안 서버

기본 구성에 SSL을 사용하려면 HTTP Listener를 보안 모드로 간단히 변경하면 됩니다.

또한 0.0.0.0:443으로 구성된 새 보안 HTTP Listener를 추가하고 server1을 새 HTTP Listener에 연결할 수도 있습니다. 이제 가상 서버에서는 HTTP Listener를 갖게 되는데, 하나는 보안 HTTP Listener를 사용하며 다른 하나는 그렇지 않습니다. 이제 서버에서는 SSL이 있거나 없는 상태로 동일한 내용을 제공합니다. 즉, http://example.com/과 https://example.com/은 동일한 내용을 제공합니다.

보안 서버



DNS

www.example.com	10.0.0.1

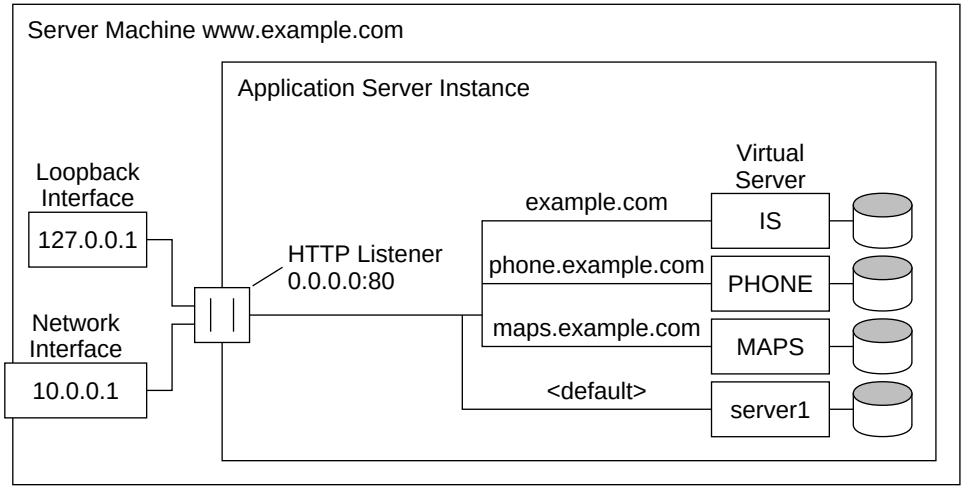
HTTP Listener에는 SSL 매개 변수가 설정되어 있습니다.

예 3: 인트라넷 호스팅

Sun ONE Application Server의 좀 더 복잡한 구성은 인트라넷 배포를 위해 서버에서 여러 가상 서버를 호스트하는 것입니다. 예를 들어, 직원이 다른 사용자의 전화 번호를 찾을 수 있으며, 캠퍼스의 지도를 볼 수 있고 정보 서비스 부서에 요청한 일의 처리 상태를 추적할 수 있는 세 가지 내부 사이트가 있습니다. 이 예에서 이전에는 phone.example.com, maps.example.com 및 is.example.com이라는 이름이 매핑되는 세 개의 컴퓨터에서 이 세 사이트를 호스트했습니다.

그러나 하드웨어 및 관리 오버헤드를 최소화하기 위해 이 세 사이트를 machine example.com에서 하나의 응용 프로그램 서버로 통합할 수 있습니다. 이 때 URL 호스트 기반 가상 서버 또는 IP 주소 기반 가상 서버를 사용하여 이를 설정할 수 있습니다. 이 두 방법에는 각각 장단점이 있습니다.

URL 호스트 기반 가상 서버를 사용하여 인트라넷 호스팅



DNS

www.example.com	10.0.0.1
is.example.com	10.0.0.1
phone.example.com	10.0.0.1
maps.example.com	10.0.0.1

URL 호스트 기반 가상 서버는 쉽게 설정할 수 있지만 다음과 같은 단점이 있습니다.

- 이 구성에서 SSL을 지원하려면, 와일드카드 인증서를 사용하는 비표준 설정이 필요합니다. 자세한 내용은 *Sun ONE Application Server Administratorís Guide to Security*를 참조하십시오.
- URL 호스트 기반 가상 서버는 레거시 HTTP 클라이언트와 함께 작동하지 않습니다.

IP 주소 기반 가상 서버는 다음과 같은 장점이 있습니다.

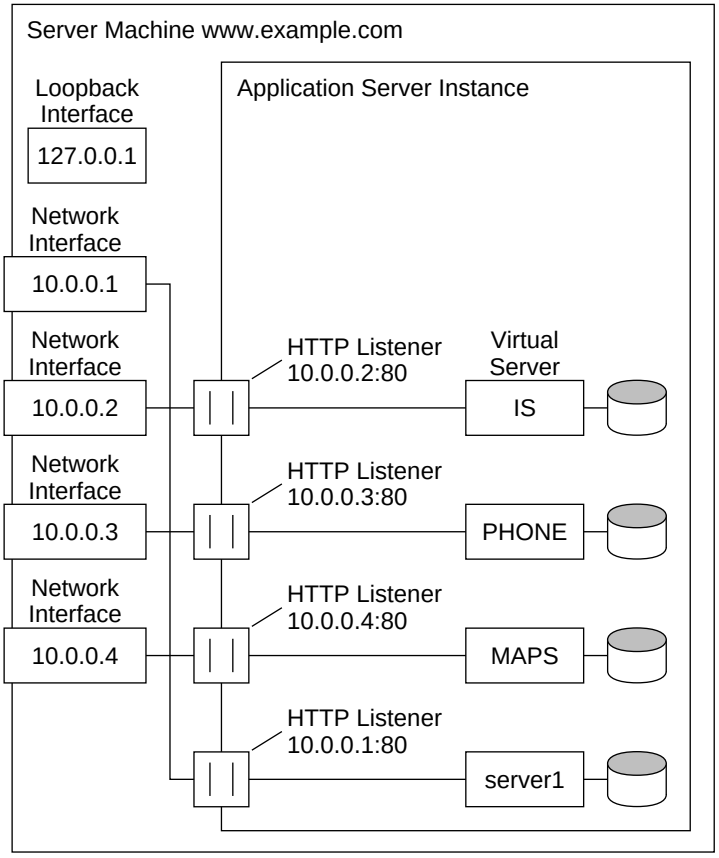
- HTTP/1.1 Host 헤더를 지원하지 않는 이전 클라이언트와 함께 작동합니다.
- SSL 지원이 간편합니다.

단점은 다음과 같습니다.

- 호스트 컴퓨터에서 구성 변경이 필요합니다(실제 또는 가상 서버 인터페이스의 구성).
- 수천 개의 가상 서버에서 구성을 조절할 수 없습니다.

두 가지 구성 방법 모두 세 이름에 대해 이름과 주소가 매핑하도록 설정해야 합니다. IP 주소 기반 구성에서 각 이름은 서로 다른 주소에 매핑됩니다. 이러한 모든 주소에서 연결을 수신하도록 호스트 시스템을 설정해야 합니다. URL 호스트 기반 구성에서 모든 이름은 같은 주소 즉, 원래 시스템에 지정된 주소로 매핑됩니다.

IP 주소 기반 가상 서버를 사용하여 인트라넷 호스팅



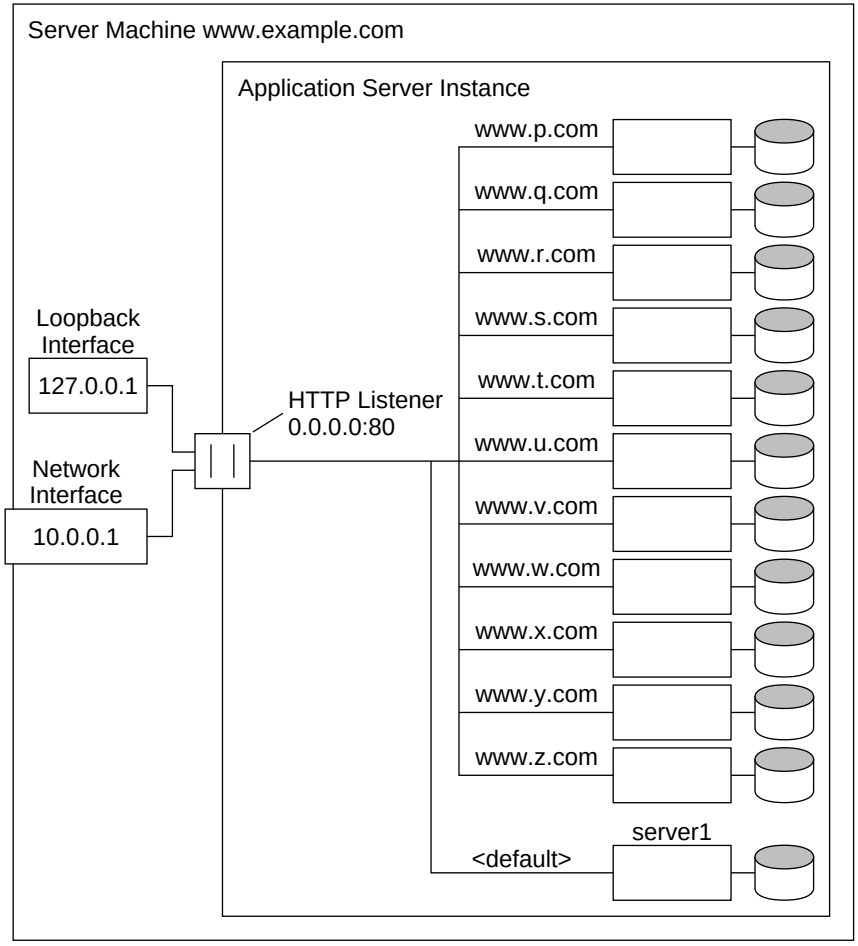
DNS

www.example.com	10.0.0.1
is.example.com	10.0.0.2
phone.example.com	10.0.0.3
maps.example.com	10.0.0.4

예 4: 다중 호스팅

다중 호스팅은 트래픽이 적은 수많은 가상 서버를 활성화하는 구성입니다. 트래픽이 적은 수많은 개인 홈 페이지를 호스트하는 ISP를 예로 들 수 있습니다. 일반적으로 가상 서버는 URL 호스트 기반입니다.

다중 호스팅



DNS

www.example.com	10.0.0.1
www.p.com	10.0.0.1
www.q.com	10.0.0.1
www.r.com	10.0.0.1
. . .	
www.z.com	10.0.0.1

기본 가상 서버인 `server1`은 계속 유지됩니다.

16장

가상 서버 내용 관리

이 장에서는 가상 서버에서 서비스하는 파일을 구성하고 관리하는 방법에 대해 설명합니다.

이 장에서는 다음 내용을 설명합니다.

- 문서 루트 변경
- 추가 문서 디렉토리 설정
- 원격 파일 조작 활성화
- htaccess 사용
- 심볼릭 링크 제한(UNIX의 경우)
- 사용자 공개 정보 디렉토리 사용자 정의(UNIX의 경우)
- 문서 환경 설정
- 오류 응답 사용자 정의
- 국가별 문자 집합 변경
- 문서 바닥글 설정
- URL 전달 구성
- 서버 구문 분석 HTML 설정
- 캐시 제어 지시문 설정
- 강력한 암호화 사용

문서 루트 변경

문서 루트는 원격 클라이언트가 사용할 수 있는 모든 파일을 저장하는 중앙 디렉토리입니다.

가상 서버를 추가할 경우 절대 경로로 문서 루트를 지정해야 합니다. 문서 루트에 대한 내용과 사용 방법에 대해서는 372페이지의 “문서 루트”를 참조하십시오.

관리 인터페이스를 사용하여 문서 루트에 다른 경로를 사용하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "일반" 탭을 누릅니다.
5. "문서 루트" 필드에 절대 디렉토리 경로를 입력합니다.

   이 디렉토리는 수동으로 만들어야 합니다.
6. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

주 일반적으로 각 가상 서버에는 고유 문서 루트가 있습니다.

추가 문서 디렉토리 설정

대부분의 경우 가상 인스턴트 또는 서버 인스턴스의 문서는 문서 루트에 있습니다. 하지만 경우에 따라 문서 루트가 아닌 디렉토리에서 문서를 서비스해야 할 수 있습니다. 이 경우 추가 문서 디렉토리를 설정하면 됩니다. 문서 루트가 아닌 문서 디렉토리에서 서비스를 제공하면 사용자에게 기본 문서 루트에 대한 액세스 권한을 부여하지 않고도 해당 사용자가 문서 그룹을 관리할 수 있습니다.

관리 인터페이스를 사용하여 추가 문서 디렉토리를 추가하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.

4. "Doc 디렉토리" 탭을 누릅니다.
5. "추가 Doc 디렉토리"를 누릅니다.
6. 매핑할 URL 접두어를 선택합니다.

   클라이언트는 문서가 필요할 경우 서버에 이 URL을 전송합니다.
7. 이 URL을 매핑할 대상 디렉토리를 지정합니다.
8. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

추가 문서 디렉토리에 대한 액세스를 제한하여 사용자가 기록할 수 없도록 해야 합니다.

원격 파일 조작 활성화

원격 파일 조작을 활성화하면 클라이언트는 파일 업로드, 파일 삭제, 디렉토리 작성, 디렉토리 제거, 디렉토리 내용 보기, 서버의 파일 이름 바꾸기를 할 수 있습니다. 가상 서버의 구성 파일인 `obj.conf`에는 원격 파일 조작을 활성화하면 실행되는 명령이 들어 있습니다. 이러한 명령이 활성화되면 원격 브라우저가 서버의 문서를 변경할 수 있습니다. 액세스 제어를 사용하여 이러한 자원에 대한 쓰기 액세스를 방지하여 인증되지 않은 액세스를 막아야 합니다.

원격 파일 조작을 활성화하더라도 Microsoft Frontpage 등과 같은 사용 중인 내용 관리 시스템에는 영향이 없습니다.

UNIX: 파일에 대한 올바른 권한이 있어야 하며 그렇지 않은 경우 이 기능이 작동하지 않습니다. 즉, 문서 루트 사용자가 서버 사용자와 같아야 합니다.

관리 인터페이스를 사용하여 원격 파일 조작을 활성화하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "Doc 디렉토리" 탭을 누릅니다.
5. "원격 파일 조작"을 누릅니다.
6. 자원 선택기에서 "전체 서버"를 선택하여 변경 내용을 전체 가상 서버에 적용하거나 가상 서버 내의 특정 디렉토리를 찾습니다.

7. "원격 파일 조작 활성화"를 선택합니다.
8. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

htaccess 사용

`htaccess` 파일은 구성 옵션 일부가 저장되어 있는 동적 구성 파일입니다. `htaccess` 파일은 Sun ONE Application Server 표준 액세스 제어와 결합하여 사용할 수 있습니다(표준 액세스 제어는 `htaccess` 액세스 제어보다 먼저 적용됨).

htaccess 사용에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Guide to Security*를 참조하십시오.

심볼릭 링크 제한(UNIX의 경우)

서버에서 파일 시스템 링크의 사용을 제한할 수 있습니다. 파일 시스템 링크는 다른 디렉토리나 파일 시스템에 저장된 파일에 대한 참조입니다. 이 참조를 통해 원격 파일을 현재 디렉토리에 있는 것처럼 액세스할 수 있습니다. 파일 시스템 링크에는 두 가지 유형이 있습니다.

- 하드 링크—하드 링크는 실제로 데이터 블록의 같은 세트를 가리키는 두 개의 파일 이름입니다. 즉, 원본 파일과 링크가 동일합니다. 따라서 하드 링크는 서로 다른 파일 시스템에 있을 수 없습니다.
- 심볼릭(소프트) 링크—심볼릭 링크는 데이터가 포함된 원본 파일과 이 원본 파일을 가리키는 다른 파일로 구성됩니다. 심볼릭 링크는 하드 링크보다 유연합니다. 심볼릭 링크는 서로 다른 파일 시스템에 사용할 수 있으며 디렉토리에 연결할 수도 있습니다.

하드 링크 및 심볼릭 링크에 대한 자세한 내용은 UNIX 시스템 설명서를 참조하십시오.

파일 시스템 링크를 사용하면 기본 문서 디렉토리 외부에 있는 문서를 가리키는 포인터를 쉽게 만들 수 있으며 이러한 링크는 누구나 만들 수 있습니다. 따라서 사용자가 민감한 파일(예: 보안 문서나 시스템 암호 파일)에 대한 포인터를 만들 수 있음을 주의해야 합니다.

관리 인터페이스를 사용하여 심볼릭 링크를 제한하는 방법은 다음과 같습니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "Doc 디렉토리" 탭을 누릅니다.
5. "심볼릭 링크"를 누릅니다.
6. 자원 선택기에서 "전체 서버"를 선택하여 변경 내용을 전체 가상 서버에 적용하거나 가상 서버 내의 특정 디렉토리를 찾습니다.
7. 소프트 및 하드 링크의 활성화 여부와 시작할 디렉토리를 선택합니다.
8. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

사용자 공개 정보 디렉토리 사용자 정의(UNIX의 경우)

사용자가 자신의 웹 페이지를 관리하고자 하는 경우가 있습니다. 이 경우 서버의 모든 사용자가 관리자의 간섭 없이 홈 페이지 및 다른 문서를 만들 수 있는 공개 정보 디렉토리를 구성할 수 있습니다.

주 Windows 시스템의 관리 인터페이스에 "사용자 문서 디렉토리" 페이지가 나타나기는 하지만 이 기능은 사용할 수 없습니다.

이 시스템에서 클라이언트는 서버가 공개 정보 디렉토리로 인식하는 특정 URL을 사용하여 서버에 액세스할 수 있습니다. 예를 들어, 접두어 ~와 디렉토리 `public_html`을 선택했다고 가정합니다. `http://www.sun.com/~jdoe/aboutjane.html`에 대한 요청이 오면 서버는 `~jdoe`가 사용자의 공개 정보 디렉토리를 가리키는 것으로 인식합니다. 따라서 시스템의 사용자 데이터베이스에서 `jdoe`를 검색하여 Jane의 홈 디렉토리를 찾습니다. 그런 다음 `~/jdoe/public_html/aboutjane.html`을 찾습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- 공개 정보 디렉토리 구성
- 내용 게시 제한
- 시작할 때 전체 암호 파일 로드

공개 정보 디렉토리 구성

관리 인터페이스를 사용하여 가상 서버가 공개 디렉토리를 사용하도록 구성하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "Doc 처리" 탭을 누릅니다.
5. "사용자 Doc 디렉토리"를 누릅니다.
6. 사용자 URL 접두어를 선택합니다.

   틸드 문자(~)가 사용자의 홈 디렉토리에 액세스하는 표준 UNIX 접두어이기 때문에 일반적으로 사용되는 접두어는 ~입니다.
7. 사용자의 홈 디렉토리에서 서버가 HTML 파일을 찾을 하위 디렉토리를 선택합니다.

   일반적으로 이 디렉토리는 `public_html`입니다.
8. 암호 파일을 지정합니다.

   서버에서는 시스템의 사용자가 나열되어 있는 파일을 찾을 위치를 알고 있어야 합니다. 서버에서는 이 파일을 사용하여 유효한 사용자 이름을 확인하고 해당 사용자의 홈 디렉토리를 찾습니다. 이를 위해 시스템 암호 파일을 사용하면 서버에서는 표준 라이브러리 호출을 사용하여 사용자를 찾습니다. 또는 다른 사용자 파일을 만들어서 사용자를 찾을 수 있습니다. 이러한 사용자 파일은 절대 경로로 지정할 수 있습니다.

   파일에 있는 각 줄의 형식은 다음과 같아야 합니다(필요 없는 `/etc/passwd` 파일의 요소에는 * 기호가 표시되어 있음).

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

9. 시작할 때 암호 데이터베이스를 로드할지 여부를 선택합니다.

자세한 내용은 399페이지의 “시작할 때 전체 암호 파일 로드”를 참조하십시오.

10. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

사용자에게 개별적인 디렉토리를 지정하는 다른 방법은 모든 사용자가 수정할 수 있는 중앙 디렉토리에 대한 URL 매핑을 만드는 것입니다.

내용 게시 제한

경우에 따라 사용자 문서 디렉토리를 통해 내용을 게시할 수 있는 사용자 계정을 제한해야 할 수 있습니다. 사용자의 게시를 제한하려면 `/etc/passwd` 파일에서 사용자의 홈 디렉토리 경로 뒤에 슬래시를 추가합니다.

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

위 코드를 다음과 같이 수정합니다.

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

이렇게 수정하면 Sun ONE Application Server는 이 사용자의 디렉토리에 있는 페이지를 제공하지 않습니다. 이 URI를 요청하는 브라우저는 "404 File Not Found" 오류를 받게 되고 이 404 오류는 액세스 로그에 기록됩니다.

나중에 이 사용자가 내용을 게시할 수 있도록 하려면 `/etc/passwd` 항목의 뒤에 있는 슬래시를 제거한 다음 응용 프로그램 서버 인스턴스를 다시 시작합니다.

시작할 때 전체 암호 파일 로드

시작할 때 전체 암호 파일을 로드할 수 있는 옵션도 있습니다. 이 옵션을 선택하면 서버는 시작할 때 메모리에 암호 파일을 로드하여 사용자를 더 빨리 찾을 수 있게 됩니다. 하지만 암호 파일이 매우 큰 경우 이 옵션을 사용하면 메모리 사용량이 지나치게 많을 수 있습니다.

문서 환경 설정

이 절에서는 다음 내용을 설명합니다.

- 색인 파일 이름 입력
- 디렉토리 색인 선택
- 서버 홈 페이지 지정
- 기본 MIME 유형 지정

관리 인터페이스를 사용하여 문서 환경 설정을 구성하려면 다음 단계를 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "Doc 처리" 탭을 누릅니다.
5. "Doc 환경 설정"을 누릅니다.
6. 다음 절의 설명에 따라 적절한 필드 값을 선택합니다.
7. "확인"을 누릅니다.

설정할 수 있는 환경 설정에 대해서는 다음 절에서 자세히 설명합니다. 자세한 내용은 온라인 도움말을 참조하십시오.

색인 파일 이름 입력

URL에 문서 이름이 지정되어 있지 않으면 서버는 자동적으로 색인 파일을 표시합니다. 기본 색인 파일은 `index.html` 및 `home.html`입니다. 두 개 이상의 색인 파일을 지정한 경우 서버는 이 필드에 이름이 나타나는 순서대로 검색합니다. 예를 들어, 색인 파일이 `index.html` 및 `home.html`인 경우 서버는 `index.html`을 찾으며 이 파일을 찾을 수 없으면 `home.html`을 찾습니다.

