JENNIFER®
Enterprise Web Business Monitoring

제니퍼 매뉴얼

- 엔터프라이즈 애플리케이션 전문 성능 모니터링
- 더욱 새로워진 사용자 중심의 직관적 대시보드
- 전거래 응답분포도와 모니터링 확장
- 다이나믹 프로파일링/스텍트레이스
- 도메인별 통합모니터링

제니퍼 **4.5** 매뉴얼

JS-0810-01

제니퍼 4.5 매뉴얼

지은이 **김성조 정성태**
펴낸이 **(주) 제니퍼소프트**
엮은이 **송연주**
초판발행일 **2008 년 10 월 28 일**
문서 업데이트 일 **2010 년 11 월 28 일**
문서번호 **JS-0810-01**
문서버전 **v4**
관련문서 **제니퍼 v4 설치가이드**

Table of Contents

저작권안내

Trademarks

본문 중에 인용한 하기의 JENNIFER™와 JENNIFERSOFT®는 당사의 고유 등록상표이며 특허법과 저작권법 등에 의해 보호를 받습니다. 당가의 허가 없이 무단으로 해당 상표를 사용 배하거나 배포하는 자는 법의 처벌을 받습니다.

JENNIFERSOFT® JENNIFER™

각 회사의 제품명을 포함하여 아래 명시된 상표는 각 개발사의 등록상표이며 특허법과 저작권법 등에 의해 보호를 받고 있습니다. 따라서 기타 모든 제품들과 회사 이름은 각각 해당 소유주의 상표로서 참조용으로만 사용됩니다. 다음 목록에 나와 있지 않은 이름이나 로고의 경우에도 이 해당 제품, 기능이나 서비스 이름 또는 로고에 지정한 모든 지적 재산권의 적용을 받습니다.

Sun, Sun Microsystems, Solaris, Java, JavaEE, EJB, GlassFish 및 JMX, JNI, JSP, JDBC, JNDI, JVM, 등은 Sun Microsystems, Inc.의 한국 및 다른 국가에서의 등록 상표 또는 상표입니다.

IBM, WebSphere, AIX, iSeries, z/OS, iSeries, DB2, CICS, CTG 등은 IBM Corporation의 한국 및 다른 국가에서의 등록 상표 또는 상표입니다 .

BEA, WebLogic, WLI, ALSB, TUXEDO, Jolt, WTC 등은 Oracle, Inc 의 한국 및 다른 국가에서의 등록 상표 또는 상표입니다 .

Microsoft, Window 2000, Vista, XP, NT 등은 Microsoft Corporation. 의 한국 및 다른 국가에서의 등록 상표 또는 상표입니다 .

Fujitsu® Interstage®
Hitachi® Cosminexus®
Sybase® EAServer®
Macromedia® JRun®
Tmaxsoft® JEUS® WebT®
Apache® Apache Derby® Jakarta Tomcat® DBCP®
Caucho Technology® Resin®
RedHat® JBoss®
Apache® JServ®
HP-UX® Itanium®
Sun Solaris®
Linux®
Compaq® True64®
Unipoint® J*Link®

서문

본 문서는 제니퍼(JENNIFER) v4설치 전 혹은 설치 후 실 운영 관리 시에 알고 있어야 할 제니퍼 서버 및 에이전트의 아키텍처, 구성방법, 사용자 인터페이스, 성능 데이터 수집방법, 분석방법 등에 대한 방대하고도 상세한 내용을 포괄적으로 제공한다.

본 문서의 주 대상자는 제니퍼소프트 협력사의 기술지원 엔지니어, 제니퍼를 지속적으로 운영하는 제니퍼 고객 운영자, 애플리케이션 개발 및 테스트 과정에서 제니퍼를 통한 애플리케이션 튜닝을 시도하는 애플리케이션 개발자, 제니퍼에 대한 상세한 기능을 세세하게 이해하고자 하는 자를 대상으로 하고 있다.

애플리케이션성능관리(APM) 솔루션인 제니퍼(JENNIFER)는 특별히 Java EE/WAS의 성능 관리를 위한 WAS운영자, 애플리케이션 성능 향상을 위한 애플리케이션 개발자, 용량 산정 및 장애진단을 목적으로 하는 성능튜닝 담당자, 애플리케이션 내부의 SQL쿼리 병목 추출 및 데이터베이스 성능 튜닝 담당자에게 특별한 가치와 효과를 전달할 것으로 믿고 있다.

본 문서를 읽기 위한 사전 지식으로는 자바가상머신(JVM, Java Virtual Machine)에 대한 기본적인 환경 이해와 자바 애플리케이션 프로그래밍 개발 및 소스코드에 대한 분석능력, 그리고 웹애플리케이션서버(WAS)에 대한 설치 및 구성, 구동에 대한 충분한 지식을 필요로 하고 있다. 또한, 시스템간 통신을 위한 TCP/IP 및 UDP 통신 네트워크 프로토콜에 대한 기본적인 지식과 시스템의 TCP/UDP 포트(Port) 설정에 대한 최소한의 이해를 필요로 한다.

그리고, WAS와 데이터베이스 간의 JDBC 연동에 대한 개념 및 자체 시스템에 적용되어 있는 구체적인 JDBC연결 구성에 대한 충분한 정보를 알고 있어야 한다. 데이터베이스 SQL 쿼리 작성에 대한 기본적인 지식도 필요하다.

또한, 웹 기반 시스템 하에서 운영되는 웹 서버, WAS서버, DB서버간의 아키텍처 구성에 대한 일반적인 이해와 방문자, 동시단말사용자, 부하량(Throughput), 응답시간 등 성능분석 용어에 대한 기초적인 이해를 필요로 하고 있다.

운영 체계에 따라서 사용해야 하는 스크립트 파일 확장자가 다르다. 유닉스와 리눅스는 확장자가 sh이고 마이크로소프트 윈도우즈는 확장자가 bat이다. 본 매뉴얼은 유닉스와 리눅스를 기준으로 작성되어 있다. 따라서 마이크로소프트 윈도우즈를 사용하는 경우에는 확장자가 sh인 스크립트 파일 대신에 확장자가 bat인 스크립트 파일을 사용하도록 한다.

본 문서에서는 제니퍼 설치에 대한 내용은 생략되어 있으며, 설치와 관련한 구체적인 사항에 대해서는 별도의 제니퍼 설치가이드를 참조토록 권고한다.

본 문서는 제니퍼 v4에 대한 매뉴얼로서의 문서이기에, 가급적 옵션을 포함한 모든 내용을 언급하고자 하였다. 반면 응용적인 활용가이드나 사례 중심적인 활용가이드의 내용은 제니퍼소프트의 웹사이트를 통해 별도의 활용가이드나 테크노트 기술문서를 참조토록 권고한다.

마지막으로, 제니퍼 v4 및 매뉴얼이 시장에 나오기까지 애정 어린 시각으로 제니퍼소프트와 제니퍼 제품에 대해 긍정적인 피드백을 보내주고 있는 고객 담당자와 불철주야 기술지원에 매진하고 있는 제니퍼소프트 협력사 엔지니어들에게 진심으로 감사함을 전한다.

저자소개

김성조(SungJo Kim) 제니퍼소프트 R&D센터 소장/기술이사

자바 기반의 소프트웨어 아키텍처와 성능 문제를 연구하는 성능분석 전문 컨설턴트이자 개발자이다. LGCNS에서 소프트웨어 아키텍트로서 아키텍처 강의 및 소프트웨어 개발방법론을 연구하였으며, 수 십여 곳의 프로젝트에서 자바애플리케이션/WAS 성능 튜닝을 진행하였다. 2005년도에 제니퍼소프트 R&D 센터에 합류하여 현재는 제니퍼의 핵심적인 코어(Core) 엔진 설계 및 로우레벨 모듈 개발을 주도하고 있다. 특히 최초로 응답시간분포도(X-View) 기반의 트랜잭션별 프로파일링 개념을 설파하고 제품에 접목하였다.

정성태(Kevin Jung) 제니퍼소프트 R&D 닷넷 개발부 차장

마이크로소프트 기술 분야에서 일해 온 경험을 바탕으로 현재 제니퍼소프트에서 자바 버전의 성능관리 솔루션인 "제니퍼"의 닷넷 버전 개발 및 그에 기반한 닷넷 서비스의 성능 관련한 기술 문서를 담당하고 있다.

전지훈(Albert Jeon) 제니퍼소프트 R&D 연구개발부 과장

TmaxCore R&D 브라우져 팀에서 UI와 네트워크 부분 등을 담당 하였으며, 2010년부터 제니퍼소프트의 R&D 연구개발부에 합류하여 주로 제니퍼 소프트에서 사용자 인터페이스(UI) 관련 내용을 담당 하고 있다. 다방면의 책 읽기를 좋아 하며 최근 테스트에 많은 관심을 가지고 있다. 기본의 중요성을 깨닫고 처음 부터 정리 반복하는 과정을 즐기고 있다. 행복한 개발자.

칼리드(Khalid Saeed) 제니퍼소프트 R&D 연구개발부

충북대에서 전자공학을 석사 과정을 마치고 2010년부터 제니퍼소프트 R&D 연구소에 입사하였다. 자바 프로그래밍 개발자로 제니퍼에 관한 각종 테스트 및 테크니컬 문서 라이팅을 담당하고 있다. 그 외에도 제니퍼 옵션에 대한 정리 및 기술된 내용에 대한 검증 및 교정, 테스트에 지원을 아끼지 않은 제니퍼소프트 기술지원부의 박노민(Nomin Park)차장과 김지훈(Jihun Kim)과장의 기여가 매우 컸다.

또한, 매뉴얼 집필 및 편집에 있어서 코디네이터 역할을 충실하게 진행해 준 제니퍼소프트 마케팅부 송연주(Amie Song) 대리의 역할도 중요했다.

독자후기

본 문서에 기술된 내용 중 기술적인 오류나, 구문오류, 혹은 내용적으로 수정하거나 추가하기를 원하는 것이 있다면 아래의 이메일 주소나 우편으로 보내면 된다.

이메일: manual@jennifersoft.com, 전화: 02)2027-0391, 팩스:02)2027-0390

주소: 우153-777 서울 금천구 가산동 60-11 스타밸리 1104호 ㈜제니퍼소프트 R&D센터

문서 구성

본 문서는 개념적 이해를 단계적으로 돕기 위해 순차적인 구성으로 되어 있다. 그러나, 관련 주제별로 필요에 따라 읽힐 수 있도록 가급적 독립적으로 구성되어 있다. 본 문서의 구성은 다음과 같다.

1장, "시작하기" 제니퍼의 개념과 효과 그리고 제니퍼 기능 요약에 대해 소개한다.

2장, "제니퍼 아키텍처" 제니퍼의 모듈 구성과 수집데이터에 대한 요약 그리고 네트워크 구성에 대해 설명한다.

3장, "제니퍼 자바 에이전트 운영관리" 모니터링 대상 시스템에 제니퍼 자바 에이전트를 설치하는 방법과 운영 설정에 대해 설명한다.

4장, "제니퍼 닷넷 에이전트 운영관리" 모니터링 대상 시스템에 제니퍼 닷넷 에이전트를 설치하는 방법과 운영 설정에 대해 설명한다.

5장, "사용자 인터페이스" 제니퍼 클라이이언트는 웹브라우저에서 동작한다. 브라우저 환경에 대한 가이드가 포함되어 있으며 제니퍼 클라이언트의 화면 구성과 조작을 위한 기능에 대해 설명한다.

6장, "서비스 및 사용자 모니터링" 애플리케이션 서비스 모니터링을 위한 설정, 부하량 제어, 실시간 액티브 서비스 모니터링, 서비스 덤프 방법등 WAS/NON-WAS 애플리케이션 모니터링 방법을 설명하고 있으며, WAS 애플리케이션에서의 사용자(방문자, 동시단말사용자 등)의 개념과 모니터링에 대해 설명한다.

7장, "차트와 대시보드" 제니퍼 클라이언트에서 사용되는 각종 차트의 특성과 이들을 이용한 대시보드 구성 방법에 대해 설명한다.

8장, "X-View와 프로파일링" 응답시간 분포도의 개념및 사례, 서비스 프로파일링과 X-View 데이터 분석 방법에 대해 설명한다. 특히 다이나믹 프로파일링, 다이나믹 스택트레이스에 대한 설명이 포함되어 있다.

9장, "자바 시스템에서 리소스 및 연계,JDBC모니터링" CPU, 메모리, 파일, 소켓 등의 기본적인 시스템 자원에 대한 모니터링을 설명하고 메모리 문제 해결을 위한 힙 메모리 관련 모니터링 방법에 대해 설명한다. 그리고 엔터프라이즈 시스템에서 중요한 JDBC/SQL 모니터링 설정과 주의 사항에 대해 설명한다.

10장, "닷넷 시스템에서 리소스 및 연계, JDBC모니터링" CPU, 메모리, 파일, 소켓 등의 기본적인 시스템 자원에 대한 모니터링을 설명하고 메모리 문제 해결을 위한 힙 메모리 관련 모니터링 방법에 대해 설명한다. 그리고 엔터프라이즈 시스템에서 중요한 DB/SQL 모니터링 설정과 주의 사항에 대해 설명한다.

11장, "경보와 예외 모니터링" 제니퍼가 애플리케이션 모니터링에서 수집하는 예외들에 대해 설명한다. 또한 애플리케이션과 서버를 모니터링 하면서 발생하는 경보 종류 및 타시스템으로 해당 경보 데이터를 전송하는 방법에 대해 설명한다.

12장, "제니퍼 서버 운영 관리" 대규모 시스템을 위한 제니퍼 구성, 제니퍼 DB의 테이블 스키마, 사용자 권한 등 서버 운영에 필요한 기능에 대해 설명한다.

13장, "성능 현황과 보고서" 실시간 모니터링, 통계 분석 그리고 제니퍼 보고서에 대해 설명한다.

14장, "고급 모니터링" 제니퍼 확장 어뎁터 종류와 관련 API를 소개한다. 또한 타시스템과 성능데이터 연동 방안에 대해 설명한다. 이것들은 제니퍼의 기능을 확장하거나 커스터마이징하기 위해 사용될 수 있다.

15장, "REMON" 쉘이나 클래스 프로그램으로 작성된 어뎁터로부터 별도의 성능데이터를 수집하기 위한 REMON 사용 방법에 대해 설명한다. 또한 제니퍼4에 포함된 실시간 로그 감시기에 대해 설명한다.

마지막으로 APPENDIX에는 에이전트 및 서버 설정 옵션과 제니퍼 내부 에러코드가 정리되어 있다.

일러두기

하단에 정리된 표기법과 용어는 본 문서에서 주로 사용되는 것들이다. 매뉴얼을 읽기 전 다음 내용을 미리 숙지하고 문서를 참조하면 쉽게 이해 할 수 있다.

표기	설명
굴림	본문 내용의 기본 폰트, Notice
Arial (Bold)	본문의 영문체, 메소드/클래스/파일/디렉토리 명, 고유명사
Body Box Courier	본문에 쓰인 코드 예) `http_service_class = mysys.AServlet;mysys.BServlet`
Courier Bold	코드강조 예) **`http_service_class = mysys.AServlet;mysys.BServlet`**
[메뉴명]-Bold	제니퍼 대시보드 메뉴 예) **[대시보드 | 제니퍼 대시보드]**
[상위 | 하위]	상위/하위 메뉴 구분
[구분자]	구분자 예) [,]
Notice/Warning	참고/주의 사항을 의미하며 박스로 표현 됨
Reference	참조 예) [트랜잭션 데이터(32 페이지)]
옵션명	**config_refresh_check_interval**

1

시작하기

1.1. 애플리케이션 성능 관리

애플리케이션 성능 관리(Application Performance Management, 이하 APM)는 애플리케이션 서비스에 대한 효율적인 성능 모니터링 및 성능 장애 대응 전략을 수립하고 미래 예측을 가늠하는 일련의 지속적인 성능 관리 체계를 구축하는 것이다.

APM은 System Management System(이하 SMS)이나 Network Management System(이하 NMS)과 같은 전통적인 시스템 관리 솔루션과는 달리 실제 서비스되고 있는 애플리케이션의 서비스 관점의 성능적 현황을 파악할 수 있고, 내부 유지보수 관점에서의 장애 대응 및 분석 역량을 강화시킨다. 따라서 APM을 도입하면 보다 지능적인 방법으로 대 고객 서비스의 안정화를 이루어 궁극적으로 총소유 비용(Total Cost Ownership, 이하 TCO)을 효과적으로 절감할수 있다.

1.2. 제니퍼(JENNIFER™)란

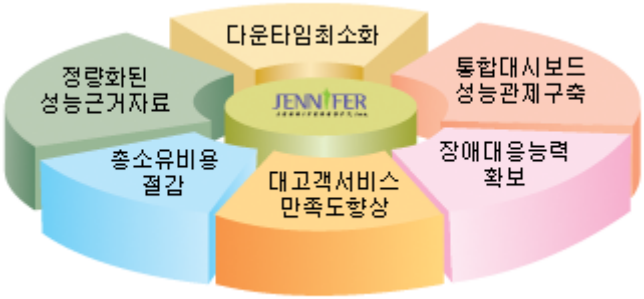
(주)제니퍼소프트의 제니퍼(JENNIFER)는 웹 어플리케이션 서버(Web Application Server, 이하 WAS)에서 동작하는 엔터프라이즈 자바 애플리케이션의 성능 관리를 위한 전문적인 APM 솔루션이다.

Notice: WAS에 기반하지 않고 독립적으로 동작하는 자바 애플리케이션도 제니퍼를 통해서 성능 모니터링을 할 수 있다.

1.3. 제니퍼 도입 효과

제니퍼 도입과 사용을 통해서 얻을 수 있는 주요 효과는 다음과 같다.

그림 1-1: 제니퍼 도입 효과



1.3.1. 다운 타임 최소화

제니퍼를 도입하면 성능 장애 현상 발생시 즉각적인 원인 분석과 신속한 해결을 통해 다운 타임을 최소화하여 시스템을 보다 안정적으로 운영할 수 있다.

1.3.2. 통합 대시보드 성능 관제 구축

제니퍼를 도입하면 서비스 관점에서의 성능 모니터링 시스템을 손쉽게 구축하여 서비스 현황을 실시간으로 통합 관제할 수 있다. 제니퍼의 실시간 통합 대시보드는 강력하고 다이나믹하며 즉시적인 성능 모니터링 인터페이스를 제공한다.

1.3.3. 장애 대응 능력 확보

제니퍼를 도입하면 지속적인 시스템의 성능 모니터링을 통해 향후 발생할 가능성이 있는 위험을 예측하고 미연에 방지할 수 있다. 또한 제니퍼는 문제 발생시 자동 경보 기능을 제공한다.

1.3.4. 대고객 서비스 만족도 향상

제니퍼를 도입하면 자동 부하량 제어(PLC) 기능을 통한 서비스 안정화, 다운 타임의 최소화 및 장애 대응 능력의 확보로 보다 안정적이고 신뢰성 높은 24 시간 x 365 일 시스템 운영이 가능하게 되어, 궁극적으로 대고객 만족도를 향상시킨다.

1.3.5. 총소유 비용 절감

제니퍼를 도입하면 다운 타임 최소화, 장애 대응 능력 확보, 애플리케이션 튜닝을 통한 성능 최적화, IT 자원 효율성의 극대화와 효율적인 인적 자원 활용으로 TCO를 절감할 수 있다.

1.3.6. 정량화된 성능 근거 자료 확보

제니퍼를 도입하면 접속자와 부하량 등에 대한 정량화된 데이터 수집 및 통계 분석을 근거로 시스템 확장 및 개편 시점을 예측하고 증설과 튜닝 작업에 필요한 정량화된 근거 자료를 확보할 수 있다.

1.4. 제니퍼 기능

제니퍼가 제공하는 주요 기능은 다음과 같다.

1.4.1. 서비스 모니터링

서비스 모니터링은 애플리케이션의 사용자와 실시간 서비스 처리 현황과 관련된 성능 데이터를 수집하는 것이다.

제니퍼가 제공하는 서비스 모니터링 항목은 다음과 같다.

- 실시간 접속 중인 동시단말 사용자
- 실시간 수행 중인 액티브 서비스
- 실시간 업무 처리량
- 실시간 트랜잭션 응답 시간 분포도
- 애플리케이션 이름 동적 치환 기능
- 외부 트랜잭션과 바인딩 변수를 포함한 SQL 추적
- HTTP POST 호출 파라미터 키와 값 추적

1.4.2. 리소스 모니터링

리소스 모니터링은 애플리케이션이 서비스를 수행하는 과정에서 사용하는 CPU와 메모리와 같은 논리적 혹은 물리적 자원과 관련된 성능 데이터를 수집하는 것이다.

제니퍼가 제공하는 리소스 모니터링 항목은 다음과 같다.

- DB 커넥션 및 커넥션 풀 상태
- 시스템 및 프로세스 CPU 사용률
- 시스템 및 프로세스 메모리 사용량
- 프로세스 힙 메모리 사용량
- 파일 IO 상태
- TCP 소켓 IO 추적
- 콜렉션 객체와 자바 라이브 오브젝트
- JMX(Java Management Extension)를 활용한 WAS 내부 리소스 상태

1.4.3. 장애 진단 및 분석

웹 애플리케이션의 성능과 관련된 장애 유형은 부하량 증가에 의한 상대적 성능 장애와 비정상적인 특정 조건 상황에서 발생하는 조건적 성능 장애로 구분된다. 제니퍼는 이런 성 장애를 진단하고 분석하기 위해 다음과 같은 기능을 제공한다.

