Sun Java™ System

Application Server Enterprise Edition 8.1 관리 설명서

2005Q1

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

부품 번호: 819-1552

목차

머리말

이 설명서에서는 Application Server를 구성 및 관리하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

대상

*관리 설명서*는 작업 환경의 정보 기술 관리자를 대상으로 합니다. 이 설명서에서는 사용자가 다음 사항에 대해 잘 알고 있다고 가정합니다.

- 기본 시스템 관리 작업
- 소프트웨어 설치
- 웹 브라우저 사용
- 데이터베이스 서버 시작

- 단말기 창에서 명령 실행

이 설명서를 읽기 전에

Application Server는 Sun Java™ Enterprise System의 구성 요소이며 네트워크나 인터넷 환경에서 배포된 엔터프라이즈 응용 프로그램을 지원하는 소프트웨어 인프라입니다. Sun Java Enterprise System과 함께 제공되는 설명서는 반드시 숙지해야 하며 http://docs.sun.com/app/docs/prod/entsys.05q1 및 http://docs.sun.com/app/docs/prod/entsys.05q1?l=ko에서 온라인으로 액세스할 수 있습니다.

본 설명서의 구성

이 설명서는 Application Server를 관리하기 위한 브라우저 기반 도구인 관리 콘솔의 레이아웃에 따라 구성되어 있습니다. 모든 장은 개념적인 정보로 시작하고, 그 다음에 관리 콘솔을 사용하여 특정 작업을 수행하는 방법을 설명하는 절차에 관한 절이 나옵니다.

본 설명서에 사용된 규칙

이 절의 표에서는 본 설명서에 사용된 규칙에 대해 설명합니다.

활자체 규약

다음 표에서는 본 설명서에서 사용된 활자체 변경 사항에 대해 설명합니다.

표 1 활자체 규약

서체	의미	예
AaBbCc123 (고정 폭)	API 및 언어 요소, HTML 태그, 웹 사이트 URL, 명령 이름, 파일 이름, 디렉토리 경로 이름, 화면 시스템 출력, 샘플 코드에 사용됩니다.	`.login` 파일을 편집합니다. `ls -a`를 사용하여 모든 파일을 나열합니다. `% You have mail.`
AaBbCc123 (굵은 고정 폭)	화면 상의 컴퓨터 출력과는 반대로 사용자가 직접 입력하는 사항입니다.	`% su` `Password:`

표 1 활자체 규약 *(계속)*

서체	의미	예
AaBbCc123 (기울임꼴)	책 제목, 새로운 용어, 강조 표시할 단어에 사용됩니다. 실제 이름이나 값으로 대체할 명령이나 경로 이름의 자리 표시자입니다.	*사용자 설명서*의 6장을 참조하십시오. 이를 *class* 옵션이라고 합니다. 파일을 저장하지 *마십시오*. 파일은 *install-dir*/bin 디렉토리에 있습니다.

기호

다음 표에서는 본 설명서에서 사용한 기호 규칙에 대해 설명합니다.

표 2 기호 규칙

기호	설명	예	의미
[ ]	선택적 명령 옵션이 포함됩니다.	ls [-l]	-l 옵션은 사용하지 않아도 됩니다.
{ | }	필수 명령 옵션에 대한 일련의 선택 항목이 포함됩니다.	-d {y|n}	-d 옵션에서는 y 인수나 n 인수를 사용해야 합니다.
-	동시에 입력하는 여러 키를 결합합니다.	Control-A	Ctrl 키를 누른 채 A 키를 누릅니다.
+	연속해서 입력하는 여러 키를 결합합니다.	Ctrl+A+N	Ctrl 키를 눌렀다가 놓은 다음 후속 키를 누릅니다.
>	그래픽 사용자 인터페이스의 메뉴 항목 선택을 표시합니다.	파일> 새로 만들기> 템플리트	파일 메뉴에서 새로 만들기를 선택합니다. 새로 만들기 하위 메뉴에서 템플리트를 선택합니다.

기본 경로 및 파일 이름

다음 표에서는 본 설명서에서 사용한 기본 경로와 파일 이름에 대해 설명합니다.

표 3 기본 경로 및 파일 이름

용어	설명
install_dir	Application Server의 기본 설치 디렉토리 위치입니다. • Solaris™플랫폼에 Sun Java Enterprise System을 설치한 경우: `/opt/SUNWappserver/appserver` • Linux 플랫폼에 Sun Java Enterprise System을 설치한 경우 : `/opt/sun/appserver/` • 기타 Solaris 및 Linux 설치, 루트가 아닌 사용자의 경우: *user's home directory*`/SUNWappserver` • 기타 Solaris 및 Linux 설치, 루트 사용자의 경우: `/opt/SUNWappserver` • Windows, 모든 설치: *SystemDrive*`:\Sun\AppServer`
domain_root_dir	모든 도메인을 포함하는 디렉토리의 기본 위치입니다. • Solaris 플랫폼에 Sun Java Enterprise System을 설치한 경우: `/var/opt/SUNWappserver/domains/` • Linux 플랫폼에 Sun Java Enterprise System을 설치한 경우: `/var/opt/sun/appserver/domains/` • 기타 모든 설치: *install_dir*`/domains/`
domain_dir	도메인 디렉토리의 기본 위치입니다. *domain_root_dir*`/`*domain_dir* 구성 파일에서 *domain_dir*이 다음과 같이 표시되어 있을 수 있습니다. `${com.sun.aas.instanceRoot}`
instance_dir	인스턴스 디렉토리의 기본 위치입니다. *domain_dir*`/`*instance_dir*

쉘 프롬프트

다음 표에서는 본 설명서에서 사용한 쉘 프롬프트를 설명합니다.

표 4 쉘 프롬프트

쉘	프롬프트
UNIX 또는 Linux의 C 쉘	*machine-name*%
UNIX 또는 Linux의 C 쉘 수퍼유저	*machine-name*#
UNIX 또는 Linux의 Bourne 쉘 및 Korn 쉘	$
UNIX 또는 Linux의 Bourne 쉘 및 Korn 쉘 수퍼유저	#
Windows 명령줄	C:\

관련 설명서

http://docs.sun.com℠ 웹 사이트에서 Sun 기술 관련 설명서를 온라인으로 이용할 수 있습니다. 아카이브를 탐색하거나 특정 책 제목 또는 주제를 검색할 수 있습니다.

install_dir/docs/index.htm에서 공식 사양에 대한 URL 디렉토리를 찾을 수 있습니다. 다음과 같은 자원을 사용할 수도 있습니다.

일반 J2EE 정보:

J2EE 1.4 Tutorial:

http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

J2EE Blueprints:

http://java.sun.com/reference/blueprints/index.html

Core J2EE Patterns: Best Practices and Design Strategies, Deepak Alur, John Crupi, & Dan Malks, Prentice Hall Publishing

Java Security, Scott Oaks, O'Reilly Publishing

서블릿 및 JSP 파일을 사용한 프로그래밍:

Java Servlet Programming, Jason Hunter, O'Reilly Publishing

Java Threads, 2nd Edition, Scott Oaks & Henry Wong, O'Reilly Publishing

EJB 구성 요소를 사용한 프로그래밍:

Enterprise JavaBeans, Richard Monson-Haefel, O'Reilly Publishing

JDBC를 사용한 프로그래밍:

Database Programming with JDBC and Java, George Reese, O'Reilly Publishing

JDBC Database Access With Java: A Tutorial and Annotated Reference(Java Series), Graham Hamilton, Rick Cattell, & Maydene Fisher

본 설명서 세트의 책

Sun Java System Application Server 설명서는 온라인상에서 PDF(Portable Document Format) 및 HTML(Hypertext Markup Language) 형식으로 볼 수 있습니다.

다음 표에서는 Application Server 핵심 설명서 세트에 포함된 책을 요약합니다.

표 5 본 설명서 세트의 책

설명서 제목	설명
릴리스 노트	소프트웨어 및 설명서 관련 최신 정보로, 지원되는 하드웨어, 운영 체제, JDK 및 JDBC/RDBMS를 표를 기반으로 종합적으로 요약합니다.
빠른 시작 설명서	Sun Java System Application Server 제품을 시작하는 방법에 대해 설명합니다.
Installation Guide	Application Server 소프트웨어와 해당 구성 요소 설치에 대해 설명합니다.
Deployment Planning Guide	사용자 사이트에 가장 적합한 방법으로 Sun Java System Application Server를 배포할 수 있도록 시스템 요구 사항과 기업을 평가하는 방법에 대해 설명합니다. Application Server를 배포할 때 알고 있어야 할 일반적인 문제점에 대해서도 설명합니다.
Developer's Guide	J2EE 구성 요소 및 API용 개방형 Java 표준 모델을 따르는 Application Server에서 실행할 Java™ 2 Platform, Enterprise Edition(J2EE™ 플랫폼) 응용 프로그램을 작성 및 구현에 대해 설명합니다. 개발자 도구, 보안, 어셈블리, 배포, 디버깅 및 라이프사이클 모듈 작성에 대한 일반 정보가 포함됩니다.
J2EE 1.4 Tutorial	J2EE 1.4 플랫폼 기술과 API를 사용하여 J2EE 응용 프로그램을 개발하고 Sun Java System Application Server에 응용 프로그램을 배포하는 방법에 대해 설명합니다.
Administrator's Guide	관리 콘솔에서 Application Server의 하위 시스템 및 구성 요소 구성, 관리 및 배포에 대해 설명합니다.
High Availability Administration Guide	Sun Java System Application Server 고가용성 기능 구성 및 관리에 대해 설명합니다.
Administration Reference	Sun Java System Application Server 구성 파일 `domain.xml` 편집에 대해 설명합니다.

표 5 본 설명서 세트의 책 *(계속)*

설명서 제목	설명
Upgrade and Migration Guide	응용 프로그램을 새로운 Sun Java System Application Server 프로그래밍 모델로 마이그레이션, 특히 Application Server 6.x 및 7에서 마이그레이션 및 제품 사양과 비호환성을 야기할 수 있는 인접한 제품 릴리스 및 구성 옵션 간의 차이를 설명합니다.
Performance Tuning Guide	성능 개선을 위한 Sun Java System Application Server 조정에 대해 설명합니다.
Troubleshooting Guide	Sun Java System Application Server 문제 해결에 대해 설명합니다.
Error Message Reference	Sun Java System Application Server 오류 메시지 해결에 대해 설명합니다.
Reference Manual	설명서 페이지 스타일로 작성된 Sun Java System Application Server와 같이 사용할 수 있는 유틸리티 명령에 대해 설명합니다. `asadmin` 명령줄 인터페이스도 설명합니다.

기타 서버 설명서

기타 서버 설명서를 보려면 다음을 참조하십시오.

- Message Queue 설명서
  http://docs.sun.com/app/docs?p=prod/s1.s1msgqu
- Directory Server 설명서
  http://docs.sun.com/coll/DirectoryServer_04q2 및
  http://docs.sun.com/coll/DirectoryServer_04q2_ko
- Web Server 설명서
  http://docs.sun.com/coll/S1_websvr61_en 및
  http://docs.sun.com/coll/S1_websvr61_ko

Sun 자원 온라인 액세스

제품 다운로드, 전문가 서비스, 패치 및 지원, 추가 개발자 정보 등을 얻으려면 다음으로 이동하십시오.

- 다운로드 센터
  http://www.sun.com/software/download/
- 전문가 서비스
  http://www.sun.com/service/sunps/sunone/index.html
- Sun Enterprise Services, Solaris 패치 및 지원
  http://sunsolve.sun.com/

- 개발자 정보
  http://developers.sun.com/prodtech/index.html

Sun 기술 지원 문의

제품 설명서에 나와 있지 않은 본 제품에 대한 기술적인 질문 사항이 있을 경우에는 http://www.sun.com/service/contacting을 방문하십시오.

타사 웹 사이트 관련 참조 사항

Sun은 이 설명서에서 언급한 타사 웹 사이트의 가용성에 대해 책임을 지지 않습니다. Sun은 해당 사이트 또는 자원을 통해 사용할 수 있는 모든 컨텐트, 광고, 제품 또는 기타 자료에 대해 보증하지 않으며 책임을 지지 않습니다. Sun은 해당 사이트 또는 자원을 통해 사용할 수 있는 컨텐트, 제품 또는 서비스 사용과 관련하여 실제로 발생했거나 발생한 것으로 추정되는 피해나 손실에 대해 책임을 지지 않습니다.

사용자 의견 환영

Sun은 해당 설명서의 내용을 개선하기 위해 노력하고 있으며 사용자의 의견 및 제안을 환영합니다.

사용자 의견을 보내시려면 http://docs.sun.com에서 의견 보내기를 누르고 온라인 양식에 설명서 제목과 부품 번호를 입력합니다. 부품 번호는 설명서의 제목 페이지나 문서 맨 위에 있는 7자리 또는 9자리 숫자입니다. 예를 들어, 본 설명서의 제목은 *Sun Java System Application Server 2005Q1 관리 설명서*이고 부품 번호는 819-1552입니다.

사용자 의견을 제출할 때 해당 양식에 영문 설명서의 제목과 부품 번호를 입력해야 할 수도 있습니다. 본 설명서의 영문 부품 번호와 제목은 Sun Java System Application Server 2005Q1 Administration Guide(819-0215) 입니다.

시작하기

이 장에서는 Sun Java™ System Application Server에 대해 설명하고 기본 관리 작업을 소개합니다. 이 장은 다음 내용으로 구성되어 있습니다.

Sun Java System Application Server 정보

Application Server

Application Server는 엔터프라이즈 응용 프로그램의 개발, 배포 및 관리를 위한 강력한 J2EE 플랫폼을 제공합니다. 주요 기능으로는 트랜잭션 관리, 성능, 확장성, 보안 및 통합 등이 있습니다. Application Server는 웹 게시부터 엔터프라이즈급 트랜잭션 처리까지의 서비스를 지원하는 한편, 개발자가 JavaServer Pages(JSP™), Java Servlet 및 Enterprise JavaBeans™(EJB™) 기술을 바탕으로 응용 프로그램을 작성할 수 있도록 합니다.

Application Server Enterprise Edition에서는 고급 클러스터링 및 페일오버 기술을 제공합니다. 이 기능을 사용하면 확장 가능하고 고도로 활용 가능한 J2EE 응용 프로그램을 실행할 수 있습니다.

- **클러스터링 -** 클러스터는 하나의 논리 엔티티로 함께 작업하는 Application Server 인스턴스 그룹입니다. 클러스터의 모든 Application Server 인스턴스에는 동일한 구성과 동일한 응용 프로그램이 배포됩니다.

  Application Server 인스턴스를 클러스터에 추가하면 시스템 용량이 늘어나서 수평 확장을 할 수 있습니다. 서비스를 중단하지 않고 Application Server 인스턴스를 클러스터에 추가할 수 있습니다. HTTP, RMI/IIOP 및 JMS 로드 균형 조정 시스템에서는 클러스터 내에서 정상적으로 작동하는 Application Server 인스턴스로 요청을 배포합니다.
- **고가용성 -** 가용성으로 클러스터에 있는 Application Server 인스턴스의 페일오버 보호를 허용합니다. 어떤 Application Server 인스턴스가 중지될 경우 다른 Application Server 인스턴스가 중지된 서버에 지정되었된 세션을 처리합니다. 세션 정보는 HADB(high-availability database)에 저장됩니다. HADB는 HTTP 세션 및 Stateful Session Bean의 지속성을 지원합니다.

Application Server 구조

이 절에서는 Application Server의 고급 구조를 보여주는 그림 1-1을 설명합니다.

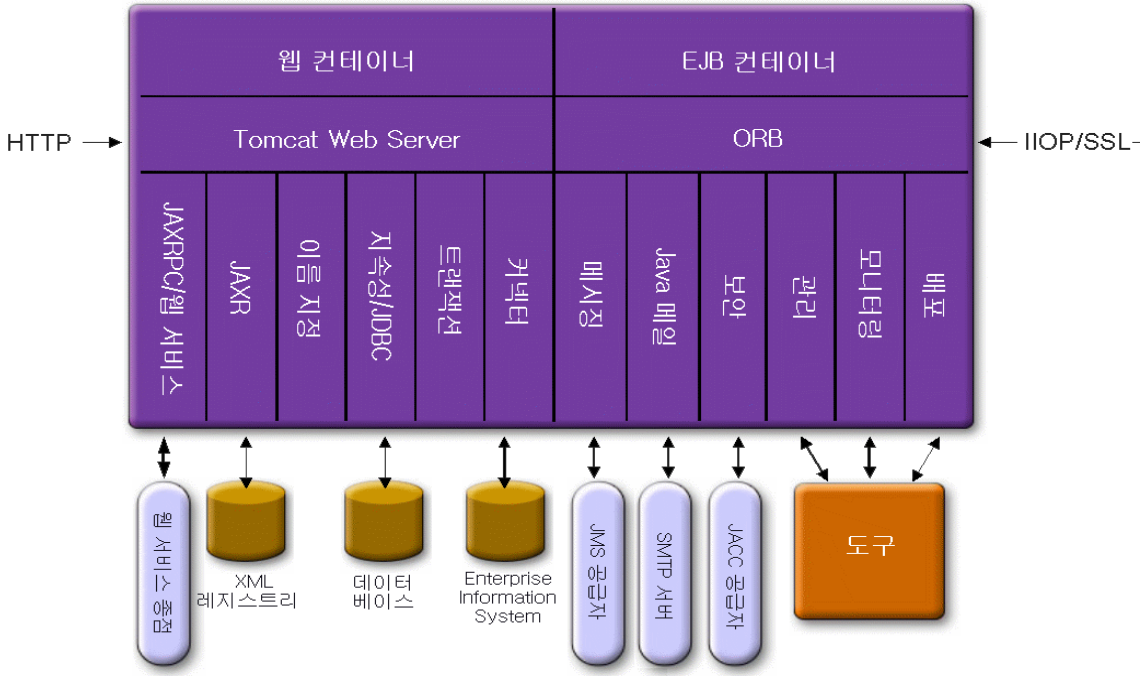


그림 1-1 Application Server 구조

- **컨테이너** - 컨테이너는 보안 및 트랜잭션 관리와 같은 서비스를 J2EE 구성 요소에 제공하는 런타임 환경입니다. 그림 1-1에서는 두 가지 유형의 J2EE 컨테이너, 즉 웹 및 EJB 컨테이너를 보여줍니다. JSP 페이지와 서블릿 같은 웹 구성 요소는 웹 컨테이너 내에서 실행됩니다. EJB 기술의 구성 요소인 Enterprise Bean은 EJB 컨테이너 내에서 실행됩니다.
- **클라이언트 액세스** - 런타임에서 브라우저 클라이언트는 인터넷에서 사용되는 프로토콜인 HTTP를 통해 웹 서버와 통신하여 웹 응용 프로그램에 액세스합니다. HTTPS 프로토콜은 보안 통신을 요구하는 응용 프로그램용입니다. Enterprise Bean 클라이언트는 IIOP 또는 IIOP/SSL(보안) 프로토콜을 통해 ORB(Object Request Broker)와 통신합니다. Application Server에는 HTTP, HTTPS, IIOP 및 IIOP/SSL 프로토콜용으로 개별 Listner가 있습니다. Listener마다 특정한 포트 번호를 독점적으로 사용합니다.

- **웹 서비스** - J2EE 플랫폼에서는 JAX-RPC(Java API for XML-Based RPC)에서 구현한 웹 서비스를 제공하는 웹 응용 프로그램을 배포할 수 있습니다. J2EE 응용 프로그램이나 구성 요소가 다른 웹 서비스에 대한 클라이언트가 될 수도 있습니다. 응용 프로그램은 JAXR(Java API for XML Registries)을 통해 XML 레지스트리에 액세스합니다.
- **응용 프로그램용 서비스** - J2EE 플랫폼은 컨테이너가 응용 프로그램용 서비스를 제공하도록 설계되었습니다. 그림 1-1에서는 다음과 같은 서비스를 보여줍니다.
  - 이름 지정 - 이름 지정 및 디렉토리 서비스는 객체를 이름에 바인딩합니다. J2EE 응용 프로그램은 객체의 JNDI 이름을 조회하여 객체를 찾습니다. JNDI는 Java Naming and Directory Interface API의 약자입니다.
  - 보안 - JACC(Java Authorization Contract for Containers)는 J2EE 컨테이너를 위해 정의된 보안 계약의 집합입니다. 클라이언트의 아이디를 기반으로 컨테이너는 컨테이너의 자원과 서비스에 대한 액세스를 제한합니다.
- **트랜잭션 관리** - 트랜잭션은 나눌 수 없는 작업 단위입니다. 예를 들어, 은행 계좌 간에 자금을 이체하는 것이 트랜잭션입니다. 트랜잭션 관리 서비스는 트랜잭션이 완전히 완료되거나 롤백되는 것을 보장합니다.

외부 시스템 액세스

J2EE 플랫폼을 사용하면 응용 프로그램이 Application Server 외부에 있는 시스템에 액세스할 수 있습니다. 응용 프로그램은 자원이라고 하는 객체를 통해 이 시스템에 연결합니다. 관리자의 책임 중 하나가 자원 구성입니다. J2EE 플랫폼을 사용하면 다음 API 및 구성 요소를 통해 외부 시스템에 액세스할 수 있습니다.

- **JDBC** - 데이터베이스 관리 시스템(DBMS)은 데이터의 저장, 구성 및 검색에 필요한 기능을 제공합니다. 대부분의 비즈니스 응용 프로그램은 관계형 데이터베이스에 데이터를 저장합니다. 응용 프로그램은 JDBC API를 통해 관계형 데이터베이스에 액세스합니다. 데이터베이스의 정보는 디스크에 저장되고 응용 프로그램이 종료된 후에도 존재하기 때문에 대개 지속성이 있다고 합니다. Application Server 번들에는 PointBase DBMS가 포함되어 있습니다.
- **메시징** - 메시징은 소프트웨어 구성 요소나 응용 프로그램간의 통신 방법입니다. 메시징 클라이언트는 다른 클라이언트와 메시지를 주고 받습니다. 응용 프로그램은 JMS(Java Messaging Service) API를 통해 메시징 공급자에 액세스합니다. Application Server에는 JMS 공급자가 포함됩니다.

- **커넥터** - J2EE 커넥터 구조를 사용하면 J2EE 응용 프로그램과 기존 EIS(Enterprise Information Systems) 간의 통합이 가능합니다. 응용 프로그램은 커넥터 또는 자원 어댑터라고 하는 이식 가능한 J2EE 구성 요소를 통해 EIS에 액세스합니다.
- **JavaMail** - JavaMail API를 통해 응용 프로그램은 SMTP 서버에 연결하여 전자 메일을 보내고 받습니다.
- **서버 관리** - 그림 1-1의 오른쪽 아래 모서리에서는 Application Server의 관리자가 수행한 작업의 일부를 보여줍니다. 예를 들어, 관리자는 응용 프로그램을 배포(설치)하고 서버의 성능을 모니터합니다. Application Server에서 제공하는 관리 도구를 사용하여 이 작업을 수행합니다.
- 관리 도구
- Application Server에는 다음 세 가지 관리 도구가 포함되어 있습니다.
  - 관리 콘솔
  - asadmin 유틸리티
  - AMX(Application Server Management Extension)

관리 콘솔

관리 콘솔은 쉽게 탐색할 수 있는 인터페이스와 온라인 도움말 기능을 제공하는 브라우저 기반의 도구입니다. 이 설명서에서는 관리 콘솔을 사용하는 데 필요한 단계별 지침을 제공합니다. 관리 콘솔을 사용하려면 Administration server를 실행해야 합니다.

Application Server를 설치한 경우 서버의 포트 번호를 선택했거나 기본 포트 4849를 사용했습니다. 또한 아이디와 마스터 비밀번호도 지정했습니다.

관리 콘솔을 시작하려면 웹 브라우저에 다음을 입력합니다.

```
http://hostname:port
```

예를 들면 다음과 같습니다.

```
https://kindness.sun.com:4949
```

Application Server가 설치된 시스템에서 관리 콘솔을 실행할 경우 호스트 이름으로 `localhost`를 지정합니다.

Windows의 경우 시작 메뉴에서 Application Server 관리 콘솔을 시작합니다.

설치 프로그램에서는 DAS(Domain Administration Server)와는 독립된 별개의 인스턴스뿐만 아니라 기본 포트 번호 4849를 사용하여 기본 관리 도메인(domain1로 이름 지정)을 작성합니다. 설치 후 추가 관리 도메인을 작성할 수 있습니다. 도메인마다 고유한 포트 번호를 갖고 있는 고유한 DAS(Domain Administration Server)가 있습니다. 관리 콘솔에 대한 URL을 지정할 때 반드시 관리할 도메인의 포트 번호를 사용하십시오.

구성에 원격 서버 인스턴스가 포함된 경우 원격 서버 인스턴스를 관리하고 사용을 용이하게 하려면 노드 에이전트를 작성합니다. 노드 에이전트의 역할은 서버 인스턴스를 작성, 시작, 중지 및 삭제하는 것입니다. 명령줄 인터페이스(CLI) 명령을 사용하여 노드 에이전트를 설정합니다.

asadmin 유틸리티

asadmin 유틸리티는 명령줄 도구입니다. asadmin 유틸리티 및 이와 연관된 명령을 사용하여 관리 콘솔에서 수행할 수 있는 동일한 작업 집합을 수행합니다. 예를 들어, 도메인을 시작 및 중지하고, 서버를 구성하며, 응용 프로그램을 배포합니다.

이러한 명령은 쉘의 명령 프롬프트에서 사용하거나 다른 스크립트 및 프로그램에서 호출합니다. 이러한 명령을 사용하여 반복적인 관리 작업을 자동화합니다.

asadmin 유틸리티를 시작하려면 다음 작업을 수행합니다.

```
$ asadmin
```

asadmin 내에서 사용 가능한 명령을 나열하려면 다음 작업을 수행합니다.

```
asadmin>help
```

쉘의 명령 프롬프트에서 asadmin 명령을 실행할 수도 있습니다.

```
$ asadmin help
```

명령의 구문과 예를 확인하려면 명령 이름 다음에 help를 입력합니다. 예를 들면 다음과 같습니다.

```
asadmin> help create-jdbc-resource
```

지정한 명령에 대한 asadmin help 정보는 명령의 Unix 설명서 페이지를 표시합니다. 이 설명서 페이지는 HTML 형식으로 사용할 수도 있습니다.

AMX(Application Server Management Extension)

Sun Java System AMX(**A**pplication Server **M**anagement e**X**tension)는 Application Server 구성을 모두 노출하고 JMX 관리 Bean을 AMX 인터페이스를 구현하는 사용하기 쉬운 클라이언트측 동적 프록시로 모니터링하는 API입니다.

AMX(Application Server Management Extension) 사용에 대한 자세한 내용은 *Sun Java System Application Server Developer's Guide*의 Using the Java Management Extensions (JMX) API 장을 참조하십시오.

Application Server 구성

Application Server 구성

Application Server 도메인은 관리자가 시스템 구성을 관리할 수 있도록 도와주기 위해 만든 논리적 또는 물리적 단위입니다. 도메인은 인스턴스와 노드 에이전트를 포함한 더 작은 단위로 세분됩니다. 서버 인스턴스는 단일 물리적 시스템에서 Application Server를 실행하는 단일 Java Virtual Machine(JVM)입니다. 도메인마다 하나 이상의 인스턴스가 있습니다. 인스턴스가 제대로 작동하려면 도메인에 최소한 하나의 연관된 노드 에이전트가 있어야 합니다. 도메인을 함께 묶어서 클러스터를 만들 수 있습니다. 클러스터를 사용하면 관리자가 하드웨어 및 소프트웨어를 그룹으로 관리할 수 있습니다.

도메인 구성

관리 도메인은 서로 다른 관리자가 특정한 그룹(도메인)의 Application Server 인스턴스를 관리할 수 있는 기본적인 보안 구조를 제공합니다. 서버 인스턴스를 별도의 도메인으로 그룹화하면 서로 다른 조직이나 관리자가 단일 Application Server 설치를 공유할 수 있습니다. 도메인마다 다른 도메인과 독립된 고유한 구성, 로그 파일 및 응용 프로그램 배포 영역이 있습니다. 한 도메인에 대한 구성을 변경해도 다른 도메인의 구성은 영향을 받지 않습니다.

관리 콘솔 세션을 사용하면 도메인을 구성 및 관리할 수 있습니다. 여러 도메인을 만든 경우 각 도메인을 관리하려면 추가 관리 콘솔 세션을 시작해야 합니다. 도메인마다 고유한 포트 번호를 가진 고유한 DAS(Domain Administration Server)가 있습니다. 관리 도메인마다 여러 Application Server 인스턴스가 있을 수 있습니다. 그러나 하나의 Application Server 인스턴스는 하나의 도메인에만 속할 수 있습니다. Application Server를 설치하면 `domain1`이라고 하는 관리 도메인이 자동으로 만들어집니다.

도메인 만들기

도메인은 `create-domain` 명령을 사용하여 작성합니다. 다음 예 명령은 `mydomain`이라고 하는 도메인을 만듭니다. 관리 서버는 포트 1234에서 청취하고 관리자 이름은 `hanan`입니다. 관리 비밀번호와 마스터 비밀번호를 묻는 명령 프롬프트가 나타납니다.

```
$ asadmin create-domain --adminport 80 --adminuser hanan mydomain
```

mydomain 도메인의 관리 콘솔을 시작하려면 브라우저에서 다음 URL을 입력합니다.

`http://`*`hostname`*`:80`

이전 `create-domain` 예의 경우 이제 도메인의 로그 파일, 구성 파일 및 배포된 응용 프로그램이 다음 디렉토리에 상주합니다.

`install_dir``/domains/mydomain`

도메인의 디렉토리를 다른 위치에 만들려면 `--domaindir` 옵션을 지정합니다. 명령의 전체 구문을 보려면 `asadmin help create-domain`을 입력합니다.

도메인 삭제

도메인은 asadmin delete-domain 명령을 사용하여 삭제합니다. 도메인을 관리할 수 있는 운영 체제 사용자(또는 루트)만 이 명령을 제대로 실행할 수 있습니다. 예를 들어, mydomain이라는 도메인을 삭제하려면 다음 명령을 입력합니다.

```
$ asadmin delete-domain mydomain
```

도메인 나열

시스템에 작성된 도메인은 asadmin list-domains 명령을 사용하여 확인할 수 있습니다. 기본 *install_dir*/domains 디렉토리에 있는 도메인을 나열하려면 다음 명령을 입력합니다.

```
$ asadmin list-domains
```

다른 디렉토리에 만든 도메인을 나열하려면 --domaindir 옵션을 지정합니다.

도메인 시작

도메인을 시작하면 관리 서버와 Application Server 인스턴스가 시작됩니다. Application Server 인스턴스가 시작되면 계속 실행되어 요청을 청취하고 수용합니다. 각 도메인을 별도로 시작해야 합니다.

도메인을 시작하려면 asadmin start-domain 명령을 입력하고 도메인 이름을 지정합니다. 예를 들어, 기본 도메인(domain1)을 시작하려면 다음을 입력합니다.

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

도메인이 하나만 있을 경우 도메인 이름을 생략합니다. 전체 명령 구문을 보려면 asadmin help start-domain을 입력합니다. 비밀번호 데이터를 생략한 경우 비밀번호를 제공하라는 메시지가 표시됩니다.

Windows에서 기본 도메인을 시작하려면

Windows 시작 메뉴에서 프로그램 -> Sun Microsystems -> Application Server -> 관리 서버 시작을 선택합니다.

서버나 도메인 다시 시작

서버를 다시 시작하는 것은 도메인을 다시 시작하는 것과 같습니다. 도메인이나 서버를 다시 시작하려면 도메인을 중지하고 시작합니다.

도메인 중지

도메인을 중지하면 관리 서버와 Application Server 인스턴스가 중지됩니다. 도메인을 중지하면 서버 인스턴스가 새로운 연결 승인을 중지하고 모든 진행 중인 연결이 완료될 때까지 기다립니다. 서버 인스턴스가 종료 과정을 완료해야 하기 때문에 이 과정은 시간이 걸립니다. 도메인을 중지하는 동안 관리 콘솔이나 asadmin 명령 대부분을 사용할 수 없습니다.

도메인을 중지하려면 asadmin stop-domain 명령을 입력하고 도메인 이름을 지정합니다. 예를 들어, 기본 도메인(domain1)을 중지하려면 다음을 입력합니다.

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

도메인이 하나만 있을 경우 도메인 이름은 선택 사항입니다. 전체 구문을 보려면 asadmin help stop-domain을 입력합니다.

관리 콘솔을 사용하여 도메인을 중지하려면

- 트리 구성 요소의 독립 실행형 인스턴스 노드에서 서버(관리 서버)를 선택합니다.
- 일반 정보 페이지에서 서버 중지를 누릅니다.

Windows에서 기본 도메인을 중지하려면

시작 메뉴에서 프로그램 -> Sun Microsystems -> Application Server -> 관리 서버 중지를 선택합니다.

Domain Administration Server 다시 만들기

미러링을 목적으로 DAS(Domain Administration Server)의 작업 복사본을 제공하려면 다음이 필요합니다.

- 원래 DAS를 포함하는 첫 번째 시스템(machine1)

- 응용 프로그램을 실행하고 클라이언트에 제공하는 서버 인스턴스가 있는 클러스터가 포함된 두 번째 시스템(machine2). DAS를 사용하여 첫 번째 시스템에 클러스터가 구성됩니다.
- 첫 번째 시스템에 문제가 있을 경우 DAS를 다시 만들어야 하는 세 번째 백업 시스템(machine3)

주 첫 번째 시스템에서 DAS의 백업을 보존해야 합니다. `asadmin backup-domain`을 사용하여 현재 도메인을 백업합니다.

DAS 마이그레이션 단계

첫 번째 시스템(machine1)에서 세 번째 시스템(machine3)으로 DAS(Domain Administration Server)를 마이그레이션하려면 다음 단계가 필요합니다.

1. 첫 번째 시스템에 설치된 것과 동일한 Application Server를 세 번째 시스템에 설치하여 세 번째 시스템을 설정합니다.

   DAS를 세 번째 시스템에 제대로 복구하고 경로 충돌을 방지하려면 이 단계가 필요합니다.

   - 명령줄(대화식) 모드를 사용하여 Application Server 관리 패키지를 설치합니다. 대화형 명령줄 모드를 활성화하려면 `console` 옵션을 사용하여 설치 프로그램을 호출하십시오.

     `./` *bundle_filename* `-console`

     명령줄 인터페이스를 사용하여 설치하려면 루트 권한이 있어야 합니다.

   - 기본 도메인을 설치하려면 옵션을 선택 해제합니다.

     백업한 도메인을 복구하는 것은 동일한 구조뿐만 아니라 **정확하게** 동일한 설치 경로를 가진 두 시스템(즉, 두 시스템에서 동일한 *install_dir* 사용)에서만 지원됩니다.

2. 첫 번째 시스템에서 백업 ZIP 파일을 세 번째 시스템의 *install_dir*`/domains`에 복사합니다. 파일을 FTP할 수도 있습니다.
3. `asadmin restore-domain` 명령을 실행하여 ZIP 파일을 세 번째 시스템에 복구합니다.

   ```
   asadmin restore-domain --filename
   install_dir/domains/sjsas_backup_v00001.zip domain1
   ```

모든 도메인을 백업할 수 있습니다. 그러나 도메인을 다시 만들 때 도메인 이름이 원본과 동일해야 합니다.

4. 세 번째 시스템의 *install_dir*/domains/domain1/generated/tmp 디렉토리 권한을 첫 번째 시스템에 있는 같은 디렉토리의 권한과 일치하도록 변경합니다.

   이 디렉토리의 기본 권한은 ?drwx------?(또는 700)입니다.

   예를 들면 다음과 같습니다.

   ```
   chmod 700 install_root/domains/domain1/generated/tmp
   ```

   위 예는 *domain1*을 백업하는 것을 가정합니다. 다른 이름으로 도메인을 백업할 경우 위 *domain1*을 백업할 도메인 이름으로 대체해야 합니다.

5. 세 번째 시스템의 경우 domain.xml의 등록 정보에 대한 호스트 값을 변경합니다.

6. 세 번째 시스템에서 *install_root*/domains/*domain1*/config/domain.xml을 업데이트합니다.

   예를 들면 다음과 같습니다.

   *machine1*을 찾아 *machine3*으로 바꿉니다. 그러면 다음을 변경할 수 있습니다.

   ```
   <jmx-connector><property name=client-hostname value=machine1/>...
   ```

   변경 후:

   ```
   <jmx-connector><property name=client-hostname value=machine3/>...
   ```

7. 다음을 변경합니다.

   ```
   <jms-service... host=machine1.../>
   ```

   변경 후:

   ```
   <jms-service... host=machine3.../>
   ```

8. machine3에서 복구된 도메인을 시작합니다.

   ```
   asadmin start-domain --user admin_user --password admin_password
   domain1
   ```

9. machine2의 nodeagent에서 등록 정보에 대한 DAS 호스트 값을 변경합니다.

10. machine2의 *install_dir*/nodeagents/*nodeagent*/agent/config/das.properties에 있는 agent.das.host 등록 정보 값을 변경합니다.

11. machine2에서 nodeagent를 다시 시작합니다.

주 `asadmin start-instance` 명령을 사용하여 클러스터 인스턴스를 시작하면 클러스터 인스턴스를 복구된 도메인과 동기화할 수 있습니다.

Application Server 인스턴스

Application Server 인스턴스 정보

Sun Java System Application Server는 설치 시 서버라고 하는 Application Server 인스턴스를 하나 만듭니다. 필요하면 서버 인스턴스를 삭제하고 다른 이름으로 새로운 인스턴스를 만들 수 있습니다.

Sun Java System Application Server 인스턴스마다 고유한 J2EE 구성, J2EE 자원, 응용 프로그램 배포 영역 및 서버 구성 설정이 있습니다. 한 Application Server 인스턴스를 변경하는 경우 다른 Application Server 인스턴스는 영향을 받지 않습니다. 한 관리 도메인 내에 많은 Application Server 인스턴스를 가질 수 있습니다.

대부분의 경우 Application Server 인스턴스는 하나만 있으면 됩니다. 하지만 환경에 따라 하나 이상의 추가 Application Server 인스턴스를 만들어야 할 수도 있습니다. 예를 들어, 개발 환경에서 다른 Application Server 인스턴스를 사용하여 다른 Sun ONE Application Server 구성을 테스트하거나 다른 응용 프로그램 배포를 비교 및 테스트할 수 있습니다. Application Server 인스턴스는 손쉽게 추가 또는 삭제할 수 있기 때문에 개발하는 동안 이러한 Application Server 인스턴스를 사용하면 실험에 사용할 임시 "샌드박스" 영역을 만들 수 있습니다.

또한 각 Application Server 인스턴스에 대해 가상 서버를 만들 수도 있습니다. 단일 설치 Application Server 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 Application Server 인스턴스에 걸쳐 있을 수 없습니다. 가상 서버에 대한 자세한 내용은 JVM 일반 설정 구성을 참조하십시오.

운영상 배포할 때 많은 용도로 여러 Application Server 인스턴스 대신 가상 서버를 사용할 수 있습니다. 그러나 가상 서버가 사용자 요구 사항을 충족시키지 못할 경우 여러 Application Server 인스턴스를 사용할 수도 있습니다.

Sun Java System Application Server 인스턴스는 자동으로 시작되지 않습니다. 인스턴스를 시작하면 중지할 때까지 인스턴스가 실행됩니다. Application Server 인스턴스를 중지하면 Application Server 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.

Application Server 인스턴스 정의

Application Server 인스턴스는 응용 프로그램 배포의 기본을 구성합니다. 각 인스턴스는 단일 도메인에 속하고 고유한 디렉토리 구조, 구성 및 배포된 응용 프로그램을 갖습니다. 또한 서버 인스턴스마다 J2EE 플랫폼 웹과 EJB 컨테이너도 포함합니다. 모든 새로운 서버 인스턴스에는 인스턴스가 상주하는 시스템을 정의하는 노드 에이전트 이름에 대한 참조가 포함되어야 합니다.

작성할 수 있는 서버 인스턴스 유형은 세 가지 입니다. 모든 서버 인스턴스는 다음 유형 중 하나입니다.

- **독립 실행형 서버** 인스턴스의 경우 다른 서버 인스턴스나 클러스터가 구성을 공유하지 않습니다.
- **공유 서버 인스턴스**의 경우 다른 인스턴스나 클러스터와 구성을 공유합니다.
- **클러스터링된 서버 인스턴스**의 경우 클러스터의 다른 인스턴스와 구성을 공유합니다.

그림 1-2에서는 Application Server 인스턴스를 자세히 보여줍니다. Application Server 인스턴스는 Application Server Enterprise Edition의 클러스터링, 로드 균형 조정 및 세션 지속성 기능의 블록 구축입니다.

- **클러스터 정의 및 사용** - 클러스터는 동일한 집합의 응용 프로그램, 자원 및 구성 정보를 공유하는 서버 인스턴스 그룹입니다. 서버 인스턴스는 하나의 클러스터에만 속할 수 있습니다. 무엇보다, 클러스터를 사용하면 여러 시스템 간 로드 배포를 통한 로드 균형 조정 및 인스턴스 수준 페일오버를 통한 고가용성이 가능합니다.

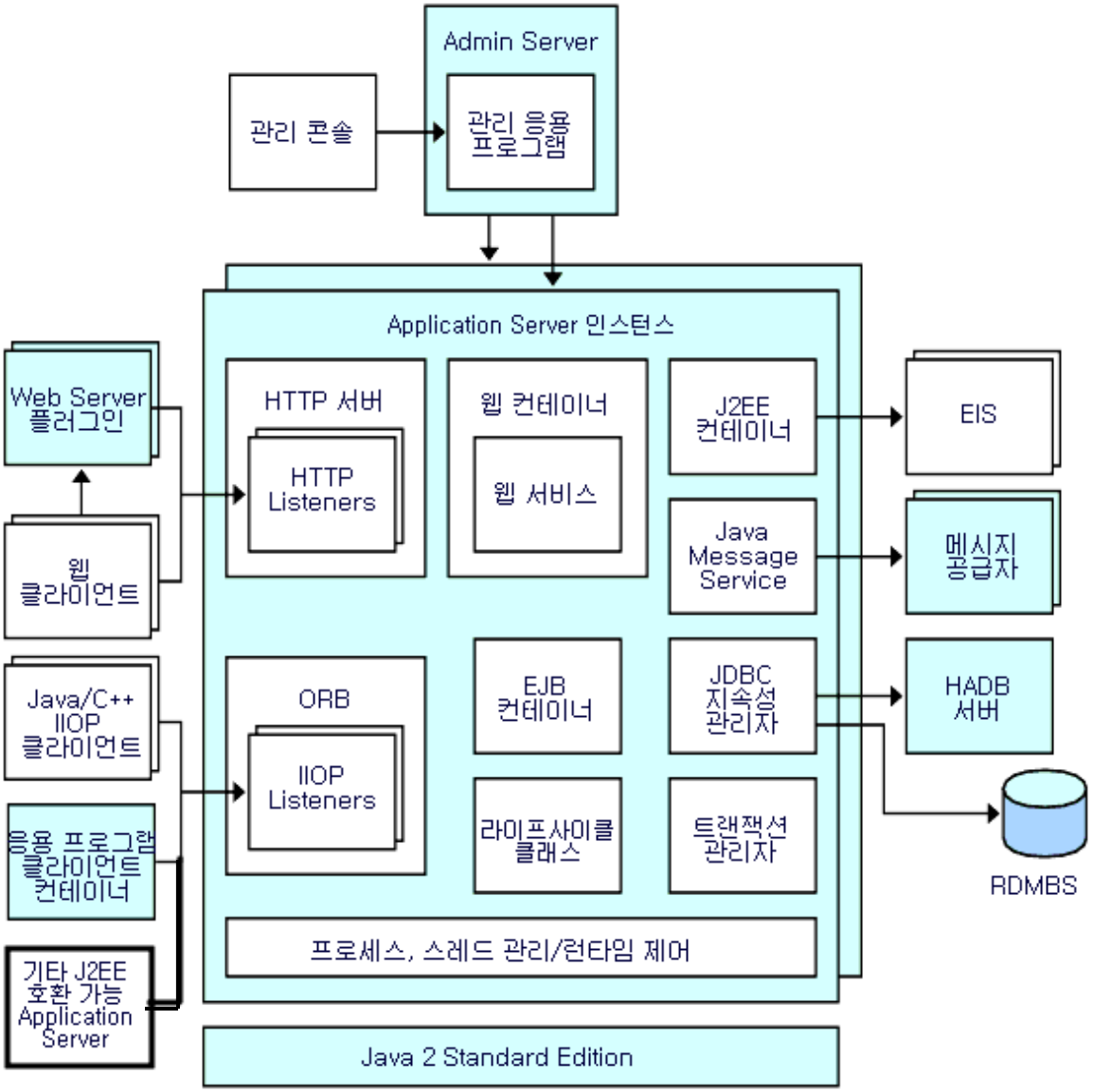


그림 1-2 Sun Java System Application Server 인스턴스

독립 실행형 인스턴스 정보

Sun Java System Application Server 인스턴스는 자동으로 시작되지 않습니다. 인스턴스를 시작하면 중지할 때까지 인스턴스가 실행됩니다. Application Server 인스턴스를 중지하면 Application Server 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.

일반 서버 정보 보기

일반 탭에서 다음 작업을 수행할 수 있습니다.

- 인스턴스 시작을 눌러 인스턴스를 시작합니다.
- 인스턴스 중지를 눌러 인스턴스를 중지합니다.
- 로그 파일 보기를 눌러 서버 로그 뷰어를 엽니다.
- 로그 파일 회전을 눌러 인스턴스의 로그 파일을 회전합니다. 이 작업은 로그 파일의 회전을 예약합니다. 다음 번에 로그 파일에 항목이 기록될 때 실제 회전이 발생합니다. 기본 서버(DAS)에 대해서는 회전이 즉시 발생하지만 다른 독립 실행형 서버에는 지연됩니다.
- JNDI 찾아보기를 눌러 실행 중인 인스턴스의 JNDI 트리를 검색합니다.
- 트랜잭션 복구를 눌러 미완료된 트랜잭션을 복구합니다.

그리고 다음 탭을 선택하여 추가 작업을 수행할 수 있습니다.

- 응용 프로그램 탭: 선택한 응용 프로그램을 배포합니다.
- 자원 탭: 선택한 자원을 관리합니다.
- 등록 정보 탭: 인스턴스 특정 등록 정보를 구성합니다.
- 모니터 탭: JVM, 서버, 스레드 풀, HTTP 서비스 및 트랜잭션 서비스에 대한 모니터링 데이터를 조회합니다.
- 고급 탭: 응용 프로그램을 배포하기 위한 일반 등록 정보를 설정합니다.

응용 프로그램 관리

응용 프로그램 탭에서 인스턴스와 연관된 선택된 응용 프로그램을 활성화, 비활성화 및 배포할 수 있습니다.

응용 프로그램을 배포하려면 다음 작업을 수행합니다.

1. 원하는 응용 프로그램의 확인란을 선택합니다.
2. 배포 드롭다운 메뉴에서 배포할 응용 프로그램 모듈 유형을 선택합니다.
   - 엔터프라이즈 응용 프로그램: EAR(Enterprise Application Archive) 파일이나 디렉토리의 J2EE 응용 프로그램
   - 웹 응용 프로그램: WAR(Web Application Archive) 파일이나 디렉토리에 패키지화된 JSP(JavaServer Page), 서블릿 및 HTML 페이지 같은 웹 자원 집합
   - EJB 모듈: EJB JAR(Java Archive) 파일이나 디렉토리에 포함된 하나 이상의 EJB(Enterprise JavaBean)
   - 커넥터 모듈: EIS(Enterprise Information System)에 연결되고 RAR(Resource Adapter Archive) 파일이나 디렉토리에 패키지화됩니다.
   - 라이프사이클 모듈: 서버 라이프사이클의 하나 이상의 이벤트에서 트리거한 작업을 수행합니다.
   - 응용 프로그램 클라이언트 모듈: J2EE 응용 프로그램 클라이언트 JAR 파일이라고도 하며 클라이언트용 서버측 루틴을 포함합니다.

자원 관리

자원 탭에서 새로운 자원 유형을 활성화, 비활성화 및 작성하여 인스턴스와 연관시킬 수 있습니다.

자원 유형을 새로 만들려면 다음 작업을 수행합니다.

1. 원하는 자원의 확인란을 선택합니다.
2. 새로 만들기 드롭다운 메뉴에서 작성하여 해당 인스턴스와 연관시킬 자원 유형을 선택합니다.
   - JDBC: 데이터베이스에 연결할 수 있는 수단을 응용 프로그램에 제공합니다.
   - 지속성 관리자: 컨테이너 관리 지속성 Bean이 있는 응용 프로그램에 필요합니다(역방향 호환성을 위해 필요).

- JMS 연결 팩토리: 응용 프로그램이 다른 JMS 객체를 프로그래밍 방식으로 작성할 수 있게 해주는 객체입니다.
- JMS 대상: 메일 및 메시징 응용 프로그램을 빌드하기 위해 플랫폼과 프로토콜에 독립적인 프레임워크를 제공하는 JavaMail API의 메일 세션을 나타냅니다.
- JavaMail: 메일 및 메시징 응용 프로그램을 빌드하기 위해 플랫폼과 프로토콜에 독립적인 프레임워크를 제공합니다.
- 사용자 정의: JNDI 하위 컨텍스트, 자원 유형 및 팩토리 클래스가 정의된 비표준 자원을 나타냅니다.
- 외부: LDAP(Lightweight Directory Access Protocol) 저장소에서 외부 자원 객체를 찾을 수 있도록 응용 프로그램을 활성화합니다.
- 커넥터: EIS(Enterprise Information System)와의 연결을 응용 프로그램에 제공하는 프로그램 객체입니다.
- 관리 객체: JSR-160 호환 원격 JMX 커넥터를 구성합니다.

관리 서버 고급 설정

관리 서버 고급 설정을 사용하면 응용 프로그램을 배포하는 데 필요한 일반 등록 정보를 설정할 수 있습니다. 이러한 일반 등록 정보를 사용하면 배포된 응용 프로그램에 대한 변경 사항을 감지하여 수정된 클래스를 다시 로드하는지 확인하고 모니터할 수 있습니다.

응용 프로그램 구성 설정

동적 재로드가 활성화된 경우 서버는 배포된 응용 프로그램의 파일의 변경 사항을 정기적으로 확인하고 변경 사항과 함께 응용 프로그램을 자동으로 다시 로드합니다. 동적 재로드는 코드 변경 사항을 바로 테스트할 수 있기 때문에 개발 환경에서 유용합니다. 그러나 작업 환경에서는 동적 재로드가 성능을 저하시킬 수 있습니다.

동적 재로드는 개발 환경을 위한 것입니다. 세션 지속성 기능인 작업 환경 기능과 호환되지 않습니다. 동적 배포가 활성화된 경우 세션 지속성을 활성화하지 마십시오.

주 동적 재로드는 기본 서버 인스턴스에 대해서만 사용할 수 있습니다.

응용 프로그램 구성 페이지에서 동적 재로드를 구성하려면 다음을 구성합니다.

- 재로드: 사용 확인란에서 동적 재로드를 활성화하거나 비활성화합니다.
- 다시 로드 폴링 간격: 서버가 배포된 응용 프로그램의 변경 사항을 확인하는 빈도를 지정합니다.
- 관리 세션 시간 초과: 얼마 후에 관리 세션이 시간 초과되고 다시 로그인해야 하는지를 지정합니다.

자동 배포 설정

자동 배포 기능을 사용하면 사전 패키지화된 응용 프로그램이나 모듈을 *install_dir*/domains/*domain_dir*/autodeploy 디렉토리에 복사하여 배포할 수 있습니다.

예를 들어, hello.war이라고 하는 파일을 *install_dir*/domains/domain1/autodeploy 디렉토리에 복사합니다. 응용 프로그램을 배포 해제하려면 autodeploy 디렉토리에서 hello.war 파일을 제거합니다.

자동 배포 기능은 개발 환경을 위한 것입니다. 세션 지속성 기능인 작업 환경 기능과 호환되지 않습니다. 동적 배포가 활성화된 경우 세션 지속성을 활성화하지 마십시오.

주 자동 배포는 기본 서버 인스턴스에 대해서만 사용할 수 있습니다.

응용 프로그램 구성 페이지에서 자동 배포 설정을 구성하려면 다음 작업을 수행합니다.

1. 사용 확인란을 선택하거나 선택 해제하여 자동 배포를 활성화하거나 비활성화합니다.
2. 자동 배포 폴링 간격 필드에서 서버가 응용 프로그램이나 모듈 파일의 자동 배포 디렉토리를 확인하는 빈도를 지정합니다. 폴링 간격을 변경해도 응용 프로그램이나 모듈을 배포하는 데 걸리는 시간에 영향을 미치지 않습니다.
3. 자동 배포 디렉토리에서 응용 프로그램을 빌드할 디렉토리를 지정한 경우 파일을 기본 자동 배포 디렉토리에 복사할 필요가 없습니다.

   기본값은 서버 인스턴스의 루트 디렉토리에 있는 autodeploy라고 하는 디렉토리입니다. 기본적으로 변수를 사용하여 여러 서버 인스턴스에 대한 디렉토리를 수동으로 변경할 필요성을 제거합니다.
4. 배포 전에 검증자를 실행하려면 검증자 사용 확인란을 선택합니다. 검증자는 파일의 구조와 내용을 검사합니다. 큰 응용 프로그램을 검증할 경우 시간이 많이 소모될 수 있습니다.

5. JSP 페이지를 사전 컴파일하려면 JSP 확인란을 선택합니다. 이 확인란을 선택하지 않으면, JSP 페이지에 처음 액세스할 때 런타임에서 컴파일됩니다. 컴파일에는 대개 시간이 소요되기 때문에 작업 환경에서는 이 확인란을 선택합니다.

추가 등록 정보 설정

등록 정보 추가 버튼을 눌러 추가 설정을 지정합니다.

도메인 속성 설정

다음 도메인 속성 등록 정보를 사용할 수 있습니다.

표 1-1 도메인 속성 값

등록 정보	정의
com.sun.aas.installRoot	Application Server가 설치된 디렉토리
com.sun.aas.instanceRoot	서버 인스턴스의 최상위 수준 디렉토리
com.sun.aas.hostName	호스트(시스템) 이름
com.sun.aas.javaRoot	.J2SE 설치 디렉토리
com.sun.aas.imqLib	Sun Java System Message Queue의 라이브러리 디렉토리
com.sun.aas.configName	서버 인스턴스에서 사용할 구성 이름
com.sun.aas.instanceName	서버 인스턴스 이름. 이 등록 정보는 default-config에서는 사용할 수 없으나 사용자 정의된 구성에서는 사용 가능합니다.
com.sun.aas.clusterName	클러스터 이름. 이 등록 정보는 클러스터링된 서버 인스턴스에만 설정됩니다. 이 등록 정보는 default-config에서는 사용할 수 없으나 사용자 정의된 구성에서는 사용 가능합니다.
com.sun.aas.domainName	도메인 이름. 이 등록 정보는 default-config에서는 사용할 수 없으나 사용자 정의된 구성에서는 사용 가능합니다.

인스턴스 특정 구성 등록 정보

인스턴스 특정 구성 등록 정보가 이 인스턴스의 값을 대체합니다.

주 기본값은 인스턴스에 바인딩된 구성에 정의됩니다.

기본값으로 값을 되돌리려면 다음 작업을 수행합니다.

1. 대체 값을 제거합니다.
2. 저장을 누릅니다.

대체 값을 설정하지 않은 경우 기본값을 사용합니다.

인스턴스 등록 정보 추가 또는 삭제

등록 정보를 추가하려면 다음 작업을 수행합니다.

- 등록 정보 추가 버튼을 눌러 추가 설정을 지정합니다.

자원을 구성하는 데 다음 등록 정보 속성 이름/값 쌍을 사용할 수 있습니다.

표 1-2 등록 정보 속성 이름/값 쌍

등록 정보	정의
HTTP_LISTENER_PORT	이 포트를 사용하여 HTTP 요청을 청취합니다. 이 등록 정보는 http-listener-1의 포트 번호를 지정합니다. 유효한 값은 1-65535입니다. UNIX의 경우 포트 1-1024에서 청취하는 소켓을 만들려면 수퍼유저 권한이 필요합니다.
HTTP_SSL_LISTENER_PORT	이 포트를 사용하여 HTTPS 요청을 청취합니다. 이 등록 정보는 http-listener-2의 포트 번호를 지정합니다. 유효한 값은 1-65535입니다. UNIX의 경우 포트 1-1024에서 청취하는 소켓을 만들려면 수퍼유저 권한이 필요합니다.
IIOP_LISTENER_PORT	이 등록 정보는 orb-listener-1이 청취하는 IIOP 연결의 ORB Listener 포트를 지정합니다.
IIOP_SSL_LISTENER_PORT	이 포트는 보안 IIOP 연결에 사용합니다.
JMX_SYSTEM_CONNECTOR_PORT	이 등록 정보는 JMX 커넥터가 청취하는 포트 번호를 지정합니다. 유효한 값은 1 - 65535입니다. UNIX의 경우 포트 1 - 1024에서 청취하는 소켓을 만들려면 수퍼유저 권한이 있어야 합니다.
IIOP_SSL_MUTUALAUTH_PORT	이 등록 정보는 SSL_MUTUALAUTH라고 하는 IIOP Listener가 청취하는 IIOP 연결의 ORB Listener 포트를 지정합니다.

등록 정보를 삭제하려면 다음 작업을 수행합니다.

1. 삭제할 등록 정보를 누릅니다.
2. 등록 정보 삭제 버튼을 누릅니다.

인스턴스 만들기

인스턴스를 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 독립 실행형 인스턴스 노드를 선택합니다.
2. 독립 실행형 서버 인스턴스 페이지에서 새로 만들기를 누릅니다.
3. 이름 필드에서 새 인스턴스의 고유한 이름을 확인합니다.
4. 노드 에이전트를 선택합니다.

   작성할 서버 인스턴스를 해당 노드 에이전트와 연관시킬 수 있도록 노드 에이전트의 호스트 시스템에서 `asadmin start-node-agent` 명령을 사용하여 노드 에이전트를 시작해야 합니다.
5. 원하는 구성을 선택합니다.
   - 기존 구성을 참조합니다. 새로운 구성이 추가되지 않습니다.
   - 기존 구성을 복사합니다. 서버 인스턴스나 클러스터를 추가하면 새로운 구성이 추가됩니다.
6. 기본적으로 default-config 구성에서 복사한 구성을 사용하여 새로운 구성이 만들어집니다. 다른 구성에서 복사하려면 새로운 인스턴스를 작성할 때 지정합니다.
7. 서버 인스턴스의 경우 새로운 구성 이름은 `instance_name-config`입니다.

   독립 실행형 서버 인스턴스를 만들기 위한 템플리트 역할을 하는 기본 구성이 default-config 구성입니다. 클러스터 해제된 서버 인스턴스나 클러스터는 default-config 구성을 참조할 수 없습니다. 이를 복사하여 새로운 구성을 만들 수만 있습니다. 복사한 새로운 구성에 올바른 초기 설정이 있도록 기본 구성을 편집합니다.

해당 `asadmin` 명령: `create-instance`

인스턴스 시작

인스턴스를 시작하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 독립 실행형 인스턴스 노드를 확장합니다.
2. 시작할 인스턴스를 선택합니다.
3. 일반 탭에서 시작을 눌러 인스턴스를 시작합니다.

인스턴스를 제대로 시작하려면 인스턴스와 연관된 노드 에이전트를 `asadmin start-node-agent` 명령을 사용하여 시작해야 합니다.

인스턴스를 시작하면 일반 탭에서 다음 작업을 수행할 수 있습니다.

- 인스턴스 중지를 눌러 인스턴스를 중지합니다.
- JNDI 찾아보기를 눌러 해당 인스턴스에 대한 JNDI 항목을 조회합니다.
- 로그 파일 보기를 눌러 로그 뷰어를 조회하고 로깅 옵션을 지정합니다.
- 로그 파일 회전을 누릅니다.
- 트랜잭션 복구를 눌러 미완료된 트랜잭션을 복구합니다.

해당 asadmin 명령: start-instance

트랜잭션 복구

서버나 자원 관리자의 문제로 인해 트랜잭션이 완료되지 않을 수 있습니다. 이 경우 문제가 있는 트랜잭션을 완료하고 실패를 복구해야 합니다. Application Server는 이 실패를 복구하고 서버 시작 시 트랜잭션을 완료하도록 설계되었습니다.

선택한 서버를 실행하면 동일한 서버에서 복구를 수행합니다. 선택한 서버를 실행하지 않을 경우 선택한 대상 서버에서 복구를 수행합니다.

인스턴스 중지

인스턴스를 중지하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 독립 실행형 인스턴스 노드를 확장합니다.
2. 중지할 인스턴스를 선택합니다.
3. 일반 탭에서 중지를 눌러 인스턴스를 중지합니다.

해당 asadmin 명령: stop-instance

서버 인스턴스 종료

관리 서버를 종료하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 독립 실행형 인스턴스 노드를 선택합니다.
2. 관리 서버 인스턴스를 선택합니다.
3. 중지를 누릅니다.

관리 서버를 종료할 것인지 확인하는 확인 대화 상자가 표시됩니다.

구성 변경

Application Server 구성 변경

다음 구성 변경을 적용하려면 서버를 다시 시작합니다.

- JVM 옵션 변경
- 포트 번호 변경
- HTTP, IIOP 및 JMS 서비스 관리
- 스레드 풀 관리

자세한 내용은 서버나 도메인 다시 시작을 참조하십시오.

동적 구성의 경우 서버를 실행하는 중에 대부분 변경 사항이 적용됩니다. 다음과 같은 구성 변경의 경우 서버를 다시 시작하지 *마십시오*.

- 응용 프로그램 배포 및 배포 해제
- JDBC, JMS, 커넥터 자원 및 풀 추가 또는 제거
- 로깅 수준 변경
- 파일 영역 사용자 추가

- 모니터링 수준 변경
- 자원 및 응용 프로그램 활성화 및 비활성화

`asadmin reconfig` 명령은 더 이상 사용되지 않으므로 필요하지 않습니다. 구성 변경 사항이 서버에 동적으로 적용됩니다.

Application Server 포트

표 1-3에서는 Application Server의 포트 Listener를 설명합니다.

표 1-3 포트를 사용하는 Application Server Listener

Listener	기본 포트 번호	설명
관리 서버	4848	관리 콘솔과 asadmin 유틸리티에서 도메인의 관리 서버에 액세스합니다. 관리 콘솔의 경우 브라우저의 URL로 포트 번호를 지정합니다. asadmin 명령을 원격으로 실행할 경우 --port 옵션을 사용하여 포트 번호를 지정합니다.
HTTP	8081	웹 서버는 포트에서 HTTP 요청을 청취합니다. 배포된 웹 응용 프로그램에 액세스하기 위해 클라이언트가 이 포트에 연결합니다.
HTTPS	8181	보안 통신을 위해 구성된 웹 응용 프로그램은 별도의 포트에서 청취합니다.
IIOP		Enterprise Bean의 원격 클라이언트(EJB 구성 요소)는 IIOP listener를 통해 Bean에 액세스합니다.
IIOP, SSL		다른 포트는 보안 통신을 위해 구성된 IIOP listener에서 사용합니다.
IIOP, SSL 및 상호 인증		다른 포트는 상호(클라이언트 및 서버) 인증을 위해 구성된 IIOP listener에서 사용합니다.

포트 번호 보기

1. 트리 구성 요소의 독립 실행형 인스턴스 노드에서 인스턴스를 선택합니다.
2. 등록 정보 탭을 선택합니다.
3. 인스턴스 특정 페이지에서 기본 포트 번호가 식별됩니다. 이 값을 대체하도록 구성을 설정할 수 있습니다.

관리 서버 포트 변경

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. server-config(관리 구성) 노드를 확장합니다.
3. HTTP 서비스 노드를 확장합니다.
4. HTTP Listener 노드를 확장합니다.
5. admin-listener 노드를 선택합니다.
6. HTTP Listener 편집 페이지에서 Listener 포트 필드의 값을 변경합니다.
7. 서버를 다시 시작합니다.

HTTP 포트 변경

1. 트리 구성 요소에서 HTTP 서비스 노드를 확장합니다.
2. HTTP Listener 노드를 확장합니다.
3. 포트 번호를 변경할 HTTP listener를 선택합니다.
4. HTTP Listener 편집 페이지에서 Listener 포트 필드의 값을 변경합니다.
5. 저장을 누릅니다.
6. 서버를 다시 시작합니다.

IIOP 포트 변경

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. server-config(관리 구성) 노드를 확장합니다.
3. ORB 노드를 확장합니다.
4. IIOP Listener 노드를 확장합니다.
5. 포트 번호를 변경할 Listener를 선택합니다.
6. IIOP Listener 편집 페이지에서 Listener 포트 필드의 값을 변경합니다.
7. 저장을 누릅니다.
8. 서버를 다시 시작합니다.

관리 서비스를 사용하여 JMX 커넥터 구성

관리 서비스를 사용하여 원격 서버 인스턴스에 맞게 JSR-160 호환 원격 JMX 커넥터를 구성함으로써 호스트 시스템에서 서버 인스턴스를 관리합니다. 이 커넥터는 DAS(Domain Administration Server)와 노드 에이전트간의 통신을 처리합니다.

서버 인스턴스가 정규 인스턴스, DAS(Domain Administration Server) 또는 조합인지 여부를 관리 서비스가 결정합니다. 사용자 응용 프로그램 요청을 처리할 수 있지만 사용자 응용 프로그램과 자원을 DAS에 배포하지 않는 점만 제외하고 DAS는 J2EE 서버 인스턴스와 유사합니다. DAS와 J2EE 서버 인스턴스간의 중요한 차이점은 DAS는 동종 서버 인스턴스 단위인 클러스터의 일부가 될 수 없다는 점입니다.

JMX 커넥터를 구성하려면 다음 작업을 수행합니다.

1. 트리에서 구성을 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 인스턴스의 config 노드를 선택합니다. 예를 들어, 기본 인스턴스인 서버의 경우 server-config 노드를 선택합니다.
   b. default-config의 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 트리에서 관리 서비스를 선택합니다.
4. 유형 드롭다운 메뉴에서 관리 서비스가 구성할 항목, 즉 DAS , DAS 및 서버, 또는 서버를 선택합니다. DAS 및 서버를 선택하는 것은 DAS를 선택하는 것과 같습니다. 서버 선택에서 비DAS 서버 인스턴스를 선택합니다.
5. JMX 커넥터 이름 필드에서 내부적으로 사용한 JMX 커넥터 이름을 입력합니다. 커넥터 이름은 시스템입니다.

JMX 커넥터 구성 편집

JMX 커넥터 편집 화면을 사용하면 JSR 160 호환 JMX 커넥터의 구성을 편집할 수 있습니다.

1. 트리에서 구성을 선택합니다.
2. 구성할 인스턴스를 선택합니다.

a. 특정 인스턴스를 구성하려면 인스턴스의 config 노드를 선택합니다. 예를 들어, 기본 인스턴스인 서버의 경우 server-config 노드를 선택합니다.

b. default-config의 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면, default-config 노드를 선택합니다.

3. 관리 서비스 노드를 확장하고 시스템을 누릅니다. 이 시스템은 내부적으로 사용된 JMX 커넥터입니다.
4. JMX 커넥터 서버의 포트를 입력합니다. JMX 서비스 URL은 JSR 160 1.0 사양에서 정의한 프로토콜, 포트 및 주소의 함수입니다.
5. 이 JMX 커넥터에서 지원해야 할 프로토콜을 입력합니다. Application Server 버전 8.1은 rmi_jrmp 프로토콜만 지원합니다.
6. 영역 이름 필드에서 특수한 관리 영역을 나타내는 이름을 입력합니다. 이 영역에서 모든 인증이 처리됩니다.
7. 사용 확인란을 선택하여 JMX 커넥터에서 전송 계층 보안을 사용해야 함을 표시합니다. 전송 계층 보안을 활성화한 경우 관리 서비스의 JMX 커넥터에 대한 보안 구성의 지침에 따라 SSL 섹션을 입력합니다.

J2SE 소프트웨어 변경

Application Server는 J2SE(Java 2 Standard Edition) 소프트웨어를 사용합니다. Application Server를 설치한 경우 J2SE 소프트웨어의 디렉토리가 지정됩니다. J2SE 소프트웨어 변경에 대한 지침은 JVM 일반 설정 구성을 참조하십시오.

온라인 도움말 사용

관리 콘솔의 온라인 도움말은 상황에 맞는 도움말입니다. 오른쪽 위 모서리에 있는 도움말 링크를 누르면 도움말 브라우저 창에서 현재 관리 콘솔 페이지와 관련된 항목을 표시합니다. 현재 페이지에 해당되는 도움말 정보가 없을 경우 온라인 도움말 사용 항목이 표시됩니다.

온라인 도움말에는 상황과 무관한 개념적인 항목도 있습니다. 이런 항목을 보려면 도움말 브라우저 창의 목차에서 항목을 선택합니다.

이전 도움말 화면으로 돌아가려면 다음 작업을 수행합니다.

1. 도움말 브라우저 창 내에서 마우스 오른쪽 버튼을 눌러 선택 메뉴를 표시합니다.
2. 뒤로를 선택합니다.

찾고 있는 정보를 찾을 수 없으면 다음 위치에서 온라인으로 사용 가능한 Application Server 관리 설명서를 확인하십시오.

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

추가 정보

- Sun Microsystems Wordwide Training - 60개국 이상에 있는 250개 이상의 교육 센터에서 웹 기반 과정을 통해 매년 250,000 이상의 학생이 Sun이나 공인된 센터에서 교육을 받고 있습니다. 자세한 내용은 다음을 참조하십시오.

  ```
  http://training.sun.com/
  ```

- *J2EE 1.4 Tutorial* - 개발자를 위해 작성된 자습서에는 JMS를 구성하고 JavaMail 자원을 설정하며 보안을 관리하는 데 필요한 관리 지침이 있습니다. 자습서에 액세스하려면 다음 URL로 이동합니다.

  ```
  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
  ```

- *Application Server Developer's Guide* - 이 설명서에는 Application Server에 관련된 개발 정보가 포함되어 있습니다. Developer's Guide는 다음 위치에서 사용할 수 있습니다.

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

- `asadmin` 설명서 페이지 - HTML 형식으로 사용 가능한 이 페이지에는 `asadmin` 유틸리티 명령을 포함한 모든 Application Server 유틸리티에 대한 구문과 예가 포함되어 있습니다. 이 HTML 페이지는 다음 URL에 게시되어 있습니다.

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

- *Application Server 릴리스 노트* - 다음 위치에서 온라인으로 사용 가능합니다.

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

- *J2EE Connectors Getting Started* - 이 설명서에는 커넥터(자원 어댑터), 연결 풀 및 커넥터 자원을 구성하는 데 필요한 지침이 있습니다.

  ```
  http://java.sun.com/j2ee/connector/
  ```

- docs.sun.com: Sun 제품 설명서 - 이 사이트에서 당사의 모든 제품 설명서를 검색 및 액세스할 수 있습니다.

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

- J2EE 1.4 설명서 페이지 - 당사의 공개 웹 사이트에 있는 이 페이지에는 J2EE 1.4 플랫폼용 기술 문서에 대한 링크가 있습니다.

  `http://java.sun.com/j2ee/1.4/docs/`

- *빠른 시작 설명서* - 이 문서에서는 단순한 웹 응용 프로그램을 배포 및 실행하는 방법을 보여줍니다. 이 설명서는 *install_dir*`/docs/QuickStart.html` 파일에 있습니다.

클러스터 구성

이 장에서는 관리 콘솔을 사용하여 클러스터를 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

클러스터 정보

클러스터

클러스터는 하나의 논리 엔티티로 함께 작동하는 0개 이상의 서버 인스턴스 모음입니다. 클러스터는 하나 이상의 J2EE 응용 프로그램에 런타임 환경을 제공합니다.

Sun Java System Application Server Enterprise Edition 8.1 환경에서 클러스터를 사용하면 다음을 제공합니다.

- 고가용성. 클러스터 내 서버 인스턴스에 대한 페일오버 보호를 허용하여 가용성이 높습니다. 한 서버 인스턴스가 중지되면 다른 서버 인스턴스에서 사용 불가능한 서버 인스턴스가 처리하면 요청을 넘겨 받습니다.

- 확장성. 서버 인스턴스를 클러스터에 추가할 수 있어 시스템 용량을 늘릴 수 있습니다. Application Server의 로드 밸런서가 클러스터 내 사용 가능한 서버 인스턴스로 요청을 배포합니다. 관리자가 서버 인스턴스를 클러스터에 추가하므로 서비스를 중단할 필요가 없습니다.

클러스터에는 다음과 같은 특성이 있습니다.

- 클러스터의 모든 인스턴스는 동일한 구성을 참조합니다.
- 클러스터의 모든 인스턴스에는 같은 응용 프로그램 집합이 배포되어 있습니다(예: J2EE 응용 프로그램 EAR 파일, 웹 모듈 WAR 파일 또는 EJB JAR 파일).
- 클러스터의 모든 인스턴스에는 동일한 자원 집합이 있으므로 동일한 JNDI 이름 공간이 만들어집니다.

도메인의 모든 클러스터에는 고유한 이름이 있습니다. 그리고 이 이름은 모든 노드 에이전트 이름, 서버 인스턴스 이름 및 구성 이름 모두에 대해 고유해야 합니다. 이름은 `domain`이 아니어야 합니다. 관리자는 클러스터링되지 않은 서버 인스턴스에서 수행하는 것과 동일한 작업(예: 응용 프로그램 배포 및 자원 작성)을 클러스터에서 수행합니다.

클러스터, 노드 에이전트 및 서버 인스턴스 사용에 대한 개요는 *Deployment Planning Guide*를 참조하십시오. 로드 밸런서에 대한 자세한 내용은 로드 균형 조정 및 페일오버 구성을 참조하십시오.

클러스터 유형

클러스터에는 *독립 실행형 클러스터*와 *공유 클러스터*의 두가지 유형이 있습니다.

- 독립 실행형 클러스터는 다른 서버 인스턴스나 클러스터와 공유하지 않는 고유한 구성을 가지고 있습니다. 기본적으로 이 구성의 이름은 *cluster_name*`-config`입니다. 여기서 *cluster_name*은 클러스터의 이름을 나타냅니다.
- 공유 클러스터는 하나 이상의 다른 클러스터나 클러스터링되지 않은 인스턴스와 구성을 공유합니다.

클러스터, 인스턴스, 로드 밸런서 및 세션

클러스터, 서버 인스턴스, 로드 밸런서 및 세션은 다음과 같이 Application Server와 관련되어 있습니다.

- 서버 인스턴스가 클러스터의 일부일 필요는 없습니다. 그러나 클러스터의 일부가 아닌 인스턴스는 세션 간의 세션 상태 전송을 통한 고가용성을 이용할 수 없습니다.
- 클러스터 내 서버 인스턴스는 다른 시스템이나 동일한 시스템에서 호스트할 수 있습니다. 즉 다른 시스템의 서버 인스턴스를 한 클러스터로 묶을 수 있습니다.
- 특정한 로드 밸런서는 요청을 여러 클러스터의 서버 인스턴스로 전달할 수 있습니다. 로드 밸런서의 이 기능을 사용하면 서비스 손실없이 온라인 업그레이드를 수행할 수 있습니다. 자세한 내용은 60페이지의 "여러 클러스터를 사용한 서비스 손실 없는 온라인 업그레이드"를 참조하십시오.
- 한 클러스터에서 여러 로드 밸런서로부터 요청을 수신할 수 있습니다. 클러스터를 여러 로드 밸런서에서 사용할 경우 각 로드 밸런서에서 동일한 방법으로 클러스터를 구성해야 합니다.
- 각 세션은 특정 클러스터에 연결됩니다. 이는 여러 클러스터에서 응용 프로그램을 배포할 수 있지만 세션은 같은 클러스터 내 서버 인스턴스에만 페일오버됨을 의미합니다.
- Application Server에서는 HTTP 세션과 SFSB(Stateful Session Bean) 세션에 대한 페일오버를 지원합니다. HTTP 세션 내에 저장된 특정 J2EE 객체 참조의 페일오버도 지원됩니다.

  HTTP 세션의 페일오버에 대한 자세한 내용은 147페이지의 "가용성 및 세션 지속성 구성"을 참조하십시오.

클러스터는 클러스터 내 서버 인스턴스의 세션 페일오버에 대한 안전한 경계 역할을 합니다. 로드 밸런서를 사용하고Application Server 내 구성 요소를 서비스 손실없이 업그레이드할 수 있습니다. 자세한 내용은 60페이지의 "여러 클러스터를 사용한 서비스 손실 없는 온라인 업그레이드"를 참조하십시오.

클러스터의 관리 콘솔 작업

- EJB 타이머 마이그레이션
- 클러스터에 대한 서버 인스턴스 만들기
- 클러스터링된 서버 인스턴스 구성
- 클러스터에 대한 응용 프로그램 구성
- 클러스터에 대한 자원 구성
- 클러스터링 삭제
- 여러 클러스터를 사용한 서비스 손실 없는 온라인 업그레이드

클러스터 만들기

클러스터를 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 클러스터 노드를 선택합니다.
2. 클러스터 페이지에서 새로 만들기를 누릅니다. 클러스터 만들기 페이지가 표시됩니다.
3. 이름 필드에서 클러스터 이름을 입력합니다. 이름은
   - 대소문자, 숫자, 밑줄, 하이픈 및 마침표(.)만 포함할 수 있습니다.
   - 모든 노드 에이전트 이름, 서버 인스턴스 이름, 클러스터 이름 및 구성 이름에서 고유해야 합니다.
   - `domain`이 아니어야 합니다.
4. 구성 필드의 드롭다운 목록에서 원하는 구성을 선택합니다.

   독립 실행형 클러스터를 만들려면 `default-config`를 선택하고 "선택한 구성의 복사본을 만듭니다" 라디오 버튼을 선택한 상태로 둡니다. 기본 구성의 복사본 이름은 *cluster_name*`-config`입니다.

   다른 구성을 참조하려면 드롭다운 목록에서 원하는 구성을 선택하고 "선택한 구성을 참조합니다" 라디오 버튼을 선택합니다. 이렇게 하면 다른 클러스터에서 구성을 사용할 경우 공유 클러스터가 만들어집니다.
5. 이제 서버 인스턴스를 추가하거나 클러스터가 만들어질 때까지 기다립니다. 클러스터에 대한 서버 인스턴스를 만들기 전에 먼저 하나 이상의 노드 에이전트나 노드 에이전트 자리 표시자를 만듭니다. 자세한 내용은 102페이지의 "노드 에이전트 자리 표시자 만들기"를 참조하십시오.

   서버 인스턴스를 만들려면 다음 단계를 수행합니다.

a. 만들 서버 인스턴스 영역에서 추가를 누릅니다.

b. 인스턴스 이름 필드에서 인스턴스 이름을 입력합니다.

c. 노드 에이전트 드롭다운 목록에서 원하는 노드 에이전트를 선택합니다.

6. 확인을 누릅니다.

7. 표시되는 "클러스터를 성공적으로 만들었습니다" 페이지에서 확인을 누릅니다.

클러스터, 서버 인스턴스 및 노드 에이전트 관리 방법에 대한 자세한 내용은 95페이지의 "노드 에이전트 배포"를 참조하십시오.

해당 asadmin 명령: create-cluster

클러스터 구성

클러스터를 구성하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 클러스터 노드를 확장합니다.

2. 원하는 클러스터 노드를 선택합니다. 일반 정보 페이지에서 다음 작업을 수행할 수 있습니다.

   - 인스턴스 시작을 눌러 클러스터링된 서버 인스턴스를 시작합니다.
   - 인스턴스 중지를 눌러 클러스터링된 서버 인스턴스를 중지합니다.
   - EJB 타이머 마이그레이션을 눌러 EJB 타이머를 중지된 서버 인스턴스에서 클러스터 내 다른 서버 인스턴스로 마이그레이션합니다.

해당 asadmin 명령: start-cluster, stop-cluster, migrate-timers

EJB 타이머 마이그레이션

서버 인스턴스가 비정상적으로 또는 예기치 않게 중단되었을 경우 해당 서버 인스턴스에 설치된 EJB 타이머를 클러스터 내 실행 중인 다른 서버 인스턴스로 이동해야할 수도 있습니다. 그렇게 하려면 다음 단계를 수행합니다.

1. 소스 드롭다운 목록에서 타이머를 마이그레이션할 중지된 서버 인스턴스를 선택합니다.

2. 대상 드롭다운 목록에서 타이머를 마이그레이션할 실행 중인 서버 인스턴스를 선택할 수도 있습니다. 이 필드를 공백으로 비워 두면 실행 중인 인스턴스가 임의로 선택됩니다.

3. 확인을 누릅니다.

4. 대상 서버 인스턴스를 중지했다가 다시 시작합니다.

소스 서버 인스턴스가 실행 중이거나 대상 서버 인스턴스가 실행되지 않을 경우 오류 메시지가 표시됩니다.

해당 asadmin 명령: `migrate-timers`

클러스터에 대한 서버 인스턴스 만들기

클러스터에 대한 서버 인스턴스를 만들려면 먼저 노드 에이전트나 노드 에이전트 자리 표시자를 만들어야 합니다. 자세한 내용은 102페이지의 "노드 에이전트 자리 표시자 만들기"를 참조하십시오.

클러스터에 대한 서버 인스턴스를 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 클러스터 노드를 확장합니다.
2. 원하는 클러스터 노드를 선택합니다.
3. 인스턴스 탭을 눌러 클러스터링된 서버 인스턴스 페이지를 표시합니다.
4. 새로 만들기를 눌러 클러스터링된 서버 인스턴스 만들기 페이지를 표시합니다.
5. 이름 필드에서 서버 인스턴스 이름을 입력합니다.
6. 노드 에이전트 드롭다운 목록에서 노드 에이전트를 선택합니다.
7. 확인을 누릅니다.

해당 asadmin 명령: `create-instance`

클러스터링된 서버 인스턴스 구성

클러스터링된 서버 인스턴스를 만든 후 변경하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 클러스터 노드를 확장합니다.
2. 서버 인스턴스를 포함하는 클러스터의 노드를 확장한 다음 편집할 서버 인스턴스 노드를 선택합니다.
3. 일반 정보 페이지에서 다음 작업을 수행할 수 있습니다.
   - 인스턴스 시작을 눌러 인스턴스를 시작합니다.
   - 인스턴스 중지를 눌러 실행 중인 인스턴스를 중지합니다.
   - JNDI 찾아보기를 눌러 실행 중인 인스턴스의 JNDI 트리를 검색합니다.

- 로그 파일 보기를 눌러 서버 로그 뷰어를 엽니다.
- 로그 파일 회전을 눌러 인스턴스의 로그 파일을 회전합니다. 이 작업을 수행하면 회전할 로그 파일이 예약됩니다. 다음 번에 로그 파일에 항목이 기록될 때 실제 회전이 발생합니다.
- 트랜잭션 복구를 눌러 불완전한 트랜잭션을 복구합니다.
- 등록 정보 탭을 눌러 인스턴스의 포트 번호를 수정합니다.
- 모니터 탭을 눌러 모니터링 등록 정보를 변경합니다.

다음과 같이 서버 인스턴스에서 작업을 수행할 수도 있습니다.

1. 트리 구성 요소에서 클러스터 노드를 확장합니다.
2. 서버 인스턴스를 포함하는 클러스터의 노드를 확장합니다.
3. 인스턴스 탭을 눌러 클러스터링된 서버 인스턴스 페이지로 이동합니다. 이 페이지에서 다음 작업을 수행할 수 있습니다.
   - 인스턴스의 확인란을 선택하고 삭제, 시작 또는 중지를 누릅니다.
   - 인스턴스 이름을 눌러 일반 정보 페이지를 표시합니다.

클러스터링된 서버 인스턴스를 클러스터에서 삭제하려면 해당 인스턴스 이름 옆에 있는 확인란을 선택하고 삭제를 누릅니다.

클러스터에 대한 응용 프로그램 구성

클러스터에 대한 응용 프로그램을 구성하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 클러스터 노드를 확장합니다.
2. 원하는 클러스터 노드를 선택합니다.
3. 응용 프로그램 탭을 눌러 응용 프로그램 페이지를 표시합니다. 이 페이지에서 다음 작업을 수행할 수 있습니다.
   - 응용 프로그램 옆에 있는 확인란을 선택하고 사용 가능이나 사용 불가능을 선택하여 클러스터에 대해 해당 응용 프로그램을 활성화하거나 비활성화합니다.
   - 배포 드롭다운 목록에서 배포할 응용 프로그램 유형을 선택합니다. 표시되는 배포 페이지에서 응용 프로그램을 지정합니다.
   - 필터 드롭다운 목록에서 목록에 표시할 응용 프로그램 유형을 선택합니다.

응용 프로그램을 편집하려면 응용 프로그램 이름을 누릅니다.

클러스터에 대한 자원 구성

클러스터에 대한 자원을 구성하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 클러스터 노드를 확장합니다.
2. 원하는 클러스터 노드를 선택합니다.
3. 자원 탭을 눌러 자원 페이지를 표시합니다. 이 페이지에서 다음 작업을 수행할 수 있습니다.
    - 자원 옆에 있는 확인란을 선택하고 사용 가능 또는 사용 불가능을 눌러 자원을 전역적으로 활성화하거나 비활성화합니다. 이렇게 해도 자원이 제거되지 않습니다.
    - 새로 만들기 드롭다운 목록에서 만들 자원 유형을 선택합니다. 자원을 만들 때 클러스터를 대상으로 지정해야 합니다.
    - 필터 드롭다운 목록에서 목록에 표시할 자원 유형을 선택합니다.

자원을 편집하려면 자원 이름을 누릅니다.

클러스터링 삭제

클러스터를 삭제하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 클러스터 노드를 선택합니다.
2. 클러스터 페이지에서 삭제할 클러스터 이름 옆에 있는 확인란을 선택합니다.
3. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-cluster`

여러 클러스터를 사용한 서비스 손실 없는 온라인 업그레이드

로드 밸런서와 여러 클러스터를 사용하여 서비스 손실 없이 Application Server 내에서 구성 요소를 업그레이드할 수 있습니다. 예를 들어, 구성 요소로는 JVM, Application Server 또는 웹 응용 프로그램 등이 있습니다.