디렉토리 색인 선택

문서 디렉토리에는 대부분 여러 개의 하위 디렉토리가 있습니다. 예를 들어, `products`라는 디렉토리와 `people`이라는 디렉토리 등이 있을 수 있습니다. 클라이언트가 이러한 디렉토리의 개요(또는 색인)에 액세스할 수 있게 허용하는 것이 유용합니다.

서버는 디렉토리 내용의 개요로 작성하여 관리하는 파일인 `index.html` 또는 `home.html` 이라는 색인 파일을 디렉토리에서 검색하여 디렉토리를 색인합니다. 자세한 내용은 400페이지의 “색인 파일 이름 입력”을 참조하십시오. 어떠한 파일이라도 이러한 기본 이름 중 하나로 이름을 지정하면 디렉토리에 대한 색인 파일로 지정할 수 있습니다. 따라서 CGI 프로그램을 색인으로 사용할 수도 있습니다.

색인 파일이 없으면 서버는 문서 루트에 있는 모든 파일이 나열된 색인 파일을 생성합니다.

주의 서버가 방화벽 외부에 있는 경우에는 디렉토리 색인 기능을 비활성화하여 디렉토리 구조와 파일 이름에 액세스하지 못하도록 해야 합니다.

서버 홈 페이지 지정

최종 사용자가 처음 서버에 액세스하면 일반적으로 홈 페이지를 제일 먼저 보게 됩니다. 일반적으로 이 파일에는 서버에 대한 일반 정보와 다른 문서에 대한 링크가 들어 있습니다.

기본적으로 서버는 "문서 기본 설정" 페이지의 "색인 파일 이름" 필드에 지정된 색인 파일을 찾아서 이 파일을 홈 페이지로 사용합니다. 하지만 홈 페이지로 사용할 파일을 지정할 수도 있습니다.

기본 MIME 유형 지정

서버에서 클라이언트에 문서를 전송할 때 클라이언트가 해당 문서를 올바른 방법으로 표시할 수 있도록 문서의 유형을 식별하는 절을 포함합니다. 하지만 서버에 문서의 확장명이 정의되어 있지 않아서 서버에서 문서의 올바른 유형을 판별할 수 없는 경우도 있습니다. 이 경우에는 기본값이 전송됩니다.

일반적으로 text/plain이 기본값이지만 사용자의 서버에 가장 많이 저장되어 있는 파일 유형으로 설정하는 것이 좋습니다. 많이 사용되는 MIME 유형은 다음과 같습니다.

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

오류 응답 사용자 정의

가상 서버에서 오류가 발생한 경우 클라이언트에게 자세한 메시지를 보내는 사용자 정의 오류 응답을 지정할 수 있습니다. 이 경우 파일을 지정하거나 CGI 프로그램이 실행되게 할 수 있습니다.

예를 들어, 특정 디렉토리에 오류가 발생할 경우의 서버 동작을 변경할 수 있습니다. 클라이언트가 액세스 제어로 보호된 서버 부분에 연결하려고 하면 계정을 얻는 방법을 설명하는 오류 메시지를 반환할 수 있습니다.

오류 응답을 사용자 정의하려면 오류에 응답할 HTML 파일을 만들거나 오류에 응답하여 CGI 프로그램이 실행되도록 해야 합니다. 이렇게 한 다음 관리 인터페이스에서 응답을 활성화합니다.

관리 인터페이스를 사용하여 사용자 정의 오류 응답을 활성화하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "Doc 처리" 탭을 누릅니다.
5. "오류 응답"을 누릅니다.
6. 자원 선택기에서 "전체 서버"를 선택하여 변경 내용을 전체 가상 서버에 적용하거나 가상 서버 내의 특정 디렉토리를 찾습니다.

7. 변경할 각 오류 코드에 오류 응답이 포함된 CGI나 파일에 대한 절대 경로를 지정합니다.
8. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

국가별 문자 집합 변경

문서의 문자 집합은 해당 문서가 작성된 언어에 의해 일부 결정됩니다. 자원을 선택하고 해당 자원에 대한 문자 집합을 입력하면 문서, 문서 집합 또는 디렉토리에 대한 클라이언트의 기본 문자 집합을 무시할 수 있습니다.

브라우저는 HTTP에 MIME 유형 `charset` 매개 변수를 사용하여 문자 집합을 변경할 수 있습니다. 서버에서 응답에 이 매개 변수를 포함하면 그에 따라 브라우저에서 문자 집합을 변경합니다. 예를 들면 다음과 같습니다.

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

다음은 RFC 1700에 지정된 `charset` 이름입니다(이름이 `x-`로 시작된다는 점만 다름).

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

또한 `us-ascii`에 대한 다음 별칭도 인식합니다.

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

iso_8859-1에 대한 다음 별칭을 인식합니다.

- latin1
- iso_8859-1
- iso_8859-1:1987
- iso-ir-100
- ibm819
- cp819

관리 인터페이스를 사용하여 문자 집합을 변경하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "Doc 처리" 탭을 누릅니다.
5. "국가별 문자"를 누릅니다.
6. 자원 선택기에서 "전체 서버"를 선택하여 변경 내용을 전체 가상 서버에 적용하거나 가상 서버 내의 특정 디렉토리를 찾습니다.
7. 서버 전체 또는 일부에 대한 문자 집합을 설정합니다.

   이 필드를 비워 두면 문자 집합이 NONE으로 설정됩니다.
8. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

문서 바닥글 설정

서버의 특정 절에 있는 모든 문서에 대해 최종 수정 시 삽입할 수 있는 문서 바닥글을 지정할 수 있습니다. 이 바닥글은 CGI 스크립트의 출력이나 구문 분석된 HTML(.shtml) 파일 이외의 모든 파일에 적용됩니다. 문서 바닥글이 CGI 스크립트 출력이나 구문 분석된 HTML 파일에도 나타나게 하려면 바닥글 텍스트를 별도의 파일에 입력한 다음, 코드나 다른 서버측 포함을 추가하여 이 파일이 페이지의 출력에 나타나도록 하십시오.

관리 인터페이스를 사용하여 문서 바닥글을 설정하려면 다음 단계를 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.

3. 편집할 가상 서버의 이름을 누릅니다.
4. "Doc 처리" 탭을 누릅니다.
5. "Doc 바닥글"을 누릅니다.
6. 자원 선택기에서 "전체 서버"를 선택하여 변경 내용을 전체 가상 서버에 적용하거나 가상 서버 내의 특정 디렉토리를 찾습니다.

   디렉토리를 선택하면 서버가 해당 디렉토리에 있는 디렉토리나 파일에 대한 URL을 수신한 경우에만 문서 바닥글이 적용됩니다.
7. 바닥글을 포함할 파일의 유형을 지정합니다.
8. 날짜 형식을 지정합니다.
9. 바닥글에 나타날 텍스트를 입력합니다.

   문서 바닥글에 사용할 수 있는 문자의 최대 수는 765자입니다. 문서 최종 수정일을 포함하려면 :LASTMOD:라는 문자열을 입력합니다.

자세한 내용은 온라인 도움말을 참조하십시오.

URL 전달 구성

URL 전달을 사용하면 문서 요청을 다른 서버로 리디렉션할 수 있습니다. URL 전달 또는 리디렉션은, 예를 들어 파일을 다른 디렉토리나 서버로 옮긴 경우 URL이 변경되었음을 사용자에게 알리는 방법입니다. 또한 다른 서버에 있는 문서를 요청하는 사람에게 해당 문서를 자연스럽게 제공할 때에도 리디렉션을 사용할 수 있습니다.

예를 들어, http://www.sun.com/info/movies를 접두어 film.sun.com에 전달하면 http://www.sun.com/info/movies라는 URL은 http://film.sun.com/info/movies로 리디렉션됩니다.

경우에 따라서는 하나의 하위 디렉토리에 있는 모든 문서에 대한 요청을 특정 URL로 리디렉션해야 할 수 있습니다. 예를 들어, 특정 디렉토리가 너무 많은 트래픽을 유발하기 때문에 옮겨야 하거나 어떤 이유로 문서를 더 이상 서비스하지 않는 경우 이러한 문서에 대한 요청을 해당 문서를 사용할 수 없는 이유에 대해 설명하는 페이지로 리디렉션할 수 있습니다. 예를 들어, /info/movies에 접두어를 사용할 경우 http://www.sun.com/explain.html로 리디렉션할 수 있습니다.

관리 인터페이스를 사용하여 URL 전달을 구성하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "HTTP/HTML" 탭을 누릅니다.
5. "URL 전달"을 누릅니다.
6. 리디렉션할 URL 접두어를 입력한 다음 다른 접두어로 리디렉션할지 정적 URL로 리디렉션할지 지정합니다.
7. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

서버 구문 분석 HTML 설정

HTML은 일반적으로 어떠한 작업 없이 디스크에 있는 그대로 전송됩니다. 하지만 서버는 문서를 전송하기 전에 HTML 파일에서 특별한 명령을 검색할 수 있습니다. 즉 HTML 구문 분석을 할 수 있습니다. 서버가 이러한 파일을 구문 분석하여 문서에 요청별 정보 또는 파일을 삽입하도록 하려면 먼저 HTML 구문 분석을 활성화해야 합니다.

관리 인터페이스를 사용하여 HTML 구문 분석을 설정하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "HTTP/HTML" 탭을 누릅니다.
5. "HTML 구문 분석"을 누릅니다.
6. 자원 선택기에서 "전체 서버"를 선택하여 변경 내용을 전체 가상 서버에 적용하거나 가상 서버 내의 특정 디렉토리를 찾습니다.

   디렉토리를 선택하면 서버는 서버에서는 해당 디렉토리 또는 해당 디렉토리에 있는 파일에 대한 URL을 수신한 경우에만 HTML을 구문 분석합니다.
7. 서버 구문 분석 HTML의 활성화 여부를 선택합니다.

   exec 태그를 제외하고 HTML 파일에 대해 활성화하거나 HTML 파일과 exec 태그에 대해 활성화할 수 있습니다. 이 경우 HTML 파일이 서버의 다른 프로그램을 실행할 수 있습니다.

8. 구문 분석할 파일을 선택합니다.

   확장명이 .shtml인 파일만 구문 분석할지 모든 HTML 파일을 구문 분석할지 선택할 수 있습니다. 모두 구문 분석하면 성능이 느려질 수 있습니다. UNIX를 사용하는 경우 실행 권한을 설정하여 UNIX 파일을 구문 분석하도록 선택할 수도 있지만 신뢰할 수 없을 수 있습니다.

9. "확인"을 누릅니다.

서버에서 구문 분석한 HTML을 허용하도록 설정하는 방법에 대한 자세한 내용은 온라인 도움말을 참조하십시오.

서버 구문 분석 HTML의 사용에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guide to Web Applications*를 참조하십시오.

캐시 제어 지시문 설정

Sun ONE Application Server는 캐시 제어 지시문을 사용하여 프록시 서버에 캐시될 정보를 제어합니다. 캐시 제어 지시문을 사용하면 프록시의 기본 캐시를 무시하여 민감한 정보가 캐시되어 검색되는 것을 방지할 수 있습니다. 이러한 지시문이 작동하려면 프록시 서버가 HTTP 1.1과 호환되어야 합니다.

HTTP 1.1에 대한 자세한 내용은 다음에서 찾을 수 있는 Hypertext Transfer Protocol--HTTP/1.1 사양(RFC 2068)을 참조하십시오.

http://www.ietf.org/

관리 인터페이스를 사용하여 캐시 제어 지시문을 설정하려면 다음 작업을 수행합니다.

1. 왼쪽 창에서 응용 프로그램 서버 인스턴스에 대해 "HTTP 서버"를 엽니다.
2. "가상 서버"를 엽니다.
3. 편집할 가상 서버의 이름을 누릅니다.
4. "HTTP/HTML" 탭을 누릅니다.
5. "캐시 제어 지시문"을 누릅니다.
6. 필드를 입력합니다. 응답 지시문에 사용할 수 있는 값은 다음과 같습니다.
   - **공개.** 응답을 모든 캐시로 캐시할 수 있습니다. 이 값이 기본값입니다.
   - **개인.** 응답을 개인(공유되지 않은) 캐시로만 캐시할 수 있습니다.
   - **캐시 안 함.** 응답을 캐시하지 않습니다.

- **저장 안 함.** 캐시는 요청이나 응답을 비휘발성 저장 장치의 어떠한 위치에도 저장하지 않아야 합니다.
- **재검증 필요.** 캐시 항목을 원래 서버에서 재검증해야 합니다.
- **최대 캐시 사용 기간(초).** 클라이언트가 이 기간보다 긴 응답은 받지 않습니다.

7. "확인"을 누릅니다.

자세한 내용은 온라인 도움말을 참조하십시오.

강력한 암호화 사용

강력한 암호화의 설정에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Guide to Security*를 참조하십시오.

부록

부록 A

명령줄 인터페이스 사용

이 부록에서는 시스템 명령 프롬프트에서 한 번에 하나의 명령을 실행하는 단일 모드, 환경 수준 정보를 다시 입력할 필요 없이 여러 개의 명령을 실행할 수 있는 다중 모드, 스크립트 및 프로그램에 명령줄 인터페이스(`asadmin` 유틸리티)를 사용하는 지침을 제공합니다. 관리 인터페이스 화면 대신 명령줄 인터페이스를 사용할 수 있습니다.

이 부록에서는 다음 내용을 설명합니다.

- 명령줄 인터페이스 정보
- asadmin 사용
- 보안 고려 사항
- 동시에 액세스할 때 고려해야 할 사항
- 명령 참조

명령줄 인터페이스 정보

이 절은 다음 내용으로 구성되어 있습니다.

- asadmin 유틸리티 정보
- Ant 작업 정보
- 기타 명령줄 유틸리티 정보

asadmin 유틸리티 정보

`asadmin` 유틸리티는 모든 구성 및 관리 작업을 수행합니다. 관리 인터페이스 대신 이 유틸리티를 사용할 수 있습니다.

Ant 작업 정보

많은 개발자들이 J2EE 응용 프로그램의 개발 프로세스를 가속화하기 위해 Ant를 사용합니다. Ant 스크립트는 몇 가지 작업에 `asadmin` 유틸리티를 사용합니다. 개발자는 응용 프로그램 구축, 모듈 및 응용 프로그램의 배포 및 배포 해제, Sun ONE Application Server 관리에 Ant 작업을 사용합니다.

Ant 작업에 대한 자세한 내용은 *Sun ONE Application Server Developer's Guid*e를 참조하십시오.

Ant에 대한 자세한 내용은 Jakarta Project 사이트 http://jakarta.apache.org/ant/를 참조하십시오.

기타 명령줄 유틸리티 정보

Sun ONE Application Server에는 추가적인 명령줄 유틸리티가 있습니다. 다음 표에는 유틸리티 목록과 그에 대한 간단한 설명이 있습니다.

기타 명령줄 유틸리티

유틸리티	정의
appclient	응용 프로그램 클라이언트 컨테이너를 시작하고 응용 프로그램 JAR 파일에 패키지화된 클라이언트 응용 프로그램을 호출합니다.
capture-schema	데이터베이스 스키마와 매핑 정보를 가져옵니다.
flexanlg	서버에 대한 통계를 생성합니다.
htpasswd	사용자 인증 파일을 만듭니다.
package-appclient	응용 프로그램 컨테이너 라이브러리와 jar 파일을 압축합니다. 자세한 내용은 *Sun ONE Application Server Developer's Guide to Clients*를 참조하십시오.
verifier	DTD를 사용하여 J2EE 배포 설명자를 확인합니다. 자세한 내용은 *Sun ONE Application Server Developer's Guide*를 참조하십시오.

기타 명령줄 유틸리티

유틸리티	정의
wscompile	서비스 정의 인터페이스를 가져와서 클라이언트 스텁 또는 서버측 기초를 생성하거나 제공된 인터페이스의 WSDL 집합을 생성합니다.
wsdeploy	배포할 수 있는 WAR 파일을 생성합니다.

이러한 유틸리티에 대한 자세한 내용은 온라인 도움말을 참조하십시오.

asadmin 사용

`asadmin` 유틸리티는 관리 작업을 수행하는 명령 집합을 포함합니다. 관리 인터페이스를 사용하여 수행할 수 있는 대부분의 작업에 이 명령들을 사용할 수 있습니다. `asadmin` 유틸리티는 *install_dir*/bin에 있으며 이 위치에서 실행할 수 있습니다. Windows에서 `asadmin.bat` 파일을 두 번 누르면 명령 창에 `asadmin` 유틸리티가 시작되고 다중 모드로 실행됩니다.

일부 HTTP 서버 관련 등록 정보와 Administration Server 등록 정보는 명령줄을 사용하여 설정할 수 없기 때문에 관리 인터페이스를 사용하여 설정해야 합니다. `server.xml` 구성 파일에 저장된 모든 등록 정보를 설정할 수 있는 반면 `init.conf` 또는 `obj.conf`에 저장된 등록 정보는 설정할 수 없습니다. 구성 파일에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Configuration File Reference*를 참조하십시오.

각 명령에 대한 자세한 내용은 431페이지의 "명령 참조" 및 온라인 도움말을 참조하십시오.

이 절은 다음 내용으로 구성되어 있습니다.

- 명령 구문 이해
- 단일 모드 및 다중 모드 사용
- 대화식 및 비대화식 옵션 사용
- 환경 명령 사용
- 암호 파일 옵션 사용
- asadmin을 로컬 또는 원격으로 실행
- 명령줄 호출 사용
- 이스케이프 문자 사용

- get 및 set 명령 사용
- 도움말 사용
- 출력 및 오류 보기

명령 구문 이해

`asadmin` 유틸리티 구문은 다음과 같습니다.

asadmin command -short-option argument --long-option argument operand

명령

명령은 수행될 작업 또는 태스크로 대소문자를 구분합니다.

옵션

옵션으로 유틸리티의 명령 수행 방법을 수정합니다. 옵션은 대소문자를 구분합니다. 짧은 옵션 앞에는 한 개의 대시(-), 긴 옵션 앞에는 두 개의 대시(--)를 붙입니다. 많은 옵션에 있어 길거나 짧은 형식 중 한 가지를 사용할 수 있습니다. 예를 들어, user 옵션은 `--user` 또는 `-u` 형식으로 사용할 수 있습니다. 옵션에는 필수 옵션과 선택 옵션이 있으며 선택 옵션은 명령 구문에서 대괄호로 표시됩니다. 명령을 실행할 때는 필수 옵션을 모두 포함시켜야 합니다. 그렇지 않으면 오류 메시지가 나타나고 명령이 실행되지 않습니다.

사용할 수 있는 긴 옵션 및 짧은 옵션 이름 목록은 431페이지의 “명령 참조”의 표 "긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수"를 참조하십시오.

대부분의 옵션에는 인수 값(예: --port *port_number*)이 필수이지만 부울 옵션은 그 예외로 기능을 on 또는 off로 전환하며 인수 값을 필요로 하지 않습니다.

또한 옵션을 환경 변수에 저장할 수 있습니다. 자세한 내용은 417페이지의 “환경 명령 사용”을 참조하십시오. 옵션에 해당하는 전체 환경 변수 목록은 464페이지의 “긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수 대응”을 참조하십시오.

부울 옵션

부울 옵션은 on 또는 off 사이를 전환합니다(예를 들어, `--interactive`는 옵션 관련 메시지가 표시되는 대화형 모드를 활성화하고 `--no-interactive`는 대화형 모드를 비활성화합니다). 긴 옵션 앞에 --no-를 붙이면 옵션이 off로 전환됩니다. 짧은 옵션 이름을 지정하면 항상 기본값의 반대로 설정됩니다.

짧은 부울 옵션을 그룹화하여 사용할 수 있습니다. 예를 들어, `-Ie`를 사용하여 interactive (짧은 옵션 `-I`)와 echo (짧은 옵션 `-e`)를 지정할 수 있습니다.

피연산자

피연산자는 하나의 공백 또는 탭으로 구분하며 명령 구문에서 순서와 상관없이 사용됩니다. 옵션을 뒤에 붙이지 않고 --를 사용하여 옵션과 피연산자를 구분할 수 있습니다. 다음 인수는 한 개의 대시(-)로 시작하더라도 피연산자로 취급됩니다. 예를 들어 다음 명령에서

```
asadmin> create-jvm-options --instance server1 -- -Xmx1500m
```

`-XMx1500m`은 대시 하나로 시작하지만 피연산자로 취급됩니다.

구문 예

`asadmin create-instance [--user` *admin_user*`] [--password` *admin_password*`] [-H` *host_name*`] [--port` *port_number*`] [--sysuser` *sys_user*`] [--domain` *domain_name*`] [--local=`*true/false*`] [--passwordfile` *file_name*`] [--secure | -s] --instanceport` *instance_port instance_name*

이 구문 예에서 `-H`는 hostname의 짧은 옵션이고 `--user`는 인수가 *admin_user*인 긴 옵션이며 *instance_name*이 피연산자입니다. 선택 옵션은 대괄호 안에 있습니다.

다음은 실제 값을 사용한 구문의 예입니다. 이 예에는 선택 옵션 중 일부가 사용되지 않았습니다.

`asadmin create-instance --user admin --password` *password* `-H austen --port 4848 --instanceport 1024 server2`

단일 모드 및 다중 모드 사용

`asadmin`를 단일 모드 또는 다중 모드로 실행할 수 있습니다. 단일 모드에서는 명령 프롬프트에서 한 번에 하나의 명령씩 실행합니다. 다중 모드에서는 환경 수준 정보를 다시 입력하지 않고도 여러 개의 명령을 실행할 수 있습니다.

파일 입력을 사용하는 경우에는 단일 모드에서 명령이 실패하면 프로그램이 종료됩니다. 다중 모드를 사용하는 경우에는 명령이 실패하면 `asadmin` 프롬프트로 되돌아 옵니다.

단일 모드

명령줄 프롬프트에서 명령줄 인터페이스의 단일 명령을 호출하면 단일 모드로 실행하는 것입니다. 명령줄 인터페이스는 명령을 실행한 후 다시 명령 프롬프트로 나갑니다. 명령 프롬프트에서 명령줄 인터페이스를 실행하려면 *install_dir*/appserv/bin 디렉토리로 이동한 다음 명령 프롬프트에 명령을 입력합니다.