- 자바 힙 메모리 누수 추적(콜렉션 객체 및 라이브 오브젝트)
- 미반환 DB Connection 추적
- 미반환 JDBC Statement/ResultSet 객체 추적

- 미처리 JDBC 트랜잭션(commit/rollback) 추적
- 애플리케이션에서 발생한 예외(Exception) 추적
- 애플리케이션에서 발생한 SQL 예외(SQLExceptoin) 추적
- 액티브 서비스를 처리하는 쓰레드에 대한 덤프
- HTTP 세션에 대한 덤프
- 로딩된 클래스 조회

1.4.4. 실시간 액티브 서비스 모니터링

제니퍼는 자바 애플리케이션이 현재 처리중인 트랜잭션을 액티브 서비스라고 한다. 그리고 이 액티브 서비스 목록을 경과 시간대 별로 나누어서 실시간 이퀄라이저 차트와 액티브 서비스 목록으로 제공한다. 이를 통해 사용자는 현 시점에서의 자바 애플리케이션 내부의 액티브 서비스에 대한 스냅 샷 정보를 확인할 수 있다.

Notice: 제니퍼가 제공하는 액티브 서비스는 지난 몇 초간의 통계 정보가 아니라 현 시점에서의 실시간 트랜잭션 처리 정보를 제공한다.

1.4.5. X-VIEW와 트랜잭션 프로파일링

응답 시간 분포도는 모든 트랜잭션을 트랜잭션의 시작 시간과 응답 시간을 기준으로 차트에 점으로 표현한 것이다. 제니퍼에서는 이를 X-View로 지칭한다.

전체 트랜잭션 처리 현황을 X-View로 모니터링하면 응답 시간의 지연을 야기시킨 특정 트랜잭션 혹은 서로 연관 관계를 갖는 트랜잭션 그룹을 파악하여 성능 저하 현상을 보다 효과적으로 분석할 수 있다.

Notice: 응답 시간 분포도를 통해서 응답 시간이 느려졌다는 것 외에 응답 시간을 느려지게 만드는 미묘한 병목 유형의 패턴을 감지할 수 있다. 모니터링 과정에서 매우 다양한 형태의 응답 시간 분포 패턴과 직면하게 되며 이런 패턴을 기반으로 시스템의 성능 현황을 직관적으로 파악할 수 있다. 따라서 제니퍼 X-View는 라인 차트 몇 개를 합한 것보다 효과적인 기능을 제공한다.

또한 제니퍼 프로파일링 기능을 통해서 개별 트랜잭션에 대한 상세한 수행 내역을 분석할 수 있다. 수행 내역에는 다음과 같은 내용이 포함된다.

- 임의의 메소드 호출 다이나믹 프로파일링
- 메소드 파라미터와 반환 값 추적
- DB와 바인딩 변수를 포함한 SQL 추적
- 레거시 시스템과의 연동과 같은 외부 트랜잭션 추적

- 파일과 소켓 IO 추적

1.4.6. 다이나믹 프로파일링

제니퍼는 자바 애플리케이션을 재시작하지 않고 임의의 자바 메소드에 대해서 트랜잭션 프로파일링을 활성화 혹은 비활성화 할 수 있는 다이나믹 프로파일링 기능을 제공한다.

1.4.7. 다이나믹 스택트레이스

자바 애플리케이션에서 스택트레이스를 추출하는 전통적인 방법은 임의의 예외가 발생하도록 소스 코드를 수정하고 의도적으로 해당 예외를 발생시켜서 관련 스택트레이스를 명시적으로 출력하는 것이다. 그러나 제니퍼는 자바 애플리케이션을 재시작 하지 않은 상태에서 소스 코드의 수정 없이 스택트레이스를 수집할 수 있다. 스택트레이스를 기록할 메소드를 지정하고 임의의 트랜잭션이 해당 메소드에 진입하면 관련 스택트레이스가 다이나믹하게 기록된다.

1.4.8. 도메인 구성을 통한 통합 모니터링

하나의 제니퍼 서버가 감당할 수 있는 제니퍼 에이전트의 숫자와 업무 처리량에는 한계가 있다. 이 한계를 초과하는 경우에는 여러 개의 제니퍼 서버를 운영하면서 부하를 분산 시켜야 한다. 그러나 여러 개의 제니퍼 서버를 운영하면 관리 비용은 증가하고 사용자 편의성은 감소하는 단점이 있다. 이런 단점을 해결하기 위해서 제니퍼에서는 도메인 구성을 통해서 여러 개의 제니퍼 서버를 통합해서 관리하는 통합 사용자 인터페이스를 제공한다.

1.4.9. 애플리케이션과 SQL 튜닝

애플리케이션의 호출 건수와 평균 응답 시간 통계를 기반으로 성능 저하를 야기하는 병목 애플리케이션을 도출하고 프로파일링 기능을 통해서 해당 애플리케이션을 수행한 트랜잭션의 병목 구간을 구체적으로 파악할 수 있다.

또한 SQL과 외부 트랜잭션이 특정 애플리케이션에서 차지하는 점유 비율을 파악해서 튜닝이 필요한 SQL과 외부 트랜잭션을 손쉽게 찾아낼 수 있다.

1.4.10. 경보와 예외 보니터링

제니퍼는 자바 애플리케이션에서 발생하는 예외를 감지하고 자바 애플리케이션 상태를 분석하여 적절한 경보를 발생시킨다. 경보는 심각, 에러, 경고 등으로 구분되는데, 사용자는 제니퍼 클라이언트를 통해서 실시간으로 경보 발생을 확인하고 과거 경보 내역을 분석할 수 있다.

1.4.11. 모니터링 확장 어뎁터

모니터링 확장 어뎁터를 통해서 미리 정의된 규약과 형식이 없는 비정형 성능 데이터를 수집할 수 있다. 모니터링 확장 어뎁터는 다음과 같다.

- REMON - 임의의 운영체계에서 임의의 스크립트를 실행시키거나, 임의의 데이터베이스에 대해서 임의의 SQL을 실행시키거나, 특정 자바 인터페이스를 구현한 자바 클래스를 실행시키는 방법으로 비정형 성능 데이터를 간편하게 수집한다.
- 엑스트라 에이전트 - 엑스트라 에이전트는 제니퍼 에이전트를 설치한 자바 애플리케이션 내부에서 임의의 성능 데이터를 수집하여 제니퍼 서버 혹은 REMON에 전송한다.
- LogWatcher - 텍스트 로그 파일에서 임의로 지정한 패턴을 검출하고 이벤트를 발생시켜서 관련 데이터를 제니퍼 서버에 전송한다.

1.4.12. 통계 분석 및 보고서

제니퍼는 수집한 성능 데이터를 데이터베이스에 저장하기 때문에 유연한 통계 분석을 가능하게 한다. 그리고 해당 데이터를 2차 가공한 다양한 사용자 정의형 보고서 작성 기능을 제공한다.

1.5. 지원 환경

제니퍼를 사용할 수 있는 최소 사양 및 지원 환경을 설명한다.

1.5.1. 제니퍼 에이전트

제니퍼 에이전트를 설치할 수 있는 운영 체계(Operating Systems)는 다음과 같다.

- AIX 4.3.3, 5.x 32bit, 64bit

- HP-UX 11.x 32bit, 64bit, Itanium 64bit
- Sun Solaris 2.8, 2.9, 10 32bit, 64bit, x68
- Intel Linux 32bit, Redhat Itanium 64bit
- Compaq Tru64 UNIX OSF1
- Microsoft Windows 2000, XP, 2003, Vista
- IBM iSeries(AS400) for WebSphere
- IBM z/OS for WebSphere, zLinux

자바 1.3 이상을 사용하는 자바 애플리케이션만을 제니퍼 에이전트로 모니터링할 수 있다. 제니퍼 에이전트로 모니터링할 수 있는 WAS는 다음과 같다.

- BEA WebLogic 5.1, 6.x, 8.x, 9.x
- IBM WebSphere Application Server 3.5, 4.x, 5.x, 6.x
- TmaxSoft JEUS 3.x, 4.x , 5.x
- Oracle Application Server 9i AS, 10g AS, OC4J, ERP
- SUN Application Server 7.x, 8.x
- Fujitsu Interstage 5.x, 6.x, 7.x, 9.x
- Hitachi Cosminexus 7
- Sybase EAServer 4.x, 5.x
- Macromedia JRun 4.x
- Apache Jakarta Tomcat 3.x, 4.x, 5.x, 6.x
- Caucho Technology Resin 2.x, 3.x
- RedHat JBoss Application Server 3.x, 4.x
- Apache JServ

1.5.2. 제니퍼 닷넷 에이전트

설치할 수 있는 운영 체계(Operating Systems)는 다음과 같다.

- Windows Server 2003 32bit, 64bit
- Windows Server 2008 32bit, 64bit
- Windows Server 2008 R2

닷넷 프레임워크

- 2.0 이상

1.5.3. 제니퍼 서버

모니터링하는 제니퍼 에이전트의 개수가 10개 이하이고, 최대 서비스 요청률이 100 TPS(Transaction Per Second)이고, 수집한 성능 데이터를 1달간 보관하는 것을 기준으로 제니퍼 서버의 하드웨어 최소 사양은 다음과 같다.

- CPU 2개 이상
- 메모리 2 GB 이상
- 하드 디스크 300 GB 이상
- 제니퍼 에이전트와 제니퍼 서버 사이의 네트워크 대역폭은 100 Mbps 이상

그리고 제니퍼 서버를 설치하려면 자바 1.5 이상이 필요하다.

1.5.4. 제니퍼 클라이언트

제니퍼 클라이언트를 사용하기 위해서는 마이크로소프트 윈도우즈 XP 이상을 사용해야 하고, 웹 브라우저로는 마이크로소프트 IE 7 이상과 모질라 파이어폭스 3.0 이상을 사용해야 한다. 또한 자바 플러그인은 자바 1.6.0_10 이상을 사용해야 한다.

그리고 모니터링하는 제니퍼 에이전트의 개수가 10개 이하이고, 최대 서비스 요청률이 100 TPS인 것을 기준으로 제니퍼 클라이언트의 하드웨어 최소 사양은 다음과 같다.

- CPU는 1.8 GHz 이상
- 메모리 2 GB 이상
-

2

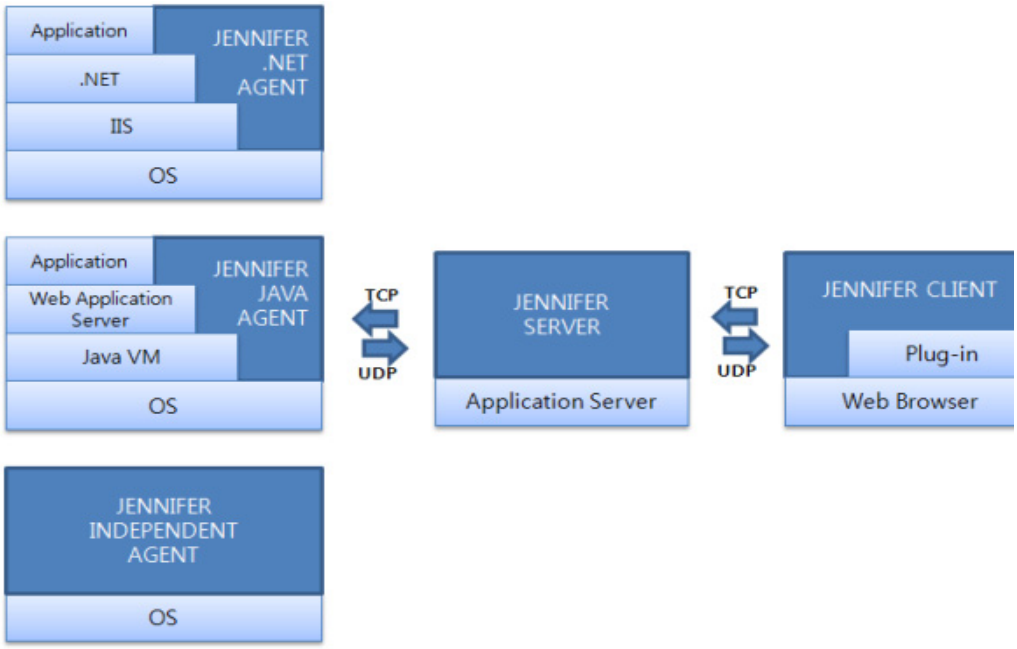
제니퍼 아키텍쳐

제니퍼 아키텍처는 엔터프라이즈 자바 애플리케이션을 부하없이 실시간으로 모니터링하는 것을 기본적인 목표로 한다.

2.1. 제니퍼의 기본 구조

제니퍼는 제니퍼 에이전트, 제니퍼 독립 에이전트, 제니퍼 서버, 그리고 제니퍼 클라이언트로 구성되어 있다.

그림 2-1: 제니퍼 구성도



제니퍼 에이전트와 제니퍼 독립 에이전트는 다양한 성능 데이터를 수집하고 이를 제니퍼 서버에 전송한다. 제니퍼 서버는 제니퍼 에이전트와 제니퍼 독립 에이전트로부터 전송받은 성능 데이터를 가공하여 파일과 데이터베이스에 저장하고, 사용자, 권한 그리고 메뉴 등에 대한 구성 정보를 관리한다. 마지막으로 제니퍼 클라이언트는 성능 데이터를 직관적으로 모니터링하는데 적합한 웹 기반의 사용자 인터페이스를 제공한다.

2.1.1. 제니퍼 자바 에이전트

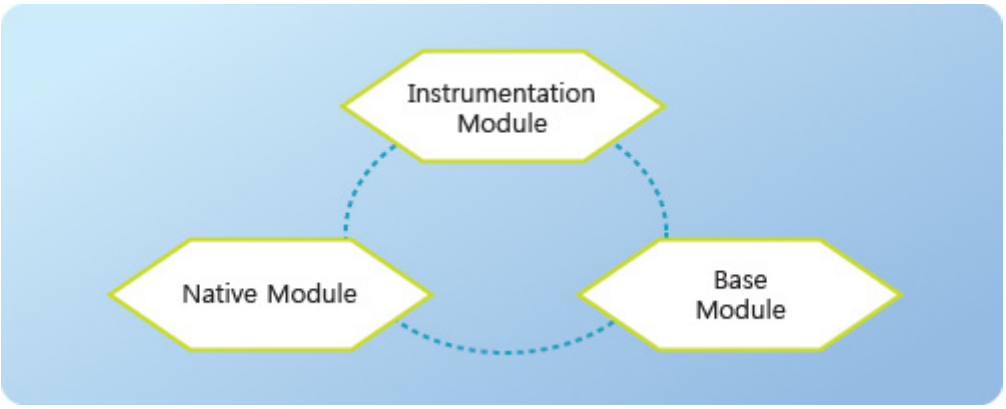
제니퍼 에이전트는 다양한 성능 데이터를 수집하고 이를 제니퍼 서버에 전송하는 일을 담당한다. 제니퍼 자바 에이전트가 수집하는 성능 데이터의 출처는 자바 애플리케이션인데 일반적으로는 Web Application Server에서 동작하는 웹 애플리케이션이다.

Notice: Web Application Server에서 동작하는 자바 애플리케이션 뿐만 아니라 독립적으로 동작하는 자바 애플리케이션도 제니퍼 에이전트로 모니터링할 수 있다.(단, JDK 1.3 이상).

따라서 제니퍼 에이전트는 자바 애플리케이션과 내부적으로 결합해서(embeded) 동작한다. 즉, 제니퍼 에이전트와 자바 애플리케이션은 동일한 자바 가상 머신(Java VIrtual Machine, 이하 JVM) 프로세스 위에서 동작한다.

제니퍼 자바 에이전트는 내부적으로 3가지 모듈, 즉 Instrumentation, Native, Base 모듈로 구성된다.

그림 2-2: 제니퍼 에이전트 구성도



- Instrumentation 모듈 - 모니터링을 하려는 자바 애플리케이션의 소스 코드를 수정하지 않고 성능 데이터 수집에 필요한 추적 코드와 프로파일 정보 추출 코드를 자바 애플리케이션을 구성하는 클래스에 삽입할 수 있어야 한다. Instrumentation 모듈은 이를 담당하는데 제니퍼에서는 LWST(Light Weight StackTrace) 모듈으로도 불린다.
- Native 모듈 - JNI(Java Native Interface)로 호출하는 C 라이브러리이다. 이 모듈은 제니퍼 에이전트를 설치한 자바 애플리케이션이 운영되는 하드웨어의 CPU 사용률과 시스템 메모리 사용량 등을 수집하는 역할을 담당한다. Native 모듈은 운영 체계에 맞추어 컴파일되어 so 혹은 dll 파일 형식으로 제니퍼 패키지에 포함되어 있다.
- Base 모듈 - 다양한 성능 데이터를 수집하고 수집한 성능 데이터를 일차적으로 가공하여 제니퍼 서버에 전송하는 역할을 담당한다.

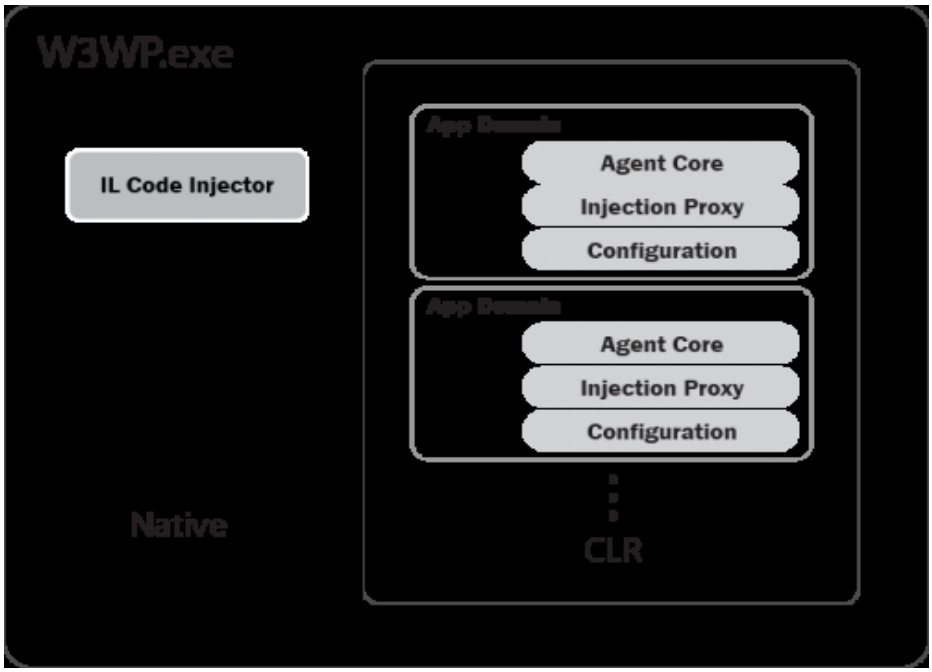
2.1.2. 제니퍼 닷넷 에이전트

닷넷 에이전트는 다양한 성능 데이터를 수집하고 이를 제니퍼 서버에 전송하는 일을 담당한다.

컴파일된 모듈은 DLL로 구성되며, IIS(Internet Information Services)에서 동작하는 웹 애플리케이션을 모니터링하는 하는 경우 w3wp.exe 프로세스의 같은 주소 공간내에서 웹 애플리케이션과 함께 동작한다.

닷넷 에이전트는 내부적으로 크게 2가지 모듈, 즉 IL 코드를 직접 변경하는 .NET Profiler COM 개체와 제니퍼 서버와의 통신을 담당하는 닷넷 에이전트 모듈로 구성된다.

그림 2-3: 제니퍼 .NET 에이전트 구성도



- .NET Profiler COM 개체 - Native 모듈로, CLR 에서 제공되는 Profiler COM 인터페이스를 구현하고 IL 코드를 변경하는 역할을 한다.
- 닷넷 에이전트 모듈 - .NET Profiler COM 개체에 의해 실행시에 동적으로 프로파일링 대상이 되는 응용 프로그램과 연결되어, AppDomain 단위로 다양한 성능 데이터를 수집, 가공하여 제니퍼 서버에 전송하는 역할을 담당한다.

2.1.3. 제니퍼 독립 에이전트

제니퍼 에이전트는 모니터링을 하려는 자바 애플리케이션과 내부적으로 결합해서 동작한다. 하지만 자바 애플리케이션과는 상관이 없는 다른 영역에서 성능 데이터를 수집할 필요도 있다. 예를 들어, 업무 데이터베이스가 운영되는 하드웨어의 CPU 사용률이나 임의의 운영 체계에서 특정 스크립트를 실행해서 얻은 결과 등을 성능 데이터로 수집할 필요가 있을 수 있다. 바로 이를 가능하게 해주는 것이 제니퍼 독립 에이전트이다.

Notice: 제니퍼 독립 에이전트는 미리 정의된 규약과 형식이 없는 비정형 성능 데이터를 수집한다.

제니퍼 독립 에이전트는 어떤 것을 구체적으로 지칭하는 것이 아니라 WMOND, REMON, LogWatcher 등의 모듈들을 논리적으로 지칭하는 용어이다.

2.1.3.1. WMOND

임의의 하드웨어에 대한 시스템 CPU 사용률 데이터를 수집하는 모듈이다. 이 모듈은 C 언어로 개발되어 있고 사용 방법이 간단하다.

2.1.3.2. REMON

비정형 성능 데이터를 수집하는 경우에 데이터의 출처에 따라서 데이터를 수집하는 방법은 달라지지만, 수집한 데이터를 관리하고 제니퍼 서버에 전송하는 것은 공통 모듈로 해결할 수 있다. 바로 REMON이 성능 데이터의 수집 방법을 유연하게 확장하게 하고, 수집된 성능 데이터를 제니퍼 서버로 전송하는 역할을 하는 공통적인 서비스이다.