다음 작업을 수행합니다.

1. 클러스터의 일반 정보 페이지에 있는 클러스터 중지 버튼을 사용하여 클러스터 중 하나를 중지합니다.
2. 해당 클러스터의 구성 요소를 업그레이드합니다.
3. 클러스터의 일반 정보 페이지에 있는 클러스터 시작 버튼을 사용하여 클러스터를 시작합니다.
4. 다른 클러스터에 대해서도 차례로 과정을 반복합니다.

한 클러스터 내 세션이 다른 클러스터 내 세션으로 페일오버되지 않기 때문에 한 가지 버전의 구성 요소를 실행 중인 서버 인스턴스에서 다른 버전의 구성 요소를 실행 중인 다른 클러스터 내 서버 인스턴스로 세션이 페일오버됨으로 버전이 불일치할 위험은 없습니다. 따라서, 클러스터 내 서버 인스턴스에 대한 세션 페일오버에 대해 클러스터는 안전한 경계 역할을 합니다.

주 다음과 같은 경우에는 이 방법을 사용할 수 없습니다.

- HADB(high-availability database)의 스키마를 변경할 경우. 자세한 내용은 *Sun Java System Application Server High Availability Administration Guide*의 Administering High Availability Database 장을 참조하십시오.
- 응용 프로그램 데이터베이스 스키마 변경이 포함된 응용 프로그램 업그레이드를 수행할 경우

주의 클러스터의 모든 서버 인스턴스를 함께 업그레이드합니다. 그렇지 않으면 인스턴스에 실행 중인 다른 버전의 구성 요소가 있을 경우 서버간에 세션이 페일오버됨으로써 버전 불일치가 발생할 위험이 있습니다.

로드 균형 조정 및 페일오버 구성

이 장에서는 Sun Java System Application Server에서 HTTP 요청의 로드 균형 조정을 설정하는 방법에 대해 설명합니다. 로드 밸런서에서 제어하는 서버 인스턴스 간의 페일오버 구성 방법도 설명합니다. 그리고 RMI-IIOP 로드 균형 조정 및 페일오버를 설명합니다.

이 장은 다음 내용으로 구성되어 있습니다.

HTTP 로드 균형 조정 및 페일오버 정보

HTTP 로드 균형 조정 및 페일오버

로드 균형 조정의 목적은 여러 독립 실행형 또는 클러스터링된 Sun Java System Application Server 인스턴스 간에 작업 로드를 똑같이 분산하여 시스템의 전체적인 처리 능력을 증가시키는 것입니다.

로드 밸런서를 사용하면 서버 인스턴스 간에 요청을 페일오버할 수 있습니다. 지속할 HTTP 세션 정보의 경우 HTTP 세션 지속성을 구성합니다. 자세한 내용은 "가용성 및 세션 지속성 구성"을 참조하십시오.

관리 콘솔이 아니라 asadmin 도구를 사용하여 HTTP 로드 균형 조정을 구성합니다.

HTTP 로드 균형 조정을 위한 요구 사항

HTTP 요청에 대해 로드 밸런서 플러그인을 사용하려면 다음 요구 사항이 충족되어야 합니다.

- Sun Java System Application Server Enterprise Edition이 설치되어 있어야 합니다.
- 웹 서버가 설치 및 구성되어 있어야 합니다.

  자세한 내용은 67페이지의 "HTTP 로드 균형 조정을 사용하도록 Web Server 구성"을 참조하십시오.
- 로드 밸런서 플러그인이 설치되어 있어야 합니다.
- Application Server가 구성되어 있어야 합니다.
  - 로드 균형 조정에 참여하는 Application Server 인스턴스나 클러스터가 만들어져 있어야 합니다.
  - 로드 균형 조정에 참여하는 모든 Application Server 인스턴스나 클러스터에 응용 프로그램이 배포되어 있어야 합니다.
  - 로드 균형 조정에 참여하는 서버 인스턴스와 클러스터의 환경이 같아야 합니다. 즉, 서버 인스턴스가 동일한 서버 구성을 참조하고 동일한 응용 프로그램이 배포되어 있어야 합니다.

할당된 요청과 할당되지 않은 요청 이해

HTTP 클라이언트에서 로드 밸런서로 처음 들어온 요청은 새로운 세션에 대한 요청입니다. 새로운 세션에 대한 요청을 *할당되지 않은* 요청이라고 합니다. 로드 밸런서는 라운드 로빈 알고리즘에 따라 클러스터의 Application Server 인스턴스로 이 요청을 보냅니다. 자세한 내용은 65페이지의 "HTTP 로드 균형 조정 알고리즘"을 참조하십시오.

Application Server 인스턴스에 세션이 만들어지면 로드 밸런서는 이 세션에 대한 모든 후속 요청의 경로를 특정 인스턴스로만 지정합니다. 기존 세션에 대한 요청을 *할당된* 요청 또는 *고정* 요청이라고 합니다.

HTTP 로드 균형 조정 알고리즘

Sun Java System Application Server 로드 밸런서는 고정 라운드 로빈 알고리즘을 사용하여 들어오는 HTTP 및 HTTPS 요청을 로드 균형 조정합니다. 지정한 세션의 모든 요청이 동일한 Application Server 인스턴스로 전송됩니다. 고정 로드 밸런서를 사용하면 세션 데이터가 클러스터의 모든 인스턴스에 분산되기 보다는 한 Application Server에 캐시됩니다.

따라서 고정 라운드 로빈 체계는 순수 라운드 로빈이 주는 로드의 균형 분산이라는 이점을 뛰어난 성능 이점으로 대체합니다.

고정 라운드 로빈 로드 균형 조정 알고리즘 정보

새로운 HTTP 요청을 로드 밸런서 플러그인으로 전송하면 단순 라운드 로빈 체계를 기반으로 Application Server 인스턴스에 전달됩니다. 계속해서 이 요청은 쿠키를 사용하거나 명시적 URL을 다시 작성하여 특정한 Application Server 인스턴스에 "고정"됩니다.

고정 정보를 사용해서 로드 밸런서 플러그인은 먼저 이전에 요청을 전달했던 인스턴스를 판별합니다. 해당 인스턴스가 정상일 경우 로드 밸런서 플러그인은 특정 Application Server 인스턴스에 그 요청을 전달합니다. 따라서 지정한 세션의 모든 요청이 동일한 Application Server 인스턴스로 전달됩니다.

로드 밸런서 플러그인은 다음 방법을 사용하여 세션 고정을 판별합니다.

- 쿠키 기반 방법
- 명시적 URL 재작성 방법

쿠키 기반 방법

쿠키 기반 방법에서 로드 밸런서 플러그인은 별도의 쿠키를 사용하여 경로 정보를 기록합니다.

주 쿠키 기반 방법을 사용하려면 HTTP 클라이언트에서 쿠키를 지원해야 합니다.

명시적 URL 재작성 방법

명시적 URL 재작성 방법에서는 고정 정보가 URL에 추가됩니다. HTTP 클라이언트에서 쿠키를 지원하지 않더라도 이 방법을 사용할 수 있습니다.

로드 균형 조정 및 페일오버 샘플 응용 프로그램

다음 디렉토리에는 로드 균형 조정 및 페일오버를 보여주는 샘플 응용 프로그램이 포함되어 있습니다.

install_dir`/samples/ee-samples/highavailability`
install_dir`/samples/ee-samples/failover`

`ee-samples` 디렉토리에는 샘플을 실행할 수 있도록 환경을 설정하는 데 필요한 정보도 포함되어 있습니다.

HTTP 로드 균형 조정 설정 개요

`asadmin` 도구를 사용하여 환경에 로드 균형 조정을 구성합니다. 다음 단계를 수행합니다.

1. 웹 서버 및 Application Server 인스턴스 또는 클러스터 설치 및 구성을 포함하여 64 페이지의 "HTTP 로드 균형 조정을 위한 요구 사항"을 완료합니다.
2. `asadmin` 명령 `create-http-lb-config`를 사용하여 로드 밸런서 구성을 만듭니다.
3. `asadmin create-http-lb-ref`를 사용하여 로드 밸런서가 관리할 수 있게 하려면 클러스터와 독립 실행형 서버 인스턴스에 대한 참조를 추가합니다.

   대상과 함께 로드 밸런서 구성을 만들었고 그 대상이 로드 밸런서가 참조하는 클러스터나 독립 실행형 서버 인스턴스일 경우 이 단계를 생략합니다.

4. asadmin enable-http-lb-server를 사용하여 로드 밸런서가 참조하는 클러스터나 독립 실행형 서버 인스턴스를 활성화합니다.
5. asadmin enable-http-lb-application을 사용하여 로드 균형 조정을 할 수 있도록 응용 프로그램을 활성화합니다.

   로드 밸런서에서 참조하는 클러스터나 독립 실행형 인스턴스에서 사용할 수 있도록 응용 프로그램을 이미 배포하여 활성화되어 있어야 합니다. 로드 균형 조정을 위한 활성화는 사용을 위한 활성화와 다른 단계입니다.
6. asadmin create-health-checker를 사용하여 상태 검사기를 만듭니다.

   상태 검사기는 비정상적인 서버 인스턴스가 다시 정상적이 되면 로드 밸런서가 새로운 요청을 전송할 수 있도록 이들을 모니터링합니다.
7. asadmin export-http-lb-config를 사용하여 로드 밸런서 구성 파일을 생성합니다.

   이 명령은 Sun Java System Application Server와 함께 제공된 로드 밸런서 플러그인에서 사용할 구성 파일을 생성합니다.
8. 로그 밸런서 플러그인 구성 파일이 저장되는 웹 서버 config 디렉토리로 로드 밸런서 구성 파일을 복사합니다.

HTTP 로드 균형 조정을 사용하도록 Web Server 구성

Web Server 구성 정보

로드 밸런서 플러그인 설치 프로그램은 웹 서버의 구성 파일을 일부 수정합니다. 변경 사항은 웹 서버에 따라 다릅니다.

주 로드 밸런서 플러그인은 Sun Java System Application Server Enterprise Edition과 함께 설치할 수도 있고 지원되는 웹 서버를 실행하는 시스템에 따로 설치할 수도 있습니다.

설치 절차에 대한 자세한 내용은 *Sun Java System Application Server Installation Guide*를 참조하십시오.

Sun Java System Web Server 수정 사항

설치 프로그램을 실행하면 Sun Java System Web Server의 구성 파일이 다음과 같이 변경됩니다.

1. 다음 로드 밸런서 플러그인 관련 항목을 웹 서버 인스턴스의 `magnus.conf` 파일에 추가합니다.

```
##EE lb-plugin
Init fn="load-modules"
shlib="web_server_install_dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough"
Thread="no"

Init fn="init-passthrough"

##end addition for EE lb-plugin
```

2. 다음과 같은 로드 밸런서 플러그인 관련 항목을 웹 서버 인스턴스의 `obj.conf` 파일에 추가합니다.

```
<Object name=default>

NameTrans fn="name-trans-passthrough" name="lbplugin"
config-file="web_server_install_dir/web_server_instance/config/loadbalancer.xml"

<Object name="lbplugin">
ObjectType fn="force-type" type="magnus-internal/lbplugin"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/lbplugin" fn="service-passthrough"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</object>
```

`lbplugin`은 `Object`를 고유하게 식별하는 이름이고, *web_server_install_dir*`/`*web_server_instance*`/config/loadbalancer.xml`은 로드 밸런서가 실행되도록 구성된 가상 서버용 XML 구성 파일 위치입니다.

설치 후 66페이지의 "HTTP 로드 균형 조정 설정 개요"에서 설명한 대로 로드 밸런서를 구성합니다.

Apache Web Server 수정 사항

Apache에 로드 밸런서 플러그인을 설치하기 전에 다음에서 Apache 컴파일 및 구성에 대한 정보를 참조하십시오. 부록 A, "Apache Web Server 컴파일 및 구성"

설치 프로그램에서 수정한 사항

로드 밸런서 플러그인 설치 프로그램은 필요한 파일을 웹 서버의 루트 디렉토리 아래 있는 `libexec`(Apache 1.3) 또는 `modules`(Apache 2.0) 폴더로 추출합니다. 로드 밸런서 플러그인에 관련된 다음 항목을 웹 서버 인스턴스의 `httpd.conf` 파일에 추가합니다.

```
<VirtualHost machine_name:443>

##Addition for EE lb-plugin

LoadFile /usr/lib/libCstd.so.1

LoadModule apachelbplugin_module libexec/mod_loadbalancer.so
#AddModule mod_apachelbplugin.cpp
<IfModule mod_apachelbplugin.cpp>
        config-file webserver_instance/conf/loadbalancer.xml
locale en
</IfModule>

<VirtualHost machine_ip_address>
DocumentRoot "webserver_instance/htdocs"
ServerName server_name
</VirtualHost>

##END EE LB Plugin ParametersVersion 7
```

주

- Apache 1.3의 경우 여러 Apache 하위 프로세스를 실행할 경우 프로세스마다 고유한 로드 균형 조정 라운드 로빈 시퀀스가 있습니다.

  예를 들어, 두 개의 Apache 하위 프로세스를 실행하고 로드 균형 조정 플로그인이 두 개의 Application Server 인스턴스에 대해 로드 균형 조정을 할 경우 첫 번째 요청은 인스턴스 #1로 보내고 두 번째 요청도 인스턴스 #1로 보냅니다. 세 번째 요청은 인스턴스 #2로 보내고, 네 번째 요청은 다시 인스턴스 #2로 보냅니다. 이 방법이 반복됩니다(instance1, instance1, instance2, instance2 등).

  이 동작은 일반적으로 예상할 수 있는 방법(즉, instance1, instance2, instance1, instance2 등)과 다릅니다. Sun Java System Application Server에서는 Apache용 로드 균형 조정 플러그인이 Apache 프로세스마다 로드 밸런서 인스턴스를 인스턴스화하여 독립적인 로드 균형 조정 시퀀스를 만듭니다.

- `--with-mpm=worker` 옵션을 사용하여 컴파일할 경우 Apache 2.0에는 멀티스레드된 동작이 발생합니다.

설치 후 수정 사항

Apache 웹 서버가 로드 밸런서 플러그인과 함께 제대로 작동하게 하려면 Apache 웹 서버를 보안 모드에서 실행해야 합니다. *apache_install_dir* 아래에 `sec_db_files`라고 하는 디렉토리를 만들고 *application_server_domain_dir*`/config/`*security_db_files*를 *apache_install_dir*`/`*sec_db_files*에 복사합니다.

Microsoft Windows의 추가 수정 사항

Microsoft Windows에서 Apache를 실행할 경우 플러그인 설치 후 일부 환경 변수를 변경해야 합니다.

시작->설정->제어판->시스템->고급->환경 변수->시스템 변수를 눌러 경로 환경 변수에 새로운 경로를 추가합니다. 다음을 포함하도록 경로 변수를 편집합니다.

application_server_install_dir`/bin`

그리고 Apache 웹 서버를 시작하기 전에 환경 변수 NSPR_NATIVE_THREADS_ONLY를 1로 설정합니다.

환경 변수 창의 시스템 변수 아래에서 새로 만들기를 누릅니다. 다음 이름 및 값 쌍을 입력합니다.

변수 이름: NSPR_NATIVE_THREADS_ONLY

변수 값: 1

시스템을 다시 시작합니다.

Microsoft IIS의 수정 사항

로드 밸런서 플러그인을 사용하도록 IIS(Microsoft Internet Information Services)를 구성하려면 Windows 인터넷 서비스 관리자의 특정 등록 정보를 수정합니다. 인터넷 서비스 관리자는 제어판 폴더의 관리 도구 폴더에 있습니다.

Sun Java System Application Server를 설치한 후 다음과 같이 수정합니다.

1. 인터넷 서비스 관리자를 엽니다.
2. 플러그인을 활성화할 웹 사이트를 선택합니다. 대개 이 웹 사이트 이름은 기본 웹 사이트입니다.
3. 웹 사이트를 마우스 오른쪽 버튼으로 누른 다음 속성을 선택하여 속성 노트북을 엽니다.
4. 새로운 ISAPI 필터를 추가하려면 ISAPI 필터 탭을 열고 추가를 누른 다음 아래 단계를 수행합니다.
   a. 필터 이름 필드에 Application Server를 입력합니다.
   b. 실행 파일 필드에서 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll`을 입력합니다.
   c. 확인을 누르고 속성 노트북을 닫습니다.
5. 새로운 가상 디렉토리를 작성하고 구성합니다.
   a. 기본 웹 사이트를 마우스 오른쪽 버튼으로 누른 다음 가상 디렉토리를 선택합니다.

      가상 디렉토리 작성 마법사가 열립니다.
   b. 별칭 필드에 `sun-passthrough`를 입력합니다.
   c. 디렉토리 필드에 `C:\Inetpub\wwwroot\sun-passthrough`를 입력합니다.
   d. 실행 권한 확인란을 선택합니다. 다른 모든 권한 관련 확인란은 선택 해제한 채로 둡니다.
   e. 마침을 누릅니다.
6. `sun-passthrough.dll` 파일의 경로 및 *application_server_install_dir*`/bin`을 시스템의 `path` 환경 변수에 추가합니다. 시스템을 다시 시작합니다.

7. 새로운 설정을 적용하려면 웹 서버를 중지했다가 다시 시작합니다.

   웹 서버를 중지하려면 웹 사이트를 오른쪽 버튼으로 누르고 중지를 선택합니다. 웹 서버를 시작하려면 웹 사이트를 마우스 오른쪽 버튼으로 누르고 시작을 선택합니다.

   그 후 웹 브라우저에 다음 사항을 입력하여 웹 응용 프로그램 컨텍스트 루트에 액세스합니다.

   `http://`*webserver_name*`/`*web_application*

   여기에서 *webserver_name*은 웹 서버의 호스트 이름이나 IP 주소이고 */web_application*은 `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` 파일에 나열된 컨텍스트 루트입니다. 웹 서버, 로드 밸런서 플러그인 및 Application Server가 제대로 작동하는지 확인합니다.

설치 프로그램에서 `sun-passthrough.properties`의 다음 등록 정보를 자동으로 구성합니다. 기본값을 변경할 수 있습니다.

표 3-1 Microsoft IIS에 맞게 자동으로 구성된 sun-passthrough 등록 정보

등록 정보	정의	기본값
lb-config-file	로드 밸런서 구성 파일 경로	*IIS_www_root*`\sun-passthrough\loadbalancer.xml`
log-file	로드 밸런서 로그 파일 경로	*IIS_www_root*`\sun-passthrough\lb.log`
log-level	웹 서버의 로그 수준	정보

다중 웹 서버 인스턴스 구성

Sun Java System Application Server 설치 프로그램에서는 한 시스템에 여러 로드 밸런서 플러그인을 설치하는 것을 허용하지 않습니다. 단일 시스템에 로드 밸런서 플러그인과 함께 복수 웹 서버를 두려면 단일 클러스터 또는 복수 클러스터든 상관없이 로드 밸런서 플러그인을 구성하기 위한 몇 가지 수동 단계가 필요합니다.

1. 로드 밸런서 플러그인을 사용하려면 68페이지의 "Sun Java System Web Server 수정 사항", 69페이지의 "Apache Web Server 수정 사항" 또는 71페이지의 "Microsoft IIS의 수정 사항"에서 설명한 대로 새로운 웹 서버 인스턴스를 구성합니다.
2. 기존 웹 서버 인스턴스의 `config` 디렉토리에서 `sun-loadbalancer_1_1.dtd` 파일을 새 인스턴스의 `config` 디렉토리에 복사합니다.

3. 동일한 로드 밸런서 구성을 사용하려면 기존 웹 서버 인스턴스의 `config` 디렉토리에서 `loadbalancer.xml` 파일을 새로운 인스턴스의 `config` 디렉토리에 복사합니다.
4. 다른 로드 밸런서 구성을 사용하려면 다음 작업을 수행합니다.
   a. `asadmin create-http-lb-config`를 사용하여 새로운 로드 밸런서 구성을 만듭니다.
   b. `asadmin export http-lb-config`를 사용하여 새로운 구성을 `loadbalancer.xml` 파일로 내보냅니다.
   c. `loadbalancer.xml` 파일을 새로운 웹 서버의 `config` 디렉토리에 복사합니다.

   로드 밸런서 구성을 작성하고 이 구성을 `loadbalancer.xml` 파일로 내보내는 방법에 대한 자세한 내용은 "HTTP 로드 밸런서 구성 작업"을 참조하십시오.

HTTP 로드 밸런서 구성 작업

HTTP 로드 밸런서 구성 만들기

로드 밸런서 구성은 로드 밸런서를 정의하는 domain.xml 파일에 명명된 구성입니다.

로드 균형 조정 구성은 매우 융통성이 있습니다.

- 로드 밸런서마다 하나의 로드 밸런서 구성만 갖지만, 각 로드 밸런서 구성에 여러 로드 밸런서를 연결할 수 있습니다.
- 로드 밸런서는 하나의 도메인만 지원하지만 한 도메인은 여러 로드 밸런서를 연결할 수 있습니다.

asadmin 명령 create-http-lb-config를 사용하여 구성을 만듭니다. 다음 매개 변수를 지정합니다.

- 응답 시간 초과

  서버 인스턴스에서 응답을 반환해야 하는 시간(초)입니다. 기간 내에 응답을 받지 못하면 서버가 비정상인 것으로 간주됩니다. 기본값은 60입니다.

- HTTPS 라우팅

  로드 밸런서에 대한 HTTPS 요청으로 서버 인스턴스에 대한 HTTPS 또는 HTTP 요청이 발생하는지 지정합니다.

  자세한 내용은 81페이지의 "HTTP 및 HTTPS 세션 페일오버 구성"을 참조하십시오.

- 다시 로드 간격

  로드 밸런서 구성 파일 loadbalancer.xml의 변경 사항을 확인하는 간격입니다. 변경 사항이 확인되면 구성 파일을 다시 로드합니다. 값 0은 다시 로드를 비활성화합니다.

  자세한 내용은 79페이지의 "동적 재구성 활성화"를 참조하십시오.

- 모니터

  로드 밸런서에 대한 모니터링을 활성화할지 여부를 지정합니다.

  자세한 내용은 83페이지의 "HTTP 로드 밸런서 플러그인 모니터링"을 참조하십시오.

- routecookie

  경로 정보를 기록할 때 로드 밸런서 플러그인에서 사용하는 쿠키 이름을 지정합니다. HTTP 클라이언트가 쿠키를 지원해야 합니다. 쿠키를 저장하기 전에 확인하도록 브라우저를 설정한 경우 쿠키 이름은 JROUTE입니다.

- 대상

  로드 밸런서 구성의 대상을 지정합니다. 대상을 지정한 경우 대상에 대한 참조를 추가한 것과 같습니다. 클러스터나 독립 실행형 인스턴스가 대상이 될 수 있습니다.

자세한 내용은 create-http-lb-config, delete-http-lb-config 및 list-http-lb-configs에 대한 설명서를 참조하십시오.

HTTP 로드 밸런서 참조 만들기

로드 밸런서에 독립 실행형 서버나 클러스터에 대한 참조를 만들면 로드 밸런서가 제어하는 대상 서버와 클러스터 목록에 그 서버나 클러스터가 추가됩니다. 참조된 서버나 클러스터를 enable-http-lb-server를 사용하여 활성화해야 관련 요청의 로드를 균형 조정할 수 있습니다. 대상과 함께 로드 밸런서 구성을 만들었다면 해당 대상이 이미 참조로 추가되어 있습니다.

create-http-lb-ref를 사용하여 참조를 만듭니다. 로드 밸런서 구성 이름과 대상 서버 인스턴스 또는 클러스터를 제공해야 합니다.

참조를 삭제하려면 delete-http-lb-ref를 사용합니다. 참조를 삭제하려면 disable-http-lb-server를 사용하여 참조된 서버나 클러스터를 비활성화해야 합니다.

자세한 내용은 create-http-lb-ref 및 delete-http-lb-ref에 대한 설명서를 참조하십시오.

로드 균형 조정을 위해 서버 인스턴스 활성화

서버 인스턴스나 클러스터에 대한 참조를 만든 후 enable-http-lb-server를 사용하여 서버 인스턴스나 클러스터를 활성화합니다. 로드 밸런서 구성을 만들 때 서버 인스턴스나 클러스터를 대상으로 사용한 경우 이를 활성화해야 합니다.

자세한 내용은 enable-http-lb-server 설명서를 참조하십시오.

로드 균형 조정을 위해 응용 프로그램 활성화

로드 밸런서에서 관리하는 모든 서버는 구성이 같아야 합니다. 즉, 배포된 응용 프로그램 집합도 같아야 합니다. 응용 프로그램을 배포하고 액세스할 수 있도록 활성화한 경우(배포 단계 중 또는 이후에 발생함) 로드 균형 조정을 위해 응용 프로그램을 활성화해야 합니다. 로드 균형 조정을 하도록 응용 프로그램을 활성화하지 않은 경우 응용 프로그램이 배포된 서버에 대한 요청은 로드 균형 조정되고 페일오버되더라도 이 응용 프로그램에 대한 요청은 로드 균형 조정되지 않고 페일오버됩니다.

응용 프로그램을 활성화할 경우 응용 프로그램 이름과 대상을 지정합니다. 로드 밸런서에서 여러 대상(예: 두 개의 클러스터)을 관리할 경우 모든 대상에서 응용 프로그램을 활성화합니다.

자세한 내용은 `enable-http-lb-application`의 온라인 도움말을 참조하십시오.

새로운 응용 프로그램을 배포할 경우 로드 균형 조정을 위해 응용 프로그램을 활성화하고 로드 밸런서 구성을 다시 내보내야 합니다.

HTTP 상태 검사기 만들기

로드 밸런서의 상태 검사기에 비정상으로 표시된 모든 구성된 Application Server 인스턴스를 정기적으로 검사합니다. 상태 검사기는 필수가 아닙니다. 그러나 상태 검사기가 없거나 상태 검사기가 비활성화된 경우 비정상 인스턴스의 정기적인 상태 검사가 수행되지 않습니다.

로드 밸런서의 상태 검사 기법은 HTTP를 사용하여 Application Server 인스턴스와 통신합니다. 상태 검사기는 지정한 URL에 HTTP 요청을 보내고 응답을 기다립니다. HTTP 응답 헤더의 상태 코드가 100과 500 사이에 있으면 인스턴스가 정상임을 의미합니다.

상태 검사기 만들기

상태 검사기를 만들려면 `asadmin create-http-health-checker` 명령을 사용합니다. 다음 매개 변수를 지정합니다.

- url

  로드 밸런서가 검사하여 상태를 확인할 Listener의 URL을 지정합니다. 기본값은 `"/"`입니다.

- `interval`

  인스턴스의 상태 검사가 발생하는 간격(초)을 지정합니다. 기본값은 30초입니다. 0을 지정하면 상태 검사기가 비활성화됩니다.

- timeout

  Listener를 정상으로 간주하기 위해 응답을 받아야 하는 시간 초과 간격(초)을 지정합니다. 기본값은 10초입니다.

Application Server 인스턴스가 비정상으로 표시되면 상태 검사기는 비정상 인스턴스를 폴링하여 인스턴스가 정상적으로 되었는지 확인합니다. 상태 검사기는 지정된 URL을 사용하여 모든 비정상 Application Server 인스턴스를 검사하고 정상 상태로 되었는지 확인합니다.

상태 검사기에서 비정상 인스턴스가 정상이 되었음을 확인하면 해당 인스턴스는 정상 인스턴스 목록에 추가됩니다.

자세한 내용은 `create-http-health-checker` 및 `delete-http-health-checker`에 대한 설명서를 참조하십시오.

정상 인스턴스에 대한 추가 상태 검사 등록 정보

`create-http-health-checker`로 만든 상태 검사기만 비정상 인스턴스를 검사합니다. 정상 인스턴스를 정기적으로 검사하려면 내보낸 `loadbalancer.xml` 파일에 추가 등록 정보를 설정합니다.

주 `loadbalancer.xml`을 내보낸 *후* 수동으로 편집해야만 이 등록 정보를 설정할 수 있습니다. 사용할 수 있는 동등한 asadmin 명령이 없습니다.

정상 인스턴스를 검사하려면 다음 등록 정보를 설정합니다.

표 3-2 상태 검사기 등록 정보

등록 정보	정의
active-healthcheck-enabled	정상 서버 인스턴스를 핑하여 정상인지 확인할지 여부를 나타내는 True/false 플래그입니다. 서버 인스턴스를 핑하려면 플래그를 true로 설정합니다.
number-healthcheck-retries	서버 인스턴스를 비정상으로 표시하기 전에 로드 밸런서의 상태 검사기가 응답이 없는 서버 인스턴스를 핑하는 횟수를 지정합니다. 유효한 값은 1과 1000 사이입니다. 설정된 기본값은 3입니다.

`loadbalancer.xml` 파일을 편집하여 등록 정보를 설정합니다. 예를 들면 다음과 같습니다.

```
<property name="active-healthcheck-enabled" value="true"/>
<property name="number-healthcheck-retries" value="3"/>
```

이 등록 정보를 추가하고 나서 loadbalancer.xml 파일을 다시 편집하고 내보낸 경우 이 등록 정보를 파일에 다시 추가해야 합니다. 새로 내보낸 구성에는 이 등록 정보가 포함되지 않기 때문입니다.

로드 밸런서 구성 파일 내보내기

Sun Java System Application Server와 함께 제공되는 로드 균형 조정 플러그인은 loadbalancer.xml이라고 하는 구성 파일을 사용합니다. asadmin 도구를 사용하여 domain.xml 파일에 로드 밸런서 구성을 만듭니다. 로드 균형 조정 환경을 구성한 후 파일로 내보냅니다.

1. asadmin 명령 export-http-lb-config를 사용하여 loadbalancer.xml 파일을 내보냅니다.

   특정한 로드 밸런서 구성을 위해 loadbalancer.xml 파일을 내보냅니다. 경로 및 다른 파일 이름을 지정할 수 있습니다. 파일 이름을 지정하지 않으면 파일 이름이 loadbalancer.xml.*load_balancer_config_name*으로 지정됩니다. 경로를 지정하지 않으면 *application_server_install_dir*/domains/*domain_name*/generated 디렉토리에 파일이 만들어집니다.

   Windows에서 경로를 지정하려면 경로를 따옴표로 묶습니다. 예를 들면, "c:\sun\AppServer\loadbalancer.xml"입니다.

2. 내보낸 로드 밸런서 구성 파일을 웹 서버의 구성 디렉토리에 복사합니다.

   예를 들어, Sun Java System Web Server의 경우 위치는 *web_server_root*/config가 될 수 있습니다.

   Web Server 구성 디렉토리의 로드 밸런서 구성 파일 이름은 loadbalancer.xml이어야 합니다. 파일 이름이 다를 경우(예: loadbalancer.xml.*load_balancer_config_name*) 이름을 변경해야 합니다.

HTTP 로드 밸런서 구성 변경

서버에 대한 참조를 작성 또는 삭제하거나, 새로운 응용 프로그램을 배포하거나, 서버나 응용 프로그램을 활성화 또는 비활성화하는 식으로 HTTP 로드 밸런서 구성을 변경한 경우 로드 밸런서 구성 파일을 다시 내보내고 웹 서버의 config 디렉토리에 복사합니다. 자세한 내용은 78페이지의 "로드 밸런서 구성 파일 내보내기"를 참조하십시오.

로드 밸런서 플러그인은 로드 밸런서 구성에 지정한 다시 로드 간격을 기반으로 정기적으로 업데이트된 구성이 있는지 확인합니다. 지정한 시간 후 로드 밸런서가 새로운 구성 파일을 발견하면 해당 구성을 사용하기 시작합니다.

동적 재구성 활성화

동적 재구성을 활성화하면 로드 밸런서 플러그인은 업데이트된 구성을 정기적으로 검사합니다. 동적 재구성을 활성화하려면 다음 작업을 수행합니다.

- 로드 밸런서 구성을 만들 때 동적 재구성을 활성화하려면 `asadmin create-http-lb-config`를 실행할 때 `--reloadinterval` 옵션을 사용합니다.

  이 옵션은 로드 밸런서 구성 파일 `loadbalancer.xml`의 변경 사항을 검사하는 간격을 설정합니다. 값 0은 재로드를 비활성화합니다. 기본적으로는 동적으로 다시 로드할 수 있도록 설정되며 간격은 60초로 설정됩니다.
- 동적으로 다시 로드할 수 없도록 한 것을 할 수 있도록 하거나 다시 로드하는 간격을 변경하려면 `asadmin set` 명령을 사용합니다.

이 설정을 변경한 후 로드 밸런서 구성 파일을 다시 내보내고 웹 서버의 `config` 디렉토리에 복사합니다.

동적 재구성을 사용하지 않도록 했다가 사용하도록 할 경우 웹 서버도 다시 시작해야 합니다.

주

- 로드 밸런서를 재구성하는 중에 로드 밸런서에 하드 디스크 오류가 발생한 경우 현재 메모리에 있는 구성을 사용합니다. 로드 밸런서는 기존 구성을 덮어쓰기 전에 수정된 구성 데이터가 DTD와 호환되는지 확인합니다.

  디스크 읽기 오류가 발생한 경우 경고 메시지가 웹 서버의 오류 로그 파일에 로깅됩니다.

  Sun Java System Web Server의 오류 로그는 *web_server_install_dir*`/`*webserver_instance*`/logs/`에 있습니다.

서버 인스턴스 또는 클러스터 비활성화(정지)

어떤 이유에서건 Application Server를 중지하기 전에 인스턴스가 서비스 요청을 끝내도록 해야 하는 경우가 있습니다. 서버 인스턴스나 클러스터를 적절하게 비활성화하는 프로세스를 정지라고 합니다.

로드 밸런서는 Application Server 인스턴스를 정지하는 데 다음과 같은 정책을 사용합니다.

- 인스턴스(독립 실행형 또는 클러스터의 일부)를 비활성화했지만 시간 초과가 만기되지 않은 경우 고정된 요청이 계속 해당 인스턴스로 전달됩니다. 그러나 새로운 요청은 비활성화된 인스턴스로 전송되지 않습니다.
- 시간 초과가 만료되면 인스턴스가 비활성화됩니다. 로드 밸런서에서 인스턴스로 열려 있는 모든 연결이 닫힙니다. 이 인스턴스에 고정된 모든 세션이 잘못되지 않았더라도 로드 밸런서는 이 인스턴스에 요청을 전송하지 않습니다. 대신 로드 밸런서는 고정된 요청을 다른 정상적인 인스턴스에 페일오버합니다.

서버 인스턴스나 클러스터를 비활성화하려면 다음 작업을 수행합니다.

1. `asadmin disable-http-lb-server`를 실행하여 시간 초과(분)를 설정합니다.
2. `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.
3. 내보낸 구성을 웹 서버 `config` 디렉토리로 복사합니다.
4. 서버 인스턴스나 인스턴스를 중지합니다.

응용 프로그램 비활성화(정지)

웹 응용 프로그램을 배포 해제하기 전에 응용 프로그램이 진행 중인 요청 처리를 완료하도록 합니다. 응용 프로그램을 적절하게 비활성화하는 프로세스를 정지라고 합니다.

로드 밸런서는 응용 프로그램을 정지하는 데 다음과 같은 정책을 사용합니다.

- 응용 프로그램이 비활성화되었지만 시간 초과가 만료되지 않은 경우 로드 밸런서는 비활성화된 응용 프로그램에 대한 새로운 요청을 전달하지 않습니다. 이 요청은 웹 서버로 반환됩니다. 고정된 요청은 시간 초과가 만료될 때까지 계속 전달됩니다.
- 시간 초과가 만료되면 응용 프로그램이 비활성화됩니다. 로드 밸런서는 고정된 요청을 포함하여 이 응용 프로그램에 대한 요청을 수락하지 않습니다.

로드 밸런서가 참조하는 모든 서버 인스턴스나 클러스터에서 응용 프로그램을 비활성화할 경우 비활성화된 응용 프로그램의 사용자는 응용 프로그램이 다시 활성화될 때까지 서비스가 손실됩니다.

다른 서버 인스턴스나 클러스터에서 응용 프로그램을 활성화한 채로 두고 하나의 서버 인스턴스나 클러스터에서만 응용 프로그램을 비활성화할 경우 사용자는 계속 그 응용 프로그램에 액세스할 수 있습니다.

응용 프로그램을 비활성화하려면 다음 작업을 수행합니다.

1. `asadmin disable-http-lb-application`을 실행하여 시간 초과(분), 비활성화할 응용 프로그램 이름, 응용 프로그램을 비활성화할 대상 클러스터나 인스턴스를 지정합니다.
2. `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.
3. 내보낸 구성을 웹 서버 `config` 디렉토리로 복사합니다.

HTTP 및 HTTPS 세션 페일오버 구성

세션이 연결된 원래 Application Server 인스턴스를 사용할 수 없게 된 경우 로드 밸런서 플러그인은 HTTP/HTTPS 세션을 다른 Application Server 인스턴스로 페일오버합니다. 이 절에서는 HTTP/HTTPS 라우팅 및 세션 페일오버를 활성화하도록 로드 밸런서 플러그인을 구성하는 방법에 대해 설명합니다.

HTTP 세션 지속성 구성에 대한 자세한 내용은 "가용성 및 세션 지속성 구성"을 참조하십시오.

이 절에서는 다음 항목에 대해 설명합니다.

- HTTPS 라우팅 정보
- HTTPS 라우팅 구성
- HTTP/HTTPS 요청의 로드 균형 조정 시 알려진 문제점

HTTPS 라우팅 정보

HTTP 또는 HTTPS 여부와 상관없이 모든 들어오는 요청은 로드 밸런서 플러그인이 Application Server 인스턴스로 라우팅합니다. 그러나 HTTPS 라우팅이 활성화된 경우 로드 밸런서 플러그인은 HTTPS 포트만 사용하는 Application Server로 HTTPS 요청을 전달합니다. 새로운 요청과 고정된 요청 모두에 대해 HTTPS 라우팅을 수행합니다.

HTTPS 요청을 수신하고 진행 중인 세션이 없을 경우 로드 밸런서 플러그인은 HTTPS 포트가 구성된 사용 가능한 Application Server 인스턴스를 선택하고 요청을 인스턴스로 전달합니다.

진행 중인 HTTP 세션에서 동일한 세션에 대한 새로운 HTTPS 요청을 받으면 세션과 HTTP 세션 중에 저장된 고정된 정보를 사용하여 HTTPS 요청을 라우팅합니다. 마지막 HTTP 요청이 처리된 동일한 서버로 새로운 HTTP S 요청 경로가 지정되지만 HTTPS 포트에 지정됩니다.

HTTPS 라우팅 구성

`create-http-lb-config` 명령의 `httpsrouting` 옵션은 로드 균형 조정에 참여하는 모든 Application Server에 대해 HTTPS 라우팅을 설정 또는 해제할 것인지를 제어합니다. 이 옵션을 false로 설정하면 모든 HTTP 및 HTTPS 요청이 HTTP로 전달됩니다. 로드 밸런서 구성을 새로 만들 경우 이 옵션을 true로 설정하거나 나중에 `asadmin set` 명령을 사용하여 변경합니다.

주

- HTTPS 라우팅을 사용하려면 HTTPS listener를 하나 이상 구성해야 합니다.
- `https-routing`을 `true`로 설정한 경우 새로운 요청이나 고정된 요청이 들어오는데 클러스터에 정상적인 HTTPS listener가 없으면 그 요청에 대해 오류가 발생합니다.

HTTP/HTTPS 요청의 로드 균형 조정 시 알려진 문제점

다음 목록에서는 HTTP/HTTPS 요청 처리와 관련하여 로드 밸런서의 제한 사항에 대해 설명합니다.

- 세션에 HTTP 및 HTTPS 요청이 모두 있을 경우 첫 번째 요청은 HTTP 요청이어야 합니다. 첫 번째 요청이 HTTPS 요청일 경우 뒤에 HTTP 요청이 올 수 없습니다. 이는 HTTPS 세션과 연관된 쿠키를 브라우저에서 반환하지 않기 때문입니다. 브라우저는 두 개의 다른 프로토콜을 두 개의 다른 서버로 해석하고 새로운 세션을 시작합니다.

  `httpsrouting`을 `true`로 설정한 경우에만 이 제한 사항이 유효합니다.

- 세션에 HTTP와 HTTPS 요청이 모두 있을 경우 Application Server 인스턴스가 HTTP와 HTTPS listener 모두에 대해 구성되어야 합니다.

  `httpsrouting`을 `true`로 설정한 경우에만 이 제한 사항이 유효합니다.

- 세션에 HTTP 및 HTTPS 요청이 모두 있을 경우 Application Server 인스턴스를 표준 포트 번호(HTTP의 경우 80, HTTPS의 경우 443)를 사용하는 HTTP 및 HTTPS listener와 함께 구성해야 합니다.

  `httpsrouting`에 설정된 값과 상관없이 이 제한 사항이 유효합니다.

멱등원(Idempotent) URL 구성

배포된 응용 프로그램의 가용성을 개선하려면 로드 밸런서에서 처리하는 모든 Application Server 인스턴스에서 실패한 멱등원(Idempotent) HTTP 요청을 재시도하도록 환경을 구성합니다. 예를 들어, 검색 요청을 다시 시도하려면 읽기 전용 요청에 대해 이 옵션을 사용합니다.

멱등원(Idempotent) 요청은 다시 시도할 때 응용 프로그램에 변경이나 불일치를 일으키지 않는 요청입니다. HTTP의 경우 일부 메소드(예: GET)는 멱등원(Idempotent)이지만 다른 메소드(예: POST)는 멱등원(Idempotent)이 아닙니다. 멱등원(Idempotent) URL을 재시도할 경우 값이 서버나 데이터베이스에서 변경되지 않습니다. 사용자가 수신한 응답의 변화만이 유일한 차이점입니다.

멱등원(Idempotent) 요청의 예에는 검색 엔진 쿼리와 데이터베이스 쿼리가 포함됩니다. 다시 시도할 때 데이터가 업데이트되거나 수정되지 않는 것이 기본 원칙입니다.

`sun-web.xml` 파일에 멱등원(Idempotent) URL을 구성합니다. 로드 밸런서 구성을 내보낼 때 멱등원(Idempotent) URL 정보가 `loadbalancer.xml` 파일에 자동으로 추가됩니다.

멱등원(Idempotent) URL 구성에 대한 자세한 내용은 *Developer's Guide*를 참조하십시오.

HTML 오류 페이지 구성

최종 사용자에게 표시할 자신만의 오류 페이지를 지정하거나 오류 페이지 URL을 지정할 수 있습니다. 오류 페이지를 지정하면 오류 보고를 위해 구성된 다른 모든 방법이 무시됩니다.

`sun-web.xml` 파일에 HTML 오류 페이지를 구성합니다. 로드 밸런서 구성을 내보낼 때 멱등원(Idempotent) HTML 오류 페이지 정보가 `sun-web.xml` 파일에서 `loadbalancer.xml` 파일로 자동으로 추가됩니다.

HTML 오류 페이지 구성에 대한 자세한 내용은 *Developer's Guide*를 참조하십시오.

HTTP 로드 밸런서 플러그인 모니터링

로그 메시지 구성

로드 밸런서 플러그인은 웹 서버의 로깅 기법을 사용하여 로그 메시지를 기록합니다. Application Server의 기본 로그 수준은 Sun Java System Web Server(INFO), Apache Web Server(WARN) 및 Microsoft IIS(INFO)의 기본 로깅 수준으로 설정됩니다. Application Server 로그 수준 FINE, FINER 및 FINEST는 웹 서버의 DEBUG 수준에 매핑됩니다.

이 로그 메시지는 웹 서버 로그 파일에 기록되고, 스크립트를 사용하여 구문 분석하거나 스프레드시트로 가져와서 필수 메트릭을 계산할 수 있는 원시 데이터 형식입니다.

로그 메시지 유형

로드 밸런서 플러그인은 다음과 같은 세 가지 다른 집합의 로그 메시지를 생성합니다.

- 로드 밸런서 구성 프로그램 로그 메시지
- 요청 디스패치 및 런타임 로그 메시지
- 구성 프로그램 오류 메시지

로드 밸런서 구성 프로그램 로그 메시지

멱등원(Idempotent) URL과 오류 페이지 설정을 사용할 경우 이 메시지가 기록됩니다.

멱등원(Idempotent) URL 패턴 구성 출력에는 다음 정보가 포함됩니다.

- 로그 수준을 FINE으로 설정한 경우:

  ```
  CONFxxxx: IdempotentUrlPattern configured <url-pattern> <no-of-retries>
  for web-module : <web-module>
  ```

- 로그 수준을 SEVERE로 설정한 경우:

  ```
  CONFxxxx: Duplicate entry of Idempotent URL element <url-pattern> for
  webModule <web-module> in loadbalancer.xml."
  ```

- 로그 수준을 WARN으로 설정한 경우:

  ```
  CONFxxxx: Invalid IdempotentUrlPatternData <url-pattern> for web-module
  <web-module>
  ```

오류 페이지 URL 구성의 출력에는 다음 정보가 포함됩니다(WARN으로 설정된 로그 수준).

```
CONFxxxx: Invalid error-url for web-module <web-module>
```

요청 디스패치 및 런타임 로그 메시지

요청을 로드 균형 조정하고 디스패치하는 동안 이 로그 메시지가 생성됩니다.

- 메소드 시작을 위한 표준 로깅 출력에는 다음 정보가 포함됩니다(FINE으로 설정된 로그 수준).

  ```
  ROUTxxxx: Executing Router method <method_name>
  ```

- 메소드 시작을 위한 라우터 로그 출력에는 다음 정보가 포함됩니다(INFO로 설정된 로그 수준).

  ```
  ROUTxxxx: Successfully Selected another ServerInstance for idempotent
  request <Request-URL>
  ```

- 런타임 로그 출력에는 다음 정보가 포함됩니다(INFO로 설정된 로그 수준).

  ```
  RNTMxxxx: Retrying Idempotent <GET/POST/HEAD> Request <Request-URL>
  ```

구성 프로그램 오류 메시지

구성 문제가 있을 경우, 예를 들어 참조하는 사용자 정의 오류 페이지가 누락된 경우 이 오류가 표시됩니다.

- INFO로 설정된 로그 수준:

  ROUTxxxx: Non Idempotent Request <Request-URL> cannot be retried

  ```
  Example :: ROUTxxxx: Non Idempotent Request
  http://sun.com/addToDB?x=11&abc=2 cannot be retried
  ```

- FINE으로 설정된 로그 수준:

  RNTMxxxx: Invalid / Missing Custom error-url / page: <error-url> for web-module: <web-module>

  ```
  Example :: RNTMxxxx: Invalid / Missing Custom error-url /  page:
  myerror1xyz for web-module: test
  ```

모니터링 구성

로드 밸런서 플러그인 로그 메시지를 설정하려면 다음 단계를 수행합니다.

1. 웹 서버에 로깅 옵션을 설정합니다.
   a. Sun Java System Web Server의 관리 콘솔에서 `Magnus Editor` 탭으로 이동합니다.

      `Log Verbose` 옵션을 `On`으로 설정합니다.
   b. Apache Web Server의 경우 로그 수준을 `DEBUG`로 설정합니다.
   c. Microsoft IIS의 경우 `sun-passthrough.properties` 파일에서 로그 수준을 `FINE`으로 설정합니다.
2. 로드 밸런서 구성의 `monitor` 옵션을 **true**로 설정합니다.

   로드 밸런서 구성을 처음 만들 경우 `asadmin create-http-lb-config` 명령을 사용하여 모니터링을 **true**로 설정하거나, `asadmin set` 명령을 사용하여 나중에 **true**로 설정합니다. 모니터링은 기본적으로 비활성화되어 있습니다.

로드 밸런서 플러그인은 다음 정보를 기록합니다.

- 모든 요청의 요청 시작/중지 정보
- 비정상 인스턴스에서 정상 인스턴스로 요청이 페일오버된 경우 페일오버된 요청 정보
- 모든 상태 검사 주기가 끝나는 시점의 비정상 인스턴스 목록

주 로드 밸런서 플러그인에서 로깅이 활성화된 경우, 그리고 웹 서버 로그 수준을 `DEBUG`로 설정하거나 자세한 메시지를 인쇄할 경우 로드 밸런서는 HTTP 세션 아이디를 웹 서버 로그 파일에 기록합니다. 따라서 로드 밸런서 플러그인을 호스트하는 웹 서버가 DMZ에 있을 경우 작업 환경에서 `DEBUG` 또는 유사한 로그 수준을 사용하지 마십시오.

`DEBUG` 로그 수준을 사용해야 할 경우 `loadbalancer.xml`에서 `require-monitor-data` 등록 정보를 `false`로 설정하여 로드 밸런서 로깅을 해제하십시오.

모니터링 메시지

로드 밸런서 플러그인 로그 메시지의 형식은 다음과 같습니다.

- HTTP 요청의 시작에는 다음 정보가 포함되어 있습니다.

  ```
  RequestStart Sticky(New) <req-id> <time-stamp> <URL>
  ```

  타임스탬프 값은 1970년 1월 1일부터의 밀리초입니다. 예를 들면 다음과 같습니다.

  ```
  RequestStart New 123456 602983
  http://austen.sun.com/Webapps-simple/servlet/Example1
  ```

- HTTP 요청의 끝에는 다음과 같이 `RequestExit` 메시지가 포함됩니다.

  ```
  RequestExit Sticky(New) <req-id> <time-stamp> <URL> <listener-id>
  <response-time> Failure-<reason for error>(incase of a failure)
  ```

  예를 들면 다음과 같습니다.

  ```
  RequestExit New 123456 603001
  http://austen.sun.com/Webapps-simple/servlet/Example1
  http://austen:2222 18
  ```

  주 RequestExit 메시지에서 <response-time>은 로드 밸런서 플러그인의 관점에서 전체 요청 반환 시간(밀리초)을 나타냅니다.

- 비정상 인스턴스 목록은 다음과 같습니다.

  ```
  UnhealthyInstances <cluster-id> <time-stamp> <listener-id>,
  <listener-id>...
  ```

  예를 들면 다음과 같습니다.

  ```
  UnhealthyInstances cluster1 701923 http://austen:2210, http://austen:3010
  ```

- 페일오버된 요청 목록은 다음과 같습니다.

  ```
  FailedoverRequest <req-id> <time-stamp> <URL> <session-id>
  <failed-over-listener-id> <unhealthy-listener-id>
  ```

  예를 들면 다음과 같습니다.

  ```
  FailedoverRequest 239496 705623
  http://austen.sun.com/Apps/servlet/SessionTest 16dfdac3c7e80a40
  http://austen:4044 http://austen:4045
  ```

응용 프로그램 업그레이드

롤링 업그레이드 정보

사용자에게 서비스 손실이 없도록 응용 프로그램을 업그레이드하려면 한 번에 하나의 서버나 클러스터의 응용 프로그램을 업그레이드합니다. 클러스터는 혼합된 버전의 환경을 유지 관리하고, 사용자는 업그레이드가 발생하는 것을 인식하지 못합니다. 이런 유형의 업그레이드를 롤링 업그레이드라고 합니다.

이전 버전과 새로운 버전의 응용 프로그램이 호환되고 한 번에 둘 다 실행할 수 있는 경우에만 롤링 업그레이드가 가능합니다. 세션 정보는 호환 가능해야 합니다. 단일 독립 실행형 클러스터나 복수 클러스터에서 혼합된 모드의 롤링 업그레이드를 수행합니다.

응용 프로그램에 주요한 변경 사항이 있을 경우, 예를 들어 데이터베이스 스키마에 대한 변경 사항이 있을 경우 혼합 모드 환경의 롤링 업그레이드가 불가능합니다. 그럴 경우 업그레이드하는 동안 응용 프로그램을 중지합니다. 응용 프로그램을 배포 해제한 다음 동일한 이름으로 업그레이드된 응용 프로그램을 다시 배포합니다.

단일 독립 실행형 클러스터에서 업그레이드

단일 독립 실행형 클러스터(다른 클러스터와 구성을 공유하지 않는 클러스터)에서 응용 프로그램을 업그레이드하려면 다음 작업을 수행합니다.

1. 응용 프로그램 이전 버전을 저장하거나 도메인을 백업합니다.

   도메인을 백업하려면 `asadmin backup-domain` 명령을 사용합니다.

2. 클러스터의 동적 재구성이 활성화된 경우 해제합니다.

   관리 콘솔 사용:

   a. 구성 노드를 확장합니다.

   b. 클러스터의 구성 이름을 누릅니다.

   c. 구성 시스템 등록 정보 페이지에서 동적 재구성 사용 가능 확인란을 선택 해제합니다.

d. 저장을 누릅니다.

해당 asadmin 명령은 asadmin set입니다. 구문은 다음과 같습니다.

asadmin set --user *user* --passwordfile *password_file*
cluster_name-config.dynamic-reconfiguration-enabled=false

3. 업그레이드된 응용 프로그램을 대상 domain에 다시 배포합니다. 관리 콘솔을 사용하여 재배포할 경우 도메인이 자동으로 대상이 됩니다. 동적 재구성을 사용할 수 없기 때문에 이전 응용 프로그램은 계속해서 클러스터에서 실행됩니다.
4. asadmin enable-http-lb-application을 사용하여 인스턴스에서 재배포된 응용 프로그램을 사용할 수 있도록 합니다.
5. asadmin disable-http-lb-server를 사용하여 한 서버 인스턴스를 비활성화합니다.
6. asadmin export-http-lb-config를 사용하여 로드 밸런서 구성 파일을 내보냅니다.
7. 내보낸 구성 파일을 웹 서버 인스턴스의 구성 디렉토리에 복사합니다. 예를 들어, Sun Java System Web Server의 경우 위치는 *web_server_install_dir*/https-*host-name*/config/loadbalancer.xml입니다.
8. 시간 초과가 만료될 때까지 대기합니다. 로드 밸런서의 로그 파일을 모니터링하여 인스턴스가 오프라인인지 확인합니다.
9. 클러스터의 다른 인스턴스가 계속 실행되는 동안 비활성화된 서버 인스턴스를 다시 시작합니다. 다시 시작하면 서버가 도메인과 동기화되고 응용 프로그램이 업데이트됩니다.
10. 다시 시작한 서버의 응용 프로그램을 테스트하여 제대로 실행되는지 확인합니다.
11. asadmin enable-http-lb-server를 사용하여 서버 인스턴스를 활성화합니다.
12. asadmin export-http-lb-config를 사용하여 로드 밸런서 구성 파일을 내보냅니다.
13. 구성 파일을 웹 서버의 구성 디렉토리에 복사합니다.
14. 클러스터의 인스턴스마다 단계 5에서 단계 13까지 반복합니다.
15. 모든 서버 인스턴스에 새로운 응용 프로그램이 있고 실행 중일 경우 해당 클러스터에 대한 동적 재구성을 다시 활성화합니다.

두 개의 클러스터에서 업그레이드

1. 이전 버전의 응용 프로그램을 저장하거나 도메인을 백업합니다.

   도메인을 백업하려면 asadmin backup-domain 명령을 사용합니다.

2. 두 클러스터의 동적 재구성이 활성화된 경우 해제합니다.

   관리 콘솔 사용:

   a. 구성 노드를 확장합니다.

   b. 한 클러스터의 구성 이름을 누릅니다.

   c. 구성 시스템 등록 정보 페이지에서 동적 재구성 사용 가능 확인란을 선택 해제합니다.

   d. 저장을 누릅니다.

   e. 두 번째 클러스터에 대해 반복합니다.

   이에 해당하는 `asadmin` 명령은 `asadmin set`입니다. 구문은 다음과 같습니다.

   `asadmin set --user` *user* `--passwordfile` *password_file*
   cluster_name`-config.dynamic-reconfiguration-enabled=false`

3. 업그레이드된 응용 프로그램을 대상 `domain`에 다시 배포합니다. 관리 콘솔을 사용하여 재배포할 경우 도메인이 자동으로 대상이 됩니다. 동적 재구성이 비활성화되어 있기 때문에 이전 응용 프로그램은 계속해서 클러스터에서 실행됩니다.

4. `asadmin enable-http-lb-application`을 사용하여 클러스터에 재배포된 응용 프로그램을 활성화합니다.

5. `asadmin disable-http-lb-server`를 사용하여 로드 밸런서에서 하나의 클러스터를 비활성화합니다.

6. `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.

7. 내보낸 구성 파일을 웹 서버 인스턴스의 구성 디렉토리에 복사합니다. 예를 들어, Sun Java System Web Server의 경우 위치는 *web_server_install_dir*`/https-`*host-name*`/config/loadbalancer.xml`입니다.

8. 시간 초과가 만료될 때까지 대기합니다. 로드 밸런서의 로그 파일을 모니터링하여 클러스터가 오프라인인지 확인합니다.

9. 다른 클러스터를 계속 실행하면서 비활성화된 클러스터를 다시 시작합니다. 다시 시작하면 클러스터가 도메인과 동기화되고 응용 프로그램이 업데이트됩니다.

10. 다시 시작한 클러스터의 응용 프로그램을 테스트하여 제대로 실행되는지 확인합니다.

11. `asadmin enable-http-lb-server`를 사용하여 클러스터를 활성화합니다.

12. `asadmin export-http-lb-config`를 사용하여 로드 밸런서 구성 파일을 내보냅니다.

13. 구성 파일을 웹 서버의 구성 디렉토리에 복사합니다.

14. 다른 클러스터에 대해 단계 5에서 단계 13까지 반복합니다.
15. 모든 서버 인스턴스에 새로운 응용 프로그램이 있고 실행 중일 경우 두 클러스터 모두에 대한 동적 재구성을 다시 활성화합니다.

RMI-IIOP 로드 균형 조정 및 페일오버 정보

RMI-IIOP 로드 균형 조정 및 페일오버에 대한 요구 사항

Sun Java™ System Application Server는 원격 EJB 참조의 고가용성과 RMI-IIOP에서의 이름 서비스 객체를 제공합니다. 이 기능을 사용하기 전에 사용자 환경이 다음 요구 사항을 충족해야 합니다.

- Sun Java System Application Server Enterprise Edition이 설치되어 있어야 합니다.
- 최소한 Application Server 인스턴스 두 개로 된 클러스터가 있어야 합니다.

  클러스터에 대한 자세한 내용은 "클러스터 구성"을 참조하십시오.
- 로드 균형 조정에 참여하는 모든 Application Server 인스턴스와 클러스터에 J2EE 응용 프로그램이 배포되어 있어야 합니다.
- RMI-IIOP 클라이언트 응용 프로그램이 로드 균형 조정을 사용하도록 되어 있어야 합니다.

  다음 RMI-IIOP 클라이언트에 대해 로드 균형 조정을 지원해야 합니다.

  - Application Server에 배포된 EJB에 액세스하는 ACC(Application Client Container)에서 실행 중인 Java 응용 프로그램
  - ACC에서 실행되지 않고 Application Server 인스턴스에 배포된 EJB에 액세스하는 Java 응용 프로그램

  RMI-IIOP 기반 응용 프로그램을 활성화하는 데 필요한 구성 설정은 클라이언트 유형에 따라 다릅니다. 로드 균형 조정을 사용하도록 RMI-IIOP 클라이언트 응용 프로그램을 구성하는 방법에 대한 자세한 내용은 *Sun Java System Application Server Developer's Guide*를 참조하십시오.

  RMI-IIOP 페일오버 및 로드 균형 조정에 대한 자세한 내용은 *Sun Java System Application Server High Availability Administration Guide*를 참조하십시오.

주 SSL에서 RMI-IIOP 페일오버는 지원되지 않습니다.

RMI-IIOP 로드 균형 조정 및 페일오버 알고리즘

Sun Java System Application Server에서는 원격 EJB 참조의 로드 균형 조정 및 RMI-IIOP 경로의 이름 서비스 객체에 대해 랜더마이즈 및 라운드 로빈 알고리즘을 사용합니다.

RMI-IIOP 클라이언트가 새로운 `InitialContext` 객체를 처음 만들면 해당 클라이언트에 대해 사용 가능한 Application Server IIOP 종점 목록이 랜더마이즈됩니다. `InitialContext` 객체의 경우 로드 밸런서는 조회 요청과 다른 `InitialContext` 작업을 목록의 첫 번째 종점으로 지정합니다. 첫 번째 종점을 사용할 수 없는 경우 목록의 두 번째 종점을 사용하는 식으로 계속합니다.

클라이언트가 계속해서 새로운 `InitialContext` 객체를 만들 때마다 종점 목록이 회전하므로 `InitialContext` 작업에 대해 다른 IIOP 종점이 사용됩니다.

`InitialContext` 객체에서 얻은 참조에서 Bean을 구하거나 만들 경우 해당 Bean은 `InitialContext` 객체에 할당된 IIOP 종점 역할을 하는 Application Server 인스턴스에 만들어집니다. 해당 Bean에 대한 참조에는 클러스터의 모든 Application Server 인스턴스의 IIOP 종점 주소가 포함됩니다.

*기본 종점*은 Bean을 조회하거나 만들 때 사용하는 `InitialContext` 종점에 해당하는 Bean 종점입니다. 클러스터의 다른 IIOP 종점은 *대체 종점*으로 지정됩니다. Bean의 기본 종점을 사용할 수 없게 되면 해당 Bean의 추가 요청이 대체 종점 중 하나로 페일오버됩니다.

RMI-IIOP 샘플 응용 프로그램

다음 디렉토리에는 ACC가 있거나 없는 RMI-IIOP 페일오버를 사용하는 것을 설명하는 샘플 응용 프로그램이 포함되어 있습니다.

install_dir`/samples/ee-samples/sfsbfailover`

ACC가 있거나 없는 응용 프로그램을 실행하기 위한 지침은 이 샘플과 같이 제공되는 `index.html` 파일을 참조하십시오. `ee-samples` 디렉토리에는 샘플을 실행하도록 환경을 설정하는 정보도 포함되어 있습니다.

노드 에이전트 구성

이 장에서는 Application Server의 노드 에이전트를 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

노드 에이전트 정보

노드 에이전트

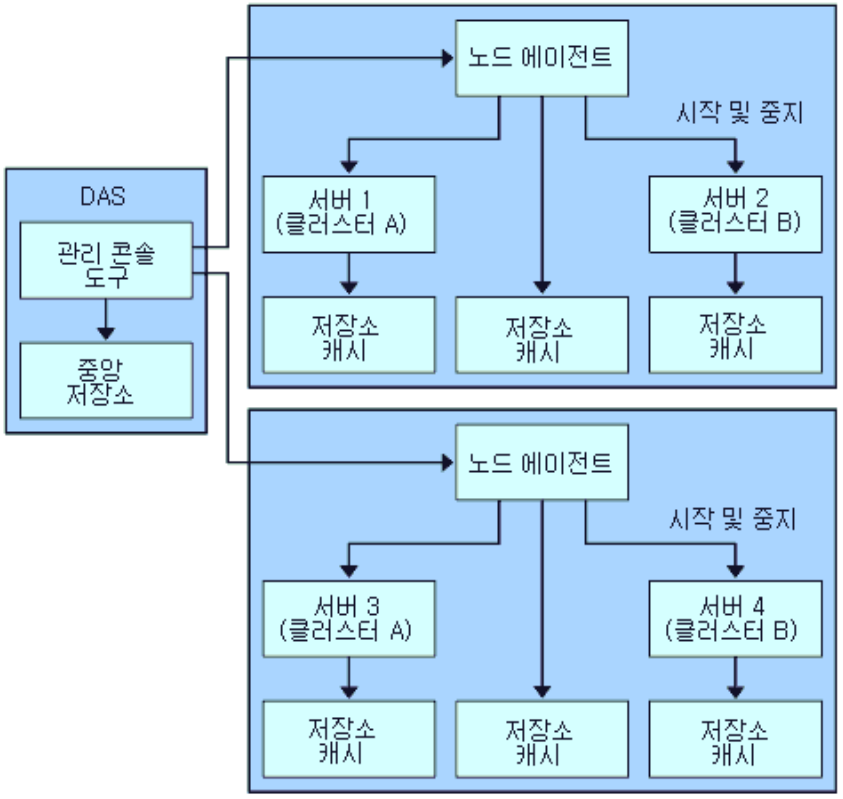
노드 에이전트는 DAS(Domain Administration Server)를 호스트하는 시스템을 포함하여 서버 인스턴스를 호스트하는 모든 시스템에서 필요한 경량의 에이전트입니다. 노드 에이전트는

- DAS(Domain Administration Server)에서 지시하는 대로 서버 인스턴스를 시작, 중지, 작성 및 삭제합니다.

- 실패한 서버 인스턴스를 다시 시작합니다.
- 실패한 서버의 로그 파일 보기를 제공합니다.
- 각 서버 인스턴스의 로컬 구성 저장소를 DAS의 중앙 저장소와 동기화합니다. 각 로컬 저장소에는 해당 서버 인스턴스나 노드 에이전트에 관련된 정보만 포함되어 있습니다.

다음 그림에서는 전반적인 노드 에이전트 구조를 설명합니다.

그림 4-1 노드 에이전트 구조



자동으로 만든 노드 에이전트

Application Server를 설치하면 기본적으로 시스템의 호스트 이름으로 노드 에이전트가 만들어집니다. 이 노드 에이전트는 로컬 시스템에서 수동으로 시작해야 작동합니다.

노드 에이전트 및 서버 인스턴스 관리

노드 에이전트를 실행하지 않더라도 서버 인스턴스를 만들고 삭제할 수 있습니다. 그러나, 노드 에이전트를 실행해야 노드 에이전트를 사용하여 서버 인스턴스를 시작 및 중지할 수 있습니다.

노드 에이전트를 중지하면 노드 에이전트가 관리하는 서버 인스턴스도 중지됩니다.

추가 노드 에이전트

노드 에이전트 하나는 한 도메인만 지원합니다. 한 시스템이 여러 도메인에서 실행하는 인스턴스를 호스트할 경우 복수 노드 에이전트를 실행해야 합니다.

노드 에이전트 자리 표시자

노드 에이전트 자리 표시자를 사용하여 기존 노드 에이전트 없이 서버 인스턴스를 만들고 삭제할 수 있습니다. 자리 표시자는 노드 에이전트의 로컬 시스템에 노드 에이전트를 만들기 전에 DAS(Domain Administration Server)에 만든 노드 에이전트 구성입니다.

주 자리 표시자 노드 에이전트를 만들었으면 이를 사용하여 해당 도메인에 인스턴스를 만들 수 있습니다. 그러나 인스턴스를 시작하기 전에 `asadmin` 명령을 사용하여 인스턴스가 상주하는 시스템에서 로컬로 실제 노드 에이전트를 만들고 시작해야 합니다. 자세한 내용은 106페이지의 "노드 에이전트 만들기" 및 107페이지의 "노드 에이전트 시작"을 참조하십시오.

노드 에이전트 배포

다음 두 가지 방법 중 하나로 노드 에이전트를 구성하고 배포합니다.

- 온라인 배포 - 구성하기 전에 사용자 토폴로지를 알고 있고 도메인에 맞는 하드웨어를 갖고 있는 경우
- 오프라인 배포 - 전체 환경을 설정하기 전에 도메인 및 서버 인스턴스를 구성할 경우

노드 에이전트를 배포하기 전

노드 에이전트를 배포하기 전에 다음 작업을 수행합니다.

1. DAS(Domain Administration Server)를 설치합니다.
2. DAS를 시작합니다.

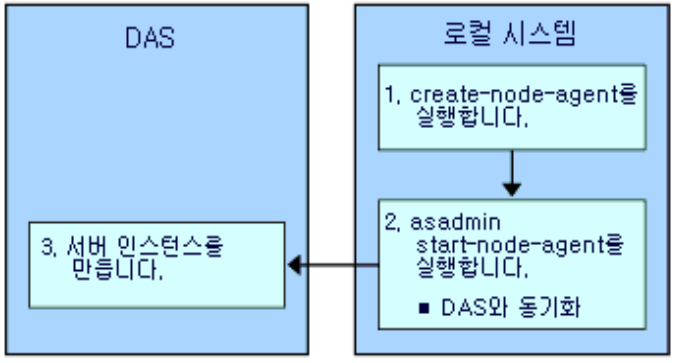
DAS를 실행하고 온라인 또는 오프라인 배포를 시작합니다.

온라인 배포

구성을 시작하기 전에 토폴로지를 알고 있고 도메인에 맞는 하드웨어를 갖고 있는 경우 도메인을 구성하려면 온라인 배포를 사용합니다.

다음 그림에서는 노드 에이전트의 온라인 배포를 요약합니다.

그림 4-2 온라인 노드 에이전트 배포



온라인 배포를 구성하려면 다음 작업을 수행합니다.

1. 서버 인스턴스를 호스트할 모든 시스템에 노드 에이전트를 설치합니다.

   설치 프로그램이나 `asadmin` 명령 `create-node-agent`를 사용합니다. 시스템에 둘 이상의 노드 에이전트가 필요할 경우 `asadmin` 명령 `create-node-agent`를 사용하여 노드 에이전트를 만듭니다.

   자세한 내용은 106페이지의 "노드 에이전트 만들기"를 참조하십시오.

2. asadmin 명령 start-node-agent를 사용하여 노드 에이전트를 시작합니다.

   노드 에이전트를 시작하면 노드 에이전트는 DAS(Domain Administration Server)와 통신합니다. 노드 에이전트가 DAS에 연결되면 노드 에이전트의 구성이 DAS에 만들어집니다. 구성이 존재하면 관리 콘솔에서 그 노드 에이전트를 볼 수 있습니다.

   자세한 내용은 107페이지의 "노드 에이전트 시작"을 참조하십시오.

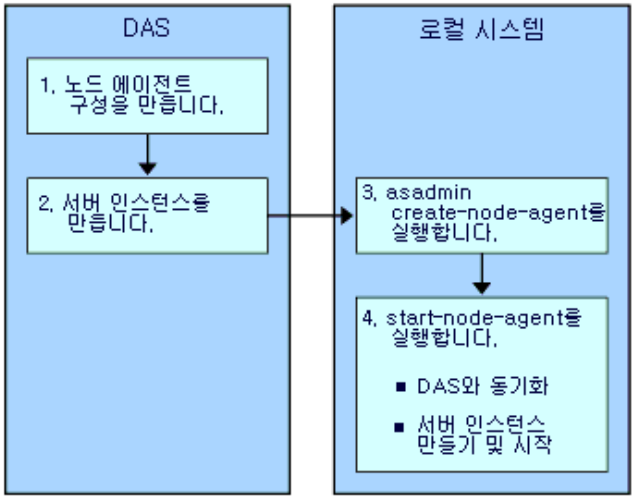
3. 서버 인스턴스 및 클러스터 작성과 응용 프로그램 배포 등 도메인을 구성합니다.

오프라인 배포

오프라인 방법을 사용하면 개별 로컬 시스템을 구성하기 전에 도메인의 항목을 쉽게 정의하고 재배열할 수 있습니다.

다음 그림에서는 오프라인 배포 단계를 요약합니다.

그림 4-3 오프라인 노드 에이전트 배포



로컬 시스템에 노드 에이전트를 설치하기 전에 도메인과 서버 인스턴스를 구성하려면(오프라인 구성) 다음 단계를 수행합니다.

1. DAS에 자리 표시자 노드 에이전트를 만듭니다.

   자세한 내용은 102페이지의 "노드 에이전트 자리 표시자 만들기"를 참조하십시오.

2. 서버 인스턴스와 클러스터를 만들고 응용 프로그램을 배포합니다.

3. 서버 인스턴스를 호스트할 모든 시스템에 노드 에이전트를 설치합니다.

   설치 프로그램이나 `asadmin` 명령 `create-node-agent`를 사용합니다. 노드 에이전트의 이름은 이전에 만든 자리 표시자 노드 에이전트와 동일해야 합니다.

   자세한 내용은 106페이지의 "노드 에이전트 만들기"를 참조하십시오.

4. `asadmin` 명령 `start-node-agent`를 사용하여 노드 에이전트를 시작합니다.

   노드 에이전트가 시작되면 DAS에 바인딩되고 이전에 연관된 모든 서버 인스턴스를 만듭니다.

   자세한 내용은 107페이지의 "노드 에이전트 시작"을 참조하십시오.

노드 에이전트 및 DAS 동기화

구성 데이터가 DAS의 저장소(중앙 저장소)에 저장되고 노드 에이전트의 로컬 시스템에도 캐시되기 때문에 이 둘을 동기화해야 합니다.

노드 에이전트 동기화

노드 에이전트를 처음 시작하면 중앙 저장소의 최신 정보에 대한 요청을 DAS에 전송합니다. DAS에 연결하여 구성 정보를 가져오면 노드 에이전트는 그 DAS에 *바인딩*됩니다.

DAS에 자리 표시자 노드 에이전트를 만든 경우 노드 에이전트를 처음 시작하면 DAS의 중앙 저장소에서 구성을 가져옵니다.

초기 시작 중에 DAS가 실행되지 않아서 노드 에이전트가 DAS에 연결할 수 없는 경우 노드 에이전트가 중지되고 *바인딩되지 않습니다*.

도메인에서 노드 에이전트 구성을 변경하면 노드 에이전트를 실행 중일 때 로컬 시스템의 노드 에이전트와 자동으로 통신합니다.

DAS에서 노드 에이전트 구성을 삭제하면 다음에 노드 에이전트가 동기화할 때 스스로 중지하고 삭제 대기로 표시합니다. 로컬 asadmin 명령 delete-node-agent를 사용하여 수동으로 삭제합니다.

서버 인스턴스 동기화

관리 콘솔이나 asadmin 도구를 사용하여 서버 인스턴스를 명시적으로 시작하면 서버 인스턴스는 중앙 저장소와 동기화됩니다. 이 동기화가 실패하면 서버 인스턴스가 시작되지 않습니다.

관리 콘솔이나 asadmin 도구를 통한 명시적인 요청 없이 노드 에이전트가 서버 인스턴스를 시작하면 서버 인스턴의 저장소 캐시가 동기화되지 않습니다. 서버 인스턴스는 해당 캐시에 저장된 구성으로 실행됩니다.

대용량 응용 프로그램 동기화

환경에 동기화할 대용량 응용 프로그램이 포함되거나 사용 가능한 메모리가 제한된 경우 JVM 옵션을 조정하여 메모리 사용을 제한할 수 있습니다. 이렇게 조정하면 메모리 부족 오류 발생 가능성이 줄어듭니다. 인스턴스 동기화 JVM은 기본 설정을 사용하지만 JVM 옵션을 구성하여 변경할 수 있습니다.

INSTANCE-SYNC-JVM-OPTIONS 등록 정보를 사용하여 JVM 옵션을 설정합니다. 등록 정보를 설정하는 명령은 다음과 같습니다.

```
asadmin set domain.node-agent.node_agent_name.property.INSTANCE-SYNC-JVM-OPTIONS="JVM_options"
```

예를 들면 다음과 같습니다.

```
asadmin set domain.node-agent.node0.property.INSTANCE-SYNC-JVM-OPTIONS="-Xmx32m -Xss2m"
```

이 예에서 노드 에이전트는 node0이고 JVM 옵션은 -Xmx32m -Xss2m입니다.

JVM 옵션에 대한 자세한 내용은 다음을 참조하십시오.

java.sun.com/docs/hotspot/VMOptions.html

주 구성에서 등록 정보를 추가하거나 변경한 경우 노드 에이전트가 자동으로 동기화되지 않으므로 INSTANCE-SYNC-JVM-OPTIONS 등록 정보를 변경한 후 노드 에이전트를 다시 시작합니다.

노드 에이전트 로그 보기

모든 노드 에이전트에는 고유한 로그 파일이 있습니다. 노드 에이전트에 문제가 생기면 다음 위치에 있는 로그 파일을 참조하십시오.

node_agent_dir/*node_agent_name*/agent/logs/server.log.

노드 에이전트 로그에서 서버의 로그를 확인하여 문제점에 대한 자세한 메시지를 확인할 것을 권장할 때도 있습니다.

서버 로그는 다음 위치에 있습니다.

node_agent_dir/*node_agent_name*/*server_name*/logs/server.log

*node_agent_dir*의 기본 위치는 *install_dir*/nodeagents입니다.

관리 콘솔 및 asadmin 도구를 통해 사용 가능한 작업

노드 에이전트의 경우 일부 작업은 노드 에이전트가 실행되는 시스템에서 로컬로 실행해야 하지만, 일부 작업은 DAS에서 실행할 수 있습니다. 로컬로 실행해야 하는 작업은 노드 에이전트가 상주하는 시스템에서 asadmin 도구를 통해서만 작업이 가능합니다. DAS에서 실행되는 작업은 관리 콘솔과 asadmin 도구를 통해 할 수 있습니다.

다음 표에서는 작업과 작업이 실행되는 위치를 요약합니다.

표 4-1 관리 콘솔과 asadmin 명령을 통해 사용 가능한 작업

작업	관리 콘솔	asadmin 명령
DAS에 노드 에이전트 자리 표시자/구성 만들기	노드 에이전트 자리 표시자 만들기 페이지	create-node-agent-config
노드 에이전트 만들기	사용할 수 없음	create-node-agent
노드 에이전트 시작	사용할 수 없음	start-node-agent
노드 에이전트 중지	사용할 수 없음	stop-node agent
DAS에서 노드 에이전트 구성 삭제	노드 에이전트 페이지	delete-node-agent-config
로컬 시스템에서 노드 에이전트 삭제	사용할 수 없음	delete-node-agent

표 4-1 관리 콘솔과 asadmin 명령을 통해 사용 가능한 작업

작업	관리 콘솔	asadmin 명령
노드 에이전트 구성 편집	노드 에이전트 페이지	set
노드 에이전트 나열	노드 에이전트 페이지	list-node-agents

노드 에이전트에 대한 관리 콘솔 작업

일반 노드 에이전트 정보 보기

노드 에이전트나 노드 에이전트 자리 표시자를 만든 경우 설정을 보려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 노드 에이전트 노드를 선택합니다.
2. 노드 에이전트 이름을 누릅니다.

   노드 에이전트가 이미 있지만 여기에 표시되지 않는 경우 `asadmin start-node-agent`를 사용하여 노드 에이전트의 호스트 시스템에서 노드 에이전트를 시작합니다. 자세한 내용은 107페이지의 "노드 에이전트 시작"을 참조하십시오.
3. 노드 에이전트의 호스트 이름을 확인합니다.

   호스트 이름이 알 수 없는 호스트일 경우 노드 에이전트가 DAS와 초기 연결하지 않았기 때문입니다.
4. 노드 에이전트 상태를 확인합니다.

   실행 중. 노드 에이전트를 제대로 만들었고 현재 실행 중입니다.

   실행 중이 아님. 다음 조건 중 하나가 존재합니다.

   - 로컬 시스템에서 노드 에이전트를 만들었지만 시작한 적이 없습니다.

- 노드 에이전트를 시작했지만 중지되었습니다.

랑데부 대기 중. 노드 에이전트가 로컬 시스템에 작성된 적이 없는 자리 표시자입니다.

노드 에이전트 작성 및 시작에 대한 자세한 내용은 106페이지의 "노드 에이전트 만들기" 및 107페이지의 "노드 에이전트 시작"을 참조하십시오.

5. 시작 시 인스턴스를 시작할지 여부를 선택합니다.

   노드 에이전트를 시작할 때 노드 에이전트와 연관된 서버 인스턴스를 자동으로 시작하려면 예를 선택합니다. 인스턴스를 수동으로 시작하려면 아니오를 선택합니다.

6. 노드 에이전트가 DAS와 연결했는지 여부를 확인합니다.

   노드 에이전트가 DAS에 연결하지 않았으면 성공적으로 시작되지 않습니다.

7. 노드 에이전트와 연결된 서버 인스턴스를 관리합니다.

   노드 에이전트가 실행 중일 경우 인스턴스 이름 옆에 있는 확인란을 선택하고 시작이나 중지를 눌러 인스턴스를 시작하거나 중지합니다.

노드 에이전트 자리 표시자 만들기

노드 에이전트를 호스트하는 시스템에서 로컬로 노드 에이전트를 만들어야 하기 때문에 관리 콘솔을 통해서만 노드 에이전트의 자리 표시자를 만들 수 있습니다. 이 자리 표시자는 노드 에이전트가 아직 없는 노드 에이전트 구성입니다.

자리 표시자를 만든 후 노드 에이전트를 호스트하는 시스템에서 `asadmin` 명령 `create-node-agent`를 사용하여 작성을 완료합니다. 자세한 내용은 106페이지의 "노드 에이전트 만들기"를 참조하십시오.

1. 트리 구성 요소에서 노드 에이전트 노드를 선택합니다.
2. 노드 에이전트 페이지에서 새로 만들기를 누릅니다.
3. 현재 노드 에이전트 자리 표시자 페이지에서 새로운 노드 에이전트의 이름을 입력합니다.

   도메인의 모든 노드 에이전트 이름, 서버 인스턴스 이름, 클러스터 이름 및 구성 이름에서 이름이 고유해야 합니다.

4. 확인을 누릅니다.

   노드 에이전트 페이지에 새로운 노드 에이전트의 자리 표시자가 나열됩니다.

해당 asadmin 명령: create-node-agent-config

노드 에이전트 구성 삭제

관리 콘솔을 통해서만 도메인에서 노드 에이전트 구성을 삭제할 수 있습니다. 실제 노드 에이전트는 삭제할 수 없습니다. 노드 에이전트 자체를 삭제하려면 노드 에이전트의 로컬 시스템에서 asadmin 명령 delete-node-agent를 실행합니다. 자세한 내용은 107페이지의 "노드 에이전트 삭제"를 참조하십시오.

노드 에이전트 구성을 삭제하기 전에 노드 에이전트를 중지해야 하고 연관된 인스턴스가 없어야 합니다. 노드 에이전트를 중지하려면 asadmin 명령 stop-node-agent를 사용합니다. 자세한 내용은 107페이지의 "노드 에이전트 중지"를 참조하십시오.

1. 트리 구성 요소에서 노드 에이전트 노드를 선택합니다.
2. 노드 에이전트 페이지에서 삭제할 노드 에이전트 옆에 있는 확인란을 선택합니다.
3. 삭제를 누릅니다.

해당 asadmin 명령: delete-node-agent-config

노드 에이전트 구성 편집

노드 에이전트 구성을 편집하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 노드 에이전트 노드를 확장합니다.
2. 편집하려면 노드 에이전트 구성을 선택합니다.
3. 노드 에이전트 일반 정보 페이지에서 에이전트를 시작할 때 에이전트의 서버 인스턴스를 시작할지 여부를 선택합니다. 이 페이지에서 인스턴스를 수동으로 시작 및 중지할 수도 있습니다.

이 구성이 자리 표시자 노드 에이전트를 위한 것일 경우 asadmin create-node-agent를 사용하여 실제 노드 에이전트를 만들 때 이 구성을 사용합니다. 노드 에이전트 만들기에 대한 자세한 내용은 106페이지의 "노드 에이전트 만들기"를 참조하십시오.

이 구성이 기존 노드 에이전트를 위한 것일 경우 노드 에이전트 구성 정보가 자동으로 동기화됩니다.

노드 에이전트 영역 편집

노드 에이전트에 연결하는 사용자를 위한 인증 영역을 설정합니다. 관리 사용자만 노드 에이전트에 액세스해야 합니다.

1. 트리 구성 요소에서 노드 에이전트 노드를 확장합니다.
2. 편집하려면 노드 에이전트 구성을 선택합니다.
3. 인증 영역 탭을 누릅니다.
4. 노드 에이전트 영역 편집 페이지에서 영역을 입력합니다.

   노드 에이전트를 만들 때 만들어진 `admin-realm`이 기본값입니다. 다른 영역을 사용하려면 도메인에서 제어하는 *모든* 구성 요소의 영역을 대체합니다. 그렇지 않으면 구성 요소가 제대로 통신하지 못합니다.
5. 클래스 이름 필드에서 영역을 구현하는 Java 클래스를 지정합니다.
6. 필수 등록 정보를 모두 추가합니다.

   인증 영역에는 공급자 관련 등록 정보가 필요한데, 특정 구현에서 요구하는 내용에 따라 다릅니다.

JMX에 대해 노드 에이전트 Listener 편집

노드 에이전트에서는 JMX를 사용하여 DAS와 통신합니다. 따라서 JMX 요청을 수신하려면 포트와 다른 listener 정보가 있어야 합니다.

1. 트리 구성 요소에서 노드 에이전트 노드를 확장합니다.
2. 편집하려면 노드 에이전트 구성을 선택합니다.
3. JMX 탭을 누릅니다.
4. Listener가 고유한 포트 값을 사용하여 서버에 대한 모든 IP 주소에서 수신할 경우 주소 필드에 0.0.0.0을 입력합니다. 그렇지 않으면 서버에 대한 유효한 IP 주소를 입력합니다.
5. 포트 필드에서 노드 에이전트의 JMX 커넥터가 수신하는 포트 값을 입력합니다. IP 주소가 0.0.0.0일 경우 포트 번호가 고유해야 합니다.
6. JMX 프로토콜 필드에서 JMX 커넥터가 지원하는 프로토콜을 입력합니다.

   기본값은 rmi_jrmp입니다.

7. 모든 IP 주소에 대한 연결을 허용하려면 모든 주소 허용 옆에 있는 확인란을 누릅니다.

   노드 에이전트는 네트워크 카드에 연결된 특정 IP 주소에서 수신하거나 모든 IP 주소에서 수신합니다. 모든 주소를 허용하면 "listening host address" 등록 정보에 `0.0.0.0` 값이 입력됩니다.

8. 영역 이름 필드에서 Listener에 대한 인증을 처리하는 영역 이름을 입력합니다.

이 페이지의 보안 섹션에서 Listener가 SSL, TLS 또는 SSL 및 TLS 보안을 둘 다 사용하도록 구성합니다.

보안 listener를 설정하려면 다음 작업을 수행합니다.

1. 보안 필드에서 사용 가능 확인란을 선택합니다.

   보안은 기본적으로 활성화되어 있습니다.

2. 이 Listener를 사용할 때 클라이언트가 자신을 서버에 인증하도록 하려면 클라이언트 인증 필드에서 사용 가능 확인란을 선택합니다.
3. 인증서 별명 필드에 기존 서버의 키 쌍과 인증서를 입력합니다. 자세한 내용은 보안 장을 참조하십시오.
4. SSL3/TLS 섹션:
   a. Listener에서 활성화할 보안 프로토콜을 선택합니다. SSL3, TLS 또는 두 가지 프로토콜을 모두 선택해야 합니다.
   b. 프로토콜이 사용하는 암호 제품군을 선택합니다. 모든 암호 제품군을 사용하려면 지원되는 모든 암호화 제품군을 선택합니다.
5. 저장을 누릅니다.

asadmin 도구에서 노드 에이전트에 대한 작업

노드 에이전트 만들기

노드 에이전트를 만들려면 노드 에이전트가 실행 중인 시스템에서 로컬로 asadmin 명령 create-node-agent를 실행합니다.