`> asadmin` *command options arguments*

예:

```
> asadmin create-instance --user admin --password password -H austen
--port 4848 --instanceport 1024 server2
```

다중 모드

다중 모드를 사용하면 환경을 처음에 설정하여 서버 이름, 포트, 암호 등의 환경 수준 정보를 다시 입력하지 않고도 여러 개의 명령을 실행할 수 있습니다. 다중 모드를 사용할 경우 가장 중요한 이점은 `asadmin`이 메모리에 남아 있기 때문에 명령을 입력하고 실행하는 시간이 크게 단축될 수 있다는 것입니다. 이러한 환경 변수가 운영 체제 수준에 설정된 경우에는 다중 모드에서 이러한 설정을 선택합니다. `asadmin` 유틸리티는 사용자가 변경할 때까지 이러한 설정을 사용합니다.

Windows에서는 `asadmin.bat` 파일을 실행하면 자동으로 다중 모드가 됩니다.

UNIX에서는 `asadmin` 유틸리티를 명령줄에서 다중 모드로 시작하려면 다음과 같이 입력합니다.

```
> asadmin multimode
```

다중 모드에서 명령 프롬프트는 `asadmin`으로 변경되며 사용자는 이 때 `asadmin` 프롬프트에 명령을 입력할 수 있습니다. 유틸리티 이름을 사용할 필요는 없습니다. 예:

```
asadmin> create-instance --user admin --password password -H austen
--port 4848 --instanceport 1024 server2
```

exit 또는 quit를 입력하여 다중 모드를 종료하면 명령 프롬프트로 되돌아 옵니다.

여러 다중 모드

다중 모드 세션에서 다음 명령을 사용하여 다중 모드를 호출할 수도 있습니다.

```
asadmin> multimode
```

일단 두 번째 다중 모드 환경을 종료하면 원래의 다중 모드 환경으로 되돌아 옵니다.

예를 들어, server1을 다중 모드에서 관리하는 경우 server2를 관리하여 두 가지 서버를 비교하려면 server1에 대한 다중 모드에서 server2에 대한 다중 모드를 호출할 수 있습니다. 사용자는 현재의 다중 모드 세션을 종료하지 않아도 되므로 환경 설정을 그대로 유지할 수 있습니다. server2에 대해 사용하는 다중 모드 세션을 종료하면 server1 다중 모드 환경으로 되돌아 옵니다.

대화식 및 비대화식 옵션 사용

명령줄 인터페이스를 사용할 때 대화형 또는 비대화형 모드로 사용할 수 있습니다. 대화형 모드를 선택한 경우 암호를 지정하지 않으면 암호를 지정하라는 메시지가 나타납니다. 대화형 모드는 기본적으로 활성화되어 있습니다.

export 명령을 사용하여 대화형 환경 변수를 설정함으로써 대화형 모드를 활성화 및 비활성화할 수 있습니다. 자세한 내용은 표 "export 명령과 함께 사용되는 환경 변수"를 참조하십시오.

단일 모드에서는 모든 상황에 대화식 옵션을 사용할 수 있습니다. 다중 모드에서는 명령 프롬프트에서 한 번에 한 개의 명령을 실행할 때와 파일에서 다중 모드로 실행할 때 대화식 옵션을 사용할 수 있습니다. 그러나 입력 스트림에서 파이프된 다중 모드의 명령과 다른 프로그램에서 호출된 명령은 대화형 모드에서 실행할 수 없습니다.

환경 명령 사용

asadmin 유틸리티는 환경 명령을 사용하여 설정할 수 있는 환경 변수 집합을 포함합니다. 다중 모드에서는 이러한 변수를 한 번 설정하면 다중 모드를 종료할 때까지 환경을 다시 설정하지 않아도 됩니다. 이러한 환경 변수를 운영 체제 수준에서 설정할 수도 있습니다. 그런 경우 이러한 환경 변수는 다중 모드에 들어갈 때 자동으로 선택되며 다중 모드를 종료한 후에도 지속됩니다.

환경 변수는 할당으로 언제든지 설정할 수 있는 이름/값 쌍입니다. 환경 변수는 옵션 이름 앞에 붙은 대문자 AS_ADMIN_ 접두어로 구성됩니다. 예를 들어 Administration Server 사용자를 설정하려면 다음과 같이 입력할 수 있습니다.

```
export AS_ADMIN_USER=administrator
```

여기서 *administrator*는 관리자의 사용자 이름입니다.

또한 다음과 같이 입력하면 AS_ADMIN_USER 값을 `asadmin` 명령에 사용할 수 있습니다.

```
asadmin multimode
asadmin> export AS_ADMIN_HOST=austen
```

이 다중 모드 세션에 있는 동안에는 Administration Server 호스트 이름이 austen으로 설정되고 다시 할당하지 않는 한 그대로 유지됩니다.

또한 한 번에 여러 환경 변수의 값을 설정하고 내보낼 수 있습니다. 예:

```
asadmin> export AS_ADMIN_PORT=4848 AS_ADMIN_USER=admin
```

현재의 환경 변수 설정을 보려면 다음과 같이 export 명령을 인수 없이 사용합니다.

```
asadmin> export
AS_ADMIN_HOST=austen
AS_ADMIN_PORT=4848
AS_ADMIN_USER=admin
```

환경에서 변수와 그 값을 제거하려면 `unset` 명령을 사용합니다. 예:

```
asadmin> unset AS_ADMIN_HOST
```

변수를 다시 설정하거나 `asadmin` 명령의 일부로 다른 값을 설정함으로써 환경 변수의 설정 값을 대체할 수 있습니다. 예:

```
asadmin> export AS_ADMIN_HOST=dickens
asadmin> show-instance-status --host austen instance-name
```

이 예는 Administration Server 호스트 `austen`에 있는 인스턴스의 인스턴스 상태를 보여주는데 그 값이 이전 호스트 값인 `dickens`를 대체하기 때문입니다.

내보낸 변수를 사용하지 않는 경우에는 대부분의 명령과 함께 다음 옵션을 사용하거나 기본값을 사용해야 합니다. 다음 기본값 목록은 464페이지의 "긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수 대응"을 참조하십시오.

- --host
- --port
- --user
- --password or --passwordfile

- --secure=true (안전한 경우)
- --instance (필요한 경우)

다음 표 "export 명령과 함께 사용되는 환경 변수"에서는 export 명령과 함께 사용하는 환경 변수에 대해 설명합니다. 이 변수들은 특히 환경을 설정하므로 가장 일반적으로 사용됩니다. 첫 번째 열은 환경 변수 이름을 나타내고 두 번째 열은 사용 목적을 설명하며 아무것도 설정되지 않은 경우에는 기본값을 보여줍니다. 전체 환경 변수 목록은 464페이지의 “긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수 대응”을 참조하십시오.

export 명령과 함께 사용되는 환경 변수

환경 변수	사용
AS_ADMIN_HOST	Administration Server의 호스트 이름. 값이 지정되지 않은 경우에는 localhost를 사용합니다.
AS_ADMIN_PORT	Administration Server의 포트 번호. 값이 지정되지 않은 경우에는 4848을 사용합니다.
AS_ADMIN_USER	명령을 실행하는 사용자의 사용자 이름
AS_ADMIN_PASSWORD	명령을 실행하는 사용자 이름의 암호. 사용자 이름과 암호는 사용자를 인증하고 사용자가 서버를 관리하도록 허용되는지를 확인하는 데 사용됩니다. 이는 관리 인터페이스를 통해 Administration Server에 액세스할 때 이루어지는 인증과 동일합니다.
AS_ADMIN_SECURE	안전한 경우 `=true`
AS_ADMIN_INSTANCE	Sun ONE Application Server의 인스턴스를 설정합니다. 인스턴스 이름을 인수로 사용하는 모든 후속 명령(피연산자로 사용되는 명령은 해당 안 됨)은 이 지정된 인스턴스를 사용합니다.

암호 파일 옵션 사용

명령줄에 암호를 입력하지 않거나 암호에 대한 환경 변수를 설정하려면 암호 파일을 만들어 명령줄 옵션으로 사용할 수 있습니다.

password 옵션을 가진 모든 명령에는 대신 사용할 수 있는 passwordfile 옵션이 있습니다. 암호 파일은 다음 행으로 이루어져 있습니다.

AS_ADMIN_PASSWORD=*value*

AS_ADMIN_ADMINPASSWORD=*value*

AS_ADMIN_USERPASSWORD=*value*

passwordfile 옵션을 사용할 경우 파일의 암호는 다중 모드 환경으로 내보내지며 지정된 password 옵션이 없는 후속 명령은 이 값을 사용합니다.

명령줄에 암호 옵션과 암호 파일 옵션을 모두 지정할 경우에는 암호 파일의 값이 다중 모드 환경으로 내보내지지만 암호 옵션이 암호 파일 옵션에 우선하므로 현재 명령은 암호 옵션에 지정된 암호를 사용합니다.

asadmin을 로컬 또는 원격으로 실행

일반적으로 asadmin 유틸리티는 Administration Server를 통해 명령을 보냅니다. 따라서 Sun ONE Application Server가 설치되어 있는 시스템에서는 asadmin을 실행할 필요가 없습니다. 그러나, 대부분의 asadmin 명령을 수행하기 위해서는 Administration Server가 실행되고 있어야 합니다.

create-instance와 같은 일부 명령은 로컬에서 실행되는 옵션을 가집니다. create-instance와 함께 --local=true 옵션을 사용하는 경우에는 서버가 설치된 시스템에서 명령을 실행해야 합니다. 그러나 인스턴스를 만들기 위해 Administration Server를 실행할 필요는 없습니다.

일부 명령은 로컬로 실행되어야 합니다. 예를 들어, Administration Server와 모든 Administration Server의 인스턴스를 시작하는 start-appserv 명령은 원격으로 실행될 수 없습니다. 이는 이 명령을 사용하여 Administration Server를 시작할 때까지 Administration Server가 실행되지 않기 때문입니다.

Administration Server에 대한 자세한 내용은 2장, “Administration Server 기본 설정”을 참조하십시오.

다음 명령은 로컬 및 원격으로 실행될 수 있습니다.

- create-instance
- delete-instance
- list-instances
- start-instance
- stop-instance

- display-license
- version
- stop-domain
- restart-instance
- list-domains

이러한 명령의 경우에는 로컬 옵션을 지정하지 않고도 명령을 로컬로 실행하도록 선택할 수 있습니다. 기본적으로 명령 구문에서 사용자, 암호, 호스트 및 포트의 값을 지정하는 경우 이러한 옵션에 대해 로컬 값을 지정할 수 있음에도 불구하고 이 명령은 원격 명령으로 취급됩니다. 이러한 옵션의 값을 지정하지 않으면 명령은 기본적으로 로컬로 실행됩니다.

명령이 로컬로 실행될 때 도메인 옵션이 있으면 도메인이 한 개뿐인 경우 외에는 필수 옵션입니다. 지정한 도메인 옵션은 명령이 원격으로 실행되는 경우에는 무시됩니다.

명령줄 호출 사용

명령줄을 다음 항목에 설명된 여러 방법으로 호출할 수 있습니다.

- 명령줄에서 asadmin 사용
- 파일 입력(스크립트)으로 asadmin 사용
- 표준 입력(파이프)으로 asadmin 사용

명령줄에서 asadmin 사용

명령을 사용하는 가장 간단한 방법은 명령줄에서 한 번에 한 개씩 사용하는 것입니다. 유틸리티에 명령, 옵션 및 인수를 입력합니다. 다중 모드에서는 유틸리티 이름과 환경 옵션(환경 변수를 설정한 경우)을 다시 입력할 필요 없이 여러 개의 명령을 입력할 수 있습니다. 단일 모드 또는 다중 모드 명령을 대화형으로 실행하거나 비대화형으로 실행할 수 있습니다. 대화형으로 실행하는 경우에는 암호 등 필요한 추가 입력을 하라는 메시지가 표시됩니다.

단일 모드 및 다중 모드에 대한 자세한 내용은 415페이지의 “단일 모드 및 다중 모드 사용”을 참조하십시오.

대화형으로 명령을 사용하는 자세한 내용은 417페이지의 “대화식 및 비대화식 옵션 사용”을 참조하십시오.

명령줄에서 사용 예

```
> asadmin create-instance --user admin --password password --host
austen --port 4848 --instanceport 1024 server2
```

명령을 완료한 후 운영 체제 프롬프트로 되돌아 옵니다.

파일 입력(스크립트)으로 asadmin 사용

사용자는 여러 `asadmin` 명령을 포함하는 스크립트를 작성할 수 있습니다. 사용자는 스크립트를 사용하여 명령을 일괄 처리하고 특정 시간에 작업이 실행되도록 설정할 수 있으며 또한 관리 작업을 단순화 및 자동화할 수 있습니다.

파일에 있는 스크립트를 호출하려면 다음 구문을 사용합니다.

```
> asadmin multimode --file filename
```

다음은 이러한 방식으로 호출할 수 있는 파일에 있는 간단한 스크립트의 예입니다.

```
# Create new instance and start it.
export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=mypassword
AS_ADMIN_HOST=austen AS_ADMIN_PORT=4848
create-instance --instanceport 9000 austen3
start-instance austen3
```

이 스크립트는 환경을 설정하고 `austen3`이라는 인스턴스를 만들며 새 인스턴스를 시작합니다. 숫자 기호(#)로 시작하는 행은 주석으로 취급되어 무시됩니다.

표준 입력(파이프)으로 asadmin 사용

다음 구문을 사용하여 `asadmin` 유틸리티에 파이프 입력할 수 있습니다.

```
cat filename | asadmin multimode
```

이 구문은 Windows에서는 작동하지 않을 수 있습니다.

이스케이프 문자 사용

콜론(:), 별표(*) 및 백슬래시(\) 등 일부 문자는 명령 구문에 사용할 때 이스케이프 문자를 함께 써야만 오류가 발생하는 것을 막을 수 있습니다. 이스케이프 문자 사용은 사용 플랫폼 및 단일 모드 또는 다중 모드 사용 여부에 따라 다릅니다.

주 `get` 명령과 `set` 명령에서는 콜론 이스케이프 문자를 사용할 필요가 없습니다.

이 절은 다음 내용으로 구성되어 있습니다.

- 단일 모드 UNIX상의 이스케이프 문자
- 단일 모드 Windows상의 이스케이프 문자
- 단일 모드 모든 플랫폼상의 이스케이프 문자
- 다중 모드 모든 플랫폼상의 이스케이프 문자

단일 모드 UNIX상의 이스케이프 문자

Solaris에서는 제한된 문자에 두 개의 백슬래시(\\) 또는 큰 따옴표(" ") 중 하나를 이스케이프 문자로 사용할 수 있습니다.

두 개의 백슬래시(\\)를 이스케이프 문자로 사용

예를 들어, 값에 콜론이 포함된 옵션을 사용하여 JDBC 연결 풀을 만들 때 백슬래시를 사용할 수 있습니다. 다음 예에서는 일부 등록 정보에 환경 변수가 설정된 것으로 가정합니다.

```
asadmin create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=jdbc\\:oracle\\:thin\\:@asperfsol8\\:1521\\:V8i:user=staging_lo
okup_app:password=staging_lookup_app OraclePoollookup
```

따옴표를 이스케이프 문자로 사용

위의 예에 따옴표(")를 사용하려면 해당 값을 큰 따옴표로 묶고 백슬래시와 함께 큰 따옴표를 이스케이프 문자로 사용합니다.

```
asadmin create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=\"jdbc:oracle:thin:@asperfsol8:1521:V8i\":user=staging_lookup_a
pp:password=staging_lookup_app OraclePoollookup
```

424페이지의 "단일 모드 모든 플랫폼상의 이스케이프 문자"에 설명된 방법을 사용할 수도 있습니다.

단일 모드 Windows상의 이스케이프 문자

Windows에서는 백슬래시 문자를 이스케이프 문자로 사용할 수 있습니다. 예를 들어, 값에 콜론이 포함된 옵션을 사용하여 JDBC 연결 풀을 만들 때 백슬래시를 사용할 수 있습니다. 다음 예에서는 일부 등록 정보에 환경 변수가 설정된 것으로 가정합니다.

```
asadmin create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url=jdbc\:oracle\:thin\:@asperfsol8\:1521\:V8i:user=staging_lookup_
app:password=staging_lookup_app OraclePoollookup
```

424페이지의 “단일 모드 모든 플랫폼상의 이스케이프 문자”에 설명된 방법을 사용할 수도 있습니다.

단일 모드 모든 플랫폼상의 이스케이프 문자

사용자는 모든 플랫폼에서 백슬래시를 이스케이프 문자로 사용하고 이스케이프 문자를 포함한 값을 큰 따옴표로 묶을 수 있습니다. 예를 들어, 값에 콜론이 포함된 옵션을 사용하여 JDBC 연결 풀을 만들 때 이스케이프 문자를 다음과 같이 사용할 수 있습니다. 다음 예에서는 일부 등록 정보에 환경 변수가 설정된 것으로 가정합니다

```
asadmin create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url="jdbc\:oracle\:thin\:@iasperfsol8\:1521\:V8i":user=staging_look
up_app:password=staging_lookup_app OraclePoollookup
```

다중 모드 모든 플랫폼상의 이스케이프 문자

다중 모드에서는 슬래시나 백슬래시 없이 따옴표만을 필요로 하는 다음 구문을 사용할 수 있습니다.

```
asadmin> create-jdbc-connection-pool  --instance server1
--datasourceclassname oracle.jdbc.pool.OracleDataSource
--failconnection=true --isconnectvalidatereq=true --property
url="jdbc:oracle:thin:@asperfsol8:1521:V8i":user=staging_lookup_app
:password=staging_lookup_app OraclePoollookup
```

get 및 set 명령 사용

`get` 명령과 `set` 명령을 사용하여 Sun ONE Application Server의 구성 설정에 액세스하고 변경할 수 있습니다. 대부분 `asadmin` 명령은 *필수* 등록 정보만 설정합니다. 선택적인 등록 정보의 값을 변경하려면 `set` 명령을 사용하십시오.

get 및 set 명령

명령	인수	사용
get	(scope) scope는 속성을 나타내며 유효한 이름입니다.	속성 값을 가져옵니다.
set	(scope=value) scope는 속성을 나타내며 유효한 이름입니다. *value*는 그 속성에 대해 설정한 값입니다.	속성의 값을 설정합니다.
reconfig	instance-name	구성 파일을 수정하는 명령을 실행한 후 reconfig 명령을 실행해야 변경 사항이 서버에 적용됩니다. 변경 사항/재구성 사항을 서버에 적용하는 데 대한 자세한 내용은 79페이지의 "응용 프로그램 서버 인스턴스에 변경 사항 적용"을 참조하십시오.

속성 사이에 공백을 사용하여 단일 명령으로 여러 개의 속성을 가져오거나 설정할 수 있습니다. 예:

```
set server1.appReloadPollInterval=20
server1.mime.mime1.file=mime.types
```

AS_ADMIN_PREFIX 환경 변수를 사용하여 후속 get 및 set 명령에 사용될 접두어를 설정할 수 있습니다. get 및 set 명령의 접두어 문자열과 피연산자 사이에는 마침표(".")가 삽입됩니다. 예:

```
asadmin>export AS_ADMIN_PREFIX=server1
asadmin>get *
server1.locale = en_US
server1.appReloadPollInterval = 2
server1.name = server1
...
```

get 명령과 set 명령에는 마침표 분리자가 필요하기 때문에 이름에 마침표가 포함된 항목에는 그 앞에 이스케이프 문자 백슬래시(\)를 사용해야 합니다. 다음 예에서는 server2.sun.com라는 마침표가 포함된 서버 인스턴스 이름 앞에 백슬래시가 사용된 것을 보여줍니다.

```
get server2\.sun\.com.*
```

백슬래시를 포함하지 않으면 오류 메시지가 표시됩니다.

get 및 set 명령 예

다음 예에서는 `get` 명령을 사용하여 속성의 값을 가져오는 방법과 `set` 명령을 사용하여 값을 설정하는 방법을 보여줍니다.

MDB 컨테이너 서비스 예

응용 프로그램 서버 인스턴스가 `server1`인 경우 다중 모드에서 환경 설정과 함께 다음 명령을 사용하여 모든 mdb-container 속성 값을 가져올 수 있습니다.

```
asadmin> get server1.mdb-container.*
```

다음은 해당 속성의 현재 값을 보여주는 이 명령의 출력 예입니다.

```
server1.mdb-container.logLevel = null
server1.mdb-container.steadyPoolSize = 10
server1.mdb-container.idleInPoolTimeoutInSeconds = 600
server1.mdb-container.maxPoolSize = 60
server1.mdb-container.monitoringEnabled = false
server1.mdb-container.poolResizeQuantity = 2
```

MDB 컨테이너 속성인 `monitoringEnabled`의 값만 가져오려면 다음 명령을 사용합니다.

```
asadmin> get server1.mdb-container.montioringEnabled
```

`monitoringEnabled` 속성의 값을 true로 설정하려면 다음 명령을 사용합니다.

```
asadmin> set server1.mdb-container.montioringEnabled=true
```

JMS 자원 예

모든 자원 구성에 대한 속성은 다음과 같은 형태여야 합니다.

instancename.resource.primary_key_value.attribute_name

예:

```
asadmin> get server1.jms-resource.myjms.*
```

`myjms`라는 JMS 대상 자원의 모든 속성을 가져옵니다. 예:

```
server1.jms-resource.myjms.resType = javax.jms.Topic
server1.jms-resource.myjms.enabled = true
server1.jms-resource.myjms.name = myjms
server1.jms-resource.myjms.description = null
```

단일 속성, 즉 `resType`과 같은 값을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jms-resource.myjms.resType
```

하나의 속성, 예를 들어 description을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.jms-resource.myjms.description=mydescription
```

이 예에서는 해당 설명을 mydescription으로 설정합니다.

여러 값 가져오기 및 설정하기의 예

하나의 명령을 사용하여 여러 개의 값을 가져오고 설정할 수 있습니다. 동시에 두 개의 속성을 설정하려면 공백을 사용하여 속성을 분리합니다. 예:

```
set server1.appReloadPollInterval=20
server1.mime.mime1.file=mime.types
```

또한 환경 변수 AS_ADMIN_PREFIX를 사용하여 여러 get 명령과 set 명령에 사용할 접두어를 설정할 수 있습니다.

get 및 set 명령을 사용한 모니터링

get 및 set 명령을 사용하여 실행 중인 서버를 모니터할 수도 있습니다. list 명령도 모니터링에 사용됩니다. monitor 옵션을 true 또는 false로 설정할 수 있습니다. 이 옵션을 true로 설정하면 지정된 속성을 모니터합니다. 명령줄 인터페이스를 사용한 Sun ONE Application Server 모니터에 대한 자세한 내용은 130페이지의 “CLI를 사용한 모니터링 데이터 추출”을 참조하십시오.