REMON을 사용하면, 임의의 운영 체계에서 임의의 스크립트를 실행시키거나, 임의의 데이터베이스에 대해서 임의의 SQL을 수행시키거나, 특정 자바 인터페이스를 구현한 자바 클래스를 실행시키는 방법으로 비정형 성능 데이터를 간편하게 수집하여 제니퍼 서버에 전송할 수 있다.

Notice: REMON 자체는 자바 언어로 구현된 애플리케이션이다.

2.1.3.3. LogWatcher

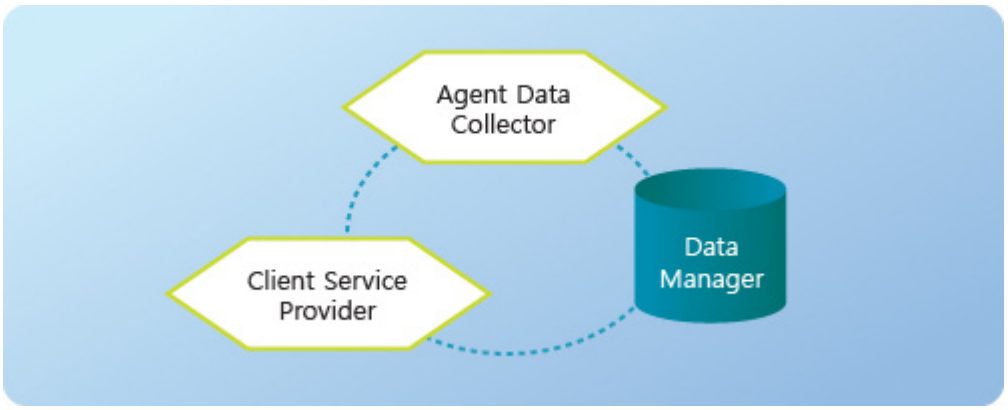
로그 감시기의 역할을 수행하는 모듈이다. LogWatcher는 텍스트 로그 파일에서 임의로 지정한 패턴을 검출하고 이벤트를 발생시켜서 관련 데이터를 제니퍼 서버에 전송하는 역할을 담당한다. 그러면 사용자는 제니퍼 서버를 통해서 이를 감지할 수 있게 된다.

2.1.4. 제니퍼 서버

제니퍼 서버는 제니퍼 에이전트와 제니퍼 독립 에이전트로부터 전송받은 성능 데이터를 가공하여 파일과 데이터베이스에 저장하고, 사용자, 권한 그리고 메뉴 등에 대한 구성 정보를 관리한다.

하나의 제니퍼 서버가 담당하는 제니퍼 에이전트의 숫자가 많을 수 있다. 따라서 제니퍼 서버는 이런 상황에서 발생하는 높은 부하를 적절하게 처리할 수 있도록 설계되어 있다. 이를 위해서 제니퍼 서버는 내부적으로 크게 3가지 모듈, 즉 Agent Data Collector, Data Manager, Client Service Provider 모듈로 구성된다.

그림 2-4: 제니퍼 서버 구성도



- Agent Data Collector 모듈 - 제니퍼 에이전트와 제니퍼 독립 에이전트가 UDP로 보내는 성능 데이터를 수신하고, 10분마다 반복적으로 개별 제니퍼 에이전트에 TCP 요청을 보내서 애플리케이션 처리 현황 통계 데이터를 수집하는 역할을 담당하는 모듈이다.
- Data Manager 모듈 - Agent Data Collector 모듈이 수집한 성능 데이터를 파일과 데이터베이스에 저장하는 역할을 담당한다. 기본적으로 제니퍼 서버는 내장 데이터베이스(아파치 더비)를 사용하는데, 상황에 따라서 오라클 데이터베이스나 IBM DB2 등을 사용할 수도 있다.
- Client Service Providor 모듈 - 제니퍼 클라이언트가 사용자 인터페이스를 구성하는데 필요한 서비스와 데이터를 제공하는 역할을 담당한다.

제니퍼 서버는 3가지 주요 모듈 이외에도 스케줄러, 보고서, 구성 관리 모듈 등을 포함하고 있다.

2.1.5. 제니퍼 클라이언트

제니퍼는 성능 데이터를 직관적으로 모니터링하는데 적합한 사용자 인터페이스를 제공하고 있으며 이를 제니퍼 클라이언트라고 한다. 기본적으로 제니퍼 클라이언트는 접근성을 높이기 위해서 XHTML, CSS, 자바스크립트, AJAX 등의 기술로 구현한 웹에 기반하고 있기 때문에 사용자는 웹 브라우저를 통해서 제니퍼를 사용한다.

그리고 성능 데이터의 변화 추이를 실시간으로 모니터링하는데 적합한 다양한 차트를 자바 애플릿으로 구현하여 제공한다. 따라서 제니퍼 클라이언트를 사용하려면 자바 플러그인 6.0 이상을 설치한 웹 브라우저가 필요하다.

그림 2-5: 제니퍼 클라이언트 구현 기술

제니퍼 서버와 제니퍼 클라이언트의 시간이 일치하지 않는 경우에, 제니퍼 클라이언트에 표시되는 데이터의 시간은 제니퍼 서버의 시간을 기준으로 한다.

2.2. 제니퍼가 수집하는 성능 데이터

제니퍼 서버가 제니퍼 에이전트와 제니퍼 독립 에이전트 등으로부터 수집하는 주요 성능 데이터에 대해서 설명한다.

2.2.1. 데이터의 출처

제니퍼 서버는 제니퍼 에이전트와 WMOND, REMON, LogWatcher 등의 제니퍼 독립 에이전트로부터 다양한 성능 데이터를 수집한다.

- 제니퍼 서버는 제니퍼 자바/닷넷 에이전트와 WMOND, REMON, LogWatcher 등의 제니퍼 독립 에이전트로부터 다양한 성능 데이터를 수집한다.
- 제니퍼 자바/닷넷 에이전트 - 제니퍼 에이전트는 서비스 요청률, 평균 응답 시간, 자바 힙 메모리 사용량 등의 일반 성능 데이터와 액티브 서비스, 애플리케이션, SQL, 외부 트랜잭션 등의 처리 현황 통계, 경보와 예외 데이터 그리고 X-View 트랜잭션과 프로파일 데이터 등을 수집한다.
- WMOND - WMOND는 임의의 시스템 CPU 사용률 데이터를 수집한다.
- REMON - REMON은 비정형 데이터를 수집한다. 데이터의 형식이나 출처에 제한이 없다.
- LogWatcher - LogWatcher는 임의의 텍스트 로그 파일에서 주요 이벤트를 수집한다.

2.2.2. 데이터의 저장

제니퍼 서버는 수집한 성능 데이터를 파일과 데이터베이스에 저장한다.

우선 제니퍼 서버는 프로파일 정보와 같이 데이터의 사이즈가 큰 것과 개별 트랜잭션 수행과 관련한 정보를 파일에 저장한다. 그리고 나머지 대부분의 성능 데이터는 데이터베이스에 저장한다.

2.2.3. 제니퍼 에이전트가 수집하는 성능 데이터

제니퍼 에이전트가 수집하는 다양한 성능 데이터는 다음과 같다.

2.2.3.1. 일반 성능 데이터

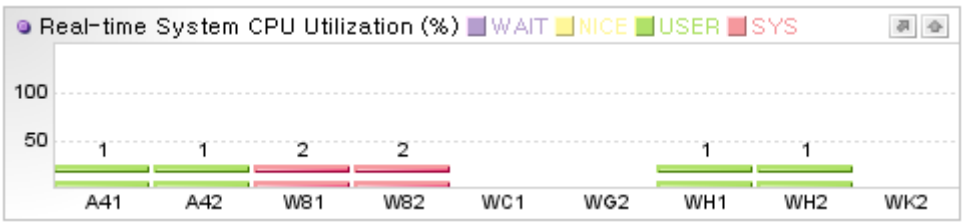
동시단말 사용자 수나 서비스 요청률 등과 같이 사용자 및 서비스 요청 처리와 관련한 데이터를 서비스 성능 데이터라고 하고, CPU 사용률, 메모리 사용량 등과 같이 시스템 상태와 관련한 데이터를 리소스 성능 데이터라고 한다. 그리고 서비스 성능 데이터와 리소스 성능 데이터를 통틀어서 일반 성능 데이터라고 한다.

제니퍼 에이전트는 일반 성능 데이터를 1초 주기로 수집하여 제니퍼 서버에 전송한다. 이렇게 수집하는 데이터는 수집하는 시점의 값이 아니라 수집하는 시점을 기준으로 이전 30초 동안의 평균 값이다. 예를 들어, 제니퍼 에이전트가 10시 28분 45초에 수집한 평균 응답 시간은 10시 28분 15초에서 10시 28분 45초 사이의 평균 값이고, 10시 28분 46초에 수집한 평균 응답 시간은 10시 28분 16초에서 10시 28분 46초 사이의 평균 값이다.

Notice: 단, 호출 건수와 동시단말 사용자 수와 같이 시간에 따라서 누적되는 정량적인 데이터는 평균 값이 아닌 해당 구간에서의 누적값을 의미한다. 그리고 액티브 서비스 개수와 DB 커넥션 개수는 30초 평균 값이 아닌 해당 시점의 값을 의미한다.

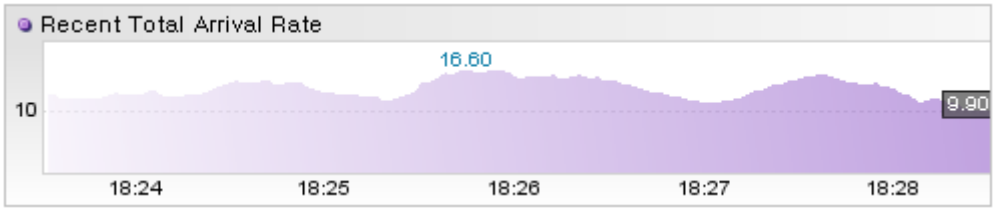
수집하는 시점의 값이 아닌 30초 평균 값을 사용하는 이유는 수집 시점의 값이 지니는 변동 폭이 클 수 있으므로 그 편차를 통계적으로 조정하기 위함이다. 이렇게 수집한 30초 평균 데이터를 이퀄라이저 차트와 런타임 라인 차트 등에서 실시간으로 확인할 수 있다.

그림 2-6: 이퀄라이저 차트



이퀄라이저 차트는 최근에 수집한 30초 평균 값을 표시한다. 단, 액티브 서비스 개수와 DB 커넥션 개수는 30초 평균 값이 아닌 해당 시점의 값을 사용한다.

그림 2-7: 런타임 라인 차트



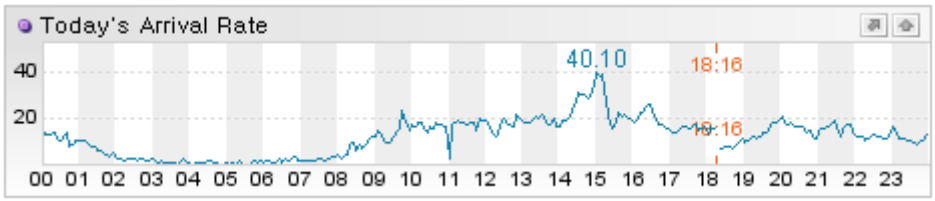
런타임 라인 차트는 5분 동안 수집한 30초 평균 값을 선을 통해서 표시한다. 런타임 라인 차트에서는 액티브 서비스 개수도 30초 평균 값을 선으로 표시하지만, DB 커넥션 개수는 해당 시점의 값을 선으로 표시한다.

그러나 제니퍼 서버는 제니퍼 에이전트로부터 수집한 30초 평균 값을 데이터베이스에 1초 주기로 저장하지는 않는다. 왜냐하면 1초 주기로 30초 평균 값을 데이터베이스에 저장하면 데이터의 양이 많아져서 저장 공간의 낭비와 제니퍼 서버의 성능 저하를 야기하기 때문이다. 따라서 제니퍼 서버는 30초 평균 값을 5분 평균으로 재가공한 값을 데이터베이스에 5분 주기로 저장한다.

Notice: 제니퍼 서버는 일반 성능 데이터를 PERF_X_01~31 테이블에 저장한다. 24시간을 5분으로 나누면 288이다. 따라서 제니퍼 에이전트 별로 하루에 288건의 일반 성능 데이터가 PERF_X_01~31 테이블에 저장된다.

5분 평균 데이터는 라인 차트에서 확인할 수 있다.

그림 2-8: 라인 차트



라인 차트는 24시간 동안의 성능 데이터에 대한 288개의 5분 평균 값을 선으로 표시한다.

2.2.3.2. 서비스 성능 데이터

다음은 서비스 성능 데이터에 대한 설명이다.

표 2-1: 서비스 성능 데이터

성능 데이터	설명
동시 단말 사용자 수	쿠키를 지원하는(클라이언트가 웹 브라우저인) 자바 웹 애플리케이션에서만 수집할 수 있는 성능 데이터로서 자바 웹 애플리케이션을 동시에 사용하는 사용자의 수를 의미한다. 사용자 별로 고유한 쿠키를 부여하여 계산하기 때문에 클러스터링 환경에서 동일 사용자가 이중으로 계산되는 것을 방지할 수 있다.
액티브 서비스 개수	자바 애플리케이션이 현재 처리하고 있는 서비스의 개수를 의미한다.
액티브 사용자 수	서비스를 요청한 사용자가 동일한 액티브 서비스들이 있을 수 있다. 액티브 서비스들에서 사용자가 동일한 서비스들을 하나로 간주하여 계산한 것이 액티브 사용자 수이다.
서비스 요청률	1초 당 들어온 호출 건수의 비율
서비스 처리율	1초 당 처리한 서비스 요청의 비율
평균 응답 시간	모든 서비스 요청을 처리하는데 소요된 응답 시간의 합을 호출 건수로 나눈 값으로서 단위는 초이다.
대기 시간	사용자가 마지막 서비스를 호출한 시간과 새로운 서비스를 호출한 시간의 차이의 평균 값을 의미한다.
호출 건수	사용자에 의한 서비스 요청 건수
일일 방문자 수	해당 일에 신규로 방문한 사용자의 수를 의미한다.
시간당 방문자 수	해당 시간에 신규로 방문한 사용자의 수를 의미한다.

Notice: 일반적으로 사용자는 웹 브라우저 프로세스를 의미한다. 따라서 하나의 컴퓨터에서 여러 개의 웹 브라우저 프로세스를 통해 자바 애플리케이션을 사용하면, 제니퍼 에이전트는 모든 웹 브라우저 프로세스를 별도의 사용자로 간주하여 사용자 수를 계산한다. 따라서 사용중인 웹 브라우저 프로세스를 종료한 후에 새로운 웹 브라우저 프로세스로 자바 애플리케이션을 사용하는 경우에도 제니퍼 에이전트는 이를 별도의 사용자로 간주한다.

2.2.3.3. 리소스 성능 데이터

다음은 리소스 성능 데이터에 대한 설명이다.

표 2-2: 리소스 성능 데이터

성능 데이터	설명
시스템 CPU 사용률	제니퍼 에이전트가 수집하는 시스템 CPU 사용률로 단위는 퍼센트이다.
프로세스 CPU 사용률	제니퍼 에이전트가 수집하는 프로세스 CPU 사용률로 단위는 퍼센트이다.
시스템 메모리 사용량	제니퍼 에이전트가 수집하는 시스템 메모리 사용량으로 단위는 MB이다.

표 2-2: 리소스 성능 데이터

성능 데이터	설명
프로세스 메모리 사용량	제니퍼 에이전트가 수집하는 애플리케이션이 사용하는 메모리 사용량으로 단위는 MB이다. 프로세스 힙 메모리 사용량과는 다르다.
프로세스 힙 메모리 전체	프로세스 힙 메모리 전체 크기로 단위는 MB이다.
물리 메모리 전체	물리 메모리 전체 크기로 단위는 MB이다.
프로세스 힙 메모리 사용량	프로세스 힙 메모리 사용량으로 단위는 MB이다.
대기 중인 DB 커넥션 개수	DB 커넥션 풀에서 대기 중인 DB 커넥션의 개수
할당된 DB 커넥션 개수	서비스 쓰레드가 DB 커넥션 풀로부터 할당받은 DB 커넥션 중에서 SQL 쿼리를 수행하고 있지 않은 DB 커넥션의 개수
사용 중인 DB 커넥션 개수	서비스 쓰레드가 DB 커넥션 풀로부터 할당받은 DB 커넥션 중에서 SQL 쿼리를 수행하고 있는 DB 커넥션의 개수

리소스 성능 데이터에 대한 자세한 사항은 [리소스, 외부 트랜잭션, DB 모니터링]을 참조한다.

2.2.3.4. 애플리케이션 처리 현황 통계 데이터

제니퍼 에이전트는 액티브 서비스, 애플리케이션, SQL, 외부 트랜잭션, 예외, JDBC 등의 처리 현황 통계 데이터를 수집한다. 이 데이터는 10분 단위로 통합되어 수집되는데 최근 10분에 해당하는 실시간 처리 현황 통계 데이터와 10분 이전에 해당하는 과거 처리 현황 통계 데이터로 구분된다. 10분이 경과하면 그 때까지의 실시간 처리 현황 통계 데이터는 데이터베이스에 저장된다.

Notice: 액티브 서비스, DB와 관련된 처리 현황 통계 데이터는 데이터베이스에 저장되지 않는다.

여기서 제공하는 데이터는 10분 단위로 요약한 통계 값이다. 예를 들어, 10분 동안의 A 애플리케이션에 대한 전체 호출 건수와 평균 응답 시간 등은 제공하지만 개별 트랜잭션에 대한 요청 시간과 응답 시간 등은 제공하지 않는다. 개별 트랜잭션을 분석하려면 [X-View와 프로파일링]을 참조한다..

Notice: 정확하게 설명하자면, 실시간 처리 현황 통계 데이터는 현재 시간을 기준으로 한 10분을 의미하지 않는다. 00분, 10분, 20분과 같이 10분 단위를 기준으로 나누어지는 시간을 의미한다. 만약 현재 시간이 09시 18분이라면 실시간 처리 현황 통계 데이터에서는 09시 10분에서 09시 20분까지의 성능 데이터를 보여준다.

2.2.3.5. 경보와 예외 데이터

제니퍼 에이전트는 자바 애플리케이션에서 발생하는 예외(Exception)를 감지하고 자바 애플리케이션 상태를 분석하여 적절한 경보(Alert)를 발생시킨다. 경보는 심각(Critical), 에러(Error), 경고(Warning) 등으로 구분되는데 사용자는 제니퍼 클라이언트를 통해서 실시간으로 경보 발생을 확인하고 과거 경보 내역을 분석할 수 있다

X-View 트랜잭션과 프로파일 데이터에 대한 자세한 사항은 [X-View와 프로파일링]을 참조한다.

2.2.3.6. X-View 트랜잭션과 프로파일 데이터

제니퍼 에이전트는 자바 애플리케이션이 처리하는 트랜잭션 정보를 수집하여 사용자에게 X-View 차트를 통해 제공한다. 제니퍼 에이전트가 트랜잭션에 대해서 수집하는 데이터를 X-View 데이터라고 하고, X-View 데이터는 트랜잭션 데이터와 프로파일 데이터로 구성된다.

X-View 트랜잭션과 프로파일 데이터에 대한 자세한 사항은 [X-View와 프로파일링]을 참조한다.

2.2.4. WMOND가 수집하는 성능 데이터

WMOND는 임의의 시스템 CPU 사용률 데이터를 수집한다. WMOND와 관련한 자세한 사항은 [제니퍼 에이전트 Native 모듈 테스트]를 참조한다.

> **Notice:** WMOND는 컴퓨터에 설치된 물리적인 CPU 별로 구분하여 CPU 사용률을 수집하고 제니퍼 에이전트는 컴퓨터 전체의 논리적인 CPU 사용률을 수집한다.

2.2.5. REMON이 수집하는 성능 데이터

REMON은 다양한 비정형 성능 데이터를 수집한다. REMON을 통해서 비정형 성능 데이터를 수집하는 방법은 [REMON]을 참조하고, 이 데이터를 모니터링 하는 방법은 [사용자 정의 대시보드]를 참조한다.

2.2.6. LogWatcher가 수집하는 성능 데이터

LogWatcher는 임의의 텍스트 로그 파일에서 임의로 지정한 패턴을 검출하여 관련 이벤트를 수집한다. LogWatcher에 대한 상세한 설명은 [LogWatcher]를 참조한다.

2.3. 네트워크 구성과 핵심 쓰레드

제니퍼를 구성하는 제니퍼 에이전트, 제니퍼 독립 에이전트, 제니퍼 서버, 제니퍼 클라이언트 등의 주요 모듈은 서로 독립적으로 운영되는 프로세스이다. 따라서 모듈 간의 데이터 전송은 UDP 혹은 TCP 방식의 네트워크 통신으로 이루어진다.