예를 들면 다음과 같습니다.

```
$ asadmin create-node-agent --host myhost --port 4849 ---user admin
nodeagent1
```

여기에서 myhost는 DAS 호스트 이름, 4849는 DAS 포트 번호, admin은 DAS 사용자, nodeagent1은 만들려는 노드 에이전트 이름입니다.

노드 에이전트의 기본 이름은 노드 에이전트가 만들어지는 호스트 이름입니다.

노드 에이전트 자리 표시자를 이미 만든 경우 노드 에이전트 자리 표시자와 동일한 이름을 사용하여 연관된 노드 에이전트를 만듭니다. 노드 에이전트 자리 표시자를 만들지 않았지만 DAS를 실행 중이고 연결할 수 있는 경우 create-node-agent 명령으로도 DAS에 노드 에이전트 구성(자리 표시자)을 만들 수 있습니다.

명령 구문에 대한 자세한 설명은 명령에 대한 온라인 도움말을 참조하십시오.

주 다음 상황에서는 DNS에 연결 가능한 호스트 이름을 지정해야 합니다.

1. 도메인이 서브넷 경계에 걸쳐 있을 경우(즉, 노드 에이전트와 DAS가 서로 다른 도메인(예: sun.com 및 java.com)에 있을 경우)
2. DNS에 등록되지 않은 호스트 이름의 DHCP 시스템을 사용할 경우

도메인과 노드 에이전트를 만들 때 이들에 대한 호스트 이름을 명시적으로 지정하여 DNS 연결 가능한 호스트 이름을 지정합니다.

create-domain --domainproperties domain.hostName=*DAS-host-name*

create-node-agent --host *DAS-host-name* --agentproperties

remoteclientaddress=*node-agent-host-name*

다른 방법은 플랫폼에 관련된 hosts hostname/IP 결정 파일을 업데이트하여 호스트 이름이 올바른 IP 주소로 변환될 수 있도록 하는 것입니다. 그러나 DHCP를 사용하여 다시 연결할 경우 다른 IP 주소가 지정될 수 있습니다. 그럴 경우 각 서버에서 호스트 결정 파일을 업데이트해야 합니다.

노드 에이전트 시작

노드 에이전트가 실행 중이어야 노드 에이전트로 서버 인스턴스를 관리할 수 있습니다. 노드 에이전트가 상주하는 시스템에서 asadmin 명령 start-node-agent를 로컬로 실행하여 노드 에이전트를 시작합니다.

예를 들면 다음과 같습니다.

```
$ asadmin start-node-agent --user admin nodeagent1
```

여기에서 admin은 관리 사용자, nodeagent1은 시작하려는 노드 에이전트입니다.

명령 구문에 대한 자세한 설명은 명령에 대한 온라인 도움말을 참조하십시오.

노드 에이전트 중지

실행 중인 노드 에이전트를 중지하려면 노드 에이전트가 상주하는 시스템에서 asadmin 명령 stop-node-agent를 실행합니다. stop-node-agent 명령은 노드 에이전트가 관리하는 모든 서버 인스턴스를 중지합니다.

예를 들면 다음과 같습니다.

```
$ asadmin stop-node-agent nodeagent1
```

여기에서 nodeagent1은 사용자 노드 에이전트입니다.

명령 구문에 대한 자세한 설명은 명령에 대한 온라인 도움말을 참조하십시오.

노드 에이전트 삭제

노드 에이전트 파일을 삭제하려면 노드 에이전트가 상주하는 시스템에서 asadmin 명령 delete-node-agent를 실행합니다.

노드 에이전트를 삭제하기 전에 노드 에이전트를 중지해야 합니다. 시작하지 않았거나 DAS에 연결하지 못했던(즉 바인딩되지 않은) 노드 에이전트를 삭제할 수도 있습니다.

예를 들면 다음과 같습니다.

```
$ asadmin delete-node-agent nodeagent1
```

여기에서 nodeagent1은 사용자 노드 에이전트입니다.

명령 구문에 대한 자세한 설명은 명령에 대한 온라인 도움말을 참조하십시오.

노드 에이전트를 삭제할 경우 관리 콘솔이나 `asadmin delete-node-agent-config` 명령을 사용하여 DAS에서 노드 에이전트의 구성도 삭제해야 합니다.

응용 프로그램 배포

이 장에서는 Application Server에서 J2EE 응용 프로그램을 배포(설치)하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

배포 정보

배포 라이프사이클

Application Server를 설치하고 도메인을 시작한 후 J2EE 응용 프로그램과 모듈을 배포(설치)할 수 있습니다. 배포 중 그리고 응용 프로그램이 변경되면 응용 프로그램이나 모듈은 다음 단계를 거칩니다.

1. 초기 배포

   응용 프로그램이나 모듈을 배포하기 전에 도메인을 시작합니다.

   특정한 독립 실행형 서버 인스턴스나 클러스터에 응용 프로그램 또는 모듈을 배포(설치)합니다. 응용 프로그램과 모듈은 아카이브 파일로 패키지화되기 때문에 배포 중에 아카이브 파일 이름을 지정합니다. 기본값은 기본 서버 인스턴스 `server`로 배포하는 것입니다.

   서버 인스턴스나 클러스터에 배포할 경우 응용 프로그램 또는 모듈은 도메인의 중앙 저장소에 존재하며 배포 대상인 클러스터나 서버 인스턴스에 의해 참조됩니다.

   관리 콘솔이 아니라 `asadmin deploy` 명령을 사용하여 도메인에 배포할 수도 있습니다. 응용 프로그램이나 모듈을 도메인에만 배포할 경우 응용 프로그램이나 모듈은 도메인의 중앙 저장소에 존재하지만 단계 3에서 설명한 대로 참조를 추가할 때까지는 서버 인스턴스나 클러스터에 의해 참조되지 않습니다.

   배포는 동적입니다. 응용 프로그램을 사용하기 위해 응용 프로그램이나 모듈을 배포한 후 서버 인스턴스를 다시 시작할 필요가 없습니다. 다시 시작할 경우 모든 배포된 응용 프로그램과 모듈이 배포되고 사용 가능합니다.

2. 활성화 또는 비활성화

   기본적으로 배포된 응용 프로그램이나 모듈은 활성화되어 있습니다. 이는 액세스 가능한 서버 인스턴스나 클러스터에 응용 프로그램 또는 모듈을 배포한 경우 이들을 실행할 수 있고 클라이언트에서 액세스할 수 있음을 의미합니다. 액세스를 방지하려면 응용 프로그램이나 모듈을 비활성화합니다. 비활성화된 응용 프로그램이나 모듈은 도메인에서 제거되지 않으므로 배포 후 쉽게 활성화할 수 있습니다.

3. 배포된 응용 프로그램이나 모듈의 대상 추가 또는 삭제

   응용 프로그램이나 모듈이 일단 배포되면 중앙 저장소에 존재하고 여러 서버 인스턴스나 클러스터에서 참조할 수 있습니다. 처음에는 배포 대상인 서버 인스턴스나 클러스터에서 응용 프로그램이나 모듈을 참조합니다.

   응용 프로그램이나 모듈을 배포한 후 이를 참조하는 서버 인스턴스와 클러스터를 변경하려면 관리 콘솔을 사용하여 응용 프로그램이나 모듈의 대상을 변경하거나 asadmin 도구를 사용하여 응용 프로그램 참조를 변경합니다. 응용 프로그램이 중앙 저장소에 저장되기 때문에 대상을 추가하거나 삭제하면 다른 대상에 있는 동일한 버전의 응용 프로그램이 추가되거나 삭제됩니다. 그러나 둘 이상의 대상에 배포된 응용 프로그램은 한 대상에서 활성화하고 다른 대상에서는 비활성화할 수 있습니다. 대상에서 응용 프로그램을 참조하더라도 해당 대상에서 응용 프로그램을 활성화하지 않으면 사용자가 사용할 수 없습니다.

4. 재배포

   배포된 응용 프로그램이나 모듈을 대체하려면 다시 배포하십시오. 재배포하면 이전에 배포된 응용 프로그램이나 모듈을 자동으로 배포 해제하고 새로운 응용 프로그램이나 모듈로 대체합니다.

   관리 콘솔을 통해 재배포하면 재배포된 응용 프로그램이나 모듈은 도메인으로 배포되고 동적 재구성을 활성화한 경우, 이를 참조하는 모든 독립 실행형 또는 클러스터링된 서버 인스턴스는 자동으로 새로운 버전을 수신합니다. `asadmin deploy` 명령을 사용하여 재배포할 경우 `domain`을 대상으로 지정합니다.

   작업 환경의 경우 서비스를 중단하지 않은 채 응용 프로그램을 업그레이드하는 롤링 업그레이드를 사용합니다. 자세한 내용은 88페이지의 "롤링 업그레이드 정보"를 참조하십시오.

5. 배포 해제

   응용 프로그램이나 모듈을 제거하려면 배포 해제합니다.

J2EE 아카이브 파일의 유형

소프트웨어 공급자가 응용 프로그램이나 모듈을 아카이브 파일로 패키지화합니다. 응용 프로그램이나 모듈을 배포하려면 아카이브 파일 이름을 지정합니다. 아카이브 파일의 내용과 구조는 J2EE 플랫폼의 사양에 의해 정의됩니다. J2EE 아카이브 파일의 유형은 다음과 같습니다.

- WAR(Web Application Archive): WAR 파일은 정적 HTML 페이지, JAR 파일, 태그 라이브러리 및 유틸리티 클래스 뿐만 아니라 서블릿 및 JSP 같은 웹 구성 요소로 구성됩니다. WAR 파일 이름은 `.war` 확장자를 갖습니다.
- EJB JAR: EJB JAR 파일에는 EJB 기술에 사용되는 구성 요소인 하나 이상의 Enterprise Bean이 포함되어 있습니다. 또한 EJB JAR 파일에는 Enterprise Bean에 필요한 유틸리티 클래스도 포함되어 있습니다. EJB JAR 파일 이름은 `.jar` 확장자를 갖습니다.
- J2EE 응용 프로그램 클라이언트 JAR: 이 JAR 파일에는 RMI/IIOP를 통해 Enterprise Bean 같은 서버측 구성 요소에 액세스하는 J2EE 응용 프로그램 클라이언트의 코드가 포함되어 있습니다. 관리 콘솔에서는 J2EE 응용 프로그램 클라이언트를 "응용 프로그램 클라이언트"라고 합니다. J2EE 응용 프로그램 클라이언트 JAR 파일 이름은 `.jar` 확장자를 갖습니다.

- RAR(Resource Adapter Archive): RAR 파일에는 자원 어댑터가 보관됩니다. J2EE Connector Architecture 사양에서 정의한 자원 어댑터는 Enterprise Bean, 웹 구성 요소 및 응용 프로그램 클라이언트가 자원 및 외부 엔터프라이즈 시스템에 액세스할 수 있게 해주는 이식 가능한 구성 요소입니다. 자원 어댑터는 커넥터라고 합니다. RAR 파일 이름은 `.rar` 확장자를 갖습니다.
- EAR(Enterprise Application Archive): EAR 파일에는 하나 이상의 WAR, EJB JAR, RAR 또는 J2EE 응용 프로그램 클라이언트 JAR 파일이 보관됩니다. EAR 파일 이름은 `.ear` 확장자를 갖습니다.

소프트웨어 공급자가 응용 프로그램을 하나의 EAR 파일이나 별도의 WAR, EJB JAR 및 응용 프로그램 클라이언트 JAR 파일로 어셈블할 수 있습니다. 관리 도구에서 배포 페이지와 명령은 모든 유형의 파일에 있어서 유사합니다.

이름 지정 규약

지정한 도메인에서 배포된 응용 프로그램과 모듈의 이름은 고유해야 합니다.

- 관리 콘솔을 사용하여 배포할 경우 응용 프로그램 이름 필드에서 이름을 지정합니다.
- `asadmin deploy` 명령을 사용하여 배포할 경우 응용 프로그램이나 모듈의 기본 이름은 배포할 JAR 파일의 접두어입니다. 예를 들어, `hello.war` 파일의 경우 웹 응용 프로그램 이름은 `hello`입니다. 기본 이름을 대체하려면 `--name` 옵션을 지정합니다.

응용 프로그램 내에서 유형이 다른 모듈은 동일한 이름을 가질 수 있습니다. 응용 프로그램을 배포할 때 개별 모듈이 있는 디렉토리 이름에는 `_jar`, `_war` 및 `_rar` 접미어가 붙습니다. 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 Java System Application Server 뿐만 아니라 다른 J2EE 응용 프로그램 서버에도 적용됩니다.

EJB에 대한 JNDI 조회 이름 역시 고유해야 합니다. 일관된 이름 지정 규약을 설정하는 것이 좋습니다. 예를 들어, EJB 이름에 응용 프로그램 이름과 모듈 이름을 추가하는 것도 고유한 이름을 유지하는 한 가지 방법입니다. 이 경우 mycompany.pkging.pkgingEJB.MyEJB는 응용 프로그램 pkging.ear에 패키지화된 pkgingEJB.jar 모듈의 EJB에 대한 JNDI 이름이 됩니다.

운영 체제에서 사용할 수 없는 공백이나 문자가 패키지 및 파일 이름에 포함되지 않도록 하십시오.

응용 프로그램을 배포하기 위한 관리 콘솔 작업

엔터프라이즈 응용 프로그램 배포

엔터프라이즈 응용 프로그램은 모든 유형의 J2EE 독립 실행형 모듈을 포함하는 아카이브 파일 유형(예: WAR 및 EJB JAR 파일)인 EAR 파일에 패키지화됩니다.

엔터프라이즈 응용 프로그램을 배포(설치)하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 엔터프라이즈 응용 프로그램 노드를 선택합니다.
3. 엔터프라이즈 응용 프로그램 페이지에서 배포를 누릅니다.
4. 배포 페이지에서 배포할 EAR 파일의 위치를 지정합니다.

   서버 시스템은 Application Server와 Domain Administration Server를 실행하는 호스트입니다. 클라이언트 시스템은 브라우저를 통해 관리 콘솔을 볼 수 있는 호스트입니다.

a. 파일이 클라이언트 시스템에 상주하거나 클라이언트 시스템에서 액세스하려면, 라디오 버튼을 눌러 Application Server에 업로드할 패키지 파일을 지정합니다.

찾아보기를 눌러 파일을 찾거나 파일의 전체 경로를 입력합니다.

b. 파일이 서버 시스템에 상주하거나 확장된 디렉토리에서 압축 해제된 응용 프로그램을 배포하려면 라디오 버튼을 눌러 패키지 파일을 지정하거나 서버에서 액세스할 수 있는 디렉토리 경로를 지정합니다.

파일 또는 디렉토리의 전체 경로를 입력합니다. 확장된 디렉토리에서 배포하는 것은 고급 개발자를 위한 것이며 작업 환경에는 권장되지 않습니다.

5. 다음을 눌러 Deploy Enterprise Application 페이지를 표시합니다.

6. 엔터프라이즈 응용 프로그램 배포 페이지에서 응용 프로그램의 설정을 지정합니다.

a. 응용 프로그램 이름 필드에서 파일 이름의 접두어인 기본 이름을 그대로 두거나 다른 이름을 입력합니다.(파일 업로드를 선택한 경우 기본 이름이 표시됩니다.) 응용 프로그램 이름은 고유해야 합니다.

b. 기본적으로 응용 프로그램은 배포하자마자 사용할 수 있습니다. 배포 후 응용 프로그램을 사용할 수 없게 비활성화하려면 비활성화 라디오 버튼을 선택합니다.

c. 응용 프로그램을 이미 배포한 경우 재배포 확인란을 선택하여 재배포합니다. 그렇지 않으면 오류가 표시됩니다. 다른 응용 프로그램 이름을 선택하여 새로운 이름 아래에 배포할 수도 있습니다.

d. 배포 전에 파일의 구조 및 내용을 검증하려면 검증자 확인란을 선택합니다. 큰 응용 프로그램을 검증할 경우 시간이 많이 소모될 수 있습니다. 파일이 손상되거나 이식 불가능한 것으로 의심될 경우 파일을 검증합니다.

e. JSP 페이지를 사전 컴파일하려면 JSP 확인란을 선택합니다. 확인란을 선택하지 않은 경우, JSP 페이지는 처음 액세스되는 런타임 시 컴파일됩니다. 컴파일은 작업 환경에서 시간이 많이 소요될 수 있으므로 확인란을 선택합니다.

f. 고가용성 설정을 선택합니다.

응용 프로그램에 대한 고가용성을 활성화하려면 가용성 확인란을 선택합니다. 응용 프로그램에 대한 가용성을 활성화한 경우 더 높은 모든 수준에서도(명명된 구성 및 웹 컨테이너 또는 EJB 컨테이너) 활성화해야 합니다.

g. 응용 프로그램을 배포할 대상을 선택합니다.

사용 가능한 대상 목록에서 대상을 선택한 후 추가를 누릅니다. 클러스터나 독립 실행형 인스턴스가 대상이 될 수 있습니다. 대상을 선택하지 않은 경우 응용 프로그램이 기본 서버 인스턴스 `server`에 배포됩니다.

재배포 중인 경우 대상을 선택하지 마십시오. 여기서 선택한 대상은 무시됩니다. 배포된 응용 프로그램을 참조하는 클러스터링된 대상이나 독립 실행형 서버 인스턴스는 클러스터나 독립 실행형 인스턴스에 대해 동적 재구성을 활성화한 경우 자동으로 새로운 재배포된 응용 프로그램을 참조합니다. 서비스의 중단 없이 응용 프로그램을 재배포하는 방법에 대한 자세한 내용은 88페이지의 "응용 프로그램 업그레이드"를 참조하십시오.

h. RMI 스텁을 생성할지 선택합니다.

RMI 스텁을 생성하기로 선택한 경우 정적 RMI-IIOP 스텁이 `client.jar`에 생성됩니다.

7. 응용 프로그램을 배포하려면 확인을 누릅니다.

해당 `asadmin` 명령: `deploy`

웹 응용 프로그램 배포

웹 응용 프로그램은 서블릿 및 JSP 페이지 같은 구성 요소를 포함하는 아카이브 파일 유형인 WAR 파일로 패키지화됩니다.

웹 응용 프로그램을 배포(설치)하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.

2. 웹 응용 프로그램 노드를 선택합니다.

3. 웹 응용 프로그램 페이지에서 배포를 누릅니다.

4. 배포 페이지에서 배포할 WAR 파일의 위치를 지정합니다.

서버 시스템은 Application Server와 Domain Administration Server를 실행하는 호스트입니다. 클라이언트 시스템은 브라우저를 통해 관리 콘솔을 볼 수 있는 호스트입니다.

a. 파일이 클라이언트 시스템에 상주하거나 클라이언트 시스템에서 액세스하려면, 라디오 버튼을 눌러 Application Server에 업로드할 패키지 파일을 지정합니다.

찾아보기를 눌러 파일을 찾거나 파일의 전체 경로를 입력합니다.

b. 파일이 서버 시스템에 상주하거나 확장된 디렉토리에서 압축 해제된 응용 프로그램을 배포하려면 라디오 버튼을 눌러 패키지 파일을 지정하거나 서버에서 액세스할 수 있는 디렉토리 경로를 지정합니다.

파일 또는 디렉토리의 전체 경로를 입력합니다. 확장된 디렉토리에서 배포하는 것은 고급 개발자를 위한 것이며 작업 환경에는 권장되지 않습니다.

5. 다음을 눌러 웹 응용 프로그램 배포 페이지를 표시합니다.

6. 웹 응용 프로그램 배포 페이지에서 응용 프로그램의 설정을 지정합니다.

a. 응용 프로그램 이름 필드에서 파일 이름의 접두어인 기본 이름을 그대로 두거나 다른 이름을 입력합니다. (파일 업로드를 선택한 경우 기본 이름이 표시됩니다.) 응용 프로그램 이름은 고유해야 합니다.

b. 컨텍스트 루트 필드에서 웹 응용 프로그램을 식별하는 문자열을 입력합니다. 웹 응용 프로그램의 URL에서 포트 번호 다음에 바로 컨텍스트 루트가 나옵니다 (`http://host:port/context-root/...`). 컨텍스트 루트는 슬래시로 시작해야 합니다(예: `/hello`).

c. 기본적으로 응용 프로그램은 배포하자마자 사용할 수 있습니다. 배포 후 응용 프로그램을 사용할 수 없게 비활성화하려면 비활성화 라디오 버튼을 선택합니다.

d. 응용 프로그램을 이미 배포한 경우 재배포 확인란을 선택하여 재배포합니다. 그렇지 않으면 오류가 표시됩니다. 다른 응용 프로그램 이름을 선택하여 새로운 이름 아래에 배포할 수도 있습니다.

e. 배포 전에 파일의 구조 및 내용을 검증하려면 검증자 확인란을 선택합니다. 큰 응용 프로그램을 검증할 경우 시간이 많이 소모될 수 있습니다. 파일이 손상되거나 이식 불가능한 것으로 의심될 경우 파일을 검증합니다.

f. JSP 페이지를 사전 컴파일하려면 JSP 확인란을 선택합니다. 확인란을 선택하지 않은 경우, JSP 페이지는 처음 액세스되는 런타임 시 컴파일됩니다. 컴파일은 작업 환경에서 시간이 많이 소요될 수 있으므로 확인란을 선택합니다.

g. 고가용성 설정을 선택합니다.

응용 프로그램에 대한 고가용성을 활성화하려면 가용성 확인란을 선택합니다. 응용 프로그램에 대한 가용성을 활성화한 경우 더 높은 모든 수준에서도(명명된 구성 및 웹 컨테이너 또는 EJB 컨테이너) 활성화해야 합니다.

h. 응용 프로그램을 배포할 대상을 선택합니다.

사용 가능한 대상 목록에서 대상을 선택한 후 추가를 누릅니다. 클러스터나 독립 실행형 인스턴스가 대상이 될 수 있습니다. 대상을 선택하지 않은 경우 응용 프로그램이 기본 서버 인스턴스 server에 배포됩니다.

재배포 중인 경우 대상을 선택하지 마십시오. 여기서 선택한 대상은 무시됩니다. 배포된 응용 프로그램을 참조하는 클러스터링된 대상이나 독립 실행형 서버 인스턴스는 클러스터나 독립 실행형 인스턴스에 대해 동적 재구성을 활성화한 경우 자동으로 재배포된 새로운 응용 프로그램을 참조합니다. 서비스의 중단 없이 응용 프로그램을 재배포하는 방법에 대한 자세한 내용은 88페이지의 "롤링 업그레이드 정보"를 참조하십시오.

i. RMI 스텁을 생성할지 선택합니다.

RMI 스텁을 생성하기로 선택한 경우 정적 RMI-IIOP 스텁이 client.jar에 생성됩니다.

7. 응용 프로그램을 배포하려면 확인을 누릅니다.

해당 asadmin 명령: deploy

배포된 웹 응용 프로그램 시작

응용 프로그램을 배포한 후 관리 콘솔에서 응용 프로그램을 시작할 수 있습니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 웹 응용 프로그램을 누릅니다.
3. 웹 응용 프로그램의 시작 링크를 누릅니다.
4. 웹 응용 프로그램 링크 페이지에서 링크를 눌러 응용 프로그램을 시작합니다.

응용 프로그램을 시작하려면 서버와 HTTP Listener가 실행 중이어야 합니다.

EJB 모듈 배포

EJB JAR 파일이라고도 하는 EJB 모듈에는 Enterprise Bean이 포함되어 있습니다.

EJB 모듈을 배포(설치)하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. EJB 모듈 노드를 선택합니다.

3. EJB 모듈 페이지에서 배포를 누릅니다.
4. 배포 페이지에서 배포할 JAR 파일의 위치를 지정합니다.

   서버 시스템은 Application Server와 Domain Administration Server를 실행하는 호스트입니다. 클라이언트 시스템은 브라우저를 통해 관리 콘솔을 볼 수 있는 호스트입니다.

   a. 파일이 클라이언트 시스템에 상주하거나 클라이언트 시스템에서 액세스하려면, 라디오 버튼을 눌러 Application Server에 업로드할 패키지 파일을 지정합니다.

      찾아보기를 눌러 파일을 찾거나 파일의 전체 경로를 입력합니다.

   b. 파일이 서버 시스템에 상주하거나 확장된 디렉토리에서 압축 해제된 응용 프로그램을 배포하려면 라디오 버튼을 눌러 패키지 파일을 지정하거나 서버에서 액세스할 수 있는 디렉토리 경로를 지정합니다.

      파일 또는 디렉토리의 전체 경로를 입력합니다. 확장된 디렉토리에서 배포하는 것은 고급 개발자를 위한 것이며 작업 환경에는 권장되지 않습니다.
5. 다음을 눌러 EJB 모듈 배포 페이지를 표시합니다.
6. EJB 모듈 배포 페이지에서 모듈에 대한 설정을 지정합니다.

   a. 응용 프로그램 이름 필드에서 파일 이름의 접두어인 기본 이름을 그대로 두거나 다른 이름을 입력합니다. (파일 업로드를 선택한 경우 기본 이름이 표시됩니다.) 응용 프로그램 이름은 고유해야 합니다.

   b. 기본적으로 모듈은 배포하자마자 사용할 수 있습니다. 배포 후 모듈을 사용할 수 없게 비활성화하려면 비활성화 라디오 버튼을 선택합니다.

   c. 모듈을 이미 배포한 경우 재배포 확인란을 선택하여 재배포합니다. 그렇지 않으면 오류가 표시됩니다. 다른 응용 프로그램 이름을 선택하여 새로운 이름 아래에 배포할 수도 있습니다.

   d. 배포 전에 파일의 구조 및 내용을 검증하려면 검증자 확인란을 선택합니다. 큰 응용 프로그램을 검증할 경우 시간이 많이 소모될 수 있습니다. 파일이 손상되거나 이식 불가능한 것으로 의심될 경우 파일을 검증합니다.

   e. 고가용성 설정을 선택합니다.

      모듈에 대한 고가용성을 활성화하려면 가용성 확인란을 선택합니다. 모듈에 대한 가용성을 활성화한 경우 더 높은 모든 수준에서도(명명된 구성 및 웹 컨테이너 또는 EJB 컨테이너) 활성화해야 합니다.

f. 모듈을 배포할 대상을 선택합니다.

   사용 가능한 대상 목록에서 대상을 선택한 후 추가를 누릅니다. 클러스터나 독립 실행형 인스턴스가 대상이 될 수 있습니다. 대상을 선택하지 않은 경우 모듈이 기본 서버 인스턴스 `server`에 배포됩니다.

   재배포 중인 경우 대상을 선택하지 마십시오. 여기서 선택한 대상은 무시됩니다. 배포된 모듈을 참조하는 클러스터링된 대상이나 독립 실행형 서버 인스턴스는 클러스터나 독립 실행형 인스턴스에 대해 동적 재구성을 활성화한 경우 자동으로 새로운 재배포된 모듈을 참조합니다. 서비스의 중단 없이 모듈을 재배포하는 방법에 대한 자세한 내용은 88페이지의 "롤링 업그레이드 정보"를 참조하십시오.

g. RMI 스텁을 생성할지 선택합니다.

   RMI 스텁을 생성하기로 선택한 경우 정적 RMI-IIOP 스텁이 `client.jar`에 생성됩니다.

7. 모듈을 배포하려면 확인을 누릅니다.

해당 `asadmin` 명령: `deploy`

커넥터 모듈 배포

자원 어댑터라고도 하는 커넥터는 RAR 파일이라고 하는 아카이브 파일로 패키지화됩니다.

커넥터 모듈을 배포(설치)하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 커넥터 모듈 노드를 선택합니다.
3. 커넥터 모듈 페이지에서 배포를 누릅니다.
4. 배포 페이지에서 배포할 RAR 파일의 위치를 지정합니다.

   서버 시스템은 Application Server와 Domain Administration Server를 실행하는 호스트입니다. 클라이언트 시스템은 브라우저를 통해 관리 콘솔을 볼 수 있는 호스트입니다.

   a. 파일이 클라이언트 시스템에 상주하거나 클라이언트 시스템에서 액세스하려면, 라디오 버튼을 눌러 Application Server에 업로드할 패키지 파일을 지정합니다.

      찾아보기를 눌러 파일을 찾거나 파일의 전체 경로를 입력합니다.

b. 파일이 서버 시스템에 상주하거나 확장된 디렉토리에서 압축 해제된 모듈을 배포하려면 라디오 버튼을 눌러 패키지 파일을 지정하거나 서버에서 액세스할 수 있는 디렉토리 경로를 지정합니다.

파일 또는 디렉토리의 전체 경로를 입력합니다. 확장된 디렉토리에서 배포하는 것은 고급 개발자를 위한 것이며 작업 환경에는 권장되지 않습니다.

5. 다음을 눌러 커넥터 모듈 배포 페이지를 표시합니다.

6. 커넥터 모듈 배포 페이지에서 모듈에 대한 설정을 지정합니다.

a. 응용 프로그램 이름 필드에서 파일 이름의 접두어인 기본 이름을 그대로 두거나 다른 이름을 입력합니다. (파일 업로드를 선택한 경우 기본 이름이 표시됩니다.) 응용 프로그램 이름은 고유해야 합니다.

b. 스레드 풀 아이디 필드에서 배포할 자원 어댑터에 대한 스레드 풀을 지정합니다.

기본적으로 Sun Java System Application Server 서비스는 모든 자원 어댑터의 요청을 기본 스레드 풀에서 작업합니다. 특정한 사용자가 만든 스레드 풀을 자원 어댑터의 서비스 작업 요청에 연관시키려면 이 필드를 사용합니다.

c. 기본적으로 모듈은 배포하자마자 사용할 수 있습니다. 배포 후 모듈을 사용할 수 없게 비활성화하려면 비활성화 라디오 버튼을 선택합니다.

커넥터 모듈을 활성화하거나 비활성화할 경우 모듈을 가리키는 커넥터 자원과 연결 풀도 활성화하거나 비활성화합니다.

d. 모듈을 이미 배포한 경우 재배포 확인란을 선택하여 재배포합니다. 그렇지 않으면 오류가 표시됩니다. 다른 응용 프로그램 이름을 선택하여 새로운 이름 아래에 배포할 수도 있습니다.

e. 배포 전에 파일의 구조 및 내용을 검증하려면 검증자 확인란을 선택합니다. 큰 응용 프로그램을 검증할 경우 시간이 많이 소모될 수 있습니다. 파일이 손상되거나 이식 불가능한 것으로 의심될 경우 파일을 검증합니다.

f. 자원 어댑터에 추가 등록 정보가 지정된 경우 등록 정보가 표시됩니다.

표를 사용하여 이 등록 정보의 기본값을 수정합니다.

g. 모듈을 배포할 대상을 선택합니다.

사용 가능한 대상 목록에서 대상을 선택한 후 추가를 누릅니다. 클러스터나 독립 실행형 인스턴스가 대상이 될 수 있습니다. 대상을 선택하지 않은 경우 모듈이 기본 서버 인스턴스 server에 배포됩니다.

재배포 중인 경우 대상을 선택하지 마십시오. 여기서 선택한 대상은 무시됩니다. 배포된 모듈을 참조하는 클러스터링된 대상이나 독립 실행형 서버 인스턴스는 클러스터나 독립 실행형 인스턴스에 대해 동적 재구성을 활성화한 경우 자동으로 재배포된 새로운 모듈을 참조합니다. 서비스의 중단 없이 모듈을 재배포하는 방법에 대한 자세한 내용은 88페이지의 "롤링 업그레이드 정보"를 참조하십시오.

7. 확인을 눌러 모듈을 배포합니다.

해당 asadmin 명령: deploy

라이프사이클 모듈 만들기

서버 라이프사이클의 하나 이상의 이벤트에서 트리거할 경우 라이프사이클 모듈에서 작업을 수행합니다. 이 서버 이벤트는 다음과 같습니다.

- 초기화
- 시작
- 요청 서비스 준비
- 종료

라이프사이클 모듈은 J2EE 사양의 일부가 아니지만 Sun Java System Application Server에 대한 개선 사항입니다.

라이프사이클 모듈을 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.

2. 라이프사이클 모듈 노드를 선택합니다.

3. 라이프사이클 모듈 페이지에서 새로 만들기를 누릅니다.

4. 라이프사이클 모듈 만들기 페이지에서 설정을 지정합니다.

a. 이름 필드에서 모듈의 기능을 나타내는 이름을 입력합니다.

b. 클래스 이름 필드에서 라이프사이클 모듈의 클래스 파일에 대한 정규화된 이름을 입력합니다.

c. 라이프사이클을 포함하는 JAR 파일이 서버의 클래스 경로에 있을 경우 클래스 경로 필드를 공백으로 남겨둡니다. 그렇지 않으면 정규화된 경로를 입력합니다.

클래스 경로를 지정하지 않은 경우 *application_server_home*/domains/*domain*/applications/lifecycle-module/*module_name*의 클래스를 압축 해제해야 합니다. 클래스 경로를 지정하는 것 외에 아무것도 필요하지 않습니다.

d. 로드 순서 필드에서 100보다 크고 운영 체제의 MAXINT 값보다 작은 정수를 입력합니다.

이 정수는 서버를 시작할 때 라이프사이클 모듈을 로드하는 순서를 결정합니다. 정수 값이 작은 모듈이 먼저 로드됩니다.

e. 서버를 시작할 경우 이미 배포된 라이프사이클 모듈을 로드합니다. 기본적으로 로드가 실패할 경우 서버에서 시작 작업을 계속합니다. 로드가 실패한 경우 서버가 시작되는 것을 방지하려면 로드 실패 시 확인란을 선택합니다.

f. 기본적으로 모듈은 배포하자마자 사용할 수 있습니다. 배포 후 모듈을 사용할 수 없게 비활성화하려면 비활성화 라디오 버튼을 선택합니다.

g. 모듈을 배포할 대상을 선택합니다.

사용 가능한 대상 목록에서 대상을 선택한 후 추가를 누릅니다. 클러스터나 독립 실행형 인스턴스가 대상이 될 수 있습니다. 대상을 선택하지 않은 경우 모듈이 기본 서버 인스턴스 server에 배포됩니다.

재배포 중인 경우 대상을 선택하지 마십시오. 여기서 선택한 대상은 무시됩니다. 배포된 모듈을 참조하는 클러스터링된 대상이나 독립 실행형 서버 인스턴스는 클러스터나 독립 실행형 인스턴스에 대해 동적 재구성을 활성화한 경우 자동으로 새로운 재배포된 모듈을 참조합니다. 서비스의 중단 없이 모듈을 재배포하는 방법에 대한 자세한 내용은 88페이지의 "롤링 업그레이드 정보"를 참조하십시오.

5. 확인을 누릅니다.

해당 asadmin 명령: create-lifecycle-module

응용 프로그램 클라이언트 모듈 배포

J2EE 응용 프로그램 클라이언트 JAR 파일이라고도 하는 응용 프로그램 클라이언트 모듈에는 클라이언트의 서버측 루틴이 포함됩니다.

응용 프로그램 클라이언트 모듈을 배포(설치)하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 응용 프로그램 클라이언트 모듈 노드를 선택합니다.
3. 응용 프로그램 클라이언트 모듈 페이지에서 배포를 누릅니다.
4. 배포 페이지에서 배포할 JAR 파일의 위치를 지정합니다.

   서버 시스템은 Application Server와 Domain Administration Server를 실행하는 호스트입니다. 클라이언트 시스템은 브라우저를 통해 관리 콘솔을 볼 수 있는 호스트입니다.

   a. 파일이 클라이언트 시스템에 상주하거나 클라이언트 시스템에서 액세스하려면, 라디오 버튼을 눌러 Application Server에 업로드할 패키지 파일을 지정합니다.

      찾아보기를 눌러 파일을 찾거나 파일의 전체 경로를 입력합니다.

   b. 파일이 서버 시스템에 상주하거나 확장된 디렉토리에서 압축 해제된 모듈을 배포하려면 라디오 버튼을 눌러 패키지 파일을 지정하거나 서버에서 액세스할 수 있는 디렉토리 경로를 지정합니다.

      파일 또는 디렉토리의 전체 경로를 입력합니다. 확장된 디렉토리에서 배포하는 것은 고급 개발자를 위한 것이며 작업 환경에는 권장되지 않습니다.

5. 다음을 눌러 응용 프로그램 클라이언트 모듈 배포 페이지를 표시합니다.
6. 응용 프로그램 클라이언트 모듈 배포 페이지에서 모듈에 대한 설정을 지정합니다.

   a. 응용 프로그램 이름 필드에서 파일 이름의 접두어인 기본 이름을 그대로 두거나 다른 이름을 입력합니다. (파일 업로드를 선택한 경우 기본 이름이 표시됩니다.) 응용 프로그램 이름은 고유해야 합니다.

   b. 모듈을 이미 배포한 경우 재배포 확인란을 선택하여 재배포합니다. 그렇지 않으면 오류가 표시됩니다. 다른 응용 프로그램 이름을 선택하여 새로운 이름 아래에 배포할 수도 있습니다.

   c. 배포 전에 파일의 구조 및 내용을 검증하려면 검증자 확인란을 선택합니다. 큰 응용 프로그램을 검증할 경우 시간이 많이 소모될 수 있습니다. 파일이 손상되거나 이식 불가능한 것으로 의심될 경우 파일을 검증합니다.

d. 모듈을 배포할 대상을 선택합니다.

사용 가능한 대상 목록에서 대상을 선택한 후 추가를 누릅니다. 클러스터나 독립 실행형 인스턴스가 대상이 될 수 있습니다. 대상을 선택하지 않은 경우 모듈이 기본 서버 인스턴스 `server`에 배포됩니다.

재배포 중인 경우 대상을 선택하지 마십시오. 여기서 선택한 대상은 무시됩니다. 배포된 모듈을 참조하는 클러스터링된 대상이나 독립 실행형 서버 인스턴스는 클러스터나 독립 실행형 인스턴스에 대해 동적 재구성을 활성화한 경우 자동으로 새로운 재배포된 모듈을 참조합니다. 서비스의 중단 없이 모듈을 재배포하는 방법에 대한 자세한 내용은 88페이지의 "롤링 업그레이드 정보"를 참조하십시오.

e. RMI 스텁을 생성할지 선택합니다.

RMI 스텁을 생성하기로 선택한 경우 정적 RMI-IIOP 스텁이 `client.jar`에 생성됩니다.

7. 확인을 눌러 모듈을 배포합니다.

클라이언트측 루틴의 경우:

- 대개 응용 프로그램 공급자는 클라이언트측 루틴을 포함하는 JAR 파일을 제공합니다.
- 응용 프로그램 공급자는 `asadmin deploy` 명령의 `--retrieve` 옵션을 지정하여 클라이언트측 스텁을 가져옵니다.

해당 `asadmin` 명령: `deploy`

응용 프로그램을 나열, 배포 해제 및 활성화하기 위한 관리 콘솔 작업

배포된 응용 프로그램 나열

배포된 응용 프로그램을 나열하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 응용 프로그램이나 모듈 유형에 대한 노드를 확장합니다.

배포된 응용 프로그램이나 모듈의 세부 정보를 보려면 다음 작업을 수행합니다.

- 트리 구성 요소에서 응용 프로그램이나 모듈의 노드를 선택합니다.
- 해당 페이지의 응응 프로그램 이름 열에서 원하는 항목을 선택합니다.

해당 asadmin 명령: `list-components`

하위 구성 요소 나열

엔터프라이즈 및 웹 응용 프로그램, EJB 모듈 및 커넥터 모듈에는 하위 구성 요소가 포함되어 있습니다. 예를 들어, 웹 응용 프로그램에는 하나 이상의 서블릿이 포함될 수 있습니다.

응용 프로그램이나 모듈의 하위 구성 요소를 나열하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 설명자를 표시할 응용 프로그램이나 모듈의 노드를 확장합니다.
3. 배포된 응용 프로그램이나 모듈의 노드를 선택합니다.
4. 응용 프로그램이나 모듈 페이지에서 하위 구성 요소 테이블의 내용을 확인합니다.

해당 asadmin 명령: `list-sub-components`

배포된 응용 프로그램의 모듈 설명자 보기

엔터프라이즈 응용 프로그램, 웹 응용 프로그램, EBJ 모듈, 커넥터 모듈 및 응용 프로그램 클라이언트 모듈의 경우 모듈 배포 설명자를 볼 수 있습니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 설명자를 조회할 응용 프로그램이나 모듈의 노드를 선택합니다.

3. 배포된 응용 프로그램이나 모듈의 노드를 선택합니다.
4. 설명자 탭을 선택합니다.
5. 설명자 파일의 텍스트를 조회하려면 파일 이름을 누릅니다.

   페이지에 파일 내용이 표시됩니다. 이 정보는 읽기 전용입니다.

응용 프로그램 배포 해제

응용 프로그램이나 모듈을 배포 해제하면 도메인에서 이들이 제거되고 모든 인스턴스에서 이들에 대한 참조가 제거됩니다.

응용 프로그램이나 모듈을 배포 해제하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 배포 해제할 응용 프로그램이나 모듈의 노드를 선택합니다.
3. 배포된 응용 프로그램을 나열하는 표에서 배포 해제할 응용 프로그램이나 모듈의 확인란을 선택합니다.
4. 배포 해제를 누릅니다.

해당 `asadmin` 명령: `undeploy`

응용 프로그램 활성화 및 비활성화

배포된 응용 프로그램이나 모듈이 활성화되어 있는 경우 클라이언트에서 액세스할 수 있습니다. 비활성화한 경우 여전히 배포되어 있지만 클라이언트에서 액세스할 수 없습니다. 응용 프로그램이나 모듈을 배포한 경우 모든 대상에서 활성화 라디오 버튼이 기본적으로 선택되기 때문에 응용 프로그램이나 모듈이 사용 가능합니다.

배포된 응용 프로그램이나 모듈을 활성화하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 응용 프로그램 유형에 대한 노드를 확장합니다.
3. 배포된 응용 프로그램이나 모듈 옆에 있는 확인란을 선택합니다.
4. 활성화 또는 비활성화를 누릅니다.

   이 버튼은 모든 대상에서 응용 프로그램을 활성화하거나 비활성화합니다.

단일 대상에서 응용 프로그램을 활성화하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.

2. 응용 프로그램 유형에 대한 노드를 확장합니다.
3. 응용 프로그램의 노드를 선택합니다.
4. 대상 탭을 누릅니다.
5. 배포된 응용 프로그램이나 모듈 옆에 있는 확인란을 선택합니다.
6. 활성화 또는 비활성화를 누릅니다.

해당 asadmin 명령: `enable` 및 `disable`

응용 프로그램 대상 관리

응용 프로그램이나 모듈을 배포한 후 대상을 관리하여 응용 프로그램이나 모듈을 참조하는 서버 인스턴스나 클러스터를 관리합니다.

1. 트리 구성 요소에서 응용 프로그램 노드를 확장합니다.
2. 응용 프로그램 유형에 대한 노드를 확장합니다.
3. 배포된 응용 프로그램의 노드를 선택합니다.
4. 대상 탭을 선택합니다.
5. 특정한 대상 인스턴스나 클러스터의 응용 프로그램을 활성화하거나 비활성화하려면 대상 옆에 있는 확인란을 누르고 활성화 또는 비활성화를 누릅니다.
6. 응용 프로그램의 대상을 추가하거나 삭제하려면 대상 관리를 선택합니다.
7. 대상을 추가하거나 제거하고 확인을 누릅니다.

   이제 수정된 대상 목록에서 응용 프로그램을 사용할 수 있습니다.

해당 asadmin 명령: `create-application-ref`, `delete-application-ref`

추가 가상 서버 배포

응용 프로그램이나 모듈을 대상 서버 인스턴스나 클러스터에 배포한 후 이를 추가 가상 서버와 연관시킬 수 있습니다.

1. 배포된 응용 프로그램이나 모듈의 대상 페이지에서 대상 옆에 있는 가상 서버 관리 링크를 누릅니다.
2. 사용 가능한 가상 서버 목록에서 가상 서버 대상을 추가하거나 제거합니다.
3. 확인을 누릅니다.

복수 대상에 재배포

응용 프로그램을 복수 대상(독립 실행형 서버 인스턴스나 클러스터)에 배포할 경우 복수 대상에 재배포할 수 있는 두 가지 방법이 있습니다. 다음 방법 중 하나를 사용하여 응용 프로그램을 참조하는 모든 서버 인스턴스가 최신 버전을 수신할 수 있도록 합니다.

개발 환경

개발 환경에서 단순히 응용 프로그램을 재배포합니다. 응용 프로그램이 도메인에 재배포되고, 응용 프로그램을 참조하는 모든 대상은 대상 서버 인스턴스에 대한 동적 재구성이 활성화된 경우 자동으로 최신 버전을 수신합니다. 기본적으로 동적 재구성이 활성화되어 있습니다. 서버 인스턴스에 대한 동적 재구성이 활성화되지 않은 경우 서버 인스턴스를 다시 시작할 때까지 계속 이전 버전을 사용합니다.

작업 환경

작업 환경에서 88페이지의 "롤링 업그레이드 정보"에 설명된 단계를 수행합니다.

동적 재로드 활성화 및 비활성화

동적 재로드가 활성화된 경우 서버는 배포된 응용 프로그램의 변경 사항을 정기적으로 확인하고 변경 사항과 함께 응용 프로그램을 자동으로 다시 로드합니다. 수동으로 만든 `.reload`라고 하는 파일의 날짜가 변경되므로 변경되었음을 알 수 있습니다. 응용 프로그램은 *server_root*`/domain/domain1/applications/`*j2ee-module_or_j2ee-apps*`/`*app_or_module_name*에 설치해야 합니다.

예를 들면 다음과 같습니다.
`AppServer/domain/domain1/applications/j2ee-module/webapps-simple`

코드 변경을 빠르게 테스트할 수 있기 때문에 동적 재로드는 개발 환경에서 유용합니다. 그러나 작업 환경에서는 동적 재로드가 성능을 저하시킬 수 있습니다.

주 동적 재로드는 기본 서버 인스턴스에 대해서만 사용할 수 있습니다.

동적 재로드는 개발 환경을 위한 것입니다. 세션 지속성 기능인 작업 환경 기능과 호환되지 않습니다. 동적 배포가 활성화된 경우 세션 지속성을 활성화하지 마십시오.

동적 재로드를 구성하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 독립 실행형 인스턴스 노드를 확장합니다.
2. 서버(관리 서버)를 누릅니다.
3. 고급을 누릅니다.
4. 응용 프로그램 구성 페이지에서 다음을 구성합니다.
   - 재로드: 사용 확인란을 사용하여 동적 재로드를 활성화하거나 비활성화합니다.
   - 다시 로드 폴링 간격: 서버가 배포된 응용 프로그램의 변경 사항을 확인하는 빈도를 지정합니다.
   - 관리 세션 시간 초과: 얼마 후에 관리 세션이 시간 초과되어 다시 로그인해야 하는지 시간을 지정합니다.

시스템에서 동적 재로드를 사용하도록 구성한 후 모든 응용 프로그램을 동적으로 다시 로드하기 위해 .reload라고 하는 파일을 만들어 응용 프로그램의 디렉토리에 보관합니다. 파일에는 내용이 없습니다. 응용 프로그램을 변경하고 파일 날짜를 변경하면(예: UNIX의 경우 `touch` 명령 사용), 변경 사항이 자동으로 다시 로드됩니다.

개발자를 위한 개발 방법

자동 배포 사용

자동 배포 기능을 사용하면 사전 패키지화된 응용 프로그램이나 모듈을 *domain_root_dir*/*domain_dir*/`autodeploy` 디렉토리에 복사하여 배포할 수 있습니다.

예를 들어, `hello.war`이라고 하는 파일을 *domain_root_dir*/`domain1/autodeploy` 디렉토리에 복사합니다. 응용 프로그램을 배포 해제하려면 `autodeploy` 디렉토리에서 `hello.war` 파일을 제거합니다.

관리 콘솔이나 asadmin 도구를 사용하여 응용 프로그램을 배포 해제할 수도 있습니다. 이러한 경우 아카이브 파일이 그대로 유지됩니다.

자동 배포 기능은 개발 환경을 위한 것입니다. 세션 지속성 기능인 작업 환경 기능과 호환되지 않습니다. 동적 배포가 활성화된 경우 세션 지속성을 활성화하지 마십시오.

주 자동 배포는 기본 서버 인스턴스에 대해서만 사용할 수 있습니다.

자동 배포 기능을 구성하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 독립 실행형 인스턴스 노드를 확장합니다.
2. 서버(관리 서버)를 누릅니다.
3. 고급을 누릅니다.
4. 응용 프로그램 구성 페이지에서 다음을 구성합니다.
   a. 사용 확인란을 선택하거나 선택 해제하여 자동 배포를 활성화하거나 비활성화합니다.
   b. 자동 배포 폴링 간격 필드에서 서버가 응용 프로그램이나 모듈 파일의 자동 배포 디렉토리를 확인하는 빈도를 지정합니다. 폴링 간격을 변경해도 응용 프로그램이나 모듈을 배포하는 데 걸리는 시간에 영향을 미치지 않습니다.
   c. 자동 배포 디렉토리에서 응용 프로그램을 빌드할 디렉토리를 지정한 경우 파일을 기본 자동 배포 디렉토리에 복사할 필요가 없습니다.

      기본값은 서버 인스턴스의 루트 디렉토리에 있는 `autodeploy`라고 하는 디렉토리입니다. 기본적으로 변수를 사용하여 복수 서버 인스턴스에 대한 디렉토리를 수동으로 변경할 필요성을 제거합니다. 이 변수에 대한 자세한 내용은 도메인 속성 설정을 참조하십시오.
   d. 배포 전에 검증자를 실행하려면 검증자 사용 확인란을 선택합니다. 검증자는 파일의 구조와 내용을 검사합니다. 큰 응용 프로그램을 검증할 경우 시간이 많이 소모될 수 있습니다.
   e. JSP 페이지를 사전 컴파일하려면 JSP 확인란을 선택합니다. 확인란을 선택하지 않은 경우, JSP 페이지는 처음 액세스되는 런타임 시 컴파일됩니다. 컴파일은 작업 환경에서 시간이 많이 소요될 수 있으므로 확인란을 선택합니다.

디렉토리에서 압축 해제된 응용 프로그램 배포

이 기능은 고급 개발자를 위한 것입니다.

기본 서버 인스턴스(서버)에 배포할 때만 디렉토리 배포를 사용합니다. 클러스터나 독립 실행형 서버 인스턴스에 배포할 때는 사용할 수 없습니다.

압축 해제된 응용 프로그램이나 모듈을 포함하는 디렉토리는 확장된 디렉토리라고 합니다. 디렉토리의 내용은 해당하는 J2EE 아카이브 파일의 내용과 일치해야 합니다. 예를 들어, 디렉토리에서 웹 응용 프로그램을 배포할 경우 디렉토리의 내용은 해당하는 WAR 파일과 동일해야 합니다. 필요한 디렉토리 내용에 대한 정보는 해당 사양을 참조하십시오.

확장된 디렉토리에서 직접 배포 설명자 파일을 변경할 수 있습니다.

동적 재로드를 사용하도록 환경을 구성한 경우 디렉토리에서 배포된 응용 프로그램을 동적으로 다시 로드할 수도 있습니다. 자세한 내용은 128페이지의 "동적 재로드 활성화 및 비활성화"를 참조하십시오.

디렉토리에서 압축 해제된 응용 프로그램을 배포하려면 다음 작업을 수행합니다.

1. 관리 콘솔에서 배포 프로세스를 시작합니다. 115페이지의 "웹 응용 프로그램 배포"를 참조하십시오.
2. 배포 페이지에서 다음을 지정합니다.
   a. 라디오 버튼을 사용하여 서버에서 액세스할 수 있는 패키지 파일이나 디렉토리 경로를 지정합니다.
   b. 파일 또는 디렉토리 필드에서 확장된 디렉토리의 이름을 입력합니다.

해당 `asadmin` 명령: `deploydir`

deploytool 유틸리티 사용

소프트웨어 개발자를 위해 설계된 `deploytool` 유틸리티는 J2EE 응용 프로그램과 모듈을 패키지화하고 배포합니다. `deploytool` 사용 방법에 대한 지침은 *J2EE 1.4 Tutorial*을 참조하십시오.

배포 계획 사용

이 기능은 고급 개발자를 위한 것입니다.

배포 계획은 Application Server에 관련된 배포 설명자만 포함하는 JAR 파일입니다. 이 배포 설명자(예: sun-application.xml)는 *Application Server Developer's Guide*에서 설명합니다. 배포 계획은 *JSR 88: J2EE 응용 프로그램 배포* 구현의 일부입니다. Application Server에 관련된 배포 설명자를 포함하지 않는 응용 프로그램이나 모듈을 배포하려면 배포 계획을 사용합니다.

배포 계획을 사용하여 배포하려면 asadmin deploy 명령의 --deploymentplan 옵션을 지정합니다. 예를 들어, 다음 명령은 mydeployplan.jar 파일에서 지정한 계획에 따라 myrosterapp.ear 파일에 엔터프라이즈 응용 프로그램을 배포합니다.

```
$ asadmin deploy --user admin ---deploymentplan mydeployplan.jar
myrosterapp.ear
```

엔터프라이즈 응용 프로그램(EAR)의 배포 계획 파일에서 sun-application.xml 파일은 루트에 있습니다. 모든 모듈의 배포 설명자는 *module-name.sun-dd-name* 구문에 따라 저장됩니다. 여기서 *sun-dd-name*은 모듈 유형에 따라 다릅니다. 모듈에 CMP 매핑 파일이 포함된 경우 파일 이름은 *module-name*.sun-cmp-mappings.xml이 됩니다. .dbschema 파일은 슬래시(/) 문자가 파운드 기호(#)로 대체되어 루트 수준에 저장됩니다. 다음 목록에서는 엔터프라이즈 응용 프로그램(EAR)에 대한 배포 계획 파일의 구조를 보여줍니다.

```
$ jar -tvf mydeployplan.jar
420 Thu Mar 13 15:37:48 PST 2003 sun-application.xml
370 Thu Mar 13 15:37:48 PST 2003 RosterClient.war.sun-web.xml
418 Thu Mar 13 15:37:48 PST 2003 roster-ac.jar.sun-application-client.xml
1281 Thu Mar 13 15:37:48 PST 2003 roster-ejb.jar.sun-ejb-jar.xml
2317 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-ejb-jar.xml
3432 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.sun-cmp-mappings.xml
84805 Thu Mar 13 15:37:48 PST 2003 team-ejb.jar.RosterSchema.dbschema
```

웹 응용 프로그램이나 모듈 파일의 배포 계획에서 Application Server에 관련된 배포 설명자는 루트 수준에 있습니다. 독립 실행형 EJB 모듈에 CMP Bean이 포함된 경우 배포 계획의 루트 수준에 sun-cmp-mappings.xml 및 .dbschema 파일이 포함됩니다. 다음 목록에서 배포 계획은 CMP Bean을 설명합니다.

```
$ jar r -tvf myotherplan.jar
3603 Thu Mar 13 15:24:20 PST 2003 sun-ejb-jar.xml
3432 Thu Mar 13 15:24:20 PST 2003 sun-cmp-mappings.xml
84805 Thu Mar 13 15:24:20 PST 2003 RosterSchema.dbschema
```

JDBC 자원

이 장에서는 데이터베이스에 액세스하는 응용 프로그램에서 필요한 JDBC 자원을 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

JDBC 자원 정보

JDBC 자원

데이터를 저장, 구성 및 검색하기 위해 대부분 응용 프로그램에서는 관계형 데이터베이스를 사용합니다. J2EE 응용 프로그램은 JDBC API를 통해 관계형 데이터베이스에 액세스합니다.

JDBC 자원(데이터 소스)은 데이터베이스에 연결하는 수단을 응용 프로그램에 제공합니다. 일반적으로 관리자는 도메인에 배포된 응용 프로그램에서 액세스한 데이터베이스마다 JDBC 자원을 만듭니다. 그러나 한 데이터베이스에 대해 JDBC 자원을 여러 개 만들 수 있습니다.

JDBC 자원을 만들려면 자원을 식별하는 고유한 JNDI 이름을 지정합니다(JNDI 이름 및 자원 절 참조). `java:comp/env/jdbc` 하위 컨텍스트에서 JDBC 자원의 JNDI 이름을 찾을 수 있습니다. 예를 들어, 급여 데이터베이스의 자원에 대한 JNDI 이름은 `java:comp/env/jdbc/payrolldb`일 수 있습니다. 모든 자원 JNDI 이름이 `java:comp/env` 하위 컨텍스트로 되어 있기 때문에 관리 콘솔에서 JDBC 자원의 JNDI 이름을 지정할 때 `jdbc/`*name*만 입력합니다. 예를 들어, 급여 데이터베이스의 경우 `jdbc/payrolldb`를 지정합니다

JDBC 연결 풀

JDBC 자원을 만들려면 연결된 연결 풀을 지정합니다. 여러 JDBC 자원이 단일 연결 풀을 지정할 수 있습니다.

JDBC 연결 풀은 특정 데이터베이스에 다시 사용할 수 있는 연결 그룹입니다. 새로운 물리적 연결을 만드는 데 시간이 많이 소모되기 때문에 서버에서는 사용 가능한 연결 풀을 유지 관리하여 성능을 증가시킵니다. 응용 프로그램에서 연결을 요청하면 풀에서 하나 받습니다. 응용 프로그램에서 연결을 닫으면 연결이 풀로 반환됩니다.

연결 풀의 등록 정보는 데이터베이스 공급업체에 따라 다를 수 있습니다. 일부 공통된 등록 정보는 데이터베이스의 이름(URL), 아이디 및 비밀번호입니다.

JDBC 자원 및 연결 풀을 함께 작업하는 방법

데이터를 저장, 구성 및 검색하기 위해 대부분 응용 프로그램에서는 관계형 데이터베이스를 사용합니다. J2EE 응용 프로그램은 JDBC API를 통해 관계형 데이터베이스에 액세스합니다. 응용 프로그램이 데이터베이스에 액세스할 수 있으려면 연결을 가져와야 합니다.

런타임 시 응용 프로그램이 데이터베이스에 연결할 경우 다음과 같은 일이 발생합니다.

1. 응용 프로그램이 JNDI API를 통해 호출하여 데이터베이스에 연관된 JDBC 자원(데이터 소스)을 가져옵니다.

   자원의 JNDI 이름으로 이름 지정 및 디렉토리 서비스에서 JDBC 자원을 찾습니다. JDBC 자원에서 연결 풀을 지정합니다.

2. JDBC 자원을 통해 응용 프로그램은 데이터베이스 연결을 가져옵니다.

   반면 Application Server는 데이터베이스에 해당하는 연결 풀에서 물리적인 연결을 검색합니다. 풀은 데이터베이스 이름(URL), 아이디 및 비밀번호 같은 연결 속성을 정의합니다.

3. 이제 데이터베이스에 연결되고, 응용 프로그램이 데이터베이스의 데이터를 읽고, 수정하고, 추가할 수 있습니다.

   응용 프로그램이 JDBC API를 호출하여 데이터베이스에 액세스합니다. JDBC 드라이버가 응용 프로그램의 JDBC 호출을 데이터베이스 서버의 프로토콜로 변환합니다.

4. 데이터베이스 액세스를 완료하면 응용 프로그램에서 연결을 닫습니다.

   Application Server가 연결을 연결 풀로 반환합니다. 연결이 풀로 반환되면 다음 응용 프로그램에 연결을 사용할 수 있습니다.

데이터베이스 액세스 설정

데이터베이스 액세스 설정을 위한 일반 단계

1. 지원되는 데이터베이스 제품을 설치합니다. Application Server에서 지원하는 데이터베이스 제품 목록은 추가 정보 절의 *릴리스 노트* 링크를 참조하십시오.
2. 데이터베이스 제품용 JDBC 드라이버를 설치합니다.
3. 드라이버의 JAR 파일이 도메인의 서버 인스턴스에 액세스할 수 있게 합니다. JDBC 드라이버 통합을 참조하십시오.
4. 데이터베이스를 만듭니다. 대개 응용 프로그램 공급자가 데이터베이스를 만들고 채우는 스크립트를 제공합니다.
5. 데이터베이스의 연결 풀을 만듭니다. JDBC 연결 풀 만들기를 참조하십시오.
6. 연결 풀을 가리키는 JDBC 자원을 만듭니다. JDBC 자원 만들기를 참조하십시오.

JDBC 드라이버 통합

JDBC 드라이버는 응용 프로그램의 JDBC 호출을 데이터베이스 서버의 프로토콜로 변환합니다. JDBC 드라이버를 관리 도메인에 통합하려면 다음 중 하나를 수행합니다.

- 드라이버가 공용 클래스 로더에 액세스할 수 있게 합니다.
  - 드라이버의 JAR 및 ZIP 파일을 *install_dir*`/domains/`*domain_dir*`/lib` 디렉토리에 복사하거나 드라이버의 클래스 파일을 *install_dir*`/domains/`*domain_dir*`/lib/ext` 디렉토리에 복사합니다.
  - 도메인을 다시 시작합니다.
- 드라이버가 시스템 클래스 로더에 액세스할 수 있게 합니다.
  - 관리 콘솔의 트리 보기에서 구성을 선택합니다.
  - 원하는 구성(예: default-config)을 선택합니다.
  - JVM 설정을 선택합니다.
  - JVM 설정 페이지에서 경로 설정 탭을 누릅니다.
  - 클래스 경로 접미어 필드에서 드라이버의 JAR 파일에 대한 정규화된 경로 이름을 입력합니다.
  - 저장을 누릅니다.
  - 서버를 다시 시작합니다.

JDBC 연결 풀 정보

JDBC 연결 풀 만들기

JDBC 연결 풀은 특정 데이터베이스에 다시 사용할 수 있는 연결 그룹입니다. 관리 콘솔에서 풀을 만들 때 관리자가 특정 데이터베이스에 대한 연결 부분을 실제로 정의합니다.

풀을 만들기 전에 먼저 JDBC 드라이버를 설치 및 통합해야 합니다.

연결 풀 만들기 페이지를 구축할 때 JDBC 드라이버와 관련된 특정 데이터와 데이터베이스 공급업체를 입력해야 합니다. 계속하기 전에 다음 정보를 수집합니다.

- 데이터베이스 공급업체 이름
- 자원 유형: `javax.sql.DataSource`(로컬 트랜잭션에만 해당)
  `javax.sql.XADataSource`(전역 트랜잭션) 등
- 데이터 소스 클래스 이름
- 필수 등록 정보: 데이터베이스 이름(URL), 아이디 및 비밀번호 등

JDBC 연결 풀을 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장합니다.
2. 자원에서 JDBC 노드를 확장합니다.
3. JDBC에서 연결 풀 노드를 선택합니다.
4. 연결 풀 페이지에서 새로 만들기를 누릅니다.
5. 연결 풀 만들기 첫 번째 페이지에서 다음과 같은 일반 설정을 지정합니다.
   a. 이름 필드에서 풀의 논리 이름을 입력합니다.

      JDBC 자원을 만들 때 이 이름을 지정합니다.

   b. 자원 유형 콤보 상자에서 항목을 선택합니다.

   c. 데이터베이스 공급업체 콤보 상자에서 항목을 선택합니다.
6. 다음을 누릅니다.
7. 연결 풀 만들기 두 번째 페이지에서 데이터 소스 클래스 이름 필드에 대한 값을 지정합니다.

   JDBC 드라이버에 이전 페이지에서 자원 유형과 데이터베이스 공급업체에 대한 데이터 소스 클래스를 지정했을 경우 데이터 소스 클래스 이름 필드의 값이 제공됩니다.
8. 다음을 누릅니다.
9. 연결 풀 만들기 세 번째이자 마지막 페이지에서 다음 작업을 수행합니다.
   a. 일반 설정 섹션에서 값이 올바른지 확인합니다.

b. 풀 설정, 연결 검증 및 트랜잭션 격리 섹션에 있는 필드의 경우 기본값을 유지합니다.

나중에 이 설정을 변경하는 것이 가장 편리합니다. JDBC 연결 풀 편집을 참조하십시오.

c. 추가 등록 정보 표에서 데이터베이스 이름(URL), 아이디 및 비밀번호 같은 필수 등록 정보를 추가합니다.

10. 마침을 누릅니다.

해당 `asadmin` 명령: `create-jdbc-connection-pool`

JDBC 연결 풀 편집

JDBC 연결 풀 편집 페이지에서 이름을 제외한 기존 풀의 모든 설정을 변경할 수 있는 방법을 제공합니다.

JDBC 연결 풀 편집 페이지에 액세스하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장합니다.

2. 자원에서 JDBC 노드를 확장합니다.

3. JDBC에서 연결 풀 노드를 확장합니다.

4. 편집할 풀의 노드를 선택합니다.

5. JDBC 연결 풀 편집 페이지에서 필요한 변경을 합니다.

변경할 수 있는 설정에 대한 설명은 다음 절을 참조하십시오.

6. 저장을 누릅니다.

일반 설정

일반 설정의 값은 설치된 특정 JDBC 드라이버에 따라 다릅니다. 이 설정은 Java 프로그래밍 언어로 된 인터페이스나 클래스의 이름입니다.

표 6-1 JDBC 연결 풀의 일반 설정

매개 변수	설명
데이터 소스 클래스 이름	DataSource/ConnectionPoolDataSource/XADataSource API를 구현하는 공급업체 관련 클래스 이름입니다. 이 클래스는 JDBC 드라이버에 있습니다.

표 6-1 JDBC 연결 풀의 일반 설정

매개 변수	설명
자원 유형	선택 항목에는 javax.sql.DataSource(로컬 트랜잭션에만 해당), javax.sql.XADataSource(전역 트랜잭션) 및 java.sql.ConnectionPoolDataSource(로컬 트랜잭션, 성능 향상 가능)가 포함됩니다.

풀 설정

물리적 데이터베이스 연결 집합은 풀에 상주합니다. 응용 프로그램에서 연결을 요청하면 풀에서 연결이 제거되고, 응용 프로그램에서 연결을 해제하면 연결이 풀로 반환됩니다.

표 6-2 JDBC 연결 풀에 대한 풀 설정

매개 변수	설명
초기 및 최소 풀 크기	풀의 최소 연결 수입니다. 이 값에 따라 풀을 처음 만들거나 Application Server를 시작할 때 풀에 있는 연결 수가 결정됩니다.
최대 풀 크기	풀의 최대 연결 수입니다.
풀 크기 조정 개수	풀이 최소 풀 크기로 축소되면 일괄 처리의 크기가 조정됩니다. 이 값에 따라 일괄 처리의 연결 수가 결정됩니다. 이 값을 너무 크게 하면 연결 재순환이 지체되며, 너무 작게 하면 효율성이 떨어집니다.
유휴 시간 초과	풀에서 연결이 유휴 상태로 있을 수 있는 최대 시간(초)입니다. 이 시간이 만료되면 연결이 풀에서 제거됩니다.
최대 대기 시간	연결을 요청한 응용 프로그램이 연결 시간 초과까지 대기하는 시간입니다. 기본 대기 시간이 길기 때문에 응용 프로그램이 무기한 중단된 것처럼 보일 수 있습니다.

연결 검증

선택에 따라 Application Server는 연결을 응용 프로그램에게 전달하기 전에 연결을 검증할 수 있습니다. 이렇게 검증을 하면 네트워크 실패나 데이터베이스 서버 충돌로 인해 데이터베이스를 사용할 수 없는 경우 Application Server가 데이터베이스 연결을 자동으로 다시 설정합니다. 연결 검증을 수행하면 추가 오버헤드가 발생하여 성능이 약간 저하됩니다.

표 6-3 JDBC 연결 풀의 연결 검증 설정

매개 변수	설명
연결 검증	연결 검증을 활성화하려면 필수 확인란을 선택합니다.
검증 방법	Application Server는 자동 완결, 메타 데이터 및 테이블 등 세 가지 방법으로 데이터베이스 연결을 검증할 수 있습니다. • 자동 완결 및 메타 데이터 - Application Server는 con.getAutoCommit() 및 con.getMetaData() 메소드를 호출하여 연결을 검증합니다. 그러나 많은 JDBC 드라이버에서 이러한 호출의 결과를 캐시하기 때문에 항상 신뢰할 수 있는 검증을 제공하는 것은 아닙니다. 이러한 호출의 캐시 여부를 판단하려면 드라이버 공급업체에 확인합니다. • 테이블 - 응용 프로그램에서 지정된 데이터베이스 테이블을 쿼리합니다. 테이블이 반드시 필요하며 액세스할 수 있어야 하지만 행은 없어도 됩니다. 행이 많이 있는 기존 테이블이나 이미 자주 액세스하는 테이블은 사용하지 마십시오.
테이블 이름	검증 방법 콤보 상자에서 테이블을 선택한 경우 여기에서 데이터베이스 테이블 이름을 지정합니다.
실패 시	모든 연결 닫기 확인란을 선택한 경우 연결이 한 번 실패하면 Application Server는 풀의 모든 연결을 닫고 연결을 다시 설정합니다. 이 확인란을 선택하지 않으면 개별 연결을 사용할 경우에만 연결이 다시 설정됩니다.

트랜잭션 격리

대개 많은 사용자가 동시에 데이터베이스에 액세스하기 때문에 한 트랜잭션에서 데이터를 읽는 동안 다른 트랜잭션에서 그 데이터를 업데이트할 수 있습니다. 트랜잭션의 격리 수준은 업데이트되는 데이터를 다른 트랜잭션에 표시하는 정도를 정의합니다. 격리 수준에 대한 자세한 내용은 데이터베이스 공급업체의 설명서를 참조하십시오.

표 6-4 JDBC 연결 풀에 대한 트랜잭션 격리 설정

매개 변수	설명
트랜잭션 격리	이 풀의 연결에 대한 트랜잭션 격리 수준을 선택할 수 있게 합니다. 이 확인란을 선택하지 않으면 연결은 JDBC 드라이버에서 제공하는 기본 격리 수준으로 실행됩니다.

표 6-4 JDBC 연결 풀에 대한 트랜잭션 격리 설정

매개 변수	설명
격리 수준 보장	격리 수준이 지정된 경우에만 적용할 수 있습니다. 보장 확인란은 선택한 경우 풀에서 가져온 모든 연결은 동일한 격리 수준을 갖게 됩니다. 예를 들어, 마지막으로 사용 시 연결의 격리 수준을 프로그래밍을 통해 변경한 경우(예: `con.setTransactionIsolation` 사용) 이 기법은 상태를 지정한 격리 수준으로 다시 변경합니다.

등록 정보

추가 등록 정보 테이블에서 데이터베이스 이름(URL), 아이디 및 비밀번호 같은 등록 정보를 지정할 수 있습니다. 데이터베이스 공급업체에 따라 등록 정보가 다르기 때문에 자세한 내용은 공급업체의 설명서를 참조하십시오.

연결 풀 설정 검증

연결 풀 설정을 검증하려면 다음 작업을 수행합니다.

1. 데이터베이스 서버를 시작합니다.
2. 핑을 누릅니다.

   관리 콘솔에서 데이터베이스에 연결을 시도합니다. 오류 메시지가 표시되면 데이터베이스 서버가 다시 시작되었는지 확인합니다.

JDBC 연결 풀 삭제

1. 트리 구성 요소에서 자원 노드를 확장합니다.
2. 자원에서 JDBC 노드를 확장합니다.
3. JDBC에서 연결 풀 노드를 선택합니다.
4. 연결 풀 페이지에서 삭제할 풀의 확인란을 선택합니다.
5. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-jdbc-connection-pool`

JDBC 자원 정보

- JDBC 자원 편집
- JDBC 자원 삭제

JDBC 자원 만들기

JDBC 자원(데이터 소스)은 응용 프로그램이 데이터베이스에 연결하는 방법을 제공합니다. JDBC 자원을 만들기 전에 먼저 JDBC 연결 풀을 만듭니다.

JDBC 자원을 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장합니다.
2. 자원에서 JDBC 노드를 확장합니다.
3. JDBC에서 JDBC 자원 노드를 선택합니다.
4. JDBC 자원 페이지에서 새로 만들기를 누릅니다.
5. JDBC 자원 만들기 페이지에서 자원의 설정을 지정합니다.
    a. JNDI 이름 필드에서 고유한 이름을 입력합니다. 일반적으로 이 이름은 `jdbc/` 문자열로 시작합니다. 예를 들면, `jdbc/payrolldb`입니다. 반드시 슬래시를 입력합니다.
    b. 풀 이름 콤보 상자에서 새로운 JDBC 자원과 연관시킬 연결 풀을 선택합니다.
    c. 기본적으로 자원은 만들자마자 사용(활성화)할 수 있습니다. 자원을 사용할 수 없게 하려면 사용 확인란을 선택 해제합니다.
    d. 설명 필드에 자원에 대한 간단한 설명을 입력합니다.
    e. 대상 섹션에서 자원을 사용할 수 있는 대상(클러스터 및 독립 실행형 서버 인스턴스)을 지정합니다. 왼쪽에서 원하는 대상을 선택하고 추가를 눌러 선택한 대상 목록에 추가합니다.
6. 확인을 누릅니다.

해당 `asadmin` 명령: `create-jdbc-resource`

JDBC 자원 편집

1. 트리 구성 요소에서 자원 노드를 확장합니다.
2. 자원에서 JDBC 노드를 확장합니다.

3. JDBC에서 JDBC 자원 노드를 확장합니다.
4. 편집할 JDBC 자원의 노드를 선택합니다.
5. JDBC 자원 편집 페이지에서 다음 작업을 수행할 수 있습니다.
   a. 풀 이름 콤보 상자에서 다른 연결 풀을 선택합니다.
   b. 설명 필드에서 자원에 대한 간단한 설명을 변경합니다.
   c. 자원을 활성화하거나 비활성화하려면 확인란을 선택하거나 선택 해제합니다.
   d. 자원을 사용할 수 있는 대상(클러스터 및 독립 실행형 서버 인스턴스)을 변경하려면 대상 탭을 선택합니다.

      목록의 기존 대상에 대한 확인란을 선택한 다음, 활성화를 눌러 해당 대상의 자원을 활성화하거나, 비활성화를 눌러 해당 대상의 자원을 비활성화합니다.

      대상 관리를 눌러 대상을 목록에 추가하거나 제거합니다. 대상 관리 페이지의 왼쪽에 있는 사용 가능 목록에서 원하는 대상을 선택하고, 추가를 눌러 선택한 대상 목록에 추가합니다. 제거를 눌러 선택한 목록에서 대상을 제거합니다.

      확인을 눌러 변경 사항을 사용 가능한 대상에 저장합니다.
6. 저장을 눌러 편집한 내용을 적용합니다.

JDBC 자원 삭제

1. 트리 구성 요소에서 자원 노드를 확장합니다.
2. 자원에서 JDBC 노드를 확장합니다.
3. JDBC에서 JDBC 자원 노드를 선택합니다.
4. JDBC 자원 페이지에서 삭제할 자원의 확인란을 선택합니다.
5. 삭제를 누릅니다.

- JDBC 자원 활성화 및 비활성화

JDBC 자원 활성화 및 비활성화

1. 트리 구성 요소에서 JDBC 자원 노드를 확장하거나 독립 실행형 인스턴스를 확장하여 서버 인스턴스 노드 자원 탭을 선택합니다.

2. 자원 페이지에서 활성화하거나 비활성화할 자원의 확인란을 선택합니다.
3. 활성화 또는 비활성화를 누릅니다.

지속성 관리자 자원 정보

- 지속성 관리자 자원 만들기

지속성 관리자 자원 만들기

이 기능은 역방향 호환성을 위해 필요합니다. Application Server 버전 7에서 실행하려면 컨테이너 관리 지속성 Bean(EJB 구성 요소 유형)이 있는 응용 프로그램을 위해 지속성 관리자 자원이 필요했습니다.

지속성 관리자 자원을 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장합니다.
2. 자원에서 지속성 관리자 노드를 선택합니다.
3. 지속성 관리자 페이지에서 새로 만들기를 누릅니다.
4. 지속성 관리자 만들기 페이지에서 다음 설정을 지정합니다.
    a. JNDI 이름 필드에서 고유한 이름을 입력합니다. 예를 들면, `jdo/mypm`을 입력합니다. 반드시 슬래시를 입력합니다.
    b. 팩토리 클래스 필드에서 이 릴리스와 함께 제공된 기본 클래스를 유지하거나 다른 구현 클래스를 입력합니다.
    c. 연결 풀 콤보 상자에서 새로운 지속성 관리자 자원이 속할 연결 풀을 선택합니다.
    d. 기본적으로 새로운 지속성 관리자 자원은 활성화됩니다. 비활성화하려면 사용 가능 확인란을 선택 해제합니다.
    e. 대상 섹션에서 자원을 사용할 수 있는 대상(클러스터 및 독립 실행형 서버 인스턴스)을 지정합니다. 왼쪽에서 원하는 대상을 선택하고 추가를 눌러 선택한 대상 목록에 추가합니다.
5. 확인을 누릅니다.

해당 `asadmin` 명령: `create-persistence-resource`

지속성 관리자 자원 편집

기존 지속성 관리자 자원 등록 정보를 편집하려면 다음 작업을 수행합니다.

1. 지속성 관리자 등록 정보 편집 탭에서 등록 정보 추가 버튼을 선택합니다.

   새로운 행이 추가 등록 정보 테이블에 추가됩니다.

2. 원하는 등록 정보와 값을 추가합니다.

자원 대상 관리

자원 대상을 관리하려면 다음 작업을 수행합니다.

1. 대상 탭을 선택하여 자원이 상주하는 대상(클러스터 및 독립 실행형 서버 인스턴스)을 변경합니다.
2. 목록의 기존 대상에 대한 확인란을 선택한 다음, 활성화를 눌러 해당 대상의 자원을 활성화하거나, 비활성화를 눌러 해당 대상의 자원을 비활성화합니다.
3. 대상 관리를 눌러 대상을 목록에 추가하거나 제거합니다. 대상 관리 페이지의 왼쪽에 있는 사용 가능 목록에서 원하는 대상을 선택하고, 추가를 눌러 선택된 대상 목록에 추가합니다. 제거를 눌러 선택된 목록에서 대상을 제거합니다.
4. 확인을 눌러 변경 사항을 사용 가능한 대상에 저장합니다.
5. 저장을 누릅니다.

지속성 관리자 자원 삭제

1. 트리 구성 요소에서 지속성 관리자 노드를 확장합니다.
2. 지속성 관리자 노드를 선택합니다.
3. 지속성 관리자 페이지에서 삭제할 지속성 관리자의 확인란을 선택합니다.
4. 삭제를 누릅니다.

해당 asadmin 명령: `delete-persistence-resource`

지속성 관리자 자원 활성화 및 비활성화

1. 트리 구성 요소에서 지속성 관리자 노드를 확장합니다.
2. 활성화하거나 비활성화할 자원의 확인란을 선택합니다.
3. 활성화 또는 비활성화를 누릅니다.

가용성 및 세션 지속성 구성

이 장에서는 Sun Java™ System Application Server Enterprise Edition 환경에서 세션 지속성 및 가용성을 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- 가용성 및 세션 지속성 정보
- 가용성 구성을 위한 관리 콘솔 작업

가용성 및 세션 지속성 정보

- 세션 지속성이 필요한 이유
- 세션 지속성 구성의 개요
- 가용성 수준
- HTTP 세션 상태에서 단일 사인 온의 가용성
- 샘플 응용 프로그램

세션 지속성이 필요한 이유

응용 프로그램 세션이 계속되면 세션의 일부이면서 전통적인 데이터베이스에 저장되지 않는 데이터가 있습니다. 장바구니 내용이 그런 데이터의 예입니다. Sun Java System Application Server는 저장소에 이 세션 데이터를 저장하거나 지속할 수 있는 기능을 제공합니다. 따라서 Application Server 인스턴스에 오류가 발생해도 세션 상태를 복구할 수 있고 정보 손실 없이 세션을 계속할 수 있습니다.

J2EE 응용 프로그램의 경우 세션 데이터는 대개 HTTP 세션이나 SFSB(Stateful Session Bean) 세션에 저장됩니다. Sun Java System Application Server는 HTTP 세션과 SFSB 세션 모두의 상태 지속성을 지원합니다. HTTP 세션과 SFSB 세션 모두에 저장된 특정 J2EE 객체 참조의 페일오버도 지원됩니다. *Developer's Guide*를 참조하십시오.

Sun Java System Application Server에 번들로 제공되는 HADB(high-availability database)는 지속성 저장소 역할을 함으로서 세션 데이터의 고가용성을 제공합니다.

세션 지속성 구성의 개요

성공적인 세션 지속성 구성을 위해 제공되는 순서대로 이 단계를 수행합니다. 일부 단계는 이전 단계 하나 이상을 사전 조건으로 필요로 하기 때문입니다.

1. 클러스터용 HADB 데이터베이스를 만듭니다. *Reference Manual*에서 `configure-ha-cluster` 명령의 설명을 참조하십시오.
2. 클러스터에 대한 HTTP 로드 균형 조정을 설정합니다. 3장, "로드 균형 조정 및 페일오버 구성"을 참조하십시오.
3. 세션 지속성을 지원해야 하는 Application Server 인스턴스 및 웹 또는 EJB 컨테이너에 대해 사용 가능성을 활성화하고 세션 지속성 설정을 구성합니다. 다음 중 하나를 선택하십시오.
   - 151페이지의 "가용성 구성을 위한 관리 콘솔 작업"을 참조하십시오.
   - *Reference Manual*에서 `configure-ha-persistence` 명령의 설명을 참조하십시오.
4. 사용 가능성을 활성화하지 *않은* 경우 원한다면 SFSB에 대한 파일 시스템 세션 저장소를 변경할 수 있습니다. 151페이지의 "가용성이 비활성화된 경우 SFSB 세션 저장소 구성"을 참조하십시오.
5. 클러스터의 모든 서버 인스턴스를 다시 시작합니다.
6. 요구하는 특정 SFSB에 대해 사용 가능성을 활성화하고 세션 상태의 검사점 지정이 필요한 메소드를 선택합니다. *Developer's Guide*를 참조하십시오.
7. 가용성이 높아야 하는 웹 모듈을 배포 가능하게 합니다. *Developer's Guide*를 참조하십시오.
8. 배포 중에 J2EE 응용 프로그램, 웹 모듈 또는 EJB 모듈에 대한 가용성을 활성화합니다. 관리 콘솔에서 가용성 사용 가능 확인란을 선택하거나 `--availabilityenabled` 옵션을 `true`로 설정하여 `deploy` 명령을 사용합니다.

주 세션 지속성은 동적 배포, 동적 재로드 및 자동 배포와 호환되지 않습니다. 이 배포 기능은 작업 환경이 아니라 개발 환경용입니다. 이 기능을 비활성화하는 방법에 대한 자세한 내용은 5장, "응용 프로그램 배포"를 참조하십시오.

주 인스턴스에서 현재 요청을 처리할 경우 인스턴스가 처리 중인 요청을 처리할 수 있는 충분한 시간을 가질 수 있도록 인스턴스를 다시 시작하기 전에 인스턴스를 중지합니다. 자세한 내용은 서버 인스턴스 또는 클러스터 비활성화(정지)를 참조하십시오.

가용성 수준

다섯 개의 다른 수준에서 가용성을 활성화할 수 있습니다.

1. 기본적으로 활성화된 서버 인스턴스
2. 기본적으로 활성화된 웹 또는 EJB 컨테이너
3. 기본적으로 비활성화된 응용 프로그램
4. 기본적으로 비활성화된 독립 실행형 웹 또는 EJB 모듈
5. 기본적으로 비활성화된 SFSB

가용성을 특정 수준에서 활성화하려면 그보다 더 높은 모든 수준에서도 활성화해야 합니다. 예를 들어, 응용 프로그램 수준에서 가용성을 활성화하려면 서버 인스턴스 및 컨테이너 수준에서도 활성화해야 합니다.

지정한 수준의 기본값은 다음으로 높은 수준의 설정값입니다. 예를 들어, 컨테이너 수준에서 가용성을 활성화한 경우 기본적으로 응용 프로그램 수준에서도 가용성이 활성화됩니다.

서버 인스턴스 수준에서 가용성을 비활성화한 경우 다른 수준에서 활성화해도 적용되지 않습니다. 서버 인스턴스 수준에서 가용성을 활성화한 경우 명시적으로 비활성화하지 않으면 모든 수준에서 가용성이 활성화됩니다.

HTTP 세션 상태에서 단일 사인 온의 가용성

단일 Application Server 인스턴스의 경우 한 응용 프로그램에서 사용자를 인증하면 동일한 인스턴스에서 실행 중인 다른 응용 프로그램에서 사용자를 개별적으로 다시 인증할 필요가 없습니다. 이를 *단일 사인 온*이라고 합니다. 단일 사인 온에 대한 자세한 내용은 225 페이지의 "단일 사인 온(SSO) 검증"을 참조하십시오.

클러스터의 다른 인스턴스에 HTTP 세션이 페일오버되더라도 이 기능이 계속 작동하려면 단일 사인 온 정보가 HADB에 지속되어야 합니다. 먼저 서버 인스턴스와 웹 컨테이너에 대한 가용성을 활성화한 다음, 단일 사인 온 상태 지속성을 활성화합니다. 152페이지의 "서버 인스턴스 수준에서 가용성 구성"을 참조하십시오.

단일 이름 및 비밀번호 조합을 통해 액세스할 수 있는 응용 프로그램이 *단일 사인 온 그룹*을 구성합니다.

단일 사인 온 그룹의 일부인 응용 프로그램에 해당하는 HTTP 세션의 경우 세션 중 하나가 시간 초과되더라도 다른 세션은 무효화되지 않고 계속 사용할 수 있습니다. 이는 한 세션의 시간 초과가 다른 세션의 가용성에 영향을 미치지 않기 때문입니다.

이 동작의 결과, 세션이 시간 초과되고 세션을 실행 중인 동일한 브라우저 창에서 해당하는 응용 프로그램에 액세스할 경우 다시 인증할 필요가 없습니다. 그러나 새로운 세션이 만들어집니다.