도움말 사용

모든 asadmin 명령에 대한 도움말은 명령 프롬프트에서 -h 또는 --help를 입력하여 볼 수 있습니다. 예를 들어, asadmin에 대한 도움말을 보려면 다음 작업을 수행합니다.

```
asadmin --help
```

모든 asadmin 명령의 목록이 표시됩니다.

특정 asadmin 명령에 대한 도움말을 보려면 다음 작업을 수행합니다.

```
asadmin command -h
```

또는

```
asadmin command --help
```

도움말에는 개요, 명령에 대한 설명, 구문 정보, 예 및 관련 명령의 목록이 포함됩니다.

명령의 어느 부분에서나 -h 또는 --help를 사용하면 그 명령에 대한 도움말을 볼 수 있습니다. 그런 경우 명령은 실행되지 않습니다.

또한 UNIX 환경에서 설명서 페이지의 명령줄 도움말 페이지에 액세스할 수 있습니다. 번들화되지 않은 설치의 경우에는 MANPATH 환경 변수에 *install_dir*/man을 추가합니다. 일단 이렇게 하면 명령 프롬프트에 `man asadmin`을 입력하여 Sun ONE Application Server 유틸리티의 설명서 페이지에 액세스할 수 있습니다.

출력 및 오류 보기

명령이 성공적으로 실행되면 수행된 내용을 알리는 메시지가 표시됩니다. 명령이 실패한 경우에는 오류 메시지가 표시됩니다.

이 절은 다음 내용으로 구성되어 있습니다.

- 종료 상태 보기
- 사용 보기

종료 상태 보기

오류 메시지가 표시되는 것 외에 `asadmin` 명령은 종료 시 항상 종료 상태를 표시합니다. 명령이 성공적이면 종료 상태는 0이고 실패하면 1입니다.

UNIX 상의 종료 상태

명령 프롬프트에 `echo $?`를 입력하여 종료 상태를 확인할 수 있습니다.

또한 스크립트에 종료 코드를 사용할 수도 있습니다. 예를 들어, 다음 Korn 셸 스크립트에서는 종료 상태를 사용하여 `list-instances` 명령이 성공적으로 실행되었는지 여부를 나타냅니다.

```
#!/bin/ksh
asadmin list-instances
if  [[ $? = 0 ]]
    then
        echo "success"
    else
        echo "error"
fi
```

Windows 상의 종료 상태

Windows에서는 .bat 스크립트에서 종료 상태를 확인할 수 있습니다. 예를 들어, 다음 두 스크립트 중 하나는 성공적인 스크립트와 그 출력 결과를 보여주고 다른 하나는 실패한 스크립트와 그 출력 결과를 보여줍니다.

성공 조건

```
myscript.bat
------------
echo off
echo Processing Command
call asadmin list-instances --domain domain1
if not %errorlevel% EQU 0  goto end
echo Command Succesful
goto program-end
:end
echo Command Failed
:program-end
```

출력:

```
 Processing Command
admin-server <not running>
server1 <not running>
 Command Succesful
```

오류 상황

```
myscript.bat
-------------
echo off
echo Processing Command
call asadmin list-instances
if not %errorlevel% EQU 0  goto end
echo Command Succesful
goto program-end
:end
echo Command Failed
:program-end
```

출력:

```
Processing Command
No default domain.  Need to enter a domain.
Command Failed
```

사용 보기

명령을 인수 없이 입력하면 명령 구문이 포함된 오류 메시지가 표시됩니다.
예:

```
asadmin> create-instance

Invalid number of operands received

USAGE: create-instance [--user admin_user] [--password
admin_password] [--host localhost] [--port 4848] [--sysuser
sys_user] [--domain domain_name] [--local=false] [--passwordfile
file_name] [secure | -s] --instanceport instanceport instancename
```

보안 고려 사항

명령줄에서 명령줄 인터페이스 실행 시에는 모든 명령과 함께 사용자 암호를 제공해야 합니다. 다중 모드로 실행 중인 경우에는 처음 환경 설정 시에 암호를 제공해야 합니다. 다중 모드를 종료한 경우에는 다중 모드를 다시 시작할 때 암호를 비롯한 환경을 다시 설정해야 합니다. 환경 명령을 사용하여 암호를 설정합니다. 자세한 내용은 417페이지의 “환경 명령 사용”을 참조하십시오.

암호 파일을 설정하여 명령줄에 암호를 입력하지 않을 수도 없습니다. 자세한 내용은 419페이지의 “암호 파일 옵션 사용”을 참조하십시오.

유효한 사용자 이름과 암호라는 인증 정보가 없으면 명령이 실행되지 않습니다.

명령줄 인터페이스는 Sun ONE Application Server에 설정한 보안 조치를 포함합니다. Sun ONE Application Server 보안에 대한 자세한 내용은 *Sun ONE Application Server Administrator's Security Guide*를 참조하십시오.

동시에 액세스할 때 고려해야 할 사항

두 사람 이상이 명령줄 인터페이스 및/또는 관리 인터페이스를 사용하여 동시에 서버를 구성하려 할 수 있습니다. 그런 경우 두 번째의 구성 요청은 첫번째 요청이 완료될 때까지 대기열 처리됩니다. 요청이 대기열에 너무 오래 있으면 시간 초과 처리됩니다.

일부 명령의 경우에는 `reconfig` 명령을 사용할 때까지 변경 사항이 적용되지 않습니다. 따라서 변경 사항이 서버에 적용되기 전에 두 사람 이상이 속성을 편집할 수 있습니다. `reconfig` 명령에 대한 자세한 내용은 79페이지의 "응용 프로그램 서버 인스턴스에 변경 사항 적용"을 참조하십시오.

명령 참조

이 절은 다음 내용으로 구성되어 있습니다.

- 명령 목록
- 점으로 구분된 이름 및 속성 목록
- 긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수 대응

명령 목록

다음 표에서는 모든 `asadmin` 명령과 그 용도에 대해 설명합니다. 명령 구문 및 그 사용법에 대한 자세한 내용은 온라인 도움말을 참조하십시오.

왼쪽 열은 명령 이름이고 오른쪽 열은 그 사용법입니다.

`asadmin` Commands

명령	사용
add-resources	한 개 이상의 jdbc, jms 또는 javamail 유형 자원 추가
create-acl	ACL (액세스 제어 목록) 만들기
create-authdb	인증 데이터베이스 만들기
create-auth-realm	인증 영역 만들기
create-custom-resource	사용자 정의 자원 만들기
create-domain	도메인 만들기
create-file-user	키 파일에 파일 영역 사용자 만들기
create-http-listener	HTTP Listener 만들기
create-http-qos	응용 프로그램 서버 인스턴스 또는 가상 서버를 위한 HTTP QOS(Quality of Service) 설정 만들기
create-iiop-listener	IIOP Listener 만들기
create-instance	응용 프로그램 서버 인스턴스 만들기

`asadmin` Commands

명령	사용
create-javamail-resource	JavaMail 자원 만들기
create-jdbc-connection-pool	JDBC 연결 풀 만들기
create-jdbc-resource	JDBC 자원 만들기
create-jmsdest	JMS (Java Message Service) 대상 만들기
create-jms-resource	JMS 자원 만들기
create-jndi-resource	JNDI 자원 만들기
create-jvm-options	java-config 또는 프로파일러 요소에 JVM 옵션 만들기
create-lifecycle-module	lifecycle 모듈 만들기
create-mime	MIME 유형 파일 만들기
create-persistence-resource	지속성 관리자 팩토리 자원 만들기
create-profiler	JVM 프로파일러 만들기
create-ssl	HTTP Listener, IIOP Listener 또는 IIOP 서비스의 SSL 설정 만들기
create-virtual-server	가상 서버 만들기
delete-acl	ACL 삭제
delete-authdb	인증 데이터베이스 삭제
delete-auth-realm	인증 영역 삭제
delete-custom-resource	사용자 정의 자원 삭제
delete-domain	도메인 삭제. 이 명령은 로컬에서만 실행할 수 있습니다.
delete-file-user	키 파일에서 파일 영역 사용자 삭제
delete-http-listener	HTTP Listener 삭제
delete-http-qos	응용 프로그램 서버 인스턴스 또는 가상 서버를 위한 HTTP QOS (Quality of Service) 설정 삭제
delete-iiop-listener	IIOP Listener 삭제
delete-instance	응용 프로그램 서버 인스턴스 삭제
delete-javamail-resource	JavaMail 자원 삭제
delete-jdbc-connection-pool	JDBC 연결 풀 삭제
delete-jdbc-resource	JDBC 자원 삭제

`asadmin` Commands

명령	사용
delete-jmsdest	JMS 대상 삭제
delete-jms-resource	JMS 자원 삭제
delete-jndi-resource	JNDI 자원 삭제
delete-jvm-options	java-config 또는 프로파일러 요소의 JVM 옵션 삭제
delete-lifecycle-module	lifecycle 모듈 삭제
delete-mime	MIME 유형 파일 삭제
delete-persistence-resource	지속성 관리자 팩토리 자원 삭제
delete-profiler	JVM 프로파일러 삭제
delete-ssl	HTTP Listener, IIOP Listener 또는 IIOP 서비스의 SSL 설정 삭제
delete-virtual server	가상 서버 삭제
deploy	EJB, WEB, 커넥터, 응용 프로그램 클라이언트 또는 응용 프로그램 구성 요소를 응용 프로그램 서버 인스턴스에 배포
deploydir	디렉토리에 있는 EJB, WEB, 커넥터, 응용 프로그램 클라이언트 또는 응용 프로그램 구성 요소를 응용 프로그램 서버 인스턴스에 배포
disable	응용 프로그램 서버 인스턴스의 배포된 구성 요소 비활성화
display-license	사용권 정보 표시. 이 명령은 로컬에서만 실행될 수 있습니다.
enable	응용 프로그램 서버 인스턴스의 배포된 구성 요소 활성화
export	asadmin 환경 변수의 값을 후속 `asadmin` 명령에서 사용하도록 내보내기
get	속성 값 가져오기
help	특정 명령에 대한 도움말(설명, 사용법, 구문, 예) 또는 `asadmin`에 대한 일반적인 도움말 표시
install-license	사용권 파일 설치. 이 명령은 로컬에서만 실행될 수 있습니다.
jms-ping	JMS 공급자 실행 여부 확인을 위한 Ping
list	구성 가능한 요소 나열
list-acls	응용 프로그램 서버 인스턴스의 ACL 나열
list-authdbs	인증 데이터베이스 나열

`asadmin` Commands

명령	사용
list-auth-realms	인증 영역 나열
list-components	서버 인스턴스의 배포된 구성 요소 나열
list-custom-resources	서버 인스턴스의 사용자 정의 자원 나열
list-domains	도메인 나열
list-file-users	서버 인스턴스의 모든 파일 영역 사용자 나열
list-file-groups	지정된 파일 영역 사용자의 모든 그룹 나열. 사용자를 지정하지 않으면 서버 인스턴스의 모든 그룹이 나열됩니다.
list-http-listeners	서버 인스턴스의 HTTP Listener 나열
list-instances	도메인에 있는 응용 프로그램 서버 인스턴스 나열
list-iiop-listeners	서버 인스턴스의 IIOP Listener 나열
list-javamail-resources	서버 인스턴스의 JavaMail 자원 나열
list-jdbc-connection-pools	서버 인스턴스의 JDBC 연결 풀 나열
list-jdbc-resources	서버 인스턴스의 JDBC 자원 나열
list-jmsdest	서버 인스턴스의 JMS 대상 나열
list-jms-resources	서버 인스턴스의 JMS 자원 나열
list-jndi-resources	서버 인스턴스의 JNDI 자원 나열
list-lifecycle-modules	서버 인스턴스의 lifecycle 모듈 나열
list-mimes	서버 인스턴스의 MIME 파일 나열
list-persistence-resources	서버 인스턴스의 지속성 관리자 팩토리 자원 나열
list-profilers	서버 인스턴스의 JVM 프로파일러 나열
list-sub-components	배포된 모듈 또는 배포된 응용 프로그램 모듈에 있는 한 개 이상의 EJB 또는 서블릿 나열
list-virtual-servers	서버 인스턴스의 가상 서버 나열
multimode	환경 설정을 유지하고 asadmin 내에 있으면서 여러 명령 실행
reconfig	변경 사항을 서버에 적용. 대부분의 변경 사항은 적용될 때까지는 반영되지 않습니다.
restart-instance	서버 인스턴스 다시 시작
set	속성 값 설정

`asadmin` Commands

명령	사용
show-component-status	배포된 구성 요소 상태 표시
show-instance-status	서버 인스턴스 상태 즉, 실행 여부 표시
shutdown	Administration Server 종료
start-appserv	Administration Server 및 모든 서버 인스턴스 시작. 이 명령은 로컬에서만 실행될 수 있습니다.
start-domain	도메인에 있는 모든 인스턴스 시작. 이 명령은 로컬에서만 실행될 수 있습니다.
start-instance	서버 인스턴스 시작
stop-appserv	Administration Server 및 모든 서버 인스턴스 중단. 이 명령은 로컬에서만 실행될 수 있습니다.
stop-domain	도메인에 있는 모든 인스턴스 중단
stop-instance	서버 인스턴스 중단
undeploy	서버 인스턴스에서 배포된 구성 요소 제거
unset	asadmin를 위해 내보낸 환경 변수 설정 해제
update-file-user	기존 파일 영역 사용자 갱신
version	Sun ONE Application Server의 버전 정보 표시

점으로 구분된 이름 및 속성 목록

`get` 및 `set` 명령을 사용하여 속성을 가져오고 설정하는 경우에는 `asadmin`에서 서비스, 자원 등에 사용하는 이름을 알아야 이 이름을 사용하여 특정 객체의 속성을 가져올 수 있습니다.

이러한 이름은 이름 사이에 마침표를 넣어 구분하는 구문을 사용하기 때문에 점으로 구분된 이름으로 불립니다.

asadmin에서 사용하는 점으로 구분된 이름

다음 표에서는 `asadmin`을 사용하여 항목을 구성하는 데 사용하는 이름을 나열합니다. 이 이름들은 다음과 같은 범주로 나뉩니다.

- 서비스 이름
- 자원 이름
- 응용 프로그램 이름
- 기타 이름

서비스 이름

다음 표에서는 서비스의 속성을 가져오고 설정하는 데 사용되는 서비스 이름을 보여줍니다.

명령줄 인터페이스의 서비스 이름

서비스	점으로 구분된 이름
JMS 서비스 구성	jms-service
트랜잭션 서비스 구성	transaction-service
MDB 컨테이너 구성	mdb-container
EJB 컨테이너 구성	ejb-container
웹 컨테이너 구성	web-container
JVM 구성	java-config
ORB 구성	orb 또는 iiop-service
ORB Listener 구성	orblistener 또는 iiop-listener orblistener 또는 iiop-listener는 유효한 이름이 아닙니다. 이 이름 뒤에 Listener의 이름이 나와야 합니다. 예: `ORB listener configuration orblistener.<listener name> or iiop-listener.<listener name>`
로그 구성	log-service
보안 구성	security-service
HTTP 구성	http-service

자원 이름

다음 표에서는 자원의 속성을 가져오고 설정하는 데 사용하는 자원 이름을 보여줍니다. 이 이름만으로는 유효하지 않으며, 이름 뒤에 자원 이름이 와야 합니다.

명령줄 인터페이스의 자원 이름

자원	점으로 구분된 이름
JDBC 자원 구성	jdbc-resource
JNDI 자원 구성	jndi-resource
JDBC 연결 풀 자원 구성	jdbc-connection-pool
사용자 정의 자원 구성	custom-resource
JMS 자원 구성	jms-resource
지속성 관리자 팩토리 자원 구성	persistence-manager-factory-resource
JavaMail 자원 구성	mail-resource

응용 프로그램 이름

다음 표에서는 응용 프로그램 관련 구성의 속성을 가져오고 설정하는 데 사용하는 점으로 구분된 이름을 보여줍니다. 이 이름만으로는 유효하지 않으며 이름 뒤에 응용 프로그램의 이름이 와야 합니다.

명령줄 인터페이스의 응용 프로그램 이름

응용 프로그램 구성 요소	점으로 구분된 이름
응용 프로그램 구성	application
EJB 모듈 구성	ejb-module
웹 모듈 구성	web-module
커넥터 모듈 구성	connector-module

기타 이름

다음 표에서는 `get` 및 `set` 명령을 사용하여 구성할 수 있는 기타 항목의 점으로 구분된 이름을 보여줍니다. 이 이름만으로는 유효하지 않으며, 이름 뒤에 자원의 이름이 와야 합니다(예: `http-listener.`*listener_name*, `lifecycle-module.`*module-name*).

명령줄 인터페이스의 기타 항목 이름

항목	점으로 구분된 이름
HTTP listener	http-listener 또는 http-server.http-listener
MIME 유형 파일	mime
ACL	acl
가상 서버	virtual-server
인증 데이터베이스	auth-db
보안 영역	authrealm
lifecycle 모듈	lifecycle-module
프로파일러 구성	프로파일러
서버 구성	서버 구성(서버 인스턴스 이름)

속성

다음 절에서는 위에 나열된 명명된 지정 항목의 속성과 그 사용 예를 설명합니다. 일부 속성은 읽기 전용으로 `get` 명령에만 사용 가능하고 `set` 명령에는 사용할 수 없습니다.

주 이 절에 포함된 예에서는 사용자, 암호, 호스트 및 포트가 환경 변수로 정의된 것으로 가정하며 구문으로 그 옵션을 나열하지 않습니다.

jms-service

다음 표에서는 왼쪽 열에 해당 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

JMS 서비스 속성

server.xml 이름	asadmin 이름
port	port
admin-username	adminUserName
admin-password	adminPassword
log-level	logLevel

JMS 서비스 속성

server.xml 이름	asadmin 이름
enabled	enabled
init-timeout-in-seconds	initTimeoutInSeconds
start-args	startArgs

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jms-service.*
```

`adminPassword`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jms-service.adminPassword
```

속성 `adminPassword`에 `admin`의 값을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.jms-service.adminPassword=admin
```

transaction-service

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

트랜잭션 서비스 속성

server.xml 이름	asadmin 이름
automatic-recovery	automaticTransactionRecovery
timeout-in-seconds	transactionRecoveryTimeout
tx-log-dir	transactionLogFile
heuristic-decision	heuristicDecision
keypoint-interval	keypointInterval
log-level	logLevel
monitoring-enabled	monitoringEnabled

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.transaction-service.*
```

transactionRecoveryTimeout라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.transaction-service.transactionRecoveryTimeout
```

속성 transactionRecoveryTimeout의 값을 49로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set
server1.transaction-service.transactionRecoveryTimeout=49
```

mdb-container

다음 표에서는 왼쪽 열에 속성의 server.xml 이름, 오른쪽 열에 asadmin에서 사용하는 이름을 보여줍니다.

MDB 컨테이너 속성

server.xml 이름	asadmin 이름
steady-pool-size	steadyPoolSize
pool-resize-quantity	poolResizeQuantity
max-pool-size	maxPoolSize
idle-timeout-in-seconds	idleInPoolTimeoutInSeconds
log-level	logLevel
monitoring-enabled	monitoringEnabled

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.mdb-container.*
```

steadyPoolSize라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.mdb-container.steadyPoolSize
```

속성 steadyPoolSize의 값을 10으로 다음 작업을 수행합니다.

```
asadmin> set server1.mdb-container.steadyPoolSize=10
```

ejb-container

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

EJB 컨테이너 속성

server.xml 이름	asadmin 이름
steady-pool-size	steadyPoolSize
pool-resize-quantity	poolResizeQuantity
max-pool-size	maxPoolSize
cache-resize-quantity	cacheResizeQuantity
max-cache-size	maxCacheSize
pool-idle-timeout-in-seconds	idleInPoolTimeoutInSeconds
cache-idle-timeout-in-seconds	idleInCacheTimeoutInSeconds
removal-timeout-in-seconds	removalTimeoutInSeconds
victim-selection-policy	victimSelectionPolicy
commit-option	commitOption
log-level	logLevel
monitoring-enabled	monitoringEnabled

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.ejb-container.*
```

`maxPoolSize`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.ejb-container.maxPoolSize
```

속성 `maxPoolSize`의 값을 12로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.ejb-container.maxPoolSize=12
```

web-container

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

웹 컨테이너 속성

server.xml 이름	asadmin 이름
log-level	logLevel
monitoring-enabled	monitoringEnabled (not used)

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.web-container.*
```

`logLevel`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.web-container.logLevel
```

`monitoringEnabled`라는 속성에 경고를 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.web-container.logLevel=WARNING
```

java-config

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

JVM 속성

server.xml 이름	asadmin 이름
java-home	javahome
debug-enabled	debugEnabled
debug-options	debugOptions
javac-options	javacoptions
rmic-options	rmicoptions
classpath-prefix	classpathprefix
server-classpath	serverClasspath
classpath-suffix	classpathsuffix
native-library-path-prefix	libpathprefix

JVM 속성

server.xml 이름	asadmin 이름
native-library-path-suffix	libpathsuffix
env-classpath-ignored	envpathignore

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.java-config.*
```

`classpathprefix`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.java-config.classpathprefix
```

속성 `classpathprefix`의 값을 `com.sun`으로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.java-config.classpathprefix=com.sun
```

orb 또는 iiop-service

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

ORB/IIOP 서비스 속성

server.xml 이름	asadmin 이름
message-fragment-size	msgSize
steady-thread-pool-size	minThreads
max-thread-pool-size	maxThreads
max-connections	maxConnections
idle-thread-timeout-in-seconds	idleThreadTimeout
log-level	log
monitoring-enabled	monitor
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls

ORB/IIOP 서비스 속성

server.xml 이름	asadmin 이름
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.orb.*
```

또는

```
asadmin> get server1.iiop-service.*
```

`msgSize`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.orb.msgSize
```

또는

```
asadmin> get server1.iiop-service.msgSize
```

`idleThreadTimeout`이라는 속성을 300으로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.orb.idleThreadTimeout=300
```

또는

```
asadmin> set server1.iiop-service.idleThreadTimeout=300
```

orblistener 또는 iiop-listener

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

IIOP Listener 속성

server.xml 이름	asadmin 이름
id	id
address	address
port	port
enabled	enabled
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers