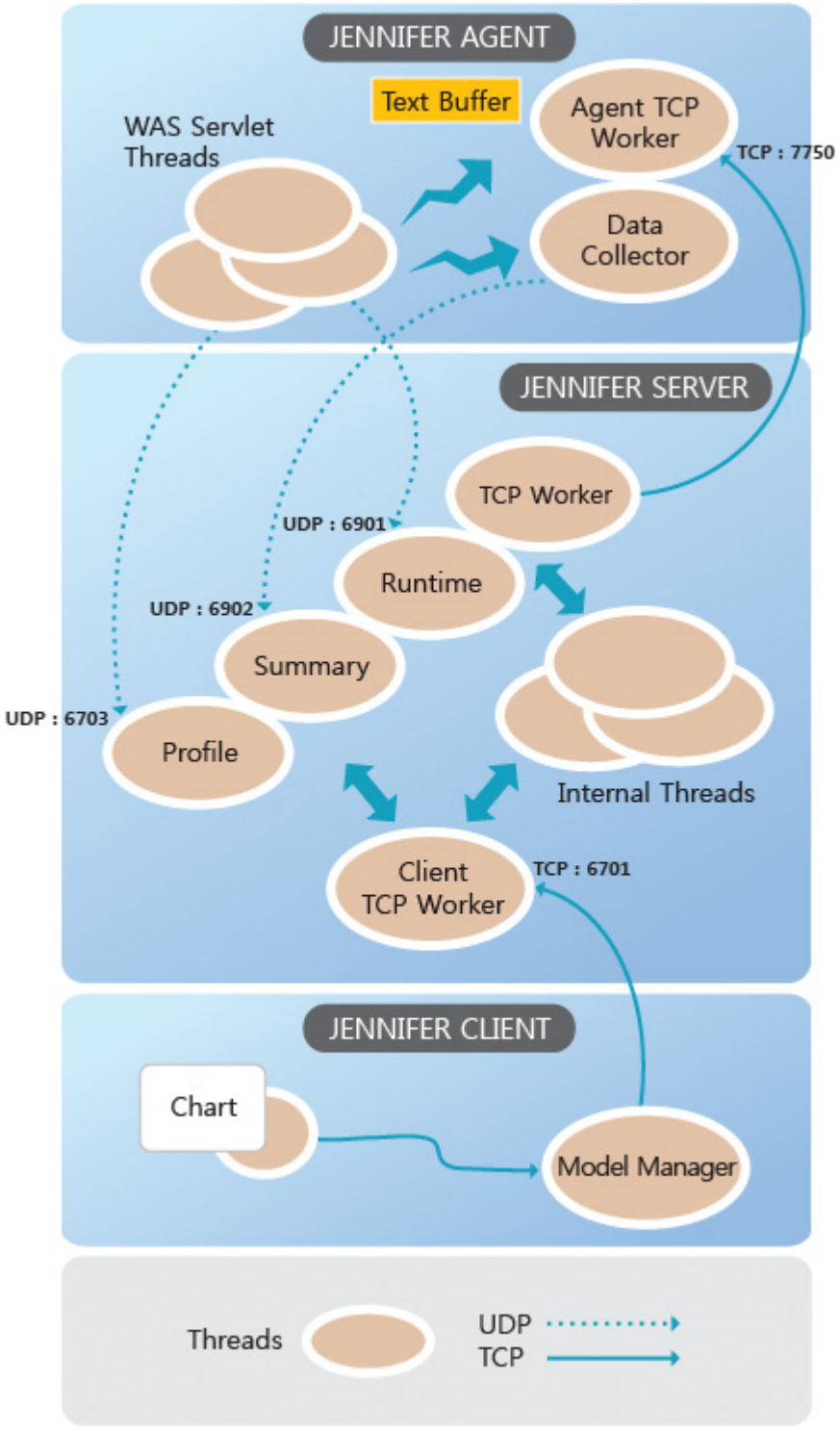
Notice: 따라서 제니퍼 서버와 제니퍼 에이전트를 동일한 하드웨어에 설치하여도 제니퍼 서버와 제니퍼 에이전트간의 데이터 전송은 네트워크 통신으로 이루어진다.

데이터 전송은 제니퍼 에이전트와 제니퍼 서버, 제니퍼 독립 에이전트와 제니퍼 서버, 그리고 제니퍼 서버와 제니퍼 클라이언트 간에만 이루어진다. 제니퍼 클라이언트와 제니퍼 에이전트 혹은 제니퍼 에이전트와 제니퍼 에이전트 간에는 데이터 전송이 이루어지지 않는다.

그리고 전송받은 데이터를 처리하는 다양한 자바 쓰레드가 제니퍼 에이전트와 제니퍼 서버에 존재한다.

Notice: 다음 그림은 네트워크와 자바 쓰레드 구성도이다. 그림에 표시된 포트 번호는 기본 값을 의미하며, 임의의 포트 번호로 변경할 수 있다.

그림 2-9: 네트워크와 자바 쓰레드 구성도



2.3.1. 제니퍼 에이전트와 제니퍼 서버

제니퍼 에이전트는 성능 데이터를 UDP 방식으로 제니퍼 서버에 전송한다. 제니퍼 서버는 제니퍼 에이전트가 보내는 성능 데이터를 3개의 UDP 포트로 분리해서 받아들인다. 그런데 각각의 포트가 받아들이는 성능 데이터는 상이하다.

제니퍼 에이전트는 제니퍼 서버의 server_udp_runtime_port 옵션으로 설정한 UDP 포트로 모든 트랜잭션의 시작과 종료와 관련한 데이터를 전송한다. 이 데이터는 크기가 매우 작으며 주로 X-View 차트를 표현하는데 사용된다. 기본 포트 번호는 6901이다.

```
server_udp_runtime_port = 6901
```

그리고 제니퍼 서버의 server_udp_runtime_port 옵션으로 설정한 포트로 들어오는 성능 데이터를 처리하는 제니퍼 서버의 자바 쓰레드를 Runtime UDP Worker라고 한다. Runtime UDP Worker의 숫자는 제니퍼 서버의 number_of_udp_runtime_workers 옵션으로 설정한다. 기본 값은 10이다.

```
number_of_udp_runtime_workers = 10
```

제니퍼 에이전트는 제니퍼 서버의 server_udp_listen_port 옵션으로 설정한 UDP 포트로 1초마다 반복적으로 서비스 요청률, 평균 응답 시간 등의 일반 성능 데이터를 전송한다. 기본 포트 번호는 6902이다.

```
server_udp_listen_port = 6902
```

그리고 제니퍼 서버의 server_udp_listen_port 옵션으로 설정한 포트로 들어오는 성능 데이터를 처리하는 제니퍼 서버의 자바 쓰레드를 Summary UDP Worker라고 한다. Summary UDP Worker의 숫자는 제니퍼 서버의 number_of_udp_listen_workers 옵션으로 설정한다. 기본 값은 10이다.

```
number_of_udp_listen_workers = 10
```

제니퍼 에이전트는 제니퍼 서버의 server_udp_lwst_call_stack_port 옵션으로 설정한 UDP 포트로 2초마다 반복적으로 X-View 트랜잭션 프로파일 데이터를 전송한다. 기본 포트 번호는 6703이다.

```
server_udp_lwst_call_stack_port = 6703
```

그리고 제니퍼 서버의 server_udp_lwst_call_stack_port 옵션으로 설정한 포트로 들어오는 성능 데이터를 처리하는 제니퍼 서버의 자바 쓰레드를 Profile UDP Worker라고 한다.

Profile UDP Worker의 숫자는 제니퍼 서버의 number_of_udp_callstack_workers 옵션으로 설정한다. 기본 값은 30이다.

```
number_of_udp_callstack_workers = 30
```

그런데 제니퍼 에이전트가 제니퍼 서버에 UDP 방식으로 전송하는 성능 데이터의 크기가 운영 체계에 설정한 UDP 전송 최대 값을 초과하는 경우에는 데이터는 전송되지 않고 유실되고 에러 메시지는 로그 파일에 기록된다. 그러나 제니퍼 에이전트와 제니퍼 서버 사이에 별도의 네트워크 장비가 존재하고, 제니퍼 에이전트가 제니퍼 서버에 UDP 방식으로 전송하는 성능 데이터의 크기가 해당 장비의 UDP 전송 최대 값을 초과하는 경우에는 에러 메시지의 기록 없이 유실된다.

Notice: UDP 방식으로 전송할 수 있는 데이터 크기(UDP Send Buffer Size)는 64 KB가 최대 값이다. 선 솔라리스와 HP HP-UX 운영 체계는 64 KB가 기본 값으로 설정되어 있지만 IBM AIX 운영 체계는 64 KB 보다 작은 수치가 기본 값이다.

server_udp_runtime_port, server_udp_listen_port, server_udp_lwst_call_stack_port 등의 옵션은 제니퍼 서버와 제니퍼 에이전트에 모두 설정해야 한다. 제니퍼 서버에서는 이 옵션을 통해서 제니퍼 에이전트가 보내는 성능 데이터를 수신할 UDP 포트를 열고, 제니퍼 에이전트는 이 옵션을 통해서 성능 데이터를 송신할 제니퍼 서버의 포트를 지정한다.

그리고 제니퍼 에이전트의 경우에는 제니퍼 서버의 IP 주소를 제니퍼 에이전트의 udp_server_host 옵션으로 설정해야 한다.

```
udp_server_host = localhost
```

제니퍼는 데이터 전송 안전성보다는 모니터링 대상 자바 애플리케이션의 안정성을 우선시하기 때문에 데이터 전송 실패가 일어나도 데이터를 재전송하지 않는다.

그리고 제니퍼 서버가 제니퍼 에이전트의 TCP 포트로 역방향 연결을 하기도 한다. 주로 액티브 서비스 목록, 10분간 애플리케이션 처리 현황 통계, 로딩 클래스 목록, 소켓/파일 목록 등과 같이 데이터의 크기가 크고 데이터베이스나 파일에 저장되지 않는 데이터를 조회하는데 TCP 역방향 연결을 사용한다. 제니퍼 에이전트의 agent_tcp_port 옵션으로 제니퍼 서버로부터의 TCP 요청을 받아들이는 제니퍼 에이전트의 포트 번호를 설정한다. 기본 포트 번호는 7750이다.

```
agent_tcp_port = 7750
```

그리고 제니퍼 에이전트의 agent_tcp_port 옵션으로 설정한 포트로 들어오는 제니퍼 서버의 요청을 처리하는 제니퍼 에이전트의 자바 쓰레드를 Agent TCP Worker라고 한다. Agent TCP Worker의 숫자는 제니퍼 에이전트의 number_of_tcp_workers 옵션으로 설정한다. 기본 값은 5이다.

```
number_of_tcp_workers = 5
```

Notice: 제니퍼 서버에서 제니퍼 에이전트 방향으로의 연결을 역방향 연결이라고 부르는 이유는 데이터를 보내고자 하는 제니퍼 에이전트에서 네트워크 연결을 하는 것이 아니라, 데이터를 조회하는 제니퍼 서버에서 네트워크 연결을 하는 것을 강조하기 위함이다.

제니퍼 서버에서 제니퍼 에이전트로의 TCP 역방향 연결이 실패하는 경우를 대비하기 위한 타임아웃 시간은 제니퍼 서버의 agent_tcp_io_timeout 옵션으로 설정한다. 기본 값은 5000이고 단위는 밀리 세컨드이다.

```
agent_tcp_io_timeout = 5000
```

2.3.2. 제니퍼 독립 에이전트와 제니퍼 서버

제니퍼 서버는 제니퍼 에이전트가 보내는 성능 데이터를 받아들이는 UDP 포트로 제니퍼 독립 에이전트가 보내는 성능 데이터도 받아들인다.

따라서 WMOND, REMON, LogWatcher 등은 제니퍼 서버의 server_udp_listen_port 옵션으로 설정한 UDP 포트로 성능 데이터를 전송한다.

2.3.3. 제니퍼 서버와 제니퍼 클라이언트

제니퍼 서버와 제니퍼 클라이언트는 웹 기반의 사용자 인터페이스 제공을 위해서 HTTP 프로토콜을 사용한다. 이 때 사용하는 기본 HTTP 포트 번호는 7900이다.

그리고 자바 애플릿은 차트 구성에 필요한 데이터를 제니퍼 서버로부터 TCP 통신을 통해서 획득한다. 이 때 사용되는 제니퍼 서버의 기본 TCP 포트 번호는 제니퍼 서버의 server_tcp_port 옵션으로 설정하고 기본 포트 번호는 6701이다.

```
server_tcp_port = 6701
```

그리고 제니퍼 서버의 server_tcp_port 옵션으로 설정한 포트로 들어오는 제니퍼 클라이언트의 요청을 처리하는 제니퍼 서버의 자바 쓰레드를 Client TCP Worker라고 한다. Client TCP Worker의 숫자는 제니퍼 서버의 number_of_tcp_pooled_workers 옵션으로 설정한다. 기본 값은 80이다.

```
number_of_tcp_pooled_workers = 80
```

그러나 제니퍼 서버에서 제니퍼 클라이언트 방향으로의 TCP 방식 호출은 이루어지지 않는다.

2.3.4. 네트워크 구성과 관련된 주의 사항

네트워크 구성과 관련된 주의 사항은 다음과 같다.

- 동일 포트의 중복 사용 방지 - 동일한 하드웨어에 복수의 제니퍼 에이전트를 설치한 경우에는, 제니퍼 에이전트의 agent_tcp_port 옵션으로 설정한 포트 번호가 중복되지 않도록 한다. 그리고 동일한 하드웨어에 복수의 제니퍼 서버를 설치한 경우에는, 제니퍼 서버의 server_udp_runtime_port, server_udp_listen_port, server_udp_lwst_call_stack_port, server_tcp_port 등의 옵션으로 설정한 포트 번호가 중복되지 않도록 한다.
- 방화벽에 대한 점검 - 제니퍼를 구성하는 주요 모듈 사이에 방화벽이 있는 경우에는, 관련 포트가 정상적으로 열려 있는지 확인해야 한다.

2.4. 제니퍼 아키텍처의 특징

제니퍼 아키텍처의 특징은 다음과 같다.

- 제니퍼 아키텍처는 모니터링으로 인해서 발생하는 운영 서버에 대한 부하를 최소화하고, 소스 코드에 대한 수정 없이 추적 코드를 삽입할 수 있는 효과적인 방법을 제공한다.
- 네트워크에 대한 부하를 줄이고 운영 안정성을 확보하기 위해서 모니터링에 필요한 성능 데이터를 최소화하는데 초점을 맞추고 있다. 그래서 엔터프라이즈 자바 애플리케이션을 모니터링하는데 실질적으로 도움을 주는 정보를 기준으로 성능 데이터의 우선 순위를 설정한 후, 그 우선 순위를 고려하여 성능 데이터를 수집한다.
- 효율적으로 네트워크 자원을 활용을 위해서 UDP 방식을 근간으로 데이터를 수집하고, 실시간으로 액티브 서비스 목록을 조회하고 장애 원인을 파악하기 위해서 TCP 역방향 연결을 사용한다.
- 저장 공간의 효율적인 사용과 효과적인 분석에 알맞은 데이터 저장 구조를 제공한다.
- 자바 애플리케이션뿐만 아니라 다양한 시스템 및 애플리케이션에서 비정형 성능 데이터를 유연하게 수집할 수 있도록 해주는 제니퍼 독립 에이전트를 제공한다.
- 제니퍼는 성능 데이터를 직관적으로 모니터링하는데 적합한 웹 기반의 사용자 인터페이스를 제공한다.

3

제니퍼 자바 에이전트 운영 관리

3.1. 제니퍼 에이전트 설정 파일

모니터링 수준과 방법, 네트워크 포트 번호 지정 등을 포함한 대부분의 설정은 제니퍼 에이전트 옵션을 통해서 이루어진다. 제니퍼 에이전트 설치 시에 자바 -Djennifer.config 옵션으로 지정한 파일이 제니퍼 에이전트 설정 파일이다.

3.1.1. 설정 파일 형식

제니퍼 에이전트 옵션은 텍스트 형식의 설정 파일에 기록된다. 이 파일은 다음과 같은 특징을 갖는다.

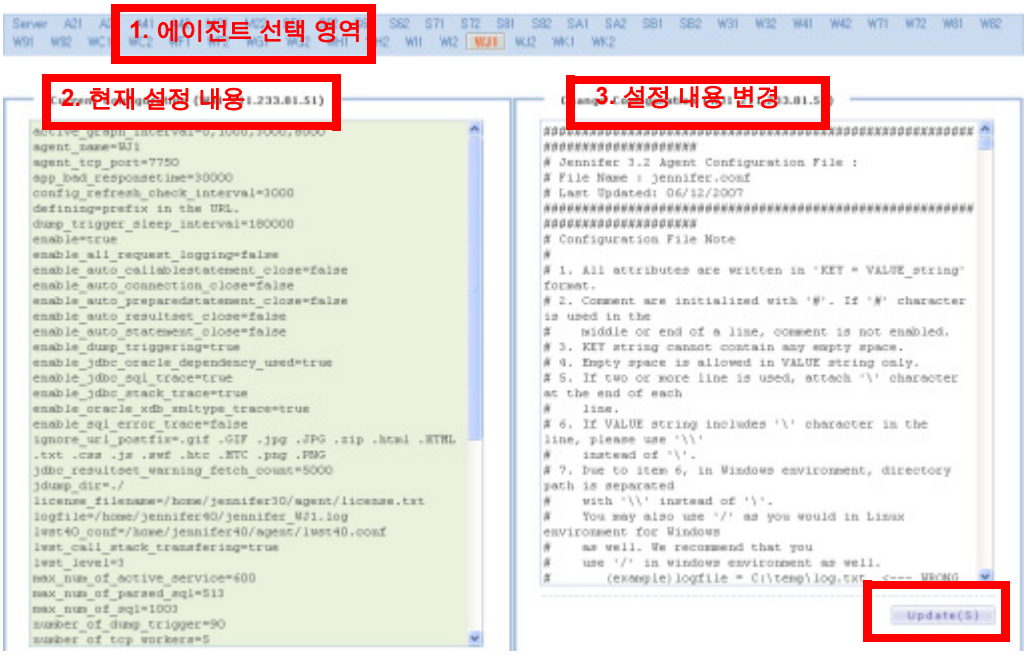
- 키 = 값 형식으로 기술한다. 키는 제니퍼 에이전트의 옵션을 의미한다.
- [#]으로 시작하는 줄은 주석이다. 그러나 문자열 중간에 있는 [#] 기호는 주석으로 인식되지 않는다.
- [=] 대신 '공백'을 키와 값의 구분자로 사용할 수 있다.
- 앞의 이유로 인하여 키에 공백을 사용하는 것을 허용하지 않는다.
- 그러나 값에는 공백을 사용할 수 있다.

- 두 줄 이상을 사용하려면 라인의 끝에 [\] 문자를 사용한다. 만약, [\] 문자 자체가 필요한 경우에는, [\]대신 [\\]을 사용한다.
- 앞의 이유로 인하여 마이크로소프트 윈도우즈 환경에서 디렉토리 구분은 [\]가 아니라 [\\]로 해야 한다. 그런데 윈도우즈 환경에서 유닉스처럼 [/]을 사용할 수 있기 때문에 윈도우즈 환경에서도 [/]을 사용하는 것을 권장한다.
- 일부 옵션을 제외한 대부분의 옵션들은 수정과 함께 실시간으로 반영된다. 수정된 옵션을 반영하기 위해 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 하는 경우에는 해당 옵션을 설명할 때 이를 명기하였다.

3.1.2. 설정 변경

설정 파일은 에디터를 통해서 직접 수정하거나, **[구성 관리 | 구성 설정]** 메뉴에서 수정할 수 있다. 단, 제니퍼 클리이언트에서는 관리자 그룹에 속한 사용자만이 옵션을 수정할 수 있다.

그림 3-1: 제니퍼 옵션 설정 화면



1. 에이전트 선택 영역 - 제니퍼 에이전트 옵션 내용을 확인하거나 수정하려면 해당 제니퍼 에이전트를 선택한다.
2. 현재 설정 내용 - 왼쪽에 있는 현재 설정 내용에서 제니퍼 에이전트 옵션을 확인한다.
3. 설정 내용 변경 - 오른쪽에 있는 설정 내용 변경 입력 폼에서 기존 옵션을 수정하거나 새로운 옵션을 추가한 후에, 하단에 있는 [수정] 버튼을 클릭한다.

제니퍼 에이전트 설정 파일은 제니퍼 에이전트의 config_refresh_check_interval 옵션으로 설정한 시간마다 반복적으로 변경 여부가 체크된다. 단위는 밀리 세컨드이다.

```
config_refresh_check_interval = 3000
```

3.1.3. 새로운 설정 파일 만들기

새로운 제니퍼 에이전트 설정 파일을 만들려면 JENNIFER_HOME/agent 디렉토리에 있는 기존 설정 파일(예, w11.conf)을 복사하여 다음의 옵션 값을 수정한다.

```
agent_name = W11
agent_tcp_port = 7750
```

동일한 하드웨어에서 두개 이상의 자바 애플리케이션을 모니터링하기 위해서는, 새로운 제니퍼 에이전트 아이디를 부여하고 에이전트 TCP 리스닝 포트 번호를 달리 설정해야 한다.

그런데 모니터링해야 하는 자바 애플리케이션이 몇 개가 아닌 수십 개가 되는 경우에, 각각을 복사하여 설정 파일을 만드는 것은 상당히 번거로운 일이다. 그래서 이러한 과정을 용이하게 하기 위한 유틸리티를 제공한다.

```
confutil.sh
```

JENNIFER_HOME/agent/confutil.sh 유틸리티는 현재 디렉토리에서 w11.conf 파일을 읽어 c01.conf 파일에서 c50.conf 파일까지 50개의 설정 파일을 자동으로 생성한다.

Notice: 만약 생성되는 파일 명을 변경하거나 숫자를 수정하려면 confutil.sh 파일을 직접 수정해야 한다.