두 개의 다른 응용 프로그램을 사용하는 단일 사인 온 그룹의 일부인 장바구니 응용 프로그램을 예로 듭니다. 다른 두 응용 프로그램의 세션 시간 초과 값이 장바구니 응용 프로그램의 세션 초과 값보다 더 높은 것으로 가정합니다. 장바구니 응용 프로그램의 세션이 시간 초과되고 세션을 실행 중인 동일한 브라우저 창에서 장바구니 응용 프로그램을 실행할 경우 다시 인증할 필요가 없습니다. 그러나 이전 장바구니는 손실되고 새로운 장바구니를 만들어야 합니다. 장바구니 응용 프로그램을 실행하는 세션이 시간 초과되더라도 다른 두 응용 프로그램은 평소대로 계속 실행됩니다.

마찬가지로 다른 두 응용 프로그램에 해당하는 세션이 시간 초과되는 것으로 가정합니다. 세션을 실행 중인 동일한 브라우저 창에서 응용 프로그램에 연결하고 있는 동안에는 다시 인증할 필요가 없습니다.

주 세션이 시간 초과된 경우에만 이 동작이 적용됩니다. 단일 사인 온이 활성화되고 `HttpSession.invalidate()`를 사용하여 세션 중 하나를 무효화한 경우 단일 사인 온 그룹에 속하는 모든 응용 프로그램의 세션이 무효화됩니다. 단일 사인 온 그룹에 속하는 응용 프로그램에 액세스할 경우 다시 인증을 받아야 하고 응용 프로그램에 액세스하는 클라이언트에 대해 새로운 세션이 만들어집니다.

샘플 응용 프로그램

다음 디렉토리에는 HTTP 및 SFSB 세션 지속성을 설명하는 샘플 응용 프로그램이 포함되어 있습니다.

install_dir`/samples/ee-samples/highavailability`
install_dir`/samples/ee-samples/failover`

가용성 구성을 위한 관리 콘솔 작업

- 가용성이 비활성화된 경우 SFSB 세션 저장소 구성
- 서버 인스턴스 수준에서 가용성 구성
- 웹 컨테이너 수준에서 가용성 구성
- EJB 컨테이너 수준에서 가용성 구성

가용성이 비활성화된 경우 SFSB 세션 저장소 구성

가용성이 비활성화된 경우 지속성이 아니라 SFSB 상태 비활성화를 위해 로컬 파일 시스템을 사용합니다. SFSB 상태가 저장되는 장소를 변경하려면 EJB 컨테이너의 세션 저장소 위치 설정을 변경합니다. 211페이지의 "일반 EJB 설정 구성"을 참조하십시오.

서버 인스턴스 수준에서 가용성 구성

관리 콘솔을 사용하여 서버 인스턴스 수준에서 가용성을 활성화하거나 비활성화하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 편집할 구성의 노드를 확장합니다.
3. 가용성 서비스 노드를 선택합니다.
4. 가용성 서비스 페이지로 이동합니다.
5. 가용성 서비스 확인란을 선택하여 인스턴스 수준 가용성을 활성화합니다. 비활성화하려면 확인란을 선택 해제합니다.

   세션 지속성을 위해 HADB 연결에 사용한 JDBC 자원을 변경한 경우 저장소 풀 이름을 변경할 수 있습니다. 자세한 내용은 *Reference Manual*의 `configure-ha-cluster` 명령을 참조하십시오.
6. 저장 버튼을 누릅니다.
7. 인스턴스 노드를 확장합니다.
8. 서버 인스턴스를 선택합니다.
9. 서버 인스턴스 페이지로 이동합니다.
10. 서버를 다시 시작합니다.

웹 컨테이너 수준에서 가용성 구성

가용성을 활성화하거나 개별 웹 응용 프로그램의 가용성 설정을 대체하려면 `sun-web.xml` 파일의 설정을 사용합니다. 자세한 내용은 *Developer's Guide*를 참조하십시오.

관리 콘솔을 사용하여 웹 컨테이너 가용성을 활성화하거나 비활성화하려면 다음 작업을 수행합니다.

1. 웹 컨테이너 가용성 탭을 선택한 다음 가용성 서비스 확인란을 선택합니다. 비활성화하려면 확인란을 선택 해제합니다. 다음 선택 사항인 설정을 변경할 수도 있습니다.

- 지속성 유형: 가용성이 활성화된 웹 응용 프로그램에 대한 세션 지속성 기법을 지정합니다. 허용되는 값은 memory(지속성 없음), file(파일 시스템) 및 ha(HADB)입니다. 가용성을 활성화한 경우 기본값은 ha입니다. 가용성을 비활성화한 경우 기본값은 memory입니다. 세션 지속성이 필요한 작업 환경의 경우 ha를 사용합니다.

  지속성 유형을 memory로 설정한 경우 sessionFilename 등록 정보를 사용하여 서버 인스턴스가 적절하게 종료된 경우 HTTP 세션 상태가 저장되는 파일 시스템 위치를 지정할 수 있습니다. 이는 내부 테스트에는 유용하지만 작업 환경에는 지원되지 않습니다.

  지속성 유형이 file로 설정된 경우 디렉토리 등록 정보를 사용하여 HTTP 세션 상태가 저장되는 파일 시스템 위치를 지정할 수 있습니다. 파일 시스템에 대한 지속성은 내부 테스트에 유용하지만 작업 환경에는 지원되지 않습니다.

- 지속성 빈도: 세션 상태가 저장되는 빈도를 저장합니다. 지속성 유형이 ha일 경우에만 해당됩니다. 허용되는 값은 다음과 같습니다.

  - web-method - 응답을 다시 클라이언트에 전송하기 전에 각 웹 요청 끝에 세션 상태가 저장됩니다. 이 모드는 오류 시 세션 상태의 완벽한 업데이트를 가장 확실하게 보장합니다. 이 값이 기본값입니다.

  - time-based - reapIntervalSeconds 저장소 등록 정보에서 설정한 빈도로 세션 상태가 백그라운드로 저장됩니다. 이 모드는 세션 상태의 완벽한 업데이트를 확실하게 보장하지 못합니다. 그러나 요청 후마다 상태를 저장하지 않기 때문에 성능이 크게 향상됩니다. 이 등록 정보를 설정하려면 210페이지의 "저장소 등록 정보 구성"을 참조하십시오.

- 지속성 범위: 세션 상태가 저장되는 양을 지정합니다. 지속성 유형이 ha일 경우에만 해당됩니다. 허용되는 값은 다음과 같습니다.

  - session - 항상 전체 세션 상태가 저장됩니다. 이 모드는 분산 가능한 웹 응용 프로그램의 경우 세션 데이터의 정확한 저장을 가장 확실하게 보장합니다. 이 값이 기본값입니다.

  - modified-session - 수정하면 전체 세션 상태가 저장됩니다. HttpSession.setAttribute() 또는 HttpSession.removeAttribute()를 호출한 경우 세션이 수정된 것으로 간주됩니다. 속성을 변경할 때마다 setAttribute()를 호출하도록 해야 합니다. 이는 J2EE 사양 요구 사항이 아니지만 이 모드가 제대로 작동하려면 필요합니다.

- `modified-attribute` - 수정된 세션 속성만 저장됩니다. 이 모드가 제대로 작동하려면 몇 가지 지침을 수행해야 합니다.

  세션 상태가 수정될 때마다 `setAttribute()`를 호출합니다.

  속성 간에는 상호 참조가 없어야 합니다. 별개 속성 키의 객체 그래프는 별도로 일련화 및 저장됩니다. 별도 키의 객체 간에 객체 상호 참조가 있을 경우 제대로 일련화 및 일련화 해제되지 않습니다.

  여러 속성에서 세션 상태를 분배하거나 최소한 읽기 전용 속성 및 수정 가능한 속성 간에 세션 상태를 분배합니다.

- 단일 사인 온 상태: 단일 사인 온 상태의 지속성을 활성화하려면 이 확인란을 선택합니다. 비활성화하려면 확인란을 선택 해제합니다.
- HTTP 세션 저장소: 세션 지속성을 위해 HADB 연결에 사용한 JDBC 자원을 변경한 경우 HTTP 세션 저장소를 변경할 수 있습니다. 자세한 내용은 *Reference Manual*의 `configure-ha-cluster` 명령 설명을 참조하십시오.

2. 저장 버튼을 누릅니다.
3. 세션 지속성에 영향을 미치는 선택적인 추가 설정을 변경하려면 209페이지의 "웹 컨테이너 세션 구성"을 참조하십시오.
4. 인스턴스 노드를 확장합니다.
5. 서버 인스턴스를 선택합니다.
6. 서버 인스턴스 페이지로 이동합니다.
7. 서버를 다시 시작합니다.

EJB 컨테이너 수준에서 가용성 구성

가용성을 활성화하고 개별 Stateful Session Bean(SFSB)에 대해 검사점을 지정할 메소드를 선택하려면 `sun-ejb-jar.xml` 파일의 설정을 사용합니다. 자세한 내용은 *Developer's Guide*를 참조하십시오.

관리 콘솔을 사용하여 EJB 컨테이너 가용성을 활성화하거나 비활성화하려면 다음 작업을 수행합니다.

1. EJB 컨테이너 가용성 탭을 선택한 다음 가용성 서비스 확인란을 선택합니다. 비활성화하려면 확인란을 선택 해제합니다. 다음과 같은 선택적인 설정을 변경할 수도 있습니다.

- HA 지속성 유형: 가용성이 활성화된 SFSB에 대한 세션 지속성 및 비활성화 기법을 지정합니다. 허용되는 값은 `file`(파일 시스템) 및 `ha`(HADB)입니다. 세션 지속성이 필요한 작업 환경의 경우 기본값인 `ha`를 사용합니다.
- SFSB 지속성 유형: 가용성이 활성화되지 *않은* SFSB에 대한 비활성화 기법을 지정합니다. 허용되는 값은 `file`(파일 시스템) 및 `ha`입니다.

  지속성 유형을 `file`로 설정한 경우 EJB 컨테이너는 비활성화된 Session Bean 상태가 저장되는 파일 시스템 위치를 지정합니다. 211페이지의 "일반 EJB 설정 구성"을 참조하십시오. 파일 시스템에 검사점을 지정하는 것은 내부 테스트에는 유용하지만 작업 환경에는 지원되지 않습니다.
- SFSB 저장소 풀 이름: 세션 지속성을 위해 HADB 연결에 사용한 JDBC 자원을 변경한 경우 SFSB 저장소 풀 이름을 변경할 수 있습니다. 자세한 내용은 *Reference Manual*의 `configure-ha-cluster` 명령 설명을 참조하십시오.

2. 저장 버튼을 누릅니다.
3. 인스턴스 노드를 확장합니다.
4. 서버 인스턴스를 선택합니다.
5. 서버 인스턴스 페이지로 이동합니다.
6. 서버를 다시 시작합니다.

JMS(Java Message Service) 자원 구성

이 장에서는 JMS(Java Message Service) API를 사용하는 응용 프로그램의 자원 구성 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- JMS 자원 정보
- JMS 연결 팩토리의 관리 콘솔 작업
- JMS 대상 자원을 위한 관리 콘솔 작업
- JMS 물리적 대상을 위한 관리 콘솔 작업
- JMS 공급자를 위한 관리 콘솔 작업

JMS 자원 정보

- Application Server의 JMS 공급자
- JMS 자원
- JMS 자원과 커넥터 자원 간의 관계

Application Server의 JMS 공급자

Application Server에서는 Application Server에 Sun Java System Message Queue(이전의 Sun ONE Message Queue)를 통합하여 JMS(Java Message Service)를 구현합니다. 기본적인 JMS API 관리 작업에 대해서는 Application Server 관리 콘솔을 사용합니다. Message Queue 클러스터 관리를 포함한 고급 작업의 경우 *install_dir*`/imq/bin` 디렉토리에 제공된 도구를 사용합니다.

Message Queue 관리에 대한 자세한 내용은 *Sun Java System Message Queue 관리 설명서*를 참조하십시오.

JMS 자원

JMS(Java Message Service) API에서는 두 종류의 관리 대상 객체를 사용합니다.

- 연결 팩토리 - 응용 프로그램에서 다른 JMS 객체를 프로그래밍 방식으로 만들 수 있게 해주는 객체
- 대상 - 메시지를 위한 저장소 역할

이러한 객체는 관리상의 목적으로 만들며, 객체를 만드는 방법은 JMS 구현마다 다릅니다. Application Server에서 다음 작업을 수행합니다.

- 연결 팩토리 자원을 만들어 연결 팩토리 만들기
- 다음 두 객체를 만들어 대상 만들기
  - 물리적 대상
  - 물리적 대상을 참조하는 대상 자원

JMS 응용 프로그램에서는 JNDI API를 사용하여 연결 팩토리와 대상 자원에 액세스합니다. JMS 응용 프로그램은 대개 최소한 연결 팩토리 하나와 대상 하나를 사용합니다. 만들 자원을 알아보려면 응용 프로그램을 살펴 보거나 응용 프로그램 개발자에게 문의하십시오.

연결 팩토리에는 다음과 같은 세 가지 유형이 있습니다.

- `QueueConnectionFactory` 객체 - 지점간 통신에 사용됩니다.
- `TopicConnectionFactory` 객체 - 게시-가입 통신에 사용됩니다.
- `ConnectionFactory` 객체 - 지점간 통신과 게시-가입 통신 모두에 사용할 수 있으므로 새로운 응용 프로그램에 권장합니다.

대상에는 다음과 같은 두 가지 종류가 있습니다.

- `Queue` 객체 - 지점간 통신에 사용됩니다.
- `Topic` 객체 - 게시-가입 통신에 사용됩니다.

*J2EE 1.4 Tutorial*의 JMS 장에서는 이 두 가지 유형의 통신과 JMS의 다른 면에 대해 자세히 설명합니다(`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html` 참조).

자원을 만드는 순서는 상관이 없습니다.

J2EE 응용 프로그램의 경우 Application Server 배포 설명자에서 다음과 같이 연결 팩토리와 대상 자원을 지정합니다.

- `resource-ref` 또는 `mdb-connection-factory` 요소에서 연결 팩토리 JNDI 이름을 지정합니다.
- Message-Driven Bean의 `ejb` 요소와 `message-destination` 요소에서 대상 자원 JNDI 이름을 지정합니다.
- Enterprise Bean 배포 설명자의 `message-driven` 요소 또는 `message-destination-ref` 요소 안에 있는 `message-destination-link` 요소에서 물리적인 대상 이름을 지정합니다. 그리고 `message-destination` 요소에서도 지정합니다. `message-destination-ref` 요소는 새로운 응용 프로그램에서 더 이상 사용하지 않는 `resource-env-ref` 요소를 대체합니다. Application Server 배포 설명자의 `message-destination` 요소에서 물리적 대상 이름을 대상 자원 이름과 링크합니다.

JMS 자원과 커넥터 자원 간의 관계

Application Server는 `jmsra`라고 하는 시스템 자원 어댑터를 사용하여 JMS를 구현합니다. 사용자가 JMS 자원을 만들면 Application Server는 관리 콘솔 트리 보기의 커넥터 노드 아래에 표시되는 커넥터 자원을 자동으로 만듭니다.

사용자가 만든 JMS 연결 팩토리마다 Application Server는 커넥터 연결 풀과 연결 자원을 만듭니다. 사용자가 만든 JMS 대상마다 Application Server는 관리 객체 자원을 만듭니다. 사용자 JMS 자원을 삭제하면 Application Server에서 커넥터 자원을 자동으로 삭제합니다.

JMS 자원 노드 대신 관리 콘솔의 커넥터 노드를 사용하여 JMS 시스템 자원 어댑터에 대한 커넥터 자원을 만들 수 있습니다. 자세한 내용은 11장, "커넥터 자원"을 참조하십시오.

JMS 연결 팩토리의 관리 콘솔 작업

JMS 연결 팩토리 자원 만들기

JMS 연결 팩토리 자원을 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JMS 자원 노드를 확장합니다.
2. 연결 팩토리 노드를 선택합니다.
3. JMS 연결 팩토리 페이지에서 새로 만들기를 누릅니다. JMS 연결 팩토리 만들기 페이지가 나타납니다.
4. JNDI 이름 필드에서 연결 팩토리 이름을 입력합니다. 예를 들면 다음과 같습니다.

   `jms/ConnectionFactory1`

   JMS 자원에 대해 이름 지정 하위 컨텍스트 접두어 `jms/`를 사용하는 것이 좋습니다.
5. 유형 드롭다운 목록에서 `javax.jms.ConnectionFactory`, `javax.jms.QueueConnectionFactory` 또는 `javax.jms.TopicConnectionFactory`를 선택합니다.
6. 런타임에서 자원을 활성화하려면 사용 가능 확인란을 선택합니다.
7. 고급 영역에서 연결 팩토리 속성에 필요한 대로 값을 변경합니다. 이 속성에 대한 자세한 내용은 194페이지의 "커넥터 연결 풀 편집"의 "커넥터 연결 풀에 대한 풀 설정" 테이블을 참조하십시오. Application Server는 연결 팩토리에 대해 작성된 커넥터 연결 풀에 이 속성을 적용합니다.

   JMS 연결 팩토리 자원의 경우 다음과 같이 트랜잭션 지원 값을 지정합니다.

- 트랜잭션 범위 내 자원이 여러 개 관련된 트랜잭션에 사용할 수 있는 자원에 대해 XATransaction(기본값)을 지정합니다 .예를 들어, 이 자원에 JDBC 자원, 커넥터 자원 또는 다른 JMS 연결 팩토리 자원을 포함합니다. 이 값은 최고의 융통성을 제공합니다. XATransaction으로 구성된 자원은 2단계 커밋 작업에 참여합니다.
- 트랜잭션 범위 내 자원 하나만 관련된 트랜잭션에 사용하거나 여러 XA 자원이 관련된 분산 트랜잭션의 마지막 에이전트로 사용할 수 있는 자원에 대해 LocalTransaction을 지정합니다. 이 값을 사용하면 성능이 훨씬 향상됩니다. LocalTransaction으로 구성된 자원은 2단계 커밋 작업에 사용되지 않습니다.
- 트랜잭션에 참여할 수 없는 자원에 대해 NoTransaction을 지정합니다. 이 설정은 JMS 응용 프로그램에서 제한적으로 사용됩니다.

8. 추가 등록 정보 영역에서 응용 프로그램에 필요한 등록 정보 값을 제공합니다. 다음 표에는 사용 가능한 등록 정보가 나열됩니다.

표 8-1 JMS 연결 팩토리의 추가 등록 정보

등록 정보 이름	설명
ClientId	영구 가입자가 사용하는 연결 팩토리에 대해 클라이언트 아이디를 지정합니다.
AddressList	응용 프로그램이 통신하는 인스턴스나 메시지 브로커 인스턴스의 이름(필요에 따라 포트 번호)을 지정합니다. 목록의 각 주소는 연결에 대한 호스트 이름(필요에 따라 호스트 포트와 연결 서비스)을 지정합니다. 예를 들어, 값은 earth 또는 earth:7677이 될 수 있습니다. 기본값(7676) 이외의 포트에서 메시지 브로커를 실행할 경우 포트 번호를 지정합니다. 클러스터링된 환경에서 등록 정보 설정이 여러 호스트 및 포트를 지정할 경우 AddressListBehavior 등록 정보가 RANDOM으로 설정되어 있지 않는 한, 목록에서 사용 가능한 첫 번째 호스트가 사용됩니다. 자세한 내용은 *Sun Java System Message Queue Developer's Guide for Java Clients*를 참조하십시오. 기본값: 로컬 호스트 및 기본 포트 번호(7676). 클라이언트는 로컬 호스트의 7676 포트에서 브로커에 연결을 시도합니다.
MessageServiceAddressList	AddressList와 동일합니다. 이 등록 정보 이름은 더 이상 사용되지 않습니다. 대신 AddressList를 사용합니다.
UserName	연결 팩토리에 대한 아이디입니다. 기본값: guest

표 8-1 JMS 연결 팩토리의 추가 등록 정보 *(계속)*

등록 정보 이름	설명
Password	연결 팩토리에 대한 비밀번호입니다. 기본값: guest
ReconnectEnabled	활성화한 경우(값 = true), 연결이 끊어지면 클라이언트 런타임에서 메시지 서버(또는 AddressList의 주소 목록)에 다시 연결하도록 지정합니다. 기본값: true
ReconnectAttempts	클라이언트 런타임에서 AddressList의 주소에 연결(또는 재연결)을 몇 번 시도한 후 목록에 있는 다음 주소로 연결을 시도하는지를 지정합니다. -1 값은 재연결 시도 횟수가 제한이 없음을 나타냅니다. 클라이언트 런타임은 성공할 때까지 첫 번째 주소에 연결을 시도합니다. 기본값: 3
ReconnectInterval	재연결 시도 간 간격(밀리초)을 지정합니다. AddressList의 각 주소에 대한 재연결 시도와 목록의 연속된 주소에 이 값이 적용됩니다. 간격이 너무 짧을 경우 브로커가 복구할 시간이 없습니다. 너무 길 경우 재연결 시 지연이 너무 길게 느껴질 수 있습니다. 기본값: 30000
AddressListBehavior	연결 시도가 AddressList 속성의 주소 순서(PRIORITY)인지 임의의 순서(RANDOM)인지 지정합니다. RANDOM은 재연결할 때 AddressList에서 임의의 주소를 선택한다는 것을 의미합니다. 여러 클라이언트에서 동일한 연결 팩토리를 사용하여 연결할 가능성이 큰 경우 이 값을 사용하면 모든 클라이언트가 동일한 주소에 연결하는 것을 방지할 수 있습니다. PRIORITY는 재연결 시 항상 AddressList의 첫 번째 서버 주소에 연결하고 첫 번째 브로커를 사용할 수 없는 경우에만 다른 주소를 사용한다는 것을 의미합니다. 기본값: RANDOM
AddressListIterations	연결을 설정하거나 재설정할 때 클라이언트 런타임이 AddressList를 반복하는 횟수를 지정합니다. -1 값은 제한이 없음을 나타냅니다. 기본값: 3

9. 대상 영역에서 다음 작업을 수행합니다.

a. 사용 가능한 열에서 배포할 자원을 사용하는 응용 프로그램에 해당하는 대상을 선택합니다. 사용 가능한 대상은 기본 서버 인스턴스 `server`뿐만 아니라 사용 가능한 클러스터 및 서버 인스턴스를 포함합니다.

b. 추가를 눌러 대상을 선택한 열로 이동합니다.

10. 확인을 눌러 연결 팩토리를 저장합니다.

해당 `asadmin` 명령: `create-jms-resource`

JMS 연결 팩토리 자원 편집

JMS 연결 팩토리 자원을 편집하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JMS 자원 노드를 확장합니다.

2. 연결 팩토리 노드를 확장합니다.

3. 편집할 연결 팩토리를 선택합니다.

4. JMS 연결 팩토리 편집 페이지에서 다음 작업을 수행할 수 있습니다.

- 설명 필드의 텍스트를 수정합니다.
- 자원을 활성화하거나 비활성화하려면 사용 가능 확인란을 선택하거나 선택 해제합니다.
- 고급 영역에서 속성 값을 변경합니다.
- 등록 정보를 추가, 제거 또는 수정합니다.

5. 필요에 따라 대상 탭을 눌러 JMS 연결 팩토리 자원 대상 페이지로 이동합니다. 이 페이지에서 다음 작업을 수행합니다.

a. 대상 관리를 눌러 자원 대상 관리 페이지를 엽니다.

이 페이지에서 사용 가능한 열과 선택한 열에 대상을 이동합니다. 선택한 열에 배포할 자원을 사용하는 응용 프로그램에 해당하는 대상을 넣어야 합니다. 사용 가능한 대상은 기본 서버 인스턴스 `server`뿐만 아니라 사용 가능한 클러스터와 서버 인스턴스를 포함합니다. 확인을 눌러 변경 사항을 저장합니다.

b. 대상에 대한 확인란을 선택한 다음, 활성화를 눌러 해당 대상의 자원을 활성화하거나, 비활성화를 눌러 해당 대상의 자원을 비활성화합니다.

6. 저장을 눌러 변경 사항을 저장합니다.

JMS 연결 팩토리 자원 삭제

JMS 연결 팩토리 자원을 삭제하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JMS 자원 노드를 확장합니다.
2. 연결 팩토리 노드를 선택합니다.
3. JMS 연결 팩토리 페이지에서 삭제할 연결 팩토리 이름 옆에 있는 확인란을 선택합니다.
4. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-jms-resource`

JMS 대상 자원을 위한 관리 콘솔 작업

- JMS 대상 자원 만들기
- JMS 대상 자원 편집
- JMS 대상 자원 삭제

JMS 대상 자원 만들기

JMS 대상 자원을 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JMS 자원 노드를 확장합니다.
2. 대상 자원 노드를 선택합니다.
3. JMS 대상 자원 페이지에서 새로 만들기를 누릅니다. JMS 대상 자원 만들기 페이지가 나타납니다.
4. JNDI 이름 필드에서 자원 이름을 입력합니다. 예를 들면 다음과 같습니다.

   `jms/Queue`

   JMS 자원에 대해 이름 지정 하위 컨텍스트 접두어 `jms/`를 사용하는 것이 좋습니다.

5. 유형 드롭다운 목록에서 javax.jms.Topic 또는 javax.jms.Queue를 선택합니다.
6. 런타임에서 자원을 활성화하려면 사용 가능 확인란을 선택합니다.
7. 추가 등록 정보 영역에서 등록 정보 값을 제공합니다. 다음 표에는 사용 가능한 등록 정보가 나열됩니다.

표 8-2 JMS 대상 자원의 추가 등록 정보

등록 정보 이름	설명
Name	(필수) 자원이 참조하는 물리적 대상 이름입니다.
Description	물리적 대상에 대한 설명입니다.

8. 대상 영역에서 다음 작업을 수행합니다.
   a. 사용 가능한 열에서 배포할 자원을 사용하는 응용 프로그램에 해당하는 대상을 선택합니다. 사용 가능한 대상은 기본 서버 인스턴스 server뿐만 아니라 사용 가능한 클러스터와 서버 인스턴스를 포함합니다.
   b. 추가를 눌러 대상을 선택한 열로 이동합니다.
9. 확인을 누릅니다.

해당 asadmin 명령: create-jms-resource

JMS 대상 자원 편집

JMS 대상 자원을 편집하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JMS 자원 노드를 확장합니다.
2. 대상 자원 노드를 확장합니다.
3. 편집할 대상 자원을 선택합니다.
4. JMS 대상 자원 편집 페이지에서 다음 작업을 수행할 수 있습니다.
   - 자원 유형을 변경합니다.
   - 설명 필드의 텍스트를 수정합니다.
   - 자원을 활성화하거나 비활성화하려면 사용 가능 확인란을 선택하거나 선택 해제합니다.

- Name 또는 Description 등록 정보를 추가, 제거 또는 수정합니다.

5. 저장을 눌러 변경 사항을 저장합니다.
6. 필요에 따라 대상 탭을 눌러 JMS 대상 자원 대상 페이지로 이동합니다. 이 페이지에서 다음 작업을 수행합니다.
   a. 대상 관리를 눌러 자원 대상 관리 페이지를 엽니다.

      이 페이지에서 사용 가능한 열과 선택한 열에 대상을 이동합니다. 선택한 열에 배포할 자원을 사용하는 응용 프로그램에 해당하는 대상을 넣어야 합니다. 사용 가능한 대상은 기본 서버 인스턴스 server뿐만 아니라 사용 가능한 클러스터와 서버 인스턴스를 포함합니다. 확인을 눌러 변경 사항을 저장합니다.

   b. 대상에 대한 확인란을 선택한 다음, 활성화를 눌러 해당 대상의 자원을 활성화하거나, 비활성화를 눌러 해당 대상의 자원을 비활성화합니다.

JMS 대상 자원 삭제

JMS 대상 자원을 삭제하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JMS 자원 노드를 확장합니다.
2. 대상 자원 노드를 선택합니다.
3. JMS 대상 자원 페이지에서 삭제할 대상 자원 이름 옆에 있는 확인란을 선택합니다.
4. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-jms-resource`

JMS 물리적 대상을 위한 관리 콘솔 작업

JMS 물리적 대상 만들기

작업 목적의 경우 항상 물리적인 대상을 만듭니다. 그러나 개발 및 테스트 단계 중에는 이 단계가 필요하지 않습니다. 응용 프로그램에서 처음 대상 자원에 액세스할 때 대상 자원의 Name 등록 정보에서 지정한 물리적 대상을 Message Queue에서 자동으로 만듭니다. 물리적 대상은 임시적이며 Message Queue 구성 등록 정보에서 지정한 기간 후에는 만료됩니다.

JMS 물리적 대상을 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장한 다음, Java Message Service 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. `default-config` 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. 물리적 대상 노드를 선택합니다.
4. 물리적 대상 페이지에서 새로 만들기를 누릅니다. 물리적 대상 만들기 페이지가 표시됩니다.
5. 물리적 대상 이름 필드에서 대상 이름(예: `PhysicalQueue`)을 입력합니다.
6. 유형 드롭다운 목록에서 `topic` 또는 `queue`를 선택합니다.
7. 추가 등록 정보 영역에서 등록 정보 추가를 눌러 등록 정보를 추가합니다. 다음 표에서는 현재 사용 가능한 등록 정보를 나열합니다.

표 8-3 JMS 물리적 대상에 대한 추가 등록 정보

등록 정보 이름	설명
`maxNumActiveConsumers`	대기열 대상의 로드 균형 조정된 전달에서 활성화할 수 있는 최대 사용자 수입니다. -1 값은 수에 제한이 없음을 의미합니다. 독립 실행형 서버 인스턴스에 대해 대상을 만든 경우 기본값은 1이고, 클러스터에 대해 대상을 만든 경우 -1입니다.

이 등록 정보 값을 수정하거나 다른 물리적 대상 등록 정보를 지정하려면 *install_dir*`/imq/bin/imqcmd` 명령을 사용합니다. 자세한 내용은 *Sun Java System Message Queue 관리 설명서*를 참조하십시오.

8. 확인을 누릅니다.

물리적 대상 페이지에서는 만료되거나 전달할 수 없는 메시지가 리디렉션되는 시스템 대상(mq.sys.dmq라는 이름의 대기열)을 보여줍니다. 이 대상에 대한 대상 자원, 사용자 및 브라우저를 만들 수 있습니다. 그러나 이 대상을 삭제하거나 이 대상에 메시지를 보낼 수 없습니다.

해당 asadmin 명령: create-jmsdest

JMS 물리적 대상 삭제

JMS 물리적 대상을 삭제하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장한 다음, Java Message Service 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 물리적 대상 노드를 선택합니다.
4. 물리적 대상 페이지에서 삭제할 대상 이름 옆에 있는 확인란을 선택합니다.
5. 삭제를 누릅니다.

시스템 대상 mq.sys.dmq를 삭제할 경우 오류 메시지가 표시됩니다.

해당 asadmin 명령: delete-jmsdest

JMS 공급자를 위한 관리 콘솔 작업

- JMS 공급자를 위한 일반 등록 정보 구성
- JMS 호스트 만들기
- JMS 호스트 편집
- JMS 호스트 삭제

JMS 공급자를 위한 일반 등록 정보 구성

JMS 서비스 페이지를 사용하여 모든 JMS 연결에서 사용하는 등록 정보를 구성합니다. 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. `default-config` 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. JMS(Java Message Service) 노드를 선택하여 JMS 서비스 페이지를 엽니다.
4. 시작 시간 초과 필드의 값을 편집하여 시작을 중단하기 전에 Application Server에서 JMS 서비스가 시작할 때까지 대기하는 시간을 변경합니다. 느리거나 오버로드된 시스템에서는 기본값(60)에서 값을 늘립니다.
5. 유형 드롭다운 목록에서
   - 로컬 호스트에서 JMS 서비스에 액세스하려면 `LOCAL`(`server-config` 구성에 대한 기본값)을 선택합니다. Application Server에서 JMS 서비스를 시작하고 관리합니다.
   - 다른 시스템이나 클러스터에 있는 JMS 서비스에 액세스하려면 `REMOTE`(`default-config` 구성에 대한 기본값)를 선택합니다. `REMOTE`를 선택할 경우 다음에 해당 서버를 시작할 때 Application Server에서 JMS 서비스를 시작하지 않습니다. 대신 Message Queue를 통해 JMS 서비스를 시작하거나 관리하므로 Message Queue 브로커를 별도로 시작해야 합니다. 브로커 시작에 대한 자세한 내용은 *Sun Java System Message Queue 관리 설명서*를 참조하십시오. 이 값을 선택하고 원격 호스트를 사용할 경우 174페이지의 "JMS 호스트 편집"의 지침에 따라 원격 호스트 이름을 지정합니다.
6. 시작 인수 필드에서 JMS 서비스 시작을 사용자 정의하는 인수를 입력합니다. *install_dir*`/imq/bin/imqbrokerd` 명령을 통해 사용 가능한 인수를 사용합니다.
7. 연결이 끊어진 경우 JMS 서비스에서 메시지 서버 또는 AddressList의 주소 목록에 다시 연결할지 여부를 지정하려면 다시 연결 확인란을 사용합니다.

   기본적으로 다시 연결이 활성화되어 있습니다.

8. 다시 연결 간격 필드에서 다시 연결을 시도하는 간격(초)을 입력합니다. AddressList의 각 주소와 목록의 후속 주소에 대한 시도에 이 값이 적용됩니다. 이 시간 간격이 너무 짧을 경우 브로커에게 복구할 시간이 없습니다. 너무 길 경우에는 지연이 지나치게 길게 느껴질 수 있습니다.

   기본값은 60초입니다.

9. 클라이언트 런타임에서 목록의 다음 주소를 사용하기 전에 AddressList의 각 주소에 대해 연결(또는 재연결)을 시도하는 횟수를 다시 연결 시도 필드에 입력합니다. -1 값은 재연결 시도 횟수에 제한이 없음을 나타냅니다. 클라이언트 런타임은 성공할 때까지 첫 번째 주소에 연결을 시도합니다.

   기본값은 3입니다.

10. 기본 JMS 호스트 드롭다운 목록에서 호스트를 선택합니다. 기본값은 `default_JMS_host`입니다.

11. 주소 목록 동작 드롭다운 목록에서 AddressList의 주소 순서대로(`priority`) 다시 연결을 시도할지 아니면 임의의 순서대로(`random`) 다시 연결을 시도할지 선택합니다.

    `priority`는 재연결 시 항상 AddressList의 첫 번째 서버 주소에 연결하고 첫 번째 브로커를 사용할 수 없는 경우에만 다른 서버 주소를 사용한다는 것을 의미합니다.

    동일한 연결 팩토리를 사용하여 연결을 시도하는 클라이언트가 많을 경우 `random`을 지정하여 클라이언트가 동일한 주소에 모두 연결되지 못하도록 합니다.

    기본값은 `random`입니다.

12. 주소 목록 반복 필드에서 연결하거나 다시 연결할 때 JMS 서비스가 AddressList를 반복하는 횟수를 입력합니다. -1 값은 횟수에 제한이 없음을 나타냅니다.

    기본값은 3입니다.

13. MQ 방법 및 MQ 서비스 필드에서 기본이 아닌 방법이나 서비스를 사용할 경우 Message Queue 주소 방법 이름과 MQ 연결 서비스 이름을 입력합니다. 메시지 서비스 주소에 대한 전체 구문은 다음과 같습니다.

scheme:*//address_syntax*

여기에서 *scheme*과 *address_syntax*는 아래 표에서 자세히 설명합니다.

다음 표의 처음 두 열에 표시된 값이 MQ 방법과 MQ 서비스입니다.

표 8-4 메시지 서버 주소 방법 및 구문

방법 이름	연결 서비스	설명	주소 구문
mq	jms 및 ssljms	MQ 클라이언트 런타임은 지정한 호스트와 포트에서 MQ Port Mapper에 연결합니다. 포트 매퍼는 동적으로 설정된 연결 서비스 포트 목록을 반환하고, MQ 클라이언트 런타임에서 지정한 연결 서비스를 호스트하는 포트에 연결합니다.	[*hostname*][:*port*][/*serviceName*] 기본값: *hostName* = localhost *port* = 7676 *serviceName* = jms 기본값은 jms 연결 서비스에만 적용됩니다. ssljms 연결 서비스의 경우 모든 변수를 지정해야 합니다. 예: mq:MyHost:7677/ssljms
mqtcp	jms	MQ 클라이언트 런타임은 MQ Port Mapper를 무시하고 지정된 호스트와 포트에 TCP 연결을 하여 연결을 설정합니다.	*hostName*:*port*/jms 예: mqtcp:localhost:7676/jms
mqssl	ssljms	MQ 클라이언트 런타임은 MQ Port Mapper를 무시하고 지정된 호스트와 포트에 SSL 연결을 하여 연결을 설정합니다.	*hostName*:*port*/ssljms 예: mqssl:localhost:7676/ssljms
http	httpjms	MQ 클라이언트 런타임은 지정된 URL에서 MQ 터널 서블릿에 HTTP 연결을 합니다. MQ *관리 설명서*에서 설명한 대로 HTTP 터널 서블릿에 액세스하려면 브로커를 구성해야 합니다.	*hostName*:*port*/*contextRoot*/tunnel 여러 브로커 인스턴스에서 동일한 터널 서블릿을 사용할 경우 임의로 선택한 브로커 인스턴스 대신 특정한 브로커 인스턴스에 연결하는 구문은 다음과 같습니다. http://*hostName*:*port*/*contextRoot*/tunnel?*serverName=hostName*:*instanceName*

표 8-4 메시지 서버 주소 방법 및 구문 *(계속)*

방법 이름	연결 서비스	설명	주소 구문
`https`	`httpsjms`	MQ 클라이언트 런타임은 지정된 MQ 터널 서블릿 URL에 보안된 HTTPS 연결을 합니다. (MQ *관리 설명서*에서 설명한 대로 HTTP 터널 서블릿에 액세스하려면 브로커를 구성해야 합니다.)	*hostName*`:`*port*`/`*contextRoot*`/tunnel` 여러 브로커 인스턴스에서 동일한 터널 서블릿을 사용할 경우 임의로 선택한 브로커 인스턴스 대신 특정한 브로커 인스턴스에 연결하는 구문은 다음과 같습니다. `http://`*hostName*`:`*port*`/`*contextRoot*`/tunnel?serverName=`*hostName*`:`*instanceName*

14. 추가 등록 정보 영역에서 등록 정보 추가를 눌러 등록 정보를 추가합니다. 다음 표에서는 사용 가능한 Message Queue 브로커 구성 등록 정보를 나열합니다.

표 8-5 JMS 공급자에 대한 추가 등록 정보

등록 정보 이름	설명
`instance-name`	전체 Sun Java System Message Queue 브로커 인스턴스 이름을 지정합니다. 기본값은 `imqbroker`입니다.
`instance-name-suffix`	전체 Sun Java System Message Queue 브로커 인스턴스 이름에 추가할 접미어를 지정합니다. 접미어와 인스턴스 이름은 밑줄 문자(_)로 구분합니다. 예를 들어, 인스턴스 이름이 `imqbroker`일 경우 `xyz` 접미어를 추가하면 인스턴스 이름이 `imqbroker_xyz`로 바뀝니다.
`append-version`	`true`일 경우 전체 Sun Java System Message Queue 브로커 인스턴스 이름 뒤에 주 버전과 부 버전 번호가 추가되며 버전 번호 앞에는 밑줄 문자(_)가 옵니다. 예를 들어, 인스턴스 이름이 `imqbroker`일 경우 버전 번호를 추가하면 인스턴스 이름이 `imqbroker_8_0`으로 바뀝니다. 기본값은 `false`입니다.

15. 저장을 눌러 변경 사항을 저장하거나, 기본값 로드를 눌러 서비스에 대한 기본값을 복구합니다.

JMS 서비스가 시작되어 실행 중인지 확인하려면 핑을 누릅니다. 실행 중이면 "Ping succeeded: JMS service is running" 메시지가 표시됩니다.

공급자와 호스트를 원격 시스템으로 변경하면 모든 JMS 응용 프로그램이 원격 서버에서 실행됩니다. 로컬 서버와 하나 이상의 원격 서버를 모두 사용하려면 원격 서버에 액세스하는 연결을 만드는 AddressList 등록 정보를 사용하여 연결 팩토리 자원을 만듭니다.

JMS 서비스 구성에 대한 자세한 내용은 *Sun Java System Application Server Developer's Guide*를 참조하십시오.

해당 asadmin 명령: jms-ping

JMS 호스트 만들기

JMS 호스트를 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. Java Message Service 노드를 확장합니다.
4. JMS 호스트 노드를 선택합니다.
5. JMS 호스트 페이지에서 새로 만들기를 누릅니다. JMS 호스트 만들기 페이지가 표시됩니다.
6. 이름 필드에서 호스트 이름을 입력합니다. 예를 들면 다음과 같습니다.

   NewJmsHost

7. 호스트 필드에서 JMS 호스트가 실행되는 시스템의 이름이나 IP(인터넷 프로토콜) 주소를 입력합니다(localhost 또는 로컬 또는 원격 시스템의 이름).
8. 포트 필드에서 JMS 서비스의 포트 번호를 입력합니다. 사용할 JMS 서비스를 기본 포트가 아닌 포트에서 실행할 경우에만 이 필드를 변경합니다. 기본 포트는 7676입니다.
9. 관리자 이름 및 관리 비밀번호 필드에서 MQ 브로커 아이디와 비밀번호를 입력합니다. 이는 Application Server 아이디 및 비밀번호와 다릅니다.
   install_dir/imq/bin/imqusermgr 명령을 사용하여 MQ 브로커 값을 변경한 경우에만 이 필드를 편집합니다. 기본값은 admin과 admin입니다.

10. 확인을 누릅니다.

해당 asadmin 명령: create-jms-host

JMS 호스트 편집

JMS 호스트를 편집하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. Java Message Service 노드를 확장합니다.
4. JMS 호스트 노드를 선택합니다.
5. JMS 호스트 페이지에서 편집할 호스트를 선택합니다.
6. JMS 호스트 편집 페이지에서 다음 작업을 수행할 수 있습니다.
   - 호스트 필드의 호스트 이름이나 IP(인터넷 프로토콜) 주소를 변경합니다.
   - 포트 필드에서 JMS 서비스의 포트 번호를 변경합니다.
   - 관리자 이름 필드와 관리 비밀번호 필드의 값을 변경합니다.
7. 저장을 눌러 변경 사항을 저장하거나, 기본값 로드를 눌러 호스트에 대한 기본값을 복구합니다.

JMS 호스트 삭제

JMS 호스트를 삭제하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.

3. Java Message Service 노드를 확장합니다.
4. JMS 호스트 노드를 선택합니다.
5. JMS 호스트 페이지에서 삭제할 호스트 이름 옆에 있는 확인란을 선택합니다.
6. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-jms-host`

JavaMail 자원 구성

이 장에서는 JavaMail API를 사용하는 응용 프로그램에 맞게 자원을 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

JavaMail 정보

JavaMail API

JavaMail API는 메일 시스템을 모델링하는 추상 API 집합입니다. API는 메일 및 메시징 응용 프로그램을 빌드할 수 있는 플랫폼과 프로토콜에 독립적인 프레임워크를 제공합니다. JavaMail API는 전자 메일을 읽고 보내는 기능을 제공합니다. 서비스 공급자가 특정 프로토콜을 구현합니다.

JavaMail API는 Java 플랫폼 선택적 패키지로 구현되고 J2EE 플랫폼의 일부로 사용할 수도 있습니다.

Application Server에는 JavaMail API와 함께 JavaMail 서비스 공급자가 포함되어 있어 응용 프로그램 구성 요소가 인터넷을 통해 전자 메일 알림을 보내고 IMAP 및 POP3 메일 서버에서 전자 메일을 읽을 수 있습니다.

JavaMail API에 대한 자세한 내용은 JavaMail 웹 사이트 (`http://java.sun.com/products/javamail/`)를 참조하십시오.

JavaMail에 대한 관리 콘솔 작업

JavaMail 세션 만들기

JavaMail 세션을 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JavaMail 세션 노드를 선택합니다.
2. JavaMail 세션 페이지에서 새로 만들기를 누릅니다. JavaMail 세션 만들기 페이지가 표시됩니다.
3. JNDI 이름 필드에 세션 이름을 입력합니다. 예를 들면 다음과 같습니다.

   `mail/MySession`

   JavaMail 자원에 `mail/` 이름 지정 하위 컨텍스트 접두어를 사용하는 것이 좋습니다.
4. 메일 호스트 필드에 기본 메일 서버의 호스트 이름을 입력합니다. 프로토콜 관련 호스트 등록 정보를 제공하지 않으면 저장소 및 전송 객체의 연결 메소드에서 이 값을 사용합니다. 이름을 실제 호스트 이름으로 확인할 수 있어야 합니다.
5. 기본 사용자 필드에서 메일 서버에 연결할 때 제공할 아이디를 입력합니다. 프로토콜 관련 아이디 등록 정보를 제공하지 않으면 저장소 및 전송 객체의 연결 메소드에서 이 값을 사용합니다.
6. 기본 반송 주소 필드에서 기본 사용자의 전자 메일 주소를 *username*`@`*host*`.`*domain* 양식으로 입력합니다.
7. 이 때 메일 세션을 활성화하지 않으려면 사용 가능 확인란을 선택 해제합니다.
8. 기본이 아닌 저장소 또는 전송 프로토콜을 사용하도록 Application Server의 메일 공급자를 다시 구성한 경우에만 고급 영역에서 필드 값을 변경합니다. 기본적으로 저장소 프로토콜은 `imap`, 저장소 프로토콜 클래스는 `com.sun.mail.imap.IMAPStore`, 전송 프로토콜은 `smtp`, 전송 프로토콜 클래스는 `com.sun.mail.smtp.SMTPTransport`입니다.

   프로토콜 추적을 포함하여 이 메일 세션에 대한 추가 디버깅 출력을 사용 가능하게 하려면 디버그 확인란을 선택합니다. JavaMail 로그 수준을 `FINE` 이상으로 설정한 경우 디버깅 출력이 생성되어 시스템 로그 파일에 포함됩니다. 로그 수준 설정에 대한 자세한 내용은 335페이지의 "로그 수준 구성"을 참조하십시오.

9. 추가 등록 정보 영역에서 프로토콜 관련 호스트나 아이디 등록 정보 같이 응용 프로그램에서 요구하는 등록 정보를 추가하려면 등록 정보 추가를 누릅니다. JavaMail API 설명서에 사용 가능한 등록 정보 (`http://java.sun.com/products/javamail/javadocs/index.html`)가 나열되어 있습니다.
10. 대상 영역에서 다음 작업을 수행합니다.
    a. 사용 가능한 열에서 배포할 자원을 사용하는 응용 프로그램에 해당하는 대상을 선택합니다. 사용 가능한 대상은 기본 서버 인스턴스 `server`뿐만 아니라 사용 가능한 클러스터와 서버 인스턴스를 포함합니다.
    b. 추가를 눌러 대상을 선택한 열로 이동합니다.
11. 확인을 눌러 세션을 저장합니다.

해당 `asadmin` 명령: `create-javamail-resource`

JavaMail 세션 편집

JavaMail 세션을 편집하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JavaMail 세션 노드를 선택합니다.
2. JavaMail 세션 페이지에서 편집할 세션을 선택합니다.
3. JavaMail 세션 편집 페이지에서 다음 작업을 수행할 수 있습니다.
   - 메일 호스트, 기본 사용자, 기본 반송 주소 및 설명 필드의 값을 수정합니다.
   - 자원을 활성화하거나 비활성화하려면 사용 가능 확인란을 선택하거나 선택 해제합니다.
   - 고급 필드의 값을 수정합니다.
   - 등록 정보를 추가, 제거 또는 수정합니다.
4. JavaMail 세션 대상 페이지로 이동하려면 대상 탭을 누릅니다. 이 페이지에서 다음 작업을 수행합니다.
   a. 대상 관리를 눌러 자원 대상 관리 페이지를 엽니다.

      이 페이지에서 사용 가능한 열과 선택한 열에 대상을 이동합니다. 선택한 열에 배포할 자원을 사용하는 응용 프로그램에 해당하는 대상을 넣어야 합니다. 사용 가능한 대상은 기본 서버 인스턴스 `server`뿐만 아니라 사용 가능한 클러스터와 서버 인스턴스를 포함합니다. 확인을 눌러 변경 사항을 저장합니다.
   b. 대상에 대한 확인란을 선택한 다음, 활성화를 눌러 해당 대상의 자원을 활성화하거나, 비활성화를 눌러 해당 대상의 자원을 비활성화합니다.

5. 저장을 눌러 변경 사항을 저장하거나, 기본값 로드를 눌러 메일 세션의 기본값을 복구합니다.

JavaMail 세션 삭제

JavaMail 세션을 삭제하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 JavaMail 세션 노드를 선택합니다.
2. JavaMail 세션 페이지에서 삭제할 세션 이름 옆에 있는 확인란을 선택합니다.
3. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-javamail-resource`

JNDI 자원

이 장에서는 관리 콘솔을 사용하여 JNDI 자원을 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- JNDI(Java Naming and Directory Interface) 정보
- 사용자 정의 자원 정보
- 외부 JNDI 저장소 및 자원 정보

JNDI(Java Naming and Directory Interface) 정보

이 절에서는 JNDI(Java Naming and Directory Interface)에 대해 설명합니다. JNDI는 다른 종류의 이름 지정 및 디렉토리 서비스에 액세스하는 데 필요한 API(Application Programming Interface)입니다. J2EE 구성 요소는 JNDI 조회 메소드를 호출하여 객체를 찾습니다.

이 절에서는 다음 항목에 대해 설명합니다.

- JNDI 이름 및 자원
- J2EE 이름 지정 서비스
- 이름 지정 참조 및 바인딩 정보

JNDI 이름 및 자원

JNDI는 Java Naming and Directory Interface API의 머리글자입니다. 응용 프로그램은 이 API를 호출하여 자원과 다른 프로그램 객체를 찾습니다. 자원은 데이터베이스 서버나 메시징 시스템 같은 시스템과의 연결을 제공하는 프로그램 객체입니다. (JDBC 자원을 데이터 소스라고도 합니다.) 모든 자원 객체는 고유하고 사용자에게 친숙한 JNDI 이름으로 식별됩니다. 자원 객체와 JNDI 이름은 Application Server에 포함된 이름 지정 및 디렉토리 서비스에 의해 함께 바인딩됩니다. 자원을 새로 만들려면 새로운 이름 객체 바인딩을 JNDI에 입력합니다.

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 요소가 JNDI API를 통해 이 서비스에 액세스하기 때문에 대개 객체는 해당 JNDI 이름을 사용합니다. 예를 들어, Pointbase 데이터베이스의 JNDI 이름은 `jdbc/Pointbase`입니다. Sun Java System Application Server를 시작하면 구성 파일에서 정보를 읽어서 JNDI 데이터베이스 이름을 이름 공간에 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 기법입니다. 응용 프로그램 구성 요소 환경을 사용하면 응용 프로그램 구성 요소의 소스 코드에 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. 배포자가 환경 항목의 값을 설정 및 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.

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 할 경우 이름을 변경할 필요가 없습니다. 또한 이러한 융통성으로 인해 기존의 구성 요소로부터 J2EE 응용 프로그램을 조립하기가 쉽습니다.

표 10-1에서는 Sun Java System Application Server에서 사용한 J2EE 자원에 대한 JNDI 조회 및 연관된 참조를 나열합니다.

표 10-1 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

표 10-1 JNDI 조회 및 관련 참조

JNDI 조회 이름	관련 참조
java:comp/env/jms	JMS 연결 팩토리 및 대상
java:comp/ORB	응용 프로그램 구성 요소 간에 공유되는 ORB 인스턴스

사용자 정의 자원 정보

사용자 정의 자원 사용

사용자 정의 자원은 로컬 JNDI 저장소에 액세스하고 외부 자원은 외부 JNDI 저장소에 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 자원에 액세스하는 방법에 대해 설명합니다.

Application Server 내에서 list-jndi-entities뿐만 아니라 자원을 작성, 삭제 및 나열할 수 있습니다.

사용자 정의 자원 만들기

사용자 정의 자원을 만들려면 다음 작업을 수행합니다.

1. 관리 콘솔의 왼쪽 창에서 수정할 JNDI 구성용 Sun Java System Application Server 인스턴스를 엽니다.
2. JNDI 탭을 열고 사용자 정의 자원을 누릅니다. 사용자 정의 자원이 이미 만들어져 있으면 오른쪽 창에 나열됩니다. 새 사용자 정의 자원을 만들려면 새로 만들기를 누릅니다. JNDI 탭을 열고 새로 만들기를 누릅니다. 새로운 사용자 정의 자원을 추가하기 위한 페이지가 표시됩니다.

3. JNDI 이름 필드에서 자원에 액세스할 때 사용할 이름을 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 따릅니다.
5. 팩토리 클래스 필드에서 작성할 사용자 정의 자원의 팩토리 클래스 이름을 입력합니다. 팩토리 클래스는 팩토리 클래스에 대한 사용자 지정 이름입니다. 이 클래스는 `javax.naming.spi.ObjectFactory` 인터페이스를 구현합니다.
6. 설명 필드에서 작성할 자원에 대한 설명을 입력합니다. 이 설명은 문자열 값이고 최대 250자를 포함할 수 있습니다.
7. 추가 등록 정보 섹션에서 등록 정보 이름과 값을 추가합니다.
8. 사용자 정의 자원 사용 가능 확인란을 선택하여 사용자 정의 자원을 활성화합니다.
9. 사용자 정의 자원을 저장하려면 확인을 누릅니다.

클러스터나 독립 실행형 인스턴스에 사용자 정의 자원을 배포할 경우 대상 탭을 사용하여 대상을 관리할 수 있습니다. 사용자 정의 자원을 만든 후 탭이 표시됩니다. 대상 이름을 입력하고 확인을 눌러 대상을 설정합니다.

해당 asadmin 명령: create-custom-resource.

사용자 정의 자원 편집

사용자 정의 자원을 편집하려면 다음 작업을 수행합니다.

1. 관리 콘솔의 왼쪽 창에서 수정할 JNDI 구성용 Sun Java System Application Server 인스턴스를 엽니다.
2. JNDI를 열고 사용자 정의 자원을 선택합니다. 사용자 정의 자원이 이미 만들어져 있으면 오른쪽 창에 나열됩니다. 사용자 정의 자원을 편집하려면 오른쪽 창에서 파일 이름을 누릅니다.
3. 자원 유형 필드, 팩토리 클래스 필드 또는 설명 필드를 편집합니다.
4. 사용자 정의 자원을 활성화하려면 사용자 정의 자원 사용 확인란을 선택합니다.
5. 저장을 눌러 사용자 정의 자원의 변경 내용을 저장합니다.

사용자 정의 자원 삭제

사용자 정의 자원을 삭제하려면 다음 작업을 수행합니다.

1. 관리 콘솔의 왼쪽 창에서 JNDI 탭을 엽니다.

2. 사용자 정의 자원을 누릅니다. 사용자 정의 자원이 이미 만들어져 있으면 오른쪽 창에 나열됩니다. 사용자 정의 자원을 삭제하려면 삭제할 자원 이름 옆에 있는 확인란을 누릅니다.
3. 삭제를 누릅니다. 사용자 정의 자원이 삭제됩니다.

해당 asadmin 명령: delete-custom-resource.

사용자 정의 자원 나열

사용자 정의 자원을 나열하려면 `asadmin ist-custom-resources` 명령을 입력합니다. 예를 들어, 호스트(plum)의 사용자 정의 자원을 나열하려면 다음을 입력합니다.

```
$asadmin list-custom-resource --host plum target6
```

전체 컨텍스트를 보려면 `asadmin help list-custom-resources`를 입력합니다.

외부 JNDI 저장소 및 자원 정보

외부 JNDI 저장소 및 자원 사용

Sun Java System Application Server에서 실행되는 응용 프로그램의 경우 외부 JNDI 저장소에 저장된 자원에 액세스해야 하는 경우가 있습니다. 예를 들어, LDAP 서버에 일반 Java 객체를 Java 스키마별로 저장할 수 있습니다. 외부 JNDI 자원 요소를 사용하면 외부 자원 저장소를 구성할 수 있습니다. 외부 JNDI 팩토리는 `javax.naming.spi.InitialContextFactory` 인터페이스를 구현해야 합니다.

다음은 외부 JNDI 자원 사용 예입니다.

```
<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>
```

외부 자원 만들기

외부 자원을 만들려면 다음 작업을 수행합니다.

1. 관리 콘솔의 왼쪽 창에서 수정할 JNDI 구성용 Sun Java System Application Server 인스턴스를 엽니다.
2. JNDI를 열고 외부 자원을 선택합니다. 외부 자원이 이미 만들어져 있으면 오른쪽 창에 나열됩니다. 새 외부 자원을 만들려면 새로 만들기를 누릅니다.
3. 자원에 액세스하는 데 사용할 이름을 JNDI 이름 필드에 입력합니다. 이 이름이 JNDI 이름 지정 서비스에 등록됩니다.
4. 위의 예에서와 같이 자원 유형 필드에 전체 유형 정의를 입력합니다. 자원 유형 정의는 `xxx.xxx` 형식을 따릅니다.
5. 외부 저장소에서 조회할 JNDI 값을 JNDI 조회 필드에 입력합니다. 예를 들어, 외부 자원을 만들어 외부 저장소에 연결하고 Bean 클래스를 테스트할 경우 JNDI 조회는 `cn=`*`testmybean`*과 같습니다.

6. com.sun.jndi.ldap와 같은 JNDI 팩토리 클래스 외부 저장소를 팩토리 클래스 필드에 입력합니다. 이 클래스는 javax.naming.spi.ObjectFactory 인터페이스를 구현합니다.
7. 설명 필드에서 작성할 자원에 대한 설명을 입력합니다. 이 설명은 문자열 값이고 최대 250자를 포함할 수 있습니다.
8. 추가 등록 정보 섹션에서 등록 정보 이름과 값을 추가합니다.
9. 외부 자원 사용 가능 확인란을 선택하여 외부 자원을 활성화합니다.
10. 확인을 눌러 외부 자원을 저장합니다.

클러스터나 독립 실행형 인스턴스에 외부 자원을 배포한 경우 대상 탭을 사용하여 대상을 관리할 수 있습니다. 외부 자원을 만든 후 탭이 표시됩니다. 대상 이름을 입력하고 확인을 눌러 대상을 설정합니다.

해당 asadmin 명령: create-jndi-resource

외부 자원 편집

외부 자원을 편집하려면 다음 작업을 수행합니다.

1. 관리 콘솔의 왼쪽 창에서 수정할 JNDI 구성용 Sun Java System Application Server 인스턴스를 엽니다.
2. JNDI를 열고 외부 자원을 선택합니다. 외부 자원이 이미 만들어져 있으면 오른쪽 창에 나열됩니다. 외부 자원을 편집하려면 오른쪽 창에서 파일 이름을 누릅니다.
3. 자원 유형 필드, JNDI 조회 필드, 팩토리 클래스 필드 또는 설명 필드를 편집합니다.
4. 외부 자원을 활성화하려면 외부 자원 사용 확인란을 선택합니다.
5. 저장을 눌러 외부 자원의 변경 내용을 저장합니다.

외부 자원 삭제

외부 자원을 삭제하려면 다음 작업을 수행합니다.

1. 관리 콘솔의 왼쪽 창에서 JNDI 탭을 엽니다.
2. 외부 자원을 누릅니다. 외부 자원이 이미 만들어져 있으면 오른쪽 창에 나열됩니다. 외부 자원을 삭제하려면 삭제할 자원 이름 옆에 있는 확인란을 누릅니다.

3. 삭제를 누릅니다. 외부 자원이 삭제됩니다.

해당 asadmin 명령: delete-jndi-resource

외부 자원 나열

외부 자원을 나열하려면 `asadmin list-jndi-resources` 명령을 입력하고 jndi 이름을 지정합니다. 예를 들어, 외부 자원을 나열하려면 다음을 입력합니다.

```
$asadmin list-jndi-resources -- target plum jndi_name_test
```

전체 컨텍스트를 보려면 `asadmin help list-jndi-resources`를 입력합니다.

커넥터 자원

이 장에서는 EIS(Enterprise Information System)에 액세스할 때 사용하는 커넥터 구성 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

커넥터 정보

커넥터 모듈, 연결 풀 및 자원

자원 어댑터라고도 하는 커넥터 모듈은 응용 프로그램이 EIS(Enterprise Information System)와 상호 작용할 수 있게 해주는 J2EE 구성 요소입니다. EIS 소프트웨어에는 ERP(Enterprise Resource Planning), 메인프레임 트랜잭션 처리, 비관계형 데이터베이스, 기타 등과 같은 다양한 유형의 시스템이 포함됩니다. 다른 J2EE 모듈과 마찬가지로 커넥터 모듈을 설치하려면 배포해야 합니다.

커넥터 연결 풀은 특정 EIS에 대해 다시 사용할 수 있는 연결 그룹입니다. 커넥터 연결 풀을 만들려면 해당 풀과 연관된 커넥터 모듈(자원 어댑터)을 지정합니다.

커넥터 자원은 응용 프로그램에 EIS와의 연결을 제공하는 프로그램 객체입니다. 커넥터 자원을 만들려면 해당 JNDI 이름 및 연관된 연결 풀을 지정합니다. 여러 커넥터 자원에서 단일 연결 풀을 지정할 수 있습니다. 응용 프로그램은 해당 JNDI 이름을 조회하여 자원을 찾습니다. (JNDI에 대한 자세한 내용은 섹션 JNDI 이름과 자원을 참조하십시오.) EIS에 대한 커넥터 자원의 JNDI 이름은 대개 `java:comp/env/`*eis-specific* 하위 컨텍스트로 되어 있습니다.

Application Server는 커넥터 모듈(자원 어댑터)을 사용하여 JMS를 구현합니다. JMS 자원과 커넥터 자원 간의 관계를 참조하십시오.

커넥터 연결 풀 작업

- EIS 액세스를 설정하는 일반 단계
- 커넥터 연결 풀 만들기
- 커넥터 연결 풀 편집
- 커넥터 연결 풀 삭제

EIS 액세스를 설정하는 일반 단계

1. 커넥터를 배포(설치)합니다. 커넥터 모듈 배포를 참조하십시오.
2. 커넥터에 대한 연결 풀을 만듭니다. 커넥터 연결 풀 만들기를 참조하십시오.
3. 연결 풀과 연관된 커넥터 자원을 만듭니다. 커넥터 자원 만들기를 참조하십시오.

커넥터 연결 풀 만들기

풀을 만들기 전에 풀과 연관된 커넥터 모듈(자원 어댑터)을 배포합니다. 새로운 풀에 지정된 값은 배포된 커넥터 모듈에 따라 다릅니다.

커넥터 연결 풀을 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 커넥터 노드를 확장합니다.
2. 커넥터 연결 풀 노드를 선택합니다.
3. 커넥터 연결 풀 페이지에서 새로 만들기를 누릅니다.
4. 커넥터 연결 풀 만들기 첫 번째 페이지에서 다음 설정을 지정합니다.

a. 이름 필드에서 풀의 논리 이름을 입력합니다.

   커넥터 자원을 만들 때 이 이름을 지정합니다.

b. 자원 어댑터 콤보 상자에서 항목을 선택합니다.

   콤보 상자에는 배포된 자원 어댑터(커넥터 모듈) 목록이 표시됩니다.

5. 다음을 누릅니다.
6. 커넥터 연결 풀 만들기 두 번째 페이지의 연결 정의 콤보 상자에서 원하는 값을 선택합니다.

   콤보 상자의 선택 항목은 자원 어댑터에 따라 다릅니다. 일반적으로 EIS에 대한 연결을 가져오기 위한 팩토리 인스턴스인 `ConnectionFactory` 유형을 지정합니다.
7. 다음을 누릅니다.
8. 커넥터 연결 풀 만들기 세 번째이자 마지막 페이지에서 다음 작업을 수행합니다.

a. 일반 설정 섹션에서 값이 올바른지 확인합니다.

b. 풀 설정 섹션의 필드의 경우 기본값을 보존할 수 있습니다.

   나중에 이 설정을 변경할 수 있습니다. 커넥터 연결 풀 편집을 참조하십시오.

c. 추가 등록 정보 테이블에서 필수 등록 정보를 추가합니다.

   이전 커넥터 연결 풀 만들기 페이지의 연결 정의 콤보 상자에서 클래스를 선택했습니다. 이 클래스가 서버의 클래스 경로에 있을 경우 추가 등록 정보 테이블에 기본 등록 정보가 표시됩니다.

9. 마침을 누릅니다.

해당 `asadmin` 명령: `create-connector-connection-pool`

- 커넥터 모듈, 연결 풀 및 자원
- 커넥터 모듈 배포

커넥터 연결 풀 편집

커넥터 연결 풀 편집 페이지에서는 풀 설정과 추가 등록 정보를 변경할 수 있습니다.

커넥터 연결 풀 편집 페이지에 액세스하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 커넥터 노드를 확장합니다.
2. 커넥터 연결 풀을 확장합니다.
3. 편집할 풀의 노드를 선택합니다.
4. 커넥터 연결 풀 편집 페이지에서 풀의 연결 수를 제어하는 설정을 변경할 수 있습니다. 표 11-1을 참조하십시오.

표 11-1 커넥터 연결 풀의 풀 설정

매개 변수	설명
초기 및 최소 풀 크기	풀의 최소 연결 수입니다. 풀 값에 따라 풀을 먼저 작성하거나 응용 프로그램 서버를 시작할 때 풀에 있는 연결 개수도 결정합니다.
최대 풀 크기	풀의 최대 연결 수입니다.
풀 크기 조정 개수	풀이 최소 풀 크기로 줄어들 경우 일괄적으로 크기가 조정됩니다. 이 값은 일괄적으로 처리할 연결 수를 지정합니다. 이 값을 너무 크게 하면 연결 재순환이 지연되고, 너무 작게 하면 효율성이 떨어집니다.
유휴 시간 초과	풀에서 연결이 유휴 상태로 있을 수 있는 최대 시간(초)입니다. 이 시간이 만료되면 풀에서 연결이 제거됩니다.
최대 대기 시간	연결을 요청한 응용 프로그램이 연결될 때까지 기다리는 시간으로 이 시간이 지나면 연결 시간 초과가 됩니다. 기본 대기 시간이 길기 때문에 응용 프로그램이 무기한 중지될 수 있습니다.
실패 시	모든 연결 닫기 확인란을 선택한 경우 단일 연결이 실패하면 응용 프로그램 서버는 풀의 모든 연결을 닫은 다음 다시 연결합니다. 그 확인란을 선택하지 않은 경우 개별 연결을 사용한 경우에만 연결이 다시 설정됩니다.

표 11-1 커넥터 연결 풀의 풀 설정

매개 변수	설명
트랜잭션 지원	트랜잭션 지원 목록을 사용하여 연결 풀에 대한 트랜잭션 지원 유형을 선택합니다. 선택한 트랜잭션 지원은 이 연결 풀과 연관된 자원 어댑터의 트랜잭션 지원 속성을 역호환 방식으로 대체합니다. 즉 자원 어댑터에 지정한 수준보다 낮은 트랜잭션 수준을 지원하거나 자원 어댑터에서 지정한 것과 같은 트랜잭션 수준을 지원할 수 있지만, 더 높은 수준은 지정할 수 없습니다. 트랜잭션 지원 옵션에는 다음이 포함됩니다. 트랜잭션 지원 메뉴에서 없음을 선택하면 자원 어댑터에서 자원 관리자 로컬이나 JTA 트랜잭션을 지원하지 않고 XAResource 또는 LocalTransaction 인터페이스를 구현하지 않음을 나타냅니다. 로컬 트랜잭션 지원은 LocalTransaction 인터페이스를 구현하여 자원 어댑터가 로컬 트랜잭션을 지원함을 의미합니다. 로컬 트랜잭션은 자원 관리자에 내부로 관리되고 외부 트랜잭션 관리자를 포함시키지 않습니다. XA 트랜잭션 지원은 LocalTransaction 및 XAResource 인터페이스를 구현하여 자원 어댑터에서 자원 관리자 로컬 및 JTA 트랜잭션을 지원함을 의미합니다. XA 트랜잭션은 자원 관리자 외부에 있는 트랜잭션 관리자가 제어 및 조정합니다. 로컬 트랜잭션은 자원 관리자 내부로 관리되고 외부 트랜잭션 관리자를 포함시키지 않습니다.

5. 추가 등록 정보 테이블에서 이름-값 쌍을 지정합니다. 지정된 등록 정보는 이 풀에서 사용한 자원 어댑터에 따라 다릅니다. 이 테이블을 사용하여 배포자가 지정한 이름-값 쌍을 사용하여 자원-어댑터 공급업체가 정의한 등록 정보로 기본값을 대체할 수 있습니다.
6. 보안 맵 탭 창에서 연결 풀의 보안 맵을 만들거나 수정합니다. 보안 맵 작성 방법에 대한 자세한 내용은 "보안 맵 정보"를 참조하십시오.
7. 저장을 누릅니다.

커넥터 연결 풀 삭제

1. 트리 구성 요소에서 자원 노드를 확장한 다음 커넥터 노드를 확장합니다.
2. 커넥터 연결 풀 노드를 선택합니다.
3. 커넥터 연결 풀 페이지에서 삭제할 풀의 확인란을 선택합니다.
4. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-connector-connection-pool`

커넥터 자원 작업

커넥터 자원 만들기

커넥터 자원(데이터 소스)은 응용 프로그램에 EIS 연결을 제공합니다. 커넥터 자원을 만들기 전에 먼저 커넥터 연결 풀을 작성합니다.

커넥터 자원을 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 커넥터 노드를 확장합니다.
2. 커넥터 자원 노드를 확장합니다.
3. 커넥터 자원 페이지에서 새로 만들기를 누릅니다.
4. 커넥터 자원 만들기 페이지에서 자원의 설정을 지정합니다.
    a. JNDI 이름 필드에서 고유한 이름(예: `eis/myERP`)을 입력합니다. 반드시 슬래시를 입력합니다.
    b. 풀 이름 콤보 상자에서 새로운 커넥터 자원이 속하는 연결 풀을 선택합니다.
    c. 기본적으로 자원은 만들자마자 사용(활성화)할 수 있습니다. 자원을 사용할 수 없게 변경하려면 모든 대상에서 비활성화 라디오 단추를 선택합니다.

d. 페이지의 대상 섹션의 사용 가능한 필드에서 커넥터 자원이 상주하는 도메인, 클러스터 또는 서버 인스턴스를 선택하고 추가를 누릅니다. 선택한 필드에 나열된 도메인, 클러스터 또는 서버 인스턴스 중 하나에 커넥터 자원을 배포하지 않을 경우 필드에서 선택한 다음 제거를 누릅니다.

5. 확인을 누릅니다.

해당 asadmin 명령: `create-connector-resource`

커넥터 자원 편집

1. 트리 구성 요소에서 자원 노드를 확장한 다음 커넥터 노드를 확장합니다.
2. 커넥터 자원 노드를 확장합니다.
3. 편집할 커넥터 자원에 대한 노드를 선택합니다.
4. 커넥터 자원 편집 페이지의 풀 이름 메뉴에서 다른 연결 풀을 선택할 수 있습니다.
5. 대상 탭 창에서 대상 관리를 눌러 커넥터 자원이 배포되는 대상을 편집할 수 있습니다. 대상에 대한 자세한 내용은 커넥터 자원 만들기를 참조하십시오.
6. 저장을 눌러 편집한 내용을 적용합니다.

커넥터 자원 삭제

1. 트리 구성 요소에서 자원 노드를 확장한 다음 연결 노드를 확장합니다.
2. 커넥터 자원 노드를 선택합니다.
3. 커넥터 자원 페이지에서 삭제할 자원의 확인란을 선택합니다.
4. 삭제를 누릅니다.

해당 asadmin 명령: `delete-connector-resource`

커넥터 서비스 구성

이 클러스터나 서버 인스턴스에 배포된 모든 자원 어댑터에 대한 커넥터 컨테이너를 구성하려면 커넥터 서비스 화면을 사용합니다.

커넥터 컨테이너를 구성하려면 다음 작업을 수행합니다.

1. 트리에서 구성을 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 서버는 server-config 노드를 선택합니다.
   b. default-config 사본을 사용하는 미래 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 커넥터 서비스 노드를 선택합니다.
4. 종료 시간 초과 필드에서 종료 시간 초과(초)를 지정합니다. 커넥터 모듈 인스턴스의 `ResourceAdapter.stop` 메소드가 완료될 때까지 응용 프로그램 서버가 대기하는 시간(초)을 나타내는 정수를 입력합니다. 지정한 종료 시간 초과보다 오래 걸리는 자원 어댑터는 응용 프로그램 서버에서 무시하고 종료 절차가 계속됩니다. 기본 종료 시간 초과는 30초입니다. 이 클러스터나 서버 인스턴스에 배포된 자원 어댑터의 기본 종료 시간 초과를 선택하려면 기본값 로드를 누릅니다.

관리 객체 자원 작업

- 관리 객체 자원 만들기
- 관리 객체 자원 편집
- 관리 객체 자원 삭제

관리 객체 자원 만들기

자원 어댑터(커넥터 모듈)에 패키지화된 관리 객체는 응용 프로그램에 대한 특수한 기능을 제공합니다. 예를 들어, 관리 객체는 자원 어댑터 및 그와 연관된 EIS에 관련된 구문 분석기에 대한 액세스를 제공할 수 있습니다. 객체를 관리할 수 있습니다. 즉 관리자가 구성할 수 있습니다. 객체를 구성하려면 관리 객체 자원 만들기 또는 편집 페이지에서 이름-값 등록 정보 쌍을 추가합니다. 관리 객체 자원을 만들 경우 관리 객체를 JNDI 이름에 연관시킵니다.

Application Server는 자원 어댑터를 사용하여 JMS를 구현합니다. 작성된 모든 JMS 대상에 대해 Application Server는 관리 객체 자원을 자동으로 작성합니다.

관리 객체 자원을 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 자원 노드를 확장한 다음 커넥터 노드를 확장합니다.
2. 관리 객체 자원 노드를 확장합니다.
3. 관리 객체 자원 페이지에서 새로 만들기를 누릅니다.
4. 관리 객체 자원 페이지에서 다음 설정을 지정합니다.
   a. JNDI 이름 필드에서 자원을 식별하는 고유한 이름을 입력합니다.
   b. 자원 유형 필드에서 자원에 대한 Java 유형을 입력합니다.
   c. 자원 어댑터 콤보 상자에서 관리 객체를 포함하는 자원 어댑터를 선택합니다.
   d. 자원을 사용하거나 사용 불가능하게 하려면 확인란을 선택하거나 선택 해제합니다.
   e. 다음을 누릅니다.
5. 관리 객체 자원 만들기 두 번째 페이지에서 다음 작업을 수행할 수 있습니다.
   a. 이름-값 등록 정보 쌍을 사용하여 관리 객체를 구성하려면 등록 정보 추가를 누릅니다.
   b. 페이지의 대상 섹션의 사용 가능한 필드에서 관리 객체가 상주하는 도메인, 클러스터 또는 서버 인스턴스를 선택하고 추가를 누릅니다. 선택한 필드에 나열된 도메인, 클러스터 또는 서버 인스턴스 중 하나에 관리 객체를 배포 해제할 경우 필드에서 선택한 다음 제거를 누릅니다.
6. 마침을 누릅니다.

해당 asadmin 명령: create-admin-object

관리 객체 자원 편집

1. 트리 구성 요소에서 자원 노드를 확장한 다음 커넥터 노드를 확장합니다.
2. 관리 객체 자원 노드를 확장합니다.
3. 편집할 관리 객체 자원의 노드를 선택합니다.
4. 관리 객체 자원 편집 페이지에서 관리 객체 자원 만들기에서 지정한 값을 수정합니다.
5. 대상 탭 창에서 대상 관리를 눌러 관리 객체를 배포할 대상을 편집합니다. 대상에 대한 자세한 내용은 관리 객체 자원 만들기를 참조하십시오.
6. 저장을 눌러 편집한 내용을 적용합니다.

관리 객체 자원 삭제

1. 트리 구성 요소에서 자원 노드를 확장한 다음 커넥터 노드를 확장합니다.
2. 관리 객체 자원 노드를 선택합니다.
3. 관리 객체 자원 페이지에서 삭제할 자원의 확인란을 선택합니다.
4. 삭제를 누릅니다.

해당 asadmin 명령: delete-admin-object

명명된 구성 관리

이 장에서는 Application Server의 명명된 서버 구성 추가, 변경 및 사용에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- 명명된 구성 정보
- 명명된 구성의 관리 콘솔 작업

명명된 구성 정보

- 명명된 구성
- default-config 구성
- 인스턴스나 클러스터를 만들 때 작성된 구성
- 고유한 포트 번호 및 구성

명명된 구성

명명된 구성은 서버 구성 정보 집합입니다. 이 정보에는 HTTP Listener, ORB/IIOP Listener, JMS 브로커, EJB 컨테이너, 보안, 로깅 및 모니터링 등에 대한 구성 설정이 포함됩니다. 명명된 구성에는 응용 프로그램과 자원이 정의되지 않습니다.

구성은 관리 도메인에 작성됩니다. 같은 도메인의 클러스터나 여러 서버 인스턴스는 동일한 구성을 참조할 수도 있고 별도의 구성을 가질 수도 있습니다.

클러스터의 경우 같은 클러스터의 모든 서버 인스턴스는 클러스터의 구성을 상속하므로 같은 클러스터에 있는 인스턴스에는 같은 환경이 보장됩니다.

명명된 구성에는 많은 필수 구성 설정이 포함되기 때문에 기존의 명명된 구성을 복사하여 새로운 구성을 작성하는 것이 좋습니다. 새로 작성된 구성은 해당 구성 설정을 변경할 때까지 복사한 구성과 동일합니다.

클러스터나 인스턴스에서 구성을 사용하는 방법에 따라 세 가지 유형의 클러스터와 인스턴스가 있습니다.

- **독립 실행형.** 독립 실행형 서버 인스턴스나 클러스터는 구성을 다른 서버 인스턴스와 공유하지 않습니다. 즉 그 명명된 구성을 참조하는 다른 서버 인스턴스나 클러스터가 없습니다.
- **공유.** 공유 서버 인스턴스나 클러스터는 다른 서버 인스턴스나 클러스터와 구성을 공유합니다. 즉 동일한 명명된 구성을 참조하는 인스턴스나 클러스터가 여러 개 있습니다.
- **클러스터링됨.** 클러스터링된 서버 인스턴스는 클러스터의 구성을 상속합니다.

default-config 구성

`default-config` 구성은 독립 실행형 서버 인스턴스나 독립 실행형 클러스터 구성을 만들 때 사용하는 템플리트 역할을 하는 특수한 구성입니다. 어떤 클러스터링된 서버 인스턴스나 클러스터도 `default-config` 구성을 참조할 수 없습니다. 이 구성은 새로운 구성을 만들기 위해서 복사만 할 수 있습니다. 기본 구성을 복사한 다음 정확한 초기 설정을 반영하도록 편집합니다.

인스턴스나 클러스터를 만들 때 작성된 구성

새로운 서버 인스턴스나 새로운 클러스터를 작성할 경우 다음 중 하나를 수행합니다.

- 기존 구성을 참조합니다. 새로운 구성이 추가되지 않습니다.
- 기존 구성을 복사합니다. 서버 인스턴스나 클러스터를 추가할 때 새로운 구성이 추가됩니다.

기본적으로 새로운 클러스터나 인스턴스는 `default-config` 구성에서 복사한 구성으로 만들어집니다. 다른 구성에서 복사하려면 새로운 인스턴스나 클러스터를 작성할 때 지정하십시오.

서버 인스턴스의 경우 새로운 구성 이름은 *instance_name*`-config`입니다. 클러스터의 경우 새로운 구성 이름은 *cluster-name*`-config`입니다.

고유한 포트 번호 및 구성

동일한 호스트 시스템의 여러 인스턴스가 동일한 구성을 참조할 경우 인스턴스마다 고유한 포트 번호에서 수신해야 합니다. 예를 들어, 두 개의 서버 인스턴스가 포트 80의 HTTP Listener를 사용하여 명명된 구성을 참조할 경우 포트 충돌로 인해 서버 인스턴스 중 하나를 시작할 수 없습니다. 개별 서버 인스턴스가 수신하는 포트 번호를 정의하는 등록 정보를 고유한 포트 번호를 사용하도록 변경합니다.

포트 번호에는 다음 원칙이 적용됩니다.

- 개별 서버 인스턴스의 포트 번호는 처음에 구성으로부터 상속됩니다.
- 서버 인스턴스를 작성할 때 포트가 이미 사용 중이면 포트 충돌을 방지하기 위해 상속된 기본값을 인스턴스 수준에서 대체합니다.
- 인스턴스에서 구성을 공유하는 것으로 가정합니다. 구성에 포트 번호 n이 있습니다. 동일한 구성을 사용하는 시스템에서 새로운 인스턴스를 만들 경우 새로운 인스턴스에는 포트 번호 n+1이 지정됩니다. 이 포트 번호를 사용할 수 없으면 n+1 다음에 사용 가능한 포트가 선택됩니다.
- 구성의 포트 번호를 변경하면 그 포트 번호를 상속하는 서버 인스턴스는 변경된 포트 번호를 자동으로 상속합니다.
- 인스턴스의 포트 번호를 변경하고 계속해서 구성의 포트 번호를 변경하면 인스턴스의 포트 번호는 바뀌지 않은 채 남아 있습니다.

명명된 구성의 관리 콘솔 작업

명명된 구성 만들기

명명된 구성을 만들려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.

2. 구성 페이지에서 새로 만들기를 누릅니다.
3. 구성 만들기 페이지에서 구성의 고유한 이름을 입력합니다.
4. 복사할 구성을 선택합니다.

   `default-config` 구성이 독립 실행형 인스턴스나 독립 실행형 클러스터를 만들 때 사용되는 기본 구성입니다.

해당 `asadmin` 명령: `copy-config`

명명된 구성의 등록 정보 편집

명명된 구성의 등록 정보를 편집하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 명명된 구성의 노드를 선택합니다.
3. 구성 시스템 등록 정보 페이지에서 동적 재구성을 사용 가능하게 할 것인지 여부를 선택합니다.

   사용 가능하게 하면 서버를 다시 시작하지 않고도 구성에 대한 변경 사항이 서버 인스턴스에 적용됩니다.
4. 등록 정보를 추가하거나, 등록 정보의 현재 값을 변경하거나, 등록 정보를 삭제합니다.

   이미 정의된 등록 정보가 포트입니다. 시스템에 서버 인스턴스가 여러 개 있을 경우 포트 번호는 고유해야 합니다.

표 12-1은 미리 정의된 등록 정보 목록과 해당 설명입니다.

표 12-1 명명된 구성의 등록 정보

등록 정보 이름	설명
HTTP_LISTENER_PORT	Http-listener-1에 대한 포트 번호를 지정합니다. 유효한 값은 1–65535입니다. UNIX의 경우 1–1024를 수신하는 소켓을 만들려면 수퍼유저 권한이 필요합니다.
HTTP_SSL_LISTENER_PORT	Http-listener-2에 대한 포트 번호를 지정합니다. 유효한 값은 1–65535입니다. UNIX의 경우 1–1024를 수신하는 소켓을 만들려면 수퍼유저 권한이 필요합니다.
IIOP_SSL_LISTENER_PORT	SSL이라고 하는 IIOP Listener가 수신하는 IIOP 연결에 대한 ORB Listener 포트를 지정합니다.

표 12-1 명명된 구성의 등록 정보

등록 정보 이름	설명
IIOP_LISTENER_PORT	ORB-listener-1이 수신하는 IIOP 연결에 대한 ORB Listener 포트를 지정합니다.
JMX_SYSTEM_CONNECTOR_PORT	JMX 커넥터가 수신하는 포트 번호를 지정합니다. 유효한 값은 1–65535입니다. UNIX의 경우 1–1024를 수신하는 소켓을 만들려면 수퍼유저 권한이 필요합니다.
IIOP_SSL_MUTUALAUTH_PORT	SSL_MUTUALAUTH라고 하는 IIOP Listener가 수신하는 IIOP 연결에 대한 ORB listener 포트를 지정합니다.

5. 구성과 연관된 모든 인스턴스에 대한 등록 정보의 현재 값을 편집하려면 인스턴스 값을 누릅니다.

해당 `asadmin` 명령: `set`

구성을 참조하는 인스턴스의 포트 번호 편집

명명된 구성을 참조하는 모든 인스턴스마다 우선 그 구성에서 포트 번호를 상속합니다. 시스템에서 포트 번호가 고유해야 하기 때문에 상속된 포트 번호를 대체해야 할 수도 있습니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 명명된 구성의 노드를 선택합니다.
3. 시스템 등록 정보 구성 페이지에서 편집할 포트 번호 옆에 있는 인스턴스 값을 누릅니다.

   예를 들어, SSL 포트 등록 정보 옆에 있는 인스턴스 값을 누르면 그 구성을 참조하는 모든 서버 인스턴스의 SSL 포트 값이 표시됩니다.
4. 포트 값을 변경하고 저장을 누릅니다.

해당 `asadmin` 명령: `set`

명명된 구성의 대상 보기

명명된 구성의 대상을 보려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 명명된 구성의 노드를 선택합니다.

   구성 시스템 등록 정보 페이지에 그 구성을 사용하는 모든 대상 목록이 표시됩니다. 클러스터 구성의 경우 대상은 클러스터입니다. 인스턴스 구성의 경우 대상은 인스턴스입니다.

명명된 구성 삭제

명명된 구성을 삭제하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성 페이지에서 삭제할 명명된 구성의 확인란을 선택합니다.

   `default-config` 구성은 삭제할 수 없습니다.
3. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-config`

J2EE 컨테이너

이 장에서는 서버에 포함된 J2EE 컨테이너를 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- J2EE 컨테이너 정보
- J2EE 컨테이너의 관리 콘솔 작업

J2EE 컨테이너 정보

이 절에서는 Application Server에 포함된 J2EE 컨테이너에 대해 설명합니다.

- J2EE 컨테이너 유형
- 웹 컨테이너
- EJB 컨테이너

J2EE 컨테이너 유형

J2EE 컨테이너에서는 J2EE 응용 프로그램 구성 요소에 대한 런타임 지원을 제공합니다. J2EE 응용 프로그램 구성 요소에서는 컨테이너의 프로토콜 및 메소드를 사용하여 서버에서 제공하는 다른 응용 프로그램 구성 요소와 서비스에 액세스합니다. Application Server에서는 응용 프로그램 클라이언트 컨테이너, 애플릿 컨테이너, 웹 컨테이너 및 EJB 컨테이너를 제공합니다. 컨테이너를 표시하는 다이어그램은 Application Server 구조 절을 참조하십시오.