IIOP Listener 속성

server.xml 이름	asadmin 이름
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
tls-rollback-enabled	tlsRollback
client-auth-enabled	clientAuth

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.orblistener.orb_listener_id.*
```

또는

```
asadmin> get server1.iiop-listener.orb_listener_id.*
```

`port`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.orblistener.orb_listener_id.port
```

또는

```
asadmin> get server1.iiop-listener.orb_listener_id.port
```

`address`라는 속성을 `bluestar`로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.orblistener.orb_listener_id.address=bluestar
```

또는

```
asadmin> set server1.iiop-listener.orb_listener_id.address=bluestar
```

log-service

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

로그 구성 속성

server.xml 이름	asadmin 이름
file	file
level	level
log-stdout	stdout
log-stderr	stderr

로그 구성 속성

server.xml 이름	asadmin 이름
echo-log-messages-to-stderr	echoToStderr
create-console	createConsole
log-virtual-server-id	LogVirtualServerId
use-system-logging	useSystemLogging

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.log-service.*
```

`level`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.log-service.level
```

`echoToStderr`라는 속성을 true로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.log-service.echoToStderr=true
```

security-service

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

보안 영역 구성 속성

server.xml 이름	asadmin 이름
default-realm	defaultRealm
default-principal	defaultPrinicpal
default-principal-password	defaultPrinicpalPassword
anonymous-role	anonymousRole
audit-enabled	auditEnabled
log-level	logLevel

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.security-service.*
```

anonymousRole이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.security-service.anonymousRole
```

encryptPasswords라는 속성을 true로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.security-service.auditEnabled=true
```

http-service

다음 표에서는 왼쪽 열에 속성의 server.xml 이름, 오른쪽 열에 asadmin에서 사용하는 이름을 보여줍니다.

HTTP 서비스 속성

server.xml 이름	asadmin 이름
qos-metrics-interval-in-seconds	qos-metrics-interval-in-seconds
qos-recompute-time-interval-in-millis	qos-recompute-time-interval-in-millis
qos-enabled	qos-enabled
bandwidth-limit	bandwidthLimit
enforce-bandwidth-limit	enforceBandwidthLimit
connection-limit	connectionLimit
enforce-connection-limit	enforceConnectionLimit

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.http-service.*
```

bandwidthLimit이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.http-service.bandwidthLimit
```

qos-enabled라는 속성을 true로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.http-service.qos-enabled=true
```

jdbc-resource

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

JDBC 자원 속성

server.xml 이름	asadmin 이름
jndi-name	name
pool-name	pool
enabled	enabled
description	description

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jdbc-resource.jdbc_resource_name.*
```

`pool`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jdbc-resource.jdbc_resource_name.pool
```

`enabled`라는 속성을 true로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.jdbc-resource.jdbc_resource_name.enabled=true
```

jndi-resource

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

JNDI 자원 속성

server.xml 이름	asadmin 이름
jndi-name	name
jndi-lookup-name	LookupName
res-type	resType
factory-class	factory
enabled	enabled
description	description

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jndi-resource.jndi_name.*
```

`factory`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jndi-resource.jndi_name.factory
```

`factory`라는 속성을 `com.sun`으로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.jndi-resource.jndi_name.factory=com.sun
```

jdbc-connection-pool

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

JDBC 연결 풀 속성

server.xml 이름	asadmin 이름
name	name
datasource-classname	dsClassName
res-type	resType
description	description
steady-pool-size	steadyPoolSize
max-pool-size	maxPoolSize
max-wait-time-in-millis	maxWaitTime
pool-resize-quantity	resizeValue
idle-timeout-in-seconds	idleTimeout
transaction-isolation-level	transactionIsolationLevel
is-isolation-level-guaranteed	isIsolationLevelGuaranteed
connection-validation-method	validationMethod
is-connection-validation-required	isValidationRequired
fail-all-connections	failAll
validation-table-name	validationTable

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jdbc-connection-pool.pool_name.*
```

dsClassName이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jdbc-connection-pool.pool_name.dsClassName
```

resizeValue라는 속성을 2로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.jdbc-connection-pool.pool_name.resizeValue=2
```

custom-resource

다음 표에서는 왼쪽 열에 속성의 server.xml 이름, 오른쪽 열에 asadmin에서 사용하는 이름을 보여줍니다.

사용자 정의 자원 속성

server.xml 이름	asadmin 이름
jndi-name	name
res-type	resType
factory-class	factory
enabled	enabled
description	description

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.custom-resource.jndi_name.*
```

factory라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.custom-resource.jndi_name.factory
```

factory라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.custom-resource.jndi_name.factory=myclass
```

jms-resource

다음 표에서는 왼쪽 열에 속성의 server.xml 이름, 오른쪽 열에 asadmin에서 사용하는 이름을 보여줍니다.

JMS 자원 속성

server.xml 이름	asadmin 이름
jndi-name	name
res-type	resType

JMS 자원 속성

server.xml 이름	asadmin 이름
enabled	enabled
description	description

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jms-resource.jms_resource_name.*
```

res-type이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.jms-resource.jms_resource_name.resType
```

enabled라는 속성을 true로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.jms-resource.jms_resource_name.enabled=true
```

persistence-manager-factory-resource

다음 표에서는 왼쪽 열에 속성의 server.xml 이름, 오른쪽 열에 asadmin에서 사용하는 이름을 보여줍니다.

지속성 관리자 팩토리 자원 속성

server.xml 이름	asadmin 이름
jndi-name	jndiName
jdbc-resource-jndi-name	JdbcResourceJndiName
factory-class	factoryClass
enabled	enabled
description	description

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.persistence-manager-factory-resource.jndi_name
```

factoryClass라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get
server1.persistence-manager-factory-resource.jndi_name.factoryClass
```

enabled라는 속성을 true로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set
server1.persistence-manager-factory-resource.jndi_name.enabled=true
```

mail-resource

다음 표에서는 왼쪽 열에 속성의 server.xml 이름, 오른쪽 열에 asadmin에서 사용하는 이름을 보여줍니다.

JavaMail 자원 속성

server.xml 이름	asadmin 이름
jndi-name	name
enabled	enabled
store-protocol	storeProtocol
store-protocol-class	storeProtocolClass
transport-protocol	transportProtocol
transport-protocol-class	transportProtocolClass
host	host
user	user
from	from
debug	debug
description	description

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.mail-resource.jndi_name.*
```

host라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.mail-resource.jndi_name.host
```

enabled라는 속성을 true로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.mail-resource.jndi_name.enabled=true
```

application

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

응용 프로그램 속성

server.xml 이름	asadmin 이름
name	name
location	location
virtual-servers	virtualServers
description	description
enabled	enabled

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.application.application_name.*
```

응용 프로그램에 있는 속성 `location`을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.application.application_name.location
```

`location`이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.application.application_name.location=
"/export/home/as7se/as1/repository/applications/ASConverter"
```

ejb-module

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

EJB 모듈 속성

server.xml 이름	asadmin 이름
name	name
location	location
description	description
enabled	enabled

인스턴스(server1)에 있는 독립 실행형 EJB 모듈의 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.ejb-module.ejb_jar_name.*
```

인스턴스(server1)에 대한 응용 프로그램의 한 EJB 모듈의 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get
server1.j2ee-application.application_name.ejb-module.ejb_jar_name.*
```

또는

```
asadmin>get server1.application.application_name.ejb-module.ejb_jar_name.*
```

독립 실행형 EJB 모듈에서 location이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.ejb-module.ejb_jar_name.location
```

응용 프로그램의 EJB 모듈에서 location이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get
server1.j2ee-application.application_name.ejb-module.ejb_jar_name.
location
```

또는

```
asadmin> get
server1.application.application_name.ejb-module.ejb_jar_name.location
```

독립 실행형 EJB 모듈에서 location이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set
server1.ejb-module.ejb_jar_name.location="/export/home/as7se/as1/repos
itory/modules/ejb_jar_name"
```

응용 프로그램에 번들된 EJB 모듈에서 location이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set
server1.j2ee-application.application_name.ejb-module.ejb_jar_name.
location="/export/home/as7se/as1/repository/modules/ejb_jar_name"
```

또는

```
asadmin>set
server1.application.application_name.ejb-module.ejb_jar_name.location="/ex
port/home/as7se/as1/repository/modules/ejb_jar_name"
```

web-module

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

WEB 모듈 속성

server.xml 이름	asadmin 이름
name	name
location	location
context-root	contextRoot
virtual-servers	virtualServers
description	description
enabled	enabled

인스턴스(server1)에 있는 독립 실행형 웹 모듈의 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.web-module.web_war_name.*
```

인스턴스(server1)의 응용 프로그램에 있는 웹 모듈의 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.web-module.application_name.web_war_name.*
```

독립 실행형 웹 모듈에서 `location`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.web-module.web_war_name.location
```

응용 프로그램에 있는 웹 모듈에서 `location`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.web-module.application_name.web_war_name.location
```

독립 실행형 모듈에 있는 `location`이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.web-module.war-ic.location=
"/export/home/as7se/as1/repository/modules/web_war_name"
```

응용 프로그램에 번들화된 웹 모듈에서 `location`이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.web-module.application_name.web_war_name.location=
"/export/home/as7se/as1/repository/modules/web_war_name"
```

connector-module

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

커넥터 모듈 속성

server.xml 이름	asadmin 이름
name	name
location	location
description	description
enabled	enabled

인스턴스(server1)에 있는 독립 실행형 커넥터 모듈의 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.connector-module.connector_rar_name.*
```

독립 실행형 커넥터 모듈에서 `location`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.connector-module.connector_rar_name.location
```

독립 실행형 커넥터 모듈에 있는 `location`이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.connector-module.connector_rar_name.location=
"/export/home/as7se/as1/repository/modules/connector_rar_name"
```

http-listener 또는 http-server.http-listener

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

HTTP Listener 속성

server.xml 이름	asadmin 이름
id	id
address	address
port	port
family	family
acceptor-threads	acceptorThreads
blocking-enabled	blockingEnabled
security-enabled	securityEnabled

HTTP Listener 속성

server.xml 이름	asadmin 이름
default-virtual-server	defaultVirtualServer
server-name	serverName
enabled	enabled
cert-nickname	cert
ssl2-enabled	ssl2
ssl2-ciphers	ssl2Ciphers
ssl3-enabled	ssl3
ssl3-tls-ciphers	ssl3Ciphers
tls-enabled	tls
	tlsRollback
client-auth-enabled	clientAuth

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.http-listener.http_listener_name.*
```

또는

```
asadmin> get server1.http-server.http-listener.http_listener_name.*
```

`factory`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.httpl-listener.http_listener_name.address
```

또는

```
asadmin> get server1.http-server.http-listener.http_listener_이름.address
```

`address`라는 속성을 IP 주소 `0.0.0.0`으로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.http-listener.http_listener_name.address=0.0.0.0
```

또는

```
asadmin> set
server1.http-server.http-listener.http_listener_name.address=0.0.0.0
```

mime

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

MIME 유형 속성

server.xml 이름	asadmin 이름
id	id
file	file

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.mime.mime_name.*
```

`file`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.mime.mime_name.file
```

`file`이라는 속성을 `mime.types`로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.mime.mime_name.file=mime.types
```

acl

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

ACL 속성

server.xml 이름	asadmin 이름
id	id
file	file

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.acl.acl_name.*
```

`file`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.acl.acl_name.file
```

`file`이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.acl.acl_name.file=com/as1.acl
```

virtual-server

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

가상 서버 속성

server.xml 이름	asadmin 이름
id	id
http-listeners	httpListeners
config-file	configFile
default-object	defaultObject
accept-language	acceptLanguage
log-file	logFile
default-web-module	defaultWebModule
hosts	hosts
mime	mime
state	state
acls	acls
bandwidth-limit	bandwidthLimit
enforce-bandwidth-limit	enforceBandwidthLimit
connection-limit	connectionLimit
enforce-connection-limit	enforceConnectionLimit
property name="dir" value=	property.dir
property name="nice" value=	property.nice
property name="user" value=	property.user
property name="group" value=	property.group
property name="chroot" value=	property.chroot
property name="docroot" value=	property.docroot
property name="accesslog" value=	property.accesslog

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get instance_name.virtual-server..virtual-server..*
```

예:

```
asadmin> get server1.virtual-server.server1.*
```

가상 서버 server1의 `httpListeners`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.virtual-server.server1.httpListeners
```

`acceptLanguage`라는 속성을 false로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.virtual-acceptLanguage=false
```

auth-db

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

인증 데이터베이스 속성

server.xml 이름	asadmin 이름
id	id
database	database
basedn	basedn
certmaps	certmaps

인스턴스에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get instancename.virtual-server.vserver_id.auth-db.authdb_id.*
```

예를 들어, 인스턴스 server1의 가상 서버 server1을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.virtual-server.server1.auth-db.authdb_id.*
```

`database`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.virtual-server.server1.auth-db.authdb_id.database
```

`database`라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set
server1.virtual-server.server1.auth-db.authdb_id.database=Oracle
```

authrealm

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

인증 영역 속성

server.xml 이름	asadmin 이름
name	name
classname	classname

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.authrealm.authrealm_id.*
```

`classname`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.authrealm.authrealm_id.classname
```

`classname`이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set
server1.authrealm.authrealm_id.classname=com.sun.as.security.auth.realm.sharedpassword.SharedPasswordRealm
```

lifecycle-module

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

lifecycle 모듈 속성

server.xml 이름	asadmin 이름
name	name
enabled	enabled
class-name	className
classpath	classPath
load-order	loadOrder
is-failure-fatal	isFailureFatal
description	description

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.lifecycle-module.lifecycle_module_id.*
```

lifecycle 모듈의 `className`이라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.lifecycle-module.lifecycle_module_id.className
```

`className`이라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set
server1.lifecycle-module.lifecycle_module_id.className=com.lifecycle_module_id.
lifecycle
```

프로파일러

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

JVM 프로파일러 구성 속성

server.xml 이름	asadmin 이름
name	name
classpath	classPath
native-library-path	nativeLibraryPath
enabled	enabled

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.profiler.*
```

`enabled`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.profiler.enabled
```

`enabled`라는 속성을 false로 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.profiler.enabled=false
```

서버 구성(서버 인스턴스 이름)

다음 표에서는 왼쪽 열에 속성의 `server.xml` 이름, 오른쪽 열에 `asadmin`에서 사용하는 이름을 보여줍니다.

서버 구성 속성

server.xml 이름	asadmin 이름
instance-name	name
locale	locale
log-root	logRoot
session-store	sessionStore
application-root	applicationRoot
dynamic-reload-enabled	appDynamicReloadEnabled
dynamic-reload-poll-interval-in-seconds	appReloadPollInterval

인스턴스(server1)에서 모든 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.*
```

`logRoot`라는 속성을 가져오려면 다음 작업을 수행합니다.

```
asadmin> get server1.logRoot
```

`logRoot`라는 속성을 설정하려면 다음 작업을 수행합니다.