3.2. 제니퍼 에이전트 기본 설정 및 관리

제니퍼 에이전트를 설치하고 기본적으로 변경해야 하는 제니퍼 에이전트 옵션들과 설치 과정에서 발생할 수 있는 예외 현상들을 설명한다.

3.2.1. 제니퍼 에이전트 아이디

하나의 제니퍼 서버는 여러 개의 제니퍼 에이전트로부터 성능 데이터를 수집할 수 있다. 그래서 제니퍼 서버가 수집한 성능 데이터를 제니퍼 에이전트 별로 식별할 수 있도록 제니퍼 에이전트 아이디는 유일해야 한다. 이는 제니퍼 에이전트의 agent_name 옵션으로 설정한다.

```
agent_name = W11
```

Warning: 제니퍼 에이전트 아이디는 영어와 숫자만으로 구성되어야 하여, 글자 수는 반드시 3자 이어야 한다.

제니퍼 에이전트의 아이디가 중복되면 제니퍼 클라이언트에서 해당 제니퍼 에이전트가 짧은 주기로 반복해서 정지된 것으로 나타나거나 ERROR_MAYBE_GC_TIME_DELAY 경보가 발령될 수 있다. 이 경우에 중복된 제니퍼 에이전트 아이디 중의 하나를 변경하고 **[구성 관리 | 구성 설정 | 실시간 운영 관리]** 메뉴에서 [에이전트 목록]과 [실시간 데이터 목록]을 삭제하면 정상적으로 작동한다.

Notice: 제니퍼 에이전트 아이디 변경은 실시간 반영된다.

3.2.2. 로그 파일

제니퍼 에이전트의 로그 파일은 제니퍼 에이전트의 logfile 옵션으로 설정한다.

```
logfile = jennifer.log
```

기본 설정의 경우에는 WORKING_DIRECTORY/jennifer.log 파일에 로그가 기록된다. 이 로그 파일에는 애플리케이션에서 발생한 예외, 서비스 덤프, 디버그 내용 등이 기록된다.

로그 파일을 일자별로 기록하려면 제니퍼 에이전트의 enable_logfile_daily_rotation 옵션을 true로 설정한다.

```
enable_logfile_daily_rotation = true
```

제니퍼 에이전트의 logfile_encoding_characterset 옵션으로 제니퍼 에이전트 로그 파일의 텍스트 인코딩을 지정할 수도 있다.

그리고 LWST와 관련한 로그 파일은 제니퍼 에이전트의 lwst_logfile 옵션으로 설정한다.

```
lwst_logfile = lwst.log
```

기본 설정의 경우에는 WORKING_DIRECTORY/lwst.log 파일에 로그가 기록된다. 이 로그 파일에는 제니퍼 에이전트의 tx-server, tx-naming, tx-client, profile 등의 옵션으로 설정한 클래스와 메소드에 대한 정보와 LWST 디버그 내용 등이 기록된다.

3.2.3. 라이센스키 파일

제니퍼 에이전트 라이센스키는 제니퍼 에이전트의 license_filename 옵션으로 설정한 텍스트 파일에 저장된다.

```
license_filename = license.txt
```

기본 설정의 경우에는 WORKING_DIRECTORY/license.txt 파일에 라이센스키가 저장된다.

3.2.4. 라이센스키와 제니퍼 에이전트 enable

IP 주소의 변경과 유효 기간 경과 등의 이유로 라이센스키가 유효하지 않게 되거나 제니퍼 에이전트의 enable 옵션을 false로 설정하면 제니퍼 에이전트는 다음과 같이 동작한다.

자바 애플리케이션 재시작 여부와 상관없이 LWST 모듈을 제외한 모든 제니퍼 에이전트의 기능은 동작하지 않는다. 즉, 모든 데이터 수집과 전송 기능이 정지한다. 물론 자바 애플리케이션의 동작에는 영향을 주지 않는다.

그러나 자바 애플리케이션을 재시작하지 않으면 LWST 모듈은 기존과 동일하게 동작한다. 이는 LWST 모듈이 클래스 로딩 시에 바이트 코드를 변경하는 방식으로 동작하기 때문이다. 즉, 이미 로딩된 클래스에 의한 프로파일링을 포함한 LWST 기능 수행을 무효화할 수는 없다. 물론 자바 애플리케이션을 재시작하면 LWST와 관련한 기능도 모두 정지한다.

3.2.5. 네트워크 설정

제니퍼 에이전트는 수집한 성능 데이터를 제니퍼 서버에 전송한다. 다음은 이를 위한 네트워크 설정에 대한 설명이다.

3.2.5.1. TCP 네트워크 설정

제니퍼 서버가 제니퍼 에이전트의 TCP 포트로 역방향 연결을 하기도 한다. 주로 액티브 서비스 목록, 10분간 애플리케이션 처리 현황 통계, 로딩 클래스 목록, 소켓/파일 목록 등과 같이 데이터의 크기가 크고 데이터베이스나 파일에 저장되지 않는 데이터를 조회하는데 TCP 역방향 연결을 사용한다. 그래서 제니퍼 에이전트의 agent_tcp_port 옵션으로 제

니퍼 서버로부터의 TCP 요청을 받아들이는 제니퍼 에이전트의 포트 번호를 설정한다. 기본 포트 번호는 7750이다.

```
agent_tcp_port = 7750
```

그리고 제니퍼 에이전트의 agent_tcp_port 옵션으로 설정한 포트로 들어오는 제니퍼 서버의 요청을 처리하는 제니퍼 에이전트의 자바 쓰레드를 Agent TCP Worker라고 한다. Agent TCP Worker의 숫자는 제니퍼 에이전트의 number_of_tcp_workers 옵션으로 설정한다. 기본 값은 5이다.

```
number_of_tcp_workers = 5
```

TCP 네트워크 설정과 관련한 옵션을 수정하는 경우에는 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

3.2.5.2. UDP 네트워크 설정

제니퍼 에이전트는 제니퍼 서버의 UDP 포트로 성능 데이터를 전송한다. 우선 제니퍼 에이전트의 udp_server_host 옵션으로 제니퍼 서버의 IP 주소를 설정한다.

```
udp_server_host = 127.0.0.1
```

그리고 제니퍼 서버에 설정한 UDP 포트 번호와 동일하게 제니퍼 에이전트에도 해당 옵션을 설정한다.

```
server_udp_runtime_port = 6901
server_udp_listen_port = 6902
server_udp_lwst_call_stack_port = 6703
```

이 두 옵션을 수정하는 경우에는 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작할 필요가 없다.

3.2.5.3. 운영 체계별 UDP 사이즈 설정

운영 체계별로 UDP 송신 버퍼 사이즈를 설정하는 방법이다. 이를 변경하려면 시스템 root 권한이 필요하다.

IBM AIX의 경우에는 다음과 같이 UDP 송신 버퍼 사이즈를 변경한다.

```
설정 값 확인 : # no -a | grep udp_sendspace
설정 값 변경 : # no -o udp_sendspace=65535
영구 설정 값 변경 : # no -p -o udp_sendspace
```

선 솔라리스의 경우에는 다음과 같이 UDP 송신 버퍼 사이즈를 변경한다.

```
설정 값 확인 : # ndd -get /dev/udp udp_xmit_hiwat
설정 값 변경 : # ndd -set /dev/udp udp_xmit_hiwat 65535
영구 설정 값 변경 : /etc/system 혹은 /etc 아래의 RC Script 안에 ndd -set /
dev/udp udp_xmit_hiwat 65535 추가한다.
```

HP hp-ux의 경우에는 다음과 같이 UDP 송신 버퍼 사이즈를 변경한다

```
설정 값 확인 : # sysconfig -q inet udp_sendspace
설정 값 변경 : # sysconfig -i inet udp_sendspace = 65535
```

3.2.5.4. 네트워크 테스트

제니퍼 서버와 제니퍼 에이전트는 UDP와 TCP 연결을 모두 사용한다. 그런데 방화벽과 같은 네트워크 장비로 인한 제약으로 통신이 정상적으로 이루어지지 않는 경우가 있다. 이러한 문제를 해결하기 위해서 네트워크가 정상인지를 확인할 필요가 있다.

TCP 연결은 간단한 텔넷 프로그램으로 해당 포트가 열려 있는지를 확인할 수 있다.

```
telnet 127.0.0.1 7750
```

그러나 UDP 연결은 제니퍼 서버가 데이터를 올바르게 받는지를 확인해야 한다. 이를 테스트할 수 있는 JENNIFER_HOME/agent/udptest.sh 유틸리티를 제공한다. 제니퍼 에이전트를 설치한 하드웨어에서 이 유틸리티를 실행하면 UDP 연결 테스트를 위한 임의의 데이터를 제니퍼 서버에 전송한다. 이 유틸리티를 실행하려면 자바가 패스에 등록되어 있어야 한다.

```
udptest.sh [jennifer_server_ip] [port] [datalen]
ex) udptest.sh 127.0.0.1 6901 100
```

만약 제니퍼 서버의 콘솔에 다음과 같은 메시지가 출력되면 UDP 연결이 정상적임을 의미한다.

```
RECV(6901) from=127.0.0.1 data=100 bytes
```

3.2.6. 다중 프로세스에 제니퍼 설치

제니퍼 에이전트는 자바 프로세스 단위로 설정해야 한다. 각 자바 프로세스의 -Djennifer.config 옵션을 다르게 설정해야 하기 때문이다.

그러나 일부 WAS나 자바 데몬이 동일 실행 명령에 설정된 값을 이용하여 여러 개의 다중 프로세스를 실행하는 경우가 있다. 이 경우에는 각 자바 프로세스를 위한 서로 다른 -Djennifer.config 옵션을 개별적으로 설정할 수 없다.

이런 다중 프로세스 환경에 제니퍼 에이전트를 설치할 때는 공통으로 사용되는 자바 실행 옵션에 -Dconfig.auto=true를 추가로 설정한다.

```
-Dconfig.auto=true
```

제니퍼는 에이전트는 프로세스 기동시 TCP 포트를 체크하여 -Djennifer.config 옵션에 설정된 파일 이름을 토대로 적절한 제니퍼 에이전트 설정 파일을 찾아낸다.

```
-Dconfig.auto=true -Djennifer.config=x01.conf
```

만약 위와 같이 설정하면 제니퍼 에이전트는 x01.conf에서부터 파일명의 숫자(뒷2자리)를 증가 시키며 각 프로세스를 위한 제니퍼 에이전트 설정 파일을 찾아서 적용한다. 다중 프로세스를 위한 제니퍼 설정을 위한 가이드는 다음과 같다

- 예상되는 프로세스 수 만큼의 config 파일을 -Djennifer.config 옵션으로 설정한 제니퍼 에이전트 설정 파일(x01.conf)이 위치한 디렉토리에 만들어 두어야 한다.
- 기본 설정 파일(x01.conf) 이름은 2자의 숫자를 포함해야 한다.
- -Djennifer.config 옵션에는 숫자가 가장 빠른 파일을 설정한다.
- 제니퍼 에이전트 설정 파일의 이름에서 숫자는 연속되어야 하고(예를 들어, x01.conf, x02.conf ... x50.conf, x51.conf), 숫자를 제외한 나머지 이름은 동일 해야 한다.
- 각 제니퍼 에이전트 설정 파일의 agent_tcp_port와 agent_name 옵션의 값은 서로 달라야 한다.

3.2.7. 제니퍼 에이전트 임시 디렉토리 변경

제니퍼 에이전트는 제니퍼 에이전트를 설치한 운영 체계의 특정 디렉토리에 로딩된 클래스나 성능 데이터를 임시로 저장한다. 이 때 사용되는 디렉토리는 제니퍼 에이전트가 설치된 자바 애플리케이션을 수행하고 있는 WORKING_DIRECTORY이다. 이 디렉토리에는 .data와 .class라는 서브디렉토리가 만들어진다. 제니퍼 에이전트의 agent_fileroot 옵션으로 해당 디렉토리를 변경할 수 있다.

```
agent_fileroot = /jennifer/agent/data
```

Notice: 해당 디렉토리를 미리 만들어야 하며, 접근 권한이 있어야 한다. 자바 애플리케이션이 동작하는 상태에서 수정할 수는 있으나 일부 데이터가 일시적으로 손실될 수 있다. 따라서 자바 애플리케이션을 정지한 후에 설정하는 것을 권고한다.

3.2.8. 제니퍼 에이전트 설치 과정에서 발생할 수 있는 예외 현상

• X-View 프로파일에서 SQL 파라미터가 정상적으로 보이지 않는 현상

제니퍼 에이전트와 제니퍼 서버의 file.encoding이 다른 경우에 제니퍼 서버가 수집한 성능 데이터가 정상적으로 보이지 않을 수 있다. 이 경우에는 명시적으로 제니퍼 서버의 file.encoding 값을 제니퍼 에이전트의 server_encoding 옵션으로 설정한다.

```
server_encoding = UTF8
```

• 선 자바 1.5, 6.0 등에서 Native Memory Leak 발생

선 자바 1.5 혹은 6.0 등에서 Native Memory Leak 현상이 발생하면, 다음 자바 실행 옵션을 설정한다.

```
-XX:CompileCommand=exclude,org/apache/jennifer/bcel/classfile/
Attribute,readAttribute
```

• 제니퍼 설치 후 애플리케이션 쿠키 정보 유실

제니퍼 에이전트는 방문자 수와 동시단말 사용자 수를 수집하기 위해서 쿠키를 사용한다. 그런데 사용할 수 있는 쿠키의 숫자와 크기에는 제한이 있기 때문에, 기존 자바 애플리케이션에서 많은 쿠키를 사용하고 있으면 쿠키 정보가 유실될 수가 있다. 이 경우에는 제니퍼 에이전트의 hotfix_remote_address_for_wmonid 옵션을 true로 설정한다.

```
hotfix_remote_address_for_wmonid = true
```

이 옵션이 true로 설정되면 remote address를 사용자 구분으로 사용한다.

Notice: hotfix_remote_address_for_wmonid 옵션을 true로 설정하면 인트라넷 시스템에서만 방문자 수와 동시단말 사용자 수가 의미를 갖는다.

• 과도한 WARNING_JDBC_CONN_ILLEGAL_ACCESS 발생

제니퍼는 java.sql.DriverManager와 javax.sql.DataSource 클래스에서 JDBC를 모니터링하는데, 이 클래스들의 앞단에서 자바 애플리케이션이 임의의 또 다른 커넥션 풀을 사용하면 WARNING_DB_CONN_ILLEGAL_ACCESS 예외가 과도하게 발생할 수 있다. 이 문제를 해결하려면 [JDBC 커넥션 객체의 중복 할당]를 참조한다.

• 부정확한 WARNING_JDBC_UN_COMMIT_ROLLBACK 발생

부정확한 WARNING_JDBC_UN_COMMIT_ROLLBACK 예외가 발생하면 제니퍼 에이전트의 ignore_rollback_uncommited_error 옵션을 true로 설정한다.

```
ignore_rollback_uncommited_error = true
```

• 오라클 JDBC에서 Class Cast Exception이 발생

오라클 JDBC를 사용하는 과정에서 java.sql.Connection이나 java.sql.ResultSet 클래스와 관련한 java.lang.ClassCastException이 발생하면 제니퍼 에이전트의 enable_jdbc_oracle_dependency_used 옵션을 true로 설정한다.

```
enable_jdbc_oracle_dependency_used = true
```

자세한 사항은 [오라클 Dependency]를 참조한다.

• 세션 덤프 설정 후 HTTP 세션 클래스가 초기화되지 않는 경우

HTTP 세션을 모니터링하기 위해서 세션 덤프를 설정한 후에 HTTP 세션 클래스가 초기화되지 않으면, 제니퍼 에이전트의 session_class 옵션을 제거한다.

• CPU 사용률이 매우 높고, 자바 코어 덤프에 네트워크 통신과 관련한 부분이 자주 나타나는 경우

CPU 사용률이 매우 높고, 자바 코어 덤프에 네트워크 통신과 관련한 부분이 자주 나타나는 경우에는, 제니퍼 에이전트의 socket_simple_trace 옵션을 true로 설정한다.

```
socket_simple_trace = true
```

• IBM JDK 1.3에서 액티브 서비스 목록 조회시 WAITING 현상 발생

자바 애플리케이션이 IBM JDK 1.3을 사용하는 경우에 **[실시간 모니터링 | 애플리케이션]** 메뉴에서 액티브 서비스 목록을 조회할 때 WAITING 현상이 발생할 수 있다. 이는 IBM JDK 1.3에서 자바 쓰레드 개수가 많아지면, 액티브 서비스를 처리하는 자바 쓰레드의 정보를 얻는 과정에서 문제가 발생할 수 있기 때문이다.

이 경우에는 제니퍼 에이전트의 hotfix_disable_thread_active_count 옵션을 true로 설정한다. 단, 이렇게 설정하면 자바 쓰레드 정보를 확인할 수 없다.

```
hotfix_disable_thread_active_count = true
```

• JRockit 1.5.0_06에서 DriverManager 클래스의 getCallerClassLoader 메소드를 호출할 때 에러가 발생하는 현상

JRockit 1.5.0_06 버전에서 DriverManager 클래스의 getCallerClassLoader 메소드를 호출할 때 에러가 발생한다. 이 문제를 해결하기 위한 패치 파일을 제공한다. 제니퍼 에이전트를 설치할 때 -Xbootclasspath를 다음과 같이 설정한다.

```
-Xbootclasspath/p:/jennifer/agent/lwst.jrockit150_06.jar:/jennifer/
agent/lwst.boot.jar:/jennifer/agent/lwst.jdk.jar
```

lwst.jrockit150_06.jar 파일을 lwst.jdk.jar 파일보다 앞에 설정해야 한다.

3.3. 제니퍼 에이전트 이름 설정

제니퍼 에이전트 아이디는 W11과 같이 3자로 제한되어 있기 때문에, 차트를 포함한 제니퍼 클라이언트에서 제니퍼 에이전트 아이디로는 업무와 관련한 의미를 전달하는데 한계가 있다. 이 문제를 해결하려면 제니퍼 에이전트에 임의의 이름을 부여하고, 그 이름을 제니퍼 클라이언트에서 사용하면 된다.

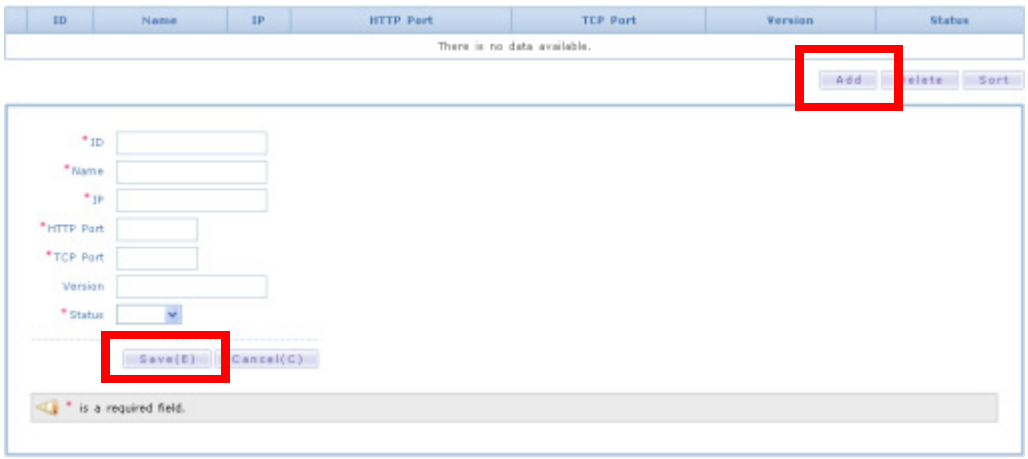
제니퍼 에이전트 이름을 부여하는 방법은 다음과 같다.

우선, **[구성 관리 | 구성 설정 | 도메인 관리]** 메뉴로 이동하여 도메인 정보를 입력한다.

Notice: 도메인은 제니퍼 서버를 의미한다. 따라서 도메인 정보에 제니퍼 서버와 관련한 정보를 입력하면 된다.

도메인 목록 하단에 있는 [추가] 버튼을 클릭하면, 도메인 입력 폼이 나타난다. 도메인 입력 폼에 내용을 입력하고 하단에 있는 [저장] 버튼을 클릭한다.

그림 3-2: 도메인 입력 화면



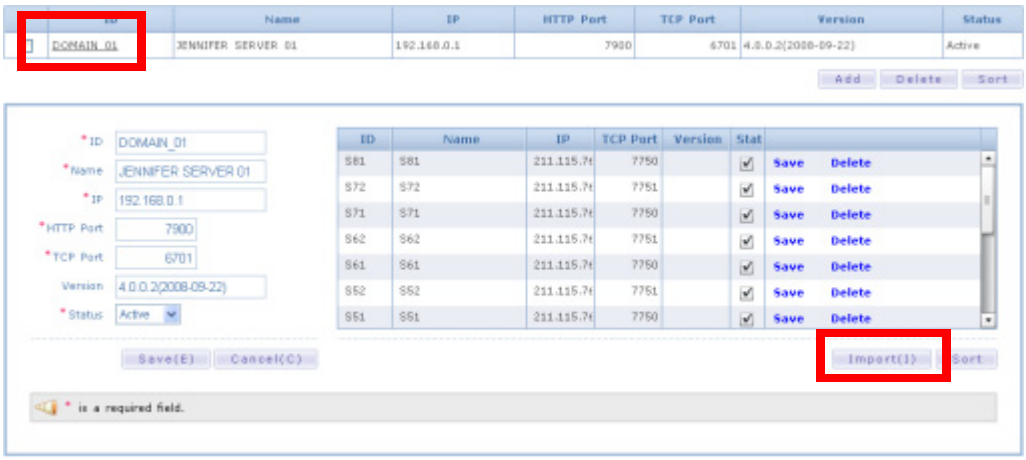
다음은 도메인 입력 폼의 필드에 대한 설명이다.