웹 컨테이너

웹 컨테이너는 웹 응용 프로그램을 호스트하는 J2EE 컨테이너입니다. 웹 컨테이너는 개발자에게 서블릿과 JSP(Java Server Page)를 실행하는 환경을 제공함으로써 웹 서버 기능을 확장합니다.

EJB 컨테이너

Enterprise Bean(EJB 구성 요소)은 비즈니스 논리를 포함하는 Java 프로그래밍 언어 서버 구성 요소입니다. EJB 컨테이너는 Enterprise Bean에 대한 로컬 및 원격 액세스를 제공합니다.

Enterprise Bean에는 Session Bean, Entity Bean 및 Message-Driven Bean 등 세 가지 유형이 있습니다. Session Bean은 임시 객체와 프로세스를 나타내며 대개 단일 클라이언트에서 사용합니다. Entity Bean은 대개 데이터베이스에서 유지 관리되는 지속성 필드를 나타냅니다. Message-Driven Bean은 응용 프로그램 모듈과 서비스에 비동기적으로 메시지를 전달하기 위해 사용합니다.

컨테이너는 Enterprise Bean을 작성하고, Enterprise Bean을 이름 지정 서비스에 바인딩하여 다른 응용 프로그램 구성 요소가 Enterprise Bean에 액세스할 수 있게 하고, 인증된 클라이언트만 Enterprise Bean의 메소드에 액세스하고, Bean의 상태를 영구 저장소에 저장하며, Bean의 상태를 캐시하고, 필요한 경우 Bean을 활성화 또는 비활성화합니다.

J2EE 컨테이너의 관리 콘솔 작업

일반 웹 컨테이너 설정 구성

이 릴리스에는 관리 콘솔의 웹 컨테이너에 대한 컨테이너 차원의 설정 값이 없습니다.

웹 컨테이너 세션 구성

이 절에서는 웹 컨테이너의 HTTP 세션 설정을 설명합니다. HTTP 세션은 영구 저장소에 상태 데이터를 기록한 고유한 웹 세션입니다.

세션 시간 초과 값을 설정하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 웹 컨테이너 노드를 선택합니다.
4. 세션 등록 정보 탭을 누릅니다.
5. 세션 시간 초과 필드에 세션이 유효한 시간(초)을 입력합니다.
6. 저장을 누릅니다.

관리자 등록 정보 구성

세션 관리자는 세션을 작성 및 완전 삭제하는 방법, 세션 상태를 저장하는 장소 및 최대 세션 수를 구성하는 방법을 제공합니다.

세션 관리자 설정을 변경하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 웹 컨테이너 노드를 선택합니다.
4. 관리자 등록 정보 탭을 누릅니다.
5. 리프 간격 값을 설정합니다.

   리프 간격 필드는 저장소에서 비활성 세션 데이터를 삭제하기 전까지의 시간(초)입니다.

6. 최대 세션 값을 설정합니다.

   최대 세션 필드는 허용된 최대 세션 수입니다.

7. 세션 파일 이름-값을 설정합니다.

   세션 파일 이름 필드는 세션 데이터를 포함하는 파일입니다.

8. 세션 아이디 생성기 클래스 이름-값을 설정합니다.

   세션 아이디 생성기 클래스 이름 필드를 사용하면 고유한 세션 아이디를 생성하는 데 필요한 사용자 정의 클래스를 지정할 수 있습니다. 서버 인스턴스 당 세션 아이디 생성기 클래스 하나만 허용되므로 클러스터의 모든 인스턴스는 세션 키 충돌을 방지하기 위해 동일한 세션 아이디 생성기를 사용해야 합니다.

   사용자 정의 세션 아이디 생성기 클래스는 `com.sun.enterprise.util.uuid.UuidGenerator` 인터페이스를 구현해야 합니다.

```
package com.sun.enterprise.util.uuid;

public interface UuidGenerator {

    public String generateUuid();
    public String generateUuid(Object obj);  //obj is the session object

}
```

   클래스는 Application Server 클래스 경로에 있어야 합니다.

9. 저장을 누릅니다.

저장소 등록 정보 구성

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. 웹 컨테이너 노드를 선택합니다.
4. 저장소 등록 정보 탭을 누릅니다.

5. 리프 간격을 설정합니다.

   리프 간격 필드는 저장소에서 비활성 세션 데이터를 삭제하기 전까지의 시간(초)입니다.

6. 저장을 누릅니다.

일반 EJB 설정 구성

이 절에서는 서버의 모든 Enterprise Bean 컨테이너에 적용되는 다음 설정에 대해 설명합니다.

- 세션 저장 위치
- 풀 설정
- 캐시 설정

컨테이너별로 기본값을 대체하려면 Enterprise Bean의 `sun-ejb-jar.xml` 파일에 있는 값을 조정합니다. 자세한 내용은 *Application Server Developer's Guide*를 참조하십시오(설명서에 대한 링크는 추가 정보 참조).

세션 저장 위치

세션 저장 위치 필드는 비활성화된 Bean과 영구 HTTP 세션을 파일 시스템에 저장하는 디렉토리를 지정합니다.

비활성화된 Bean은 파일 시스템의 파일에 상태를 기록한 Enterprise Bean입니다. 비활성화된 Bean은 대개 일정 기간 동안 유휴 상태였고 현재 클라이언트가 액세스하지 않은 것입니다.

비활성화된 Bean과 마찬가지로 영구 HTTP 세션은 파일 시스템의 파일에 자신의 상태를 기록한 개별 웹 세션입니다.

완결 옵션 필드는 트랜잭션 간의 비활성화된 Entity Bean 인스턴스를 컨테이너가 캐시하는 방법을 지정합니다.

옵션 B는 트랜잭션 간의 Entity Bean 인스턴스를 캐시하며 기본적으로 선택되어 있습니다. 옵션 C는 캐시를 비활성화합니다.

풀 설정

Bean을 작성하여 생기는 성능 적중 없이 클라이언트 요청에 응답할 수 있도록 컨테이너는 Enterprise Bean 풀을 유지 관리합니다. 이 설정은 Stateless Session Bean과 Entity Bean에만 적용됩니다.

배포된 Enterprise Bean을 사용하는 응용 프로그램에 성능 문제가 있을 경우 풀을 만들거나 기존 풀에서 유지 관리하는 Bean 수를 늘리면 응용 프로그램의 성능을 늘리는 데 도움이 될 수 있습니다.

기본적으로 컨테이너는 Enterprise Bean의 풀을 유지 관리합니다.

Enterprise Bean의 컨테이너 풀 구성을 조정하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. EJB 컨테이너 노드를 선택합니다.
4. 초기 및 최소 풀 크기 필드의 풀 설정에서 컨테이너가 풀에 만든 최소 Bean 개수를 입력합니다.
5. 최대 풀 크기 필드에서 컨테이너가 언제든지 풀에서 유지하는 최대 Bean 수를 입력합니다.
6. 풀 크기 조정 개수 필드에서 풀 유휴 시간 초과에 지정한 시간 이상 Bean이 유휴 상태일 경우 풀에서 제거되는 Bean 수를 입력합니다.
7. 풀 유휴 시간 초과 필드에 풀에서 Bean을 제거하기 전에 얼마 동안 풀의 Bean이 유휴 상태이어야 하는지 시간(초)을 입력합니다.
8. 저장을 누릅니다.
9. 서버를 다시 시작합니다.

캐시 설정

컨테이너는 가장 많이 사용하는 Enterprise Bean에 대한 Enterprise Bean 데이터의 캐시를 유지 관리합니다. 그러면 컨테이너가 Enterprise Bean의 데이터에 대한 다른 응용 프로그램 모듈의 요청에 더 신속하게 응답할 수 있습니다. 이 절은 Stateful Session Bean과 Entity Bean에만 적용됩니다.

캐시된 Enterprise Bean의 상태는 활성, 유휴 또는 비활성 중 하나입니다. 활성화된 Enterprise Bean은 현재 클라이언트가 액세스하고 있습니다. 유휴 Enterprise Bean의 데이터는 현재 캐시에 있지만 Bean에 액세스하는 클라이언트가 없습니다. 비활성화된 Bean의 데이터는 임시로 저장되지만 클라이언트가 Bean을 요청할 경우 다시 캐시로 읽어들입니다.

캐시된 Enterprise Bean의 설정을 조정하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. EJB 컨테이너 노드를 선택합니다.
4. 최대 캐시 크기 필드의 최대 캐시 크기를 조정합니다.

   Bean 작성 및 완전 삭제의 오버헤드를 제거하려면 캐시할 Bean의 최대 개수를 늘립니다. 그러나 캐시를 늘릴 경우 서버에서 더 많은 메모리와 자원을 소모합니다. 운영 환경이 캐시 설정에 충분해야 합니다.
5. 캐시 크기 조정 개수 필드에서 캐시 크기 조정 개수를 조정합니다.

   캐시된 Bean이 최대 개수에 도달하면 컨테이너는 백업 저장소에서 비활성화된 Bean을 제거합니다. 이 개수는 기본적으로 32로 설정됩니다.
6. 캐시 유휴 시간 초과 필드에서 Entity Bean에 예약된 캐시 정리 속도(초)를 조정합니다.

   캐시된 Entity Bean이 일정 기간 동안 유휴 상태일 경우 비활성화됩니다. 즉 Bean을 상태가 백업 저장소에 기록됩니다.
7. 제거 시간 초과 필드에서 캐시 또는 비활성화된 저장소에서 Stateful Session Bean을 얼마 후에 제거할지 시간(초)을 조정합니다.
8. 제거 선택 정책 필드에서 Stateful Session Bean을 제거하기 위해 컨테이너에서 사용하는 정책을 구성합니다.

   컨테이너는 제거 선택 정책 필드에 설정된 정책을 기반으로 제거할 Stateful Session Bean을 결정합니다. 캐시에서 Bean을 제거하기 위해 컨테이너에서 사용할 수 있는 세 가지 정책은 다음과 같습니다.

   - 최근에 사용되지 않음(NRU)
   - 선입선출(FIFO)

- 가장 오래 전에 사용됨(LRU)

NRU 정책은 최근에 사용하지 않은 Bean을 제거합니다. FIFO 정책은 캐시에서 가장 오래된 Bean을 제거합니다. LRU 정책은 가장 오래 전에 사용된 Bean을 제거합니다. 기본적으로 컨테이너에서는 NRU 정책을 사용합니다.

Entity Bean은 항상 FIFO 정책을 사용하여 제거됩니다.

9. 저장을 누릅니다.
10. 응용 프로그램 서버를 다시 시작합니다.

Message-Driven Bean 설정 구성

Message-Driven Bean의 풀은 "일반 EJB 설정 구성"에서 설명한 Session Bean의 풀과 유사합니다.

기본적으로 컨테이너는 Message Bean의 풀을 유지 관리합니다.

이 풀의 구성을 조정하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. EJB 컨테이너 노드를 선택합니다.
4. MDB 설정 탭을 누릅니다.
5. 초기 및 최소 풀 크기 필드의 풀 설정에서 컨테이너가 풀에 작성한 Message Bean의 최소 개수를 입력합니다.
6. 최대 풀 크기에서 언제든지 컨테이너가 풀에 유지하는 Bean의 최대 개수를 입력합니다.
7. 풀 크기 조정 개수 필드에서 Bean이 풀 유휴 시간 초과에 지정된 시간 이상으로 유휴 상태일 경우 풀에서 제거되는 Bean 개수를 입력합니다.
8. 풀 유휴 시간 초과 필드에서 풀에서 Bean을 제거하기 전에 풀의 Bean이 유휴 상태를 유지할 수 있는 시간(초)을 입력합니다.
9. 저장을 누릅니다.

10. 서버를 다시 시작합니다.

EJB 타이머 서비스 설정 구성

타이머 서비스는 Enterprise Bean에서 사용하는 알림이나 이벤트를 예약하기 위해 Enterprise Bean 컨테이너에서 제공하는 영구적인 트랜잭션 알림 서비스입니다. Stateful Session Bean을 제외한 모든 Enterprise Bean은 타이머 서비스에서 알림을 받을 수 있습니다. 서비스에서 설정한 타이머는 서버가 종료되거나 다시 시작될 경우 완전 삭제되지 않습니다.

타이머 서비스 구성

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. EJB 컨테이너 노드를 선택합니다.
4. EJB 타이머 서비스 탭을 누릅니다.
5. 최소 전달 간격 필드에서 최소 전달 간격(밀리초)을 설정합니다. 최소 전달 간격은 특정 타이머의 다음 타이머 만료가 발생하기 전에 허용된 최소 밀리초입니다. 이 간격을 너무 작게 설정하면 서버가 오버로드될 수 있습니다.
6. 최대 재전송 필드에서 타이머 서비스가 알림을 전달하는 최대 시도 횟수를 설정합니다.
7. 재전송 간격 필드에서 재전송 시도간의 간격(밀리초)을 설정합니다.
8. 저장을 누릅니다.
9. 서버를 다시 시작합니다.

타이머 서비스에 외부 데이터베이스 사용

기본적으로 타이머 서비스는 내장된 데이터베이스를 사용하여 타이머를 저장합니다.

외부 데이터베이스를 사용하여 타이머를 저장하려면 다음 작업을 수행합니다.

1. 142페이지의 "JDBC 자원 만들기"에서 설명한 대로 데이터베이스에 대해 JDBC 자원을 설정합니다.
2. 타이머 데이터 소스 필드에 자원의 JNDI 이름을 입력합니다.
3. 저장을 누릅니다.
4. 서버를 다시 시작합니다.

<*INSTALL_DIR*>`/lib/install/databases/`에서 PointBase 및 Oracle에 대한 샘플 타이머 데이터베이스 작성 파일이 제공됩니다.

보안 구성

이 장에서는 핵심 Application Server 보안 개념과 Sun Java System Application Server 8.1 2005Q1의 보안 구성 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- Application Server 보안 정보
- 보안에 대한 관리 콘솔 작업
- 영역에 대한 관리 콘솔 작업
- JACC 공급자에 대한 관리 콘솔 작업
- 감사 모듈에 대한 관리 콘솔 작업
- 메시지 보안에 대한 관리 콘솔 작업
- Listener 및 JMX 커넥터에 대한 관리 콘솔 작업
- 커넥터 연결 풀에 대한 관리 콘솔 작업
- 인증서 및 SSL 작업
- 자세한 정보

Application Server 보안 정보

- 보안 개요
- 인증 및 권한 부여 정보
- 사용자, 그룹, 역할 및 영역 이해
- 인증서 및 SSL 소개
- 방화벽 정보

- 관리 콘솔을 사용하여 보안 관리

보안 개요

보안이란 데이터를 보호하는 것이며 데이터를 저장하거나 전송할 때 해당 데이터에 대한 인증되지 않은 액세스나 손상을 방지하는 방법입니다. Application Server에는 J2EE 표준을 기반으로 한 동적이며 확장 가능한 보안 구조가 있습니다. 내장된 보안 기능에는 암호화 도구, 인증 및 권한 부여, 공개 키 인프라가 포함됩니다. Application Server는 시스템이나 사용자에 대한 잠재적인 위험 없이 응용 프로그램을 안전하게 실행할 수 있는 샌드박스를 사용하는 Java 보안 모델에 구축되어 있습니다. 이 장은 다음 내용으로 구성되어 있습니다.

- 응용 프로그램 및 시스템 보안 이해
- 보안 관리 도구
- 비밀번호 보안 관리
- 보안 책임 지정

응용 프로그램 및 시스템 보안 이해

대개 다음과 같은 두 종류의 응용 프로그램 보안이 있습니다.

- *프로그래밍 방식의 보안*의 경우 개발자가 작성한 응용 프로그램 코드에서 보안 작업을 처리합니다. 관리자로서 이 체계에 대한 제어권을 갖고 있지 않습니다. 일반적으로 J2EE 컨테이너를 통해 구성을 관리하는 대신 응용 프로그램에 보안 구성을 하드 코드하기 때문에 프로그래밍 방식의 보안을 권장하지 않습니다.
- *선언적 보안*의 경우 컨테이너(Application Server)는 응용 프로그램의 배포 설명자를 통해 보안을 처리합니다. 배포 설명자를 직접 편집하거나 `deploytool` 같은 도구를 사용하여 선언적 보안을 제어할 수 있습니다. 응용 프로그램을 개발한 후 배포 설명자를 변경할 수 있기 때문에 선언적 보안을 사용하면 더 많은 융통성이 허용됩니다.

응용 프로그램 보안 외에도 Application Server 시스템의 모든 응용 프로그램에 영향을 미치는 *시스템 보안*이 있습니다.

프로그래밍 방식의 보안은 응용 프로그램 개발자가 제어하므로, 이 문서에서 설명하지 않습니다. 선언적 보안의 경우 응용 프로그램 개발자가 덜 제어하므로 이 문서에서 가끔 설명합니다. 이 문서는 기본적으로 시스템 관리자를 대상으로 하므로 시스템 보안에 대해 중점적으로 설명합니다.

보안 관리 도구

Application Server에서는 보안을 관리하기 위한 다음 도구를 제공합니다.

- 관리 콘솔은 전체 서버의 보안을 구성하고, 사용자, 그룹 및 영역을 관리하며, 시스템 차원의 다른 보안 작업을 수행하기 위해 사용하는 브라우저 기반의 도구입니다. 관리 콘솔에 대한 일반적인 소개는 "관리 도구"를 참조하십시오. 관리 콘솔로 수행할 수 있는 보안 작업 개요는 "관리 콘솔을 사용하여 보안 관리"를 참조하십시오.
- `asadmin`은 관리 콘솔과 동일한 많은 작업을 수행하는 명령줄 도구입니다. 관리 콘솔로 수행할 수 없는 일부 작업을 `asadmin`으로 수행할 수 있습니다. 명령 프롬프트나 스크립트에서 `asadmin` 명령을 수행하여 반복적인 작업을 자동화할 수 있습니다. `asadmin`에 대한 일반적인 소개는 "관리 도구"를 참조하십시오.
- `deploytool`은 그래픽 패키지 및 배포 도구로, 개별 응용 프로그램의 보안을 제어하기 위해 응용 프로그램 배포 설명자를 편집하는 데 사용합니다. `deploytool`은 응용 프로그램 개발자를 대상으로 하기 때문에 이 문서에서는 자세히 설명하지 않습니다. `deploytool` 사용에 대한 지침은 `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`에서 이 도구의 온라인 도움말과 *J2EE 1.4 Tutorial*을 참조하십시오.

Java 2 Platform, Standard Edition(J2SE)에서는 보안을 관리하기 위한 다음 두 가지 도구를 제공합니다.

- `keytool`은 디지털 인증서와 키 쌍을 관리하기 위한 명령줄 유틸리티입니다. `keytool`을 사용하여 `certificate` 영역의 사용자를 관리합니다.
- `policytool`은 시스템 차원의 Java 보안 정책을 관리하기 위한 그래픽 유틸리티입니다. 관리자는 `policytool`을 거의 사용할 필요가 없습니다.

`keytool`, `policytool` 및 기타 Java 보안 도구 사용에 대한 자세한 내용은 `http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security`에서 *Java 2 SDK Tools and Utilities*를 참조하십시오.

Enterprise Edition의 경우 NSS(Network Security Services)를 구현하는 다른 두 개의 도구를 보안 관리에 사용할 수 있습니다. NSS에 대한 자세한 내용을 보려면 `http://www.mozilla.org/projects/security/pki/nss/`로 이동하십시오. 보안 관리 도구는 다음과 같습니다.

- `certutil`는 인증서 및 키 데이터베이스를 관리하기 위한 명령줄 유틸리티입니다.
- `pk12util`은 인증서/키 데이터베이스와 PKCS12 형식의 파일 간에 키와 인증서를 내보내고 가져오기 위해 사용하는 명령줄 유틸리티입니다.

certutil, pk12util 및 다른 NSS 보안 도구 사용에 대한 자세한 내용은 http://www.mozilla.org/projects/security/pki/nss/tools에서 *NSS 보안 도구*를 참조하십시오.

비밀번호 보안 관리

이 릴리스의 Application Server의 경우 특정 도메인의 사양을 포함하는 domain.xml 파일에는 처음부터 일반 텍스트로 된 IMQ 브로커 비밀번호가 포함되어 있습니다. 이 비밀번호를 포함하는 domain.xml 파일의 요소는 jms-host 요소의 admin-password 속성입니다. 설치 시 이 비밀번호를 변경할 수 없기 때문에 보안에 큰 영향을 미치지 않습니다.

그러나 관리 콘솔을 사용하여 사용자와 자원을 추가하고 이 사용자와 자원에 비밀번호를 지정합니다. 예를 들어, 데이터베이스에 액세스하기 위한 비밀번호처럼 이 비밀번호 중 일부는 domain.xml 파일에 일반 텍스트로 기록됩니다. 이 비밀번호를 domain.xml 파일에 일반 텍스트로 기록하면 보안 위험이 있을 수 있습니다. 다음 절차를 수행하여 admin-password 속성이나 데이터베이스 비밀번호를 포함하여 domain.xml의 비밀번호를 암호화할 수 있습니다.

1. domain.xml 파일이 상주하는 디렉토리(기본적으로 *install_dir*/domains/*domain_dir*/config)에서 다음 asadmin 명령을 실행합니다.

   ```
   asadmin create-password-alias <alias-name>
   ```

   예를 들면 다음과 같습니다.

   ```
   asadmin create-password-alias jms-password
   ```

   비밀번호 프롬프트(이 경우 admin)가 표시됩니다. 자세한 내용은 create-password-alias, list-password-aliases, delete-password-alias 명령의 설명서 페이지를 참조하십시오.

2. domain.xml의 비밀번호를 제거하고 바꿉니다. asadmin set 명령을 사용하여 이를 수행합니다. 이러한 목적을 위한 set 명령 사용 예는 다음과 같습니다.

   ```
   asadmin set
   server.jms-service.jms-host.default_JMS_host.admin-password=${ALIAS=jms
   -password}
   ```

3. 관련 도메인에 대한 Application Server를 다시 시작합니다.

암호화된 비밀번호를 사용하여 파일 보호

일부 파일에는 파일 시스템 권한을 사용하여 보호해야 하는 암호화된 비밀번호가 포함되어 있습니다. 이 파일에는 다음 내용이 포함되어 있습니다.

- *install_dir*/domains/*domain_dir*/master-password

이 파일에는 암호화된 마스터 비밀번호가 포함되어 있고 파일 시스템 권한 600을 사용하여 이 파일을 보호해야 합니다.

- asadmin에 대한 --passwordfile 인수를 사용하여 인수로 전달하기 위해 만든 비밀번호 파일은 파일 시스템 권한 600을 사용하여 보호해야 합니다.

마스터 비밀번호 변경

마스터 비밀번호(MP)는 전체적으로 공유하는 비밀번호입니다. 마스터 비밀번호는 인증에 사용되지 않고 네트워크를 통해 전송되지 않습니다. 이 비밀번호는 전체적인 보안의 억제 지점입니다. 필요할 경우 수동으로 입력하도록 선택하거나 파일에 숨길 수 있습니다. 비밀번호는 시스템에서 가장 중요한 데이터입니다. 이 파일을 제거하여 마스터 비밀번호 요구를 강제로 실행할 수 있습니다. 마스터 비밀번호를 변경한 경우 마스터 비밀번호 키 저장소에 다시 저장됩니다.

마스터 비밀번호를 변경하려면 다음 절차를 수행합니다.

1. 도메인에 대한 Application Server를 중지합니다. 이전 비밀번호와 새 비밀번호를 프롬프트한 다음 모든 하위 종속 항목을 다시 암호화하는 asadmin 명령 change-master-password를 사용합니다. 예를 들면 다음과 같습니다.

   ```
   asadmin change-master-password>
   Please enter the master password>
   Please enter the new master password>
   Please enter the the new master password again>
   ```

2. 서버를 다시 시작합니다.

경고: 이 때 실행 중인 서버 인스턴스를 시작하지 말고 해당하는 노드의 SMP 에이전트가 변경될 때까지 실행 중인 서버 인스턴스를 다시 시작하지 말아야 합니다. SMP를 변경하기 전에 서버 인스턴스를 다시 시작한 경우 서버 인스턴스가 시작되지 않습니다.

3. 노드 에이전트 및 관련된 서버를 한 번에 하나씩 중지합니다. asadmin change-master-password 명령을 다시 실행한 다음, 노드 에이전트 및 관련된 서버를 다시 시작합니다.

4. 모든 노드 에이전트를 주소 지정할 때까지 다음 노드 에이전트를 계속합니다. 이런 방법으로 변경 사항 롤백을 수행할 수 있습니다.

관리 비밀번호 변경

관리 비밀번호 암호화는 "비밀번호 보안 관리"에서 설명합니다. 관리 비밀번호를 암호화할 것을 강력하게 권장합니다. 관리 비밀번호를 암호화하기 전에 변경할 경우 asadmin set 명령을 사용합니다. 다음은 이러한 용도로 set 명령을 사용하는 예입니다.

```
asadmin set
server.jms-service.jms-host.default_JMS_host.admin-password=new_pwd
```

관리 콘솔을 사용하여 관리 비밀번호를 변경할 수도 있습니다. 관리 콘솔을 사용하여 관리 비밀번호를 변경하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스인 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 영역 노드를 확장합니다.
5. admin-realm 노드를 선택합니다.
6. 사용자를 추가, 수정, 삭제하려면 사용자 관리 버튼을 누릅니다.
7. admin이라는 사용자를 선택합니다.
8. 새로운 비밀번호를 입력하고 비밀번호를 확인합니다.
9. 저장을 눌러 저장하거나, 닫기를 눌러 저장하지 않고 닫습니다.

보안 책임 지정

다음 항목에 보안 책임이 지정됩니다.

- 응용 프로그램 개발자
- 응용 프로그램 배포자
- 시스템 관리자

응용 프로그램 개발자

응용 프로그램 개발자의 책임은 다음과 같습니다.

- 응용 프로그램 구성 요소에 대한 역할 및 역할 기반 액세스 제한 사항 지정
- 응용 프로그램의 인증 방법 정의 및 보안을 설정할 응용 프로그램의 부분 지정

응용 프로그램 개발자는 deploytool 같은 도구를 사용하여 응용 프로그램 배포 설명자를 편집할 수 있습니다. 다음 URL에서 확인할 수 있는 *J2EE 1.4 Tutorial*의 *Security* 장에서 이 보안 작업을 자세히 설명합니다.

http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

응용 프로그램 배포자

응용 프로그램 배포자의 책임은 다음과 같습니다.

- 사용자나 그룹(또는 둘 다)을 보안 역할에 매핑
- 특정 배포 시나리오의 요구 사항에 맞게 구성 요소 메소드에 액세스하는 데 필요한 권한 조정

응용 프로그램 배포자는 `deploytool` 같은 도구를 사용하여 응용 프로그램 배포 설명자를 편집할 수 있습니다. 다음 URL에서 확인할 수 있는 *J2EE 1.4 Tutorial*의 *Security* 장에서 이 보안 작업을 자세히 설명합니다.

`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`

시스템 관리자

시스템 관리자의 책임은 다음과 같습니다.

- 보안 영역 구성
- 사용자 계정 및 그룹 관리
- 감사 로그 관리
- 서버 인증서 관리 및 서버의 SSL(Secure Socket Layer) 사용 구성
- 커넥터 연결 풀에 대한 보안 맵, 추가 JACC 공급자 등과 같은 시스템 차원의 다양한 다른 보안 기능 처리

시스템 관리자는 관리 콘솔을 사용하여 서버 보안 설정을 관리하고 `certutil`을 사용하여 인증서를 관리합니다. 이 문서는 기본적으로 시스템 관리자를 대상으로 합니다.

인증 및 권한 부여 정보

인증 및 권한 부여는 응용 프로그램 서버 보안의 핵심 개념입니다. 인증 및 권한 부여와 관련해서 다음 내용을 설명합니다.

- 엔티티 인증
- 사용자 권한 부여
- JACC 공급자 지정
- 인증 및 권한 부여 결정 사항 감사

- 메시지 보안 구성

엔티티 인증

*인증*은 엔티티(사용자, 응용 프로그램 또는 구성 요소)가 다른 엔티티를 확인하는 방법입니다. 엔티티는 *보안 자격 증명*을 사용하여 자신을 인증합니다. 아이디 및 비밀번호, 디지털 인증서 등이 자격 증명이 될 수 있습니다.

일반적으로 인증은 아이디와 비밀번호를 사용하여 응용 프로그램에 로그인하는 것을 의미하지만, 서버의 자원을 요청할 경우 EJB 공급 보안 자격 증명을 가리킬 수도 있습니다. 대개 서버나 응용 프로그램에서는 클라이언트의 인증을 요구합니다. 그리고 클라이언트에서는 서버가 자신을 인증할 것을 요구할 수도 있습니다. 인증이 양방향일 경우 이를 *상호 인증*이라고 합니다.

엔티티가 보호된 자원에 액세스할 경우 Application Server는 해당 자원에 대해 구성된 인증 체계를 사용하여 액세스를 부여할지 여부를 결정합니다. 예를 들어, 사용자는 웹 브라우저에서 아이디와 비밀번호를 입력할 수 있습니다. 응용 프로그램에서 해당 자격 증명을 확인하면 사용자가 인증됩니다. 세션의 나머지 부분에서 사용자는 이 인증된 보안 아이디와 연관되어 있습니다.

Application Server는 표 14-1에서 설명한 네 가지 유형의 인증을 지원합니다. 응용 프로그램은 배포 설명자 내에서 사용하는 인증 유형을 지정합니다. 응용 프로그램에 대한 인증 방법을 구성하기 위한 `deploytool` 사용에 대한 자세한 내용은 다음 URL의 *J2EE 1.4 Tutorial*을 참조하십시오.

`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`

표 14-1 Application Server 인증 방법

인증 방법	통신 프로토콜	설명	사용자 자격 증명 암호화
기본	HTTP(SSL 선택 사항)	서버에 내장된 팝업 로그인 대화 상자를 사용합니다.	SSL을 사용하지 않을 경우 없음
양식 기반	HTTP(SSL 선택 사항)	응용 프로그램에서 고유한 사용자 정의 로그인 및 오류 페이지를 제공합니다.	SSL을 사용하지 않을 경우 없음
클라이언트 인증서	HTTPS (SSL을 통한 HTTP)	서버는 공개 키 인증서를 사용하여 클라이언트를 인증합니다.	SSL

단일 사인 온(SSO) 검증

단일 사인 온을 사용하면 하나의 가상 서버 인스턴스의 여러 응용 프로그램이 사용자 인증 상태를 공유할 수 있습니다. 단일 사인 온을 사용할 경우 사용자가 하나의 응용 프로그램에 로그인하면 동일한 인증 정보를 요구하는 다른 응용 프로그램에 암시적으로 로그인됩니다.

단일 사인 온은 그룹을 기반으로 합니다. 배포 설명자가 동일한 *그룹*을 정의하고 동일한 인증 방법(기본, 양식, 다이제스트, 인증서)을 사용하는 모든 웹 응용 프로그램은 단일 사인 온을 공유합니다.

Application Server에 대해 정의된 가상 서버에는 기본적으로 단일 사인 온이 활성화되어 있습니다. 단일 사인 온 비활성화에 대한 자세한 내용은 "단일 사인 온(SSO) 구성"을 참조하십시오.

사용자 권한 부여

사용자가 인증되면 *권한 부여*의 수준은 수행할 수 있는 작업을 결정합니다. 사용자의 권한 부여는 *역할*을 기반으로 합니다. 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. 역할에 대한 자세한 내용은 "사용자, 그룹, 역할 및 영역 이해"를 참조하십시오.

JACC 공급자 지정

JACC(Java Authorization Contract for Containers)는 플러그 가능한 권한 부여 공급자의 인터페이스를 정의하는 J2EE 1.4 사양의 일부입니다. JACC를 사용하면 관리자는 타사 플러그인 모듈에서 권한 부여를 수행하도록 설정할 수 있습니다.

기본적으로 Application Server는 JACC 사양과 함께 컴파일되는 단순한 파일 기반의 권한 부여 엔진을 제공합니다. 다른 타사 JACC 공급자를 지정할 수도 있습니다.

JACC 공급자는 JAAS(Java Authentication and Authorization Service) API를 사용합니다. JAAS를 사용하면 서비스에서 사용자에 따라 액세스 제어를 인증하고 강제할 수 있습니다. JAAS는 표준 PAM(Pluggable Authentication Module) 프레임워크의 Java 기술 버전을 구현합니다.

인증 및 권한 부여 결정 사항 감사

Application Server는 *감사 모듈*을 통해 모든 인증 및 권한 부여 결정 사항의 감사 추적을 제공할 수 있습니다. Application Server는 기본 감사 모듈뿐만 아니라 감사 모듈을 사용자 정의할 수 있는 기능도 제공합니다. 사용자 정의 감사 모듈 개발에 대한 자세한 내용은 Application Server *Developer's Guide*를 참조하십시오. *Developer's Guide* 링크는 "자세한 내용"을 참조하십시오.

메시지 보안 구성

*메시지 보안*을 사용하면 서버는 메시지 계층에서 웹 서비스 호출 및 응답의 종단 간 인증을 수행할 수 있습니다. Application Server는 SOAP 계층에서 메시지 보안 공급자를 사용하여 메시지 보안을 구현합니다. 메시지 보안 공급자는 요청 및 응답 메시지에 필요한 인증 유형 같은 정보를 제공합니다. 지원되는 인증 유형은 다음과 같습니다.

- 아이디 비밀번호 인증을 포함하는 보낸 사람 인증
- XML 디지털 서명을 포함하는 내용 인증

두 개의 메시지 보안 공급자가 이 릴리스에 포함되어 있습니다. SOAP 계층의 인증을 위해 메시지 보안 공급자를 구성할 수 있습니다. 구성할 수 있는 공급자에는 `ClientProvider`와 `ServerProvider`가 포함됩니다.

메시지 계층 보안에 대한 지원이 플러그 가능한 인증 모듈 양식으로 Application Server와 클라이언트 컨테이너에 통합되어 있습니다. 기본적으로 메시지 계층 보안은 Application Server에서 비활성화됩니다.

전체 Application Server나 특정 응용 프로그램 또는 메소드에 대해 메시지 수준 보안을 구성할 수 있습니다. "메시지 보안 구성"에서 Application Server 수준의 메시지 보안 구성을 설명합니다. *Developer's Guide*의 *Securing Applications* 장에서 응용 프로그램 수준의 메시지 보안 구성에 대해 설명합니다.

사용자, 그룹, 역할 및 영역 이해

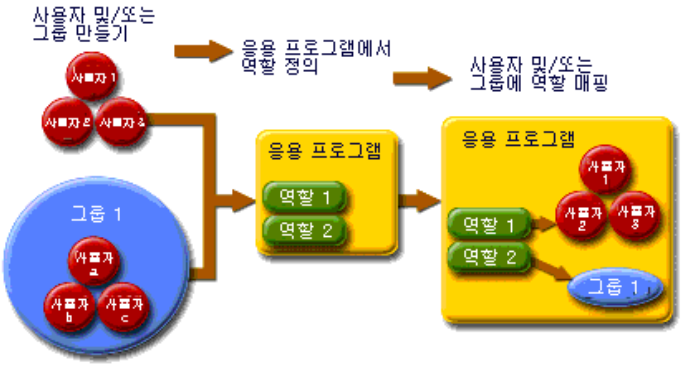
Application Server는 다음 엔티티에 대해 인증 및 권한 부여 정책을 실행합니다.

- 사용자: *Application Server에 정의된* 개별 아이디. 일반적으로 사용자는 개인, 소프트웨어 구성 요소(예: Enterprise Bean) 또는 서비스일 수도 있습니다. 인증된 사용자를 *기본*이라고도 합니다. 사용자를 *주제*라고도 합니다.
- 그룹: 공통된 특성으로 분류된, *Application Server에 정의된* 사용자 집합
- 역할: *응용 프로그램에서 정의한* 이름 지정된 권한 부여 수준. 잠금을 연 키와 역할을 비교할 수 있습니다. 많은 사용자가 키 사본을 갖고 있을 수 있습니다. 잠금에서는 액세스하는 사용자는 관계없으며, 올바른 키를 사용하는 지만 관계가 있습니다.

- 영역: 사용자와 그룹 정보 및 연관된 보안 자격 증명을 포함하는 저장소. 영역을 *보안 정책 도메인*이라고도 합니다.

주: 사용자와 그룹은 전체 Application Server에 대해 지정되는 반면, 각 응용 프로그램에서는 고유한 역할을 정의합니다. 응용 프로그램을 패키지화 및 배포할 경우 다음 그림에서 설명한 대로 응용 프로그램은 사용자/그룹 및 역할 간의 매핑을 지정합니다.

역할 매핑



사용자

*사용자*는 Application Server에 정의된 개인 또는 응용 프로그램 아이디입니다. 사용자를 그룹과 연관시킬 수 있습니다. Application Server 인증 서비스는 여러 영역의 사용자를 관리할 수 있습니다.

그룹

J2EE 그룹(또는 단순한 그룹)은 직위나 고객 프로필 같은 공통된 특성으로 분류된 사용자 범주입니다. 예를 들어, 전자 상거래 응용 프로그램의 사용자는 `customer` 그룹에 속할 수 있지만, 대량 소비자는 `preferred` 그룹에 속할 수 있습니다. 사용자를 그룹으로 범주화하면 많은 수의 사용자 액세스를 더 쉽게 제어할 수 있습니다.

역할

*역할*은 어떤 응용 프로그램과 응용 프로그램 사용자의 어떤 부분이 액세스할 수 있는지 그리고 그들이 수행할 수 있는 작업을 정의합니다. 즉, 역할은 사용자의 권한 부여 수준을 결정합니다.

예를 들어, 인사 프로그램에서 모든 직원이 전화 번호와 전자 메일 주소에 액세스할 수 있지만 급여 정보는 관리자만 액세스할 수 있습니다. 응용 프로그램은 최소한 `employee`와 `manager`의 두 가지 역할을 정의할 수 있습니다. 그리고 `manager` 역할의 사용자만 급여 정보를 볼 수 있습니다.

역할은 응용 프로그램의 기능을 정의하는 반면, 그룹은 연관된 사용자 집합이라는 점에서 역할과 사용자 그룹은 다릅니다. 예를 들어, 인사 응용 프로그램에서 `full-time`, `part-time` 및 `on-leave` 같은 그룹이 있을 수 있지만, 이 모든 그룹의 사용자는 여전히 `employee` 역할에 있습니다.

역할은 응용 프로그램 배포 설명자에 정의됩니다. 반대로 그룹은 전체 서버와 영역에 대해 정의됩니다. 응용 프로그램 개발자나 배포자는 배포 설명자의 각 응용 프로그램에 대한 하나 이상의 그룹에 역할을 매핑합니다.

영역

보안 정책 도메인 또는 *보안 도메인*이라고도 하는 *영역*은 서버가 공용 보안 정책을 정의 및 실행하는 범위입니다. 실제적인 면에서 영역은 서버가 사용자 및 그룹 정보를 저장하는 저장소입니다.

Application Server에는 `file`(초기 기본 영역), `certificate` 및 `admin-realm`의 세 가지 영역이 미리 구성되어 있습니다. `ldap`, `solaris` 또는 사용자 정의 영역을 설정할 수도 있습니다. 응용 프로그램은 배포 설명자에서 사용할 영역을 지정할 수 있습니다. 영역을 지정하지 않을 경우 Application Server는 기본 영역을 사용합니다.

`file` 영역의 경우 서버는 사용자 자격 증명을 `keyfile`이라고 하는 파일에 로컬로 저장합니다. 관리 콘솔을 사용하여 `file` 영역의 사용자를 관리할 수 있습니다. 자세한 내용은 "File 영역 사용자 관리"를 참조하십시오.

`certificate` 영역의 경우 서버는 인증서 데이터베이스에 사용자 인증서를 저장합니다. `certificate` 영역을 사용할 경우 서버는 HTTPS 프로토콜과 함께 인증서를 사용하여 웹 클라이언트를 인증합니다. 인증서에 대한 자세한 내용은 "인증서 및 SSL 소개"를 참조하십시오.

`admin-realm`은 `FileRealm`이기도 하고 `admin-keyfile`이라고 하는 파일에 관리자 자격 증명을 로컬로 저장합니다. `file` 영역에서 사용자를 관리하는 것과 같은 방법으로 관리 콘솔을 사용하여 이 영역의 사용자를 관리합니다. 자세한 내용은 "File 영역 사용자 관리"를 참조하십시오.

`ldap` 영역의 경우 서버는 Sun Java System Directory Server같은 LDAP(Lightweight Directory Access Protocol) 서버에서 사용자 자격 증명을 가져옵니다. LDAP는 공용 인터넷을 사용하는지 기업 인트라넷을 사용하는지 여부와 상관없이 조직, 개인 및 네트워크의 파일이나 장치 같은 다른 자원을 찾을 수 있게 해주는 프로토콜입니다. `ldap` 영역에서 사용자 및 그룹을 관리하는 것에 대한 자세한 내용은 LDAP 서버 설명서를 참조하십시오.

`solaris` 영역의 경우 서버는 Solaris 운영 체제에서 사용자 자격 증명을 가져옵니다. 이 영역은 Solaris 9 OS 이상에서 지원됩니다. `solaris` 영역에서 사용자 및 그룹을 관리하는 것에 대한 자세한 내용은 Solaris 설명서를 참조하십시오.

사용자 정의 영역은 관계형 데이터베이스나 타사 구성 요소 같은 사용자 자격 증명의 다른 저장소입니다. 자세한 내용은 "사용자 정의 영역 만들기" 또는 *Developer's Guide*의 *Securing Applications* 장을 참조하십시오.

인증서 및 SSL 소개

이 절에서는 다음 항목에 대해 설명합니다.

- 디지털 인증서 정보
- SSL(Secure Sockets Layer) 정보

디지털 인증서 정보

디지털 인증서(또는 인증서)는 인터넷의 사용자와 자원을 고유하게 식별하는 전자 파일입니다. 인증서는 두 엔티티 간에 안전하고 비밀을 유지하는 통신도 가능하게 해줍니다.

인증서의 종류는 여러 가지가 있는데, 개인이 사용하는 개인 인증서, SSL(Secure Socket Layer) 기술을 통해 서버와 클라이언트 간에 안전한 세션을 설정하기 위해 사용하는 서버 인증서 등이 있습니다. SSL에 대한 자세한 내용은 "SSL(Secure Sockets Layer) 정보"을 참조하십시오.

인증서는 *공개 키 암호화 도구*를 기반으로 합니다. 이 도구에서는 대상 수신자만 읽을 수 있도록 정보를 *암호화*하는 디지털 *키 쌍*(매우 긴 번호)을 사용합니다. 수신자는 정보를 *해독*하여 읽습니다.

키 쌍에는 공개 키와 개인 키가 포함됩니다. 소유자는 공개 키를 배포하여 모든 사용자가 사용할 수 있게 합니다. 그러나 소유자는 개인 키는 배포하지 않고 항상 비밀로 합니다. 키가 수학적으로 관련되어 있기 때문에 하나의 키로 암호화된 데이터는 키 쌍의 다른 키로만 해독할 수 있습니다.

인증서는 여권과 같습니다. 소유자를 식별하고 다른 중요한 정보를 제공합니다. *인증 기관*(CA)이라고 하는 신뢰할 수 있는 제 3자가 인증서를 발행합니다. 인증 기관은 여권 사무소와 유사합니다. 인증서 소유자의 아이디를 검증하고 인증서를 위조하거나 변경할 수 없도록 인증서에 서명합니다. 인증 기관이 인증서에 서명하면 소유자는 신원 증명으로 인증서를 제공하고 암호화된 기밀 통신을 설정할 수 있습니다.

가장 중요한 점은 인증서가 소유자의 공개 키를 소유자의 아이디에 바인드한다는 점입니다. 여권이 소유자에 대한 개인 정보와 사진을 함께 제공하는 것과 같이 인증서는 소유자에 대한 정보에 공개 키를 바인드합니다.

공개 키 외에 인증서에는 대개 다음과 같은 정보가 포함되어 있습니다.

- 소유자의 이름 및 인증서를 사용한 웹 서버의 URL 같은 기타 식별 정보 또는 개인의 전자 메일 주소
- 인증서를 발행한 인증 기관 이름
- 만료일

디지털 인증서는 x.509 형식의 기술 사양으로 관리됩니다. `certificate` 영역의 사용자 아이디를 확인하기 위해 인증 서비스는 X.509 인증서의 공통 이름 필드를 기본 이름으로 사용하여 X.509 인증서를 검증합니다.

인증서 체인 정보

웹 브라우저에는 브라우저가 자동으로 신뢰하는 *루트* CA 인증서 집합이 사전 구성되어 있습니다. 모든 인증서에는 자신의 유효성을 검증하기 위한 *인증서 체인*이 함께 제공되어야 합니다. 인증서 체인은 연속적인 CA에서 발행하여 루트 CA 인증서로 종료되는 일련의 인증서입니다.

인증서를 처음 생성한 경우에는 *자체 서명된* 인증서입니다. 자체 서명된 인증서는 발급자(서명자)가 주제(공개 키를 인증서에서 인증하는 엔티티)와 동일한 인증서입니다. 소유자가 인증서 서명 요청(CSR)을 인증 기관에 전송한 다음 응답을 가져오면, 자체 서명된 인증서는 인증서 체인으로 대체됩니다. 체인 맨 아래에는 주제의 공개 키를 인증하는 인증 기관에서 발행한 인증서(응답)가 있습니다. 체인의 다음 인증서는 CA의 공개 키를 인증하는 인증서입니다. 대개는 자체 서명된 인증서(즉, 해당 공개 키를 인증하는 인증 기관의 인증서)이고 체인의 마지막 인증서입니다.

다른 경우 CA는 인증서 체인을 반환할 수 있습니다. 이 경우 인증서 체인의 맨 아래 인증서는 동일하지만(키 항목의 공개 키를 인증하는 CA에서 서명한 인증서) 체인의 두 번째 인증서는 CSR을 전송한 CA의 공개 키를 인증하는 다른 CA에서 서명한 인증서입니다. 그런 다음 체인의 다음 인증서는 두 번째 CA의 키를 인증하는 인증서입니다. 자체 서명된 *루트* 인증서에 도달할 때까지 이런 식으로 계속됩니다. 체인의 각 인증서(첫째 인증서 제외)는 체인의 이전 인증서의 서명자 공개 키를 인증합니다.

SSL(Secure Sockets Layer) 정보

*SSL(Secure Sockets Layer)*은 인터넷 통신 및 트랜잭션을 보안하기 위한 가장 일반적인 표준입니다. 웹 응용 프로그램은 서버와 클라이언트간 안전한 기밀 통신을 보장하기 위해 디지털 인증서를 사용하는 HTTPS(SSL상의 HTTP)를 사용합니다. SSL 연결에서 클라이언트와 서버는 모두 데이터를 전송하기 전에 암호화한 다음 요청 시 해독합니다.

웹 브라우저(클라이언트)에서 보안 사이트에 연결할 경우 *SSL 핸드셰이크*가 발생합니다.

- 브라우저는 대개 `http` 대신 `https`로 시작하는 URL을 요청하여 보안 세션을 요청하는 네트워크에서 메시지를 전송합니다.
- 서버는 공개 키를 포함하는 인증서를 전송하여 응답합니다.
- 브라우저는 서버의 인증서가 유효한지 그리고 인증서가 브라우저의 데이터베이스에 있고 신뢰할 수 있는 인증 기관에서 서명한 것인지 검증합니다. 그리고 CA 인증서가 만료되지 않았는지 확인합니다.
- 인증서가 유효하면 브라우저는 고유한 일회용 *세션 키*를 생성하고 이를 서버의 공개 키와 함께 암호화합니다. 브라우저는 암호화된 세션 키를 서버에 전송하여 둘 다 복사본을 갖고 있도록 합니다.
- 서버는 개인 키를 사용하여 메시지를 해독하고 세션 키를 복구합니다.

핸드셰이크 후 클라이언트는 웹 사이트의 아이디를 검증했고 클라이언트와 웹 서버만 세션 키의 사본을 갖고 있습니다. 이 때부터 클라이언트와 서버는 세션 키를 사용하여 서로 간의 모든 통신을 암호화합니다. 따라서 이 통신은 보안이 안전합니다.

SSL 표준의 최신 버전을 TLS(Transport Layer Security)라고 합니다. Application Server는 SSL(Secure Sockets Layer) 3.0 및 TLS(Transport Layer Security) 1.0 암호화 프로토콜을 지원합니다.

SSL을 사용하려면 Application Server에 외부 인터페이스에 대한 인증서나 보안 연결을 승인하는 IP 주소가 있어야 합니다. 디지털 인증서가 설치되지 않으면 대부분 웹 서버의 HTTPS 서비스가 실행되지 않습니다. 웹 서버가 SSL에 사용할 수 있는 디지털 인증서를 설정하려면 "서버 인증서 생성"에서 설명한 절차를 사용합니다.

암호화 정보

*암호화*는 암호화나 해독에 사용한 암호화 알고리즘입니다. SSL 및 TLS 프로토콜은 서버와 클라이언트가 서로간에 인증하고, 인증서를 전송하며, 세션 키를 설정하기 위해 사용한 다양한 암호화를 지원합니다.

일부 암호는 다른 암호보다 더 강력하고 더 안전합니다. 클라이언트와 서버는 서로 다른 암호화 그룹을 지원할 수 있습니다. SSL3 및 TLS 프로토콜에서 암호화를 선택합니다. 세션 연결 중에 클라이언트와 서버는 서로가 통신을 위해 설정한 더 강력한 암호화 사용을 승인하므로 대개 모든 암호화를 충분히 사용할 수 있습니다.

이름 기반의 가상 호스트 사용

보안 응용 프로그램에 대해 이름 기반의 가상 호스트를 사용하는 것은 문제가 될 수 있습니다. 이는 SSL 프로토콜의 설계 한계입니다. 클라이언트 브라우저가 서버 인증서를 승인하는 SSL 핸드셰이크는 HTTP 요청에 액세스하기 전에 발생해야 합니다. 따라서 가상 호스트 이름을 포함하는 요청 정보를 인증 전에 확인할 수 없습니다. 그러므로 단일 IP 주소에 복수 인증서를 지정할 수 없습니다.

단일 IP 주소의 모든 가상 호스트를 동일한 인증서에 대해 인증해야 할 경우 복수 가상 호스트를 추가해도 서버의 정상 SSL 작업을 방해하지 않습니다. 그러나 대부분의 브라우저는 (공식적인 CA 서명된 인증서에 우선적으로 적용할 수 있는 경우) 서버의 도메인 이름을 인증서에 나열된 도메인 이름과 비교합니다. 도메인 이름이 일치하지 않을 경우 이 브라우저에서 경고를 표시합니다. 일반적으로 주소 기반 가상 호스트만 작업 환경에서 SSL과 같이 사용됩니다.

방화벽 정보

*방화벽*은 둘 이상의 네트워크간 데이터 흐름을 제어하고 네트워크간 링크를 관리합니다. 방화벽은 하드웨어 및 소프트웨어 요소 둘 다로 구성될 수 있습니다. 이 절에서는 일반 방화벽 구조와 방화벽 구성을 설명합니다. 여기에서는 주로 Application Server에 관련된 것을 다룹니다. 특정 방화벽 기술에 대한 자세한 내용은 방화벽 공급업체의 설명서를 참조하십시오.

일반적으로 클라이언트가 필요한 TCP/IP 포트에 액세스할 수 있도록 방화벽을 구성합니다. 예를 들어, HTTP listener가 포트 8080에서 작동할 경우 포트 8080에서만 HTTP 요청을 허용하도록 방화벽을 구성합니다. 마찬가지로 포트 8181에 대한 HTTPS 요청을 설정한 경우 포트 8181에서 HTTPS 요청을 허용하도록 방화벽을 구성해야 합니다.

인터넷에서 EJB 모듈로 직접 RMI-IIOP(Remote Method Invocations over Internet Inter-ORB Protocol) 액세스가 필요한 경우 RMI-IIOP listener 포트도 엽니다. 그러나 보안 위험이 생기기 때문에 열지 않는 것이 좋습니다.

이중 방화벽 구조의 경우 HTTP 및 HTTPS 트랜잭션을 허용하도록 외부 방화벽을 구성해야 합니다. 방화벽 뒤에서 Application Server와 통신하려면 HTTP 서버 플러그인을 허용하도록 내부 방화벽을 구성해야 합니다.

관리 콘솔을 사용하여 보안 관리

관리 콘솔은 보안의 다음 측면을 관리하기 위한 수단을 제공합니다.

- 서버 보안 설정
- 영역 및 파일 영역 사용자
- JACC 공급자
- 감사 모듈
- 메시지 보안
- HTTP 및 IIOP Listener 보안
- 관리 서비스 보안
- 보안 맵

서버 보안 설정

보안 설정 페이지에서 기본 영역, 익명 역할, 기본 아이디 및 비밀번호 지정을 포함한 전체 서버에 대한 등록 정보를 설정합니다. 자세한 내용은 "보안 설정 구성"을 참조하십시오.

영역 및 파일 영역 사용자

영역의 개념은 "사용자, 그룹, 역할 및 영역 이해"에서 소개합니다. 관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

- 새로운 영역 만들기
- 기존 영역 삭제
- 기존 영역의 구성 수정
- `file` 영역의 사용자 추가, 수정 및 삭제

- 기본 영역 설정

이 작업에 대한 자세한 내용은 "영역에 대한 관리 콘솔 작업"을 참조하십시오.

JACC 공급자

JACC 공급자는 "JACC 공급자 지정"에서 소개합니다. 관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

새로운 JACC 공급자 추가

- 기존 JACC 공급자 삭제 또는 수정

이 작업에 대한 자세한 내용은 "JACC 공급자에 대한 관리 콘솔 작업"을 참조하십시오.

감사 모듈

감사 모듈은 "인증 및 권한 부여 결정 사항 감사"에서 소개합니다. 감사는 오류나 보안 위반 같은 중요한 이벤트를 나중에 검토하기 위해 기록하는 방법입니다. 모든 인증 이벤트는 Application Server 로그에 기록됩니다. 전체 액세스 로그는 Application Server 액세스 이벤트의 순차적인 추적을 제공합니다.

관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

- 새로운 감사 모듈 추가
- 기존 감사 모듈 삭제 또는 수정

이 작업에 대한 자세한 내용은 "감사 모듈에 대한 관리 콘솔 작업"을 참조하십시오.

메시지 보안

메시지 보안 개념은 "메시지 보안 구성"에서 소개합니다. 관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

- 메시지 보안 사용
- 메시지 보안 공급자 구성
- 기존 메시지 보안 구성 또는 공급자 삭제 또는 구성

이 작업에 대한 자세한 내용은 "메시지 보안 구성"을 참조하십시오.

HTTP 및 IIOP Listener 보안

HTTP 서비스의 모든 가상 서버는 하나 이상의 *HTTP listener*를 통한 네트워크 연결을 제공합니다. HTTP 서비스 및 HTTP listener에 대한 일반적인 정보는 "HTTP 서비스"를 참조하십시오.

Application Server는 IIOP(Internet Inter-Orb Protocol)를 사용하여 네트워크에서 통신하는 CORBA(Common Object Request Broker Architecture) 객체를 지원합니다. *IIOP listener*는 EJB의 원격 클라이언트와 다른 CORBA 기반 클라이언트에서 들어오는 연결을 승인합니다. IIOP listener에 대한 일반적인 내용은 "IIOP Listener"를 참조하십시오.

관리 콘솔을 사용하여 다음 작업을 수행합니다.

- 새로운 HTTP 또는 IIOP listener를 만들고 이들이 사용하는 보안을 지정합니다.
- 기존 HTTP 또는 IIOP listener에 대한 보안 설정을 수정합니다.

이 작업에 대한 자세한 내용은 "Listener 및 JMX 커넥터에 대한 관리 콘솔 작업"을 참조하십시오.

관리 서비스 보안

관리 서비스는 서버 인스턴스가 일반 인스턴스인지 DAS(Domain Administration Server) 인지 아니면 둘의 조합인지 여부를 확인합니다. 관리 서비스를 사용하여 JSR-160 호환 원격 JMX 커넥터를 구성합니다. 이 커넥터는 DAS(Domain Administration Server)와 노드 에이전트 간의 통신을 처리합니다. 이 때 노드 에이전트는 원격 서버 인스턴스의 경우 호스트 시스템에서 서버 인스턴스를 관리합니다.

관리 콘솔을 사용하여 다음 작업을 수행합니다.

- 관리 서비스 관리
- JMX 커넥터 편집
- JMX 커넥터에 대한 보안 설정 수정

이 작업에 대한 자세한 내용은 "관리 서비스의 JMX 커넥터에 대한 보안 구성"을 참조하십시오.

보안 맵

커넥터 연결 풀에 대한 보안 맵 개념은 "보안 맵 정보"에서 소개합니다. 관리 콘솔을 사용하여 다음 작업을 수행할 수 있습니다.

- 보안 맵을 기존 커넥터 연결 풀에 추가
- 기존 보안 맵 삭제 또는 구성

이 작업에 대한 자세한 내용은 "커넥터 연결 풀에 대한 관리 콘솔 작업"을 참조하십시오.

보안에 대한 관리 콘솔 작업

보안 설정 구성

관리 콘솔의 보안 페이지를 사용하면 시스템 차원의 다양한 보안 설정을 설정할 수 있습니다.

이 설정을 편집하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 선택합니다.

   보안 페이지가 표시됩니다.
4. 필요하면 값을 수정합니다. 일반 보안 옵션은 표 14-2에서 설명합니다.

표 14-2 일반 보안 설정

설정	설명
감사 로깅	감사 로깅을 활성화하려면 선택합니다. 활성화한 경우 서버가 감사 모듈 설정에 지정된 모든 감사 모듈을 로드 및 실행합니다. 비활성화한 경우 서버가 감사 모듈에 액세스하지 않습니다. 기본적으로 비활성화되어 있습니다.
기본 영역	서버가 인증에 사용하는 활성(기본) 영역입니다. 응용 프로그램이 배포 설명자에 다른 영역을 지정하지 않는 한 이 영역을 사용합니다. 모든 구성된 영역이 목록에 표시됩니다. 초기 기본 영역은 file 영역입니다.
익명 역할	기본 역할이나 익명 역할의 이름입니다. 모든 사용자에게 익명 역할이 지정됩니다. 응용 프로그램은 배포 설명자에서 이 역할을 사용하여 모든 사용자에게 권한을 부여합니다.

표 14-2 일반 보안 설정 (*계속*)

설정	설명
기본 기본값	기본 아이디를 지정합니다. 기본값을 제공하지 않은 경우 서버는 이 값을 사용합니다. 이 필드에 값을 입력한 경우 기본 비밀번호 기본값 필드에 해당 값을 입력합니다. 정상적인 서버 작업의 경우 이 속성이 필요하지 않습니다.
기본 비밀번호 기본값	기본 기본값 필드에 지정된 기본 기본값의 비밀번호입니다. 정상적인 서버 작업의 경우 이 속성이 필요하지 않습니다.
JACC	구성된 JACC 공급자의 클래스 이름입니다. JACC 공급자 추가에 대한 자세한 내용은 "JACC 공급자 작성"을 참조하십시오.
감사 모듈	쉼표로 구분한 감사 모듈 공급자 클래스 목록입니다. 여기에 나열된 모듈은 이미 구성되어 있어야 합니다. 감사 로깅을 활성화한 경우 이 설정에서 감사 모듈을 나열해야 합니다. 기본적으로 서버는 `default`라고 하는 감사 모듈을 사용합니다. 새로운 감사 모듈 만들기에 대한 자세한 내용은 "감사 모듈 만들기"를 참조하십시오.

5. 추가 등록 정보 섹션에서 Java Virtual Machine(JVM)에 전달할 추가 등록 정보를 입력합니다.

   유효한 등록 정보는 기본 영역 필드에서 선택한 영역 유형에 따라 다릅니다. 유효한 등록 정보는 다음 절에서 설명합니다.

   - File 영역과 admin-realm 영역 편집
   - 인증서 영역 편집
   - Solaris 영역 만들기
   - Ldap 영역 만들기
   - 사용자 정의 영역 만들기

6. 변경 사항을 저장하려면 저장을 선택하거나, 기본값을 복구하려면 기본값 로드를 선택합니다.

관리 도구에 대한 액세스 제어

`asadmin` 그룹의 사용자만 관리 콘솔과 `asadmin` 명령줄 유틸리티에 액세스할 수 있습니다.

사용자에게 이 관리 도구에 대한 액세스를 제공하려면 `admin-realm`의 `asadmin` 그룹에 이 관리 도구를 추가합니다. 이를 수행하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.

2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 영역 노드를 확장합니다.
5. admin-realm 노드를 선택합니다.
6. 사용자를 추가, 수정, 삭제하려면 사용자 관리 버튼을 누릅니다.

   설치 후 처음에는 설치 중 입력한 관리자 이름과 비밀번호가 admin-keyfile이라는 파일에 나열됩니다. 기본적으로 이 사용자는 Application Server를 수정할 수 있는 권한을 제공하는 asadmin 그룹에 속합니다. 사용자에게 Application Server에 대한 관리자 권한을 부여할 경우에만 사용자를 이 그룹에 할당합니다.

   사용자를 admin-realm 영역에 추가하고 사용자를 asadmin 이외의 그룹에 지정한 경우 사용자 정보는 admin-keyfile이라는 파일에 계속 기록되지만 사용자는 관리 도구나 file 영역의 응용 프로그램에 액세스할 수 있는 권한이 없습니다.
7. 새로 만들기를 눌러 새로운 사용자를 admin-realm 영역에 추가합니다.
8. 사용자 아이디, 비밀번호 및 그룹 목록 필드에 정확한 정보를 입력합니다. 사용자에게 Application Server를 수정할 수 있는 권한을 부여하려면 그룹 목록에서 asadmin 그룹을 포함시킵니다.
9. 확인을 눌러 이 사용자를 admin-realm 영역에 추가하거나 취소를 눌러 저장하지 않고 중지합니다.

영역에 대한 관리 콘솔 작업

영역 만들기

Application Server에는 `file`, `certificate` 및 `admin-realm`의 세 가지 영역이 미리 구성되어 있습니다. `ldap`, `solaris` 및 사용자 정의 영역을 만들 수도 있습니다. 일반적으로 서버에는 각 유형의 한 가지 영역이 있는데, Application Server의 경우 `file` 및 `admin-realm`의 두 가지 파일 영역이 있습니다. 이는 두 가지 다른 용도에 사용한 동일한 유형의 두 가지 영역입니다. 시스템의 각 가상 서버에 대해 다른 인증서 데이터베이스를 보유할 수도 있습니다.

보안 영역을 만들려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 영역 노드를 선택합니다.
5. 영역 페이지에서 새로 만들기를 누릅니다.

   영역 만들기 페이지가 표시됩니다.
6. 이름 필드에 영역 이름을 입력합니다.

7. 만들 영역의 클래스 이름을 지정합니다. 유효한 선택 항목이 표 14-3에 표시되어 있습니다.

표 14-3 영역 클래스 이름에 유효한 값

영역 이름	클래스 이름
`file`	`com.sun.enterprise.security.auth.realm.file.FileRealm`
`certificate`	`com.sun.enterprise.security.auth.realm.certificate.CertificateRealm`
`ldap`	`com.sun.enterprise.security.auth.realm.ldap.LDAPRealm`
`solaris`	`com.sun.enterprise.security.auth.realm.solaris.SolarisRealm`
custom	로그인 영역 클래스의 이름

8. 영역에 대한 필수 등록 정보와 원하는 선택적인 등록 정보를 추가합니다.

   등록 정보를 추가하려면 다음 작업을 수행합니다.

   a. 등록 정보 추가를 누릅니다.

   b. 이름 필드에 등록 정보 이름을 입력합니다.

   - `file` 영역 등록 정보에 대한 설명은 "File 영역과 admin-realm 영역 편집"을 참조하십시오.
   - `certificate` 영역 등록 정보에 대한 설명은 "인증서 영역 편집"을 참조하십시오.
   - `ldap` 영역 등록 정보에 대한 설명은 "Ldap 영역 만들기"를 참조하십시오.
   - `solaris` 영역 등록 정보에 대한 설명은 "Solaris 영역 만들기"를 참조하십시오.
   - custom 영역 등록 정보에 대한 설명은 "사용자 정의 영역 만들기"를 참조하십시오.

   c. 값 필드에 해당 등록 정보의 값을 입력합니다.

9. 확인을 누릅니다.

해당 `asadmin` 명령: `create-auth-realm`

Ldap 영역 만들기

`Ldap` 영역은 LDAP 서버의 정보를 사용하여 인증을 수행합니다. 사용자 정보에는 아이디, 비밀번호 및 사용자가 속한 그룹이 포함됩니다. LDAP 영역을 사용하려면 LDAP 디렉토리에 사용자 및 그룹을 이미 정의했어야 합니다.

LDAP 영역을 만들려면 새로운 영역을 추가하는 데 필요한 "영역 만들기" 단계를 수행하고 표 14-4에 표시된 등록 정보를 추가합니다.

표 14-4 Ldap 영역의 필수 등록 정보

등록 정보 이름	설명	값
디렉토리	디렉토리 서버의 LDAP URL입니다.	`ldap://`*hostname*`:`*port* 양식의 *LDAP URL* 예: `ldap://myldap.foo.com:389`.
base-dn	트리 범위 검색을 수행한 후 사용자 데이터 위치에 대한 기본 고유 이름(DN)입니다. 이 때 사용자 데이터 위치는 사용자 데이터 이상의 어떤 수준이나 가능합니다. 검색 트리를 더 작게 할수록 성능이 더 좋아집니다.	검색의 도메인. 예: `dc=siliconvalley, dc=BayArea, dc=sun, dc=com`.
jaas-context	해당 영역을 사용하는 로그인 모듈 유형입니다.	`ldapRealm`이어야 합니다.

표 14-5에는 ldap 영역에 대한 선택적인 등록 정보가 표시됩니다.

표 14-5 ldap 영역에 대한 선택적인 등록 정보

등록 정보 이름	설명	기본값
search-filter	사용자를 찾기 위해 사용하는 검색 필터입니다.	`uid=%s`(`%s`는 주제 이름으로 확장)
group-base-dn	그룹 데이터의 위치에 대한 기본 DN입니다.	`base-dn`과 동일하지만 필요한 경우 조정할 수 있습니다.
group-search-filter	사용자의 그룹 구성원을 찾기 위한 검색 필터입니다.	`uniquemember=%d`(`%d`은(는) 사용자 요소 DN으로 확장)
group-target	그룹 이름 항목을 포함하는 LDAP 속성 이름입니다.	CN
search-bind-dn	검색 필터 조회를 수행하기 위해 디렉토리 인증에 사용한 선택적인 DN입니다. 익명의 검색을 허용하지 않는 디렉토리에만 필요합니다.	
search-bind-password	`search-bind-dn`에 지정한 DN의 LDAP 비밀번호입니다.	

예:

예를 들어, LDAP 디렉토리에 LDAP 사용자 Joe Java가 다음과 같이 정의되어 있다고 가정합니다.

```
uid=jjava,ou=People,dc=acme,dc=com
uid=jjava
givenName=joe
objectClass=top
objectClass=person
objectClass=organizationalPerson
objectClass=inetorgperson
sn=java
cn=Joe Java
```

Ldap 영역을 만들거나 편집할 때 예 코드를 사용하면 표 14-6에 표시된 값을 입력할 수 있습니다.

표 14-6 예 ldap 영역 값

등록 정보 이름	등록 정보 값
directory	서버에 대한 LDAP URL 예: ldap://ldap.acme.com:389
base-dn	ou=People,dc=acme,dc=com. 더 높게 루트 지정할 수 있습니다(예: dc=acme, dc=com). 그러나 검색이 트리의 더 많은 부분을 통과할 수록 성능이 떨어집니다.
jaas-context	ldapRealm

Solaris 영역 만들기

Solaris 영역은 시스템 구성에서 확인한 대로 기본 Solaris 사용자 데이터베이스에서 사용자 및 그룹 정보를 가져옵니다. Solaris 영역은 인증을 위해 기본 PAM 기반구조를 호출합니다. 구성된 PAM 모듈에 루트 권한이 필요할 경우 이 영역을 사용하려면 도메인을 루트로 실행해야 합니다. 자세한 내용은 Solaris 설명서에서 보안 서비스 부분을 참조하십시오.

Solaris 영역에는 사용할 로그인 모듈 유형을 지정하는 하나의 필수 등록 정보 jaas-context가 있습니다. 등록 정보 값은 solarisRealm이어야 합니다.

주: solaris 영역은 Solaris 9 이상만 지원합니다.

사용자 정의 영역 만들기

네 개의 기본 제공 영역 외에 사용자 데이터를 다른 방식(예: 관계형 데이터베이스)으로 저장하는 사용자 정의 영역을 만들 수 있습니다. 사용자 정의 영역의 개발은 이 문서에서 다루지 않습니다. 자세한 내용은 Application Server *Developer's Guide*의 Securing Applications 장을 참조하십시오.

관리자로서 기본적으로 알아야 할 사항은 JAAS(Java Authentication and Authorization Service) 패키지에서 파생된 `LoginModule`이라고 하는 클래스에서 사용자 정의 영역을 구현한다는 점입니다.

Application Server가 사용자 정의 영역을 사용하도록 구성하려면 다음 작업을 수행합니다.

1. 사용자 정의 영역 이름과 `LoginModule` 클래스 이름을 입력하여 "영역 만들기"의 절차를 수행합니다. 사용자 정의 영역에 고유한 이름(예: `myCustomRealm`)을 사용할 수 있습니다.
2. 표 14-7에 표시된 등록 정보를 추가합니다.

표 14-7 사용자 정의 영역에 유효한 등록 정보

등록 정보 이름	등록 정보 값
jaas-context	`LoginModule` 클래스 이름(예: `simpleCustomRealm`)입니다.
auth-type	영역의 설명(예: "간단한 사용자 정의 영역 예")입니다.

3. 확인을 누릅니다.
4. 다음과 같이 도메인의 로그인 구성 파일 *install_dir*`/domains/`*domain_name*`/config/login.conf`를 편집하고 파일 끝에 JAAS `LoginModule`의 정규화된 클래스 이름을 추가합니다.

```
realmName {
    fully-qualified-LoginModule-classname required;
};
```

예를 들어,

```
myCustomRealm {
    com.foo.bar.security.customrealm.simpleCustomLoginModule required;
};
```

5. `LoginModule` 클래스와 모든 종속 클래스를 *install_dir*`/domains/`*domain_name*`/lib/classes` 디렉토리에 복사합니다.
6. 콘솔에 다시 시작해야 함이 표시되면 서버를 다시 시작합니다.

7. 영역이 제대로 로드되었는지 확인합니다.

   install_dir/domains/*domain_name*/logs/server.log를 확인하여 서버가 영역을 로드했는지 확인합니다. 서버가 영역의 init() 메소드를 호출해야 합니다.

영역 편집

영역을 편집하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 영역 노드를 확장합니다.
5. 기존 영역의 이름을 선택합니다.

   영역 편집 페이지가 표시됩니다.
6. 기존 등록 정보와 등록 정보 값을 원하는 대로 편집합니다.

   File 영역 등록 정보에 대한 자세한 내용은 "File 영역과 admin-realm 영역 편집"을 참조하십시오. File 영역의 사용자를 관리하려면 사용자 관리 버튼을 누릅니다. 자세한 내용은 "File 영역 사용자 관리"를 참조하십시오.

   Certificate 영역 등록 정보에 대한 자세한 내용은 "인증서 영역 편집"을 참조하십시오.
7. 다른 등록 정보를 추가하려면 등록 정보 추가 버튼을 누릅니다. 페이지에 새로운 행이 표시됩니다. 유효한 등록 정보 이름과 등록 정보 값을 입력합니다. 구성할 수 있는 선택적인 등록 정보에 대한 설명은 다음 표를 참조하십시오.
   - 표 14-4, Ldap 영역의 필수 등록 정보
   - 표 14-5, ldap 영역에 대한 선택적인 등록 정보
   - 표 14-7, 사용자 정의 영역에 유효한 등록 정보
   - 표 14-8, File 영역의 필수 등록 정보

- 표 14-9, certificate 영역의 선택적 등록 정보

8. 저장을 눌러 변경 사항을 저장합니다.

File 영역과 admin-realm 영역 편집

서버는 모든 사용자, 그룹 및 비밀번호 정보를 file 영역의 경우 keyfile이라는 파일에, admin-realm의 경우 admin-keyfile이라는 파일에 보관합니다. 두 경우 모두 file 등록 정보는 키 파일의 위치를 지정합니다. 표 14-8에서는 file 영역의 필수 등록 정보를 설명합니다.

표 14-8 File 영역의 필수 등록 정보

등록 정보 이름	설명	기본값
file	키 파일의 전체 경로 및 이름입니다.	*install_dir*/domains/*domain-name*/config/keyfile
jaas-context	해당 영역을 사용하는 로그인 모듈 유형입니다.	fileRealm만이 유효한 값입니다.

keyfile은 처음에는 비어 있어서 file 영역을 사용하려면 사용자를 추가해야 합니다. 자세한 내용은 "File 영역 사용자 관리"를 참조하십시오.

admin-keyfile에는 처음에 관리자 이름, 암호화된 형식의 관리자 비밀번호 및 사용자가 속한 그룹(기본적으로 asadmin)이 포함되어 있습니다. 사용자를 admin-realm에 추가하는 방법에 대한 자세한 내용은 "관리 도구에 대한 액세스 제어"를 참조하십시오.

주: admin-realm에 있는 asadmin 그룹의 사용자는 관리 콘솔과 asadmin 도구를 사용할 수 있는 권한이 있습니다. 서버 관리 권한이 있는 사용자만 이 그룹에 추가합니다.

NSS(Network Security Services)를 사용하여 사용자 관리

Enterprise Edition의 경우에만 "File 영역 사용자 관리"에서 설명한 대로 관리 콘솔을 사용하여 사용자를 관리하거나 NSS 도구를 사용하여 사용자를 관리할 수 있습니다. NSS(Network Security Services)는 보안을 사용하는 클라이언트 및 서버 응용 프로그램의 교차 플랫폼 개발을 지원하기 위해 설계된 라이브러리 집합입니다. NSS가 빌드된 응용 프로그램은 SSL v2 및 v3, TLS, PKCS #5, PKCS #7, PKCS #11, PKCS #12, S/MIME, X.509 v3 인증서 및 기타 보안 표준을 지원할 수 있습니다. 자세한 내용은 다음 URL 링크를 참조하십시오.

- http://www.mozilla.org/projects/security/pki/nss/의 NSS(Network Security Services)

- http://www.mozilla.org/projects/security/pki/nss/tools/의 NSS 보안 도구
- http://www.mozilla.org/projects/security/pki/nss/overview.html의 NSS 개요

File 영역 사용자 관리

관리 콘솔을 사용하여 file 영역 사용자를 관리합니다. File 영역의 사용자와 그룹은 file 등록 정보로 위치가 지정된 키 파일에 나열됩니다.

주: 이 단계를 사용하여 admin-realm을 포함한 모든 파일 영역에 사용자를 추가할 수도 있습니다. 이 절에서 참조한 file 영역 대신 대상 영역 이름으로 대체하기만 하면 됩니다.

File 영역의 사용자는 공통된 특성으로 분류한 사용자 범주인 *J2EE 그룹*에 속할 수 있습니다. 예를 들어, 전자 상거래 응용 프로그램의 고객은 CUSTOMER 그룹에 속할 수 있지만, 대량 소비자는 PREFERRED 그룹에 속할 수 있습니다. 사용자를 그룹으로 범주화하면 많은 수의 사용자 액세스를 더 쉽게 제어할 수 있습니다.

Application Server 설치 후 처음에는 설치 중에 입력한 관리자만이 유일한 사용자입니다. 기본적으로 이 사용자는 Application Server를 수정할 수 있는 권한을 제공하는 admin-realm 영역의 asadmin 그룹에 속합니다. 이 그룹에 지정된 사용자는 관리자 특권을 갖게 됩니다. 즉, asadmin 도구와 관리 콘솔에 액세스할 수 있는 권한을 갖습니다.

File 영역 사용자를 관리하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 영역 노드를 확장합니다.
5. File 노드를 선택합니다.
6. 사용자를 추가, 수정, 삭제하려면 사용자 관리 버튼을 누릅니다.

   파일 사용자 페이지가 표시됩니다. 이 페이지에서 다음 작업을 수행합니다.

   - 사용자 추가

- 사용자 편집
- 사용자 삭제

사용자 추가

파일 사용자 페이지에서 다음 단계를 수행하여 새로운 사용자를 추가합니다.

1. 새로 만들기를 눌러 새로운 사용자를 `file` 영역에 추가합니다.
2. 파일 사용자 페이지에서 다음 정보를 입력합니다.
   - **사용자 아이디***(필수)* - 아이디입니다.
   - **비밀번호***(필수)* - 사용자의 비밀번호입니다.
   - **비밀번호 확인***(필수)* - 검증을 위해 다시 입력한 사용자 비밀번호입니다.
   - **그룹 목록***(선택 사항)* - 쉼표로 구분한 사용자가 속하는 그룹 목록입니다. 이 그룹을 다른 곳에 정의할 필요는 없습니다.
3. 확인을 눌러 이 사용자를 `file` 영역의 사용자 목록에 추가합니다. 취소를 눌러 저장하지 않고 종료합니다.

해당 `asadmin` 명령: `create-file-user`

사용자 편집

파일 사용자 페이지에서 다음 단계를 수행하여 사용자 정보를 변경합니다.

1. 사용자 아이디 열에서 수정할 아이디를 누릅니다.

   파일 영역 사용자 편집 페이지가 표시됩니다.
2. 비밀번호 필드와 비밀번호 확인 필드에 새로운 비밀번호를 입력하여 사용자의 비밀번호를 변경합니다.
3. 그룹 목록 필드에서 그룹을 추가하거나 삭제하여 사용자가 속한 그룹을 변경합니다. 그룹 이름을 쉼표로 구분합니다. 그룹을 이전에 정의할 필요는 없습니다.
4. 저장을 눌러 이 사용자를 `file` 영역의 사용자 목록에 저장합니다. 닫기를 눌러 저장하지 않고 종료합니다.

사용자 삭제

파일 사용자 페이지에서 다음 단계를 수행하여 사용자를 삭제합니다.

1. 삭제할 아이디 왼쪽에 있는 확인란을 선택합니다.
2. 삭제를 누릅니다.
3. 닫기를 눌러 영역 편집 페이지로 돌아갑니다.

해당 asadmin 명령: delete-file-user

인증서 영역 편집

Certificate 영역은 SSL 인증을 지원합니다. 이 영역은 Application Server의 보안 컨텍스트로 사용자 아이디를 설정하고, truststore 및 keystore 파일의 암호로 확인된 클라이언트 인증서에서 얻은 사용자 데이터로 이를 채웁니다("인증서 파일 정보" 참조). certutil을 사용하여 이 파일에 사용자를 추가합니다. Certificate 영역을 사용하여 J2EE 컨테이너는 사용자 인증서의 고유 이름(DN)을 기반으로 한 인증 처리를 수행합니다. 고유 이름(DN)은 인증서가 공개 키를 식별하는 엔티티 이름입니다. 이 이름에서는 X.500 표준을 사용하므로 인터넷에서 고유합니다. Keystores 및 truststores에 대한 자세한 내용은 "CertUtil 유틸리티 정보"의 certutil 설명서를 참조하십시오.

표 14-9에서는 certificate 영역에 대한 선택적 등록 정보를 나열합니다.

표 14-9 certificate 영역의 선택적 등록 정보

등록 정보	설명
assign-groups	쉼표로 구분한 그룹 이름 목록. 유효한 인증서를 제공하는 모든 클라이언트가 이 그룹에 할당됩니다. 예를 들면, 사용자 그룹의 이름인 employee,manager입니다.
jaas-context	해당 영역을 사용하는 로그인 모듈 유형입니다. Certificate 영역의 경우 값은 certificateRealm이어야 합니다.

상호 인증 구성

- 모든 응용 프로그램의 상호 인증 활성화
- 응용 프로그램에서 상호 SSL 인증 활성화

상호 인증의 경우 서버측 인증과 클라이언트측 인증이 모두 활성화되어 있습니다. 상호 인증을 테스트하려면 유효한 인증서를 가진 클라이언트가 있어야 합니다. 상호 인증에 대한 자세한 내용은 다음 위치에 있는 *J2EE 1.4 Tutorial*의 *Security* 장을 참조하십시오.

http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

모든 응용 프로그램의 상호 인증 활성화

Application Server는 HTTPS 인증을 위해 certificate 영역을 사용합니다.

이 영역을 사용하는 모든 응용 프로그램에 대해 상호 인증을 지정하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.

2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 영역 노드를 확장합니다.
5. Certificate 영역을 선택합니다.
6. 등록 정보 추가 버튼을 누릅니다.
   - 이름 필드에 clientAuth를 입력합니다.
   - 값 필드에 true를 입력합니다.
7. 저장을 누릅니다.
8. 콘솔에 다시 시작해야 함이 표시되면 Application Server를 다시 시작합니다.

   서버를 다시 시작한 후, certificate 영역을 사용하는 모든 응용 프로그램의 경우 클라이언트 인증이 필요합니다.

응용 프로그램에서 상호 SSL 인증 활성화

특정 응용 프로그램에 대한 상호 인증을 활성화하려면 deploytool을 사용하여 인증 방법을 Client-Certificate로 설정합니다. deploytool 사용에 대한 자세한 내용은 다음 위치에 있는 *J2EE 1.4 Tutorial*의 *Security* 장을 참조하십시오.

http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html.

영역 삭제

영역을 삭제하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.

4. 영역 노드를 선택합니다.
5. 삭제할 영역 옆에 있는 확인란을 선택합니다.
6. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-auth-realm`

기본 영역 설정

*기본 영역*은 응용 프로그램의 배포 설명자에서 영역을 지정하지 않은 경우 Application Server가 인증 및 권한 부여를 위해 사용하는 영역입니다.

기본 영역을 설정하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. 보안 노드를 선택합니다.

   보안 페이지가 표시됩니다.
4. 기본 영역 필드의 드롭다운 목록에서 원하는 영역을 선택합니다.
5. 변경 사항을 저장하려면 저장을 누르고, 변경 사항을 삭제하고 Application Server 기본값을 복구하려면 기본값 로드를 누릅니다.
6. 콘솔에 다시 시작해야 함이 표시된 경우 서버를 다시 시작합니다.

JACC 공급자에 대한 관리 콘솔 작업

JACC 공급자 작성

JACC(Java Authorization Contract for Containers)는 플러그 가능한 권한 부여 공급자의 인터페이스를 정의하는 J2EE 1.4 사양의 일부입니다. JACC를 사용하면 관리자는 타사 *플러그인* 모듈을 권한 부여를 수행하도록 설정할 수 있습니다. 기본적으로 Application Server는 JACC 호환 파일 기반 권한 부여 엔진을 제공합니다.

JACC 공급자를 만들려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. JACC 공급자 노드를 선택합니다.
5. JACC 공급자 페이지에서 새로 만들기를 누릅니다.
6. JACC 공급자 만들기 페이지에서 다음을 입력합니다.
   - **이름** - 이 공급자를 식별하기 위해 사용하는 이름입니다.
   - **정책 구성** - 정책 구성 팩토리를 구현하는 클래스 이름입니다. 기본 공급자는 `com.sun.enterprise.security.provider.PolicyConfigurationFactoryImpl`을 사용합니다.
   - **정책 공급자** - 정책 팩토리를 구현하는 클래스 이름입니다. 기본 공급자는 `com.sun.enterprise.security.provider.PolicyWrapper`를 사용합니다.
7. 등록 정보 추가 버튼을 눌러 등록 정보를 공급자에 추가합니다. 유효한 등록 정보는 다음과 같습니다.
   - `repository`: 정책 파일을 포함하는 디렉토리입니다. 기본 공급자의 경우 이 값은 *install_dir*`/domains/`*domain_dir*`/generated/policy`입니다.
8. 이 구성을 저장하려면 확인을 누르고, 저장하지 않고 중지하려면 취소를 누릅니다.

JACC 공급자 편집

JACC 공급자를 편집하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. JACC 공급자 노드를 확장합니다.
5. 편집할 JDBC 공급자의 노드를 선택합니다.
6. JACC 공급자 편집 페이지에서 원하는 대로 공급자 정보를 수정합니다.
   - **정책 구성** - 정책 구성 팩토리를 구현하는 클래스 이름입니다.
   - **정책 공급자** - 정책 팩토리를 구현하는 클래스 이름입니다.
7. 등록 정보를 추가하려면 추가 버튼을 누릅니다. 등록 정보에 대한 이름과 값을 입력합니다. 유효한 항목은 다음과 같습니다.
   - repository: 정책 파일을 포함하는 디렉토리입니다. 기본 공급자의 경우 이 값은 ${com.sun.aas.instanceRoot}/generated/policy입니다.
8. 기존 등록 정보를 삭제하려면 등록 정보 왼쪽에 있는 확인란을 누른 다음 등록 정보 삭제를 누릅니다.
9. 저장하려면 저장을 누르고, 저장하지 않고 취소하려면 브라우저의 뒤로 버튼을 누릅니다.

JACC 공급자 삭제

JACC 공급자를 삭제하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.

b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.

3. 보안 노드를 확장합니다.
4. JACC 공급자 노드를 선택합니다.
5. 삭제할 JACC 공급자 왼쪽에 있는 확인란을 선택합니다.
6. 삭제를 누릅니다.

활성 JACC 공급자 설정

JACC 공급자를 지정하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. 보안 노드를 선택합니다.

   보안 페이지가 표시됩니다.
4. JACC 필드에 서버에서 사용할 JACC 공급자 이름을 입력합니다.

   사용 가능한 JACC 공급자를 모를 경우 트리의 JACC 공급자 구성 요소를 확장하여 구성된 JACC 공급자를 모두 표시합니다.
5. 변경 사항을 저장하려면 저장을 선택하고, 기본값으로 돌아가려면 기본값 로드를 선택합니다.
6. 콘솔에 다시 시작해야 함이 표시되면 Application Server를 다시 시작합니다.