```
asadmin> set server1.logRoot="/space/log"
```

긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수 대응

다음 표는 명령줄 옵션의 긴 형식과 짧은 형식의 목록입니다. 짧은 형식이 나열되지 않은 경우에는 해당 옵션에 짧은 형식이 없는 것입니다.

긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수

옵션 이름	긴 형식	짧은 형식	기본값	환경 변수
acceptlang	--acceptlang			AS_ADMIN_ACCEPT_
acceptorthreads	--acceptorthreads			AS_ADMIN_ACCEPTOR_THREADS
acls	--acls			AS_ADMIN_ACLS
address	--address			AS_ADMIN_ADDRESS
adminpassword	--adminpassword			AS_ADMIN_ADMINPASSWD
adminport	--adminport		4848	AS_ADMIN_ADMINPORT
adminuser	--adminuser			AS_ADMIN_ADMINUSER
basedn	--basedn			AS_ADMIN_BASEDN
blockingenabled	--blockingenabled			AS_ADMIN_BLOCKINGENABLED
bwlimit	--bwlimit			AS_ADMIN_BWLIMIT
certmaps	--certmaps			AS_ADMIN_CERTMAPS
certname	--certname			AS_ADMIN_CERTNAME
classname	--classname			AS_ADMIN_CLASSNAME
classpath	--classpath			AS_ADMIN_CLASSPATH
clientauthenabled	--clientauthenabled			AS_ADMIN_CLIENTAUTHENABLED
configfile	--configfile			AS_ADMIN_CONFIGFILE
connectionpoolid	--connectionpoolid			AS_ADMIN_CONNECTIONPOOLID
connlimit	--connlimit			AS_ADMIN_CONNLIMIT
contextroot	--contextroot			AS_ADMIN_CONTEXTROOT
database	--database			AS_ADMIN_DATABASE
debug	--debug		false	AS_ADMIN_DEBUG

긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수

옵션 이름	긴 형식	짧은 형식	기본값	환경 변수
defaultobj	--defaultobj			AS_ADMIN_DEFAULTOBJ
defaultwebmodule	--defaultwebmodule			AS_ADMIN_DEFAULTWEBMODULE
description	--description			AS_ADMIN_DESCRIPTION
discardmanualchanges	--discardmanualchanges	-d	false	AS_ADMIN_DISCARDMANUALCHANGES
echo	--echo	-e	false	AS_ADMIN_ECHO
enabled	--enabled			AS_ADMIN_ENABLED
enforcebwlimit	--enforcebwlimit			AS_ADMIN_ENFORCEBWLIMIT
enforceconnlimit	--enforceconnlimit			AS_ADMIN_ENFORCECONNLIMIT
failconnection	--failconnection		false	AS_ADMIN_FAILCONNECTION
failurefatal	--failurefatal		false	AS_ADMIN_FAILUREFATAL
family	--family			AS_ADMIN_FAMILY
file	--file	-f		AS_ADMIN_FILE
force	--force	-F	true	AS_ADMIN_FORCE
help	--help	-h		AS_ADMIN_HELP
host	--host	-H		AS_ADMIN_HOST
hosts	--hosts			AS_ADMIN_HOSTS
httplistenerid	--httplistenerid			AS_ADMIN_HTTPLISTENERID
httplisteners	--httplisteners			AS_HTTP_LISTENERS
idletimeout	--idletimeout		300	AS_ADMIN_IDLETIMEOUT
instance	--instance	-i	server1	AS_ADMIN_INSTANCE
instanceport	--instanceport			AS_ADMIN_INSTANCEPORT
interactive	--interactive	-I	true	AS_AMDIN_INTERACTIVE
isconnectvalidaterequired	--isconnectvalidaterequired		false	AS_ADMIN_ISCONNECTVALIDATEREQUIRED
jdbcjndiname	--jdbcjndiname	-a		AS_ADMIN_JDBCJNDINAME
jndilookupname	--jndilookupname	-l		AS_ADMIN_JNDILOOKUPNAME

긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수

옵션 이름	긴 형식	짧은 형식	기본값	환경 변수
keepmanualchanges	--keepmanualchanges	-k	false	AS_ADMIN_KEEPMANUALCHANGES
loadorder	--loadorder			AS_ADMIN_LOADORDER
local	--local	-l	false	
logfile	--logfile			AS_ADMIN_LOGFILE
maxpoolsize	--maxpoolsize		32	AS_ADMIN_MAXPOOLSIZE
maxwait	--maxwait		6000	AS_ADMIN_MAXWAIT
mime	--mime			AS_ADMIN_MIME
mimefile	--mimefile			AS_ADMIN_MIMEFILE
monitor	--monitor	-m	false	AS_ADMIN_MONITOR
name	--name	-n		AS_ADMIN_NAME
nativelibpath	--nativelibpath			AS_ADMIN_NATIVELIBPATH
objtype	--objtype	-o		AS_ADMIN_OBJTYPE
password	--password	-w		AS_ADMIN_PASSWORD
poolresize	--poolresize		2	AS_ADMIN_POOLRESIZE
port	--port	-p	8000	AS_ADMIN_PORT
prefix	--prefix	-x		AS_ADMIN_PREFIX
printprompt	--printprompt	-P	true	AS_ADMIN_PROMPT
property	--property			AS_ADMIN_PROPERTY
securityenabled	--securityenabled			AS_ADMIN_SECURITYENABLED
servername	--servername			AS_ADMIN_SERVERNAME
ssl2ciphers	--ssl2ciphers			AS_ADMIN_SSL2CIPHERS
ssl2enabled	--ssl2enabled			AS_ADMIN_SSL2ENABLED
ssl3enabled	--ssl3enabled			AS_ADMIN_SSL3ENABLED
ssl3tlsciphers	--ssl3tlsciphers			AS_ADMIN_SSL3TLSCIPHERS
state	--state			AS_ADMIN_STATE
steadypoolsize	--steadypoolsize	8		AS_ADMIN_STEADYPOOLSIZE
storeprotocol	--storeprotocol			AS_ADMIN_STOREPROTOCOL

긴 옵션 및 짧은 옵션 형식, 기본값 및 환경 변수

옵션 이름	긴 형식	짧은 형식	기본값	환경 변수
storeprotocolclass	--storeprotocolclass			AS_ADMIN_STOREPROTOCOLCLASS
tlsenabled	--tlsenabled			AS_ADMIN_TLSENABLED
tlsrollbackenabled	--tlsrollbackenabled			AS_ADMIN_TLSROLLBACKENABLED
transprotocol	--transprotocol		smtp	AS_ADMIN_TRANSPROTOCOL
type	--type			S_ADMIN_TRANSPROTOCOLCLASS
upload	--upload	-U	true	AS_ADMIN_TYPE
url	--url			AS_ADMIN_URL
user	--user	-u		AS_ADMIN_USER
validationmethod	--validationmethod		auto-commit	AS_ADMIN_VALIDATIONMETHOD
validationtable	--validationtable			AS_ADMIN_VALIDATIONTABLE
version	--version	-v		AS_AMDIN_VERSION
virtualserver	--virtualserver			AS_ADMIN_VIRTUALSERVER

타사 저작권 고지

본 제품에는 RSA Security, Inc.의 사용권 허가를 받은 코드가 포함되어 있습니다.

본 제품의 일부는 ANTLR을 사용하여 개발되었습니다. ANTLR 1989-2000은 jGuru.com(http://www.ANTLR.org 및 http://www.jGuru.com)에서 개발했습니다.

이 제품에는 Sun Public License에 따라 Netbeans Project (http://www.netbeans.org)를 통해 개발된 소프트웨어가 포함되어 있습니다. 사용 가능한 경우 이러한 소프트웨어는 www.netbeans.org에서 찾을 수 있습니다.

이 제품에는 Perl이 포함됩니다. 사용 가능한 경우 Perl은 http://public.ActiveState.com/gsar/APC/에서 찾을 수 있습니다.

이 제품에는 Exolab Project (http://www.exolab.org)를 통해 개발된 소프트웨어가 포함되어 있습니다.

이 제품에는 DOM4J Project (http://dom4j.org/)를 통해 개발된 소프트웨어가 포함되어 있습니다.

이 제품에는 Apache Foundation에서 개발한 소프트웨어가 포함되어 있습니다. Copyright (c) 1999-2001 The Apache Software Foundation. All rights reserved.

이 제품에는 Regents of University of California에서 개발한 소프트웨어가 포함되어 있습니다. Copyright (c) 1991, 1993 The Regents of University of California. All rights reserved.

이 제품에는 International Business Machines Corporation에서 개발한 소프트웨어가 포함되어 있습니다. Copyright (c) 1995-2001 International Business Machines Corporation and others. All rights reserved. IBM 코드는 ICU 사용권에 따라 사용되었습니다. 다음을 참조하십시오.

ICU 사용 권 - ICU 1.8.1 이상

용어집

이 용어집에서는 Sun ONE Application Server 배포 및 배포 환경을 설명하는 데 사용되는 용어에 대해 정의합니다. 표준 J2EE 용어의 용어집은 다음 주소의 J2EE 용어집을 참조하십시오.

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

액세스 제어 누가, 어떤 대상이 액세스할 수 있는지를 제어하여 Sun ONE Application Server의 보안을 유지하는 방법입니다.

ACL Access Control List (액세스 제어 목록). ACL은 Sun ONE Application Server에 저장된 자원에 액세스할 수 있는 사용자를 식별하는 목록이 포함된 텍스트 파일입니다. 일반 ACL도 *참조*하십시오.

활성화 Enterprise Bean의 상태를 보조 저장 장치에서 메모리로 전송하는 프로세스입니다.

관리 인터페이스 Sun ONE Application Server를 구성 및 관리할 때 사용하는 브라우저 기반 양식의 집합입니다. CLI도 *참조*하십시오.

Administration Server 배포, 브라우저 기반 관리 및 CLI (명령줄 인터페이스)와 IDE (Integrated Development Environment)에서의 액세스를 포함하여 Sun ONE Application Server의 관리 기능을 제공하는 전용 응용 프로그램 서버 인스턴스입니다.

관리 도메인 Sun ONE Application Server에는 여러 관리 도메인이 있어서 여러 관리 사용자가 각각의 도메인을 만들어 관리할 수 있습니다. 도메인은 단일 시스템에 설치된 공통 이진 세트를 사용하여 작성된 인스턴스 세트입니다.

API Applications Program Interface의 약자. 컴퓨터 프로그램이 해당 API를 해석하도록 디자인된 다른 소프트웨어나 하드웨어와 통신할 때 사용할 수 있는 명령어 집합입니다.

애플릿 웹 브라우저에서 실행되는 Java로 작성된 작은 응용 프로그램입니다. 일반적으로 애플릿은 웹 페이지에서 호출되거나 웹 페이지에 삽입되어 특별한 기능을 제공합니다. 이와 반대로 *서블릿*은 서버에서 실행되는 작은 응용 프로그램입니다.

응용 프로그램 J2EE 응용 프로그램 배포 설명자로 `.ear` 파일에 패키지된 구성 요소 그룹입니다. 구성 요소, 모듈도 *참조*하십시오.

응용 프로그램 클라이언트 컨테이너 컨테이너를 *참조*하십시오.

응용 프로그램 서버 비즈니스 응용 프로그램이 실행되는 안정적이고, 안전하며, 확장 가능한 소프트웨어 플랫폼입니다. 응용 프로그램 서버는 일반적으로 구성 요소 lifecycle, 위치, 분산 및 트랜잭션 자원 액세스 등과 같은 높은 수준의 서비스를 응용 프로그램에 제공합니다.

응용 프로그램 계층 J2EE 응용 프로그램의 개념적 분할입니다. *클라이언트 계층*: 사용자 인터페이스(UI)입니다. 최종 사용자는 클라이언트 소프트웨어(예: 웹 브라우저)와 상호 작용하여 응용 프로그램을 사용합니다. *서버 계층*: 응용 프로그램의 구성 요소에 정의되어 있는, 응용 프로그램을 구성하는 비즈니스 논리 및 표현 논리입니다. *데이터 계층*: 응용 프로그램이 데이터 소스와 상호 작용할 수 있도록 하는 데이터 액세스 논리입니다.

어셈블리 응용 프로그램의 다양한 구성 요소를 배포 가능한 하나의 단위로 결합하는 프로세스입니다. 배포도 *참조*하십시오.

비동기 통신 메시지 전송자가 전송 메시지 반환을 기다리지 않고도 다른 작업을 계속할 수 있는 통신 모드입니다.

속성 서블릿으로 설정할 수 있는 요청 객체의 이름-값 쌍입니다. XML 파일의 요소를 수정하는 이름-값 쌍이기도 합니다. *매개 변수*와 반대입니다. 더 일반적으로 보면 속성은 메타데이터의 단위입니다.

감사 나중에 검사할 수 있도록 중요 이벤트를 기록하는 방법으로, 일반적으로 오류나 보안 침해 상태를 기록합니다.

인증 엔티티(예: 사용자)가 특정한 신원(사용자의 보안 신원)으로 동작하고 있음을 다른 엔티티(예: 응용 프로그램)에게 입증하는 프로세스입니다. Sun ONE Application Server에서는 기본적인, 양식 기반의 SSL 상호 인증을 지원합니다. 클라이언트 인증, 다이제스트 인증, 호스트 IP 인증, 플러그형 인증도 참조하십시오.

인증 메소드나 자원에 대한 액세스가 결정되는 프로세스입니다. J2EE 플랫폼의 인증은 인증을 통한 요청에 연관된 사용자가 해당 보안 규칙에 해당하는지 여부에 따라 달라집니다. 예를 들어, 인사 관리 응용 프로그램은 관리자에게 모든 직원의 사원 정보를 볼 수 있도록 허용하지만 직원들에게는 자신의 개인 정보만 볼 수 있도록 허용할 수 있습니다.

백업 저장소 데이터 리포지토리이며 일반적으로 파일 시스템 또는 데이터베이스입니다. 백그라운드 스레드(또는 스위퍼 스레드)에서 백업 저장소를 모니터링하여 원하지 않는 항목을 제거할 수 있습니다.

Bean 관리 지속성(BMP) Entity Bean의 변수 및 데이터 저장소 사이의 데이터 전송입니다. 데이터 액세스 논리는 일반적으로 개발자가 JDBC (Java Database Connectivity) 또는 기타 데이터 액세스 기술에 의해 제공됩니다. 컨테이너 관리 지속성(CMP)도 *참조*하십시오.

Bean 관리 트랜잭션(BMT) Enterprise Bean의 트랜잭션 경계가 개발자에 의해 프로그래밍 방식으로 제어됩니다. 컨테이너 관리 트랜잭션(CMT)도 *참조*하십시오.

BLOB Binary Large OBject의 약자입니다. 복잡한 객체 필드를 저장 및 검색하는 데 사용하는 데이터 유형입니다. BLOB는 대용량 바이트 배열로 변환된 다음 컨테이너가 관리하는 지속성 필드로 일련화되는 그림 등의 이진 또는 일련화 가능 객체입니다.

BMP Bean 관리 지속성(BMP)을 *참조*하십시오.

BMT Bean 관리 트랜잭션(BMT)을 *참조*하십시오.

브로커 JMS 메시지 라우팅, 전달, 지속성, 보안 및 로깅을 관리하고 관리자가 성능과 자원 사용을 모니터링 및 조정할 수 있는 인터페이스를 제공하는 Sun ONE Message Queue 엔티티입니다.

비즈니스 논리 데이터 통합이나 표현 논리가 아니라 응용 프로그램의 필수 비즈니스 규칙을 구현하는 코드입니다.

CA 인증 기관(CA) 또는 커넥터 구조(CA)를 참조하십시오.

캐시된 행 집합 `CachedRowSet` 객체를 사용하면 데이터 소스에서 데이터를 검색한 다음 데이터를 검사하고 수정하면서 데이터 소스에서 분리시킬 수 있습니다. 캐시된 행 집합은 검색된 원본 데이터 및 응용 프로그램의 데이터 변경 사항을 모두 추적할 수 있습니다. 응용 프로그램이 원본 데이터 소스를 업데이트하려고 하면 행 집합이 데이터 소스로 다시 연결되며 변경된 행만 다시 데이터베이스로 병합됩니다.

캐시 제어 지시문 Sun ONE Application Server는 캐시 제어 지시문을 사용하여 프록시 서버가 캐시할 정보를 제어합니다. 캐시 제어 지시문을 사용하면 프록시의 기본 캐시를 무시하여 민감한 정보가 캐시되어 검색되는 것을 방지할 수 있습니다. 이러한 지시문이 작동하려면 프록시 서버가 HTTP 1.1과 호환되어야 합니다.

호출 가능 문 저장 프로시저에서의 결과 집합 반환을 지원하는 데이터베이스에 대한 데이터베이스 프로시저 또는 함수 호출을 캡슐화하는 클래스입니다.

인증서 개인, 회사 또는 기타 엔티티의 이름을 지정하며 해당 엔티티에 속한 인증서에 포함되어 있는 공개 키를 인증하는 디지털 데이터입니다. 클라이언트와 서버 모두에 인증서가 있을 수 있습니다.

인증 기관(CA) 인터넷을 통해 인증서를 판매하는 회사 또는 회사의 인트라넷이나 익스트라넷을 위한 인증서를 발급하는 부서입니다.

암호화 암호화 또는 암호 해독에 사용되는 암호화 알고리즘(수학적 함수)입니다.

CKL Compromised Key List (손상된 키 목록)의 약자입니다. 클라이언트 사용자나 서버 사용자가 더 이상 신뢰해서는 안되는 인증서를 나타내는 목록으로, 인증 기관에서 발행합니다. 이러한 키는 손상된 키입니다. CRL도 *참조*하십시오.

클래스 로드 특정 규칙에 따라 Java 클래스를 로드할 책임이 있는 Java 구성 요소입니다. 클래스 경로도 *참조*하십시오.

클래스 경로 Java 클래스가 저장된 디렉토리 및 JAR 파일을 식별하는 경로입니다. 클래스 로드도 *참조*하십시오.

CLI 명령줄 인터페이스로, 사용자 프롬프트에 실행 가능한 명령을 입력할 수 있도록 해주는 인터페이스입니다. 관리 인터페이스도 *참조*하십시오.

클라이언트 인증 신뢰할 수 있는 CA 목록의 CA에 대해 인증서 서명과 인증서 체인을 암호로 확인함으로써 클라이언트 인증서를 인증하는 프로세스입니다. 인증, 인증 기관(CA)도 참조하십시오.

클라이언트 계약 클라이언트와 EJB 컨테이너 사이의 통신 규칙을 결정하고 Enterprise Bean을 사용하는 응용 프로그램의 통일된 개발 모델을 설정하며, 클라이언트와의 관계를 표준화하여 Bean의 활용도를 높이는 계약입니다.

CMP 컨테이너 관리 지속성(CMP)을 *참조*하십시오.

CMR 컨테이너 관리 관계(CMR)를 *참조*하십시오.

CMT 컨테이너 관리 트랜잭션(CMT)을 *참조*하십시오.

co-locate 원격 프로시저 호출을 방지하고 성능을 향상시키기 위해 같은 메모리 공간의 구성 요소를 관련된 구성 요소로 배포합니다.

열 데이터베이스 테이블의 필드입니다.

완결 데이터베이스에 명령을 전송하여 트랜잭션을 완료하는 것입니다. 롤백, 트랜잭션을 *참조*하십시오.

구성 요소 웹 응용 프로그램, Enterprise Bean, Message-Driven Bean, 응용 프로그램 클라이언트 또는 커넥터입니다. 응용 프로그램, 모듈도 *참조*하십시오.

구성 요소 계약 Enterprise Bean과 해당 컨테이너 사이의 관계를 설정하는 계약입니다.

구성 서버를 조정하거나 구성 요소에 메타데이터를 제공하는 프로세스입니다. 일반적으로 특정 구성 요소의 구성은 구성 요소의 배포 설명자 파일에 저장됩니다. Administration Server, 배포 설명자도 *참조*하십시오.

연결 팩토리 J2EE 구성 요소가 자원에 액세스할 수 있도록 해주는 연결 객체를 생성하는 객체입니다. 응용 프로그램 코드가 제공된 JMS 구현을 사용할 수 있도록 해주는 JMS 연결(TopicConnection 또는 QueueConnection)을 만드는 데 사용합니다. 응용 프로그램 코드는 JNDI 서비스를 사용하여 JNDI 이름을 사용한 연결 팩토리 객체를 찾습니다.

연결 풀 물리적 연결을 캐시 및 재사용하여 데이터베이스 액세스 효율성을 높이고, 이를 통해 연결 오버헤드를 감소시키며 많은 수의 스레드가 적은 수의 연결을 공유할 수 있도록 합니다. JDBC 연결 풀도 *참조*하십시오.

커넥터 컨테이너가 EIS에 연결성을 제공하는 표준 확장 메커니즘입니다. 커넥터는 EIS에 고유하며 EIS 연결성을 위한 자원 어댑터 및 응용 프로그램 개발 도구로 구성됩니다. 자원 어댑터는 커넥터 아키텍처에 정의된 시스템 수준 계약의 지원을 통해 컨테이너에 연결됩니다.

커넥터 구조(CA) J2EE 응용 프로그램과 EIS의 통합을 위한 구조입니다. 이 구조에는 두 부분이 있습니다. 하나는 EIS 공급업체 제공 자원 어댑터이고 다른 하나는 이 자원 어댑터를 연결할 수 있도록 하는 J2EE 서버입니다. 이 구조는 트랜잭션, 보안 및 자원 관리와 같이 J2EE 서버에 연결하기 위해 자원 어댑터가 지원해야 하는 계약 집합을 정의합니다.

컨테이너 특정 유형의 J2EE 구성 요소에 lifecycle 관리, 보안, 개발 및 런타임 서비스를 제공하는 엔티티입니다. Sun ONE Application Server는 웹 및 EJB 컨테이너를 제공하며 응용 프로그램 클라이언트 컨테이너를 지원합니다. 구성 요소도 *참조*하십시오.

컨테이너 관리 지속성(CMP) EJB 컨테이너에서 Entity Bean 지속성을 관리합니다. Entity Bean의 변수와 데이터 저장소 사이의 데이터 전송이며 Sun ONE Application Server가 데이터 액세스 논리를 제공하는 위치입니다. Bean 관리 지속성(BMP)도 *참조*하십시오.

컨테이너 관리 관계(CMR) 관계의 한 측에서의 동작이 다른 측에 영향을 미치는 클래스 쌍의 필드 사이의 관계입니다.

컨테이너 관리 트랜잭션(CMT) Enterprise Bean의 트랜잭션 경계가 선언적으로 지정되며 EJB 컨테이너에 의해 자동으로 관리됩니다. Bean 관리 트랜잭션(BMT)도 *참조*하십시오.