표 3-1: 도메인 입력 폼

필드	설명
아이디	해당 제니퍼 서버를 지칭하는 고유한 아이디로, 띄어쓰기 없이 영어와 숫자만을 사용하여 입력한다.
이름	해당 제니퍼 서버의 이름
IP	해당 제니퍼 서버의 IP 주소
HTTP 포트	해당 제니퍼 서버의 HTTP 포트 번호
TCP 포트	해당 제니퍼 서버의 server_tcp_port 옵션으로 설정한 포트 번호
버전	해당 제니퍼 서버의 버전으로 4.0을 입력한다. 제니퍼 4.0 이하의 버전은 도메인을 통해서 통합 할 수 없다.
상태	Inactive로 설정한 제니퍼 서버는 사용하지 않는다.

그러면 도메인 목록에 새로 추가한 도메인 정보가 나타나고, 도메인 정보의 아이디를 클릭하면 하단에 도메인 정보가 나타난다. 여기서 하단 오른쪽에 있는 [불러오기] 버튼을 누르면 제니퍼 에이전트 목록이 나타난다.

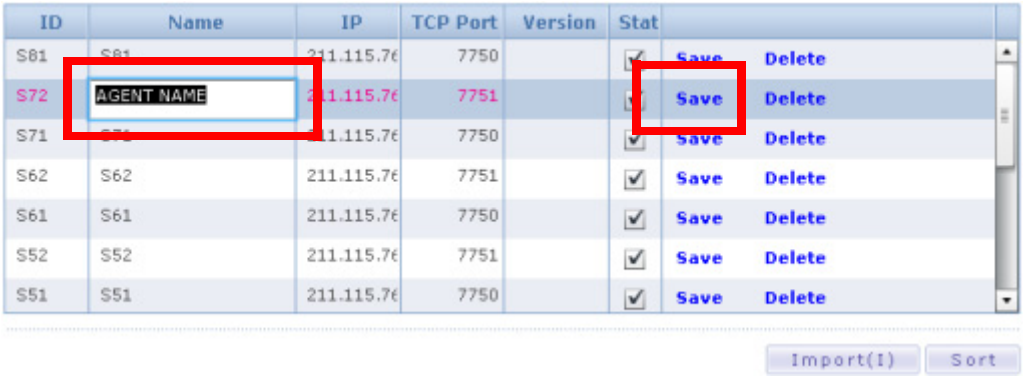
그림 3-3: 제니퍼 에이전트 불러오기



Notice: [불러오기] 버튼을 클릭하면 현재 운영 중인 제니퍼 에이전트만이 나타난다. 추후에 제니퍼 에이전트를 추가한 경우에는, **[불러오기]** 버튼을 다시 클릭하면 추가된 제니퍼 에이전트가 목록에 나타난다.

기본적으로 제니퍼 에이전트 이름은 제니퍼 에이전트 아이디와 동일하다. 제니퍼 에이전트 이름 칼럼에서 이름을 수정한 후에 [저장] 버튼을 클릭하면, 제니퍼 에이전트의 이름이 변경된다.

그림 3-4: 제니퍼 에이전트 목록



제니퍼 에이전트 목록 하단에 있는 **[정렬]** 버튼을 통해서 제니퍼 에이전트가 화면에 나타나는 순서를 설정할 수 있다. 기본 정렬 기준은 제니퍼 에이전트 아이디이다.

Notice: 다시 로그인을 해야 수정된 제니퍼 에이전트 이름이 제니퍼 클라이언트에 나타난다.

3.4. Byte Code Instrumentation

효율적인 모니터링을 하려면 모니터링을 할 자바 애플리케이션의 소스 코드를 수정하지 않고, 성능 데이터 수집에 필요한 추적 코드와 프로파일 정보 추출 코드를 자바 애플리케이션을 구성하는 클래스에 삽입할 수 있어야 한다. 이를 Byte Code Intrumentation(이하 BCI)이라 한다. 제니퍼에서 BCI를 담당하는 모듈을 Instrumentation 혹은 LWST 모듈이라고 부른다.

3.4.1. LWST 빌드와 설치

LWST 모듈에 의한 BCI는 lwst.boot.jar와 lwst.javaagent.jar 파일에 의한 런타임 처리와 오프라인에서 java.lang.ClassLoader, java.net.Socket 등의 주요 자바 클래스를 패치하여 패키지한 lwst.jdk.jar 파일에 의한 처리로 구분된다.

우선 lwst.boot.jar 파일은 LWST 코어 모듈로서 bootclasspath에 설치한다. 그리고 자바 1.5 이상에서만 사용하는 lwst.javaagent.jar 파일은 javaagent로 등록한다.

그리고 오프라인에서 java.lang.ClassLoader, java.net.Socket 등의 주요 자바 클래스를 패치하여 lwst.jdk.jar 파일로 패키지하는 것을 LWST 빌드라고 한다. LWST 빌드를 통해서 JAVA_HOME/jre/lib/rt.jar 파일이나 그와 동일한 역할을 하는 JVM 모듈을 기반으로

lwst.jdk.jar 파일을 생성한다. 그리고 LWST 빌드로 생성한 lwst.jdk.jar 파일은 bootclass-path에 설치한다.

Notice: 자바 1.4.x 이하를 사용하는 자바 애플리케이션에서는 LWST 빌드로 java.lang.ClassLoader 클래스를 패치하여 BCI를 수행한다. 하지만 자바 1.5 이상을 사용하는 자바 애플리케이션에서는 javaagent로 BCI를 수행한다. 따라서 자바 1.4.x 이하를 사용하는 자바 애플리케이션에서는 lwst.javaagent.jar 파일을 사용하지 않는다.

LWST 빌드를 하는 방법은 다음과 같다. 이 유틸리티는 JENNIFER_HOME/agent 디렉토리에 존재한다.

```
lwst40.sh [command option] [rt.jar]
```

lwst40.sh 유틸리티를 실행할때 사용하는 첫번째 파라미터를 COMMAND 옵션이라 한다. COMMAND 옵션은 대/소문자를 구분하지 않는다.

표 3-2: LWST 빌드 COMMAND 옵션

옵션	설명
JDK15	자바 1.5 이상을 위한 lwst.jdk.jar 파일을 생성한다. java.lang.ClassLoader 클래스가 패치되지 않는다. **Warning:** 만약 자바 1.5 이상에서 이 옵션을 지정하지 않으면 중복 Instrumentation이 발생한다.
AS400	AS 400 웹스피어 시스템을 위한 옵션이다. rt.jar 파일로부터 어떤 클래스도 패치하지 않고, SQL 추적을 위한 클래스들만으로 lwst.jdk.jar 파일을 만든다.
JROCKIT	BEA JROCKIT 자바 1.4.x 이하를 위한 옵션이다.
ZOS13	메인프레임 운영 체계에서 실행되는 자바 1.3.x 버전을 위한 옵션이다.
SAFE	제니퍼 에이전트를 설치한 후에, 자바 애플리케이션을 시작하는 과정에서 이상이 발생하면 이 옵션을 지정한다. LWST 빌드시에 java.lang.ClassLoader 클래스에 대해서 LWST 빌드가 바이트 코드를 변경하는 위치가 다르다.

두번째 파라미터에는 java.lang.ClassLoader, java.net.Socket 등의 주요 자바 클래스를 포함하고 있는 자바 라이브러리 파일의 위치를 지정한다. 선의 JVM을 사용하는 경우에는 주로 JAVA_HOME/lib/rt.jar 파일이다.

LWST 빌드를 통해서 JDBC, 자바 쓰레드, 클래스로더 등의 모니터링 여부를 결정할 수 있다. 이를 LWST 빌드 패치 옵션이라고 한다. 이 옵션은 lwst40.sh 파일을 직접 수정해서 설정해야 한다.

다음은 lwst40.sh 파일에서 LWST 빌드 패치 옵션과 관련한 부분이다.

```
OPT="$OPT -Dbuild_jdbc=true"
OPT="$OPT -Dbuild_classloader=true"
OPT="$OPT -Dbuild_thread=false"

OPT="$OPT -Dbuild_collection=false"
#OPT="$OPT -Dtype_collection=TRU64"
#OPT="$OPT -Dbuild_collection_map=false"
#OPT="$OPT -Dbuild_collection_list=false"

OPT="$OPT -Dbuild_file=true"
OPT="$OPT -Dbuild_socket=true"
OPT="$OPT -Dbuild_xml=true"
```

다음은 각 옵션에 대한 설명이다.

표 3-3: LWST 빌드 패치 옵션

옵션	설명
build_jdbc	JDBC와 SQL에 대한 모니터링 여부를 설정한다. 미설정시 기본 값은 false이다.
build_classloader	java.lang.ClassLoader 클래스의 패치 여부를 설정한다. 이 옵션을 false로 하면 자바 1.4.x 이하에서는 대부분의 LWST 기능이 동작하지 않는다. 미설정시 기본 값은 false이다.
build_thread	java.lang.Thread 객체의 생성에 대한 모니터링 여부를 설정한다. 자바 쓰레드와 관련한 장애가 발생하면 이 옵션을 false로 한다. 미설정시 기본 값은 false이다.
build_collection	java.util.HashMap, java.util.ArrayList 등의 자바 콜레션 클래스에 대한 모니터링 여부를 설정한다. 자바 콜레션 클래스에 대한 모니터링은 성능 저하를 발생시킬 수 있기 때문에, 자바 힙 메모리 누수의 원인 파악 등이 필요한 경우에만 이 옵션을 true로 지정한다. 미설정시 기본 값은 false이다.
build_collection_map	build_collection 옵션이 true인 경우에, java.util.HashMap, java.util.HashTable 등의 맵 형의 자료 구조 클래스에 대한 모니터링 여부를 설정한다.
build_collection_list	build_collection 옵션이 true인 경우에, java.util.ArrayList, java.util.Vector 등의 리스트 형의 자료 구조 클래스에 대한 모니터링 여부를 설정한다.
build_file	java.io.FileOutputStream, java.io.FileInputStream 등의 클래스의 패치 여부를 설정한다. 파일 IO가 많은 애플리케이션을 모니터링할 때 성능 저하가 일어나면, 이 옵션을 false로 한다. 미설정시 기본 값은 false이다.
build_socket	java.net.Socket 클래스의 패치 여부를 설정한다. 소켓을 모니터링하는 과정에서 성능 저하가 일어나면, 이 옵션을 false로 한다. 이 옵션을 true로 지정한 경우에도 제니퍼 에이전트의 socket_simple_trace 옵션으로 소켓에 대한 모니터링을 제어할 수 있다. 미설정시 기본 값은 false이다.

표 3-3: LWST 빌드 패치 옵션

옵션	설명
build_xml	자바 표준 XML 파서에 대한 모니터링 여부를 설정한다. 이 옵션을 true로 지정하면, javax.xml.parsers.SAXParser와 javax.xml.parsers.DocumentBuilder 클래스의 XML 파싱 시간을 추적할 수 있다. 미설정시 기본값은 false이다.

Notice: LWST 빌드를 하면 콘솔에 LWST 빌드 패치 옵션이 출력된다. 이 메시지를 주의 깊게 확인하도록 한다.

3.4.2. LWST 적용 과정에서 주의 사항

LWST로 프로파일 등을 수행할 때 다음의 사항에 주의한다.

• 크기가 큰 JSP 파일을 프로파일링하면 부하가 발생할 수 있다.

JSP 파일도 운영시에는 WAS에 의해서 클래스 파일로 전환되는데, 이 클래스를 LWST로 프로파일링할 수 있다. 그런데 크기가 약 2 MB 이상의 JSP 파일을 프로파일링하면 CPU 사용률이나 메모리 사용량이 높아질 수 있다.

제니퍼 에이전트에는 이를 방지하기 위한 옵션이 존재한다. 제니퍼 에이전트의 hook_class_max_size 옵션으로 설정한 크기 이상의 클래스에 대해서는 BCI를 하지 않는다. 기본 값은 1048576(1mb)이고 단위는 바이트이다.

```
hook_class_max_size = 1048576
```

만약 이 옵션으로 설정한 크기보다 큰 클래스를 프로파일링하면 LWST 로그 파일에 'too big class[class name] len nnn bytes'라는 메세지가 출력된다. BCI 부하에 대한 염려가 없는 경우에는 이 값을 늘려 줄 수 있다.

• Data Transfer Object로 java.util.ArrayList 등과 같은 콜렉션 클래스를 사용하는 경우에, 콜렉션 모니터링을 하면 부하가 발생할 수 있다.

예외적으로 보고된 현상으로 java.util.HashMap, java.util.ArrayList 클래스 등을 Data Transfer Object로 사용하고 있는 경우에, 콜렉션 모니터링을 하면 자바 가비지 콜렉션이 빈번하게 일어나는 등의 부하가 발생할 수 있다.

• 자바 1.5 이상에서 LWST 빌드 COMMAND 옵션으로 JDK15를 사용하지 않으면 중복 Instrumentation이 발생한다.

기본적으로 제니퍼는 클래스로더를 패치하여 BCI를 수행한다. 그러나 자바 1.5 이상에서는 javaagent 기능을 사용한다.

따라서 자바 1.5 이상에서 LWST 빌드를 할 때는 COMMAND 옵션으로 JDK15를 지정하여 java.lang.ClassLoader 클래스를 패치하지 않아야 한다. 만약 이렇게 하지 않으면서 javaagent 옵션에 lwst.javaagent.jar 파일을 설정하면, LWST 로그 파일에 중복 Instrumentation이 시도되었다는 경고 메세지가 기록된다.

```
[WARNING] Dupplicated instrumentation!!
```

중복 Instrumentation으로 인해 버그가 발생하지는 않지만, 자바 1.5 환경에서 기능 제약이 발생할 수 있다. 따라서 JDK15 옵션으로 LWST 빌드를 다시 하도록 한다.

그리고 어떤 방식으로 Instrumentation 모듈이 초기화가 되는지를 로그 파일에서 확인할 수 있다.

- lwst.javaagent.jar 파일에 의한 Instrumentation 모듈 초기화

```
[INF] -javaagent used!!
```

- lwst.jdk.jar 파일에 의한 Instrumentation 모듈 초기화

```
[INF] The instrumentation mode is for java14/java13!!
```

3.5. 제니퍼 에이전트 유틸리티

제니퍼 에이전트 설치 과정에서 유용하게 사용할 수 있는 유틸리티에 대해서 설명한다.

3.5.1. 클래스 FINDER

클래스 FINDER는 클래스패스에서 클래스 정보를 검색하는 유틸리티이다. 이 유틸리티는 JENNIFER_HOME/agent 디렉토리에 존재한다.

```
finder.sh [mode] [classpath] [target]
```

다음은 옵션에 대한 설명이다.

- [mode] 옵션에는 CALL, CLASS, SUPER, INTERFACE 중에 하나를 지정한다.
- [classpath] 옵션에는 jar 파일을 세미콜론(;)을 구분자로 하여 지정한다.
- [target] 옵션에는 찾고자 하는 클래스나 메소드 정보를 지정한다.

rt.jar 파일에서 java.lang.Thread 클래스의 sleep 메소드를 호출하는 클래스는 다음과 같이 검색한다.

```
finder.sh CALL /usr/java/jre/lib/rt.jar java.lang.Thread.sleep
```

rt.jar 파일에서 클래스와 상관없이 sleep 메소드를 호출하는 클래스는 다음과 같이 검색한다.

```
finder.sh CALL /usr/java/jre/lib/rt.jar *.sleep
```

rt.jar 파일에서 java.lang.Thread 클래스를 사용하는 클래스는 다음과 같이 검색한다.

```
finder.sh CALL /bin/java/jre/lib/rt.jar java.lang.Thread.*
```

rt.jar 파일에서 java.lang.Thread 클래스를 찾으려면 다음과 같이 검색한다.

```
finder.sh CLASS /usr/java/jre/lib/rt.jar java.lang.Thread
```

rt.jar 파일에서 getConnection 메소드를 가지고 있는 클래스를 찾으려면 다음과 같이 검색한다.

```
finder.sh METHOD /usr/java/jre/lib/rt.jar getConnection
```

rt.jar 파일에서 java.lang.Thread 클래스의 하위 클래스를 찾으려면 다음과 같이 검색한다.

```
finder.sh SUPER /usr/java/jre/lib/rt.jar java.lang.Thread
```

rt.jar 파일에서 java.lang.Runnable 인터페이스를 구현한 클래스를 찾으려면 다음과 같이 검색한다.

```
finder.sh INTERFACE /usr/java/jre/lib/rt.jar java.lang.Runnable
```

3.5.2. JSP 유틸리티

JENNIFER_HOME/agent/tunning 디렉토리에는 제니퍼 에이전트 설치와 관련한 오류를 파악하는데 사용할 수 있는 JSP 파일이 존재한다. 이 JSP 파일들을 WAS에 복사하여 실행시킨다.

다음은 각 JSP 파일에 대한 설명이다.

표 3-4: JSP 유틸리티

JSP파일	설명
env.jsp	WAS의 환경 변수를 조회한다. 제니퍼가 정상적으로 설치되었는지 등을 검사할 때 유용하다.
jarcheck.jsp	특정 클래스가 어느 JAR 파일로부터 로딩되었는지를 검사한다. 애플리케이션 배포가 잘못되어 클래스패스가 정상적이지 않은 경우에 활용할 수 있다.

4

제니퍼 닷넷 에이전트 운영 관리

4.1. 닷넷 에이전트 설정 파일

모니터링 수준과 방법, 네트워크 포트 번호 지정 등을 포함한 대부분의 설정은 제니퍼 에이전트 옵션을 통해서 이루어진다. web.config 파일의 appSettings/add[@key='Jennifer.FileName'] 설정에 value 속성으로 지정한 파일이 제니퍼 에이전트 설정 파일이다.

4.1.1. 설정 파일 형식

제니퍼 에이전트 옵션은 텍스트 형식의 설정 파일에 기록된다. 이 파일은 다음과 같은 특징을 갖는다.

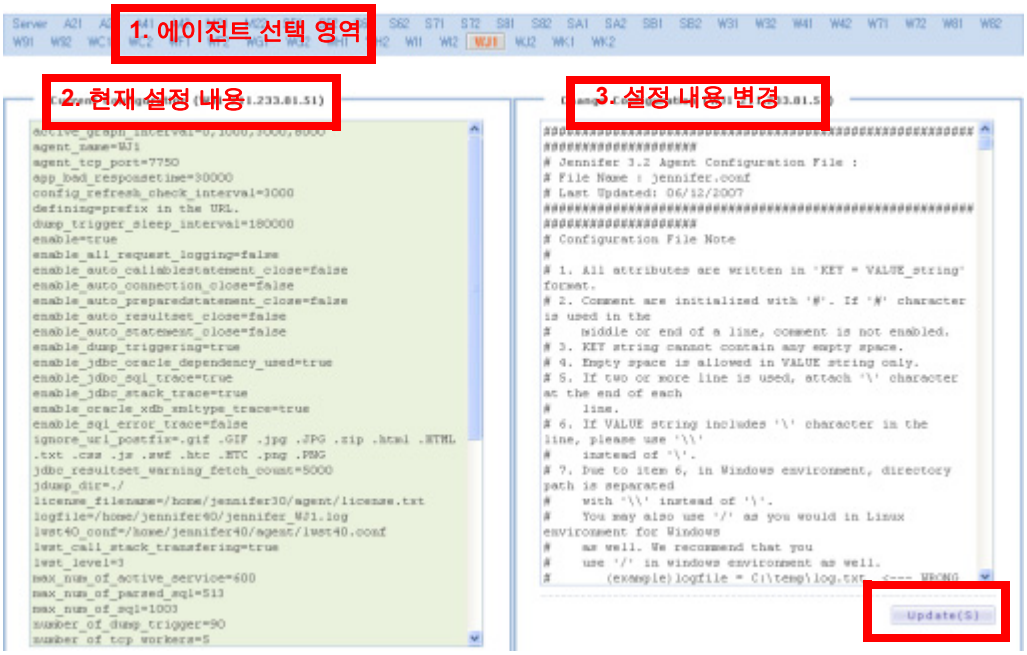
- 키 = 값 형식으로 기술한다. 키는 제니퍼 에이전트의 옵션을 의미한다.
- [#]으로 시작하는 줄은 주석이다. 그러나 문자열 중간에 있는 [#] 기호는 주석으로 인식되지 않는다.
- [=] 대신 '공백'을 키와 값의 구분자로 사용할 수 있다.
- 앞의 이유로 인하여 키에 공백을 사용하는 것을 허용하지 않는다.

- 그러나 값에는 공백을 사용할 수 있다.
- 두 줄 이상을 사용하려면 라인의 끝에 [\] 문자를 사용한다. 만약, [\] 문자 자체가 필요한 경우에는, [\]대신 [\\]을 사용한다.
- 앞의 이유로 인하여 마이크로소프트 윈도우즈 환경에서 디렉토리 구분은 [\]가 아니라 [\\]로 해야 한다. 그런데 윈도우즈 환경에서 유닉스처럼 [/]을 사용할 수 있기 때문에 윈도우즈 환경에서도 [/]을 사용하는 것을 권장한다.
- 일부 옵션을 제외한 대부분의 옵션들은 수정과 함께 실시간으로 반영된다. 수정된 옵션을 반영하기 위해 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 하는 경우에는 해당 옵션을 설명할 때 이를 명기하였다.