감사 모듈에 대한 관리 콘솔 작업

- 활성 감사 모듈 설정
- 감사 로깅 활성 및 비활성화

감사 모듈 만들기

Application Server는 간단한 기본 감사 모듈을 제공합니다. 자세한 내용은 "기본 감사 모듈 사용"을 참조하십시오.

감사 모듈을 새로 만들려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 감사 모듈 노드를 선택합니다.
5. 감사 모듈 페이지에서 새로 만들기를 누릅니다.
6. 감사 모듈 만들기 페이지에서 다음 정보를 입력합니다.
   - **이름** - 감사 모듈을 식별하기 위해 사용한 이름입니다.
   - **클래스 이름** - 이 모듈을 구현하는 클래스의 정규화된 이름입니다. 기본 감사 모듈의 클래스 이름은 com.sun.enterprise.security.Audit입니다.
7. JVM 등록 정보를 이 모듈에 추가하려면 등록 정보 추가를 누릅니다. 등록 정보마다 이름과 값을 지정합니다. 유효한 등록 정보는 다음과 같습니다.
   - auditOn - 이 구현 클래스를 활성화할지 여부를 지정합니다. 유효한 값은 true와 false입니다.
8. 입력을 저장하려면 확인을 누르고, 저장하지 않고 중지하려면 취소를 누릅니다.

감사 모듈 편집

감사 모듈은 기본적으로 설정되어 있지 않습니다. 감사 모듈 활성 방법에 대한 자세한 내용은 "감사 로깅 활성 및 비활성화"를 참조하십시오.

감사 모듈을 편집하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 감사 모듈 노드를 확장합니다.
5. 편집할 감사 모듈 노드를 누릅니다.
6. 감사 모듈 편집 페이지에서 필요한 경우 클래스 이름을 수정합니다.
7. 추가 버튼을 선택하고 등록 정보의 이름과 값을 입력하여 모듈에 대한 추가 등록 정보를 입력합니다. 유효한 등록 정보는 다음과 같습니다.
   - auditOn - 이 감사 모듈을 사용할지 여부를 지정합니다. 유효한 값은 true와 false입니다.
8. 수정할 이름이나 값을 선택하고 텍스트 필드에 변경 사항을 직접 입력하여 기존 등록 정보를 수정합니다.
9. 등록 정보 왼쪽에 있는 확인란을 선택하고 등록 정보 삭제를 눌러 등록 정보를 삭제합니다.
10. 저장하려면 저장을 누르고, 저장하지 않고 취소하려면 브라우저의 뒤로 버튼을 누릅니다.

감사 모듈 삭제

감사 모듈을 삭제하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.

3. 보안 노드를 확장합니다.
4. 감사 모듈 노드를 선택합니다.
5. 삭제할 감사 모듈 왼쪽에 있는 확인란을 선택합니다.
6. 삭제를 누릅니다.

감사 로깅 활성 및 비활성화

서버에서 사용하는 감사 모듈을 지정하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. 보안 노드를 선택합니다.

   보안 페이지가 표시됩니다.
4. 로깅을 사용하려면 감사 로깅 확인란을 선택합니다. 비활성화하려면 확인란을 선택 해제합니다. 이 옵션을 선택하면 감사 모듈을 로드하고 Application Server의 감사 라이브러리에서 감사 시점의 감사 모듈을 호출합니다.
5. 감사 로깅을 사용할 경우 "활성 감사 모듈 설정"에서 설명한 대로 기본 감사 모듈을 지정합니다.
6. 저장을 선택하여 변경 사항을 저장합니다.
7. 콘솔에 다시 시작해야 함이 표시되면 Application Server를 다시 시작합니다.

활성 감사 모듈 설정

서버에서 사용하는 감사 모듈을 지정하려면 "감사 로깅 활성 및 비활성화"에서 설명한 대로 감사 모듈을 활성화하고 다음 단계를 수행합니다.

1. 감사 모듈 필드에 서버에서 사용할 감사 모듈 이름을 입력합니다(사전 구성된 감사 모듈을 `default`라고 함). 이 감사 모듈의 auditOn이 "기본 감사 모듈 사용 및 사용 안 함"에서 설명한 대로 true로 설정되어 있는지 확인합니다.

2. 변경 사항을 저장하려면 저장을 선택하고, 취소하려면 기본값 로드를 선택합니다.
3. 콘솔에 다시 시작해야 함이 표시되면 Application Server를 다시 시작합니다.

기본 감사 모듈 사용

`default` 감사 모듈은 인증 및 권한 부여 요청을 서버 로그 파일에 로깅합니다. 로그 파일 위치 변경에 대한 자세한 내용은 "일반 로깅 설정 구성"을 참조하십시오.

인증 로그 항목에는 다음 정보가 포함됩니다.

- 인증을 시도한 아이디
- 액세스 요청을 처리한 영역
- 요청한 웹 모듈 URI 또는 EJB 구성 요소
- 요청의 성공 또는 실패

감사 로깅 사용 여부와 상관없이 Application Server는 거부된 인증 이벤트를 모두 로그합니다.

권한 부여 로그 항목에는 다음 정보가 포함됩니다.

- 인증된 아이디
- 요청한 웹 URI 또는 EJB 구성 요소
- 요청의 성공 또는 실패

기본 감사 모듈 사용 및 사용 안 함

로깅 사용 외에도 특정 필수 감사 모듈에서 요구하는 등록 정보를 설정합니다. 기본 감사 모듈의 경우 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. 보안 노드를 확장합니다.
4. 감사 모듈 노드를 확장합니다.

5. `Default` 노드를 누릅니다.
6. `auditOn` 등록 정보 값을 `true`로 설정합니다.
7. 저장을 선택하여 변경 사항을 저장합니다.
8. 콘솔에 다시 시작해야 함이 표시되면 Application Server를 다시 시작합니다.

Listener 및 JMX 커넥터에 대한 관리 콘솔 작업

HTTP Listener에 대한 보안 구성

HTTP 서비스의 가상 서버마다 하나 이상의 *HTTP listener*를 통해 네트워크 연결을 제공합니다. 관리 콘솔을 사용하여 새로운 HTTP listener를 만들고 기존 HTTP listener의 보안 설정을 편집합니다.

기존 HTTP listener의 보안 설정을 편집하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. HTTP 서비스 노드를 확장합니다.
4. HTTP Listeners 노드를 선택합니다.
5. HTTP listener를 선택하여 기존 listener를 편집하거나, 새로 만들기를 누르고 "HTTP Listener 만들기"의 절차를 수행하여 listener를 새로 만듭니다.
6. "Listener 보안 등록 정보 설정"의 절차를 수행하여 보안 등록 정보를 설정합니다.

7. 변경 사항을 저장하려면 저장을 누르고, 저장하지 않고 취소하려면 브라우저의 뒤로 버튼을 누릅니다.

해당 asadmin 명령: `create-http-listener`

IIOP Listener의 보안 구성

Application Server에서는 CORBA(Common Object Request Broker Architecture) 객체를 지원합니다. 이 객체는 IIOP(Internet Inter-Orb Protocol)를 사용하여 네트워크에서 통신합니다. *IIOP listener*는 EJB의 원격 클라이언트와 다른 CORBA 기반 클라이언트에서 들어오는 연결을 받아들입니다. 관리 콘솔을 사용하여 새로운 IIOP listener를 만들고 기존 IIOP listener의 설정을 편집합니다.

IIOP listener의 보안 등록 정보를 편집하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. ORB 노드를 확장합니다.
4. IIOP Listeners 노드를 선택합니다.
5. IIOP listener를 선택하여 해당 listener를 편집하거나, 새로 만들기를 누르고 "IIOP Listener 만들기"의 절차를 수행하여 listener를 새로 만듭니다.
6. "Listener 보안 등록 정보 설정"의 절차를 수행하여 보안 등록 정보를 설정합니다.
7. 저장을 눌러 변경 사항을 저장하거나, 기본값 로드를 눌러 등록 정보를 기본값으로 복구합니다.

Listener를 새로 만든 경우 IIOP Listener 페이지의 현재 Listener 테이블에 표시됩니다.

해당 asadmin 명령: `create-iiop-listener`

관리 서비스의 JMX 커넥터에 대한 보안 구성

관리 서비스의 JMX 커넥터에 대한 보안 등록 정보를 편집하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 server는 server-config 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 확장합니다.
3. 관리 서비스 노드를 확장합니다.
4. 수정할 관리 서비스를 선택합니다.
5. "Listener 보안 등록 정보 설정"의 절차를 수행하여 보안 등록 정보를 설정합니다.
6. 저장을 눌러 변경 사항을 저장하거나, 기본값 로드를 눌러 등록 정보를 기본값으로 복구합니다.

Listener 보안 등록 정보 설정

HTTP listener, IIOP listener 및 JMX 커넥터 보안 등록 정보를 설정하기 위한 공통된 다음 절차를 수행합니다.

1. HTTP Listener 편집, IIOP Listener 편집 또는 JMX 커넥터 편집 페이지에서 SSL 섹션으로 이동합니다.
2. 보안 필드에서 사용 가능 확인란을 선택하여 이 listener에 대한 보안을 사용합니다. 이 옵션을 선택하면 SSL3 또는 TLS를 선택하여 사용하는 보안 유형을 지정하고 인증서 별명을 입력해야 합니다.
3. 이 Listener를 사용할 경우 클라이언트가 자신을 Application Server에 인증하려면 클라이언트 인증 필드에서 사용 가능 확인란을 선택합니다.

4. 사용 가능 확인란을 선택한 경우 인증서 별칭 필드에 keystore 별칭을 입력합니다. Keystore 별칭은 기존 서버 키 쌍과 인증서를 식별하는 단일 값입니다. 기본 keystore에 대한 인증서 별명은 `s1as`입니다.

   인증서 별명을 찾아보려면 "CertUtil 유틸리티 정보"에서 설명한 대로 `certutil` 유틸리티를 사용합니다.

5. 사용 가능 확인란을 선택한 경우 SSL3 또는 TLS를 선택합니다. 기본적으로 SSL3과 TLS는 모두 활성화되어 있습니다.

6. 필요한 경우 개별 암호화 그룹을 활성화합니다. 기본적으로 모든 지원되는 암호 그룹이 활성화되어 있습니다. 암호는 "암호화 정보"에서 설명합니다.

7. 변경 사항을 저장하려면 저장을 선택하고, 취소하려면 기본값 로드를 선택합니다.

가상 서버의 관리 콘솔 보안 작업

단일 사인 온(SSO) 구성

단일 사인 온을 사용하면 응용 프로그램마다 별도의 사용자 로그온을 갖고 있어야 할 필요 없이 여러 응용 프로그램에서 사용자 로그온 정보를 공유할 수 있습니다. 단일 사인 온을 사용하는 응용 프로그램은 사용자를 한 번만 인증하고 인증 정보를 다른 모든 관련된 응용 프로그램에 전달합니다.

동일한 영역과 가상 서버에 대해 구성된 웹 응용 프로그램에 단일 사인 온이 적용됩니다.

주: 단일 사인 온에서는 HTTP 쿠키를 사용하여 각 요청을 저장한 사용자 아이디와 연관시키는 토큰을 전달합니다. 따라서 브라우저 클라이언트가 쿠키를 지원할 경우에만 단일 사인 온을 사용할 수 있습니다.

단일 사인 온은 다음과 같은 규칙에 따라 작동합니다.

- 사용자가 웹 응용 프로그램의 보호된 자원에 액세스할 경우 서버는 해당 웹 응용 프로그램에 정의된 방법을 사용하여 사용자가 자신을 인증할 것을 요구합니다.
- 인증이 되면 Application Server는 사용자가 개별 응용 프로그램 각각에 자신을 인증할 필요 없이 가상 서버의 모든 웹 응용 프로그램에서 권한을 결정할 때 사용자와 연관된 역할을 사용합니다.

- 사용자가 하나의 웹 응용 프로그램에서 명시적으로 또는 세션 만료 때문에 로그아웃할 경우 모든 웹 응용 프로그램의 사용자 세션이 무효화됩니다. 따라서, 이 후 응용 프로그램의 보호된 자원에 액세스하려면 로그인해야 합니다.

Application Server의 경우 단일 사인 온이 기본적으로 활성화되어 있습니다. 단일 사인 온을 비활성화하거나 다른 등록 정보를 구성하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 확장합니다.
   a. 특정 인스턴스를 구성하려면 해당 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스의 경우 `server`는 `server-config` 노드를 확장합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 확장합니다.
3. HTTP 서비스 노드를 확장합니다.
4. 가상 서버 노드를 확장하고, 단일 사인 온 지원을 위해 구성할 가상 서버를 선택합니다.
5. 등록 정보 추가를 누릅니다.

   목록 맨 아래에 빈 등록 정보 항목이 추가됩니다.
6. 이름 필드에 `sso-enable`을 입력합니다.
7. 비활성화할 값 필드에 `false`를 입력하고, SSO를 활성화하려면 `true`를 입력합니다. SSO는 기본적으로 활성화되어 있습니다.
8. 등록 정보 추가를 누르고 적용 가능한 SSO 등록 정보를 구성하여 다른 단일 사인 온 등록 정보를 추가하거나 변경합니다. 유효한 SSO 등록 정보는 표 14-10에서 설명합니다.

표 14-10 가상 서버 SSO 등록 정보

등록 정보 이름	설명	값
sso-max-inactive-seconds	클라이언트 작업이 수신되지 않는 경우 사용자의 단일 사인 온 레코드를 제거할 때까지 대기해야 하는 시간(초)입니다. 가상 서버에 있는 어떤 응용 프로그램이라도 일단 액세스하면 단일 사인 온 레코드의 활성화가 유지됩니다.	기본값은 300초(5분)입니다. 더 높은 값을 사용하면 사용자의 지속성이 더 길어지지만 서버의 더 많은 메모리를 사용합니다.
sso-reap-interval-seconds	만료된 단일 사인 온 레코드를 제거하는 간격(초)입니다.	기본값은 60입니다.

9. 저장을 누릅니다.
10. 콘솔에 다시 시작해야 함이 표시되면 Application Server를 다시 시작합니다.

커넥터 연결 풀에 대한 관리 콘솔 작업

커넥터 연결 풀 정보

커넥터 모듈(자원 어댑터라고도 함)을 사용하면 J2EE 응용 프로그램이 EIS(Enterprise Information Systems)와 상호 작용할 수 있습니다. *커넥터 자원*은 응용 프로그램에 EIS와의 연결을 제공합니다. *커넥터 연결 풀*은 특정 EIS에 대한 재사용 가능한 연결 그룹입니다.

*보안 맵*을 사용하면 J2EE 사용자 및 그룹과 EIS 사용자 및 그룹 간에 매핑을 작성할 수 있습니다. 관리 콘솔을 사용하여 커넥터 연결 풀의 보안 맵을 작성, 업데이트, 나열 및 삭제합니다.

주: 이 컨텍스트에서 사용자는 기본값이라고도 합니다. EIS(Enterprise Information System)는 정보를 가지고 있는 시스템입니다. 메인프레임, 메시징 시스템, 데이터베이스 시스템 또는 응용 프로그램입니다.

보안 맵 정보

보안 맵을 사용하여 응용 프로그램(기본값 또는 사용자 그룹)의 호출자 아이디를 컨테이너 관리 트랜잭션 기반 시나리오의 적절한 EIS 기본값으로 매핑합니다. 응용 프로그램 기본값이 EIS에 요청을 시작하면 Application Server는 먼저 커넥터 연결 풀에 정의된 보안 맵을 사용하여 정확한 기본값을 확인해서 매핑된 백엔드 EIS 기본값을 확인합니다. 정확하게 일치하는 값이 없으면 Application Server는 와일드카드 문자 사양을 사용하여 매핑된 백엔드 EIS 기본값을 확인합니다. EIS의 특정 아이디로 실행해야 할 EIS 작업을 응용 프로그램 사용자가 실행해야 할 경우 보안 맵을 사용합니다.

보안 맵 만들기

커넥터 연결 풀의 보안 맵은 사용자와 그룹(기본값)을 EIS 기본값에 매핑합니다. EIS의 특정 아이디가 필요한 EIS 작업을 응용 프로그램 사용자가 실행해야 할 경우 보안 맵을 사용합니다.

지정한 커넥터 연결 풀에 대한 보안 맵을 만들려면 다음 단계를 수행합니다.

1. 자원 노드를 확장합니다.
2. 커넥터 노드를 확장합니다.
3. 커넥터 연결 풀 노드를 선택합니다.
4. 현재 풀 목록에서 이름을 선택하여 커넥터 연결 풀을 선택하거나, 현재 풀 목록에서 새로 만들기를 선택하고 "커넥터 연결 풀 만들기"의 지침을 수행하여 새로운 커넥터 연결 풀을 만듭니다.
5. 보안 맵 페이지를 선택합니다.
6. 새로 만들기를 눌러 새로운 보안 맵을 만듭니다.
7. 보안 맵 만들기 페이지에서 다음 등록 정보를 입력합니다.
    - **이름** - 이 특정 보안 맵을 참조하기 위해 사용할 이름을 입력합니다.
    - **사용자 그룹** - 해당하는 EIS 기본값에 매핑되는 응용 프로그램의 호출자 아이디입니다. 쉼표로 구분된 응용 프로그램별 사용자 그룹 목록을 입력하거나, 와일드카드 별표(*)를 입력하여 모든 사용자나 모든 사용자 그룹을 표시합니다. 기본 또는 사용자 그룹 옵션 중 하나만을 지정합니다.
    - **기본값** - 적절한 EIS 기본값에 매핑되는 응용 프로그램의 호출자 아이디입니다. 쉼표로 구분된 응용 프로그램별 기본값 목록을 입력하거나, 와일드카드 별표(*)를 입력하여 모든 기본값을 표시합니다. 기본 또는 사용자 그룹 옵션 중 하나만을 지정합니다.
8. 백엔드 기본값 섹션에 다음 등록 정보를 입력합니다.

- **아이디** - EIS 아이디를 입력합니다. EIS(Enterprise Information System)는 정보를 가지고 있는 시스템입니다. 메인프레임, 메시징 시스템, 데이터베이스 시스템 또는 응용 프로그램입니다.
- **비밀번호** - EIS 사용자의 비밀번호를 입력합니다.

9. 확인을 눌러 보안 맵을 만들거나, 취소를 눌러 저장하지 않고 중지합니다.

해당 asadmin 명령: `create-connector-security-map`

보안 맵 편집

지정한 커넥터 연결 풀의 보안 맵을 수정하려면 다음 단계를 수행합니다.

1. 자원 노드를 확장합니다.
2. 커넥터 노드를 확장합니다.
3. 커넥터 연결 풀 노드를 선택합니다.
4. 현재 풀 목록에서 이름을 선택하여 커넥터 연결 풀을 선택합니다.
5. 보안 맵 페이지를 선택합니다.
6. 보안 맵 페이지의 현재 보안 맵 목록에서 보안 맵을 선택합니다.
7. 보안 맵 편집 페이지에서 필요한 경우 다음 등록 정보를 수정합니다.
   - **사용자 그룹** - 해당하는 EIS 기본값에 매핑되는 응용 프로그램의 호출자 아이디입니다. 쉼표로 구분된 응용 프로그램별 사용자 그룹 목록을 입력하거나, 와일드카드 별표(*)를 입력하여 모든 사용자나 모든 사용자 그룹을 표시합니다. 기본 또는 사용자 그룹 옵션 중 하나만을 지정합니다.
   - **기본값** - 적절한 EIS 기본값에 매핑되는 응용 프로그램의 호출자 아이디입니다. 쉼표로 구분된 응용 프로그램별 기본값 목록을 입력하거나, 와일드카드 별표(*)를 입력하여 모든 기본값을 표시합니다. 기본 또는 사용자 그룹 옵션 중 하나만을 지정합니다.
8. 백엔드 기본값 섹션에 다음 등록 정보를 입력합니다.
   - **아이디** - EIS 아이디를 입력합니다. EIS(Enterprise Information System)는 정보를 가지고 있는 시스템입니다. 메인프레임, 메시징 시스템, 데이터베이스 시스템 또는 응용 프로그램입니다.
   - **비밀번호** - EIS 사용자의 비밀번호를 입력합니다.

9. 저장을 눌러 보안 맵의 변경 사항을 저장합니다.

유용한 asadmin 명령: `list-connector-security-maps`, `update-connector-security-maps`

보안 맵 삭제

지정한 커넥터 연결 풀의 보안 맵을 삭제하려면 다음 단계를 수행합니다.

1. 자원 노드를 확장합니다.
2. 커넥터 노드를 확장합니다.
3. 커넥터 연결 풀 노드를 선택합니다.
4. 현재 풀 목록에서 이름을 선택하여 커넥터 연결 풀을 선택합니다.
5. 보안 맵 페이지를 선택합니다.
6. 보안 맵 페이지에서 삭제할 보안 맵 이름 왼쪽에 있는 확인란을 선택합니다.
7. 삭제를 누릅니다.

해당 asadmin 명령: `delete-connector-security-map`

인증서 및 SSL 작업

인증서 파일 정보

Application Server를 설치하면 내부 테스트에 적절한 NSS 형식으로 디지털 인증서가 생성됩니다. 기본적으로 Application Server는 *install_dir*`/domains/`*domain_name*`/config` 디렉토리의 두 파일에 인증서 정보를 저장합니다.

- **Keystore 파일** - 기본적으로 지정된 `key3.db`에는 Application Server의 개인 키를 포함하여 해당 디지털 인증서가 포함됩니다. Keystore 파일은 비밀번호로 보호됩니다. `asadmin change-master-password` 명령을 사용하여 비밀번호를 변경합니다. `certutil`에 대한 자세한 내용은 "CertUtil 유틸리티 정보"를 참조하십시오.

  모든 keystore 항목에는 고유한 별칭이 있습니다. 설치 후 Application Server keystore에는 `s1as` 별칭을 가진 단일 항목이 있습니다.

- **Trust-store 파일** - 기본적으로 지정된 `cert8.db`에는 다른 항목의 공개 키를 포함하여 Application Server의 트러스트된 인증서가 포함됩니다. 트러스트된 인증서의 경우 서버는 인증서의 공개 키가 인증서 소유자에게 속하는지 확인했습니다. 일반적으로 트러스트된 인증서에는 인증 기관(CA)의 인증서가 포함됩니다.

Platform Edition의 경우 서버측에서 Application Server는 `keytool`을 사용하여 인증서와 keystore를 관리하는 JSSE 형식을 사용합니다. Enterprise Edition의 경우 서버측에서 Application Server는 `certutil`을 사용하여 개인 키와 인증서를 저장하는 NSS 데이터베이스를 관리하는 NSS를 사용합니다. 두 Edition의 경우 클라이언트측(appclient 또는 독립 실행형)에서는 JSSE 형식을 사용합니다.

기본적으로 Application Server는 예 응용 프로그램과 같이 작동하고 개발 용도인 keystore 및 truststore가 구성되어 있습니다. 작업을 위해 인증서 별칭을 변경하거나, 다른 인증서를 truststore에 추가하거나, keystore 및 trust-store 파일 이름 또는 위치를 변경할 수 있습니다.

인증서 파일 위치 변경

개발을 위해 제공한 keystore 및 trust-store 파일은 *install_dir*`/domains/`*domain_name*`/config` 디렉토리에 저장됩니다. Keystore 및 trust-store 파일의 이름 및 위치를 변경하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리에서 구성을 확장합니다.
2. Server-config(Admin Config) 노드를 확장합니다.
3. JVM 설정 노드를 선택합니다.
4. JVM 옵션 탭을 누릅니다.
5. JVM 옵션 페이지의 값 필드에서 다음 값을 추가하거나 수정하여 인증서 파일의 새로운 위치를 반영합니다.

-Dcom.sun.appserv.nss.db=${com.sun.aas.instanceRoot}/*NSS_database_directory*

여기서 *ks_name*은 keystore 파일 이름이고 *ts_name*은 trust-store 파일 이름입니다.

6. 저장을 누릅니다.
7. 콘솔에 다시 시작해야 함이 표시되면 Application Server를 다시 시작합니다.

키 도구 유틸리티 정보

Keytool을 사용하여 Platform Edition에서 JSSE 디지털 인증서를 설정 및 작업합니다. J2SE SDK에는 keytool이 함께 제공되므로 관리자가 공개/개인 키 쌍 및 연관된 인증서를 관리할 수 있습니다. 사용자가 통신 피어의 공개 키를 인증서 양식으로 캐시할 수도 있습니다.

keytool을 실행하려면 J2SE /bin 디렉토리가 경로에 있도록 쉘 환경을 구성하거나 도구에 대한 전체 경로를 명령줄에서 제공해야 합니다. keytool에 대한 자세한 내용은 다음 위치에 있는 keytool 설명서를 참조하십시오.

http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html

CertUtil 유틸리티 정보

Certutil을 사용하여 **Enterprise Edition**에서만 NSS 디지털 인증서를 설정 및 작업합니다. 인증서 데이터베이스 도구인 certutil은 Netscape Communicator cert8.db 및 key3.db 데이터베이스 파일을 작성 및 수정할 수 있는 명령줄 유틸리티입니다. 또한 cert8.db 파일 내에 인증서를 나열, 생성, 수정 또는 삭제하거나, 비밀번호를 작성 또는 변경하거나, 새로운 공개 및 개인 키 쌍을 생성하거나, 키 데이터베이스 내용을 표시하거나, key3.db 파일 내의 키 쌍을 삭제할 수도 있습니다.

키와 인증서 관리 프로세스는 대개 키 데이터베이스에 키를 만든 다음, 인증서 데이터베이스에 인증서를 생성 및 관리하는 것으로 시작됩니다. 아래 나열된 문서에서는 certutil 유틸리티의 구문을 포함하여 NSS를 사용한 인증서 및 키 데이터베이스 관리에 대해 설명합니다.

http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html

PKCS12 형식의 인증서/키 데이터베이스 및 파일 간에 키와 인증서를 가져오고 내보내기 위해 사용한 명령줄 유틸리티가 pk12util입니다. pk12util 유틸리티에 대한 자세한 설명은 다음을 참조하십시오.

http://www.mozilla.org/projects/security/pki/nss/tools/pk12util.html

certutil, pk12util 및 기타 NSS 보안 도구 사용에 대한 자세한 내용은 http://www.mozilla.org/projects/security/pki/nss/tools/에 있는 *NSS Security Tools*를 참조하십시오.

도구는 *install_dir*/lib/ 디렉토리에 있습니다.

서버 인증서 생성

Certutil을 사용하여 인증서를 생성하고, 가져오고, 내보냅니다. 수행 방법에 대한 자세한 내용은 "CertUtil 유틸리티 정보"를 참조하십시오.

디지털 인증서 서명

디지털 인증서를 만든 후 소유자는 위조를 막기 위해 인증서에 서명해야 합니다. 전자 상거래 사이트나 아이디 인증이 중요한 사이트는 잘 알려진 인증 기관(CA)에서 인증서를 구매할 수 있습니다. 인증이 중요하지 않을 경우, 예를 들어 개인적인 보안 통신만 필요할 경우 CA 인증서를 얻는 데 필요한 시간과 경비를 절약하고 자체 서명된 인증서를 사용합니다.

인증 기관의 인증서 사용

인증 기관에서 서명한 디지털 인증서를 사용하려면 다음 작업을 수행합니다.

1. CA의 웹 사이트에서 지침을 수행하여 인증서 키 쌍을 생성합니다.
2. 생성된 인증서 키 쌍을 다운로드합니다.

   서버 keystore 및 trust-store 파일을 포함하는 디렉토리(기본적으로 *install_dir*/domains/*domain-dir*/config 디렉토리)에 인증서를 저장합니다. 이 위치 변경에 대한 자세한 내용은 "인증서 파일 위치 변경"을 참조하십시오.
3. 쉘에서 인증서가 있는 디렉토리로 변경합니다.
4. Certutil을 사용하여 인증서를 로컬 keystore로 가져오고, 필요한 경우 로컬 trust-store로도 가져옵니다.
5. Application Server를 다시 시작합니다.

certutil 사용에 대한 자세한 내용은 다음 위치에 있는 certutil 설명서를 참조하십시오.

http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html

인증서 삭제

기존 인증서를 삭제하려면 `certutil` 유틸리티를 사용합니다. `Certutil` 유틸리티에 대한 자세한 내용은 "CertUtil 유틸리티 정보"를 참조하십시오.

자세한 정보

- Java 2 Standard Edition 보안 설명은 다음 위치에서 확인할 수 있습니다.
  `http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html`
- *J2EE 1.4 Tutorial*의 *Security* 장은 다음 위치에서 확인할 수 있습니다.
  `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`
- *관리 설명서* "메시지 보안 구성" 장
- *Developer's Guide*의 *Securing Applications* 장

메시지 보안 구성

이 장에서는 Sun Java System Application Server 8.1 2005Q1의 웹 서비스에 대한 메시지 계층 보안 구성에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- 메시지 보안 정보
- 메시지 보안에 대한 관리 콘솔 작업

이 장의 일부 내용은 보안 및 웹 서비스 개념에 대한 기본적인 이해가 필요합니다. 이 개념에 대한 자세한 내용을 보려면 이 장을 시작하기 전에 "자세한 내용"에 있는 자원을 확인하십시오.

메시지 보안 정보

- 메시지 보안 개요
- Application Server의 메시지 보안 이해
- 웹 서비스 보안
- 샘플 응용 프로그램 보안
- 메시지 보안을 위해 Application Server 구성

메시지 보안 개요

*메시지 보안*의 경우 보안 정보가 메시지에 삽입되므로 네트워킹 계층을 지나 메시지와 함께 메시지 대상에 도달합니다. 메시지 보안을 사용하여 메시지 전송에서 메시지 보호를 분리하여 전송 후에도 메시지가 보호되도록 하는 전송 계층 보안(*J2EE 1.4 Tutorial*의 *Security* 장에서 설명)과 메시지 보안은 다릅니다.

WS-Security(웹 서비스 보안: SOAP 메시지 보안)는 Sun Microsystems를 포함한 모든 웹 서비스 기술 주요 제공업체가 OASIS로 공동 개발한 상호 운영 가능한 웹 서비스의 국제적인 표준입니다. WS-Security는 XML 암호화 및 XML 디지털 서명을 사용하여 SOAP에서 전송된 웹 서비스 메시지를 보안 처리하는 메시지 보안 체계입니다. WS-Security 사양에서는 SOAP 웹 서비스 메시지를 암호화하기 위한 X.509 인증서, SAML 명제 및 아이디/비밀번호 토큰 등을 포함한 다양한 보안 토큰 사용을 정의합니다.

다음 URL에서 WS-Security 사양을 확인할 수 있습니다.

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

Application Server의 메시지 보안 이해

Sun Java System Application Server 8.1 2005Q1에서는 웹 서비스 클라이언트와 서버측 컨테이너의 WS-Security 표준에 대한 통합된 지원을 제공합니다. Application Server의 컨테이너가 응용 프로그램을 대신하여 웹 서비스 보안을 실행하고 응용 프로그램의 구현을 변경하지 않고 웹 서비스 응용 프로그램을 보호하는 데 적용할 수 있도록 이 기능이 통합되어 있습니다. Application Server에서는 SOAP 계층 메시지 보안 공급자와 메시지 보호 정책을 컨테이너와 컨테이너에 배포된 응용 프로그램에 바인드하는 기능을 제공하여 이러한 효과를 달성합니다.

메시지 보안 책임 지정

Sun Java System Application Server 8.1 2005Q1의 경우 시스템 관리자 및 응용 프로그램 배포자 역할이 메시지 보안 구성에 대한 기본적인 책임을 갖습니다. 일반적인 경우에는 다른 역할이 개발자 관여 없이 기존 응용 프로그램 구현을 변경하지 않고도 기존 응용 프로그램을 보안할 수 있지만 응용 프로그램 개발자가 기여하는 경우도 있습니다. 다음 절에는 다양한 역할의 책임이 정의되어 있습니다.

- 시스템 관리자
- 응용 프로그램 배포자
- 응용 프로그램 개발자

시스템 관리자

시스템 관리자의 책임은 다음과 같습니다.

- Application Server에 메시지 보안 공급자 구성
- 사용자 데이터베이스 관리
- Keystore 및 Truststore 파일 관리
- 암호화를 사용하고 Java SDK 1.5.0 이전 버전을 실행할 경우 Java Cryptography Extension(JCE) 공급자 구성
- 샘플 서버 설치. `xms` 샘플 응용 프로그램을 사용하여 메시지 계층 웹 서비스 보안 사용을 보여줄 경우에만 수행합니다.

시스템 관리자는 관리 콘솔을 사용하여 서버 보안 설정을 관리하고 명령줄 도구를 사용하여 인증서 데이터베이스를 관리합니다. PE의 경우 인증서 및 개인 키는 키 저장소에 저장되고 `keytool`을 사용하여 관리됩니다. SE 및 EE에서는 NSS 데이터베이스에 인증서와 개인 키를 저장하고 `certutil`을 사용하여 이들을 관리합니다. 이 문서는 기본적으로 시스템 관리자를 대상으로 합니다. 메시지 보안 작업의 개요는 "메시지 보안을 위해 Application Server 구성"을 참조하십시오.

응용 프로그램 배포자

응용 프로그램 배포자의 책임은 다음과 같습니다.

- 상위 스트림 역할(개발자나 어셈블러)에서 응용 프로그램 관련 메시지 보호 정책을 지정하지 않은 경우 응용 프로그램 어셈블리에서 이러한 필수 정책을 지정

- 웹 서비스 종점 및 서비스 참조에 대한 응용 프로그램 관련 메시지 보호 정책 정보(즉, message-security-binding 요소)를 지정하기 위해 Sun 관련 배포 설명자 수정

이 보안 작업은 *Developer's Guide*의 *Securing Applications* 장에서 설명합니다. "자세한 내용"에 해당 장에 대한 링크가 있습니다.

응용 프로그램 개발자

응용 프로그램 개발자는 메시지 보안을 설정할 수 있지만 그럴 책임은 없습니다. 모든 웹 서비스가 보안 처리되도록 시스템 관리자가 메시지 보안을 설정하거나, 응용 프로그램에 바인드된 공급자나 보호 정책이 컨테이너에 바인드된 것과 달라야 하는 경우 응용 프로그램 개발자가 메시지 보안을 설정할 수 있습니다.

응용 프로그램 개발자나 어셈블러의 책임은 다음과 같습니다.

- 응용 프로그램에서 응용 프로그램 관련 메시지 보호 정책을 요구하는 지 여부를 결정합니다. 요구할 경우 응용 프로그램 배포자와 통신하여 수행할 수 있는 응용 프로그램 어셈블리에서 필요한 정책을 지정합니다.

보안 토큰 및 보안 체계 정보

WS-Security 사양에서는 보안 토큰을 사용하여 SOAP 웹 서비스 메시지를 인증 및 암호화하는 것과 관련해서 확장 가능한 체계를 제공합니다. Application Server와 같이 설치된 SOAP 계층 메시지 보안 공급자를 사용하면 아이디/비밀번호 및 X509 인증서 보안 토큰을 사용하여 SOAP 웹 서비스 메시지를 인증 및 암호화할 수 있습니다. SAML 명제를 포함하여 다른 보안 토큰을 사용하는 추가 공급자가 Application Server의 후속 릴리스와 함께 설치됩니다.

아이디 토큰 정보

Application Server는 SOAP 메시지의 *아이디 토큰*을 사용하여 메시지 *보낸 사람*의 인증 아이디를 설정합니다. 포함된 비밀번호 내에 아이디 토큰이 포함된 메시지의 수신자는 보낸 사람이 사용자의 비밀(즉, 비밀번호)을 알고 있는지 확인하여 토큰에서 식별된 사용자 역할을 할 수 있는 권한이 있는지 확인합니다.

아이디 토큰을 사용할 경우 Application Server에 유효한 사용자 데이터베이스를 구성해야 합니다. 이 항목에 대한 자세한 내용은 "영역 편집"을 참조하십시오.

디지털 서명 정보

Application Server는 XML 디지털 서명을 사용하여 인증 아이디를 메시지 *내용*에 바인드합니다. 클라이언트는 디지털 서명을 사용하여 전송 계층 보안을 사용할 경우 동일한 작업을 수행하기 위해 기본 인증 또는 SSL 클라이언트 인증서 인증을 사용한 것과 유사하게 호출자 아이디를 설정합니다. 메시지 수신자가 메시지 보낸 사람과 다를 수 있는 메시지 내용의 원본을 인증하기 위해 디지털 서명을 확인합니다.

디지털 서명을 사용할 경우 Application Server에 keystore 및 truststore 파일을 구성해야 합니다. 이 항목에 대한 자세한 내용은 "인증서 파일 정보"을 참조하십시오.

암호화 정보

암호화의 목적은 대상만 이해할 수 있도록 데이터를 수정하는 것입니다. 원본 내용을 암호화된 요소로 대체하여 수정합니다. 공개 키 암호화 도구에서 서술한 경우 암호화를 사용하여 메시지를 읽을 수 있는 당사자의 아이디를 설정할 수 있습니다.

암호화를 사용할 경우 암호화를 지원하는 JCE 공급자를 설치해야 합니다. 이 항목에 대한 자세한 내용은 "JCE 공급자 구성"을 참조하십시오.

메시지 보호 정책 정보

요청 메시지 처리와 응답 메시지 처리에 대해 메시지 보호 정책이 정의되고 원본 또는 수신자 인증을 위한 요구 사항과 관련하여 메시지 보호 정책을 표시합니다. 원본 인증 정책은 메시지를 보냈거나 메시지 내용을 정의한 엔티티의 아이디를 메시지에 설정하여 메시지 수신자가 이를 인증할 수 있도록 하는 요구 사항을 나타냅니다. 수신자 인증 정책은 메시지를 수신할 수 있는 엔티티의 아이디를 메시지 보낸 사람이 설정할 수 있도록 하는 요구 사항을 나타냅니다. SOAP 웹 서비스 메시지 측면에서 메시지 보호 정책이 실현되도록 공급자는 특정 메시지 보안 체계를 적용합니다.

공급자를 컨테이너에 구성할 경우 요청 및 응답 메시지 보호 정책이 정의됩니다. 응용 프로그램 또는 응용 프로그램 클라이언트의 Sun 특정 배포 설명자 내에 웹 서비스 포트나 작업의 단위로 응용 프로그램 관련 메시지 보호 정책을 구성할 수도 있습니다. 어떤 경우든 메시지 보호 정책이 정의된 경우 클라이언트의 요청 및 응답 메시지 보호 정책은 서버의 요청 및 응답 메시지 보호 정책과 일치해야 합니다. 응용 프로그램 관련 메시지 보호 정책 정의에 대한 자세한 내용은 *Developer's Guide*의 *Securing Applications* 장을 참조하십시오. "자세한 내용"에 해당 장에 대한 링크가 있습니다.

메시지 보안 용어의 용어집

이 문서에서 사용한 용어를 아래에서 설명합니다. "메시지 보안을 위해 Application Server 구성"에서 개념도 설명합니다.

- 인증 계층

  *인증 계층*은 인증 처리를 수행해야 하는 메시지 계층입니다. Application Server는 SOAP 계층에서 웹 서비스 메시지 보안을 실행합니다.

- 인증 공급자

  Sun Java System Application Server 이번 릴리스에서 Application Server는 *인증 공급자*를 호출하여 SOAP 메시지 계층 보안을 처리합니다.

  - *클라이언트측 공급자*는 서명 또는 아이디/비밀번호로 요청 메시지의 원본 아이디를 설정하고 암호화로 요청 메시지를 대상 수신자만 볼 수 있도록 보호합니다. 클라이언트측 공급자는 수신한 응답을 해독하여 해당 컨테이너를 수신한 응답의 인증된 수신자로 설정하고 응답의 비밀번호나 서명을 검증하여 응답과 연관된 원본 아이디를 인증합니다. Application Server에 구성된 클라이언트측 공급자를 사용하여 다른 서비스의 클라이언트 역할을 하는 서버측 구성 요소(서블릿 및 EJB)에서 전송한 요청 메시지와 수신한 응답 메시지를 보호할 수 있습니다.
  - *서버측 공급자*는 수신한 요청을 해독하여 해당 컨테이너를 수신한 요청의 인증된 수신자로 설정하고 요청의 비밀번호나 서명을 검증하여 요청과 연관된 원본 아이디를 인증합니다. 서버측 공급자는 서명 또는 아이디/비밀번호로 응답 메시지의 원본 아이디를 설정하고 암호화로 응답 메시지를 대상 수신자만 볼 수 있도록 보호합니다. 서버측 공급자는 서버측 컨테이너에서만 호출합니다.

- 기본 서버 공급자

  특정 서버 공급자가 바인드되지 않은 응용 프로그램에 대해 호출할 서버 공급자를 식별하기 위해 *기본 서버 공급자*를 사용합니다. *기본 서버 공급자*는 *기본 공급자*라고 합니다.

- 기본 클라이언트 공급자

  특정 클라이언트 공급자가 바인드되지 않은 응용 프로그램에 대해 호출할 클라이언트 공급자를 식별하기 위해 *기본 클라이언트 공급자*를 사용합니다.

- 요청 정책

  *요청 정책*은 인증 공급자가 수행한 요청 처리와 연관된 인증 정책 요구 사항을 정의합니다. 정책은 메시지 수신자가 해당 서명을 확인하기 전에 메시지를 해독해야 함을 의미하는 내용이 표시된 이후에 암호화가 발생하는 요구 사항과 같은 메시지 보낸 사람 순서에 표시됩니다.

- 응답 정책

  *응답 정책*은 인증 공급자가 수행한 응답 처리와 연관된 인증 정책 요구 사항을 정의합니다. 정책은 메시지 수신자가 해당 서명을 확인하기 전에 메시지를 해독해야 함을 의미하는 내용이 표시된 이후에 암호화가 발생하는 요구 사항과 같은 메시지 보낸 사람 순서에 표시됩니다.

웹 서비스 보안

SOAP 계층 메시지 보안 공급자와 메시지 보호 정책을 응용 프로그램이 배포된 컨테이너나 응용 프로그램에서 처리한 웹 서비스 종점에 바인드하여 Application Server에 배포된 웹 서비스가 보안 처리됩니다. SOAP 계층 메시지 보안 공급자와 메시지 보호 정책을 클라이언트 컨테이너나 클라이언트 응용 프로그램에서 선언한 이식 가능한 서비스 참조에 바인드하여 Application Server의 클라이언트측 컨테이너에 SOAP 계층 메시지 보안 기능이 구성됩니다.

Application Server가 설치된 경우 SOAP 계층 메시지 보안 공급자는 Application Server의 클라이언트 및 서버측 컨테이너에 구성됩니다. 이들 공급자는 컨테이너나 컨테이너에 배포된 개별 응용 프로그램이나 클라이언트에서 사용할 바인딩에 사용할 수 있습니다. 설치 중 공급자는 간단한 메시지 보호 정책으로 구성됩니다. 즉, 컨테이너나 컨테이너의 응용 프로그램 또는 클라이언트에 바인드된 경우 모든 요청 및 응답 메시지의 내용 원본을 XML 디지털 서명으로 인증하게 됩니다.

Application Server의 관리 인터페이스를 사용하여 Application Server의 서버측 컨테이너에서 사용하기 위해 기존 공급자를 바인드하거나, 공급자가 실행한 메시지 보호 정책을 수정하거나, 대체 메시지 보호 정책으로 새로운 공급자 구성을 만들 수 있습니다. "메시지 보안에 대한 관리 콘솔 작업"에서 이러한 작업을 정의합니다. 클라이언트 응용 프로그램의 메시지 보안 활성화에서 정의한 응용 프로그램 클라이언트 컨테이너의 SOAP 메시지 계층 보안 구성에서 유사한 관리 작업을 수행할 수 있습니다.

기본적으로 메시지 계층 보안은 Application Server에서 비활성화됩니다. Application Server의 메시지 계층 보안을 구성하려면 "메시지 보안을 위해 Application Server 구성"에서 설명한 단계를 수행합니다. 웹 서비스 보안을 사용하여 Application Server에 배포된 모든 웹 서비스 응용 프로그램을 보호하려면 "메시지 보안용 공급자 활성화" 및 "클라이언트 응용 프로그램의 메시지 보안 활성화"의 단계를 수행합니다.

Application Server 재시작이 포함될 수도 있는 위 단계를 완료하면 Application Server에 배포된 모든 웹 서비스 응용 프로그램에 웹 서비스 보안이 적용됩니다.

응용 프로그램 관련 웹 서비스 보안 구성

응용 프로그램의 Sun 관련 배포 설명자에 message-security-binding 요소를 정의하여 응용 프로그램 어셈블리시 응용 프로그램 관련 웹 서비스 보안 기능이 구성됩니다. 이 message-security-binding 요소를 사용하여 특정 공급자나 메시지 보호 정책을 웹 서비스 종점이나 서비스 참조와 연관시킬 수 있고, 해당하는 종점이나 참조 대상 서비스의 특정 포트나 메소드에 적용할 수 있습니다.

응용 프로그램 관련 메시지 보호 정책 정의에 대한 자세한 내용은 *Developer's Guide*의 *Securing Applications* 장을 참조하십시오. "자세한 내용"에 해당 장에 대한 링크가 있습니다.

샘플 응용 프로그램 보안

Application Server에는 `xms`라고 하는 샘플 응용 프로그램이 함께 제공됩니다. `xms` 응용 프로그램은 J2EE EJB 종점 및 Java Servlet 종점 모두에서 구현하는 간단한 웹 서비스 기능을 합니다. 두 종점이 동일한 서비스 종점 인터페이스를 공유합니다. 서비스 종점 인터페이스는 문자열 인수를 갖고 있고 호출 인수 앞에 `Hello`를 추가하여 만든 `String`을 반환하는 단일 작업 `sayHello`를 정의합니다.

기존 웹 서비스 응용 프로그램을 보안하는 Application Server의 WS-Security 기능 사용을 설명하기 위해 `xms` 샘플 응용 프로그램이 제공됩니다. 샘플과 함께 제공된 지침에서는 `xms` 응용 프로그램을 보안하기 위해 사용한 Application Server의 WS-Security 기능을 활성화하는 방법에 대해 설명합니다. 샘플에서는 응용 프로그램 관련 웹 서비스 보안 구성에서 설명한 대로 응용 프로그램에 적용되도록 응용 프로그램에 직접 WS-Security 기능을 바인딩하는 것도 설명합니다.

xms 샘플 응용 프로그램은 *install_dir*\samples\webservices\security\ejb\apps\xms\ 디렉토리에 설치됩니다.

xms 샘플 응용 프로그램의 컴파일, 패키지화 및 실행에 대한 자세한 내용은 *Developer's Guide*의 *Securing Applications* 장을 참조하십시오. "자세한 내용"에 해당 장에 대한 링크가 있습니다.

메시지 보안을 위해 Application Server 구성

Application Server는 해당 SOAP 처리 계층에 통합된 메시지 보안 공급자를 사용하여 메시지 보안을 구현합니다. 메시지 보안 공급자는 Application Server의 다른 보안 기능에 따라 다릅니다.

이 다른 기능을 구성하려면 다음 단계를 수행합니다.

1. 1.5.0 이전의 Java SDK 버전을 사용하고 암호화 기술을 사용할 경우 JCE 공급자를 구성합니다.

   JCE 공급자 구성은 "JCE 공급자 구성"에서 설명합니다.

2. 아이디 토큰을 사용할 경우 필요하면 사용자 데이터베이스를 구성합니다. Username/password 토큰을 사용할 경우 해당하는 영역을 구성하고 영역에 해당하는 사용자 데이터베이스를 구성해야 합니다.

   사용자 데이터베이스 구성은 "영역 편집"에서 설명합니다.

3. 필요한 경우 인증서 및 개인 키를 관리합니다.

   인증서 및 개인 키 관리는 "인증서 파일 정보"에서 설명합니다.

메시지 보안 공급자가 사용하기 위해 Application Server의 기능을 구성한 경우 Application Server와 함께 설치된 공급자를 메시지 보안용 공급자 활성화에서 설명한 대로 활성화할 수 있습니다.

JCE 공급자 구성

J2SE 1.4.x에 포함된 JCE(Java Cryptography Extension) 공급자는 RSA 암호화를 지원하지 않습니다. WS-Security에서 정의한 XML 암호화는 대개 RSA 암호화를 기반으로 하기 때문에 WS-Security를 사용하여 SOAP 메시지를 암호화하려면 RSA 암호화를 지원하는 JCE 공급자를 다운로드 및 설치해야 합니다.

주: RSA는 RSA Data Security, Inc.에서 개발한 공개 키 암호화 기술입니다. RSA는 기술 발명자인 Rivest, Shamir, Adelman의 약자입니다.

버전 1.5의 Java SDK에서 Application Server를 실행할 경우 이미 JCE 공급자가 올바로 구성되어 있습니다. 버전 1.4.x의 Java SDK에서 Application Server를 실행할 경우 다음 단계를 수행하여 JCE 공급자를 JDK 환경의 일부로 정적으로 추가합니다.

1. JCE 공급자 JAR(Java ARchive) 파일을 다운로드하여 설치합니다. 다음 URL에서는 RSA 암호화를 지원하는 JCE 공급자 목록을 제공합니다.

   `http://java.sun.com/products/jce/jce14_providers.html`

2. JCE 공급자 JAR 파일을 <*JAVA_HOME*>`/jre/lib/ext/`에 복사합니다.
3. Application Server를 중지합니다. Application Server가 중지되지 않고 이 프로세스에서 나중에 다시 시작할 경우 Application Server가 JCE 공급자를 인식하지 않습니다.
4. 텍스트 편집기에서 <*JAVA_HOME*>`/jre/lib/security/java.security` 등록 정보 파일을 편집합니다. 다운로드한 JCE 공급자를 이 파일에 추가합니다. `java.security` 파일에는 이 공급자를 추가하기 위한 자세한 지침이 포함되어 있습니다. 기본적으로 유사한 등록 정보를 가진 위치에 다음 형식의 행을 추가해야 합니다.

   ```
   security.provider.<n>=<provider class name>
   ```

   이 예에서 <*n*>은 보안 공급자를 평가할 때 Application Server에서 사용하는 기본 설정 순서입니다. 추가한 JCE 공급자에 대해 <*n*>을 2로 설정합니다.

   예를 들어, Legion of the Bouncy Castle JCE 공급자를 다운로드한 경우 다음 행을 추가합니다.

   ```
   security.provider.2=org.bouncycastle.jce.provider.
   BouncyCastleProvider
   ```

   Sun 보안 공급자가 값 1을 가진 가장 높은 기본 설정을 유지하는지 확인합니다.

   ```
   security.provider.1=sun.security.provider.Sun
   ```

다른 보안 공급자의 수준을 하향 조정하여 각 수준에 하나의 보안 공급자만 있도록 합니다.

다음은 필요한 JCE 공급자를 제공하고 기존 공급자를 올바른 위치에 보관하는 java.security 파일의 예입니다.

```
security.provider.1=sun.security.provider.Sun
security.provider.2=org.bouncycastle.jce.provider.
BouncyCastleProvider
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.rsajca.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
```

5. 파일을 저장하고 닫습니다.
6. 서버를 다시 시작합니다.

메시지 보안에 대한 관리 콘솔 작업

메시지 보안을 사용하기 위해 Application Server를 설정하는 대부분의 단계는 관리 콘솔이나 asadmin 명령줄 도구를 사용하여 수행하거나 시스템 파일을 수동으로 편집하여 수행할 수 있습니다. 일반적으로 시스템 파일을 편집하는 것은 의도하지 않은 변경으로 인해 Application Server가 제대로 실행되지 못하게 할 수 있기 때문에 권장하지 않습니다. 따라서 가능한 한 관리 콘솔을 사용하여 Application Server를 구성하는 단계가 먼저 표시되고 asadmin 도구 명령을 사용한 단계는 나중에 표시됩니다. 관리 콘솔이나 해당 asadmin 명령이 없을 경우에만 시스템 파일을 수동으로 편집하는 단계가 표시됩니다.

메시지 계층 보안 지원이 플러그 가능한 인증 모듈 양식으로 클라이언트 컨테이너와 Application Server에 통합됩니다. 기본적으로 메시지 계층 보안은 Application Server에서 비활성화됩니다. 다음 절에서는 메시지 보안 구성 및 공급자를 활성화, 작성, 편집 및 삭제하는 것과 관련한 세부 사항을 제공합니다.

- 메시지 보안용 공급자 활성화
- 메시지 보안 공급자 구성
- 메시지 보안 공급자 만들기
- 메시지 보안 구성 삭제
- 메시지 보안 공급자 삭제

- 클라이언트 응용 프로그램의 메시지 보안 활성화

대부분의 경우 위에 나열된 관리 작업을 수행한 후 Application Server를 다시 시작해야 합니다. 작업을 수행할 때 Application Server에 이미 배포된 응용 프로그램에 관리 변경 사항을 적용할 경우에 특히 Application Server를 다시 시작해야 합니다.

메시지 보안용 공급자 활성화

Application Server에 배포된 웹 서비스 종점에 대한 메시지 보안을 활성화하려면 서버측에서 기본적으로 사용되는 공급자를 지정해야 합니다. 메시지 보안을 위한 기본 공급자를 활성화한 경우 Application Server에 배포된 웹 서비스의 클라이언트에서 사용할 공급자도 활성화해야 합니다. 클라이언트에서 사용한 공급자를 활성화하는 데 필요한 정보는 "클라이언트 응용 프로그램의 메시지 보안 활성화"에서 설명합니다.

배포된 종점에서 발생한 웹 서비스 호출에 대해 메시지 보안을 활성화하려면 기본 클라이언트 공급자를 지정해야 합니다. Application Server에 대한 기본 클라이언트 공급자를 활성화한 경우 Application Server에 배포된 종점에서 호출한 모든 서비스가 메시지 계층 보안과 호환 가능하게 구성되도록 해야 합니다.

Application Server에 대한 기본 공급자를 활성화하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. 보안 노드를 확장합니다.
4. 메시지 보안 노드를 확장합니다.
5. SOAP 노드를 확장합니다.
6. 메시지 보안 탭을 선택합니다.
7. 메시지 보안 구성 편집 탭에서 바인드되지 않은 특정 공급자의 모든 응용 프로그램에 대해 서버측에서 사용할 공급자와 클라이언트측에서 사용할 공급자를 지정합니다. 다음 선택적 등록 정보를 수정하여 이를 수행합니다.

- **기본 공급자** - 특정 서버 공급자가 바인드되지 않은 응용 프로그램에 대해 호출할 서버 공급자의 아이디입니다.

  기본적으로 Application Server에 대한 공급자 구성은 선택되어 있지 않습니다. 서버측 공급자를 식별하려면 `ServerProvider`를 선택합니다. Null 옵션을 선택하면 기본적으로 서버측에서 메시지 보안 공급자가 호출되지 않음을 의미합니다.

  이 필드에 대해 일반적으로 `ServerProvider`를 선택합니다.

- **기본 클라이언트 공급자** - 특정 클라이언트 공급자가 바인드되지 않은 응용 프로그램에 대해 호출할 클라이언트 공급자의 아이디입니다.

  기본적으로 Application Server에 대한 공급자 구성은 선택되어 있지 않습니다. 클라이언트측 공급자를 식별하려면 `ClientProvider`를 선택합니다. Null 옵션을 선택하면 기본적으로 클라이언트측에서 메시지 보안 공급자가 호출되지 않음을 의미합니다.

  이 필드에 대해 일반적으로 Null을 선택합니다. 기본 공급자와 메시지 보호 정책을 활성화하여 Application Server에 배포된 웹 서비스 종점에서 시작된 웹 서비스 호출에 적용하려면 ClientProvider를 선택합니다.

8. 저장을 누릅니다.
9. 클라이언트 또는 서버 공급자를 활성화하고 활성화된 공급자의 메시지 보호 정책을 수정할 경우 이 단계에서 활성화한 메시지 보안 공급자의 구성 수정에 대한 자세한 내용은 "메시지 보안 공급자 구성"을 참조하십시오.

해당 asadmin 명령:

- 기본 서버 공급자를 지정하려면 다음 작업을 수행합니다.

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  default_provider=ServerProvider
  ```

- 기본 클라이언트 공급자를 지정하려면 다음 작업을 수행합니다.

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  default_client_provider=ClientProvider
  ```

메시지 보안 공급자 구성

일반적으로 공급자 유형, 구현 클래스 및 공급자별 구성 등록 정보를 수정할 수 있더라도 메시지 보호 정책을 수정하기 위해 공급자가 다시 구성됩니다. 메시지 보안 공급자를 다시 구성하려면 아래 나열된 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 보안 노드를 확장합니다.
4. 메시지 보안 노드를 확장합니다.
5. SOAP 노드를 확장합니다.
6. 공급자 탭을 선택합니다.
7. 편집할 메시지 보안 공급자를 선택합니다. Application Server에는 ClientProvider와 ServerProvider가 함께 제공됩니다.
8. 공급자 구성 편집 페이지의 공급자 구성 섹션에서 다음 등록 정보를 수정할 수 있습니다.
   - **공급자 유형** - 공급자를 클라이언트 인증 공급자, 서버 인증 공급자 또는 둘 다(클라이언트-서버 공급자)로 사용할지 여부를 설정하려면 client, server 또는 client-server를 선택합니다.
   - **클래스 이름** - 공급자의 Java 구현 클래스를 입력합니다. 클라이언트 인증 공급자는 com.sun.xml.wss.provider.ClientSecurityAuthModule 인터페이스를 구현해야 합니다. 서버측 공급자는 com.sun.xml.wss.provider.ServerSecurityAuthModule 인터페이스를 구현해야 합니다. 공급자는 두 가지 인터페이스를 모두 구현할 수 있지만, 공급자 유형에 해당하는 인터페이스를 구현해야 합니다.
9. 공급자 구성 만들기 페이지의 요청 정책 섹션에서 필요한 경우 다음과 같은 **선택적** 값을 입력합니다. 이 등록 정보는 선택적이지만, 지정하지 않을 경우 요청 메시지에 인증이 적용되지 않습니다.

*요청 정책*은 인증 공급자가 수행한 요청 처리와 연관된 인증 정책 요구 사항을 정의합니다. 정책은 메시지 수신자가 해당 서명을 확인하기 전에 메시지를 해독해야 함을 의미하는 내용이 표시된 이후에 암호화가 발생하는 요구 사항과 같은 메시지 보낸 사람 순서에 표시됩니다.

- **인증 소스** - `sender`, `content` 또는 Null(빈 옵션)을 선택하여 각각 메시지 계층 보낸 사람 인증을 위한 요구 사항(예: 아이디 비밀 번호), 내용 인증을 위한 요구 사항(예: 디지털 서명) 또는 요청 메시지에 적용할 인증 없음을 정의합니다. Null을 지정한 경우 요청의 소스 인증이 필요하지 않습니다.
- **인증 수신자** - `beforeContent` 또는 `afterContent`를 선택하여 요청 메시지의 수신자의 보낸 사람에 대한 메시지 계층 인증을 위한 요구 사항(예: XML 암호화)을 정의합니다. 값이 지정되지 않은 경우에는 기본값인 `afterContent`로 지정됩니다.

SOAP 메시지 보안 공급자가 수행한 작업에 대한 자세한 내용은 "요청 및 응답 정책 구성 작업"을 참조하십시오.

10. 공급자 구성 만들기 페이지의 응답 정책 섹션에서 필요한 경우 다음과 같은 **선택적** 값을 입력합니다. 이 등록 정보는 선택적이지만, 지정하지 않을 경우 응답 메시지에 인증이 적용되지 않습니다.

*응답 정책*은 인증 공급자가 수행한 응답 처리와 연관된 인증 정책 요구 사항을 정의합니다. 정책은 메시지 수신자가 해당 서명을 확인하기 전에 메시지를 해독해야 함을 의미하는 내용이 표시된 이후에 암호화가 발생하는 요구 사항과 같은 메시지 보낸 사람 순서에 표시됩니다.

- **인증 소스** - `sender`, `content` 또는 Null(빈 옵션)을 선택하여 응답 메시지에 적용할 메시지 계층 보낸 사람 인증을 위한 요구 사항(예: 아이디 비밀 번호)이나 내용 인증을 위한 요구 사항(예: 디지털 서명)을 정의합니다. Null을 지정한 경우 응답의 소스 인증이 필요하지 않습니다.
- **인증 수신자** - `beforeContent` 또는 `afterContent`를 선택하여 응답 메시지의 수신자의 보낸 사람에 대한 메시지 계층 인증을 위한 요구 사항(예: XML 암호화)을 정의합니다. 값이 지정되지 않은 경우에는 기본값인 `afterContent`로 지정됩니다.

SOAP 메시지 보안 공급자가 수행한 작업에 대한 자세한 내용은 "요청 및 응답 정책 구성 작업"을 참조하십시오.

11. 등록 정보 추가 버튼을 눌러 다른 등록 정보를 추가합니다. Application Server와 함께 제공된 공급자는 아래 나열된 등록 정보를 지원합니다. 다른 공급자를 사용할 경우 등록 정보와 유효한 값에 대한 자세한 내용은 해당 설명서를 참조하십시오.
    - `server.config` - 서버 구성 정보를 포함하는 XML 파일의 디렉토리와 파일 이름입니다. 예를 들면, *install_dir*`/domains/`*domain_dir*`/config/wss-server-config.xml`입니다.
12. 저장을 누릅니다.

해당 `asadmin` 명령은 다음과 같습니다. 응답 정책을 설정하려면 다음 명령의 단어 `request`를 `response`로 바꿉니다.

- 요청 정책을 클라이언트에 추가하고 인증 소스를 설정합니다.

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  provider-config.ClientProvider.request-policy.auth_source=
  <sender | content>
  ```

- 요청 정책을 서버에 추가하고 인증 소스를 설정합니다.

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  provider-config.ServerProvider.request-policy.auth_source=
  <sender | content>
  ```

- 요청 정책을 클라이언트에 추가하고 인증 수신자를 설정합니다.

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  provider-config.ClientProvider.request-policy.auth_recipient=
  <before-content | after-content>
  ```

- 요청 정책을 서버에 추가하고 인증 수신자를 설정합니다.

  ```
  asadmin  set --user  <admin-user> --port <admin-port>
  server-config.security-service.message-security-config.SOAP.
  provider-config.ServerProvider.request-policy.auth_recipient=
  <before-content | after-content>
  ```

메시지 보안 공급자 만들기

새로운 메시지 보안 공급자를 만들려면 다음 단계를 수행합니다. 기존 공급자를 구성하려면 "메시지 보안 공급자 구성"의 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.

2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 보안 노드를 확장합니다.
4. 메시지 보안 노드를 확장합니다.
5. SOAP 노드를 확장합니다.
6. 공급자 탭을 선택합니다.
7. 공급자 구성 페이지에서 새로 만들기를 누릅니다.
8. 공급자 구성 만들기 페이지의 공급자 구성 섹션에서 다음을 입력합니다.
   - **기본 공급자** - 새로운 메시지 보안 공급자를 특정 공급자가 바인드되지 않은 응용 프로그램에 대해 호출할 공급자로 만들려면 이 필드 옆에 있는 확인란을 선택합니다. 공급자가 기본 클라이언트 공급자, 기본 서버 공급자 또는 둘 다가 될지 여부는 공급자 유형에 대해 선택한 값에 따라 다릅니다.
   - **공급자 유형** - client, server 또는 client-server를 선택하여 공급자를 클라이언트 인증 공급자, 서버 인증 공급자 또는 둘 다(클라이언트-서버 공급자)로 사용할지 여부를 설정합니다.
   - **공급자 아이디** - 이 공급자 구성에 대한 아이디를 입력합니다. 현재 공급자 구성 목록에 이 이름이 표시됩니다.
   - **클래스 이름** - 공급자의 Java 구현 클래스를 입력합니다. 클라이언트 인증 공급자는 com.sun.xml.wss.provider.ClientSecurityAuthModule 인터페이스를 구현해야 합니다. 서버측 공급자는 com.sun.xml.wss.provider.ServerSecurityAuthModule 인터페이스를 구현해야 합니다. 공급자는 두 가지 인터페이스를 모두 구현할 수 있지만, 공급자 유형에 해당하는 인터페이스는 반드시 구현해야 합니다.
9. 공급자 구성 만들기 페이지의 요청 정책 섹션에서 필요한 경우 다음과 같은 **선택적** 값을 입력합니다. 이 등록 정보는 선택적이지만, 지정하지 않을 경우 요청 메시지에 인증이 적용되지 않습니다.

- **인증 소스** - sender, content 또는 Null(빈 옵션)을 선택하여 메시지 계층 보낸 사람 인증을 위한 요구 사항(예: 아이디 비밀 번호), 내용 인증을 위한 요구 사항(예: 디지털 서명) 또는 요청 메시지에 적용할 인증 없음을 정의합니다. Null을 지정한 경우 요청의 소스 인증이 필요하지 않습니다.
- **인증 수신자** - beforeContent 또는 afterContent를 선택하여 요청 메시지의 수신자의 보낸 사람에 대한 메시지 계층 인증을 위한 요구 사항(예: XML 암호화)을 정의합니다. 값이 지정되지 않은 경우에는 기본값인 afterContent로 지정됩니다.

SOAP 메시지 보안 공급자가 수행한 작업에 대한 자세한 내용은 "요청 및 응답 정책 구성 작업"을 참조하십시오.

10. 공급자 구성 만들기 페이지의 응답 정책 섹션에서 필요한 경우 다음과 같은 **선택적** 값을 입력합니다. 이 등록 정보는 선택적이지만, 지정하지 않을 경우 응답 메시지에 인증이 적용되지 않습니다.
    - **인증 소스** - sender, content 또는 Null(빈 옵션)을 선택하여 응답 메시지에 적용할 메시지 계층 보낸 사람 인증을 위한 요구 사항(예: 아이디 비밀 번호)이나 내용 인증을 위한 요구 사항(예: 디지털 서명)을 정의합니다. Null을 지정한 경우 응답의 소스 인증이 필요하지 않습니다.
    - **인증 수신자** - beforeContent 또는 afterContent를 선택하여 응답 메시지의 수신자의 보낸 사람에 대한 메시지 계층 인증을 위한 요구 사항(예: XML 암호화)을 정의합니다. 값이 지정되지 않은 경우에는 기본값인 afterContent로 지정됩니다.

    SOAP 메시지 보안 공급자가 수행한 작업에 대한 자세한 내용은 "요청 및 응답 정책 구성 작업"을 참조하십시오.
11. 등록 정보 추가 버튼을 눌러 다른 등록 정보를 추가합니다. Application Server와 함께 제공된 공급자는 아래 나열된 등록 정보를 지원합니다. 다른 공급자를 사용할 경우 등록 정보와 유효한 값에 대한 자세한 내용은 해당 설명서를 참조하십시오.
    - server.config - 서버 구성 정보를 포함하는 XML 파일의 디렉토리와 파일 이름입니다. 예를 들면, *install_dir*/domains/*domain_dir*/config/wss-server-config.xml입니다.
12. 이 구성을 저장하려면 확인을 누르거나, 저장하지 않고 중지하려면 취소를 누릅니다.

해당 asadmin 명령: create-message-security-provider

요청 및 응답 정책 구성 작업

표 15-1에서는 메시지 보호 정책 구성과 해당 구성의 WS-Security SOAP 메시지 보안 공급자가 수행한 메시지 보안 작업 결과를 보여줍니다.

표 15-1 메시지 보호 정책 대 WS-Security SOAP 메시지 보안 작업 매핑

메시지 보호 정책	WS-Security SOAP 메시지 보호 작업 결과
auth-source="sender"	메시지에는 wsse:UsernameToken(비밀번호 포함)을 포함하는 wsse:Security 헤더가 들어 있습니다.
auth-source="content"	SOAP 메시지 본문의 내용을 서명합니다. 메시지에는 ds:Signature로 표시되는 메시지 본문 서명을 포함하는 wsse:Security 헤더가 들어 있습니다.
auth-source="sender" auth-recipient="before-content" 또는 auth-recipient="after-content"	SOAP 메시지 본문의 내용이 암호화되고 xend:EncryptedData 결과와 대체됩니다. 메시지에는 wsse:UsernameToken(비밀번호 포함) 및 xenc:EncryptedKey가 포함된 a wsse:Security 헤더가 들어 있습니다. xenc:EncryptedKey는 SOAP 메시지 본문을 암호화하는 데 사용한 키를 포함합니다. 키는 수신자의 공개 키에 암호화되어 있습니다.
auth-source="content" auth-recipient="before-content"	SOAP 메시지 본문의 내용이 암호화되고 xend:EncryptedData 결과와 대체됩니다. xenc:EncryptedData가 서명됩니다. 메시지에는 xenc:EncryptedKey 및 ds:Signature가 포함된 a wsse:Security 헤더가 들어 있습니다. xenc:EncryptedKey는 SOAP 메시지 본문을 암호화하는 데 사용한 키를 포함합니다. 키는 수신자의 공개 키에 암호화되어 있습니다.
auth-source="content" auth-recipient="after-content"	SOAP 메시지 본문의 내용이 서명 및 암호화되고 xend:EncryptedData 결과와 대체됩니다. 메시지에는 xenc:EncryptedKey 및 ds:Signature가 포함된 a wsse:Security 헤더가 들어 있습니다. xenc:EncryptedKey는 SOAP 메시지 본문을 암호화하는 데 사용한 키를 포함합니다. 키는 수신자의 공개 키에 암호화되어 있습니다.
auth-recipient="before-content" 또는 auth-recipient="after-content"	SOAP 메시지 본문의 내용이 암호화되고 xend:EncryptedData 결과와 대체됩니다. 메시지에는 xenc:EncryptedKey가 포함된 a wsse:Security 헤더가 들어 있습니다. xenc:EncryptedKey는 SOAP 메시지 본문을 암호화하는 데 사용한 키를 포함합니다. 키는 수신자의 공개 키에 암호화되어 있습니다.
지정된 정책이 없습니다.	모듈에서 보안 작업을 수행하지 않습니다.

메시지 보안 구성 삭제

메시지 보안 구성을 삭제하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 보안 노드를 확장합니다.
4. 메시지 보안 노드를 선택합니다.
5. 삭제할 메시지 보안 구성 왼쪽에 있는 확인란을 선택합니다.
6. 삭제를 누릅니다.

메시지 보안 공급자 삭제

메시지 보안 공급자를 삭제하려면 다음 단계를 수행합니다.

1. 관리 콘솔 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 보안 노드를 확장합니다.
4. 메시지 보안 노드를 확장합니다.
5. SOAP 노드를 확장합니다.
6. 공급자 페이지를 선택합니다.
7. 삭제할 공급자 구성 왼쪽에 있는 확인란을 선택합니다.
8. 삭제를 누릅니다.

해당 asadmin 명령: delete-message-security-provider

클라이언트 응용 프로그램의 메시지 보안 활성화

클라이언트 공급자의 메시지 보호 정책이 클라이언트 공급자가 상호 작용하는 서버측 공급자의 메시지 보호 정책과 동등하도록 구성해야 합니다. Application Server를 설치한 경우 구성되었지만 활성화되지 않은 공급자의 경우 이미 그렇게 되어 있습니다.

클라이언트 응용 프로그램에 대한 메시지 보안을 활성화하려면 응용 프로그램 클라이언트 컨테이너의 Sun Java System Application Server 관련 구성을 수정합니다.

응용 프로그램 클라이언트의 기본 클라이언트 공급자를 활성화하려면 다음 단계를 수행합니다.

1. 클라이언트 컨테이너 설명자를 사용하는 클라이언트 응용 프로그램을 중지합니다.
2. 텍스트 편집기에서 *install_dir*/domains/*domain_dir*/config/sun-acc.xml에 있는 Sun 응용 프로그램 클라이언트 컨테이너 설명자를 엽니다.
3. **굵은체** 텍스트를 파일에 추가하여 응용 프로그램 클라이언트의 기본 클라이언트 공급자를 활성화합니다. 클라이언트 응용 프로그램의 메시지 보안을 활성화하기 위한 코드가 있는 곳을 표시하기 위해 다른 코드를 제공합니다. 굵은체가 아닌 코드는 사용자 설치에서 약간 다를 수 있습니다. 굵은체가 아닌 텍스트를 변경하지 마십시오.

```
<client-container>
  <target-server name="<your_host>" address="<your_host>" port="<your_port>"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config
      class-name="com.sun.xml.wss.provider.ClientSecurityAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender"/>
      <response-policy/>
       <property name="security.config"
      value="C:/Sun/AppServer/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

클라이언트 컨테이너에 구성된 메시지 보안 공급자에는 개인 키와 트러스트된 인증서에 대한 액세스가 필요합니다. 응용 프로그램 클라이언트 시작 스크립트에 다음 시스템 등록 정보에 대한 해당 값을 정의하여 이를 수행합니다.

-Djavax.net.ssl.keyStore

-Djavax.net.ssl.trustStore

응용 프로그램 클라이언트 구성에 대한 요청 및 응답 정책 설정

*요청 및 응답 정책*에서는 인증 공급자가 수행한 요청 및 응답 처리와 연관된 인증 정책 요구 사항을 정의합니다. 정책은 메시지 수신자가 해당 서명을 확인하기 전에 메시지를 해독해야 함을 의미하는 내용이 표시된 이후에 암호화가 발생하는 요구 사항과 같은 메시지 보낸 사람 순서에 표시됩니다.

메시지 보안을 달성하려면 서버와 클라이언트 모두에서 요청 및 응답 정책을 활성화해야 합니다. 클라이언트와 서버에 정책을 구성할 경우 응용 프로그램 수준 메시지 바인딩에서 요청/응답 보호를 위해 서버 정책과 클라이언트 정책이 일치해야 합니다.

응용 프로그램 클라이언트 구성에 대한 요청 정책을 설정하려면 "클라이언트 응용 프로그램의 메시지 보안 활성화"에서 설명한 대로 응용 프로그램 클라이언트 컨테이너에 대한 Sun Java System Application Server 관련 구성을 수정합니다. 응용 프로그램 클라이언트 구성 파일에서 **굵은체** 텍스트를 추가하여 요청 정책을 설정합니다. 참조를 위해 다른 코드가 제공됩니다. 굵은체가 아닌 코드는 사용자 설치에서 약간 다를 수 있습니다. 굵은체가 아닌 텍스트를 변경하지 마십시오.

```
<client-container>
  <target-server name="<your_host>" address="<your_host>" port="<your_port>"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config
      class-name="com.sun.xml.wss.provider.ClientSecurityAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
      value="install_dir/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

auth-source에 대한 유효한 값에는 sender 및 content가 포함됩니다. auth-recipient에 대한 유효한 값에는 before-content 및 after-content가 포함됩니다. "요청 및 응답 정책 구성 작업"에는 이 값의 다양한 조합 결과를 설명하는 표가 있습니다.

요청이나 응답 정책을 지정하지 않으려면 예를 들어, 다음 요소를 비워 둡니다.

```
<response-policy/>
```

자세한 내용

- Java 2 Standard Edition 보안 설명은 다음 위치에서 확인할 수 있습니다.
  http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html
- *J2EE 1.4 Tutorial*의 *Security* 장은 다음 위치에서 확인할 수 있습니다.
  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html
- *관리 설명서*의 "보안 구성" 장
- *Developer's Guide*의 *Securing Applications* 장
- *Oasis Web Services Security: SOAP Message Security (WS-Security)* 사양은 다음 위치에서 확인할 수 있습니다.
  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
- *OASIS Web Services Security Username Token Profile 1.0*은 다음 URL에서 확인할 수 있습니다.
  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf
- *OASIS Web Services Security X.509 Certificate Token Profile 1.0*은 다음 URL에서 확인할 수 있습니다.
  http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf
- *XML-Signature Syntax and Processing* 문서는 다음 URL에서 확인할 수 있습니다.
  http://www.w3.org/TR/xmldsig-core/
- *XML Encryption Syntax and Processing* 문서는 다음 URL에서 확인할 수 있습니다.
  http://www.w3.org/TR/xmlenc-core/

트랜잭션

분할할 수 없는 작업 단위에 하나 이상의 단계를 포함시켜서 트랜잭션에서 데이터 무결성과 일관성을 보장합니다. 이 장은 다음 내용으로 구성되어 있습니다.

트랜잭션 정보

트랜잭션

트랜잭션은 응용 프로그램에서 모두 성공적으로 완료하지 않으면 각 작업의 변경 사항이 철회되는 일련의 작업입니다. 예를 들어, 당좌 계좌의 자금을 저축 계좌로 대체하는 것은 다음 단계로 구성된 트랜잭션입니다.

1. 당좌 계좌에 대체에 충분한 자금이 있는지 확인합니다.
2. 당좌 계좌에 충분한 자금이 있을 경우 당좌 계좌의 금액을 차변에 기입합니다.
3. 저축 계좌의 대변에 자금을 기입합니다.
4. 당좌 계좌 로그에 대체를 기록합니다.
5. 저축 계좌 로그에 대체를 기록합니다.

이 단계 중 어느 한 단계라도 실패할 경우 이전 단계의 모든 변경 사항이 철회되고, 당좌 계좌와 저축 계좌는 트랜잭션 시작 전과 동일한 상태가 되어야 합니다. 이 이벤트를 *롤백*이라고 합니다. 모든 단계가 성공적으로 완료되면 트랜잭션은 *완결* 상태가 됩니다. 트랜잭션은 완결 또는 롤백 상태로 종료됩니다.

J2EE 기술의 트랜잭션

J2EE 기술에서 트랜잭션을 처리하는 데는 다음 참가자가 관련됩니다.

- 트랜잭션 관리자
- Application Server
- 자원 관리자
- 자원 어댑터
- 사용자 응용 프로그램

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.
- Application Server는 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 Application Server 또는 클라이언트가 자원 관리자와 연결할 때 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.

- Application Server 환경에서 실행되도록 개발된 트랜잭션 사용자 응용 프로그램은 JNDI를 사용하여 트랜잭션 데이터 소스를 조회할 뿐만 아니라 필요에 따라 트랜잭션 관리자도 조회합니다. 응용 프로그램은 Enterprise Bean의 선언적 트랜잭션 속성 설정이나 명시적인 프로그램형 트랜잭션 경계를 사용할 수 있습니다.

트랜잭션의 관리 콘솔 작업

Application Server는 관리 콘솔의 설정을 기반으로 트랜잭션을 처리합니다.

트랜잭션 구성

이 절에서는 다음 트랜잭션 속성을 구성하는 방법에 대해 설명합니다.

- 트랜잭션 복구
- 트랜잭션 시간 초과
- 트랜잭션 로깅

트랜잭션 복구

서버 충돌이나 자원 관리자 충돌 때문에 트랜잭션이 완료되지 않을 수 있습니다. 이 오류가 발생한 트랜잭션을 완료하고 오류를 복구하는 것이 중요합니다. Application Server는 이 오류를 복구하고 서버 시작 시 트랜잭션을 완료하도록 설계되어 있습니다.

복구를 수행하는 중 일부 자원에 연결할 수 없을 경우 트랜잭션을 복구하려고 하기 때문에 서버 재시작이 지연될 수 있습니다.

트랜잭션이 여러 서버에 걸쳐 있는 경우 트랜잭션을 시작한 서버는 다른 서버에 연결하여 트랜잭션의 결과를 가져올 수 있습니다. 다른 서버에 연결할 수 없는 경우 트랜잭션은 발견적 판단 필드를 사용하여 결과를 확인합니다.

트랜잭션에서 Application Server를 복구하는 방법을 구성하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.

b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.

3. 트랜잭션 서비스 노드를 선택합니다.
4. 완료되지 않은 트랜잭션 복구를 활성화하려면 다시 시작할 때 복구 필드를 선택합니다.
5. 재시도 시간 초과 필드에서 Application Server가 연결할 수 없는 서버에 연결을 시도하는 시간(초)을 설정합니다. 기본값은 10분(600초)입니다.
6. 발견적 판단 필드에서 트랜잭션 중 연결할 수 없는 서버에 대한 정책을 설정합니다.

   이 필드를 완결로 설정할 이유가 없을 경우 발견적 판단을 롤백으로 설정해 둡니다. 확정되지 않은 트랜잭션을 완결하면 응용 프로그램의 데이터 무결성이 손상될 수 있습니다.
7. 저장을 누릅니다.
8. 서버를 다시 시작합니다.

트랜잭션 시간 초과

기본적으로 서버는 트랜잭션을 시간 초과하지 않습니다. 즉 서버는 트랜잭션이 완료될 때까지 무기한 기다립니다. 트랜잭션의 시간 초과 값을 설정하면 트랜잭션이 구성된 시간 내에 완료되지 않을 경우 Application Server는 트랜잭션을 롤백합니다.

시간 초과 값을 설정하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.

   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.

   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. 트랜잭션 서비스 노드를 선택합니다.
4. 트랜잭션 시간 초과 필드에 트랜잭션이 시간 초과되기 전의 시간(초)을 입력합니다.

   트랜잭션 시간 초과의 기본값은 0초입니다. 그러면 트랜잭션 시간 초과가 비활성화됩니다.

5. 저장을 누릅니다.
6. 서버를 다시 시작합니다.

트랜잭션 로깅

트랜잭션 로그는 관련된 자원의 데이터 무결성을 유지하고 오류를 복구하기 위해 각 트랜잭션에 대한 정보를 기록합니다. 트랜잭션 로그 위치 필드에서 지정한 디렉토리의 tx 하위 디렉토리에 트랜잭션 로그가 저장됩니다. 이 로그는 사람이 읽을 수 없습니다.

트랜잭션 로그 위치를 설정하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 트랜잭션 서비스 노드를 선택합니다.
4. 트랜잭션 로그 위치 필드에 트랜잭션 로그 위치를 입력합니다.

   tx 하위 디렉토리가 만들어지고 트랜잭션 로그가 이 디렉토리 아래에 저장됩니다.

   기본값은 ${com.sun.aas.instanceRoot}/logs입니다.
   ${com.sun.aas.instanceRoot} 변수는 인스턴스 이름이고, Application Server 인스턴스를 시작할 때 이 변수가 설정됩니다. ${com.sun.aas.instanceRoot}의 값을 확인하려면 실제 값을 누릅니다.
5. 저장을 누릅니다.
6. 서버를 다시 시작합니다.

키포인트 작업이 트랜잭션 로그 파일을 압축합니다. 키포인트 간격은 로그의 키포인트 작업 간의 트랜잭션 수입니다. 키 포인트 작업이 트랜잭션 로그 파일 크기를 줄일 수 있습니다. 키포인트 간격이 크면(예: 2048) 트랜잭션 로그 파일이 더 커지고, 키포인트 작업이 적어질수록 성능이 더 좋아질 가능성이 있습니다. 키포인트 간격이 작아지면(예: 256) 로그 파일이 작아지지만, 키 포인트 작업 빈도가 높아져서 성능이 약간 낮아질 수 있습니다.

키포인트 간격을 설정하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 구성 노드를 선택합니다.

2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 트랜잭션 서비스 노드를 선택합니다.
4. 키포인트 간격 필드에서 키포인트 작업 간의 트랜잭션 수를 입력합니다.

   기본값은 2048입니다.
5. 저장을 누릅니다.
6. 서버를 다시 시작합니다.

HTTP 서비스 구성

이 장에서는 Application Server의 HTTP 서비스 구성 요소에 대한 가상 서버 및 HTTP Listener 구성 방법에 대해 설명합니다.

HTTP 서비스 정보

HTTP 서비스

HTTP 서비스는 웹 응용 프로그램을 배포하고 배포된 웹 응용 프로그램을 HTTP 클라이언트에서 액세스할 수 있게 하는 기능을 제공하는 Application Server의 구성 요소입니다. 115페이지의 "웹 응용 프로그램 배포"를 참조하십시오. 이 기능은 가상 서버와 HTTP Listener라는 두 종류의 관련된 객체 수단으로 제공됩니다.

가상 서버

가상 호스트라고도 하는 가상 서버는 동일한 물리적 서버가 여러 인터넷 도메인 이름을 호스트할 수 있게 해주는 객체입니다. 동일한 물리적 서버에서 호스트되는 모든 가상 서버는 해당 서버의 인터넷 프로토콜(IP) 주소를 공유합니다. 가상 서버는 서버의 도메인 이름(예: www.aaa.com)을 Application Server가 실행 중인 특정 서버와 연관시킵니다.

주: 인터넷 도메인을 Application Server의 관리 도메인과 혼동하지 마십시오.

예를 들어, 물리적 서버에서 다음 도메인을 호스트하려고 합니다.

```
www.aaa.com
www.bbb.com
www.ccc.com
```

www.aaa.com, www.bbb.com 및 www.ccc.com에 각각 연관된 web1, web2 및 web3 웹 모듈이 있다고 가정합니다.

이 모든 URL을 한 물리적 서버에서 처리한다는 것을 의미합니다.

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

첫 번째 URL은 www.aaa.com 가상 호스트에 매핑되고, 두 번째 URL은 www.bbb.com 가상 호스트에 매핑되며, 세 번째 URL은 www.ccc.com 가상 호스트에 매핑됩니다.

한편 web3이 www.bbb.com에 등록되지 않았기 때문에 다음 URL은 404개의 반환 코드를 발생합니다.