제어 설명자 Bean 메소드와 Enterprise Bean 트랜잭션 및 보안 등록 정보에 대한 선택적 개별 등록 정보 오버라이드를 지정할 수 있도록 하는 Enterprise Bean 구성 항목 집합입니다.

대화 상태 객체의 상태가 같은 클라이언트와의 반복적 상호 작용에 따라 변경되는 상태입니다. 지속성 상태도 *참조*하십시오.

쿠키 호출 웹 브라우저로 전송된 다음 서버가 같은 클라이언트의 호출을 인식할 수 있도록 해당 브라우저의 이후 호출마다 검색되는 정보의 작은 모음입니다. 쿠키는 도메인에 고유하며 응용 프로그램과 서버 사이의 다른 데이터 상호 교환과 같은 웹 서버 보안 기능의 장점을 활용할 수 있습니다.

CORBA Common Object Request Broker Architecture의 약자입니다. 객체 지향적 분산 컴퓨팅의 표준 구조 정의입니다.

COSNaming 서비스 IIOP 기반 이름 지정 서비스입니다.

CosNaming 제공자 전역 JNDI 이름 공간(IIOP 응용 프로그램 클라이언트에 액세스 가능)을 지원하기 위해 Sun ONE Application Server에는 CORBA 참조(원격 EJB 참조)의 바인딩을 지원하는 J2EE 기반 CosNaming 제공자가 포함됩니다.

create 메소드 작성할 때 Enterprise Bean을 사용자 정의하기 위한 메소드입니다.

CRL Certificate Revocation List의 약자입니다. 클라이언트 사용자나 서버 사용자가 더 이상 신뢰해서는 안되는 인증서를 나타내는 목록으로, 인증 기관에서 발행합니다. 이 경우 인증서가 호출되었습니다. CKL도 *참조*하십시오.

데이터 액세스 논리 데이터 소스와의 상호 작용이 관련된 비즈니스 논리입니다.

데이터베이스 RDBMS (Relational Database Management System)를 가리키는 일반 용어입니다. 많은 양의 서로 관련되며, 구성된 데이터의 작성 및 조작이 가능한 소프트웨어 패키지입니다.

데이터베이스 연결 데이터베이스 연결은 데이터베이스 또는 다른 데이터 소스와의 통신 연결입니다. 구성 요소는 여러 데이터베이스 연결을 동시에 만들고 조작하여 데이터에 액세스할 수 있습니다.

데이터 소스 데이터베이스와 같은 데이터의 소스에 대한 핸들입니다. 데이터 소스는 iPlanet Application Server에 등록된 다음 프로그래밍 방식으로 검색되어 데이터 소스와의 연결과 상호 작용이 설정됩니다. 데이터 소스 정의는 데이터 소스에 연결하는 방법을 지정합니다.

DataSource 객체 DataSource 객체에는 표현되는 실제 데이터 소스를 식별 및 정의하는 등록 정보 집합이 있습니다.

선언적 보안 구성 요소의 구성 파일에 보안 등록 정보를 선언하며 구성 요소의 컨테이너(예: Bean의 컨테이너 또는 서블릿 엔진)가 보안을 명시적으로 관리할 수 있도록 합니다. 이러한 유형의 보안에는 프로그래밍 방식의 제어가 필요 없습니다. 프로그래밍 방식 보안의 반대입니다. 컨테이너 관리 지속성(CMP)을 *참조*하십시오.

선언적 트랜잭션 컨테이너 관리 트랜잭션(CMT)을 참조하십시오.

암호 해독 암호화된 정보를 다시 해독이 가능하도록 변환하는 프로세스입니다.

위임 객체의 구성을 구현 전략으로 사용하기 위한 객체 지향적 기법입니다. 동작의 결과를 책임지는 한 객체가 조작을 피위임자인 다른 객체에 위임합니다. 예를 들어, 클래스 로더는 일부 클래스의 로드를 상위 클래스 로더에 위임하기도 합니다.

배포 응용 프로그램에 필요한 파일을 응용 프로그램 서버에 배포하여 응용 프로그램이 응용 프로그램 서버에서 실행되도록 하는 프로세스입니다. 어셈블리도 *참조*하십시오.

배포 설명자 각 모듈 및 응용 프로그램과 함께 제공되는 XML 파일로 배포 방법이 설명되어 있습니다. 배포 설명자는 모듈이나 응용 프로그램을 특정 컨테이너 옵션으로 배포하도록 배포 도구에 지시하며 배포자가 해결해야 하는 특정 구성 요구 사항에 대해 정의합니다.

대상 자원 Topic 또는 Queue 대상을 나타내는 객체입니다. 응용 프로그램에서 Queues 또는 게시/구독을 Topics에 읽기/쓰기할 때 사용합니다. 응용 프로그램 코드는 JNDI 서비스를 사용하여 JNDI 이름을 사용한 JMS 자원 객체를 찾습니다.

다이제스트 인증 사용자 이름과 암호를 일반 텍스트로 보내지 않고 사용자를 사용자 이름과 암호에 따라 인증할 수 있는 인증 방식입니다.

디지털 서명 메시지와 서명자 모두를 인증하는 데 사용되는 전자 보안 메커니즘입니다.

디렉토리 서버 Sun ONE Directory Server를 참조하십시오.

고유 이름 DN, DN 속성을 *참조*하십시오.

배포 가능 세션 클러스터의 모든 서버로 배포 가능한 사용자 세션입니다.

분산 트랜잭션 별도의 서버에 있을 수 있는 여러 이기종 데이터베이스에 적용할 수 있는 단일 트랜잭션입니다.

문서 루트 문서 루트(기본 문서 디렉토리라고도 함)는 원격 클라이언트가 사용할 수 있도록 만들고자 하는 모든 가상 서버의 파일이 포함된 중앙 디렉토리입니다.

도메인 레지스트리 도메인 레지스트리는 도메인 이름, 도메인 위치, 도메인 포트, 도메인 호스트 등과 같이 Sun ONE Application Server의 설치에 작성 및 구성된 모든 도메인에 대한 도메인 고유 정보가 포함된 단일 데이터 구조입니다.

DN Distinguished Name (고유 이름)의 약자로, 디렉토리 서버에 있는 항목의 이름을 나타내는 문자열입니다.

DN 속성 Distinguished Name (고유 이름) 속성으로, 연관된 사용자, 그룹 또는 객체의 식별 정보가 포함된 텍스트 문자열입니다.

DTD Document Type Definition의 약자로, XML 파일 클래스의 구조 및 등록 정보에 대한 설명입니다.

동적 재배포 서버를 다시 시작하지 않고 구성 요소를 재배포하는 프로세스입니다.

동적 재로드 서버를 다시 시작하지 않고 구성 요소를 업데이트 및 다시 로드하는 프로세스입니다. 기본적으로 서블릿, JavaServer Page (JSP) 및 Enterprise Bean 구성 요소는 동적으로 다시 로드할 수 있습니다. 버전 지정이라고도 합니다.

EAR 파일 Enterprise ARchive 파일의 약자로, J2EE 응용 프로그램이 포함된 압축 파일입니다. EAR 파일의 확장명은 `.ear`입니다. JAR 파일도 *참조*하십시오.

e-commerce 전자 상거래를 의미하며, 인터넷을 통해 수행되는 비즈니스를 가리키는 용어입니다.

EIS Enterprise Information System의 약자로, 패키지된 엔터프라이즈 응용 프로그램, 트랜잭션 시스템 또는 사용자 응용 프로그램으로 해석될 수 있습니다. EIS라고도 합니다. EIS의 예로 R/3, PeopleSoft, Tuxedo 및 CICS 등이 있습니다.

EJB 컨테이너 컨테이너를 *참조*하십시오.

EJB QL EJB Query Language의 약자로, 컨테이너 관리 관계로 정의되는 Entity Bean의 네트워크 전체를 탐색하는 쿼리 언어입니다.

EJB 기술 Enterprise Bean은 응용 프로그램의 비즈니스 논리를 캡슐화하는 서버측 구성 요소입니다. 비즈니스 논리는 응용 프로그램의 용도를 수행하는 코드입니다. 예를 들어, 재고 관리 응용 프로그램에서 Enterprise Bean은 `checkInventoryLevel` 및 `orderProduct`라는 메소드에 비즈니스 논리를 구현할 수 있습니다. 원격 클라이언트는 이 메소드를 호출하여 응용 프로그램이 제공하는 재고 관리 서비스에 액세스할 수 있습니다. 컨테이너, Entity Bean, Message-Driven Bean 및 Session Bean도 *참조*하십시오.

ejbc 유틸리티 Enterprise Beans의 컴파일러입니다. 이것은 모든 EJB 클래스 및 인터페이스의 EJB 사양 호환성을 확인하고 스텁 및 스켈레톤을 생성합니다.

요소 배열 내의 데이터 단위 또는 논리 요소 등과 같은 더 큰 집합의 구성원을 가리킵니다. XML 파일에서는 기본 구조적 단위입니다. XML 요소에는 하위 데이터나 데이터 또는 속성이 포함될 수 있습니다.

캡슐화 모듈 내의 지식을 현지화합니다. 객체가 데이터와 구현을 캡슐화하기 때문에 객체의 사용자는 객체를 서비스를 제공하는 블랙 박스로 볼 수 있습니다. 인스턴스 변수 및 메소드를 추가, 삭제 또는 변경할 수 있지만 객체가 제공하는 서비스가 그대로 유지되면 객체를 사용하는 코드는 다시 작성하지 않고 이를 계속 사용할 수 있습니다.

암호화 의도된 수신자 이외의 사람이 알 수 없도록 정보를 변환하는 프로세스입니다.

Entity Bean 데이터베이스의 행과 같이 물리적 데이터에 연관된 Enterprise Bean입니다. Entity Bean은 지속성 데이터에 연결되어 있기 때문에 수명이 깁니다. Entity Bean은 항상 트랜잭션이 가능하며 다중 사용자를 인식합니다. Message-Driven Bean, Read-Only Bean, Session Bean을 *참조*하십시오.

ERP Enterprise Resource Planning의 약자로, 엔터프라이즈 자원 계획을 지원하는 다중 모듈 소프트웨어 시스템입니다. ERP 시스템에는 일반적으로 구매, 재고, 인사, 고객 서비스, 선적, 재정 계획 및 기타 비즈니스의 중요 측면을 관리하기 위한 관계형 데이터베이스와 응용 프로그램이 포함됩니다.

이벤트 모듈이나 응용 프로그램에서 응답을 트리거하는 명명된 동작입니다.

외부 JDNI 자원 JNDI 서비스가 JNDI 서버로의 브리지 역할을 하도록 합니다.

facade 다양한 Enterprise JavaBean (EJB)를 관리하는 데 사용되는 응용 프로그램 고유 Stateful Session Bean입니다.

팩토리 클래스 지속성 관리자를 생성하는 클래스입니다. 연결 팩토리도 *참조*하십시오.

페일오버 Bean이 서버 충돌에서도 투명하게 생존할 수 있는 복구 프로세스입니다.

검색 메소드 클라이언트가 전역에서 사용 가능한 디렉토리의 Bean 모음 또는 Bean을 조회할 수 있도록 하는 메소드입니다.

파일 캐시 파일 캐시에는 파일 및 정적 파일 내용에 대한 정보가 들어 있습니다. 파일 캐시는 기본적으로 활성화됩니다.

방화벽 네트워크 관리자가 보안을 강화하기 위해 네트워크 전체에서 정보의 흐름을 제한할 수 있는 전자적 경계입니다.

양식 작업 처리기 양식의 명명된 버튼을 기반으로 작업을 수행하는 서블릿 또는 응용 프로그램 논리에서 특별하게 정의된 메소드입니다.

FQDN Fully Qualified Domain Name (정규화된 도메인 이름)의 약자로, 호스트 이름과 도메인 이름이 포함된 시스템의 전체 이름입니다.

일반 ACL 사용자 또는 그룹에 하나 이상의 사용 권한을 연관시키는 Sun ONE Directory Server의 명명된 목록입니다. 이 목록은 임의로 정의 및 액세스하여 원하는 사용 권한 집합을 기록할 수 있습니다.

일반 서블릿 `javax.servlet.GenericServlet`을 확장하는 서블릿입니다. 일반 서블릿은 프로토콜 독립적입니다. 따라서 HTTP 또는 기타 전송 프로토콜에 대한 기본 지원이 포함되어 있지 않습니다. HTTP 서블릿과 반대입니다.

전역 데이터베이스 연결 여러 구성 요소가 사용할 수 있는 데이터베이스 연결입니다. 자원 관리자가 필요합니다.

전역 트랜잭션 트랜잭션 관리자가 관리 및 조정하며 여러 데이터베이스와 프로세스에 관련되는 트랜잭션입니다. 트랜잭션 관리자는 일반적으로 XA 프로토콜을 사용하여 데이터베이스 백엔드와 상호 작용합니다. 로컬 트랜잭션를 *참조*하십시오.

세분성 수준 응용 프로그램을 여러 단위로 나누는 방법입니다. *높은 수준의 세분성*은 응용 프로그램이 여러 작고 더 세밀하게 정의된 Enterprise JavaBean (EJB)로 분리되었다는 의미입니다. *낮은 수준의 세분성*은 응용 프로그램이 더 작은 수의 단위로 분리되어 프로그램이 더 커진다는 의미입니다.

그룹 특정 방법으로 관련된 사용자 그룹입니다. 그룹 멤버십은 일반적으로 로컬 사용자 관리자가 관리합니다. 사용자, 역할을 *참조*하십시오.

핸들 Enterprise Bean을 식별하는 객체입니다. 클라이언트는 핸들을 일련화한 다음 다시 일련화를 해제하여 Bean에 대한 참조를 얻을 수 있습니다.

발견적 결정 특정 트랜잭션이 사용하는 트랜잭션 모드입니다. 트랜잭션은 완결 또는 롤백이어야 합니다.

홈 인터페이스 클라이언트가 Enterprise Bean을 작성 및 제거할 수 있도록 하는 메소드를 정의하는 메커니즘입니다.

호스트 IP 인증 Administration Server 또는 웹 사이트의 파일 및 디렉토리에 특정 컴퓨터를 사용하는 클라이언트만 액세스할 수 있도록 제한하는 데 사용되는 보안 메커니즘입니다.

HTML Hypertext Markup Language의 약자로, 웹 브라우저에 표시할 수 있는 문서를 만들 때 사용되는 코딩 마크업 언어입니다. 각 텍스트 블록 앞뒤에는 텍스트의 특성을 나타내는 코드가 추가됩니다.

HTML 페이지 HTML로 코딩되어 웹 브라우저에 표시되는 페이지입니다.

HTTP Hypertext Transfer Protocol의 약자로, 원격 호스트에서 하이퍼텍스트 객체를 불러오는 인터넷 프로토콜입니다. TCP/IP를 기반으로 합니다.

HTTP 서블릿 `javax.servlet.HttpServlet`을 확장하는 서블릿입니다. 이러한 서블릿에는 HTTP 프로토콜에 대한 지원이 내장되어 있습니다. 일반 서블릿과 반대입니다.

HTTPS HyperText Transmission Protocol, Secure의 약자로, 보안 트랜잭션을 위한 HTTP입니다.

IDE Integrated Development Environment의 약자로, 하나의 사용하기 쉬운 인터페이스에서 코드를 작성, 어셈블, 배포 및 디버그할 수 있는 소프트웨어입니다.

IIOP Internet Inter-ORB Protocol의 약자로, IIOP의 RMI (Remote Method Invocation) 및 CORBA (Common Object Request Broker Architecture)가 사용하는 전송 수준 프로토콜입니다.

IIOP Listener IIOP Listener는 특정 포트를 청취하여 CORBA 기반 클라이언트 응용 프로그램의 받는 연결(incoming connection)을 수용합니다.

IMAP Internet Message Access Protocol (인터넷 메시지 액세스 프로토콜)의 약자입니다.

IP 주소 TCP/IP 네트워크에서 컴퓨터나 기타 장치의 구조화된 숫자 식별자입니다. IP 주소의 형식은 점으로 구분된 네 부분의 숫자로 적힌 32비트 숫자 주소입니다. 각 숫자는 0부터 255 사이의 값입니다. 예를 들면 123.231.32.2와 같은 IP 주소가 있을 수 있습니다.

격리 수준 트랜잭션 격리 수준를 *참조*하십시오.

J2EE Java 2 Enterprise Edition의 약자로, 다중 계층, 웹 기반 엔터프라이즈 응용 프로그램을 개발 및 배포하기 위한 환경입니다. J2EE 플랫폼은 이러한 응용 프로그램을 개발하기 위한 기능을 제공하는 서비스, 응용 프로그램 프로그래밍 인터페이스(API) 및 프로토콜로 구성되어 있습니다.

JAF JAF (JavaBeans Activation Framework)는 MIME 데이터 유형에 대한 지원을 Java 플랫폼에 통합합니다. Mime 유형을 참조하십시오.

JAR 파일 Java ARchive 파일의 약자로, 여러 파일을 하나의 파일로 압축하는 데 사용되는 파일입니다. JAR 파일의 확장명은 `.jar`입니다.

JAR 파일 계약 Enterprise Bean 패키지에 있어야 하는 정보를 지정하는 Java ARchive 계약입니다.

JAR 파일 형식 Java ARchive 파일 형식으로, 여러 파일을 하나의 파일로 압축하는 플랫폼 독립적인 파일 형식입니다. 여러 애플릿과 각각의 필수 구성 요소(클래스 파일, 이미지, 사운드 및 기타 자원 파일)를 JAR 파일에 번들하여 하나의 HTTP 트랜잭션에서 브라우저로 다운로드할 수 있습니다. JAR 파일 형식은 파일 압축 및 디지털 서명도 지원합니다.

JavaBean 이식 가능하고, 플랫폼 독립적인 재사용 가능한 구성 요소 모델입니다.

Java IDL Java Interface Definition Language의 약자로, CORBA (Common Object Request Broker Architecture)와의 표준 기반 호환성과 연결성을 제공하는 Java 프로그래밍 언어로 작성된 API입니다.

JavaMail 세션 응용 프로그램이 우편 저장소와 상호 작용하기 위해 사용하는 객체입니다. 응용 프로그램 코드는 JNDI 서비스를 사용하여 JNDI 이름을 사용한 JavaMail 세션 자원 객체를 찾습니다.

JAXM Java API for XML Messaging의 약자입니다. 이것을 통해 응용 프로그램이 SOAP 표준을 사용하여 문서 지향적 XML 메시지를 보내고 받을 수 있습니다. 이러한 메시지에는 첨부 파일이 있거나 없을 수 있습니다.

JAXP Java API for XML Processing의 약자로, DOM, SAX 및 XSLT을 사용한 XML 문서의 처리를 지원하는 Java API입니다. 이를 통해 응용 프로그램이 XML 문서를 특정 XML 처리 구현과 독립적으로 구문 분석하고 변환합니다.

JAXR Java API for XML Registry의 약자로, 여러 종류의 XML 레지스트리에 액세스하기 위한 일관된 표준 Java API를 제공합니다. 사용자는 이를 통해 웹 서비스를 구축, 배포 및 검색할 수 있습니다.

JAX-RPC Java API for XML-based Remote Procedure Calls의 약자로, 개발자가 XML 기반 RPC 프로토콜을 기반으로 상호 운영 가능한 웹 응용 프로그램과 웹 서비스를 개발할 수 있게 합니다.

JDBC Java Database Connectivity의 약자로, 개발자가 데이터 인식 구성 요소를 작성할 수 있도록 하는 클래스 및 인터페이스의 표준 기반 집합입니다. JDBC는 플랫폼 및 공급 업체에 독립적인 방식으로 데이터 소스에 연결하고 상호 작용할 수 있는 메소드를 구현합니다.

JDBC 연결 풀 데이터베이스 연결 지정에 사용된 JDBC 데이터 소스 등록 정보를 연결 풀 등록 정보와 결합하는 풀입니다.

JDBC 자원 응용 프로그램 서버 내에서 실행 중인 응용 프로그램을 기존 JDBC 연결 풀을 사용하여 데이터베이스에 연결하는 데 사용되는 자원입니다. JNDI 이름(응용 프로그램에서 사용)과 기존 JDBC 연결 풀의 이름으로 구성됩니다.

JDK Java Development Kit의 약자로, 개발자가 Java 2 플랫폼 이전의 다양한 Java 플랫폼 버전용 응용 프로그램을 작성하는 데 필요한 API와 도구들이 포함된 소프트웨어입니다. JDK도 *참조*하십시오.

JMS Java Message Service의 약자로, JMS 클라이언트가 JMS 메시지 서비스의 기능에 액세스하는 방법을 정의하는 인터페이스 및 의미의 표준 집합입니다. 이러한 인터페이스는 Java 프로그램이 메시지를 작성하고, 보내고, 받고, 읽을 수 있는 표준 방법을 제공합니다.

JMS 관리 객체 하나 이상의 JMS 클라이언트에서 사용하기 위해 관리자가 작성하는 미리 구성된 JMS 객체(연결 팩토리 또는 대상)입니다. 관리되는 객체를 사용하면 JMS 클라이언트는 공급자에 독립적일 수 있습니다. 즉 공급자의 독점적 측면과 격리됩니다. 관리자가 이러한 객체를 JNDI 이름 공간에 위치시키며 JMS 클라이언트가 JNDI 조회를 사용하여 액세스합니다.

JMS 클라이언트 JMS 메시지를 사용하여 메시지를 교환하는 다른 JMS 클라이언트와 상호 작용하는 응용 프로그램 또는 소프트웨어 구성 요소입니다.

JMS 연결 팩토리 JMS 클라이언트가 JMS 메시지 서비스에 연결할 때 사용하는 JMS 관리 객체입니다.

JMS 대상 생성된 메시지가 라우팅을 위해 전달된 후 소비자에게 전달되는 JMS 메시지 서비스의 실제 대상입니다. 이 실제 대상은 JMS 클라이언트가 메시지를 생성하고 메시지를 사용할 대상을 지정할 때 사용하는 JMS 관리 객체에 의해 식별되고 캡슐화됩니다.

JMS 메시지 JMS 클라이언트가 사용하는 비동기 요청, 보고서 및 이벤트입니다. 메시지에는 헤더(추가 필드를 추가할 수 있음)와 본문이 있습니다. 메시지 헤더는 표준 필드와 선택적 등록 정보를 지정합니다. 메시지 본문에는 전송되는 데이터가 들어 있습니다.

JMS 공급자 메시지 시스템에 대한 JMS 인터페이스를 구현하고 완전한 생산에 필요한 관리 및 제어 기능을 추가하는 제품입니다.

JMS 서비스 JMS 클라이언트 연결, 메시지 라우팅 및 전달, 지속성, 보안, 로깅을 포함하여 JMS 메시징 시스템에 서비스를 제공하는 소프트웨어입니다. 메시지 서비스는 JMS 클라이언트가 메시지를 보내고 소비 클라이언트로 메시지를 보낼 실제 대상을 유지합니다.

JNDI Java Naming and Directory Interface의 약자로, Java 기술을 적용한 응용 프로그램에 엔터프라이즈의 여러 이름 지정 및 디렉토리 서비스에 대한 통일된 인터페이스를 제공하는 Java 플랫폼에 대한 표준 확장입니다. Java Enterprise API 집합의 일부인 JNDI는 이기종 엔터프라이즈 이름 지정 및 디렉토리 서비스에 자연스러운 연결성을 제공합니다.

JNDI 이름 JNDI 이름 지정 서비스에 등록된 자원에 액세스할 때 사용하는 이름입니다.

JRE Java Runtime Environment (Java 런타임 환경)의 약자로, Java 가상 머신, Java 코어 클래스, Java 프로그래밍 언어로 작성된 응용 프로그램에 런타임 지원을 제공하는 지원 파일로 구성된 Java Development Kit (JDK)의 일부입니다. JDK*도 참조*하십시오.

JSP JavaServer Page. HTML 또는 XML 태그, JSP 태그 및 Java 코드의 결합을 사용하여 작성한 텍스트 페이지입니다. JSP는 표준 브라우저 페이지의 레이아웃 기능을 프로그래밍 언어의 성능과 결합합니다.

jspc 유틸리티 JSP의 컴파일러로, 모든 JSP가 JSP 사양과 호환되는지 확인합니다.

JTA Java Transaction API. 응용 프로그램 및 J2EE 서버가 트랜잭션에 액세스할 수 있도록 하는 API입니다.

JTS Java Transaction Service. 트랜잭션을 처리하기 위한 Java 서비스입니다.

키 쌍 파일 *참조:* 트러스트 데이터베이스.

LDAP Lightweight Directory Access Protocol 약자입니다. TCP/IP에서 실행되는 공개 디렉토리 액세스 프로토콜입니다. 전역 크기와 수백만 항목으로 확장할 수 있습니다. 제공되는 LDAP 서버인 Sun ONE Directory Server를 사용하면 엔터프라이즈의 모든 정보를 모든 응용 프로그램 서버에서 네트워크를 통해 액세스할 수 있는 디렉토리 정보의 하나의 중앙 집중식 리포지토리에 저장할 수 있습니다.

LDIF LDAP Data Interchange Format의 약자입니다. Sun ONE Directory Server 항목을 텍스트 형식으로 나타낼 때 사용하는 형식입니다.

lifecycle 이벤트 시작 또는 종료 등과 같은 서버 lifecycle의 한 단계입니다.

lifecycle 모듈 서버 lifecycle에서 대기하며 이벤트에 대한 응답으로 해당 작업을 수행하는 모듈입니다.

Listener 이벤트가 발생한 경우 수행할 작업을 지시하는 게시 객체에 등록된 클래스입니다.

로컬 데이터베이스 연결 로컬 연결의 트랜잭션 컨텍스트는 여러 프로세스에 분산되거나 여러 데이터 소스에 걸쳐 있는 것이 아니라, 현재 프로세스와 현재 데이터 소스에 대해 로컬입니다.

로컬 인터페이스 Session 또는 Bean에 액세스하는 Entity Bean과 같은 Java 가상 머신(JVM)에 있는 클라이언트에 메커니즘을 제공하는 인터페이스입니다.