4.1.2. 설정 변경

설정 파일은 에디터를 통해서 직접 수정하거나, **[구성 관리 | 구성 설정]** 메뉴에서 수정할 수 있다. 단, 제니퍼 클리이언트에서는 관리자 그룹에 속한 사용자만이 옵션을 수정할 수 있다.

그림 4-1: 제니퍼 옵션 설정 화면



1. 에이전트 선택 영역 - 제니퍼 에이전트 옵션 내용을 확인하거나 수정하려면 해당 제니퍼 에이전트를 선택한다.
2. 현재 설정 내용 - 왼쪽에 있는 현재 설정 내용에서 제니퍼 에이전트 옵션을 확인한다.

3. 설정 내용 변경 - 오른쪽에 있는 설정 내용 변경 입력 폼에서 기존 옵션을 수정하거나 새로운 옵션을 추가한 후에, 하단에 있는 [수정] 버튼을 클릭한다.

4.1.3. 새로운 설정 파일 만들기

새로운 제니퍼 에이전트 설정 파일을 만들려면 JENNIFER_HOME/agent 디렉토리에 있는 기존 설정 파일(예, w11.conf)을 복사하여 다음의 옵션 값을 수정한다.

```
agent_pool =[agent1 id]:[agent1 port], [agent2 id]:[agent2 port] ……
```

동일한 하드웨어에서 두 개 이상의 애플리케이션을 모니터링 하기 위해서는, 새로운 제니퍼 에이전트 아이디를 부여하고 에이전트 TCP 리스닝 포트 번호를 달리 설정해야 한다.

예를 들어, “A” 웹 사이트에서 “app_A_pool.conf” 파일을, “B” 웹 사이트에서 “app_B_pool.conf” 파일을 만들었다고 가정할 때 개별 conf 파일에서는 아래와 같이 서로 다른 식별자와 포트를 지정해야 한다.

[app_A_pool.conf 파일]

```
agent_pool = N10:9000
```

[app_B_pool.conf 파일]

```
agent_pool = T10:9100
```

4.2. 닷넷 에이전트 기본 설정 및 관리

제니퍼 에이전트를 설치하고 기본적으로 변경해야 하는 제니퍼 에이전트 옵션들과 설치 과정에서 발생할 수 있는 예외 현상들을 설명한다

4.2.1. 제니퍼 에이전트 아이디 풀

하나의 제니퍼 서버는 여러 개의 제니퍼 에이전트로부터 성능 데이터를 수집할 수 있다. 그래서 제니퍼 서버가 수집한 성능 데이터를 제니퍼 에이전트 별로 식별할 수 있도록 제

니퍼 에이전트 아이디는 유일해야 한다. 이는 제니퍼 에이전트의 agent_pool 옵션으로 설정한다.

```
agent_pool = N10:9000
```

Notice: 제니퍼 에이전트 아이디는 영어와 숫자만으로 구성되어야 하여, 글자 수는 반드시 3자이어야 한다.

응용 프로그램이 한 개의 인스턴스(EXE)만 실행되는 경우라면 위에서와 같이 한 개의 "[식별자]:[포트번호]" 만을 지정하는 것이 가능하다. 하지만, 해당 웹 응용 프로그램이 속한 "AppPool" 의 속성에서 "Web Garden" 의 값을 2 이상으로 설정하면 그 수에 따라 w3wp.exe 프로세스가 다중으로 실행되어 웹 응용 프로그램을 호스팅 하게 된다.

이 때문에 Web Garden의 수에 따라 에이전트 아이디와 포트 번호가 대응되도록 입력해 두어야 한다. 예를 들어, 아래는 "Web Garden" 의 설정값이 2일 때 해당한다.

```
agent_pool = N10:9000,N11:9001
```

Notice: 자바 버전의 제니퍼 에이전트와는 달리 닷넷 버전의 에이전트에서는 아이디 변경은 실시간 반영되지 않는다. 즉, 변경 후 w3wp.exe가 재 시작(Recycle) 되어야 반영된다.

4.2.2. 로그 파일

제니퍼 에이전트의 로그 파일은 logfile 옵션으로 설정한다.

```
logfile = jennifer_{0}.log
```

"{0}" 포맷은 제니퍼 에이전트에서 응용 프로그램의 성격에 따라 다음과 같이 자동으로 설정한다.

```
웹 응용 프로그램: Jennifer_[App Pool 이름]_[에이전트 식별 ID]_[오늘날짜].log
일반 응용 프로그램: Jennifer_[에이전트 식별 ID]_[오늘날짜].log
```

오늘 날짜가 2010년 3월 16일인 경우, 기본적으로 JENNIFER_HOME/agent.net/log/jennifer_DefaultAppPool_N10_20100316.log 파일에 로그가 기록된다. 이 로그 파일에는 애플리케이션에서 발생한 예외, 서비스 덤프, 디버그 내용 등이 기록된다. 로그 파일을 일자 별로 기록하지 않으려면 제니퍼 에이전트의 enable_logfile_daily_rotation 옵션을 false로 설정한다.

```
enable_logfile_daily_rotation = false
```

4.2.3. 라이센스키 파일

제니퍼 에이전트 라이센스키는 제니퍼 에이전트의 license_filename 옵션으로 설정한 텍스트 파일에 저장된다.

```
license_filename = license.txt
```

기본 설정의 경우에는 JENNIFER_HOME/agent.net/license.txt 파일에 라이센스키가 저장된다.

4.2.4. 라이센스 키와 제니퍼 에이전트 활성

IP 주소의 변경과 유효 기간 경과 등의 이유로 라이센스 키가 유효하지 않게 되거나 제니퍼 에이전트의 enable 옵션을 false로 설정하면 제니퍼 에이전트는 다음과 같이 동작한다.

애플리케이션 재 시작 여부와 상관없이 모든 제니퍼 에이전트의 기능은 동작하지 않는다. 즉, 모든 데이터 수집과 전송 기능이 정지한다. (단, 애플리케이션의 요청/응답 프로파일링과 관계없는 CPU, DB 연결 수, 메모리 등의 데이터는 수집된다.) 물론 애플리케이션의 동작에는 영향을 주지 않는다.

그러나 내부적으로 여전히 프로파일링 관련 코드는 활성화되어 있다. 따라서 프로파일링 코드까지도 비활성화하고 싶다면 제니퍼 프로그램을 등록해제하고 응용 프로그램을 재시작해야 한다. 왜냐하면, 닷넷 프로파일링은 모듈이 로딩 시에 IL(Intermediate Language)코드를 변경하는 방식으로 동작하기 때문이다. 즉, 이미 로딩된 모듈은 IL코드가 변경되었으므로 변경되기 이전의 코드로 실행될 수는 없다.

4.2.5. 네트워크 설정

제니퍼 에이전트는 수집한 성능 데이터를 제니퍼 서버에 전송한다. 다음은 이를 위한 네트워크 설정에 대한 설명이다.

4.2.5.1. TCP 네트워크 설정

제니퍼 서버가 제니퍼 에이전트의 TCP 포트로 역방향 연결을 하기도 한다. 주로 액티브 서비스 목록, 10분간 애플리케이션 처리 현황 통계 등과 같이 데이터의 크기가 크고 데이터베이스나 파일에 저장되지 않는 데이터를 조회하는데 TCP 역방향 연결을 사용한다. 그

래서 제니퍼 에이전트의 agent_pool 에 에이전트 ID와 함께 포트 번호를 설정한다. (처음 설치했을 때 제공되는 app_pool.conf 설정 파일에 기본 설정된 포트는 7800번이다.)

```
agent_pool=N10:7800
```

그리고 제니퍼 에이전트 포트로 들어오는 제니퍼 서버의 요청을 처리하는 제니퍼 에이전트의 쓰레드를 Agent TCP Worker라고 한다. Agent TCP Worker의 숫자는 제니퍼 에이전트의 number_of_tcp_workers 옵션으로 설정한다. 기본 값은 5이다.

```
number_of_tcp_workers = 5
```

TCP 네트워크 설정과 관련한 옵션을 수정하는 경우에는 제니퍼 에이전트를 설치한 애플리케이션을 재시작해야 한다. .

4.2.5.2. UDP 네트워크 설정

제니퍼 에이전트는 제니퍼 서버의 UDP 포트로 성능 데이터를 전송한다. 우선 제니퍼 에이전트의 udp_server_host 옵션으로 제니퍼 서버의 IP 주소를 설정한다.

```
udp_server_host = 127.0.0.1
```

그리고 제니퍼 서버에 설정한 UDP 포트 번호와 동일하게 제니퍼 에이전트에도 해당 옵션을 설정한다.

```
server_udp_runtime_port = 6901
server_udp_listen_port = 6902
server_udp_lwst_call_stack_port = 6703
```

이 두 옵션을 수정하는 경우에는 제니퍼 에이전트를 설치한 애플리케이션을 재시작할 필요가 없다.

4.2.5.3. 네트워크 테스트

제니퍼 서버와 제니퍼 에이전트는 UDP와 TCP 연결을 모두 사용한다. 그런데 방화벽과 같은 네트워크 장비로 인한 제약으로 통신이 정상적으로 이루어지지 않는 경우가 있다. 이러한 문제를 해결하기 위해서 네트워크가 정상인지를 확인할 필요가 있다.

TCP 연결은 간단한 텔넷 프로그램으로 해당 포트가 열려 있는지를 확인할 수 있다.

```
telnet 127.0.0.1 7750
```

그러나 UDP 연결은 제니퍼 서버가 데이터를 올바르게 받는지를 확인해야 한다. 이를 테스트할 수 있는 JENNIFER_HOME/agent.net/Utility/udptest.exe 유틸리티를 제공한다. 제

니퍼 에이전트를 설치한 하드웨어에서 이 유틸리티를 실행하면 UDP 연결 테스트를 위한 임의의 데이터를 제니퍼 서버에 전송한다.

```
udptest.sh [jennifer_server_ip] [port] [datalen]
ex) udptest.sh 127.0.0.1 6901 100
```

만약 제니퍼 서버의 콘솔에 다음과 같은 메시지가 출력되면 UDP 연결이 정상적임을 의미한다.

```
RECV(6901) from=127.0.0.1 data=100 bytes
```

4.2.6. 제니퍼 에이전트 설치 과정에서 발생할 수 있는 예외 현상

- **제니퍼 설치 후 애플리케이션 쿠키 정보 유실**

제니퍼 에이전트는 방문자 수와 동시단말 사용자 수를 수집하기 위해서 쿠키를 사용한다. 그런데 사용할 수 있는 쿠키의 숫자와 크기에는 제한이 있기 때문에, 기존 웹 애플리케이션에서 많은 쿠키를 사용하고 있으면 쿠키 정보가 유실될 수가 있다. 이 경우에는 제니퍼 에이전트의 hotfix_remote_address_for_wmonid 옵션을 true로 설정한다.

```
hotfix_remote_address_for_wmonid = true
```

이 옵션이 true로 설정되면 remote address를 사용자 구분으로 사용한다.

Notice: hotfix_remote_address_for_wmonid 옵션을 true로 설정하면 인트라넷 시스템에서만 방문자 수와 동시단말 사용자 수가 의미를 갖는다.

4.3. 제니퍼 에이전트 이름 설정

제제니퍼 에이전트 아이디는 N10과 같이 3자로 제한되어 있기 때문에, 차트를 포함한 제니퍼 클라이언트에서 제니퍼 에이전트 아이디로는 업무와 관련한 의미를 전달하는데 한계가 있다. 이 문제를 해결하려면 제니퍼 에이전트에 임의의 이름을 부여하고, 그 이름을 제니퍼 클라이언트에서 사용하면 된다.

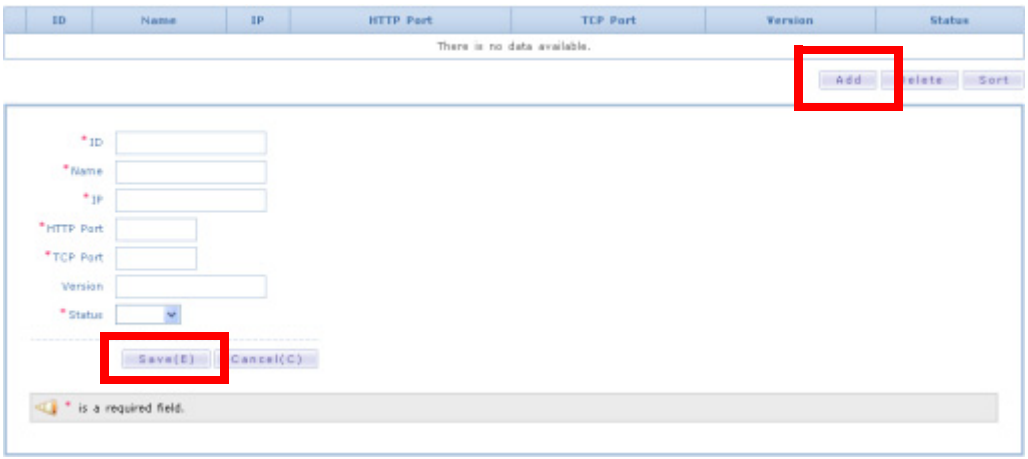
제니퍼 에이전트 이름을 부여하는 방법은 다음과 같다.

우선, **[구성 관리 | 구성 설정 | 도메인 관리]** 메뉴로 이동하여 도메인 정보를 입력한다.

Notice: 도메인은 제니퍼 서버를 의미한다. 따라서 도메인 정보에 제니퍼 서버와 관련한 정보를 입력하면 된다.

도메인 목록 하단에 있는 [추가] 버튼을 클릭하면, 도메인 입력 폼이 나타난다. 도메인 입력 폼에 내용을 입력하고 하단에 있는 [저장] 버튼을 클릭한다.

그림 4-2: 도메인 입력 화면



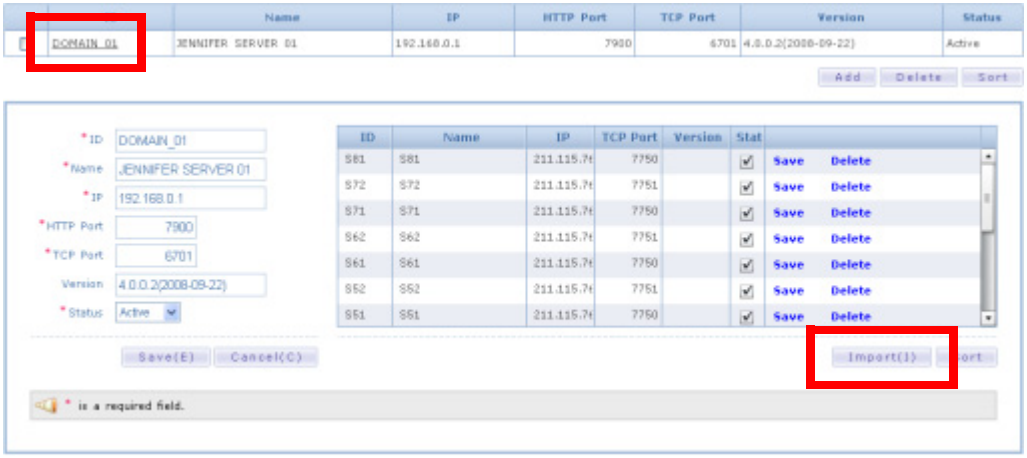
다음은 도메인 입력 폼의 필드에 대한 설명이다.

표 4-1: 도메인 입력 폼

필드	설명
아이디	해당 제니퍼 서버를 지칭하는 고유한 아이디로, 띄어쓰기 없이 영어와 숫자만을 사용하여 입력한다.
이름	해당 제니퍼 서버의 이름
IP	해당 제니퍼 서버의 IP 주소
HTTP 포트	해당 제니퍼 서버의 HTTP 포트 번호
TCP 포트	해당 제니퍼 서버의 server_tcp_port 옵션으로 설정한 포트 번호
버전	해당 제니퍼 서버의 버전으로 4.0을 입력한다. 제니퍼 4.0 이하의 버전은 도메인을 통해서 통합 할 수 없다.
상태	Inactive로 설정한 제니퍼 서버는 사용하지 않는다.

그러면 도메인 목록에 새로 추가한 도메인 정보가 나타나고, 도메인 정보의 아이디를 클릭하면 하단에 도메인 정보가 나타난다. 여기서 하단 오른쪽에 있는 [불러오기] 버튼을 누르면 제니퍼 에이전트 목록이 나타난다.

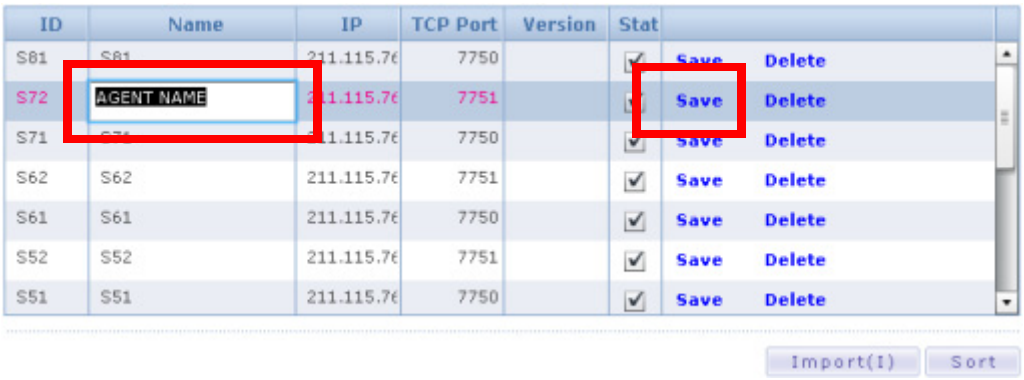
그림 4-3: 제니퍼 에이전트 불러오기



Notice: [불러오기] 버튼을 클릭하면 현재 운영 중인 제니퍼 에이전트만이 나타난다. 추후에 제니퍼 에이전트를 추가한 경우에는 **[불러오기]** 버튼을 다시 클릭하면 추가된 제니퍼 에이전트가 목록에 나타난다.

기본적으로 제니퍼 에이전트 이름은 제니퍼 에이전트 아이디와 동일하다. 제니퍼 에이전트 이름 칼럼에서 이름을 수정한 후에 [저장] 버튼을 클릭하면 제니퍼 에이전트의 이름이 변경된다.

그림 4-4: 제니퍼 에이전트 목록



제니퍼 에이전트 목록 하단에 있는 **[정렬]** 버튼을 통해서 제니퍼 에이전트가 화면에 나타나는 순서를 설정할 수 있다. 기본 정렬 기준은 제니퍼 에이전트 아이디이다.

Notice: 다시 로그인을 해야 수정된 제니퍼 에이전트 이름이 제니퍼 클라이언트에 나타난다.

4.4. 제니퍼 에이전트 유틸리티

제니퍼 에이전트 설치 과정에서 유용하게 사용할 수 있는 유틸리티에 대해서 설명한다.

4.4.1. 유틸리티

JENNIFER_HOME/agent.net/sample, JENNIFER_HOME/agent.net/utility 디렉토리에는 제니퍼 에이전트 설치와 관련한 오류를 파악하는데 사용할 수 있는 유틸리티 파일이 존재한다. 확장자가 aspx 인 경우에는 IIS 의 웹 응용 프로그램 폴더에 복사하여 실행시킨다.

다음은 각각의 유틸리티 파일에 대한 설명이다.

표 4-2: JSP 유틸리티

파일	설명
env.aspx	IIS가 실행중인 컴퓨터의 환경 변수를 조회한다. 제니퍼가 정상적으로 설치되었는지 등을 검사할 때 유용하다.
comTest.aspx	제니퍼 닷넷 에이전트의 프로파일러 모듈이 활성화되었는지 검사한다.
ipchecker.exe	실행 중인 컴퓨터의 IP 목록을 보여준다. 라이선스 키를 받을 때 알려주어야 할 IP를 구할 수 있다.
udptest.exe	제니퍼 서버가 설치된 컴퓨터로의 UDP 포트가 열려 있는지 확인할 수 있다.

5

사용자 인터페이스

5.1. 로그인

웹 브라우저의 주소 창에 제니퍼 서버를 설치한 서버의 IP 혹은 도메인 주소와 포트 번호로 구성된 다음의 URL을 입력하면 로그인 화면이 나타난다. 제니퍼의 모든 기능은 로그인 후에만 사용할 수 있다. 제니퍼 서버의 기본 포트 번호는 7900이다.

```
http://jennifer_server_ip:7900
```

그림 5-1: 로그인 화면

로그인을 하려면 아이디와 패스워드를 입력하고 **[Login]** 버튼을 클릭한다. 초기 관리자 계정의 아이디는 admin이며 패스워드도 admin이다. 처음 로그인을 한 후에 패스워드를 수정하는 것을 권장한다.

로그인 화면 오른쪽에 [News & Event]가 나타난다. 이를 숨기려면 제니퍼 서버의 ui_hide_news 옵션을 true로 설정한다.

```
ui_hide_news = true
```

로그인에 성공하면 기본적으로는 **[대시보드 | 제니퍼 대시보드]** 메뉴로 이동한다. 로그인 후에 나타나는 첫번째 화면은 사용자와 그룹에 따라서 다르게 설정할 수 있다.

5.2. 클라이언트 설정

제니퍼 클라이언트는 웹에 기반한다. 따라서 사용자는 웹 브라우저를 통해서 제니퍼 사용자 인터페이스에 접근한다. 그리고 차트는 자바 애플릿으로 구현되어 있기 때문에 자바

플러그인이 필요하고, 그리드는 플래시로 구현되어 있기 때문에 플래시 플레이어가 필요하다.