```
http://www.bbb.com:8080/web3
```

이 매핑이 작동하려면 www.aaa.com, www.bbb.com 및 www.ccc.com이 모두 작업 중인 물리적 서버의 IP 주소로 연결되어야 합니다. 이들을 네트워크의 DNS 서버에 등록해야 합니다. 그리고 UNIX 시스템의 경우 이 도메인을 /etc/hosts 파일에 추가합니다 (/etc/nsswitch.conf 파일의 hosts에 대한 설정에 files가 포함된 경우).

Application Server를 시작하면 다음과 같은 가상 서버를 자동으로 시작합니다.

- 모든 사용자 정의 웹 모듈을 호스트하는 server라고 하는 가상 서버
- 모든 관리 관련 웹 모듈(특히 관리 콘솔)을 호스트하는 __asadmin이라고 하는 가상 서버. 이 서버는 제한적이므로 이 가상 서버에 웹 모듈을 배포할 수 없습니다.

비작업 환경에서 웹 서비스를 개발, 테스트 및 배포하기 위해서는 대개 server만이 유일하게 필요한 가상 서버입니다. 작업 환경에서는 물리적 서버가 하나만 있더라도 고유한 웹 서버를 가진 것으로 표시되도록 추가 가상 서버에서 사용자와 고객을 위한 호스팅 기능을 제공합니다.

HTTP Listener

가상 서버마다 하나 이상의 HTTP listener를 통해 서버와 클라이언트간의 연결을 제공합니다. 각 HTTP Listener는 IP 주소, 포트 번호, 서버 이름 및 기본 가상 서버를 가진 수신 소켓입니다.

HTTP Listener의 포트 번호와 IP 주소의 조합은 고유해야 합니다. 예를 들어, HTTP listener는 IP 주소 0.0.0.0을 지정하여 시스템에 대해 지정된 포트에서 구성된 모든 IP 주소를 청취할 수 있습니다. 또는 HTTP Listener는 각 listener에 대해 고유한 IP 주소를 지정할 수 있지만 동일한 포트를 사용합니다.

HTTP Listener는 IP 주소와 포트 번호의 조합이기 때문에 동일한 IP 주소와 다른 포트 번호(예: 1.1.1.1:8081 및 1.1.1.1:8082)를 가지거나 다른 IP 주소와 동일한 포트 번호(예: 시스템이 이 두 주소에 응답하도록 구성된 경우 1.1.1.1:8081 및 1.2.3.4:8081)를 가진 여러 HTTP Listener가 있을 수 있습니다.

그러나 특정 포트에서 모든 IP 주소에서 수신하는 0.0.0.0 IP 주소를 HTTP Listener에서 사용할 경우, 같은 포트에서 특정한 IP 주소를 수신하는 추가 IP 주소를 위한 HTTP Listener를 만들 수 없습니다. 예를 들어, HTTP Listener에서 0.0.0.0:8080(포트 8080의 모든 IP 주소)을 사용할 경우 다른 HTTP Listener에서 1.2.3.4:8080을 사용할 수 없습니다.

Application Server를 실행 중인 시스템은 대개 하나의 IP 주소만 액세스하기 때문에 HTTP Listener는 대개 0.0.0.0 IP 주소와 다른 포트 번호를 사용합니다. 각 포트 번호는 다른 용도에 사용됩니다. 시스템이 둘 이상의 IP 주소에 액세스할 경우 각 주소를 다른 용도로 사용할 수 있습니다.

기본적으로 Application Server가 시작되면 다음과 같은 HTTP Listener를 갖고 있습니다.

- server라고 하는 가상 서버와 연관된 http-listener-1 및 http-listener-2라고 하는 두 개의 HTTP Listener. http-listener-1이라고 하는 Listener에는 보안이 활성화되지 않고, http-listener-2에는 보안이 활성화되어 있습니다.
- __asadmin이라고 하는 가상 서버와 연관된 admin-listener라고 하는 HTTP Listener. 이 Listener에는 보안이 활성화되어 있습니다.

이 Listener는 모두 IP 주소 0.0.0.0과 Application Server 설치 중에 HTTP 서버 포트 번호로 지정한 포트 번호를 사용합니다. Application Server가 기본 포트 번호 값을 사용할 경우 `http-listener-1`는 포트 8080을 사용하고, `http-listener-2`는 포트 8181을 사용하며, `admin-listener`는 포트 4849를 사용합니다.

HTTP Listener마다 기본 가상 서버가 있습니다. 기본 가상 서버는 HTTP listener가 호스트 구성 요소가 그 HTTP listener와 연관된 가상 서버와 일치하지 않는 모든 요청 URL을 라우팅하는 서버입니다. 가상 서버는 `http-listeners` 속성에 HTTP Listener를 나열하여 HTTP Listener와 연관됩니다.

그리고 HTTP listener의 억셉터 스레드 수를 지정합니다. 억셉트 스레드는 연결을 기다리는 스레드입니다. 스레드는 연결을 승인하고 이 연결을 작업자 스레드에서 선택할 수 있도록 연결 대기열이라고 하는 대기열에 둡니다. 새로운 요청이 있을 때 항상 사용할 수 있도록 억셉터 스레드를 충분히 구성합니다. 그러나 시스템에 많은 부담을 주지 않도록 적당히 구성합니다. 연결 대기열에는 억셉터 스레드에서 승인한 새로운 연결과 연결 유지 연결 관리 하위 시스템에서 관리하는 지속적인 연결이 모두 포함됩니다.

스레드 처리 요청은 연결 대기열에서 들어오는 HTTP 요청을 검색하여 요청을 처리합니다. 이 스레드는 HTTP 헤더를 구문 분석하고, 해당하는 가상 서버를 선택한 다음, 요청 처리 엔진을 실행하여 요청을 서비스합니다. 처리할 요청이 더 이상 없지만 연결에서 HTTP/1.1을 사용하거나 `Connection: keep-alive` 헤더를 전송하여 지속성을 유지할 수 있는 경우 요청 처리 스레드에서는 연결이 유휴 상태인 것으로 가정하고 그 연결을 연결 유지 연결 관리 하위 시스템으로 전달합니다.

연결 유지 하위 시스템은 그러한 유휴 연결을 정기적으로 폴링하고 앞으로 처리하기 위해 해당 연결을 연결 대기열에 둡니다. 여기에서 요청 처리 스레드는 다시 연결을 검색하고 요청을 처리합니다. 연결 유지 하위 시스템은 수 만개의 연결을 관리할 수 있기 때문에 다중 스레딩되어 있습니다. 연결 개수를 더 작은 하위 집합으로 나누는 효율적인 폴링 기술을 사용하여 요청이 준비된 연결과 닫힌 것으로 간주해도 될 정도로(허용 가능한 최대 연결 유지 시간을 초과한 것으로) 충분히 유휴 상태였던 연결을 확인합니다.

HTTP listener의 서버 이름은 서버가 리디렉션의 일부로 클라이언트에 전송한 URL에 표시되는 호스트 이름입니다. 이 속성은 서버에서 자동으로 생성하는 URL에 영향을 주지만, 서버에 저장된 디렉토리 및 파일의 URL에는 영향을 주지 않습니다. 서버에서 별칭을 사용하는 경우 이 이름은 대개 별칭 이름입니다. 클라이언트가 `Host:` 헤더를 보내면 호스트 이름이 리디렉션에서 HTTP listener 서버 이름-값을 대체합니다.

원래 요청에 지정된 것과는 다른 포트 번호를 사용하도록 리디렉션 포트를 지정합니다. 다음 상황 중 하나에서 *리디렉션*이 발생합니다.

- 클라이언트가 지정한 URL에 더 이상 존재하지 않는 자원(즉, 다른 위치로 이동한 자원)에 액세스할 경우 서버는 404를 반환하는 대신 지정한 응답 코드를 반환하고 응답의 위치 헤더에 새로운 위치를 포함시켜서 클라이언트를 새로운 위치로 리디렉션합니다.
- 클라이언트가 정규 HTTP 포트에서 보호된(예: SSL) 자원에 액세스할 경우 서버는 요청을 SSL 가능 포트로 리디렉션합니다. 이 경우 서버는 원래 비보안 포트가 SSL 가능 포트로 대체된 위치 응답 헤더에 새로운 URL을 반환합니다. 그러면 클라이언트가 이 새로운 URL에 연결됩니다.

HTTP listener에 대한 보안 활성화 여부 및 사용한 보안 종류(예: SSL 프로토콜 및 암호화)도 지정합니다.

Application Server에 배포된 웹 응용 프로그램에 액세스하려면 웹 응용 프로그램에 지정된 컨텍스트 루트와 함께 URL `http://localhost:8080/`(보안 처리된 응용 프로그램일 경우 `https://localhost:8181/`)을 사용합니다. 관리 콘솔에 액세스하려면 URL `https://localhost:4849/` 또는 `https://localhost:4849/asadmin/`(기본 컨텍스트 루트)을 사용합니다.

가상 서버에서 기존 HTTP listener를 지정해야 하고 다른 가상 서버에서 이미 사용 중인 HTTP listener는 지정할 수 없기 때문에 새로운 가상 서버를 만들기 전에 최소한 하나의 HTTP listener를 만들어야 합니다.

HTTP 서비스의 관리 콘솔 작업

HTTP 서비스 구성

HTTP 서비스를 구성하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.

b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.

3. HTTP 서비스 노드를 선택합니다.

4. HTTP 서비스 페이지에서 모든 서비스의 HTTP listener에 적용되는 등록 정보를 설정할 수 있습니다.

다음 표에 등록 정보가 나열되어 있습니다.

표 17-1 HTTP 서비스 등록 정보

등록 정보 이름	설명	기본값
traceEnabled	True로 설정한 경우 TRACE 작업이 활성화됩니다. Application Server가 사이트간 스크립팅 공격에 덜 민감하게 하려면 이 등록 정보를 False로 설정합니다.	false
monitoringCacheEnabled	True로 설정하면 Application Server는 HTTP 서비스의 통계 로컬 값을 캐시하여 통계 쿼리에 응답합니다. 이 값은 성능을 향상시킵니다. False로 설정하면 Application Server는 통계 값마다 HTTP 서비스에 쿼리합니다.	true
monitoringCacheRefreshInMillis	모니터링 캐시를 새로 고치는 간격(밀리초)을 지정합니다.	5000
sslCacheEntries	캐시할 수 있는 SSL 세션 개수를 지정합니다. 상한은 없습니다.	10000
sslSessionTimeout	SSL2 세션이 얼마 후에 시간 초과되는지 시간(초)을 지정합니다.	100
ssl3SessionTimeout	SSL3 세션이 얼마 후에 시간 초과되는지 시간(초)을 지정합니다.	86400
sslClientAuthDataLimit	클라이언트 인증서 핸드셰이크 단계 중에 버퍼링되는 응용 프로그램 데이터의 최대량(바이트)을 지정합니다.	1048576
sslClientAuthTimeout	클라이언트 인증서 핸드셰이크 단계가 얼마 후에 시간 초과되는지 시간(초)을 지정합니다.	60
keepAliveQueryMeanTime	원하는 연결 유지 대기 시간(밀리초)을 지정합니다.	100
keepAliveQueryMaxSleepTime	앞으로의 요청을 위해 연결 유지 연결을 폴링한 후 일시 정지하는 상한 시간(밀리초)을 지정합니다.	100

표 17-1 HTTP 서비스 등록 정보 *(계속)*

등록 정보 이름	설명	기본값
stackSize	원 스레드의 최대 스택 크기를 지정합니다.	OS/ 시스템 종속
statsProfilingEnabled	False로 설정하면 HTTP 서비스의 모니터링 통계 기록이 비활성화되므로 성능이 향상됩니다. 이 등록 정보를 false로 설정하면 HTTP 서비스에 대한 모니터링 활성화가 아무런 영향을 주지 않습니다.	true
chunkedRequestBufferSize	요청 데이터를 청크 해제하는 데 사용하는 기본 버퍼 크기(바이트)를 지정합니다.	8192
chunkedRequestTimeoutSeconds	요청 데이터의 청크 해제 기본 시간 초과(초)를 지정합니다.	60
dnsCacheEnabled	True로 설정하면 사용자가 DNS 캐싱과 관련된 통계를 모니터링할 수 있습니다. HTTP 프로토콜 탭의 DNS 조회 상자를 선택한 경우에만 이 등록 정보가 효과가 있습니다. 그렇지 않으면 등록 정보 설정이 무시됩니다.	false

5. 액세스 로그 탭을 눌러 액세스 로그 회전을 구성합니다. 다른 탭을 눌러 요청 처리, 연결 유지 하위 시스템, 연결 풀, HTTP 프로토콜 및 HTTP 파일 캐시를 구성합니다.
6. 저장을 누릅니다.

HTTP 서비스 액세스 로그 구성

이 페이지를 사용하여 가상 서버에 대한 액세스 로그의 회전을 활성화 및 구성합니다. 이 로그는 *domain_root_dir*/*domain_dir*/logs/access 디렉토리에 있고 다음과 같이 이름이 지정됩니다.

*virtual_server_name*_access_log.*yyyy-mm-dd*.txt

기본값 로드를 눌러 기본값을 로드합니다. 이 로그의 회전 등록 정보를 변경하려면 다음 작업을 수행합니다.

- 파일 회전 상자를 선택하여 파일 회전을 활성화합니다. 기본적으로 파일 회전이 활성화됩니다.

- 회전 정책 드롭다운 목록에서 정책을 선택합니다. 사용 가능한 유일한 정책은 time입니다.
- 회전 간격 필드에서 숫자 값을 입력하여 액세스 로그 회전 간 시간(분)을 지정합니다. 회전 정책이 time일 경우에만 이 필드가 유효합니다. 기본값은 1440분입니다.
- 회전 접미어 필드에서 문자열 값을 입력하여 회전 후 로그 파일 이름에 추가되는 접미어를 지정합니다. 기본값은 %YYYY;%MM;%DD;-%hh;h%mm;m%ss;s입니다.
- 형식 필드에서 문자열 값을 입력하여 액세스 로그의 형식을 지정합니다. 아래 표에 표시된 형식을 사용합니다. 기본 형식은 %client.name% %auth-user-name% %datetime% %request% %status% %response.length%입니다.

표 17-2 액세스 로그 형식의 토큰 값

데이터	토큰
클라이언트 호스트 이름	%client.name%
클라이언트 DNS	%client.dns%
시스템 날짜	%datetime%
전체 HTTP 요청 행	%request%
상태	%status%
응답 내용 길이	%response.length%
참조자 헤더	%header.referer%
사용자 에이전트	%header.user-agent%
HTTP 메소드	%http-method%
HTTP URI	%http-uri%
HTTP 쿼리 문자열	%query-str%
HTTP 프로토콜 버전	%http-version%
승인 헤더	%header.accept%
날짜 헤더	%header.date%
If-Modified-Since 헤더	%header.if-mod-since%
인증 헤더	%header.auth%
RFC 2616에 정의된 유효한 HTTP 헤더 값(any도 유효한 헤더 값이며 여기서 변수로 지정됨)	%header.*any*%
인증된 사용자 이름	%auth-user-name%

표 17-2 액세스 로그 형식의 토큰 값 *(계속)*

데이터	토큰
쿠키 값	%cookie.value%
가상 서버 아이디	%vs.id%

- 저장을 눌러 변경 사항을 저장하거나 기본값 로드를 눌러 기본 설정으로 돌아갑니다.

HTTP 서비스 요청 처리 스레드 구성

이 페이지를 사용하여 HTTP 서비스에 대한 요청 처리 스레드를 구성합니다.

- 기본값 로드를 눌러 기본값을 로드합니다.
- 스레드 수 필드에 숫자 값을 입력하여 요청 처리 스레드 최대 수를 지정합니다. 기본값은 128입니다.
- 초기 스레드 수 필드에 서버를 시작할 때 사용 가능한 요청 처리 스레드 수를 입력합니다. 기본값은 48입니다.
- 스레드 증분 필드에 요청 수가 초기 스레드 수를 초과할 경우 추가되는 요청 처리 스레드 수를 입력합니다. 기본값 10입니다.
- 요청 시간 초과 필드에서 요청이 시간 초과된 후의 시간(초)을 입력합니다. 기본값은 30초입니다.
- 버퍼 길이 필드에 요청 처리 스레드가 요청 데이터를 읽는 데 사용하는 버퍼 크기(바이트)를 입력합니다. 기본값은 4096 바이트입니다.
- 저장을 눌러 변경 사항을 저장하거나 기본값 로드를 눌러 기본 설정으로 돌아갑니다.

HTTP 서비스 연결 유지 하위 시스템 구성

이 페이지를 사용하여 HTTP 서비스의 연결 유지 하위 시스템을 구성합니다.

- 기본값 로드를 눌러 기본값을 로드합니다.

- 스레드 수 필드에 사용할 연결 유지 스레드 수를 입력합니다. 기본값은 1입니다.
- 최대 연결 필드에서 유지 관리할 지속적인 연결 최대 수를 입력합니다. 기본값은 256입니다.
- 시간 초과 필드에서 연결 유지 연결을 열어 두는 최대 시간(초)을 입력합니다. 기본값은 30초입니다.
- 저장을 눌러 변경 사항을 저장하거나 기본값 로드를 눌러 기본 설정으로 돌아갑니다.

참고 항목:

- HTTP Listener

HTTP 서비스 연결 풀 구성

이 페이지를 사용하여 HTTP 서비스 연결 대기열의 연결 풀을 구성합니다.

- 기본값 로드를 눌러 기본값을 로드합니다.
- 보류 중인 최대 연결 수 필드에서 HTTP listener에 허용되는 보류 중인 연결 최대 수를 입력합니다. 기본값은 4096입니다.
- 대기열 크기 필드에서 연결 대기열의 최대 크기(바이트)를 입력합니다. 이 값은 서버에 유지할 수 있는 해결되지 않은 연결 최대 수도 지정합니다. 기본값은 4096입니다.
- 수신 버퍼 크기 필드에서 HTTP listener에 대한 수신 버퍼 크기를 입력합니다. 기본값은 4096입니다.
- 전송 버퍼 크기 필드에서 HTTP listener에 대한 전송 버퍼 크기를 입력합니다. 기본값은 8192입니다.
- 저장을 눌러 변경 사항을 저장하거나 기본값 로드를 눌러 기본 설정으로 돌아갑니다.

HTTP 서비스에 대해 HTTP 프로토콜 구성

이 페이지를 사용하여 HTTP 서비스에 대해 HTTP 프로토콜을 구성합니다.

- 기본값 로드를 눌러 기본값을 로드합니다.

- 버전 필드에서 사용할 HTTP 프로토콜의 버전(`HTTP/1.0` 또는 `HTTP/1.1`)을 입력합니다. 기본값은 `HTTP/1.1`입니다.
- DNS 조회 상자를 선택하여 클라이언트에 대한 DNS 항목 조회를 활성화합니다. 기본값은 false입니다.
- SSL 상자에서 확인 표시를 제거하여 서버의 보안을 전역적으로 비활성화합니다. 보안이 활성화된 Listener에 대해 SSL을 사용할 수 있게 하려면 이 값을 true로 설정해 둡니다. 기본값은 true입니다.
- 강제 응답 유형 필드에서 확장자와 일치하는 사용 가능한 MIME 매핑이 없을 경우 사용할 응답 유형을 입력합니다. 기본값은 `text/html; charset=iso-8859-1`입니다.
- 기본 응답 유형 필드에서 기본 응답 유형을 입력합니다. 기본값은 `text/html; charset=iso-8859-1`입니다. 값은 내용 유형, 인코딩, 언어 및 문자 집합으로 구성된 세미콜론으로 구분된 문자열입니다.
- 저장을 눌러 변경 사항을 저장하거나 기본값 로드를 눌러 기본 설정으로 돌아갑니다.

HTTP 서비스에 대해 HTTP 파일 캐시 구성

이 페이지를 사용하여 HTTP 서비스에 대해 HTTP 파일 캐시를 구성합니다.

파일 캐시는 정적 내용을 저장하므로 서버에서 관련 내용에 대한 요청을 신속하게 처리합니다.

- 기본값 로드를 눌러 기본값을 로드합니다.
- 전역 상자를 선택하여 파일 캐시를 활성화합니다. 기본값은 true입니다.
- 파일 전송 상자를 선택하여 Windows에서 `TransmitFileSystem` 메소드 사용을 활성화합니다. 기본값은 false입니다.
- 최대 사용 시간 필드에서 유효한 캐시 항목의 최대 사용 시간(초)을 입력합니다. 기본값은 30초입니다.
- 최대 파일 수 필드에서 파일 캐시의 최대 파일 수를 입력합니다. 기본값은 1,024입니다.
- 해시 초기 크기 필드에서 해시 버킷의 초기 수를 입력합니다. 기본값은 0입니다.
- 중간 파일 크기 제한 필드에서 메모리 매핑된 파일로 캐시할 수 있는 파일의 최대 크기(바이트)를 입력합니다. 기본값은 537,600바이트입니다.

- 중간 파일 크기 필드에서 메모리 매핑된 파일로 캐시된 모든 파일의 전체 크기(바이트)를 입력합니다. 기본값은 10,485,760바이트입니다.
- 작은 파일 크기 제한 필드에서 메모리로 읽어 들일 수 있는 파일의 최대 크기(바이트)를 입력합니다. 기본값은 2,048바이트입니다.
- 작은 파일 크기 필드에서 메모리로 읽어 들인 모든 파일의 전체 크기(바이트)를 입력합니다. 기본값은 1,048,576바이트입니다.
- 파일 캐싱 사용 가능 드롭다운 목록에서 ON 또는 OFF를 선택하여 파일 크기가 중간 파일 크기 제한보다 작을 경우 파일 내용을 캐싱할지 하지 않을지를 설정합니다. 기본값은 ON입니다.
- 저장을 눌러 변경 사항을 저장하거나 기본값 로드를 눌러 기본 설정으로 돌아갑니다.

가상 서버의 관리 콘솔 작업

가상 서버 만들기

가상 서버를 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. HTTP 서비스 노드를 확장합니다.
4. 가상 서버 노드를 선택합니다.
5. 가상 서버 페이지에서 새로 만들기를 누릅니다. 가상 서버 만들기 페이지가 표시됩니다.

6. 아이디 필드에서 가상 서버에 대한 고유한 이름을 입력합니다. 이 값을 사용하여 가상 서버를 내부적으로 식별합니다. HTTP 클라이언트에게는 이 값이 노출되지 않습니다. HTTP 클라이언트에게 노출되는 호스트 이름은 호스트 필드에서 지정해야 합니다.
7. 호스트 필드에서 서버가 실행 중인 시스템의 이름이나 호스트 이름을 입력합니다. 네트워크의 DNS 서버(UNIX 시스템의 경우 `/etc/hosts` 파일)에 등록된 실제 또는 가상 호스트 이름을 사용합니다.
8. 반대 상태 영역에서 설정, 해제 또는 사용 안 함을 선택합니다. 기본값은 설정입니다.
9. HTTP Listener 필드를 비워둡니다. HTTP listener를 만들어 이 서버와 연관시키면 자동으로 입력됩니다.

   이 필드를 사용하려면 기존 HTTP listener를 지정해야 합니다. 그러나 다른 가상 서버에서 사용하는 Listener는 지정하지 마십시오. 지정할 경우 서버 로그에 오류가 표시됩니다. Listener를 만들 때 기존 가상 서버와 연관시켜야 하므로 모든 기존 listener는 다른 가상 서버에서 사용하고 있습니다.
10. 기본 웹 모듈 드롭다운 목록에서 해당 가상 서버에 배포된 다른 웹 모듈에 매핑할 수 없는 모든 요청에 응답하는 배포된 웹 모듈을 선택합니다.

    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.
11. 로그 파일 필드에서 이 가상 서버의 로깅 메시지가 표시되는 파일의 경로 이름을 입력합니다. 로깅 메시지를 기본 서버 로그, *domain_root_dir*/*domain_dir*/`logs/server.log`에 전송하려면 이 필드를 비워둡니다.
12. 추가 등록 정보 영역에서 등록 정보 추가를 눌러 가상 서버에 대한 등록 정보를 추가합니다. 등록 정보 지정 여부와 상관없이 새로운 서버에는 기본 등록 정보 `docroot` 및 `accesslog`가 기본값으로 설정됩니다.
13. 확인을 눌러 가상 서버를 저장합니다.

다음 표에서는 사용 가능한 등록 정보를 나열합니다.

표 17-3 가상 서버 등록 정보

등록 정보 이름	설명
`docroot`	서버의 루트 문서 디렉토리 절대 경로입니다. 기본값은 *domain_root_dir*/*domain_dir*/`docroot`입니다.

표 17-3 가상 서버 등록 정보 *(계속)*

등록 정보 이름	설명
accesslog	서버 액세스 로그 절대 경로입니다. 기본값은 *domain_root_dir*/*domain_dir*/logs/access입니다.
sso-enabled	False일 경우 이 가상 서버에 대해 단일 사인 온이 비활성화되므로 사용자는 가상 서버의 모든 응용 프로그램마다 별도로 인증해야 합니다. 서블릿과 JSP 페이지에서는 Application Server의 응용 프로그램에서 단일 사인 온을 지원합니다. 이 기능을 사용하면 응용 프로그램마다 사용자가 별도로 사인 온할 필요 없이 같은 사용자 사인 온 정보를 여러 응용 프로그램에서 공유할 수 있습니다. 기본값은 true입니다.
sso-max-inactive-seconds	클라이언트 작업 수신 없이 얼마 동안 사용자의 단일 사인 온 레코드가 유효한지 시간(초)을 지정합니다. 단일 사인 온은 같은 가상 서버의 여러 응용 프로그램에 모두 적용되므로 한 응용 프로그램만 액세스해도 단일 사인 온 레코드가 활성화 상태로 유지됩니다. 기본값은 300초(5분)입니다. 값이 클수록 사용자에 대한 단일 사인 온 지속성이 더 길어지지만 서버에서 더 많은 메모리를 사용하게 됩니다.
sso-reap-interval-seconds	만료된 단일 사인 온 레코드의 제거 간격(초)을 지정합니다. 기본값은 60입니다.
allowLinking	True일 경우 심볼릭 링크인 자원이 이 가상 서버에 배포된 모든 웹 응용 프로그램에 대해 작동합니다. sun-web.xml 파일의 sun-web-app 등록 정보 allowLinking을 사용하여 개별 웹 응용 프로그램이 이 설정을 대체할 수 있습니다. `<sun-web-app>` `  <property name="allowLinking" value="{true|false}"/>` `</sun-web-app>` 기본값은 true입니다.

해당 asadmin 명령: create-virtual-server

가상 서버 편집

가상 서버를 편집하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.

   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server는 server-config 노드를 선택합니다.

   b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.

3. HTTP 서비스 노드를 확장합니다.
4. 가상 서버 노드를 선택합니다.
5. 편집할 가상 서버를 선택합니다.
6. 가상 서버 편집 페이지에서 다음 작업을 수행할 수 있습니다.
   - 호스트 필드의 호스트 이름을 변경합니다.
   - 상태 설정 값을 변경합니다.
   - HTTP listener를 추가하거나 제거합니다.
   - 기본 웹 모듈 선택을 변경합니다.
   - 로그 파일 값을 변경합니다.
   - 등록 정보를 추가, 제거 또는 수정합니다.
7. 저장을 눌러 변경 사항을 저장합니다.

가상 서버 삭제

가상 서버를 삭제하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.

   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server는 server-config 노드를 선택합니다.

   b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.

3. HTTP 서비스 노드를 확장합니다.
4. 가상 서버 노드를 선택합니다.
5. 가상 서버 페이지에서 삭제할 가상 서버 이름 옆에 있는 상자를 선택합니다.

6. 삭제를 누릅니다.

__asadmin 가상 서버를 삭제할 수 있지만 권장하지 않습니다. 삭제할 경우 먼저 Application Server의 domain.xml 파일에 있는 virtual-server 요소를 안전한 장소에 복사하여 필요할 경우 설정을 복구할 수 있게 합니다.

해당 asadmin 명령: delete-virtual-server

HTTP Listener의 관리 콘솔 작업

HTTP Listener 만들기

HTTP listener를 만들려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. default-config 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. HTTP 서비스 노드를 확장합니다.
4. HTTP Listener 노드를 선택합니다.
5. HTTP Listener 페이지에서 새로 만들기를 누릅니다. HTTP Listener 만들기 페이지가 표시됩니다.
6. 이름 필드에서 listener에 대한 이름을 입력합니다.
7. 서버를 다시 시작할 때 Listener를 사용하지 않을 경우 Listener 필드에서 사용 가능 상자의 선택 표시를 제거합니다.
8. Listener가 고유한 포트 값을 사용하여 서버의 모든 IP 주소에서 수신하게 할 경우 네트워크 주소 필드에 0.0.0.0을 입력합니다. 그렇지 않으면 서버에 대한 유효한 IP 주소를 입력합니다.

9. Listener 포트 필드에 네트워크 주소 필드가 0.0.0.0일 경우 고유한 포트 값을 입력하거나 다른 IP 주소를 사용할 경우 원하는 포트 값을 입력합니다.
10. 기본 가상 서버 드롭다운 목록에서 가상 서버를 선택합니다.
11. 서버 이름 필드에서 서버가 클라이언트에 전송하는 URL에서 사용할 호스트 이름을 입력합니다. 서버에서 별칭을 사용하는 경우 이 이름은 별칭 이름입니다. 서버에서 별칭을 사용하지 않을 경우 이 필드를 비워둡니다.
12. 고급 영역에서 다음 작업을 수행합니다.
    - 요청을 다른 포트로 리디렉션하려면 리디렉션 포트 필드에 값을 입력합니다. 다음 두 가지 조건이 있을 경우 Application Server는 요청을 자동으로 리디렉션합니다.
      - 이 listener가 비 SSL 요청을 지원합니다.
      - 일치하는 보안 제한 조건에서 SSL 전송을 요구하는 요청을 수신합니다.

      기본적으로 Application Server는 원래 요청에 지정된 포트 번호를 사용합니다.
    - 억셉터 스레드 수를 변경합니다.
    - 서블릿에서 생성한 HTTP 응답 헤더에 `X-Powered-By: Servlet/2.4` 헤더가 포함되지 않도록 하려면 제공 상자에서 확인 표시를 제거합니다.

      Java Servlet 2.4 사양에서는 이 헤더를 정의하여 컨테이너가 서블릿에서 생성한 응답에 추가할 수 있습니다. 마찬가지로 JavaServer Pages™ (JSP™) 2.0 사양에서는 JSP 기술을 사용하는 응답에 경우에 따라 추가할 `X-Powered-By: JSP/2.0` 헤더를 정의합니다. 웹 응용 프로그램의 경우 기본적으로 `X-Powered-By: JSP/2.0` 헤더 포함이 활성화되어 있습니다. 이 헤더의 목적은 서블릿 및 JSP 기술 사용에 대한 통계 데이터를 수집하는 데 있어 웹 사이트 관리자를 지원하는 것입니다.

      JSP 페이지에 대한 `X-Powered-By` 헤더 활성화 및 비활성화에 대한 자세한 내용은 *Application Server Developer's Guide*의 "Deployment Descriptor Files" 장을 참조하십시오. 이 문서 링크는 51페이지의 "추가 정보"를 참조하십시오.

      작업 환경에서 `X-Powered-By` 헤더 생성을 생략하여 사용 기술을 숨길 수 있습니다.
13. 보안되지 않은 listener를 만들려면 확인을 누릅니다.

이 페이지의 SSL 절에서 SSL, TLS 또는 SSL 및 TLS 보안 둘 다를 사용하도록 Listener를 구성할 수 있습니다.

보안 listener를 설정하려면, 다음 작업을 수행합니다.

1. 보안 필드에서 사용 가능 확인란을 선택합니다.
2. 이 Listener를 사용할 때 사용자가 자신을 서버에 인증하게 하려면 클라이언트 인증 필드에서 사용 가능 상자를 선택합니다.
3. 인증서 별명 필드에 기존 서버 키 쌍 및 인증서의 이름을 입력합니다. 자세한 내용은 보안 장을 참조하십시오.
4. SSL3/TLS 섹션:
   a. Listener에서 활성화할 보안 프로토콜을 선택합니다. SSL3, TLS 또는 둘 다를 선택합니다.
   b. 프로토콜이 사용하는 암호 제품군을 선택합니다. 모든 암호 제품군을 사용하려면 지원되는 모든 암호화 제품군을 선택합니다. 개별 암호 제품군을 사용할 수도 있습니다.

이제 기본 가상 서버로 지정된 가상 서버의 HTTP Listener 필드에 Listener가 나열됩니다.

해당 `asadmin` 명령: `create-http-listener`, `create-ssl`

HTTP Listener 편집

HTTP listener를 편집하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. `default-config` 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. HTTP 서비스 노드를 확장합니다.
4. HTTP Listener 노드를 선택합니다.
5. 편집할 HTTP listener를 선택합니다.
6. HTTP Listener 편집 페이지에서 설정을 수정합니다.

7. 저장을 눌러 변경 사항을 저장합니다.

HTTP Listener 삭제

HTTP listener를 삭제하려면 다음 단계를 수행합니다.

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어, 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. `default-config` 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. HTTP 서비스 노드를 확장합니다.
4. HTTP Listener 노드를 선택합니다.
5. HTTP Listener 페이지에서 삭제할 HTTP Listener 이름 옆에 있는 상자를 선택합니다.
6. 삭제를 누릅니다.

`http-listener-1`, `http-listener-2` 및 `admin-listener` HTTP listener를 삭제할 수 있지만 권장하지 않습니다. 삭제할 경우 먼저 Application Server의 `domain.xml` 파일에 있는 `http-listener` 요소를 안전한 장소에 복사하여 필요할 경우 설정을 복구할 수 있게 합니다.

해당 `asadmin` 명령: `delete-http-listener`

ORB(Object Request Broker) 구성

이 장에서는 ORB(Object Request Broker) 및 IIOP listener를 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

ORB(Object Request Broker) 정보

CORBA

Application Server는 표준 프로토콜 및 형식 집합을 지원하여 상호 운용성을 보장합니다. 이 프로토콜 중 일부는 CORBA에서 정의합니다.

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 호출된 메소드에 대한 서비스 공급자와 매개 변수의 객체 이름(객체 참조라고 함)을 포함하여 수행해야 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.

ORB

ORB(Object Request Broker)는 CORBA의 핵심 구성 요소입니다. ORB는 객체를 식별해서 찾고, 연결 관리를 처리하고, 데이터를 전송하며, 통신을 요청하는 데 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. 원격 ORB는 적절한 객체를 찾고, 요청을 처리하며, 결과를 반환합니다.

IIOP는 RMI-IIOP를 사용하여 응용 프로그램이나 객체에서 RMI(Remote Method Invocation) 프로토콜로 사용할 수 있습니다. Enterprise Bean(EJB 모듈)의 원격 클라이언트는 RMI-IIOP를 통해 Application Server와 통신합니다.

IIOP Listener

IIOP listener는 Enterprise Bean의 원격 클라이언트와 다른 CORBA 기반 클라이언트에서 들어오는 연결을 받아들이는 수신 소켓입니다. Application Server에 대해 여러 IIOP Listener를 구성할 수 있습니다. 각 listener에 대해 포트 번호, 네트워크 주소 그리고 필요에 따라 보안 속성을 지정합니다. 자세한 내용은 323페이지의 "IIOP Listener 만들기"를 참조하십시오.

ORB에 대한 관리 콘솔 작업

ORB 구성

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.

a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.

b. `default-config` 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.

3. ORB 노드를 선택합니다.
4. 스레드 풀 아이디 드롭다운 목록에서 ORB가 사용하는 스레드 풀을 선택합니다.

   ORB는 스레드 풀을 사용하여 RMI-IIOP를 통해 통신하는 다른 클라이언트와 Enterprise Bean의 원격 클라이언트에서 들어오는 요청에 응답합니다. 자세한 내용은 327페이지의 "Application Server의 스레드 풀" 및 328페이지의 "스레드 풀 만들기"를 참조하십시오.
5. 최대 메시지 단편 크기 필드에서 IIOP 메시지에 대한 최대 단편 크기를 설정합니다.

   이 크기보다 큰 메시지는 단편화됩니다.
6. 총 연결 수 필드에서 모든 IIOP listener에 대한 들어오는 연결의 최대 수를 설정합니다.
7. IIOP 클라이언트 인증이 필요한 경우 필요 확인란을 선택합니다.
8. 저장을 눌러 변경 사항을 저장하거나, 기본값 로드를 눌러 기본값을 로드합니다.
9. 서버를 다시 시작합니다.

IIOP Listener에 대한 관리 콘솔 작업

IIOP Listener 만들기

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.

   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.

   b. `default-config` 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.

3. ORB 노드를 확장합니다.
4. IIOP Listener를 선택합니다.
5. 새로 만들기를 누릅니다.
6. 이름 필드에서 listener를 식별하는 이름을 입력합니다.
7. 네트워크 주소 필드에서 listener의 네트워크 주소를 입력합니다.

   이 주소는 IP 주소 또는 DNS 확인 가능한 호스트 이름일 수 있습니다.
8. Listener 포트 필드에서 Listener가 수신하는 포트 번호를 입력합니다.
9. Listener를 활성화하려면 Listener 필드에서 사용 가능 확인란을 선택합니다.
10. 추가 등록 정보 영역에서 응용 프로그램이 요구하는 등록 정보 값을 제공합니다.
11. 보안되지 않은 Listener를 만들려면 확인을 누릅니다.

이 페이지의 보안 섹션에서 Listener가 SSL, TLS 또는 SSL 및 TLS 보안을 모두 사용하도록 구성할 수 있습니다.

보안 listener를 설정하려면 다음 작업을 수행합니다.

1. 보안 필드에서 사용 가능 확인란을 선택합니다.
2. 이 listener를 사용할 경우 클라이언트가 자신을 서버에 인증하도록 하려면 클라이언트 인증 필드에서 사용 가능 확인란을 선택합니다.
3. 인증서 별명 필드에 기존 서버의 키 쌍과 인증서를 입력합니다.
4. SSL3/TLS 섹션:

   a. Listener에서 활성화할 보안 프로토콜을 선택합니다. SSL3 또는 TLS를 선택하거나 두 가지 프로토콜을 모두 활성화합니다.

   b. 프로토콜이 사용하는 암호 제품군을 선택합니다. 모든 암호 제품군을 사용하려면 지원되는 모든 암호화 제품군을 선택합니다. 개별 암호 제품군을 활성화할 수도 있습니다.
5. 확인을 누릅니다.

이제 IIOP Listener 페이지의 현재 Listener 테이블에 Listener가 나열됩니다.

해당 `asadmin` 명령: `create-iiop-listener`, `create-ssl`

IIOP Listener 편집

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. `default-config` 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. ORB 노드를 확장합니다.
4. IIOP Listener 노드를 선택합니다.
5. 현재 Listener 테이블에서 수정할 Listener를 선택합니다.
6. Listener의 설정을 수정합니다. 수정 가능한 필드의 설명은 323페이지의 "IIOP Listener 만들기"를 참조하십시오.
7. Listener의 포트 번호를 변경한 경우 서버를 다시 시작합니다.

IIOP Listener 삭제

1. 트리 구성 요소에서 구성 노드를 확장합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. `default-config` 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. ORB 노드를 확장합니다.
4. IIOP Listener 노드를 선택합니다.
5. 현재 Listener 테이블에서 삭제할 Listener를 선택합니다.
6. 삭제를 누릅니다.

해당 `asadmin` 명령: `delete-iiop-listener`

스레드 풀

이 장에서는 스레드 풀을 작성, 편집 및 삭제하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

스레드 풀 정보

이 절에서는 스레드 풀과 Application Server에서 작동 방법에 대해 설명합니다.

Application Server의 스레드 풀

Java Virtual Machine(JVM)에서는 한 번에 여러 스레드 실행을 지원할 수 있습니다. 성능 향상을 위해 Application Server에서는 하나 이상의 스레드 풀을 유지합니다. 특정 스레드 풀을 커넥터 모듈과 ORB에 할당할 수 있습니다.

하나의 스레드 풀이 여러 커넥터 모듈과 Enterprise Bean을 처리할 수 있습니다. 요청 스레드는 응용 프로그램 구성 요소에 대한 사용자 요청을 처리합니다. 서버는 요청을 받으면 요청을 스레드 풀의 여유 스레드에 할당합니다. 스레드는 클라이언트의 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 프로그램의 요청에 예약된 스레드의 최소 수와 최대 수를 지정합니다. 스레드 풀은 이러한 두 값 사이에서 동적으로 조절됩니다. 지정한 최소 스레드 풀 크기는 응용 프로그램 요청에 대한 예약에 적어도 그 수만큼의 스레드를 할당하라는 신호를 서버에게 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.

Application Server 스레드를 서로 다른 스레드 풀로 분리하여 하나의 자원 어댑터나 응용 프로그램이 Application Server의 모든 스레드를 사용하는 스레드 결핍을 방지합니다.

스레드 풀의 관리 콘솔 작업

스레드 풀 만들기

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 `server`는 `server-config` 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 `default-config` 노드를 선택합니다.
3. 스레드 풀 노드를 선택합니다.
4. 현재 풀에서 새로 만들기를 누릅니다.
5. 스레드 풀 아이디 필드에 스레드 풀 이름을 입력합니다.
6. 최소 스레드 풀 크기 필드에 이 대기열에서 요청을 처리하는 스레드 풀의 최소 스레드 수를 입력합니다.

   이 스레드 풀을 인스턴스화할 경우 이 스레드가 위에 만들어집니다.
7. 최대 스레드 풀 크기 필드에 이 대기열에서 요청을 처리하는 스레드 풀의 최대 스레드 수를 입력합니다.

   스레드 풀에 존재하는 스레드 수에 대한 상한 값입니다.
8. 유휴 시간 초과 필드에 풀에서 유휴 스레드를 제거하는 시간(초)을 입력합니다.

9. 작업 대기열 수 필드에 이 스레드 풀에서 처리하는 총 작업 대기열 수를 입력합니다.
10. 확인을 누릅니다.
11. 서버를 다시 시작합니다.

해당 asadmin 명령: create-threadpool

스레드 풀 편집

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 스레드 풀 노드를 선택합니다.
4. 현재 풀에서 변경할 스레드 풀 이름을 선택합니다.
5. 최소 스레드 풀 크기 필드에 이 대기열에서 요청을 처리하는 스레드 풀의 최소 스레드 수를 입력합니다.

   이 스레드 풀을 인스턴스화할 경우 이 스레드가 위에 만들어집니다.
6. 최대 스레드 풀 크기 필드에 이 대기열에서 요청을 처리하는 스레드 풀의 최대 스레드 수를 입력합니다.

   스레드 풀에 존재하는 스레드 수에 대한 상한 값입니다.
7. 유휴 시간 초과 필드에 풀에서 유휴 스레드를 제거하는 시간(초)을 입력합니다.
8. 작업 대기열 수 필드에 이 스레드 풀에서 처리하는 총 작업 대기열 수를 입력합니다.
9. 저장을 누릅니다.
10. 서버를 다시 시작합니다.

스레드 풀 삭제

1. 트리 구성 요소에서 구성 노드를 선택합니다.
2. 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. 모든 인스턴스의 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 스레드 풀 노드를 선택합니다.
4. 현재 풀 테이블에서 삭제할 스레드 풀 이름을 선택합니다.
5. 삭제를 누릅니다.
6. 서버를 다시 시작합니다.

해당 asadmin 명령: delete-threadpool

로깅 구성

이 장에서는 로깅을 구성하고 서버 로그를 조회하기 위해 관리 콘솔을 사용하는 방법에 대해 간단하게 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

로깅 정보

로그 레코드

Application Server에서는 JSR 047에 지정된 Java 2 플랫폼 로깅 API를 사용합니다. Application Server 로깅 메시지는 대개 *domain_root_dir*/*domain_dir*/`logs/server.log`에 있는 서버 로그에 기록됩니다.

domain_root_dir/*domain_dir*/`logs/` 디렉토리에는 서버 로그외에 다른 두 종류의 로그가 있습니다. `access` 하위 디렉토리에는 HTTP 서비스 액세스 로그가 있고, `tx` 하위 디렉토리에는 트랜잭션 서비스 로그가 있습니다. 이러한 로그에 대한 자세한 내용은 307페이지의 "HTTP 서비스 액세스 로그 구성" 및 297페이지의 "트랜잭션 구성"을 참조하십시오.

Application Server의 구성 요소가 로깅 출력을 생성합니다. 응용 프로그램 구성 요소에서도 로깅 출력을 생성할 수 있습니다.

응용 프로그램 구성 요소에서 Apache Commons Logging Library를 사용하여 메시지를 로그할 수 있습니다. 그러나 더 좋은 로그 구성을 위해서는 플랫폼 표준 JSR 047 API를 권장합니다.

로그 레코드는 다음과 같은 일관된 형식을 따릅니다.

[#|*yyyy-mm-ddThh:mm:ss.SSS-Z*|*Log Level*|*ProductName_Version*|*LoggerName*|*Key Value Pairs*|*Message*|#]

예를 들면 다음과 같습니다.

```
[#|2004-10-21T13:25:53.852-0400|INFO|sun-appserver-ee8.1|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

이 예에서

- [# 및 #]은 레코드의 시작과 끝을 표시합니다.
- 세로 막대(|)는 레코드 필드를 구분합니다.
- 2004-10-21T13:25:53.852-0400은 날짜와 시간을 지정합니다.
- *Log Level*은 INFO입니다. 이 수준은 SEVERE, WARNING, INFO, CONFIG, FINE, FINER 및 FINEST 값 중 하나일 수 있습니다.
- *ProductName_Version*은 sun-appserver-ee8.1입니다.
- *LoggerName*은 로그 모듈의 소스를 식별하는 계층적인 로거 이름 공간입니다. 이 경우에는 javax.enterprise.system.core입니다.
- *Key Value Pairs*는 키 이름과 값입니다. 대개는 _ThreadID=14; 같은 스레드 아이디입니다.
- *Message*는 로그 메시지의 텍스트입니다. 모든 Application Server SEVERE 및 WARNING 메시지와 많은 INFO 메시지의 경우 모듈 코드와 숫자 값으로 구성된 메시지 아이디로 시작됩니다. 이 경우에는 CORE5004입니다.

앞으로의 릴리스에서 로그 레코드 형식이 변경되거나 향상될 수 있습니다.

로거 이름 공간 계층

Application Server에서는 각 모듈에 대한 로거를 제공합니다. 다음 표는 모듈 이름과 각 로거에 대한 이름 공간이 관리 콘솔의 로그 수준 페이지에 표시되는 알파벳 순서로 나열합니다. 335페이지의 "로그 수준 구성"을 참조하십시오. 표의 마지막 세 모듈은 로그 수준 페이지에 표시되지 않습니다.

표 20-1 Application Server 로거 이름 공간

모듈 이름	이름 공간
관리	javax.enterprise.system.tools.admin

표 20-1 Application Server 로거 이름 공간 *(계속)*

모듈 이름	이름 공간
클래스 로더	javax.enterprise.system.core.classloading
CMP	javax.enterprise.system.container.cmp
구성	javax.enterprise.system.core.config
커넥터	javax.enterprise.resource.resourceadapter
CORBA	javax.enterprise.resource.corba
배포	javax.enterprise.system.tools.deployment
EJB 컨테이너	javax.enterprise.system.container.ejb
JavaMail	javax.enterprise.resource.javamail
JAXR	javax.enterprise.resource.webservices.registry
JAX-RPC	javax.enterprise.resource.webservices.rpc
JDO	javax.enterprise.resource.jdo
JMS	javax.enterprise.resource.jms
JTA	javax.enterprise.resource.jta
JTS	javax.enterprise.system.core.transaction
MDB 컨테이너	javax.enterprise.system.container.ejb.mdb
이름 지정	javax.enterprise.system.core.naming
노드 에이전트(Enterprise Edition에만 해당)	javax.ee.enterprise.system.nodeagent
루트	javax.enterprise
SAAJ	javax.enterprise.resource.webservices.saaj
보안	javax.enterprise.system.core.security
서버	javax.enterprise.system
동기화(Enterprise Edition에만 해당)	javax.ee.enterprise.system.tools.synchronization
Util	javax.enterprise.system.util
검증자	javax.enterprise.system.tools.verifier
웹 컨테이너	javax.enterprise.system.container.web
코어	javax.enterprise.system.core
시스템 출력(System.out.println)	javax.enterprise.system.stream.out
시스템 오류(System.err.println)	javax.enterprise.system.stream.err

로깅을 위한 관리 콘솔 작업

일반 로깅 설정 구성

1. 트리 구성 요소에서 노드 에이전트나 구성 노드를 확장합니다.
2. 노드 에이전트를 선택하거나 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. default-config의 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 노드 에이전트의 경우 로거 설정 탭을 선택합니다. 구성의 경우 로거 설정 노드를 선택합니다.
4. 로깅 설정 페이지에서 다음 필드를 사용하여 로깅을 사용자 정의합니다.
   - **로그 파일**: 서버 로그 파일에 대한 대체 이름이나 위치를 지정하려면 텍스트 필드에 새로운 경로 이름을 입력합니다. 기본 위치는 *domain_root_dir*/*domain_dir*/logs/server.log입니다.
   - **경보**: JMX 프레임워크를 통해 SEVERE 및 WARNING 메시지를 라우팅하려면 사용 가능 확인란을 선택합니다.
   - **시스템 로그에 쓰기**: Solaris 및 Linux 시스템의 경우 로깅 출력을 서버 로그 외에 syslog 기능에 전송하려면 사용 가능 확인란을 선택합니다.
   - **로그 처리기**: server.log 또는 syslog 이외의 다른 대상에 로그를 전송할 수 있도록 사용자 정의 로그 처리기를 플러그인할 수 있습니다. 사용자 정의 처리기에서 클래스 java.util.logging.Handler(JSR 047 호환 API)를 확장해야 합니다. 로그 처리기 필드에서 처리기의 절대 클래스 이름을 입력합니다. 서버 시작 중에 처리기가 설치되도록 Application Server 클래스 경로에 처리기 클래스를 포함합니다. 사용자 정의 처리기의 로그 레코드 형식은 331페이지의 "로그 레코드"에서 설명한 것과 같습니다.

- **로그 필터**: server.log, syslog 같은 대상이나 사용자 정의 로그 처리기에서 정의한 대상으로 전송한 로그 레코드를 필터링할 수 있도록 사용자 정의 로그 필터를 플러그인할 수 있습니다. 사용자 정의 필터에 java.util.logging.Filter 인터페이스를 구현해야 합니다. 로그 필터 필드에서 필터의 절대 클래스 이름을 입력합니다. 서버 시작 중에 필터가 설치되도록 Application Server 클래스 경로에 필터 클래스도 포함합니다.
- **파일 회전 제한**: 서버 로그가 지정한 크기(바이트)에 도달하면 server.log라고 하는 새로운 빈 파일을 만들고 이전 파일의 이름을 server.log_*date*로 바꿉니다. 여기서 *date*는 파일이 회전된 날짜와 시간입니다. 기본값은 2MB입니다. 제한의 최소값은 500KB입니다. 더 낮은 값을 지정해도 500KB에 도달해야 파일이 회전됩니다. 로그 파일 회전을 해제하려면 값을 0으로 설정합니다.
- **파일 회전 시간 제한**: 지정한 시간(분)이 지나면 서버 로그를 회전합니다. 기본값은 0으로 파일 회전 제한 필드에 지정한 크기에 도달하면 파일이 회전됨을 의미합니다. *1분 이상을 지정할 경우 시간 제한이 크기 제한보다 우선합니다.*

5. 저장을 눌러 변경 사항을 저장합니다. 로그 파일 보기를 눌러 서버 로그를 확인합니다.

로그 수준 구성

1. 트리 구성 요소에서 노드 에이전트나 구성 노드를 확장합니다.
2. 노드 에이전트를 선택하거나 구성할 인스턴스를 선택합니다.
   a. 특정 인스턴스를 구성하려면 그 인스턴스의 구성 노드를 선택합니다. 예를 들어 기본 인스턴스 server는 server-config 노드를 선택합니다.
   b. default-config의 사본을 사용하는 앞으로의 인스턴스에 대한 기본 설정을 구성하려면 default-config 노드를 선택합니다.
3. 노드 에이전트의 경우 로그 수준 탭을 선택합니다. 구성의 경우 로거 설정 노드를 선택한 다음 로그 수준 탭을 선택합니다.
4. 모듈 로그 수준 페이지에서 로그 수준을 변경할 모듈 반대쪽에 있는 드롭다운 목록에서 새로운 값을 선택합니다. 기본값은 INFO로, 이보다 더 높은 수준의 메시지(WARNING, SEVERE)가 로그에 표시됨을 의미합니다. 최고값에서 최저값까지 나열된 다음 값 중에서 선택합니다.
   - SEVERE
   - WARNING

- INFO
- CONFIG
- FINE
- FINER
- FINEST고
- OFF

5. 추가 등록 정보 영역을 사용하여 응용 프로그램 로거에 대한 로그 수준을 구성합니다. 등록 정보 이름은 로거 이름 공간이고, 값은 여덟 가지 가능한 수준 중 하나입니다. 예를 들어, 등록 정보 이름은 samples.logging.simple.servlet일 수 있고, 값은 FINE일 수 있습니다.

   이 영역을 사용하여 하위 모듈(예: CORBA 모듈의 전송 하위 모듈)에 대한 로그 수준도 변경합니다.

   javax.enterprise.resource.corba.*ORBId*.transport

6. 저장을 눌러 변경 사항을 저장하거나, 기본값 로드를 눌러 기본값을 복구합니다.

System.out.println에 대한 호출은 로거 이름 javax.enterprise.system.stream.out을 사용하여 INFO 수준에서 로깅됩니다. System.err.println에 대한 호출은 로거 이름 javax.enterprise.system.stream.err을 사용하여 WARNING 수준에서 로깅됩니다. 이 소스의 로그를 해제하려면 추가 등록 정보 영역에서 OFF 값과 함께 로거 이름을 지정합니다.

로그 수준 설정에 대한 변경 사항은 즉시 적용됩니다. 이 변경 사항은 서버를 다시 시작할 때 사용할 수 있도록 domain.xml 파일에도 저장됩니다.

서버 로그 보기

1. 트리 구성 요소에서 확인할 로그의 서버 인스턴스에 대한 노드를 확장합니다.
2. 일반 정보 페이지에서 로그 파일 보기를 누릅니다.

검색 기준 영역을 사용하여 로그 뷰어를 사용자 정의하거나 필터링합니다. 다음과 같은 기본 필드를 사용합니다.

- **인스턴스 이름**: 드롭다운 목록에서 인스턴스 이름을 선택하여 해당 서버 인스턴스에 대한 로그를 확인합니다. 기본값은 현재 서버 인스턴스입니다.

- **로그 파일**: 드롭다운 목록에서 로그 파일 이름을 선택하여 그 로그 내용을 확인합니다. 기본값은 `server.log`입니다.
- **타임스탬프**: 가장 최근 메시지를 보려면 가장 최근(기본값)을 선택합니다. 특정 기간의 메시지만 보려면 특정 범위를 선택하고 표시되는 시작 및 끝 필드에 날짜와 시간 값을 입력합니다. 시간 값의 경우 구문은 다음 형식을 따라야 합니다. 여기에서 *sss*는 밀리초의 약자입니다.

  hh:*mm*:*ss*.*SSS*

  예를 들면 다음과 같습니다.

  `17:10:00.000`

  시작 값이 끝 값보다 이후일 경우 오류 메시지가 표시됩니다.
- **로그 수준**: 로그 수준별로 메시지를 필터링하려면 드롭다운 목록에서 로그 수준을 선택합니다. 기본적으로 선택한 로그 수준과 더 심각한 수준에서 서버 로그에 표시되는 모든 메시지가 표시됩니다. 선택한 수준의 메시지만 표시하려면 "더 심각한 메시지를 포함하지 않습니다" 확인란을 선택합니다.

  확인할 메시지가 서버 로그에 표시되게 하려면 먼저 로그 수준 페이지에서 적절한 로그 수준을 설정합니다. 335페이지의 "로그 수준 구성"을 참조하십시오.

  로그 수준을 기준으로 로그 메시지를 필터링하도록 선택한 경우 지정한 필터 기준에 맞는 메시지만 표시됩니다. 그러나 이 필터링은 어떤 메시지를 서버 로그에 로깅할지에는 영향을 미치지 않습니다.

로깅 설정 페이지와 로그 수준 페이지에서 지정한 설정과 함께 서버 로그의 가장 최근 40개 항목이 표시됩니다.

타임스탬프 헤더 옆에 있는 삼각형을 눌러 가장 최근 항목이 마지막으로 표시되도록 메시지를 정렬합니다.

메시지를 형식이 지정된 모양으로 보려면 표시된 링크를 누릅니다.

`(details)`

로그 항목 세부 정보 창이 형식 지정된 버전의 메시지와 함께 표시됩니다.

항목 목록 끝에서 버튼을 눌러 로그 파일의 이전 또는 이후 항목을 확인합니다.

검색 기준 영역에서 고급 검색을 눌러 로그 뷰어를 더 구체화합니다. 다음과 같은 고급 옵션 필드를 사용합니다.

- **로거**: 모듈별로 필터링하려면 드롭다운 목록에서 하나 이상의 이름 공간을 선택합니다. Shift 또는 Ctrl 키를 누른 상태에서 선택하면 여러 이름 공간을 선택할 수 있습니다.

  더 높은 수준의 이름 공간을 선택하면 그 아래 있는 모든 이름 공간이 선택됩니다. 예를 들어, `javax.enterprise.system`을 선택하면 그 이름 공간 아래에 있는 모든 모듈(예: `javax.enterprise.system.core`, `javax.enterprise.system.tools.admin` 등)에 대한 로거도 선택됩니다.

- **사용자 정의 로거**: 특정 응용 프로그램에 관련된 로거의 메시지를 보려면 텍스트 필드에서 로거 이름을 한 줄에 하나씩 입력합니다. 응용 프로그램에 여러 모듈이 있을 경우 모두 표시하거나 선택해서 표시할 수 있습니다. 예를 들어, 응용 프로그램에 다음과 같은 이름의 로거가 있다고 가정합니다.

  ```
  com.mycompany.myapp.module1
  com.mycompany.myapp.module2
  com.mycompany.myapp.module3
  ```

  응용 프로그램의 모든 모듈의 메시지를 나타내려면 `com.mycompany.myapp`를 입력합니다. `module2`의 메시지만 확인하려면 `com.mycompany.myapp.module2`를 입력합니다.

  하나 이상의 사용자 정의 로거를 지정하면 로거 영역에서 로거를 명시적으로 지정한 경우에만 Application Server 모듈의 메시지가 표시됩니다.

- **이름-값 쌍**: 특정한 스레드의 출력을 보려면 텍스트 필드에서 해당 스레드에 대한 키 이름과 값을 입력합니다. 키 이름은 `_ThreadID`입니다. 예를 들면 다음과 같습니다.

  ```
  _ThreadID=13
  ```

  `com.mycompany.myapp.module2`가 여러 스레드에서 실행되는 것으로 가정합니다. 단일 스레드의 출력만 표시하도록 로거 뷰어를 구체화하려면 사용자 정의 로거 필드에서 모듈의 로거를 지정한 다음 이 필드에서 스레드 아이디를 지정합니다.

- **표시**: 한 번에 40개 이상(기본값)의 메시지를 보려면 드롭다운 목록에서 사용 가능한 값(100, 250 또는 1000) 중 하나를 선택합니다.

  스택 추적을 보려면 "과도하게 긴 메시지 제한합니다" 확인란을 선택 해제합니다. 기본적으로 스택 추적은 뷰어에 표시되지 않습니다. 스택 추적을 보려면 메시지의 `(details)` 링크를 누릅니다.

고급 옵션 영역을 숨기려면 기본 검색을 누릅니다.

구성 요소 및 서비스 모니터링

이 장에서는 Application Server 관리 콘솔을 사용하여 구성 요소를 모니터링하는 것에 대한 정보를 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- 모니터링 정보
- 모니터링 활성화 또는 비활성화를 위한 관리 콘솔 작업
- 모니터링 데이터 보기를 위한 관리 콘솔 작업

모니터링 정보

- Application Server에서 모니터링
- 모니터링 개요
- 모니터링 가능한 객체의 트리 구조 정보
- 모니터된 구성 요소 및 서비스에 대한 통계 정보

Application Server에서 모니터링

Sun Java System Application Server Enterprise Edition 8.1 2005Q1의 서버 인스턴스에 배포된 다양한 구성 요소와 서비스의 런타임 상태를 확인하려면 모니터링을 사용합니다. 런타임 구성 요소와 프로세스의 상태에 대한 정보를 사용하면 성능 조정을 목적으로 성능 병목 현상을 확인하고, 용량 계획을 지원하며, 실패를 예상하고, 실패 시 근본적인 원인 분석을 수행하고, 모든 항목이 예상대로 기능하도록 할 수 있습니다.

모니터링을 설정하면 오버헤드가 증가하여 성능이 저하됩니다.

모니터링 개요

Application Server를 모니터링하려면 다음 단계를 수행합니다.

1. 관리 콘솔이나 asadmin 도구를 사용하여 특정 서비스와 구성 요소의 모니터링을 활성화합니다.

   이 단계에 대한 자세한 내용은 "모니터링 활성화 또는 비활성화를 위한 관리 콘솔 작업"을 참조하십시오.

2. 관리 콘솔이나 asadmin 도구를 사용하여 지정한 서비스나 구성 요소에 대한 모니터링 데이터를 확인합니다.

   이 단계에 대한 자세한 내용은 "모니터링 데이터 보기를 위한 관리 콘솔 작업"을 참조하십시오.

모니터링 가능한 객체의 트리 구조 정보

Application Server에서는 트리 구조를 사용하여 모니터링 가능한 객체를 추적합니다. 모니터링 객체의 트리가 동적이기 때문에 인스턴스에 구성 요소가 추가, 업데이트 또는 제거되면 트리도 변경됩니다. 트리의 루트 객체는 서버 인스턴스 이름(예: server)입니다. Platform Edition에서는 한 서버 인스턴스만 허용됩니다.

다음 명령은 트리의 최상위 수준을 표시합니다.

```
asadmin> list --monitor server
server.applications
server.http-service
server.connector-service
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

다음 절에서는 이 하위 트리를 설명합니다.

- 응용 프로그램 트리
- HTTP 서비스 트리
- 커넥터 서비스 트리
- 커넥터 서비스 트리
- JMS 서비스 트리

- ORB 트리
- 스레드 풀 트리

응용 프로그램 트리

다음 계통도에서는 엔터프라이즈 응용 프로그램의 다양한 구성 요소에 대한 상위 노드와 자식 노드를 표시합니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다. 자세한 내용은 "EJB 컨테이너 통계" 및 "웹 컨테이너 통계"를 참조하십시오.

그림 21-1 응용 프로그램 노드 트리 구조

```
applications
   |--- application1
   |     |--- ejb-module-1
   |     |         |--- ejb1 *
   |     |                   |--- cache (for entity/sfsb) *
   |     |                   |--- pool (for s1sb/mdb/entity) *
   |     |                   |--- methods
   |     |                        |---method1 *
   |     |                        |---method2 *
   |     |                   |--- stateful-session-store (for sfsb)*
   |     |                   |--- timers (for s1sb/entity/mdb) *
   |     |--- web-module-1
   |     |         |--- virtual-server-1 *
   |     |                         |---servlet1  *
   |     |                         |---servlet2  *
   |--- standalone-web-module-1
   |     |         |----- virtual-server-2 *
   |     |                         |---servlet3 *
   |     |                         |---servlet4 *
   |     |         |----- virtual-server-3 *
   |     |                        |---servlet3 *(same servlet on different vs)
   |     |                        |---servlet5 *
   |--- standalone-ejb-module-1
   |     |         |--- ejb2 *
   |     |                   |--- cache (for entity/sfsb) *
   |     |                   |--- pool (for s1sb/mdb/entity) *
   |     |                   |--- methods
   |     |                        |--- method1 *
   |     |                        |--- method2 *
   |--- application2
```

HTTP 서비스 트리

다음 계통도에는 HTTP 서비스 노드가 표시됩니다. 모니터링 정보를 사용할 수 있는 노드에는 별표(*)가 표시됩니다. "HTTP 서비스 트리"를 참조하십시오.

그림 21-2 HTTP 서비스 계통도(PE 버전)

```
http-service
        |--- virtual-server-1
           |--- http-listener-1 *
           |--- http-listener-2 *
        |--- virtual-server-2
           |--- http-listener-1 *
           |--- http-listener-2 *
```

그림 21-3 HTTP 서비스 계통도(EE 버전)

```
http-service *
            |---connection-queue *
            |---dns *
            |---file-cache *
            |---keep-alive *
            |---pwc-thread-pool *
            |---virtual-server-1*
            |        |--- request *
            |---virtual-server-2*
            |        |--- request *
```

자원 트리

자원 노드에는 풀(예: JDBC 연결 풀이나 커넥터 연결 풀)에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 다양한 자원 구성 요소에 대한 상위 노드와 자식 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다. "JDBC 연결 풀 통계"와 "JMS/커넥터 서비스 통계"를 참조하십시오.

그림 21-4 자원 계통도

```
resources
       |---connection-pool1(either connector-connection-pool or jdbc)*
       |---connection-pool2(either connector-connection-pool or jdbc)*
```

커넥터 서비스 트리

커넥터 서비스 노드에는 풀(예: 커넥터 연결 풀)에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 다양한 커넥터 서비스 구성 요소에 대한 상위 노드와 자식 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다. "JMS/커넥터 서비스 통계"를 참조하십시오.

그림 21-5 커넥터 서비스 계통도

```
connector-service
       |--- resource-adapter-1
       |       |-- connection-pools
       |       |       |-- pool-1 (All pool stats for this pool)
       |       |-- work-management (All work mgmt stats for this RA)
```

JMS 서비스 트리

JMS 서비스 노드에는 풀(예: 커넥터 연결 풀)에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 다양한 JMS 서비스 구성 요소에 대한 상위 노드와 자식 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다.

그림 21-6 JMS 서비스 계통도

```
jms-service
       |-- connection-factories [AKA conn. pools in the RA world]
       |       |-- connection-factory-1 (All CF stats for this CF)
       |-- work-management (All work mgmt stats for the MQ-RA)
```

ORB 트리

ORB 노드에는 연결 관리자에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 ORB 구성 요소에 대한 상위 노드와 자식 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다. "ORB의 연결 관리자용 통계"를 참조하십시오.

그림 21-7 ORB 계통도

```
orb
        |--- connection-managers
        |       |--- connection-manager-1 *
        |       |--- connection-manager-1 *
```

스레드 풀 트리

스레드 풀 노드에는 연결 관리자에 대한 모니터링 가능한 속성이 들어 있습니다. 다음 계통도에서는 ORB 구성 요소에 대한 상위 노드와 자식 노드를 보여줍니다. 모니터링 통계를 사용할 수 있는 노드는 별표(*)가 표시됩니다. "스레드 풀 통계"를 참조하십시오.

그림 21-8 스레드 풀 계통도