로컬 세션 하나의 서버에서만 볼 수 있는 사용자 세션입니다.

로컬 트랜잭션 한 데이터베이스에 고유하며 하나의 프로세스로 제한된 트랜잭션입니다. 로컬 트랜잭션은 단일 백엔드만 대상으로 하여 동작합니다. 로컬 트랜잭션은 일반적으로 JDBC API를 사용하여 구분합니다. 전역 트랜잭션도 *참조*하십시오.

매핑 객체 지향적 모델을 관계형 데이터 모델, 일반적으로 관계형 데이터베이스의 스키마로 결합하는 기능입니다. 스키마를 다른 구조로 변환하는 프로세스입니다. 사용자를 보안 역할에 매핑하는 것을 가리키기도 합니다.

MDB Message-Driven Bean을 *참조*하십시오.

Message-Driven Bean 비동기 메시지 소비자인 Enterprise Bean입니다. Message-Driven Bean에는 특정 클라이언트에 대한 상태가 없지만 인스턴스 변수에 열려 있는 데이터베이스 연결 및 EJB 객체에 대한 객체 참조를 포함하여 클라이언트 메시지의 처리에 대한 상태가 포함될 수 있습니다. 클라이언트는 Message-Driven Bean이 Message Listener인 대상에 메시지를 전송하여 Message-Driven Bean에 액세스합니다.

메시징 느슨하게 결합된 응용 프로그램이 정보를 안정적이고 안전하게 전송할 수 있도록 하는 엔터프라이즈 응용 프로그램에서 사용하는 비동기 요청, 보고 또는 이벤트의 시스템입니다.

메타데이터 이름, 동작에 대한 사양 등과 같은 구성 요소에 대한 정보입니다.

관리 정보 베이스(MIB) 마스터 SNMP 에이전트가 액세스할 수 있는 변수를 정의하는 트리 형태 구조입니다. MIB는 HTTP 서버의 네트워크 구성, 상태 및 통계에 대한 액세스를 제공합니다. SNMP를 사용하면 네트워크 관리 워크스테이션(NMS)에서 이 정보를 볼 수 있습니다. 네트워크 관리 스테이션(NMS) 및 SNMP도 *참조*하십시오.

MIME 데이터 유형 MIME (Multi-purpose Internet Mail Extension) 유형은 시스템에서 지원하는 멀티미디어 파일의 유형을 조절할 수 있습니다.

모듈 응용 프로그램 외부에 개별적으로 배포된 웹 응용 프로그램, Enterprise Bean, Message-Driven Bean, 응용 프로그램 클라이언트 또는 커넥터입니다. 응용 프로그램, 구성 요소, lifecycle 모듈도 *참조*하십시오.

네트워크 관리 스테이션(NMS) 특정 네트워크를 원격에서 관리하는 데 사용하는 시스템입니다. 일반적으로 NMS 소프트웨어는 수집된 데이터를 표시할 그래프를 제공하거나 해당 데이터를 사용하여 서버가 특정 허용 한계 내에서 작동하는지 확인합니다. SNMP도 *참조*하십시오.

NTV 이름, 유형, 값.

객체 지속성 지속성을 *참조*하십시오.

O/R 매핑 도구 객체-관계형 [데이터베이스] 도구. Entity Bean에 대한 XML 배포 설명자를 작성하는 Sun ONE Application Server 관리 인터페이스 내의 매핑 도구입니다.

패키지 공통 디렉토리에 저장된 관련된 클래스들의 모음입니다. Java 압축 JAR 파일에 함께 패키지화되기도 합니다. 어셈블리, 배포를 *참조*하십시오.

매개 변수 양식 필드 데이터, HTTP 헤더 정보 등을 포함하여 클라이언트에서 전송되어 요청 객체에 캡슐화된 이름/값 쌍입니다. 속성과 반대입니다. 일반적으로 보면 Java 메소드 또는 database- prepared 명령의 인수입니다.

passivation Bean을 완전 삭제하지 않고 메모리에서 Bean의 자원을 해제시키는 메소드입니다. 이런 방법을 통해 Bean을 지속시킬 수 있으며 인스턴스화의 오버헤드 없이 호출할 수 있습니다.

사용 권한 사용자 또는 그룹에 허용되거나 거부된 권한 집합입니다. ACL도 *참조*하십시오.

지속성 Enterprise Bean의 경우 Entity Bean의 상태를 인스턴스 변수와 기본 데이터베이스 사이로 전송하기 위한 프로토콜입니다. 유동성(transience)의 반대입니다. 세션의 경우 세션 저장 메커니즘입니다.

지속성 관리자 컨테이너에 설치된 Entity Bean의 지속성을 담당하는 엔티티입니다.

지속성 상태 객체의 상태가 지속적인 저장소(일반적으로 데이터베이스)에 보관되는 경우입니다.

플러그형 인증 J2EE 응용 프로그램이 J2SE 플랫폼의 Java 인증 및 인증 서비스(JAAS) 기능을 사용할 수 있도록 하는 메커니즘입니다. 개발자는 각자의 인증 메커니즘을 플러그할 수 있습니다.

지점 간 전달 모델 특정 대기열에 대한 주소 메시지를 생성하고 소비자는 자신의 메시지 보관용으로 설정된 대기열에서 메시지를 추출합니다. 메시지를 단일 메시지 소비자에게만 전달합니다.

풀링 성능 향상을 위해 여러 개의 미리 구성된 자원을 제공하는 프로세스입니다. 자원이 풀링되면 구성 요소는 새 인스턴스를 인스턴스화하는 것이 아니라 풀에 있는 기존 인스턴스를 사용할 수 있습니다. Sun ONE Application Server에서는 데이터베이스 연결, 서블릿 인스턴스 및 Enterprise Bean 인스턴스를 풀링할 수 있습니다.

POP3 Post Office Protocol

준비된 명령 보다 효율적인 반복 실행을 위해 미리 컴파일된 데이터베이스 명령(SQL에서)입니다. 준비된 명령에는 매개 변수가 포함될 수 있습니다. 준비된 실행문에는 하나 이상의 준비된 명령이 포함됩니다.

준비된 실행문 데이터를 가져오기 위해 반복적으로 사용되는 `QUERY`, `UPDATE` 또는 `INSERT` 문을 캡슐화하는 클래스입니다. 준비된 실행문에는 하나 이상의 준비된 명령이 포함됩니다.

표현 레이아웃 웹 페이지 내용의 형식입니다.

표현 논리 요청 처리, 응답으로 내용 생성 및 클라이언트를 위한 페이지 서식 지정을 포함하여 응용 프로그램에서 페이지를 작성하는 작업입니다. 일반적으로 웹 응용 프로그램에서 처리합니다.

기본 키 클라이언트가 특정 Entity Bean을 찾을 수 있도록 해주는 고유한 식별자입니다.

기본 키 클래스 이름 Bean의 기본 키의 전체 클래스 이름을 지정하는 변수입니다. JNDI 조회에 사용합니다.

기본 인증의 결과로 엔티티에 할당된 신분입니다.

개인 키 공개 키 암호화를 *참조*하십시오.

프로세스 활성 프로그램의 실행 순서입니다. 프로세스는 하나 이상의 스레드로 구성됩니다.

프로그래밍 방식 보안 보안을 구성 요소의 컨테이너(예: Bean의 컨테이너 또는 서블릿 엔진)가 처리하는 것이 아니라 코드에서 명시적으로 제어하는 프로세스입니다. 선언적 보안과 반대입니다.

프로그래머 구분 트랜잭션 Bean 관리 트랜잭션(BMT)을 *참조*하십시오.

등록 정보 응용 프로그램 구성 요소의 동작을 정의하는 단일 속성입니다. `server.xml` 파일에서 등록 정보는 이름/값 쌍이 포함된 요소입니다.

공개 키 암호화 각 사용자가 공개 키와 개인 키를 갖고 있는 암호화 형태입니다. 메시지는 수신자의 공개 키로 암호화되어 전송되고, 수신자를 개인 키를 사용하여 메시지 암호를 해독합니다. 이런 방법을 사용하면 개인 키가 사용자 이외의 다른 사람에게 공개할 필요가 없습니다.

공개/가입 전달 모델 게시자와 가입자는 일반적으로 익명이며 주제를 동적으로 게시 또는 가입할 수 있습니다. 시스템은 주제의 여러 게시자가 보낸 메시지를 여러 가입자에게 배포합니다.

QOS QOS (서비스 품질)는 서버 인스턴스 또는 가상 서버에 설정한 성능 한계를 가리킵니다. 예를 들어, ISP에서는 제공되는 대역폭의 양에 따라 서로 다른 요금을 부과할 수 있습니다. 대역폭의 양과 연결의 수를 제한할 수 있습니다.

대기열 관리자가 지점 간 전달 모델을 구현하기 위해 작성하는 객체입니다. 대기열은 메시지를 소비하는 클라이언트가 비활성화 상태인 경우에도 항상 메시지를 저장할 수 있습니다. 대기열은 생성자와 사용자 간의 중간 저장소 역할을 합니다.

RAR 파일 Resource ARchive의 약자로, 자원 어댑터가 포함된 JAR 압축 파일입니다.

RDB 관계형 데이터베이스.

RDBMS 관계형 데이터베이스 관리 시스템.

Read-Only Bean EJB 클라이언트가 수정하지 않는 Entity Bean입니다. Entity Bean도 *참조*하십시오.

영역 공통 보안 정책이 적용되고 보안 서비스의 보안 관리자가 보안을 적용하는 범위입니다. J2EE 사양에서 *보안 정책 도메인* 또는 *보안 도메인*이라고도 합니다.

원격 인터페이스 Enterprise JavaBean의 두 인터페이스 중 하나입니다. 원격 인터페이스는 클라이언트가 호출할 수 있는 비즈니스 메소드를 정의합니다.

요청 객체 클라이언트가 생성하여 서블릿 또는 JavaServer Page (JSP)에 입력 매개 변수로 전달되는 페이지 및 세션 데이터가 포함된 객체입니다.

자원 관리자 데이터베이스 또는 메시지 브로커 또는 Sun ONE Application Server 프로세스와 같은 자원의 클라이언트 간의 중개자 역할을 하는 객체입니다. 전역에서 사용 가능한 데이터 소스를 제어합니다.

자원 참조 자원에 대한 구성 요소의 코드화된 이름을 식별하는 배포 설명자의 요소입니다.

응답 객체 호출 클라이언트를 참조하며 클라이언트를 위한 출력을 생성할 메소드를 제공하는 객체입니다.

ResultSet `java.sql.ResultSet` 인터페이스를 구현하는 객체입니다. `ResultSets`은 데이터베이스 또는 다른 테이블 형식 데이터 소스에서 검색되는 행 집합을 캡슐화하는 데 사용됩니다.

재사용 가능 구성 요소 하나 이상의 자원 또는 응용 프로그램 등과 같이 여러 가지 기능에 사용할 수 있도록 작성한 구성 요소입니다.

RMI Remote Method Invocation (원격 메소드 호출)의 약자입니다. 개발자가 객체를 원격 프로세스에 전달할 수 있는 원격 인터페이스를 작성할 수 있도록 하는 Java 표준 API 집합입니다.

RMIC Remote Method Invocation Compiler (원격 메소드 호출 컴파일러)의 약자입니다.

역할 배포된 환경에서 하나 이상의 그룹으로 나타나는, 응용 프로그램의 역할의 기능적 그룹화. 사용자, 그룹도 *참조*하십시오.

롤백 트랜잭션의 취소.

행 테이블의 각 열에 대한 값이 포함되어 있는 단일 데이터 레코드.

RowSet 데이터베이스 또는 다른 테이블 형식 데이터 소스에서 검색되는 행 집합을 캡슐화하는 객체입니다. `RowSet`은 `java.sql.ResultSet` 인터페이스를 확장하여 `ResultSet`이 JavaBeans 구성 요소 역할을 하도록 합니다.

RPC Remote Procedure Call (원격 프로시저 호출)의 약자입니다. 원격 객체 또는 서비스에 액세스하기 위한 메커니즘입니다.

런타임 시스템 프로그램이 실행되는 소프트웨어 환경. 런타임 시스템에는 Java 프로그래밍 언어로 작성된 프로그램을 로드하고, 기본 메소드를 동적으로 링크하고, 메모리를 관리하고, 예외를 처리하는 데 필요한 모든 코드가 포함되어 있습니다. Java 가상 머신의 구현이 포함되며, 이것은 Java 인터프리터일 수 있습니다.

SAF Server Application Function (서버 응용 프로그램 함수). 요청 처리 및 기타 서비스 활동에 참여하는 함수입니다.

스키마 테이블 이름, 열 이름 및 유형, 색인 정보 및 관계(기본 및 외래 키) 정보를 포함한 기본 데이터베이스의 구조.

보안 소켓 계층 *참조:* SSL.

보안 인증된 클라이언트만 응용 프로그램 자원에 액세스할 수 있도록 하는 스크린 메커니즘.

일련화 가능 객체 해체 및 재구성이 가능하여 여러 서버 간에 저장 또는 배포될 수 있는 객체입니다.

서버 인스턴스 Sun ONE Application Server에는 같은 시스템의 같은 설치에 여러 개의 인스턴스가 있을 수 있습니다. 각 인스턴스에는 고유 디렉토리 구조, 구성 및 배포 응용 프로그램이 있습니다. 각 인스턴스에는 여러 가상 서버도 포함됩니다. 가상 서버도 *참조*하십시오.

서블릿 `Servlet` 클래스의 인스턴스입니다. 서블릿은 서버에서 실행되는 재사용 가능한 응용 프로그램입니다. Sun ONE Application Server에서 서블릿은 표현 논리를 수행하고, 비즈니스 논리를 호출하고, 표현 레이아웃을 호출 또는 수행하여 응용 프로그램의 각 상호 작용에 대한 중앙 디스패처 역할을 합니다.

서블릿 엔진 모든 서블릿 조작을 처리하는 내부 객체입니다. 인스턴스화 및 실행을 포함하여, 서블릿에 대한 서비스를 제공하는 프로세스 집합입니다.

서블릿 실행자 요청 객체와 응답 객체로 서블릿을 호출하는 서블릿 엔진의 일부입니다. 서블릿 엔진을 *참조*하십시오.

세션 서블릿이 여러 HTTP 요청에서 웹 응용 프로그램과 사용자의 상호 작용을 추적하기 위해 사용하는 객체입니다.

Session Bean 클라이언트가 작성한 Enterprise Bean으로, 일반적으로 하나의 클라이언트-서버 세션 동안에만 존재합니다. Session Bean은 다른 EJB의 계산 또는 액세스 등과 같은 클라이언트에 대한 작업을 수행합니다. Session Bean은 트랜잭션이 가능할 수 있지만 시스템 크래시가 발생하면 복구할 수 없습니다. Session Bean 객체는 상태가 없거나(특정 클라이언트에 연관되지 않음) 상태가 있을 수(특정 클라이언트에 연관됨) 있습니다. 즉, 메소드와 트랜잭션 간에 대화 상태를 유지할 수 있습니다. stateful session bean, stateless session bean도 *참조*하십시오.

세션 쿠키 사용자 세션 식별자가 포함된 클라이언트에 반환되는 쿠키. *참조:* 고정 쿠키.

세션 시간 초과 Sun ONE Application Server가 사용자 세션을 무효화할 수 있는 지정된 기간. 세션을 *참조*하십시오.

단일 사인 온(SSO) 사용자의 인증 상태를 하나의 가상 서버 인스턴스에서 여러 J2EE 응용 프로그램이 공유할 수 있는 상태.

SMTP Simple Mail Transport Protocol

SNMP SNMP (Simple Network Management Protocol)는 네트워크 활동에 대한 데이터를 교환하는 데 사용되는 프로토콜입니다. 데이터는 SNMP를 통해 관리되는 장치와 네트워크 관리 스테이션(NMS) 사이에서 송수신됩니다. 관리되는 장치는 SNMP를 실행하는 모든 것, 즉 호스트, 라우터, 웹 서버 및 네트워크의 기타 서버입니다. NMS는 해당 네트워크를 원격에서 관리하는 데 사용하는 시스템입니다.

SOAP Simple Object Access Protocol (SOAP)은 XML 기반 데이터 구조 및 Hyper Text Transfer Protocol (HTTP)을 사용하여 인터넷의 다양한 운영 환경에 분산된 객체에서 메소드를 호출하는 표준화된 방법을 정의합니다.

SQL Structured Query Language의 약자로, 관계형 데이터베이스 응용 프로그램에서 널리 사용되는 언어입니다. SQL2 및 SQL3은 언어의 버전을 나타냅니다.

SSL Secure Sockets Layer (보안 소켓 계층). 인터넷에 보안 통신을 제공하기 위한 프로토콜입니다.

상태 **1.** 특정 시간의 엔티티의 환경 또는 상태. **2.** Sun ONE Application Server 기능 인터페이스 `IState2`를 사용하여 응용 프로그램의 상태를 저장할 수 있는 분산 데이터 저장 메커니즘입니다. 대화 상태, 지속성 상태도 *참조*하십시오.

stateful session bean 특정 클라이언트와의 세션을 나타내며 여러 클라이언트 호출 메소드에서 상태를 자동으로 유지하는 Session Bean입니다.

stateless session bean 상태 없는 서비스를 나타내는 Session Bean입니다. Stateless Session Bean은 제한된 시간 동안 특정 클라이언트가 필요로 하는 비즈니스 논리의 일시적 부분을 완전히 일시화 및 캡슐화합니다.

고정 쿠키 항상 같은 서버 프로세스에 연결되도록 클라이언트에 반환되는 쿠키입니다. 세션 쿠키도 *참조*하십시오.

저장 프로시저 SQL로 작성되어 데이터베이스에 저장된 실행문 블록입니다. 저장 프로시저를 사용하여 레코드 수정, 삽입 또는 삭제 등의 모든 유형의 데이터베이스 작업을 수행할 수 있습니다. 저장 프로시저를 사용하면 네트워크로 전송되는 정보의 양이 감소되므로 데이터베이스 성능이 향상됩니다.

스트리밍 HTTP를 통해 데이터를 통신하는 방법을 관리하기 위한 기술입니다. 결과를 스트리밍하면 데이터의 첫 번째 부분을 곧바로 사용할 수 있게 됩니다. 결과를 스트리밍하지 않으면 결과 전체를 받아야 일부라도 사용할 수 있습니다. 스트리밍을 사용하면 많은 양의 반환된 데이터를 보다 효율적으로 사용할 수 있으므로 응용 프로그램의 체감 성능을 높일 수 있습니다.

Sun ONE Directory Server LDAP (Lightweight Directory Access Protocol)의 Sun ONE 버전입니다. Sun ONE Application Server의 모든 인스턴스는 Sun ONE Directory Server를 사용하여 사용자 및 그룹에 대한 정보를 포함한 공유된 서버 정보를 저장합니다. LDAP도 *참조*하십시오.

Sun ONE Message Queue JMS (Java Message Service) 공개 표준을 구현하는 Sun ONE 엔터프라이즈 메시징 시스템입니다. JMS 공급자입니다.

시스템 관리자 Sun ONE Application Server 소프트웨어를 관리하고 Sun ONE Application Server 응용 프로그램을 배포하는 사람입니다.

테이블 데이터베이스의 행과 열에 있는 관련된 데이터의 명명된 그룹입니다.

스레드 프로세스 내의 실행 순서입니다. 멀티 스레드를 사용하는 경우 프로세스에서 여러 개의 스레드를 동시에 실행할 수 있습니다. 프로세스가 각 스레드를 차례로 실행하는 것은 단일 스레드입니다.

TLS Transport Layer Security (전송 계층 보안)의 약자로, 클라이언트 및 서버 응용 프로그램을 크게 변경하지 않고 데이터를 보안 채널로 전송할 수 있도록 해주는 전송 계층에서의 암호화 및 인증을 제공하는 프로토콜입니다.

주제 관리자가 게시/가입 전달 모델을 구현하기 위해 작성하는 객체입니다. 주제는 해당 주소로 전송된 메시지를 수집 및 배포하는 역할을 하는 내용 계층 구조에서 노드로 표시됩니다. 주제를 매개체로 사용하여 메시지 게시자와 메시지 가입자를 구분할 수 있습니다.

트랜잭션 그룹 단위로 성공하거나 실패하는 데이터베이스 명령 집합입니다. 관련된 모든 명령이 성공해야 전체 트랜잭션이 성공합니다.

트랜잭션 속성 트랜잭션 속성은 트랜잭션 범위를 제어합니다.

트랜잭션 컨텍스트 트랜잭션의 로컬 또는 전역 범위. 로컬 트랜잭션, 전역 트랜잭션를 *참조*하십시오.

트랜잭션 격리 수준 데이터베이스의 동시 트랜잭션이 서로에게 나타날 수 있는 범위를 결정합니다.

트랜잭션 관리자 일반적으로 XA 프로토콜을 사용하여 전역 트랜잭션을 제어하는 객체입니다. 전역 트랜잭션을 *참조*하십시오.

트랜잭션 복구 분산 트랜잭션의 자동 또는 수동 복구.

유동성(transience) 사용되지 않으면 자원을 해제하는 프로토콜입니다. 지속성과 반대입니다.

트러스트 데이터베이스 공개 및 개인 키가 포함된 보안 파일로 키 쌍 파일이라고도 합니다.

UDDI Universal Description, Discovery, and Integration의 약자로, 발견 및 통합을 위한 웹 서비스의 세계적 레지스트리를 제공합니다.

URI Universal Resource Identifier의 약자로, 도메인의 특정 자원을 설명합니다. 로컬에서는 기본 디렉토리의 하위 집합으로 설명되므로 `/ham/burger`가 기본 디렉토리이며 URI는 `toppings/cheese.html`을 지정하는 경우, 해당 URL은 `http://domain:port/toppings/cheese.html`이 됩니다.

URL Uniform Resource Locator의 약자로, HTML 페이지 또는 기타 자원을 고유하게 식별하는 주소입니다. 웹 브라우저는 URL을 사용하여 표시할 페이지를 지정합니다. URL은 전송 프로토콜(예: HTTP, FTP), 도메인(예: `www.my-domain.com`) 및 선택적으로 URI을 정의합니다.

사용자 응용 프로그램을 사용하는 사람입니다. 프로그래밍 관점에서 보면 사용자는 응용 프로그램이 클라이언트를 인식할 수 있는 사용자 이름, 암호 및 속성으로 구성됩니다. 그룹, 역할도 *참조*하십시오.

사용자 세션 서버에서 추적하는 일련의 사용자 응용 프로그램 상호 작용입니다. 세션은 사용자 상태, 지속성 객체 및 신원 인증을 관리합니다.

버전 관리 동적 재로드를 *참조*하십시오.

가상 서버 특정 URL을 대상으로 하여 내용을 서비스하는 가상 웹 서버입니다. 여러 가상 서버가 같거나 다른 호스트 이름, 포트 번호 또는 IP 주소를 사용하여 내용을 서비스할 수 있습니다. HTTP 서비스는 받는 웹 요청을 URL을 기준으로 다른 가상 서버에 리디렉션할 수 있습니다. 가상 호스트라고도 합니다. 웹 응용 프로그램을 특정 가상 서버에 할당할 수 있습니다. 서버 인스턴스에는 여러 가상 서버가 있을 수 있습니다. 서버 인스턴스도 *참조*하십시오.

WAR 파일 Web ARchive의 약자로, 웹 모듈이 포함된 Java 압축 파일입니다. WAR 파일의 확장명은 .war입니다.

웹 응용 프로그램 서블릿, JavaServer Page, HTML 문서, 그리고 이미지 파일, 압축 파일 및 기타 데이터가 포함될 수 있는 기타 자원의 모음입니다. 웹 응용 프로그램을 압축 파일(WAR 파일)로 패키지화되거나 공개 디렉토리 구조에 있을 수 있습니다. Sun ONE Application Server는 SHTML 및 CGI와 같은 Java가 아닌 웹 응용 프로그램 기술도 지원합니다.

웹 캐시 성능을 향상시키기 위해 서블릿이나 JSP가 특정 기간 동안 결과를 캐시하도록 할 수 있는 Sun ONE Application Server 기능입니다. 기간 내에 해당 서블릿이나 JSP를 다시 호출하면 캐시된 결과가 반환되므로 서블릿이나 JSP를 다시 실행할 필요가 없습니다.

웹 커넥터 플러그 인 웹 서버가 Sun ONE Application Server와 통신할 수 있도록 하는 웹 서버의 확장입니다.

웹 컨테이너 컨테이너를 *참조*하십시오.

웹 모듈 개별적으로 배포된 웹 응용 프로그램입니다. 웹 응용 프로그램을 *참조*하십시오.

웹 서버 HTML 페이지와 웹 응용 프로그램을 저장하고 관리하지만 완전한 J2EE 응용 프로그램은 아닌 호스트입니다. 웹 서버는 웹 브라우저를 통한 사용자 요청에 응답합니다.

웹 서버 플러그 인 웹 서버 플러그 인은 Sun One Web Server 또는 Sun ONE Application Server가 특정 HTTP 요청을 다른 서버에 전달하도록 지시할 수 있는 HTTP 역 프록시 플러그 인입니다.

웹 서비스 웹을 통해 제공되는 서비스입니다. 인터넷이나 인트라넷을 통해 시스템이 보낸 요청을 받아서 처리한 다음 응답을 반환할 수 있는, 자체 정의가 가능한 독립 모듈 형식의 응용 프로그램입니다.

WSDL Web Service Description Language의 약자로, 표준화된 방식으로 웹 서비스를 정의하는 데 사용되는 XML 기반 언어입니다. 기본적으로 웹 서비스의 세 가지 핵심 등록 정보인 웹 서비스의 정의, 웹 서비스 액세스 방법 및 웹 서비스의 위치를 정의합니다.

XA 프로토콜 분산 트랜잭션을 위한 데이터베이스 산업 표준 프로토콜입니다.

XML Extensible Markup Language의 약자로, HTML 스타일 태그를 사용하여 문서에 서식을 지정하고 문서에 사용된 정보 종류를 식별하는 언어입니다.

색인

가

나

다

라

마

바

사

아

자

차

카

타

파

NUMERICS

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X