```
thread-pools
        |   |--- thread-pool-1 *
        |   |--- thread-pool-2 *
```

모니터된 구성 요소 및 서비스에 대한 통계 정보

이 절에서는 사용 가능한 모니터링 통계에 대해 설명합니다.

- EJB 컨테이너 통계
- 웹 컨테이너 통계
- HTTP 서비스 통계
- JDBC 연결 풀 통계
- JMS/커넥터 서비스 통계
- ORB의 연결 관리자용 통계
- 스레드 풀 통계
- 트랜잭션 서비스 통계
- Java Virtual Machine(JVM) 통계
    - J2SE 5.0의 JVM 통계
- PWC(Production Web Container) 통계

EJB 컨테이너 통계

표 21-1에서는 EJB 통계를 설명합니다.

표 21-1 EJB 통계

속성 이름	데이터 유형	설명
createcount	Count Statistic	EJB의 create 메소드를 호출한 횟수입니다.
removecount	Count Statistic	EJB의 remove 메소드를 호출한 횟수입니다.
pooledcount	Range Statistic	풀링된 상태의 Entity Bean 개수입니다.
readycount	Range Statistic	준비 상태의 Entity Bean 개수입니다.
messagecount	Count Statistic	Message-Driven Bean에 대해 수신한 메시지 개수입니다.
methodreadycount	Range Statistic	MethodReady 상태의 Stateful 또는 Stateless Session Bean 개수입니다.
passivecount	Range Statistic	Passive 상태의 Stateful Session Bean 개수입니다.

표 21-2에는 EJB 메소드 호출에 사용할 수 있는 통계가 나열되어 있습니다.

표 21-2 EJB 메소드 통계

속성 이름	데이터 유형	설명
methodstatistic	Time Statistic	작업을 호출한 횟수와 호출에 소요된 전체 시간 등입니다.
totalnumerrors	Count Statistic	메소드 실행 결과 예외가 발생한 횟수입니다. EJB 컨테이너에 대해 모니터링을 사용하는 경우 Stateless 및 Stateful Session Bean에 대해 이 통계를 수집합니다.
totalnumsuccess	Count Statistic	메소드가 성공적으로 실행된 횟수입니다. EJB 컨테이너에 대해 모니터링을 사용하는 경우 Stateless 및 Stateful Session Bean에 대해 이 통계를 수집합니다.
executiontime	Count Statistic	작업을 실행하기 위한 마지막 성공/실패 시도에 대해 메소드를 실행하는 데 걸린 시간입니다. EJB 컨테이너에 대해 모니터링을 사용하는 경우 Stateless 및 Stateful Session Bean에 대해 이 통계를 수집합니다.

표 21-3에는 EJB 세션 저장소에 대한 통계가 나열되어 있습니다.

표 21-3 EJB 세션 저장소 통계

속성 이름	데이터 유형	설명
currentSize	Range Statistic	현재 저장소에 있는 비활성화된 세션이나 검사점이 지정된 세션 수입니다.
activationCount	Count Statistic	저장소에서 활성화된 세션 수입니다.
activationSuccessCount	Count Statistic	저장소에서 성공적으로 활성화된 세션 수입니다.
activationErrorCount	Count Statistic	작업을 실행하기 위한 마지막 성공/실패 시도에 대해 메소드를 실행하는 데 걸린 시간입니다. EJB 컨테이너에 대해 모니터링을 사용하는 경우 Stateless 및 Stateful Session Bean에 대해 이 통계를 수집합니다.
passivationCount	Count Statistic	이 저장소를 사용하여 비활성화된 세션 수입니다.
passivationSuccessCount	Count Statistic	이 저장소를 사용하여 비활성화된 세션 수입니다.
passivationErrorCount	Count Statistic	이 저장소를 사용하여 비활성화할 수 없는 세션 수입니다.
expiredSessionCount	Count Statistic	이 저장소에서 제거한 만료된 세션 수입니다.
passivatedBeanSize	Count Statistic	전체, 최소 및 최대를 포함하여 이 저장소에서 비활성화한 전체 바이트 수입니다.
passivationTime	Count Statistic	전체, 최소 및 최대를 포함하여 Bean을 저장소로 비활성화하는 데 걸린 시간입니다.
checkpointCount(EE only)	Count Statistic	이 저장소를 사용하여 검사점이 지정된 세션 수입니다.
checkpointSuccessCount(EE only)	Count Statistic	검사점이 지정된 세션 수입니다.
checkpointErrorCount(EE only)	Count Statistic	검사점을 지정할 수 없는 세션 수입니다.
checkpointedBeanSize(EE only)	Value Statistic	저장소에서 검사점을 지정한 Bean의 총 개수입니다.
checkpointTime(EE only)	Time Statistic	Bean을 저장소로 검사점 지정하는 데 걸린 시간입니다.

표 21-4에는 EJB 풀에 사용할 수 있는 통계가 나열되어 있습니다.

표 21-4 EJB 풀 통계

속성 이름	데이터 유형	설명
numbeansinpool	Bounded Range Statistic	풀 변경 방법에 대한 정보를 제공하는 연관된 풀의 EJB 수입니다.
numthreadswaiting	Bounded Range Statistic	사용 가능한 Bean을 기다리는 스레드 수로 요청이 정체될 수 있음을 나타냅니다.
totalbeanscreated	Count Statistic	데이터 수집이 시작된 후 연관된 풀에서 만들어진 Bean 수입니다.
totalbeansdestroyed	Count Statistic	데이터 수집이 시작된 후 연관된 풀에서 삭제된 Bean 수입니다.
jmsmaxmessagesload	Count Statistic	Message-driven Bean을 위해 JMS 세션에 한 번에 로드하는 최대 메시지 수입니다. 기본값은 1이며, Message-driven Bean의 풀에만 적용됩니다.

표 21-5에는 EJB 캐시에 사용 가능한 통계가 나열되어 있습니다.

표 21-5 EJB 캐시 통계

속성 이름	데이터 유형	설명
cachemisses	Bounded Range Statistic	사용자 요청 시 캐시에서 Bean을 찾지 못한 횟수입니다.
cachehits	Bounded Range Statistic	사용자 요청 시 캐시에서 항목을 찾은 횟수입니다.
numbeansincache	Bounded Range Statistic	캐시에 있는 Bean의 수입니다. 이 값은 캐시의 현재 크기입니다.
numpassivations	Count Statistic	비활성화 처리 횟수입니다. Stateful Session Bean에만 적용됩니다.
numpassivationerrors	Count Statistic	비활성화 처리를 수행하는 동안 발생한 오류 횟수입니다. Stateful Session Bean에만 적용됩니다.
numexpiredsessionsremoved	Count Statistic	정리 스레드로 제거된 만료된 세션의 수입니다. Stateful Session Bean에만 적용됩니다.
numpassivationsuccess	Count Statistic	비활성화 처리가 성공적으로 완료된 횟수입니다. Stateful Session Bean에만 적용됩니다.

표 21-6에는 타이머에 사용할 수 있는 통계가 나열되어 있습니다.

표 21-6 타이머 통계

통계	데이터 유형	설명
numtimerscreated	CountStatistic	시스템에서 만들어진 타이머 수입니다.
numtimersdelivered	CountStatistic	시스템에서 전달한 타이머 수입니다.
numtimersremoved	CountStatistic	시스템에서 제거한 타이머 수입니다.

웹 컨테이너 통계

웹 컨테이너는 그림 21-1에 표시된 것처럼 객체 트리에 해당합니다. 모든 개별 웹 응용 프로그램에 대한 웹 컨테이너 통계가 표시됩니다. 표 21-7에는 서블릿용 웹 컨테이너에 사용 가능한 통계가 표시되고 표 21-8에는 웹 모듈에 사용 가능한 통계가 표시됩니다.

표 21-7 웹 컨테이너(서블릿) 통계

통계	단위	데이터 유형	설명
errorcount	수	CountStatistic	이유 코드가 400보다 크거나 같은 경우의 누적 수입니다.
maxtime	밀리초	CountStatistic	웹 컨테이너가 요청을 기다리는 최대 시간입니다.
processingtime	밀리초	CountStatistic	각 요청을 처리하는 데 필요한 시간의 누적 값입니다. 처리 시간은 요청 처리 시간을 요청 수로 나눈 평균 값입니다.
requestcount	수	CountStatistic	지금까지 처리한 요청의 총 수입니다.

표 21-8에는 웹 모듈에 사용 가능한 통계가 표시됩니다.

표 21-8 웹 컨테이너(웹 모듈) 통계

통계	데이터 유형	설명
jspcount	CountStatistic	웹 모듈에 로드된 JSP 페이지 수입니다.
jspreloadcount	CountStatistic	웹 모듈에 다시 로드된 JSP 페이지 수입니다.

표 21-8 웹 컨테이너(웹 모듈) 통계 *(계속)*

통계	데이터 유형	설명
sessionstotal	CountStatistic	웹 모듈에 대해 만들어진 총 세션 수입니다.
activesessionscurrent	CountStatistic	웹 모듈에 대해 현재 활성화된 세션 수입니다.
activesessionshigh	CountStatistic	웹 모듈에 대해 동시에 활성화된 최대 세션 수입니다.
rejectedsessionstotal	CountStatistic	웹 모듈에 대해 거부된 총 세션 수입니다. 허용된 최대 세션 수가 활성화되었기 때문에 만들어지지 않은 세션 수입니다.
expiredsessionstotal	CountStatistic	웹 모듈에 대해 만료된 총 세션 수입니다.
sessionsize(EE only)	AverageRange Statistic	웹 모듈에 대한 세션 크기입니다. 값은 높음, 낮음 또는 평균이거나 일련화된 세션의 경우 바이트입니다.
containerlatency(EE only)	AverageRange Statistic	전체 대기 시간 요청의 웹 컨테이너 부분에 대한 대기 시간입니다. 값은 높음, 낮음 또는 평균입니다.
sessionpersisttime(EE only)	AverageRange Statistic	웹 모듈의 백 엔드 저장소에 HTTP 세션 상태를 지속시키는 데 걸린 시간(ms, 낮음, 높음 또는 평균)입니다.
cachedsessionscurrent(EE only)	CountStatistic	웹 모듈의 메모리에 현재 캐시된 세션 수입니다.
passivatedsessionscurrent(EE only)	CountStatistic	웹 모듈에 대해 현재 비활성화된 세션 수입니다.

HTTP 서비스 통계

표 21-9에는 HTTP 서비스에 사용 가능한 통계가 표시됩니다. 이 통계는 Platform Edition에만 적용됩니다. Enterprise Edition의 HTTP 서비스에 대한 통계는 표 21-32를 참조하십시오.

표 21-9 HTTP 서비스 통계(Platform Edition에만 적용)

통계	단위	데이터 유형	설명
bytesreceived	바이트	Count Statistic	각 요청 프로세서에서 수신한 누적 바이트 값입니다.
bytessent	바이트	Count Statistic	각 요청 프로세서에서 전송한 누적 바이트 값입니다.
currentthreadcount	수	Count Statistic	Listener 스레드 풀에서 현재 처리 중인 스레드 수입니다.
currentthreadsbusy	수	Count Statistic	요청을 처리하는 Listener 스레드 풀에서 현재 사용 중인 요청 처리 스레드 수입니다.
errorcount	수	Count Statistic	오류 누적 수는 응답 코드가 400보다 크거나 같은 경우의 수를 나타냅니다.
maxsparethreads	수	Count Statistic	존재할 수 있는 사용하지 않은 응답 처리 스레드의 최대 수입니다.
minsparethreads	수	Count Statistic	존재할 수 있는 사용하지 않은 응답 처리 스레드의 최소 수입니다.
maxthreads	수	Count Statistic	Listener에서 만든 요청 처리 스레드의 최대 수입니다.
maxtime	밀리초	Count Statistic	처리 스레드에 대한 최대 시간입니다.
processing-time	밀리초	Count Statistic	각 요청을 처리하는 데 걸린 시간의 누적 값입니다. 처리 시간은 요청 처리 시간을 요청 수로 나눈 평균 값입니다.
request-count	수	Count Statistic	지금까지 처리한 총 요청 수입니다.

JDBC 연결 풀 통계

성능을 측정하고 런타임시 자원 사용을 수집하기 위해 JDBC 자원을 모니터합니다. JDBC 연결을 만들면 부담이 크고 응용 프로그램의 성능 병목 상태를 자주 일으키기 때문에 JDBC 연결 풀에서 새로운 연결을 해제 및 작성하는 방법과 많은 스레드가 특정 풀에서 연결을 검색하기 위해 대기하는 방법을 모니터하는 것이 중요합니다.

표 21-10에는 JDBC 연결 풀에 사용 가능한 통계가 표시됩니다.

표 21-10 JDBC 연결 풀 통계

통계	단위	데이터 유형	설명
numconnfailedvalidation	수	Count Statistic	시작 시간 이후 마지막 샘플 시간까지 연결 풀에서 검증에 실패한 총 연결 수입니다.
numconnused	수	Range Statistic	연결 사용 통계를 제공합니다. 현재 사용하고 있는 총 연결 수 외에 사용한 최대 연결 수(고수위 표시)에 대한 정보도 제공합니다.
numconnfree	범위 통계	Count Statistic	마지막 샘플링 시에 풀에서 사용 가능한 총 연결 수입니다.
numconntimedout	개수 통계	Bounded Range Statistic	시작 시간과 마지막 샘플 시간 사이에 시간 초과된 풀의 총 연결 수입니다.
averageconnwaittime	수	Count Statistic	커넥터 연결 풀에 대한 연결 요청의 평균 연결 대기 시간을 나타냅니다.
waitqueuelength	수	Count Statistic	대기열에서 처리를 기다리는 연결 요청 수입니다.
connectionrequestwaittime		Range Statistic	연결 요청의 가장 긴 대기 시간과 가장 짧은 대기 시간입니다. 현재 값은 풀에서 처리한 마지막 요청의 대기 시간을 나타냅니다.
numconncreated	밀리초	CountStatistic	마지막 재설정 후 만들어진 물리적 연결 수입니다.

표 21-10 JDBC 연결 풀 통계 *(계속)*

통계	단위	데이터 유형	설명
numconndestroyed	수	Count Statistic	마지막 재설정 후 삭제된 물리적 연결 수입니다.
numconnacquired	수	Count Statistic	풀에서 얻은 논리적 연결 수입니다.
numconnreleased	수	Count Statistic	풀로 해제된 논리적 연결 수입니다.

JMS/커넥터 서비스 통계

표 21-11에는 커넥터 연결 풀에 사용 가능한 통계가 표시됩니다. 표 21-12에는 Connector Work Management에 대한 통계가 표시됩니다.

표 21-11 커넥터 연결 풀 통계

통계	단위	데이터 유형	설명
numconnfailed validation	수	Count Statistic	시작 시간 이후 마지막 샘플 시간까지 연결 풀에서 검증에 실패한 총 연결 수입니다.
numconnused	수	Range Statistic	연결 사용 통계를 제공합니다. 현재 사용하고 있는 총 연결 수 외에 사용한 최대 연결 수(고수위 표시)에 대한 정보도 제공합니다.
numconnfree	수	Range Statistic	마지막 샘플링 시에 풀에서 사용 가능한 총 연결 수입니다.
numconntimedout	수	Count Statistic	시작 시간과 마지막 샘플 시간 사이에 시간 초과된 풀의 총 연결 수입니다.
averageconnwaittime	수	Count Statistic	연결 풀에서 처리할 때까지 연결의 평균 대기 시간입니다.
waitqueuelenght	수	Count Statistic	대기열에서 처리를 기다리는 연결 요청 수입니다.

표 21-11 커넥터 연결 풀 통계 *(계속)*

통계	단위	데이터 유형	설명
connectionrequest waittime		Range Statistic	연결 요청의 가장 긴 대기 시간과 가장 짧은 대기 시간입니다. 현재 값은 풀에서 처리한 마지막 요청의 대기 시간을 나타냅니다.
numconncreated	밀리초	CountStatistic	마지막 재설정 후 만들어진 물리적 연결 수입니다.
numconndestroyed	수	Count Statistic	마지막 재설정 후 삭제된 물리적 연결 수입니다.
numconnacquired	수	Count Statistic	풀에서 얻은 논리적 연결 수입니다.
numconnreleased	수	Count Statistic	풀에 해제된 논리적 연결 수입니다.

표 21-12에는 Connector Work Management에 사용 가능한 통계가 나열되어 있습니다.

표 21-12 Connector Work Management 통계

통계	데이터 유형	설명
activeworkcount	Range Statistic	커넥터에서 실행한 작업 객체 수입니다.
waitqueuelength	Range Statistic	실행하기 전에 대기열에서 대기 중인 작업 객체 수입니다.
workrequestwaittime	Range Statistic	실행되기 전 작업 객체의 가장 긴 대기 및 가장 짧은 대기 시간입니다.
submittedworkcount	Count Statistic	커넥터 모듈에서 제출한 작업 객체 수입니다.
rejectedworkcount	Count Statistic	Application Server에서 거부한 작업 객체 수입니다.
completedworkcount	Count Statistic	완료한 작업 객체 수입니다.

ORB의 연결 관리자용 통계

표 21-13에는 ORB의 연결 관리자에 사용 가능한 통계가 나열되어 있습니다.

표 21-13 연결 관리자(ORB) 통계

통계	단위	데이터 유형	설명
connectionsidle	수	CountStatistic	ORB에 대해 유휴인 총 연결 수를 제공합니다.
connectionsinuse	수	CountStatistic	ORB에 대한 사용 중인 총 연결 수를 제공합니다.
totalconnections	수	BoundedRange Statistic	ORB에 대한 총 연결 수입니다.

스레드 풀 통계

표 21-14에는 스레드 풀에 사용 가능한 통계가 나열되어 있습니다.

표 21-14 스레드 풀 통계

통계	단위	데이터 유형	설명
averagetimeinqueue	밀리초	RangeStatistics	처리하기 전에 대기열에서 요청이 대기한 평균 시간(밀리초)입니다.
averageworkcompletion-time	밀리초	RangeStatistics	할당을 완료하는 데 걸린 평균 시간(밀리초)입니다.
currentnumberofthreads	수	BoundedRange Statistic	현재 스레드를 처리하는 요청 수입니다.
numberofavailablethreads	수	CountStatistic	사용 가능한 스레드 수입니다.
numberofbusythreads	수	CountStatistic	사용 중인 스레드 수입니다.
totalworkitemsadded	수	CountStatistic	지금까지 작업 대기열에 추가된 총 작업 항목 수입니다.

트랜잭션 서비스 통계

클라이언트는 트랜잭션 서비스를 통해 트랜잭션 하위 시스템을 중단하여 트랜잭션을 롤백하고 중단 시에 처리 중인 트랜잭션을 확인할 수 있습니다. 표 21-15에는 트랜잭션 서비스에 사용 가능한 통계가 나열되어 있습니다.

표 21-15 트랜잭션 서비스 통계

통계	데이터 유형	설명
`activecount`	CountStatistic	현재 활성화된 트랜잭션 수입니다.
`activeids`	String Statistic	현재 활성화된 트랜잭션의 아이디입니다. 이러한 각 트랜잭션은 트랜잭션 서비스를 고정한 후 롤백할 수 있습니다.
`committedcount`	CountStatistic	완결된 트랜잭션 수입니다.
`rolledbackcount`	CountStatistic	롤백된 트랜잭션 수입니다.
`state`	String Statistic	트랜잭션이 고정되었는지 여부를 나타냅니다.

Java Virtual Machine(JVM) 통계

JVM에는 항상 활성화되어 있는 모니터 가능한 속성이 있습니다. 표 21-16에는 JVM에 사용 가능한 통계가 나열되어 있습니다.

표 21-16 JVM 통계

통계	데이터 유형	설명
`heapsize`	BoundedRange Statistic	상주 메모리 범위는 JVM 메모리 힙 크기의 상한 및 하한입니다.
`uptime`	CountStatistic	JVM이 실행되고 있는 시간입니다.

J2SE 5.0의 JVM 통계

J2SE 버전 5.0 이상에서 실행되도록 Application Server를 구성한 경우 JVM에서 추가 모니터링 정보를 얻을 수 있습니다. 이 추가 정보 표시를 활성화하려면 모니터링 수준을 낮음으로 설정합니다. 시스템의 라이브 스레드에 관련된 정보도 표시하려면 모니터링 수준을 높음으로 설정합니다. J2SE 5.0에서 사용 가능한 추가 모니터링 기능에 대한 자세한 내용은 다음 URL에서 사용할 수 있는 *Monitoring and Management for the Java Platform* 문서를 참조하십시오.

http://java.sun.com/j2se/1.5.0/docs/guide/management/

J2SE 5.0 모니터링 도구는 다음에서 설명하고 있습니다.

http://java.sun.com/j2se/1.5.0/docs/tooldocs/#manage

표 21-17에는 J2SE 5.0에서 JVM의 클래스 로딩에 사용할 수 있는 통계가 나열되어 있습니다.

표 21-17 J2SE 5.0용 JVM 통계 - 클래스 로딩

통계	데이터 유형	설명
loadedclasscount	CountStatistic	현재 JVM에 로드된 클래스 수입니다.
totalloadedclasscount	CountStatistic	JVM에서 실행을 시작한 후 로드된 총 클래스 수입니다.
unloadedclasscount	CountStatistic	JVM에서 실행을 시작한 후 JVM에서 언로드된 클래스 수입니다.

표 21-18에는 J2SE 5.0에서 JVM의 컴파일에 사용 가능한 통계가 나열되어 있습니다.

표 21-18 J2SE 5.0용 JVM 통계 - 컴파일

통계	데이터 유형	설명
totalcompilationtime	CountStatistic	컴파일에 소비된 누적 시간(밀리초)입니다.

표 21-19에는 J2SE 5.0에서 JVM의 가비지 컬렉션에 사용 가능한 통계가 나열되어 있습니다.

표 21-19 J2SE 5.0용 JVM 통계 - 가비지 컬렉션

통계	데이터 유형	설명
collectioncount	CountStatistic	발생한 총 컬렉션 수입니다.
collectiontime	CountStatistic	누적된 컬렉션 시간(밀리초)입니다.

표 21-20에는 J2SE 5.0에서 JVM의 메모리에 사용 가능한 통계가 표시됩니다.

표 21-20 J2SE 5.0용 JVM 통계 - 메모리

통계	데이터 유형	설명
objectpendingfinalizationcount	CountStatistic	완성을 보류 중인 대략적인 객체 수입니다.
initheapsize	CountStatistic	JVM에서 처음 요청한 힙 크기입니다.
usedheapsize	CountStatistic	현재 사용 중인 힙 크기입니다.
maxheapsize	CountStatistic	메모리 관리에 사용할 수 있는 최대 메모리 양(바이트)입니다.
committedheapsize	CountStatistic	JVM에서 사용하기 위해 완결한 메모리 양(바이트)입니다.
initnonheapsize	CountStatistic	JVM에서 처음 요청한 힙이 아닌 영역의 크기입니다.
usednonheapsize	CountStatistic	현재 사용 중인 힙이 아닌 영역의 크기입니다.
maxnonheapsize	CountStatistic	메모리 관리를 위해 사용할 수 있는 최대 메모리 양(바이트)입니다.
committednonheapsize	CountStatistic	JVM에서 사용하기 위해 완결한 메모리 양(바이트)입니다.

표 21-21에는 J2SE 5.0에서 JVM의 운영 체제에 사용 가능한 통계가 나열되어 있습니다.

표 21-21 J2SE 5.0용 JVM 통계 - 운영 체제

통계	데이터 유형	설명
arch	StringStatistic	운영 체제 구조
availableprocessors	CountStatistic	JVM에 사용 가능한 프로세서 수입니다.
name	StringStatistic	운영 체제 이름입니다.
version	StringStatistic	운영 체제 버전입니다.

표 21-22에는 J2SE 5.0에서 JVM의 런타임에 사용 가능한 통계가 나열되어 있습니다.

표 21-22 J2SE 5.0용 JVM 통계 - 런타임

통계	데이터 유형	설명
name	StringStatistic	실행 중인 JVM을 나타내는 이름입니다.
vmname	StringStatistic	JVM 구현 이름입니다.
vmvendor	StringStatistic	JVM 구현 공급업체입니다.
vmversion	StringStatistic	JVM 구현 버전입니다.
specname	StringStatistic	JVM 사양 이름입니다.
specvendor	StringStatistic	JVM 사양 공급업체입니다.
specversion	StringStatistic	JVM 사양 버전입니다.
managementspecversion	StringStatistic	JVM에서 구현한 관리 사양 버전입니다.
classpath	StringStatistic	클래스 파일을 검색하기 위해 시스템 클래스 로더에서 사용하는 클래스 경로입니다.
librarypath	StringStatistic	Java 라이브러리 경로입니다.
bootclasspath	StringStatistic	클래스 파일을 검색하기 위해 부트스트랩 클래스 로더에서 사용하는 클래스 경로입니다.
inputarguments	StringStatistic	JVM에 전달된 입력 인수입니다. main 메소드에 대한 인수를 포함하지 않습니다.
uptime	CountStatistic	JVM의 가동 시간(밀리초)입니다.

표 21-23에는 J2SE 5.0의 JVM에서 ThreadInfo에 사용 가능한 통계가 나열되어 있습니다.

표 21-23 J2SE 5.0용 JVM 통계 - ThreadInfo

통계	데이터 유형	설명
threadid	CountStatistic	스레드의 아이디입니다.
threadname	StringStatistic	스레드의 이름입니다.
threadstate	StringStatistic	스레드의 상태입니다.
blockedtime	CountStatistic	스레드가 BLOCKED 상태가 된 후 경과된 시간(밀리초)입니다. 스레드 경합 모니터링이 비활성화된 경우 -1을 반환합니다.
blockedcount	CountStatistic	스레드가 BLOCKED 상태가 된 총 횟수입니다.
waitedtime	CountStatistic	스레드가 WAITING 상태인 경과 시간(밀리초)입니다. 스레드 경합 모니터링이 비활성화된 경우 -1을 반환합니다.
waitedcount	CountStatistic	스레드가 WAITING 또는 TIMED_WAITING 상태인 총 횟수입니다.
lockname	StringStatistic	스레드의 입력이 차단되거나 Object.wait 메소드를 통해 통지를 기다리는 모니터 잠금을 나타내는 문자열입니다.
lockownerid	CountStatistic	이 스레드가 차단되는 객체의 모니터 잠금을 수용하는 스레드의 아이디입니다.
lockownername	StringStatistic	이 스레드가 차단되는 객체의 모니터 잠금을 수용하는 스레드의 이름입니다.
stacktrace	StringStatistic	이 스레드와 연관된 스택 추적입니다.

표 21-24에는 J2SE 5.0에서 JVM의 스레드에 사용 가능한 통계가 나열되어 있습니다.

표 21-24 J2SE 5.0용 JVM 통계 - 스레드

통계	데이터 유형	설명
threadcount	CountStatistic	라이브 데몬과 데몬이 아닌 스레드의 현재 수입니다.

표 21-24 J2SE 5.0용 JVM 통계 - 스레드 *(계속)*

통계	데이터 유형	설명
peakthreadcount	CountStatistic	JVM이 시작되거나 최대값을 다시 설정한 후 최대 라이브 스레드 수입니다.
totalstartedthreadcount	CountStatistic	JVM을 시작한 후 만들거나 시작한 총 스레드 수입니다.
daemonthreadcount	CountStatistic	라이브 데몬 스레드의 현재 수입니다.
allthreadids	StringStatistic	모든 라이브 스레드 아이디 목록입니다.
currentthreadcputime	CountStatistic	CPU 시간 측정을 활성화한 경우 현재 스레드의 CPU 시간(나노초)입니다. CPU 시간 측정을 비활성화한 경우 -1을 반환합니다.
monitordeadlockedthreads	StringStatistic	모니터가 교착 상태인 스레드 아이디 목록입니다.

PWC(Production Web Container) 통계

Application Server의 Enterprise Edition(EE)에 있는 다음 PWC 구성 요소와 서비스에 대한 통계를 사용할 수 있습니다.

- 표 21-25, PWC 가상 서버
- 표 21-26, PWC 요청
- 표 21-27, PWC 파일 캐시
- 표 21-28, PWC 연결 유지
- 표 21-29, PWC DNS
- 표 21-30, PWC 스레드 풀
- 표 21-31, PWC 연결 대기열
- 표 21-32, PWC HTTP 서비스

표 21-25에는 PWC 가상 서버에 대한 통계가 나열되어 있습니다.

표 21-25 PWC 가상 서버 통계(EE에만 해당)

속성 이름	데이터 유형	설명
id	String Statistic	가상 서버의 아이디입니다.

표 21-25 PWC 가상 서버 통계(EE에만 해당) *(계속)*

속성 이름	데이터 유형	설명
mode	String Statistic	가상 서버의 모드입니다. 옵션에는 unknown 또는 active가 포함됩니다.
hosts	String Statistic	이 가상 서버에서 처리한 호스트 이름입니다.
interfaces	String Statistic	가상 서버가 구성된 인터페이스(listener) 유형입니다.

표 21-26에는 PWC 요청에 사용 가능한 통계가 나열되어 있습니다.

표 21-26 PWC 요청 통계(EE에만 해당)

속성 이름	데이터 유형	설명
method	String Statistic	요청에 사용되는 메소드입니다.
uri	String Statistic	사용한 마지막 URI입니다.
Countrequests	Count Statistic	사용한 요청 수입니다.
countbytestransmitted	Count Statistic	전송된 바이트 수입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
countbytesreceived	Count Statistic	수신된 바이트 수입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
ratebytesreceived	Count Statistic	서버에서 정의한 시간 간격 동안 데이터가 전송된 속도입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
maxbytestransmissionrate	Count Statistic	서버에서 정의한 시간 간격 동안 데이터가 전송된 최대 속도입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
countopenconnections	Count Statistic	현재 열려 있는 연결의 수입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
maxopenconnections	Count Statistic	동시에 열려 있는 연결의 최대 수입니다. 이 정보를 사용할 수 없는 경우에는 0입니다.
count2xx	Count Statistic	코드 2XX의 총 응답 수입니다.
count3xx	Count Statistic	코드 3XX의 총 응답 수입니다.

표 21-26 PWC 요청 통계(EE에만 해당) *(계속)*

속성 이름	데이터 유형	설명
count4xx	Count Statistic	코드 4XX의 총 응답 수입니다.
count5xx	Count Statistic	코드 5XX의 총 응답 수입니다.
countother	Count Statistic	다른 응답 코드를 가진 총 응답 수입니다.
count200	Count Statistic	코드 200의 총 응답 수입니다.
count302	Count Statistic	코드 302의 총 응답 수입니다.
count304	Count Statistic	코드 304의 총 응답 수입니다.
count400	Count Statistic	코드 400의 총 응답 수입니다.
count401	Count Statistic	코드 401의 총 응답 수입니다.
count403	Count Statistic	코드 403의 총 응답 수입니다.
count404	Count Statistic	코드 404의 총 응답 수입니다.
count503	Count Statistic	코드 503의 총 응답 수입니다.

캐시 정보 절에서는 파일 캐시 사용 방법에 대한 정보를 제공합니다. 표 21-27에는 PWC 파일 캐시에 대한 통계가 나열되어 있습니다.

표 21-27 PWC 파일 캐시 통계(EE에만 해당)

속성 이름	데이터 유형	설명
flagenabled	Count Statistic	파일 캐시가 활성화되어 있는지 여부를 나타냅니다. 유효한 값은 0(no) 또는 1(yes)입니다.
secondsmaxage	Count Statistic	유효한 캐시 항목의 최대 사용 시간(초)입니다.
countentries	Count Statistic	현재 캐시 항목 수 입니다. 단일 캐시 항목은 단일 URI를 나타냅니다.
maxentries	Count Statistic	동시 캐시 항목의 최대 수입니다.

표 21-27 PWC 파일 캐시 통계(EE에만 해당) *(계속)*

속성 이름	데이터 유형	설명
countopenentries	Count Statistic	열려 있는 파일과 관련된 항목 수입니다.
maxopenentries	Count Statistic	열려 있는 파일과 관련된 동시 캐시 항목의 최대 수입니다.
sizeheapcache	Count Statistic	캐시 내용에 사용되는 힙 공간입니다.
maxheapcachesize	Count Statistic	캐시 파일 내용에 사용되는 최대 힙 공간입니다.
sizemmapcache	Count Statistic	메모리 매핑된 파일 내용에서 사용하는 주소 공간 양입니다.
maxmmapcachesize	Count Statistic	파일 캐시에서 메모리 매핑된 파일 내용에 사용하는 최대 주소 공간 양입니다.
counthits	Count Statistic	성공한 캐시 조회 수입니다.
countmisses	Count Statistic	실패한 캐시 조회 수입니다.
countinfohits	Count Statistic	성공한 파일 정보 조회 횟수입니다.
countinfomisses	Count Statistic	캐시된 파일 정보의 오류 수입니다.
countcontenthits	Count Statistic	캐시된 파일 내용의 적중 횟수입니다.
countcontentmisses	Count Statistic	파일 정보 조회가 실패한 횟수입니다.

이 절에서는 서버의 HTTP 수준 연결 유지 시스템에 대한 정보를 제공합니다. 표 21-28에는 PWC 연결 유지에 사용 가능한 통계가 나열되어 있습니다.

표 21-28 PWC 연결 유지 통계(EE에만 해당)

속성 이름	데이터 유형	설명
countconnections	Count Statistic	연결 유지 모드의 연결 수입니다.
maxconnections	Count Statistic	연결 유지 모드에서 동시에 허용되는 최대 연결 수입니다.

표 21-28 PWC 연결 유지 통계(EE에만 해당) *(계속)*

속성 이름	데이터 유형	설명
counthits	Count Statistic	후속적으로 유효한 요청이 이루어진 연결 유지 모드의 총 연결 횟수입니다.
countflushes	Count Statistic	서버가 연결 유지 연결을 닫은 횟수입니다.
countrefusals	Count Statistic	너무 많은 지속성 연결 때문에 연결 유지 스레드에 서버가 연결을 분배할 수 없는 횟수입니다.
counttimeouts	Count Statistic	클라이언트 연결이 작업 없이 시간 만료되었기 때문에 서버에서 연결 유지 연결을 종료한 횟수입니다.
secondstimeout	Count Statistic	유휴 연결 유지 연결을 닫기 전의 시간(초)입니다.

DNS 캐시가 IP 주소와 DNS 이름을 캐시합니다. 서버의 DNS 캐시는 기본적으로 비활성화되어 있습니다. 단일 캐시 항목은 단일 IP 주소나 DNS 이름 조회를 나타냅니다.
표 21-29에는 PWC DNS에 사용 가능한 통계가 나열되어 있습니다.

표 21-29 PWC DNS 통계(EE에만 해당)

속성 이름	데이터 유형	설명
flagcacheenabled	Count Statistic	DNS 캐시가 활성화되어 있는지 여부를 표시합니다. 0(off) 또는 1(on)입니다.
countcacheentries	Count Statistic	현재 캐시에 있는 DNS 항목의 수입니다.
maxcacheentries	Count Statistic	캐시에서 수용할 수 있는 최대 DNS 항목 수입니다.
countcachehits	Count Statistic	DNS 캐시 조회가 성공한 횟수입니다.
countcachemisses	Count Statistic	DNS 캐시 조회가 실패한 횟수입니다.
flagasyncenabled	Count Statistic	비동기 DNS 조회 활성화(on) 여부를 나타냅니다. 0(off) 또는 1(on)입니다.
countasyncnamelookups	Count Statistic	비동기 DNS 이름 조회 총 수입니다.
countasyncaddrlookups	Count Statistic	비동기 DNS 주소 조회 총 수입니다.

표 21-29 PWC DNS 통계(EE에만 해당) *(계속)*

속성 이름	데이터 유형	설명
countasynclookupsinprogress	Count Statistic	처리 중인 비동기 조회 수입니다.

표 21-30에는 PWC 스레드 풀에 대한 통계가 나열되어 있습니다.

표 21-30 PWC 스레드 풀 통계(EE에만 해당)

속성 이름	데이터 유형	설명
id	String Statistic	스레드 풀의 아이디입니다.
countthreadsidle	Count Statistic	현재 유휴 상태인 요청 처리 스레드의 수입니다.
countthreads	Count Statistic	현재 요청 처리 스레드의 수입니다.
maxthreads	Count Statistic	동시에 존재할 수 있는 요청 처리 스레드의 최대 수입니다.
countqueued	Count Statistic	이 스레드 풀에서 처리하기 위해 대기열에 있는 요청 수입니다.
peakqueued	Count Statistic	대기열에 동시에 존재한 최대 요청 수입니다.
maxqueued	Count Statistic	대기열에 한 번에 있을 수 있는 최대 요청 수입니다.

연결 대기열은 요청이 처리되기 전에 대기하는 대기열입니다. 연결 대기열에 대한 통계에서는 대기열의 세션 수, 연결이 승인되기 전의 평균 지연을 나타냅니다. 표 21-31에는 PWC 연결 대기열에 대한 통계가 나열되어 있습니다.

표 21-31 PWC 연결 대기열 통계(EE에만 해당)

속성 이름	데이터 유형	설명
id	String Statistic	연결 대기열의 아이디입니다.
counttotalconnections	Count Statistic	승인된 총 연결 수입니다.

표 21-31 PWC 연결 대기열 통계(EE에만 해당) *(계속)*

속성 이름	데이터 유형	설명
countqueued	Count Statistic	현재 대기열에 있는 연결 수입니다.
peakqueued	Count Statistic	대기열에 동시에 존재한 최대 연결 수입니다.
maxqueued	Count Statistic	연결 대기열의 최대 크기입니다.
countoverflows	Count Statistic	대기열이 가득 차서 연결을 수용할 수 없었던 횟수입니다.
counttotalqueued	Count Statistic	대기열에 있던 연결의 총 수입니다. 특정 연결이 여러 번 대기열에 있을 수 있으므로 CountTotalQueued 값은 CountTotalConnections 보다 크거나 같을 수 있습니다.
tickstotalqueued	Count Statistic	연결이 대기열에서 보낸 시간의 총 눈금 수입니다. 눈금은 시스템 종속 시간 단위입니다.
countqueued1minuteaverage	Count Statistic	마지막 1분 동안 대기열에 있던 연결의 평균 수입니다.
countqueued5minuteaverage	Count Statistic	마지막 5분 동안 대기열에 있던 연결의 평균 수입니다.
countqueued15minuteaverage	Count Statistic	마지막 15분 동안 대기열에 있던 연결의 평균 수입니다.

표 21-32에는 PWC HTTP 서비스에 대한 통계가 나열되어 있습니다.

표 21-32 PWC HTTP 서비스 통계(EE에만 해당)

속성 이름	데이터 유형	설명
id	String Statistic	HTTP 서비스의 인스턴스 이름입니다.
versionserver	String Statistic	HTTP 서비스의 버전 번호입니다.
timestarted	String Statistic	HTTP 서비스가 시작된 시간(GMT)입니다.
secondsrunning	Count Statistic	HTTP 서비스가 시작된 이후의 시간(초)입니다.

표 21-32 PWC HTTP 서비스 통계(EE에만 해당) *(계속)*

속성 이름	데이터 유형	설명
maxthreads	Count Statistic	각 인스턴스에 있는 최대 작업자 스레드 수입니다.
maxvirtualservers	Count Statistic	각 인스턴스에 구성할 수 있는 최대 가상 서버 수입니다.
flagprofilingenabled	Count Statistic	HTTP 서비스 성능 프로필의 활성화 여부입니다. 유효한 값은 0 또는 1입니다.
flagvirtualserveroverflow	Count Statistic	maxvirtualservers 이상으로 구성되었는지 여부를 나타냅니다. 이 속성이 1로 설정되면 모든 가상 서버에 대한 통계가 추적되지 않습니다.
load1minuteaverage	Count Statistic	마지막 1분 동안의 요청에 대한 평균 로드입니다.
load5minuteaverage	Count Statistic	마지막 5분 동안의 요청에 대한 평균 로드입니다.
load15minuteaverage	Count Statistic	마지막 15분 동안의 요청에 대한 평균 로드입니다.
ratebytestransmitted	Count Statistic	서버가 정의한 간격 동안 데이터가 전송된 속도입니다. 이 정보를 사용할 수 없을 경우 결과는 0입니다.
ratebytesreceived	Count Statistic	서버가 정의한 간격 동안 데이터가 수신된 속도입니다. 이 정보를 사용할 수 없을 경우 결과는 0입니다.

모니터링 활성화 또는 비활성화를 위한 관리 콘솔 작업

- 관리 콘솔을 사용하여 모니터링 수준 구성
- asadmin 도구를 사용하여 모니터링 구성

관리 콘솔을 사용하여 모니터링 수준 구성

1. 모니터링 서비스 페이지에 액세스합니다. 이 페이지에 액세스하려면 다음 작업을 수행합니다.

a. 트리에서 구성 노드를 확장합니다.

b. 트리에서 모니터링을 위해 구성할 서버 인스턴스 노드(예: `server-config`)를 확장합니다.

c. 트리에서 모니터링을 선택합니다.

2. 모니터링 서비스 페이지에서 모니터링 수준을 변경할 구성 요소나 서비스 반대편에 있는 콤보 상자에서 적절한 값을 선택합니다.

   항상 모니터 가능한 Java Virtual Machine(JVM)을 제외한 모든 구성 요소 및 서비스에 대해 기본적으로 모니터링이 해제되어 있습니다. 모니터링을 설정하려면 콤보 상자에서 낮음 또는 높음을 선택합니다. 모니터링을 해제하려면 콤보 상자에서 해제를 선택합니다. 다음 구성 요소 및 서비스에 대해 모니터링을 설정하거나 해제할 수 있습니다.

   - **JVM** - 이 옵션에서 Java Virtual Machine을 모니터링하려면 모니터링 수준을 낮음으로 설정합니다.
   - **HTTP 서비스** - 이 옵션에서 모든 HTTP listener와 가상 서버를 모니터링하려면 모니터링 수준을 낮음으로 설정합니다.
   - **트랜잭션 서비스** - 이 옵션에서 모든 트랜잭션 하위 시스템을 모니터링하려면 모니터링 수준을 낮음으로 설정합니다.
   - **JMS/커넥터 서비스** - 이 옵션에서 모든 JMS(Java Message Service)를 모니터링하려면 모니터링 수준을 낮음으로 설정합니다.
   - **ORB** - 이 옵션에서 Application Server 코어 및 해당하는 연결 관리자가 사용한 시스템 ORB를 모니터링하려면 모니터링 수준을 낮음으로 설정합니다.
   - **웹 컨테이너** - 이 옵션에서 모든 배포된 서블릿을 모니터링하려면 모니터링 수준을 낮음으로 설정합니다.
   - **EJB 컨테이너** - 이 옵션에서 모든 배포된 EJB, EJB 풀 및 EJB 캐시를 모니터링하려면 이 모니터링 수준을 낮음으로 설정합니다. EJB 비즈니스 메소드를 모니터링하려면 이 메소드를 높음으로 설정합니다.
   - **JDBC 연결 풀** - 이 옵션에서 모든 JDBC 연결 풀을 모니터링하려면 모니터링 수준을 낮음으로 설정합니다.
   - **스레드 풀** - 이 옵션에서 모든 스레드 풀을 모니터링하려면 모니터링 수준을 낮음으로 설정합니다.

3. 저장을 누릅니다.

이 릴리스에는 추가 모니터링 서비스 등록 정보가 없으므로 추가 등록 정보 테이블은 무시합니다.

해당 asadmin 명령: set
예를 들어, HTTP 서비스에 대한 모니터링을 설정하려면 다음 asadmin 명령을 사용합니다.

```
asadmin> set --user admin_user
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

asadmin 도구를 사용하여 모니터링 구성

모니터링을 해제하거나 구성 요소 또는 서비스를 모니터링하는 수준을 설정하려면 "관리 콘솔을 사용하여 모니터링 수준 구성"에서 설명한 관리 콘솔을 사용하거나 이 절에서 설명한 asadmin 도구를 사용합니다.

1. get 명령을 사용하여 현재 모니터링이 활성화된 서비스와 구성 요소를 확인합니다.

   ```
   asadmin> get --user admin_user
   server.monitoring-service.module-monitoring-levels.*
   ```

   반환 결과:

   ```
   server.monitoring-service.module-monitoring-levels.
   connector-connection-pool = OFF
   server.monitoring-service.module-monitoring-levels.
   connector-service = OFF
   server.monitoring-service.module-monitoring-levels.ejb-container = OFF
   server.monitoring-service.module-monitoring-levels.http-service = OFF
   server.monitoring-service.module-monitoring-levels.jdbc-connection-pool
   = OFF
   server.monitoring-service.module-monitoring-levels.jms-service = OFF
   server.monitoring-service.module-monitoring-levels.jvm = OFF
   server.monitoring-service.module-monitoring-levels.orb = OFF
   server.monitoring-service.module-monitoring-levels.thread-pool = OFF
   server.monitoring-service.module-monitoring-levels.transaction-service
   = OFF
   server.monitoring-service.module-monitoring-levels.web-container = OFF
   ```

2. set 명령을 사용하여 모니터링을 활성화합니다.

   예를 들어, HTTP 서비스에 대한 모니터링을 활성화하려면 다음을 사용합니다.

```
asadmin> set --user admin_user
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

모니터링을 비활성화하려면 `set` 명령을 사용하고 모니터링 수준에 대해 `OFF`를 지정합니다.

모니터링 데이터 보기를 위한 관리 콘솔 작업

관리 콘솔에서 모니터링 데이터 보기

Application Server 관리 콘솔을 사용하여 서버 인스턴스에 배포된 구성 요소나 서비스에 대한 모니터링 데이터를 보려면 다음 단계를 수행합니다. 모든 구성 요소나 서비스의 속성에 대한 자세한 설명은 "모니터된 구성 요소 및 서비스에 대한 통계 정보"를 참조하십시오.

1. 모니터링 페이지에 액세스하려면 다음 작업을 수행합니다.
   a. 트리 구성 요소에서 독립 실행형 인스턴스 노드(예: `server(Admin Server)`)를 확장합니다.
   b. 목록에서 독립 실행형 서버 인스턴스를 선택합니다.
   c. 모니터 페이지를 선택합니다.
   d. 모니터 페이지에서 모니터 탭을 선택합니다.
2. 보기 목록에서 서버 인스턴스에 배포되고 모니터링이 활성화된 구성 요소나 서비스를 선택합니다.

   보기 필드 아래에 선택한 구성 요소나 서비스의 모니터링 데이터가 표시됩니다. 모니터 가능한 등록 정보에 대한 설명은 "모니터된 구성 요소 및 서비스에 대한 통계 정보"를 참조하십시오.

   이 페이지에서 해당 구성 요소와 서비스에 대한 모니터링이 활성화된 경우 JVM, 서버, 스레드 풀, HTTP 서비스 및 트랜잭션 서비스에 대한 모니터링 데이터를 볼 수 있습니다. "모니터링 가능한 객체의 트리 구조 정보"에는 이 구성 요소와 서비스 구성 방법을 보여주는 다이어그램이 있습니다.

3. 이 목록에 모니터할 구성 요소나 서비스가 없을 경우 모니터링 구성 링크를 선택하여 선택한 구성 요소와 서비스에 대한 모니터링을 활성화 및 비활성화합니다. 구성 요소나 서비스에 대한 모니터링을 비활성화하려면 해제를 선택합니다. 구성 요소나 서비스에 대한 모니터링을 활성화하려면 낮음 또는 높음을 선택합니다.

   모니터링 활성화 및 비활성화에 대한 자세한 내용은 "관리 콘솔을 사용하여 모니터링 수준 구성" 또는 "asadmin 도구를 사용하여 모니터링 구성"을 참조하십시오.

4. 서버 인스턴스에 배포되고 모니터링이 활성화된 응용 프로그램 구성 요소의 모니터링 데이터를 조회하려면 모니터 페이지의 *응용 프로그램* 탭을 선택합니다. 응용 프로그램 목록에서 응용 프로그램을 선택합니다. 구성 요소 목록에서 특정 구성 요소를 선택합니다.

   응용 프로그램이나 구성 요소에 대한 모니터링 데이터가 표시되지 않을 경우 모니터링 구성 링크를 선택하여 구성 요소나 서비스의 모니터링을 활성화하거나 비활성화합니다. 응용 프로그램을 모니터링하려면 응용 프로그램이 실행된 컨테이너를 설정합니다. 예를 들어, 웹 응용 프로그램의 경우 웹 컨테이너에 대해 또는 EJB 응용 프로그램의 경우 EJB 컨테이너에 대해 낮음 또는 높음을 선택합니다.

   응용 프로그램에 대한 모니터링 데이터가 표시되지 않으면 응용 프로그램이 존재하지 않거나 응용 프로그램이 실행되지 않은 것 같습니다. 응용 프로그램이 존재하고, 응용 프로그램에 대한 모니터링이 활성화되고, 응용 프로그램을 실행 중일 경우에만 응용 프로그램 모니터링 데이터를 사용할 수 있습니다. 응용 프로그램이 실행되면 모니터링 레지스트리에 응용 프로그램이 등록되고 모니터링 데이터가 표시됩니다.

   관리 콘솔을 사용하여 원격 응용 프로그램 및 인스턴스를 모니터링합니다. 이를 위해서는 원격 인스턴스가 실행 중이고 구성이 설정되어야 합니다.

   선택한 구성 요소 아래에 선택한 구성 요소의 모니터링 데이터가 표시됩니다. 모니터 가능한 등록 정보에 대한 설명은 "모니터된 구성 요소 및 서비스에 대한 통계 정보"를 참조하십시오. "모니터링 가능한 객체의 트리 구조 정보"에서 이 구성 요소 및 서비스를 응용 프로그램에 대해 구성하는 방법을 보여주는 다이어그램을 확인할 수 있습니다.

5. 서버 인스턴스에 배포되고 모니터링이 활성화된 자원의 모니터링 데이터를 보려면 *자원* 페이지를 선택합니다. 보기 목록에서 자원을 선택합니다. 모니터링 데이터를 조회할 자원이 표시되지 않으면 모니터링 구성 링크를 선택하여 자원에 대한 모니터링을 활성화하거나 비활성화합니다.

   자원에 대한 모니터링 데이터가 표시되지 않으면 자원이 존재하지 않거나 자원이 실행되지 않는 것 같습니다. 자원이 존재하고, 자원에 대한 모니터링이 높음 수준으로 활성화되고, 자원이 실행 중일 경우에만 자원 모니터링 데이터를 사용할 수 있습니다. 예를 들어, JDBC 커넥터 서비스를 만들었지만, 해당 커넥터 서비스를 사용하는 응용

프로그램이 서비스에서 커넥터를 요청하지 않은 경우, 그리고 해당 서비스가 아직 만들어지지 않은 경우, 서비스가 존재하지 않으므로 모니터링 데이터를 사용할 수 없습니다. JDBC 응용 프로그램이 실행되고 서비스에서 커넥터를 요청한 경우 서비스가 모니터링 레지스트리에 등록되고 모니터링 데이터가 표시됩니다.

보기 필드 아래에 선택한 구성 요소나 서비스의 모니터링 데이터가 표시됩니다. 모니터 가능한 등록 정보에 대한 설명은 "모니터된 구성 요소 및 서비스에 대한 통계 정보"를 참조하십시오. "모니터링 가능한 객체의 트리 구조 정보"에서 이 구성 요소 및 서비스를 자원에 대해 구성하는 방법을 보여주는 다이어그램을 확인할 수 있습니다.

6. *트랜잭션* 페이지를 선택하여 트랜잭션 하위 시스템을 중단하여 트랜잭션을 롤백하고 중단 시에 처리 중인 트랜잭션을 확인합니다. 트랜잭션 서비스에 대한 모니터링을 활성화하려면 모니터링 구성 링크를 선택하고 트랜잭션 서비스를 낮음으로 설정합니다. 트랜잭션을 롤백하기 위해 트랜잭션 서비스를 중단하려면 중단을 선택합니다. 트랜잭션을 롤백하려면 트랜잭션 옆에 있는 확인란을 선택하고 롤백을 누릅니다.

해당 asadmin 명령: `get --monitor`
예를 들어, JVM에 대한 모니터링 데이터를 보려면 다음 asadmin 명령을 사용합니다.

```
asadmin> get --monitor server.jvm.*
```

asadmin 도구를 사용하여 모니터링 데이터 보기

asadmin 도구를 사용하여 모니터링 데이터 보기

asadmin 도구를 사용하여 모니터링 데이터를 보려면 모니터 가능한 객체의 점으로 구분된 이름 다음에 나오는 asadmin 목록과 asadmin get 명령을 사용합니다. asadmin 도구를 사용하여 모니터링 데이터를 조회하기 위한 일반적인 방법으로 다음 단계를 수행합니다.

1. 모니터링할 수 있는 객체 이름을 보려면 asadmin list 명령을 사용합니다. 예를 들어, 서버 인스턴스에 대한 모니터링을 활성화한 응용 프로그램 구성 요소와 하위 시스템 목록을 보려면 단말기 창에 다음 명령을 입력합니다.

   ```
   asadmin> list --monitor server
   ```

   위 명령은 모니터링이 활성화된 응용 프로그램 구성 요소와 하위 시스템 목록을 반환합니다. 예를 들면, 다음과 같습니다.

   ```
   server.resources
   server.connector-service
   server.log
   server.jms-service
   server.jvm
   server.applications
   server.http-service
   server.thread-pools
   ```

   list 명령 사용에 대한 자세한 예는 "list 및 get 명령 예"를 참조하십시오. list 명령과 같이 사용할 수 있는 점으로 구분된 이름에 대한 자세한 내용은 "점으로 구분된 이름 이해 및 지정"을 참조하십시오.

2. 모니터링이 활성화된 응용 프로그램 구성 요소나 하위 시스템에 대한 모니터링 통계를 표시하려면 asadmin get 명령을 사용합니다. 통계를 구하려면 단말기 창에서 asadmin get 명령을 입력하고 이전 단계의 list 명령에서 표시한 이름을 지정합니다. 다음 예에서는 특정 객체의 하위 시스템에서 모든 속성을 가져옵니다.

   ```
   asadmin> get --monitor server.jvm.*
   ```

명령은 다음 속성과 데이터를 반환합니다.

```
server.jvm.dotted-name = server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information
about the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
```

```
been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

get 명령 사용에 대한 자세한 예는 "list 및 get 명령 예"를 참조하십시오. get 명령과 같이 사용할 수 있는 점으로 구분된 이름에 대한 자세한 내용은 "점으로 구분된 이름 이해 및 지정"을 참조하십시오.

점으로 구분된 이름 이해 및 지정

asadmin list 및 get 명령에서 모니터 가능한 객체의 점으로 구분된 이름을 지정합니다. 점(.) 문자를 구분자로 사용하여 모든 자식 객체의 주소를 지정하므로 이를 *점으로 구분된 이름*이라고 합니다. 자식 노드가 싱글톤 유형인 경우에는 그 객체를 나타내는 데 모니터링 객체 유형만 필요하고 그렇지 않은 경우에는 type.name 형태의 이름이 필요합니다.

예를 들어, http-service는 유효한 모니터링 가능 객체 유형 중 하나이며 싱글톤입니다. server 인스턴스의 http-service를 를 나타내는 싱글톤 자식 노드를 표시하려면 다음 이름을 사용합니다.

```
server.http-service
```

다른 예를 들어보면, application은 유효한 모니터링 가능 객체 유형이며 싱글톤이 아닙니다. Petstore 응용 프로그램을 나타내는데 싱글톤이 아닌 자식 노드를 표시하려면 다음과 같은 점으로 구분된 이름을 사용합니다.

```
server.applications.petstore
```

점으로 구분된 이름을 사용하여 모니터할 수 있는 객체의 특정 속성을 나타낼 수도 있습니다. 예를 들어, http-service는 모니터할 수 있는 bytereceived-lastsampletime이라는 속성을 가집니다. 다음 이름은 bytesreceived 속성을 나타냅니다.

```
server.http-service.server.http-listener-1.
  bytesreceived-lastsampletime
```

관리자는 asadmin list 및 get 명령에 유효한 점으로 구분된 이름을 알지 못합니다. list 명령을 사용하여 모니터할 수 있는 객체를 표시하는 한편 get 명령을 와일드카드 매개변수와 함께 사용하여 모니터할 수 있는 객체의 모든 사용 가능한 속성을 검사할 수 있습니다.

점으로 구분된 이름과 함께 list 및 get 명령을 사용하기 위해 다음을 가정합니다.

- 뒤에 와일드카드(*)가 나오지 **않는** 점으로 구분된 이름을 가진 `list` 명령은 현재 노드의 자식을 표시합니다. 예를 들어, `list --monitor server`는 `server` 노드에 속하는 모든 자식을 표시합니다.
- 점으로 구분된 이름 다음에 `.*` 양식의 와일드카드가 있는 `list` 명령은 현재 노드에서 자식 노드의 계층 트리를 만듭니다. 예를 들어, `list --monitor server.applications.*`는 `applications`의 모든 자식과 후속 자식 노드를 나열합니다.
- 점으로 구분된 이름이 앞에 있거나 **dottedname*, *dotted * name* 또는 **`dotted name *`** 양식의 와일드카드가 뒤에 나오는 `list` 명령은 제공한 비교 패턴으로 만들어진 정규 표현식과 일치하는 모든 노드와 자식을 표시합니다.
- 뒤에 `.*` 또는 `*`가 나오는 `get` 명령은 비교할 현재 노드에 속하는 속성 및 속성 값 세트를 표시합니다.

자세한 내용은 "모든 수준의 list 및 get 명령에 대한 예상 출력"을 참조하십시오.

list 및 get 명령 예

이 절에서는 다음 항목에 대해 설명합니다.

- list --monitor 명령 예
- get --monitor 명령 예
- Petstore 예

list --monitor 명령 예

`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. `list` 예의 전체 목록은 "모든 수준의 list 및 get 명령에 대한 예상 출력"을 참조하십시오.

예 1

```
asadmin> list --monitor server
```

예를 들어, 이전 명령에서는 모니터링이 활성화된 응용 프로그램 구성 요소와 하위 시스템 목록을 반환합니다.

```
server.resources
server.orb
server.jvm
server.jms-service
```

```
server.connector-service
server.applications
server.http-service
server.thread-pools
```

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 2

```
asadmin> list --monitor server.applications
```

반환 결과:

```
server.applications.adminapp
server.applications.admingui
server.applications.myApp
```

보다 많은 예는 "Petstore 예"를 참조하십시오.

get --monitor 명령 예

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 "모든 수준의 list 및 get 명령에 대한 예상 출력"을 참조하십시오.

예 1

하위 시스템에서 특정 객체에 대한 모든 속성을 검색하려는 경우:

```
asadmin> get --monitor server.jvm.*
```

반환 결과:

```
server.jvm.dotted-name= server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information
about the JVMs memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
```

```
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

예 2

J2EE 응용 프로그램에서 모든 속성을 검색하려는 경우:

```
asadmin> get --monitor server.applications.myJ2eeApp.*
```

반환 결과:

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

J2EE 응용 프로그램 수준에 모니터할 수 있는 속성이 없기 때문에 이 응답이 표시됩니다.

예 3

하위 시스템에서 특정 속성을 검색하려는 경우:

```
asadmin> get --monitor server.jvm.uptime-lastsampletime
```

반환 결과:

```
server.jvm.uptime-lastsampletime = 1093215374813
```

예 4

하위 시스템 속성 내에서 알 수 없는 속성을 가져오려는 경우:

```
asadmin> get --monitor server.jvm.badname
```

반환 결과:

```
No such attribute found from reflecting the corresponding Stats
interface: [badname]
CLI137 Command get failed.
```

Petstore 예

다음 예에서는 모니터링을 위해 asadmin 도구를 사용하는 방법에 대해 설명합니다.

사용자가 샘플 Petstore 응용 프로그램을 Application Server에 배포한 후 메소드가 호출된 횟수를 검사하려고 합니다. 응용 프로그램이 배포된 인스턴스 이름은 server입니다. list 명령과 get 명령을 조합하여 메소드에 대한 원하는 통계에 액세스합니다.

1. Application Server 및 asadmin 도구를 시작합니다.
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. server 인스턴스에 대해 모니터할 수 있는 구성 요소를 나열합니다.

   ```
   asadmin> list --monitor server*
   ```

   반환 결과(다음과 유사한 출력):

   ```
   server
   server.applications
   server.applications.CometEJB
   server.applications.ConverterApp
   server.applications.pestore
   server.http-service
   server.resources
   server.thread-pools
   ```

   모니터할 수 있는 구성 요소 목록에는 thread-pools, http-service, resources와 배포 및 활성화된 applications가 포함됩니다.

4. Petstore 응용 프로그램의 모니터할 수 있는 하위 구성 요소를 나열합니다(--monitor 대신 -m을 사용할 수 있음).

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

   반환 결과:

   ```
   server.applications.petstore.signon-ejb_jar
   server.applications.petstore.catalog-ejb_jar
   server.applications.petstore.uidgen-ejb_jar
   server.applications.petstore.customer-ejb_jar
   ```

```
server.applications.petstore.petstore-ejb_jar
server.applications.petstore.petstore\.war
server.applications.petstore.AsyncSenderJAR_jar
server.applications.petstore.cart-ejb_jar
```

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

   ```
   asadmin>list -m server.applications.petstore.signon-ejb_jar
   ```

   반환 결과:

   ```
   server.applications.petstore.signon-ejb_jar.SignOnEJB
   server.applications.petstore.signon-ejb_jar.UserEJB
   ```

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

   ```
   asadmin>list -m server.applications.petstore.signon-ejb_jar.UserEJB
   ```

   반환 결과(공간 때문에 제거한 점으로 구분된 이름이 있음):

   ```
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-cache
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-pool
   ```

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

   ```
   asadmin>list  -m server.applications.petstore.signon-ejb_jar.
   UserEJB.bean-methods.getUserName
   ```

   반환 결과:

   ```
   Nothing to list at server.applications.petstore.signon-ejb_jar.
   UserEJB.bean-methods.getUserName. To get the valid names beginning with
   a string, use the wildcard "*" character. For example, to list all names
   that begin with "server", use "list server*".
   ```

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

   ```
   asadmin>get -m server.applications.petstore.signon-ejb_jar.
   UserEJB.bean-methods.getUserName.*
   ```

   반환 결과:

   ```
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
   getUserName.executiontime-count = 0
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
   getUserName.executiontime-description = Provides the time in
   milliseconds spent during the last successful/unsuccessful attempt
   ```

```
to execute the operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-lastsampletime = 1079981809259
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-description = rovides the number of times an
operation was called, the total time that was spent during the
invocation and so on.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-lastsampletime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-maxtime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-mintime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-totaltime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-unit =
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-description = Provides the total number of
errors that occured during invocation or execution of an operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-lastsampletime = 1079981809273
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-name = TotalNumErrors
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
```

```
getUserName.totalnumsuccess-description = Provides the total number of
successful invocations of the method.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-lastsampletime = 1079981809255
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-name = TotalNumSuccess
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-unit = count
```

9. 실행 시간과 같은 특정 통계를 얻으려면 다음 명령을 사용합니다.

```
asadmin>get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count
```

반환 결과:

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1
```

모든 수준의 list 및 get 명령에 대한 예상 출력

다음 표에서는 트리의 모든 수준에서의 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 21-33 최상위 수준

명령	점으로 구분된 이름	출력
`list -m`	`server`	`server.applications` `server.thread-pools` `server.resources` `server.http-service` `server.transaction-service` `server.orb.connection-managers` `server.orb.connection-managers.o` `rb\.Connections\.Inbound\.` `AcceptedConnections` `server.jvm`
`list -m`	`server.*`	이 노드 아래의 자식 노드 계층입니다.
`get -m`	`server.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.

표 21-34에서는 응용 프로그램 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 21-34 응용 프로그램 수준

명령	점으로 구분된 이름	출력
`list -m`	`server.applications` 또는 `*applications`	`appl1` `app2` `web-module1_war` `ejb-module2_jar` `...`
`list -m`	`server.applications.*` 또는 `*applications.*`	이 노드 아래의 자식 노드 계층입니다.
`get -m`	`server.applications.*` 또는 `*applications.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.

표 21-35에서는 응용 프로그램 수준의 독립 실행형 모듈 및 엔터프라이즈 응용 프로그램에 대한 명령, 점으로 구분된 이름 및 해당 출력을 보여줍니다.

표 21-35 응용 프로그램 - 엔터프라이즈 응용 프로그램 및 독립 실행형 모듈

명령	점으로 구분된 이름	출력
`list -m`	`server.applications.app1` 또는 `*app1` *주: 엔터프라이즈 응용 프로그램이 배포된 경우에만 이 수준을 적용할 수 있습니다. 독립 실행형 모듈이 배포된 경우에는 적용할 수 없습니다.*	`ejb-module1_jar` `web-module2_war` `ejb-module3_jar` `web-module3_war` `...`
`list -m`	`server.applications.app1.*` 또는 `*app1.*`	이 노드 아래의 자식 노드 계층입니다.
`get -m`	`server.applications.app1.*` 또는 `*app1.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.

표 21-35 응용 프로그램 - 엔터프라이즈 응용 프로그램 및 독립 실행형 모듈 *(계속)*

명령	점으로 구분된 이름	출력
`list -m`	`server.applications.app1.ejb-module1_jar` 또는 `*ejb-module1_jar` 또는 `server.applications.ejb-module1_jar`	`bean1` `bean2` `bean3` `...`
`list -m`	`server.applications.app1.ejb-module1_jar` 또는 `*ejb-module1_jar` 또는 `server.applications.ejb-module1_jar`	이 노드 아래의 자식 노드 계층입니다.
`get -m`	`server.applications.app1.ejb-module1_jar.*` 또는 `*ejb-module1_jar.*` 또는 `server.applications.ejb-module1_jar.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.applications.app1.ejb-module1_jar.bean1` *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	자식 노드 목록: `bean-pool` `bean-cache` `bean-method`
`list -m`	`server.applications.app1.ejb-module1_jar.bean1` *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	자식 노드의 계층 구조 및 이 노드와 모든 후속 자식 노드에 대한 모든 속성 목록입니다.

표 21-35 응용 프로그램 - 엔터프라이즈 응용 프로그램 및 독립 실행형 모듈 *(계속)*

명령	점으로 구분된 이름	출력
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.*` *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	속성 및 해당하는 연관된 값은 다음과 같습니다. `CreateCount_Count` `CreateCount_Description` `CreateCount_LastSampleTime` `CreateCount_Name` `CreateCount_StartTime` `CreateCount_Unit` `MethodReadyCount_Current` `MethodReadyCount_Description` `MethodReadyCount_HighWaterMark` `MethodReadyCount_LastSampleTime` `MethodReadyCount_LowWaterMark` `MethodReadyCount_Name` `MethodReadyCount_StartTime` `MethodReadyCount_Unit` `RemoveCount_Count` `RemoveCount_Description` `RemoveCount_LastSampleTime` `RemoveCount_Name` `RemoveCount_StartTime` `Attribute RemoveCount_Unit`
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-pool` *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-pool.*` *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	표 1-4에서 설명한 EJB 풀 속성에 해당하는 속성 및 값 목록입니다.
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-cache` *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-cache.*` *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	표 1-5에서 설명한 EJB 캐시 속성에 해당하는 속성 및 값 목록입니다.

표 21-35 응용 프로그램 - 엔터프라이즈 응용 프로그램 및 독립 실행형 모듈 *(계속)*

명령	점으로 구분된 이름	출력
list -m	server.applications.app1.ejb-module1_jar.bean1.bean-method.method1 *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
get -m	server.applications.app1.ejb-module1_jar.bean1.bean-method.method1.* *주: 독립 실행형 모듈의 경우 응용 프로그램 이름(이 예에서 app1)이 포함된 노드는 표시되지 않습니다.*	표 1-2에서 설명한 EJB 메소드 속성에 해당하는 속성 및 값 목록입니다.
list -m	server.applications.app1.web-module1_war	모듈에 할당된 가상 서버를 표시합니다.
get -m	server.applications.app1.web-module1_war.*	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
list -m	server.applications.app1.web-module1_war.virtual_server	등록된 서블릿 목록을 표시합니다.
get -m	server.applications.app1.web-module1_war.virtual_server.*	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
list -m	server.applications.app1.web-module1_war.virtual_server.servlet1	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
get -m	server.applications.app1.web-module1_war.virtual_server.servlet1.*	표 1-7에서 설명한 웹 컨테이너(서블릿) 속성에 해당하는 속성 및 값 목록입니다.

표 21-36에서는 HTTP 서비스 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 표시합니다.

표 21-36 HTTP 서비스 수준

명령	점으로 구분된 이름	출력
list -m	server.http-service	가상 서버 목록입니다.
get -m	server.http-service.*	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
list -m	server.http-service.server	HTTP Listener 목록입니다.
get -m	server.http-service.server.*	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.

표 21-36 HTTP 서비스 수준 *(계속)*

명령	점으로 구분된 이름	출력
`list -m`	`server.http-service.server http-listener1`	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.http-service.server.*`	표 1-9에서 설명한 HTTP 서비스 속성에 해당하는 속성 및 값 목록.

표 21-37에서는 스레드 풀 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 표시합니다.

표 21-37 스레드 풀 수준

명령	점으로 구분된 이름	출력
`list -m`	`server.thread-pools`	스레드 풀 이름 목록입니다.
`get -m`	`server.thread-pools.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.thread-pools.orb\.threadpool\ .thread-pool-1`	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.thread-pools..orb\.threadpool\ .thread-pool-1.*`	표 1-14에서 설명한 스레드 풀 속성에 해당하는 속성 및 값 목록입니다.

표 21-38에서는 자원 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 표시합니다.

표 21-38 자원 수준

명령	점으로 구분된 이름	출력
`list -m`	`server.resources`	풀 이름 목록입니다.
`get -m`	`server.resources.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.resources.jdbc-connection-pool-pool.connection-pool1`	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.

표 21-38 자원 수준 *(계속)*

명령	점으로 구분된 이름	출력
`get -m`	`server.resources.jdbc-connection-pool-pool.connection-pool1.*`	표 1-10에서 설명한 연결 풀 속성에 해당하는 속성 및 값 목록입니다.

표 21-39에서는 트랜잭션 서비스 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 표시합니다.

표 21-39 트랜잭션 서비스 수준

명령	점으로 구분된 이름	출력
`list -m`	`server.transaction-service`	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.transaction-service.*`	표 1-15에서 설명한 트랜잭션 서비스 속성에 해당하는 속성 및 값 목록입니다.

표 21-40에서는 ORB 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 표시합니다.

표 21-40 ORB 수준

명령	점으로 구분된 이름	출력
`list -m`	`server.orb`	`server-orb.connection-managers`
`get -m`	`server.orb.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.orb.connection-managers`	ORB 연결 관리자의 이름입니다.
`get -m`	`server.orb.connection-managers.*`	이 노드에는 해당하는 속성이 없음을 알리는 메시지만 표시됩니다.
`list -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections`	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.

표 21-40 ORB 수준 *(계속)*

명령	점으로 구분된 이름	출력
`get -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections.*`	표 1-13에서 설명한 ORB 연결 관리자 속성에 해당하는 속성 및 값 목록입니다.

표 21-34에서는 JVM 수준에 대한 명령, 점으로 구분된 이름 및 해당 출력을 표시합니다.

표 21-41 JVM 수준

명령	점으로 구분된 이름	출력
`list -m`	`server.jvm`	해당 속성은 없지만 "Use get command with the --monitor option to view this node's attributes and values" 메시지가 표시됩니다.
`get -m`	`server.jvm.*`	표 1-16에서 설명한 JVM 속성에 해당하는 속성 및 값 목록입니다.

Jconsole 사용

Jconsole을 Application Server와 같이 사용하려면 JMX 커넥터에 대한 보안을 비활성화해야 합니다. Application Server의 현재 버전(SE/EE)에는 기본적으로 보안이 활성화되어 있습니다.

JMX 커넥터에 대한 보안을 비활성화하려면 다음 방법 중 하나를 사용합니다.

1. 관리 콘솔을 사용하여 JMX 커넥터에 대한 보안을 비활성화합니다. 관리 콘솔에서 다음 작업을 수행합니다.
   a. 구성 노드를 확장합니다.
   b. 관리 서비스 노드를 확장합니다.
   c. `system` 노드를 선택합니다.
   d. SSL 섹션에서 SSL3 및 TLS를 선택 해제합니다.
   e. 저장을 선택합니다.
2. `asadmin`을 사용하여 JMX 커넥터에 대한 보안을 비활성화합니다. 단말기 창이나 명령 프롬프트에서 이를 수행하려면 다음 작업을 수행합니다.

a. 다음 명령을 입력합니다.

```
asadmin set
server.admin-service.jmx-connector.system.security-enabled=false
```

b. DAS(도메인 응용 프로그램 서버)를 다시 시작합니다.

PE 버전의 경우 기본적으로 JMX 커넥터가 비활성화되어 있으므로 PE의 구성을 변경할 필요가 없습니다.

3. Jconsole을 시작하고 로그인하려면 고급 탭에서 JMX URL, 아이디 및 비밀번호를 입력합니다. JMX URL의 양식은 다음과 같습니다.

```
service:jmx:rmi:///jndi/rmi://<your machine
name>:<port>/management/rmi-jmx-connector
```

주: `message ADM1501`을 검색할 경우 관리 `server.log` 파일에서 정확한 JMX URL을 가져올 수 있습니다.

Java Virtual Machine(JVM) 및 고급 설정

이 장에서는 Java Virtual Machine(JVM™) 및 기타 고급 설정을 구성하는 방법에 대해 설명합니다. 이 장은 다음 내용으로 구성되어 있습니다.

- JVM™ 설정을 위한 관리 콘솔 작업
- 고급 설정을 위한 관리 콘솔 작업

JVM™ 설정을 위한 관리 콘솔 작업

- JVM 일반 설정 구성
- JVM 클래스 경로 설정 구성
- JVM 옵션 구성
- 보안 관리자 비활성화
- JVM 프로필러 설정 구성

JVM 일반 설정 구성

Java Virtual Machine(JVM)은 Application Server에 필요한 J2SE™(Java 2 Standard Edition) 소프트웨어에 포함되어 있습니다. JVM 설정을 잘못하면 서버가 실행되지 않기 때문에 이 설정을 변경할 경우 주의를 기울여야 합니다.

Application Server에서 사용하는 JVM의 일반 설정을 구성하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 서버 노드를 선택합니다.

2. ORB 탭을 누릅니다.
3. 기본적으로 탭 아래 있는 일반 링크는 이미 선택되어 있습니다.
4. JVM 일반 설정 페이지에서 다음을 지정할 수 있습니다.
   a. Java 홈 필드에서 J2SE(Java 2 Standard Edition) 소프트웨어의 설치 디렉토리 이름을 입력합니다.

      Application Server는 J2SE 소프트웨어를 사용합니다. 지정한 J2SE 버전이 이 릴리스에서 지원되는지 확인하려면 릴리스 노트를 참조하십시오. 추가 정보 절의 링크를 참조하십시오.

      주: 존재하지 않는 디렉토리 이름을 입력하거나 지원되지 않는 J2SE 소프트웨어 버전의 설치 디렉토리 이름을 입력하면 Application Server가 시작되지 않습니다.

   b. Javac 필드에서 Java 프로그래밍 언어 컴파일러에 대한 명령줄 옵션을 입력합니다.

      EJB 구성 요소가 배포되면 Application Server에서 컴파일러를 실행합니다.

   c. JPDA(Java Platform Debugger Architecture)를 사용한 디버깅을 설정하려면 디버그 사용 확인란을 선택하고 디버그 옵션 필드에서 옵션을 지정합니다.

      JPDA는 응용 프로그램 개발자가 사용합니다. 자세한 내용은 Application Server Developer's Guide의 Debugging J2EE Applications 장을 참조하십시오. 설명서 링크는 추가 정보를 참조하십시오.

   d. RMI 컴파일 옵션 필드에서 rmic 컴파일러에 대한 명령줄 옵션을 입력합니다.

      EJB 구성 요소가 배포되면 Application Server에서 rmic 컴파일러를 실행합니다.

   e. 바이트 코드 선행 프로세서 필드에서 클래스 이름을 쉼표로 구분해서 입력합니다.

      클래스마다 `com.sun.appserv.BytecodePreprocessor` 인터페이스를 구현해야 합니다. 지정한 순서대로 클래스가 호출됩니다.

      프로필러 같은 도구를 사용하려면 바이트 코드 선행 프로세서 필드의 항목이 필요합니다. 프로필러는 서버 성능을 분석하는 데 필요한 정보를 생성합니다. 프로필링에 대한 자세한 내용은 Application Server Developer's Guide의 Debugging J2EE Applications 장을 참조하십시오.
5. 저장을 누릅니다.

6. 서버를 다시 시작합니다.

JVM 클래스 경로 설정 구성

클래스 경로는 Java 런타임 환경에서 클래스와 다른 자원 파일을 찾는 JAR 파일 목록입니다.

Application Server의 JVM 클래스 경로를 구성하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 서버 노드를 선택합니다.
2. JVM 설정 탭을 누릅니다.
3. 탭 아래 있는 경로 설정 링크를 선택합니다.
4. JVM 클래스 경로 설정 페이지에서 다음을 지정할 수 있습니다.
   a. 환경 클래스 경로 확인란에서 기본 선택을 유지하여 CLASSPATH 환경 변수를 무시합니다.

      프로그래밍의 기본 자습서에서는 CLASSPATH 환경 변수가 편리하지만 엔터프라이즈 환경에는 권장하지 않습니다.

   b. Application Server의 클래스 경로를 확인하려면 서버 클래스 경로 필드의 읽기 전용 내용을 검사합니다.
   c. JAR 파일을 서버의 클래스 경로 첫 부분에 삽입하려면 클래스 경로 접두어 필드에 파일의 전체 경로 이름을 입력합니다.
   d. JAR 파일을 서버의 클래스 경로 끝에 추가하려면 클래스 경로 접미어 필드에 파일의 전체 경로 이름을 입력합니다.

      예를 들어, 데이터베이스 드라이버의 JAR 파일을 지정합니다. JDBC 드라이버 통합를 참조하십시오.

   e. 원시 라이브러리 경로 접두어 및 접미어 필드에서 원시 라이브러리 경로 앞 또는 뒤에 항목을 추가할 수 있습니다.

      원시 라이브러리 경로는 원시 공유 라이브러리에 대한 서버 상대 경로, 표준 JRE 원시 라이브러리 경로, 쉘 환경 설정(UNIX의 LD_LIBRARY_PATH) 및 JVM 프로필러 설정 페이지에 지정되어 있는 경로를 연결한 것입니다.
5. 저장을 누릅니다.
6. 서버를 다시 시작합니다.

JVM 옵션 구성

JVM 옵션 페이지에서 Application Server를 실행하는 Java 응용 프로그램 실행 프로그램(java 도구)의 옵션을 지정할 수 있습니다. -D 옵션은 Application Server에 관련된 등록 정보를 지정합니다.

JVM 옵션을 구성하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 서버 노드를 선택합니다.
2. ORB 탭을 누릅니다.
3. 탭 아래 있는 JVM 옵션을 선택합니다.
4. JVM 옵션 페이지에서 옵션을 수정하려면 값 필드를 편집합니다.
5. 옵션을 추가하려면 다음 작업을 수행합니다.
   a. JVM 옵션 추가를 누릅니다.
   b. 표시된 빈 행에서 값 필드에 정보를 입력합니다.
6. 옵션을 제거하려면 다음 작업을 수행합니다.
   a. 옵션 옆에 있는 확인란을 선택합니다.
   b. 삭제를 누릅니다.
7. 저장을 누릅니다.
8. 서버를 다시 시작합니다.

JVM 옵션에 대한 자세한 내용은 다음을 참조하십시오.

- http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html
- java.sun.com/docs/hotspot/VMOptions.html

보안 관리자 비활성화

Application Server의 보안 관리자를 비활성화하면 일부 유형의 응용 프로그램의 성능이 향상될 수 있습니다. 보안 관리자를 비활성화해도 J2EE 권한 부여 및 인증 기능은 여전히 작동합니다. 개발 환경에서는 보안 관리자를 비활성화하더라도 작업 환경에서는 비활성화하지 마십시오.

보안 관리자를 비활성화하려면 다음 작업을 수행합니다.

1. 관리 콘솔의 JVM 옵션 페이지로 이동합니다.

   자세한 내용은 JVM 옵션 구성을 참조하십시오.
2. JVM 옵션 페이지에서 다음 옵션을 제거합니다.

   `-Djava.security.policy`
3. 저장을 누릅니다.
4. 서버를 다시 시작합니다.

JVM 프로필러 설정 구성

프로필러 도구는 성능을 분석하고 잠재적인 병목 현상을 확인하는 데 사용하는 데이터를 생성합니다.

Application Server에 대한 프로필러 설정을 구성하려면 다음 작업을 수행합니다.

1. 트리 구성 요소에서 응용 프로그램 서버 노드를 선택합니다.
2. JVM 설정 탭을 누릅니다.
3. 탭 아래 있는 프로필러 링크를 선택합니다.
4. JVM 프로필러 설정 페이지에서 지정하는 정보는 사용 중인 프로필러 제품에 따라 다릅니다.

   예와 설명은 Application Server Developer's Guide의 Debugging J2EE Applications 장을 참조하십시오. 설명서 링크는 추가 정보를 참조하십시오.
5. 저장을 누릅니다.
6. 서버를 다시 시작합니다.

고급 설정을 위한 관리 콘솔 작업

- 고급 도메인 속성 설정

고급 도메인 속성 설정

1. 트리 구성 요소에서 Application Server 노드를 선택합니다.
2. 고급 탭을 선택합니다.

3. 도메인 속성 페이지에서 다음 작업을 수행합니다.

   a. 응용 프로그램 루트 필드에서 응용 프로그램이 배포되는 전체 디렉토리 경로를 지정합니다.

   b. 로그 루트 필드에서 서버 인스턴스 로그 파일이 보관되는 장소를 지정합니다.

   c. 대개 호스트의 기본 로켈을 사용하려면 로켈 필드를 공백으로 남겨둡니다.

      로켈은 언어와 지역 조합을 지정하는 식별자입니다. 예를 들어, 미국식 영어의 로켈은 en_US이고 일본어의 로켈은 ja_JP입니다. 비영어 로켈을 사용하려면 Application Server를 현지화해야 합니다. 현지화는 영어를 다른 영어로 번역하는 것을 포함한 프로세스입니다.

4. 저장을 누릅니다.

5. 서버를 다시 시작합니다.

Apache Web Server 컴파일 및 구성

이 부록에서는 Apache 소스 코드를 컴파일하고 Sun Java System Application Server 로드 밸런서 플러그인을 사용하도록 Apache Web Server 설치를 구성하는 방법에 대해 설명합니다.

해당하는 Apache 소스 코드를 다운로드합니다. Sun Java System Application Server에 대해 지원되는 Apache Web Server의 버전과 플랫폼에 대한 자세한 내용은 *Sun Java System Application Server 릴리스 노트*를 참조하십시오.

이 부록에서는 다음 내용을 설명합니다.

- 최소 요구 사항
- SSL 인식 Apache 설치

최소 요구 사항

이 절에서는 Apache Web Server를 컴파일하고 로드 밸런서 플러그인을 실행하는데 필요한 최소 요구 사항을 설명합니다. Apache 소스는 SSL과 함께 실행하도록 컴파일하고 빌드해야 합니다.

이 절에서는 다음 항목에 대해 설명합니다.

- Apache 1.3의 최소 요구 사항
- Apache 2의 최소 요구 사항

Apache 1.3의 최소 요구 사항

Microsoft Windows 플랫폼의 경우 요구 사항은 다음을 참조하십시오.
http://httpd.apache.org/docs/windows_html#req
http://httpd.apache.org/docs/win_compiling.html

기타 플랫폼의 요구 사항:

- openssl-0.9.7d(소스)
- mod_ssl-2.8.16-1.3.29(소스)
- apache_1.3.29(소스)
- gcc-3.3-sol9-sparc-local packages(Solaris 9 SPARC/x86용)
- flex-2.5.4a-sol9-sparc-local packages(Solaris 9 SPARC용)
- flex-2.5.4a-sol9-intel-local packages(Solaris 9 x86용)

Apache를 컴파일하기 전에:

- Linux의 경우 같은 시스템에 Sun Java System Application Server를 설치합니다.
- Solaris 8의 경우 gcc 및 make가 PATH에 있어야 합니다.
- Solaris 9의 경우 gcc 버전 3.3 및 make가 PATH에 있고 flex가 설치되어야 합니다.
- Red Hat Enterprise Linux Advanced Server 2.1에서 gcc를 사용할 경우 gcc 3.0 이후 버전이어야 합니다.

주

- 다른 C 컴파일러를 사용하려면 C 컴파일러의 경로를 설정하고 PATH 환경 변수에서 유틸리티를 만듭니다. 예를 들면 다음과 같습니다.

  exportLD_LIBRARY_PATH=$LD_LIBRARY_PATH:appserver_installdir/lib
- 이 소프트웨어 소스는 http://www.sunfreeware.com에서 사용할 수 있습니다.

Apache 2의 최소 요구 사항

Microsoft Windows 플랫폼의 경우 요구 사항은 다음을 참조하십시오.

http://httpd.apache.org/docs-2.0/platform/windows.html

기타 플랫폼의 요구 사항:

- openssl-0.9.7d(소스)

- `httpd-2.0.49`(소스)
- `gcc-3.3-sol9-sparc-local packages`(Solaris 9 SPARC/x86용)
- `flex-2.5.4a-sol9-sparc-local packages`(Solaris 9 SPARC용)
- `flex-2.5.4a-sol9-intel-local packages`(Solaris 9 x86용)

Apache를 컴파일하기 전에:

- Linux의 경우 같은 시스템에 Sun Java System Application Server를 설치합니다.
- Solaris 8의 경우 `gcc` 및 `make`가 `PATH`에 있어야 합니다.
- Solaris 9의 경우 `gcc` 버전 3.3과 `make`가 `PATH`에 있어야 하고 `flex`가 설치되어야 합니다.
- Red Hat Enterprise Linux Advanced Server 2.1에서 `gcc`를 사용할 경우 gcc 3.0 이후 버전이어야 합니다.

주

- 다른 C 컴파일러를 사용하려면 C 컴파일러의 경로를 설정하고 PATH 환경 변수에서 유틸리티를 작성합니다. 예를 들면 다음과 같습니다.

  ```
  export LD_LIBRARY_PATH=app_server_install_dir/lib:$LD_LIBRARY_PATH
  ```

  이 예는 `sh`용입니다.
- 이 소프트웨어 소스는 http://www.sunfreeware.com에서 사용할 수 있습니다.

SSL 인식 Apache 설치

Microsoft Windows 플랫폼에서 Apache를 컴파일하고 설치하는 데 필요한 지침은 다음 웹 사이트를 참조하십시오.

Apache 1.3:

http://httpd.apache.org/docs/win_compiling.html

Apache 2:

http://httpd.apache.org/docs-2.0/platform/win_compiling.html

다른 플랫폼에서 SSL 인식 Apache Web Server를 컴파일, 구성, 설치하려면 다음 단계를 수행합니다. 예에서는 Apache 1.3.29를 컴파일하고 빌드하는 것을 보여주지만 Apache 2에는 동일한 절차가 적용됩니다.

주 동일한 디렉토리 수준에 `mod_ssl`, OpenSSL 및 Apache의 압축을 풉니다.

OpenSSL 컴파일 및 빌드

Linux와 함께 설치된 OpenSSL 버전이 0.9.7d일 경우 Linux에서는 이 단계가 필요 없습니다.

OpenSSL에 대한 자세한 내용은 다음을 참조하십시오.

http://www.openssl.org/

openssl-0.9.7d 소스의 압축을 풀고 다음 단계를 수행합니다.

1. `cd openssl-0.9.7d`
2. `./config`
3. `make`
4. `make test`
5. `make install`

소스에서 OpenSSL 빌드에 대한 자세한 내용은 openssl 디렉토리의 INSTALL 파일을 참조하십시오.

mod_ssl을 사용하여 Apache 구성

이 절은 Apache 1.3에만 적용됩니다. Apache 2.0이 설치된 경우 401페이지의 "Apache 컴파일 및 빌드"로 건너뜁니다.

mod_ssl에 대한 자세한 내용은 다음을 참조하십시오.

http://www.modssl.org/

1. apache_1.3.29 소스 배포판을 다운로드합니다.

   소스 배포판의 압축을 풉니다. 소스 배포판은 압축되어 있습니다. apache_1.3.29의 경우 소스 배포판의 이름은 apache_1.3.29.tar.gz입니다.

2. 다음 명령을 사용하여 아카이브의 압축을 풉니다.

   `tar-zxvf apache_1.3.29.tar.gz`

   이 명령은 현재 작업 디렉토리에 `apache_1.3.29`라는 디렉토리를 만듭니다.

3. `mod_ssl-2.8.14-1.3.29` 소스의 압축을 풉니다.

4. `cd mod_ssl-2.8.14-1.3.29`

5. `./configure --with-apache=../apache_1.3.29 --with-ssl=../openssl-0.9.7d --prefix=`*`install path`* `--enable-module=ssl --enable-shared=ssl --enable-rule=SHARED_CORE --enable-module=so`를 실행합니다.

위의 명령 예에 지정된 디렉토리는 변수입니다. *prefix* 인수로 Apache를 설치할 위치를 지정합니다. 이 명령을 실행하면 화면에 여러 줄이 표시됩니다.

이 명령은 시스템 구성에 따라 빌드에 대한 `make` 파일을 작성합니다. `configure`의 오류로 인해 일부 헤더 파일이나 유틸리티 프로그램이 누락될 수 있습니다. 계속하려면 이들을 먼저 설치합니다.

Apache 컴파일 및 빌드

Apache 버전에 따라 Apache 컴파일 및 빌드 지침이 다릅니다.

- Apache 1.3 컴파일 및 빌드
- Apache 2 컴파일 및 빌드

Apache 1.3 컴파일 및 빌드

다음 절차는 400페이지의 "mod_ssl을 사용하여 Apache 구성"에서 설명한 `--prefix` 속성에서 제공하는 위치에 Apache를 설치합니다.

1. Linux에서는 `End of automatically generated section` 다음의 `src/MakeFile`에 다음 줄을 포함합니다.

   ```
   LIBS+= -licuuc -licui18n -lnspr4 -lpthread -lxerces-c -lsupport
   -lnsprwrap -lns-httpd40
   ```

   `LDFLAGS+= -L/`*`appserver_installdir`*`/lib`

2. Linux에서는 LD_LIBRARY_PATH에 Application Server 설치 디렉토리를 만듭니다.

   `export LD_LIBRARY_PATH=/`*app_server_install_dir*`/lib:$LD_LIBRARY_PATH`

3. 다음과 같이 `make` 명령을 사용하여 Apache를 컴파일합니다.

   a. `mod_ssl` 디렉토리로 이동합니다.

   b. `make`

   c. `make certificate`

   d. `make install`

주 `make certificate` 명령에서 비밀번호를 요청합니다. Apache의 안전한 시작을 위해 이 비밀번호가 필요하므로 기억하는 것이 좋습니다.

`make install` 명령을 실행하면 프로세스가 Apache 소스 코드를 컴파일하고 Apache를 링크하고 있음을 나타내는 몇 줄의 출력이 화면에 표시됩니다. 이 프로세스는 대개 오류 없이 마무리됩니다. 하지만 오류가 발생하면 Apache의 모든 라이브러리 파일 및 유틸리티 프로그램이 제대로 다운로드되었는지 확인하십시오.

apache_install_path`/conf/httpd.conf` 파일에 사용자 환경에 해당하는 값을 입력하여 Apache 설치를 구성합니다.

Apache 2 컴파일 및 빌드

1. Apache 2_0_*NN* 소스 배포본을 다운로드합니다.

   *NN*은 부버전(예: 52)을 나타냅니다.

2. 소스 배포판의 압축을 풉니다.

   소스 배포판은 압축되어 있습니다. Apache 2_0_*NN*의 경우 소스 배포판 아카이브는 `httpd-2_0_`*NN*`.tar.gz`입니다.

3. 다음 명령을 사용하여 아카이브의 압축을 풉니다.

   `tar -zxvf httpd-2_0_`*NN*`.tar.gz`

   이 명령은 현재 작업 디렉토리에 `httpd-2_0_`*NN*이라는 디렉토리를 작성합니다.

4. `cd httpd-2_0_`*NN*.

5. `./configure --with-ssl=`*open_ssl_install_path* `--prefix=`*install_path* `--enable-ssl --enable-so` 를 실행합니다.
6. Linux에서는 *apache_src*`/build/config_vars.mk`를 수정하고 다음 줄을 추가합니다.

   ```
   EXTRA_LIBS += -licuuc -licui18n -lnspr4 -lpthread -lxerces-c -lsupport
   -lnsprwrap -lns-httpd40
   ```

   ```
   LDFLAGS+=-L<appserver install dir>/lib
   ```

7. Linux에서는 LD_LIBRARY_PATH에 Application Server 설치 디렉토리를 만듭니다.

   ```
   export LD_LIBRARY_PATH=/app_server_install_dir/lib:$LD_LIBRARY_PATH
   ```

8. 다음과 같이 make 명령을 사용하여 Apache를 컴파일합니다.

   `httpd-2_0_`*NN* 디렉토리에서

   a. `make`

   b. `make install`

   `make install` 명령을 실행하면 프로세스가 Apache 소스 코드를 컴파일하고 Apache를 링크하고 있음을 나타내는 몇 줄의 출력이 화면에 표시됩니다. 이 프로세스는 대개 오류 없이 마무리됩니다. 하지만 오류가 발생하면 Apache의 모든 라이브러리 파일 및 유틸리티 프로그램이 제대로 다운로드되었는지 확인하십시오.

   apache_install_path`/conf/httpd.conf` 파일에 사용자 환경에 해당하는 값을 입력하여 Apache 설치를 구성합니다.

주 오류가 발생하면 PATH에 Application Server 설치 디렉토리를 만듭니다.

```
export LD_LIBRARY_PATH=/app_server_install_dir/lib
```

또는 예를 들어, OpenSSL 라이브러리를 추가합니다.

```
export
LD_LIBRARY_PATH=/openssl_install_dir/lib:/app_server_install_dir/l
ib
```

주 Apache 2에서는 인증서를 수동으로 작성 및 설치해야 합니다. 자세한 내용은 Apache 설명서를 참조하십시오.

Apache 시작 및 중지

Apache에는 Apache를 시작, 중지 및 다시 시작하는 `apachectl`이라는 스크립트가 번들되어 있습니다.

- Apache를 시작하려면 다음 명령을 실행합니다.

  apache_install_dir`/bin/apachectl start`

- SSL 모드에서 Apache를 시작하려면 다음 명령을 실행합니다.

  apache_install_dir`/bin/apachectl startssl`

- Apache를 중지하려면 다음 명령을 실행합니다.

  apache_install_dir`/bin/apachectl stop`

Apache를 시작한 후 설치를 테스트합니다. Apache가 실행되면 웹 브라우저에 주소 http://*server_name*:*port_number*/를 입력합니다. 설치가 성공하고 Apache가 실행되면 테스트 페이지가 표시됩니다.

Apache 설치를 완료했으면 플러그인 설치 중 또는 설치 후 Apache 구성에 대한 내용은 69페이지의 "Apache Web Server 수정 사항"을 참조하십시오.

도메인 또는 노드 에이전트 자동 재시작

도메인이나 노드 에이전트가 예기치 않게 정지된 경우(예: 시스템을 다시 시작해야 할 경우) 도메인이나 노드 에이전트를 자동으로 다시 시작하도록 시스템을 구성할 수 있습니다.

이 부록은 다음 내용으로 구성되어 있습니다.

UNIX 플랫폼에서 자동 재시작

UNIX 플랫폼에서 도메인을 다시 시작하려면 `/etc/inittab` 파일에 텍스트 줄을 추가하십시오.

예를 들어, `opt/SUNWappserver` 디렉토리에 설치된 Application Server의 경우 `password.txt`라고 하는 비밀번호 파일을 사용하여 domain1을 다시 시작하려면 다음을 입력합니다.

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user admin
--passwordfile /opt/SUNWappserver/password.txt domain1
```

텍스트를 한 줄로 입력합니다. 처음 세 문자는 프로세스의 고유한 지정자이며 변경할 수 있습니다.

노드 에이전트를 다시 시작하기 위한 구문도 유사합니다. 예를 들어, `opt/SUNWappserver` 디렉토리에 설치된 Application Server의 경우 `password.txt`라고 하는 비밀번호 파일을 사용하여 agent1을 다시 시작하려면 다음을 입력합니다.

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-node-agent --user admin
--passwordfile /opt/SUNWappserver/password.txt agent1
```

Microsoft Windows 플랫폼에서 자동 재시작

Microsoft Windows에서 자동으로 다시 시작하려면 Windows Service를 작성합니다. Microsoft에서 제공하는 서비스 제어 명령(sc.exe)과 함께 Sun Java System Application Server에 포함된 appservService.exe 및 appserverAgentService.exe 실행 파일을 사용합니다.

sc.exe 명령은 Windows XP와 함께 제공되며 C:\windows\system32 디렉토리나 C:\winnt\system32 디렉토리에 있습니다. 이 설명서를 작성할 시점에는 Windows 2000 sc.exe를 ftp://ftp.microsoft.com/reskit/win2000/sc.zip에서 다운로드할 수 있습니다. sc.exe 사용에 대한 자세한 내용은 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndllpro/html/msdn_scmslite.asp를 참조하십시오.

다음과 같이 appservService.exe 및 appservAgentService.exe를 사용합니다.

```
C:\winnt\system32\sc.exe create service_name binPath=
\"fully_qualified_path_to_appservService.exe  \"fully_qualified_path_to_asadmin.bat start_command\"
\"fully_qualified_path_to_asadmin.bat stop_command\"" start= auto DisplayName=
"display_name"
```

예를 들어, C:\Sun\AppServer\password.txt 비밀번호 파일을 사용하여 domain1 도메인을 시작 및 중지하는 SunJavaSystemAppServer DOMAIN1이라고 하는 서비스를 작성하려면 다음을 입력합니다.

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe \"C:\Sun\AppServer\bin\asadmin.bat
start-domain --user admin --passwordfile C:\Sun\AppServer\password.txt
domain1\" \"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\"" start=
auto DisplayName= "SunJavaSystemAppServer DOMAIN1"
```

노드 에이전트 agent1을 시작 및 중지하는 서비스를 작성하려면 다음을 입력합니다.

```
C:\windows\system32\sc.exe create agent1 binPath=
"C:\Sun\AppServer\lib\appservAgentService.exe
\"C:\Sun\AppServer\bin\asadmin.bat start-node-agent --user admin
--passwordfile C:\Sun\AppServer\password.txt agent1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-node-agent agent1\"" start= auto
DisplayName= "SJESAS_SE8.1 AGENT1"
```

주 binPath= 매개 변수의 일부로 입력한 시작 및 중지 명령의 구문이 정확해야 합니다. 테스트하려면 명령 프롬프트에서 명령을 실행합니다. 명령을 실행해도 도메인이나 노드 에이전트가 제대로 시작되거나 중지되지 않을 경우 서비스가 올바로 작동하지 않는 것입니다.

주 asadmin 시작 및 중지 명령과 서비스 시작 및 중지를 같이 사용하지 마십시오. 둘을 같이 사용하면 서버 상태가 비동기화될 수 있습니다. 예를 들어, 구성 요소가 실행되지 않아도 서비스에서 구성 요소가 실행되었다고 표시할 수 있습니다. 이런 상황을 방지하려면 서비스를 사용할 때 항상 sc.exe 명령을 사용하여 구성 요소를 시작 및 중지하십시오.

자동 재시작 보안

시작할 때 필요한 비밀번호와 마스터 비밀번호를 다음과 같은 방법 중 하나로 처리합니다.

- Microsoft Windows에서는 사용자에게 비밀번호를 요청하도록 서비스를 구성합니다.
  a. 서비스 제어판에서 작성한 서비스를 두 번 누릅니다.
  b. 등록 정보 창에서 로그온 탭을 누릅니다.
  c. 구성 요소를 시작할 때 필요한 비밀번호를 묻는 프롬프트를 표시하려면 "서비스와 데스크탑 상호 작용 허용"을 선택합니다.

  로그인 해야 프롬프트가 나타납니다. 입력할 때 입력한 내용이 표시되지 않습니다. 이 방법이 서비스 옵션을 사용하는 가장 안전한 방법이지만 사용자 입력이 있어야 서비스를 사용할 수 있습니다.

  "데스크탑과 상호 작용" 옵션을 설정하지 않은 경우 서비스가 "시작-보류 중" 상태를 유지하고 중단된 것으로 표시됩니다. 서비스 프로세스를 중단하고 이 상태에서 복구합니다.
- Windows나 UNIX에서 --savemasterpassword=true 옵션을 사용하여 도메인을 작성하고 비밀번호 파일을 작성하여 관리자 비밀번호를 저장합니다. 구성 요소를 시작할 때 --passwordfile 옵션을 사용하여 비밀번호를 포함하는 파일을 가리킵니다.

  예를 들면 다음과 같습니다.

a. 저장한 마스터 비밀번호를 사용하여 도메인을 작성합니다. 이 구문에서는 관리자 비밀번호와 마스터 비밀번호를 묻습니다.

```
asadmin create-domain --adminport 4848 --adminuser admin
--savemasterpassword=true --instanceport 8080 domain1
```

b. Windows에서는 비밀번호 파일을 사용하여 서비스를 작성하여 관리자 비밀번호를 채웁니다.

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe
\"C:\Sun\AppServer\bin\asadmin.bat start-domain --user admin
--passwordfile C:\Sun\AppServer\password.txt domain1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\"" start=
auto DisplayName= "SJESAS_PE8.1 DOMAIN1"
```

비밀번호 파일 `password.txt`의 경로는 `C:\Sun\AppServer\password.txt`입니다. 이 파일에는 다음 형식의 비밀번호가 포함되어 있습니다.

`AS_ADMIN_password=`*`password`*

예를 들어, `adminadmin` 비밀번호의 경우 다음과 같습니다.

```
AS_ADMIN_password=adminadmin
```

c. UNIX에서는 `inittab` 파일에 추가한 줄에서 `--passwordfile` 옵션을 사용합니다.

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user
admin --passwordfile /opt/SUNWappserver/password.txt domain1
```

비밀번호 파일 `password.txt`의 경로는 `/opt/SUNWappserver/password.txt`입니다. 이 파일에는 다음 형식의 비밀번호가 포함되어 있습니다.

`AS_ADMIN_password=`*`password`*

예를 들어, `adminadmin` 비밀번호의 경우 다음과 같습니다.

```
AS_ADMIN_password=adminadmin
```

domain.xml의 점으로 구분된 이름 속성

이 부록에서는 Mbean과 그 속성을 지정하는 데 사용할 수 있는 점으로 구분된 이름 속성에 대해 설명합니다. `domain.xml` 파일의 모든 요소에는 해당하는 Mbean이 있습니다. 이 이름을 사용하는 구문에서는 점으로 이름을 구분해야 하기 때문에 이 이름을 점으로 구분된 이름이라고 합니다.

이 부록은 다음 내용으로 구성되어 있습니다.

최상위 수준 요소

domain.xml 파일의 모든 최상위 수준 요소에 대해서 다음 조건을 준수해야 합니다.

1. 모든 서버, 구성, 클러스터 또는 노드 에이전트에는 고유한 이름이 있어야 합니다.
2. 서버, 구성, 클러스터 또는 노드 에이전트 이름을 "domain"으로 지정할 수 없습니다.
3. 서버 인스턴스 이름을 "agent"로 지정할 수 있습니다.

다음 표에서는 최상위 수준 요소와 해당하는 점으로 구분된 이름 접두어를 설명합니다.

표 C-1 최상위 수준 요소

요소 이름	점으로 구분된 이름 접두어
applications	domain.applications
resources	domain.resources

표 C-1 최상위 수준 요소

요소 이름	점으로 구분된 이름 접두어
configurations	domain.configs
servers	domain.servers 이 요소에 포함된 모든 서버는 *server-name*으로 액세스할 수 있습니다. 여기에서 *server-name*은 서버 하위 요소에 대한 이름 속성 값입니다.
clusters	domain.clusters 이 요소에 포함된 모든 클러스터는 *cluster-name*으로 액세스할 수 있습니다. 여기에서 *cluster-name*은 클러스터 하위 요소의 이름 속성 값입니다.
node-agents	domain.note-agents
lb-configs	domain.lb-configs
system-property	domain.system-property

두 가지 수준의 별칭을 사용할 수 있습니다.

1. 첫 번째 수준의 별칭을 사용하면 domain.servers 또는 domain.clusters 접두어를 검색하지 않고 서버 인스턴스나 클러스터의 속성에 액세스할 수 있습니다. 따라서, "server1"처럼 점으로 구분된 이름은 점으로 구분된 이름 domain.servers.server1(여기에서 server1은 서버 인스턴스)에 연결됩니다.
2. 두 번째 수준의 별칭을 사용하여 클러스터나 독립 실행형 서버 인스턴스(대상)의 구성, 응용 프로그램 및 자원을 참조합니다.

다음 표에서는 도메인의 최상위 수준 이름에 대한 별칭인 서버 이름이나 클러스터 이름으로 시작하는 점으로 구분된 이름을 설명합니다.

표 C-2 도메인의 점으로 구분된 이름 서버 이름

점으로 구분된 이름	별칭	설명
target.applications.*	domain.applications.*	별칭은 *target*에서만 참조하는 응용 프로그램으로 변환됩니다.
target.resources.*	domain.resources.*	별칭은 모든 `jdbc-connection-pool`, `connector-connection-pool`, `resource-adapter-config` 및 *target*에서 참조하는 다른 모든 자원으로 변환됩니다.

다음 표에서는 서버나 클러스터에서 참조하는 구성 내의 최상위 수준 이름으로 별칭 지정된 서버 이름이나 클러스터 이름으로 시작하는 점으로 구분된 이름을 설명합니다.

표 C-3 서버나 클러스터에서 참조한 구성에 대한 점으로 구분된 이름

점으로 구분된 이름	별칭
target.http-service	*config-name*.http-service
target.iiop-service	*config-name*.iiop-service
target.admin-service	*config-name*.admin-service
target.web-container	*config-name*.web-container
target.ejb-container	*config-name*.ejb-container
target.mdb-container	*config-name*.mdb-container
target.jms-service	*config-name*.jms-service
target.log-service	*config-name*.log-service
target.security-service	*config-name*.security-service
target.transaction-service	*config-name*.transaction-service
target.monitoring-service	*config-name*.monitoring-service
target.java-config	*config-name*.java-config
target.availability-service	*config-name*.availability-service
target.thread-pools	*config-name*.thread-pools

별칭이 지정되지 않은 요소

클러스터링된 인스턴스는 별칭을 지정하지 말아야 합니다. 클러스터링된 인스턴스의 시스템 등록 정보를 가져오려면 점으로 구분된 이름 속성에서 clustered-instance-name.system-property가 아니라 domain.servers.clustered-instance-name.system-property를 사용해야 합니다.

색인

가

나

다

라

마

바

사

아

자

카

타

파

하

A

B

C

D

E

G

H

I

J

K

L

M

N

O

P

R

S

T

W

W