5.2.1. 운영 체계

제니퍼 클라이언트는 마이크로소프트 윈도우즈 XP와 비스타를 지원한다. 윈도우즈 XP를 사용하는 경우에 테마로 Windows 고전을 사용하면 BLACK 스타일이 정상적으로 나타나지 않는다. 현재 사용하고 있는 테마는 바탕화면에서 오른쪽 마우스를 클릭하면 나타나는 컨텍스트 메뉴에서 [속성] 메뉴를 선택하면 확인할 수 있다.

Notice: 공식적으로 제니퍼 클라이언트는 리눅스와 맥 운영 체계 등을 지원하지 않는다. 그러나 파이어 폭스 3.0 이상과 자바 플러그인 1.6.0_10 이상을 설치할 수 있는 운영 체계라면 제니퍼를 사용하는데 큰 문제는 없을 것이다.

5.2.2. 웹 브라우저

마이크로소프트 IE 7.0 혹은 IE 8.0 그리고 모질라 파이어 폭스 3.0 등의 웹 브라우저를 지원한다.

Notice: IE 6.0은 CSS를 포함한 웹 표준 지원이 미흡하고 투명한 PNG 이미지 파일을 처리하지 못하는 단점을 가지고 있다. 따라서 IE 6.0은 지원하지 않는다. IE 7.0이나 IE 8.0으로 업그레이드 할 수 없는 환경에서는 파이어폭스 3.0을 사용하는 것을 권장한다.

제니퍼를 사용하려면 쿠키와 자바 스크립트를 사용할 수 있도록 웹 브라우저를 설정해야 한다. 대부분의 웹 브라우저는 기본적으로 이 기능들을 사용하도록 설정되어 있다.

5.2.3. 자바 플러그인

제니퍼 클라이언트의 일부는 자바 애플릿으로 구현되어 있기 때문에 선 자바 플러그인 1.6.0_10 이상을 설치해야 한다.

Warning: 제니퍼 4.0부터는 마이크로소프트 자바 VM을 지원하지 않는다.

5.2.3.1. 설치와 제거

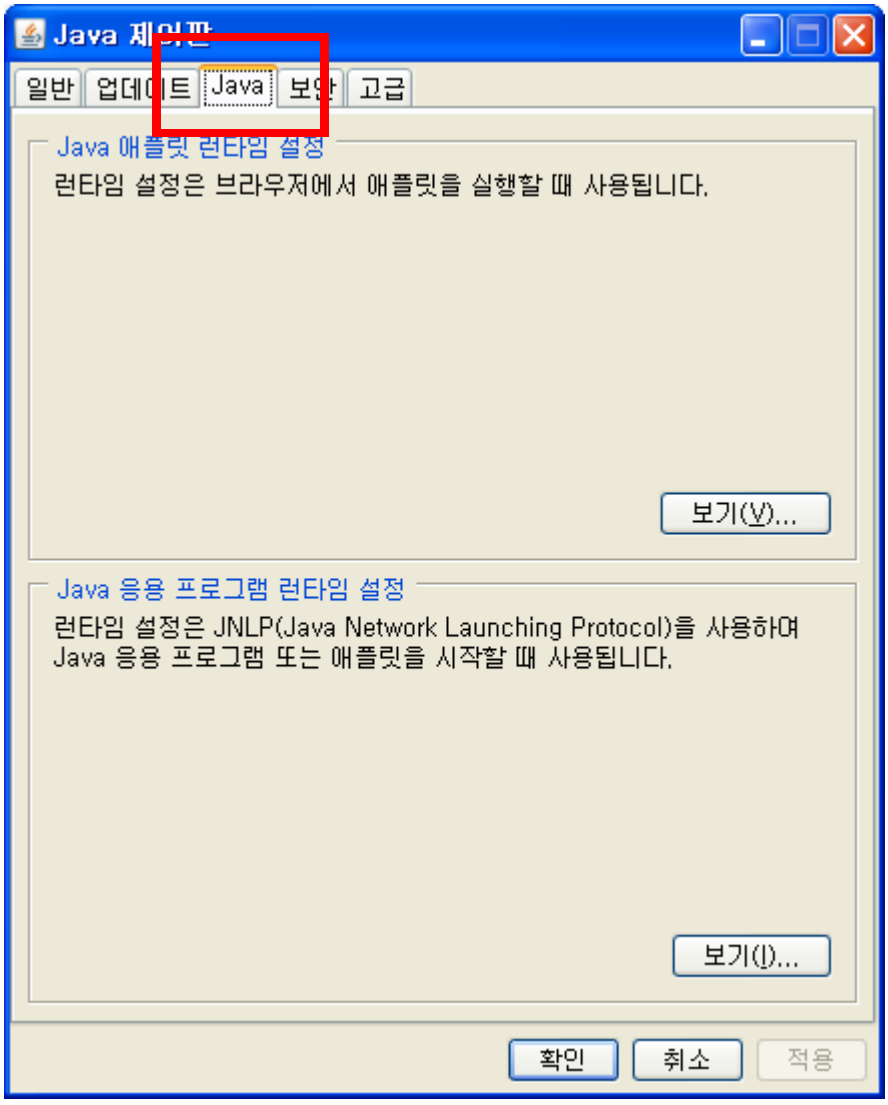
자바 플러그인을 설치하고 제거하는 것은 선의 자바 사이트(http://java.sun.com/javase/downloads/index.jsp)를 참고한다.

5.2.3.2. 메모리 설정

미설정시 자바 애플릿이 사용할 수 있는 최대 자바 힙 메모리는 96 MB(자바 1.6.0_6 업데이트 6까지는 64MB)이다. 제니퍼 클라이언트를 안정적으로 사용하려면 자바 힙 메모리 최대 값과 최소 값을 지정하여야 한다. 특히 에이전트의 숫자가 많거나 업무 처리량이 많은 경우에는 자바 힙 메모리 최대 값을 높게 설정할 필요가 있다. 윈도우즈 운영 체계에서 자바 플러그인의 자바 힙 메모리 크기는 다음과 같이 설정한다.

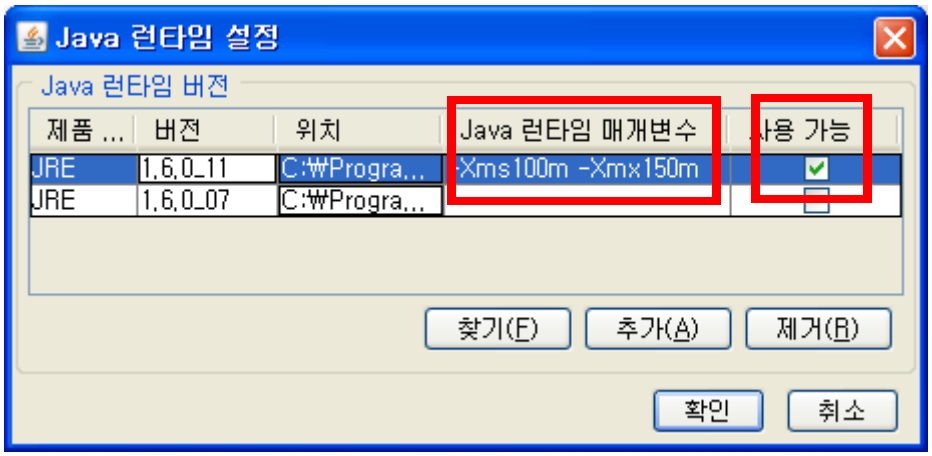
1. 제어판으로 이동한다.
2. 제어판에서 자바를 클릭하면 Java 제어판이 나타난다. 여기에서 Java 탭을 선택한다.

그림 5-2: Java 제어판



3. Java 애플릿 런타임 설정 영역의 보기 버튼을 클릭하면 Java 런타임 설정 화면이 나타난다. 여러 개의 자바를 설치한 경우에는 [사용 가능] 칼럼을 통해서 사용할 자바를 선택한다.

그림 5-3: Java 런타임 설정

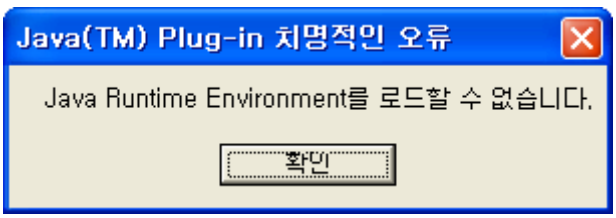


4. 예를 들어, 자바 힙 메모리 최소 값을 100 MB, 최대 값을 150 MB로 설정하려면 JRE의 Java 런타임 매개 변수 칼럼에 다음과 같이 입력한다.

```
-Xms100m -Xmx150m
```

설정 가능한 자바 힙 메모리 최대 값은 컴퓨터 사양과 환경에 영향을 받는다. 설정 가능한 값보다 큰 값을 지정하면 다음과 같은 오류가 발생하거나 웹 브라우저가 비정상적으로 종료될 수 있다.

그림 5-4: 과도한 자바 힙 메모리 최대 값 설정에 따른 오류



그런데 설정 가능한 자바 힙 메모리 최대 값에 대한 명확한 규정이 없기 때문에 사용자는 시행 착오를 통해서 자바 힙 메모리 최대 값을 설정해야 한다.

5.2.3.3. CPU 사용률 최적화

작업 관리자의 프로세스 탭에서 java.exe라는 이름을 갖는 이미지로 자바 애플릿 CPU 사용률을 확인할 수 있다.

CPU 사용률은 제니퍼 대시보드 메뉴를 열어 놓은 상태에서 확인한다.

몇 가지 옵션으로 자바 애플릿이 사용하는 CPU 사용률을 최적화할 수 있다. 다음 각 옵션 중에서 하나만을 설정한 상태에서 CPU 사용률을 점검한다. 옵션 설정 방법은 자바 힙 메모리를 설정하는 방법과 동일하다.

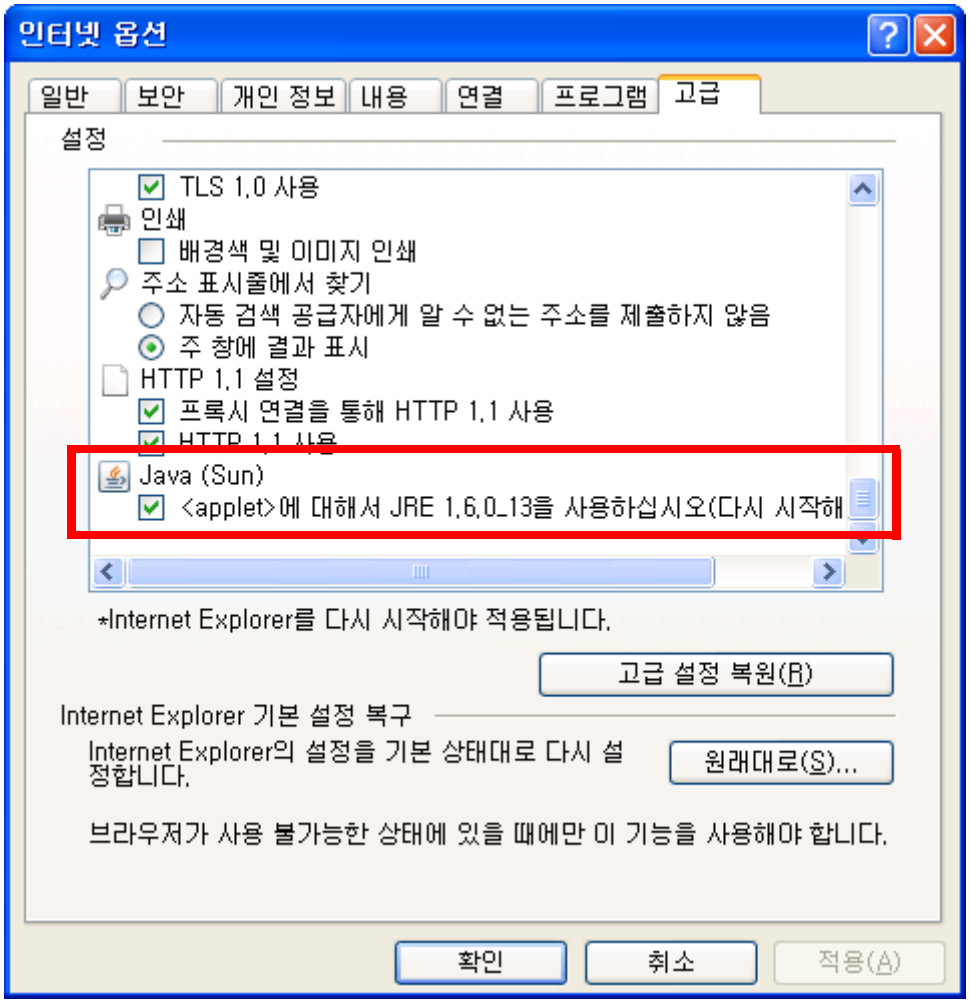
```
-Dsun.java2d.d3d=false
-Dsun.java2d.ddoffscreen=false
-Dsun.java2d.noddraw=true
```

최종적으로 가장 적은 CPU 사용률을 나타내는 옵션만을 설정한다. 하드웨어에 따라서는 옵션을 설정하지 않는 것이 CPU 사용률이 가장 적을 수도 있다.

5.2.3.4. 웹 브라우저 설정

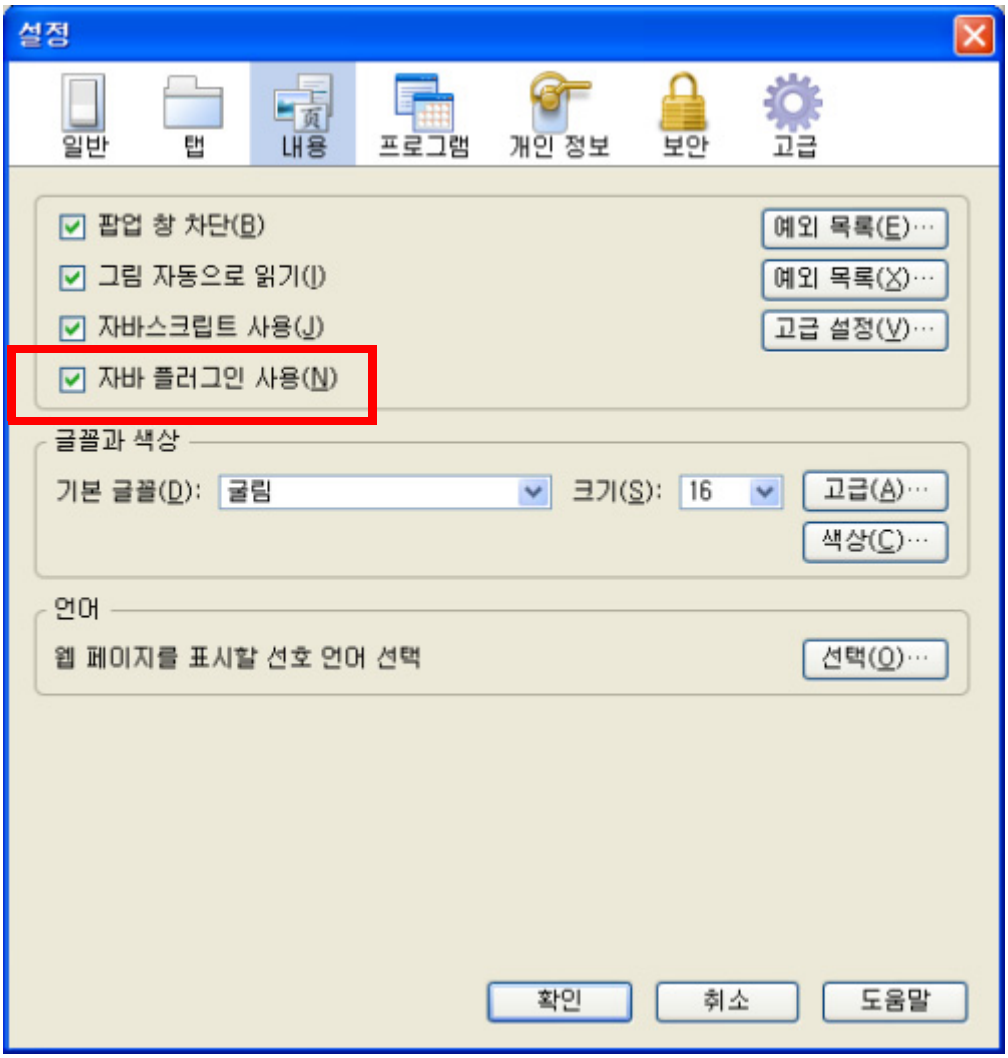
마이크로소프트 IE 7.0 혹은 IE 8.0을 사용하는 경우에는 **[도구 | 인터넷 옵션]** 메뉴에서 자바 플러그인 사용 여부를 설정한다. 인터넷 옵션 창에서 고급 탭을 선택한 후에 맨 하단으로 내려가면 설치된 자바 플러그인 정보가 나타난다. 여기서 선 자바 플러그인을 사용하도록 설정한다.

그림 5-5: IE 8.0 설정



모질라 파이어 폭스 3.0을 사용하는 경우에는 **[도구 | 설정]** 메뉴에서 자바 플러그인 사용 여부를 설정한다. 설정 화면에서 내용 탭을 선택한 후에 자바 사용 옵션을 선택한다. 기본으로 선택되어 있다.

그림 5-6: 파이어 폭스 설정



5.2.3.5. 플래시 플레이어

제니퍼 클라이언트의 일부는 플래시로 구현되어 있기 때문에 어도비 플래시 플레이어 9.0 이상을 설치해야 한다.

5.3. 사용자 인터페이스 구조

제니퍼 사용자 인터페이스는 크게 상단 영역, 툴바 영역, 보드 영역, 메인 영역 등으로 구성 된다.

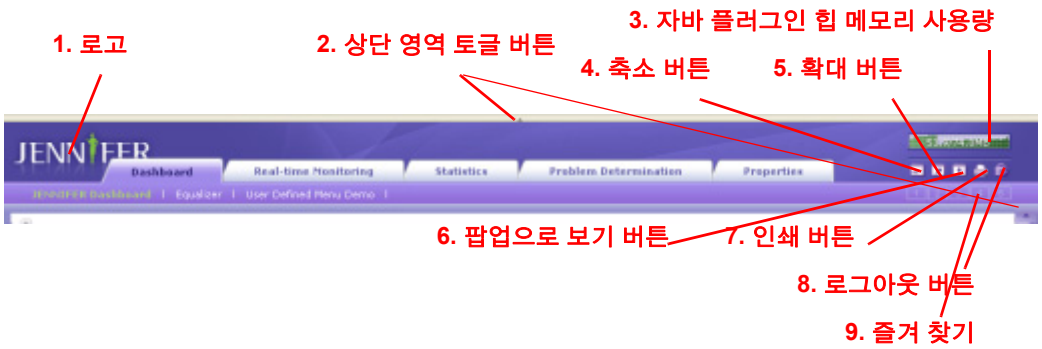
표 5-1: 사용자 인터페이스 구조

영역	설명
상단 영역	메뉴와 공통 기능 버튼 등을 제공
툴바 영역	각종 도구와 새 소식 등을 제공
보드 영역	실시간 경보 내역과 서비스 현황 정보 등을 제공
메인 영역	메뉴 선택에 따른 다양한 정보 제공

5.3.1. 상단 영역

상단 영역의 대부분을 차지하는 메뉴는 상중하 3단계로 구성된다. 제니퍼가 제공하는 대부분의 기능은 이 메뉴를 통해서 접근한다.

그림 5-7: 상단 영역



상단 영역의 주요 기능은 다음과 같다.

1. 로고 - 초기 설정은 제니퍼 로고 이미지로 되어 있으나 다른 로고 이미지로 변경할 수 있다. 변경 방법은 [로고 및 타이틀 변경(77 페이지)]을 참조한다.
2. 상단 영역 토글 버튼 - 상단 영역 토글 버튼을 클릭하면 상단 영역이 사라진다. 반대로 상단 영역이 숨겨진 상태에서 상단 영역 토글 버튼을 클릭하면 상단 영역이 다시 나타난다.
3. 자바 플러그인 힙 메모리 사용량 차트와 컨텍스트 메뉴 - 자바 플러그인 힙 메모리 사용량을 보여준다. 이 차트를 클릭하면 컨텍스트 메뉴가 나타나고 이 메뉴를 통해서 다양한 기능(언어 변경, 스타일 변경, 가비지 콜렉션, 임시 파일 지우기, 스크린 캡쳐를 게시판에 저장 등)을 수행할 수 있다.
4. 축소 버튼 - 이 버튼을 클릭하면 메인 영역의 넓이가 10% 축소된다. 기본 크기의 10% 이하로 축소할 수는 없다.
5. 확대 버튼 - 이 버튼을 클릭하면 메인 영역의 넓이가 10% 확대된다. 최대 크기에는 제한이 없다.

6. 팝업으로 보기 버튼 - 현재 화면을 팝업 창으로 열 때 사용한다. 팝업 창에서는 상단 영역이 나타나지 않는다. 기본적으로 동일한 팝업 창을 사용하는데 Shift 키를 누른 상태에서 팝업 창으로 보기 버튼을 클릭하면 항상 새로운 창을 사용한다.
7. 인쇄 버튼 - 현재 화면 내용을 인쇄한다. 대시보드를 올바르게 인쇄하려면 마이크로소프트 IE 6.0 혹은 IE 7.0을 사용해야 한다.
8. 로그아웃 버튼 - 로그아웃을 하려면 해당 버튼을 클릭한다. 로그아웃 후에는 로그인 화면으로 이동한다.
9. 즐겨 찾기 - 자주 사용하는 메뉴를 즐겨 찾기하는데 사용한다.

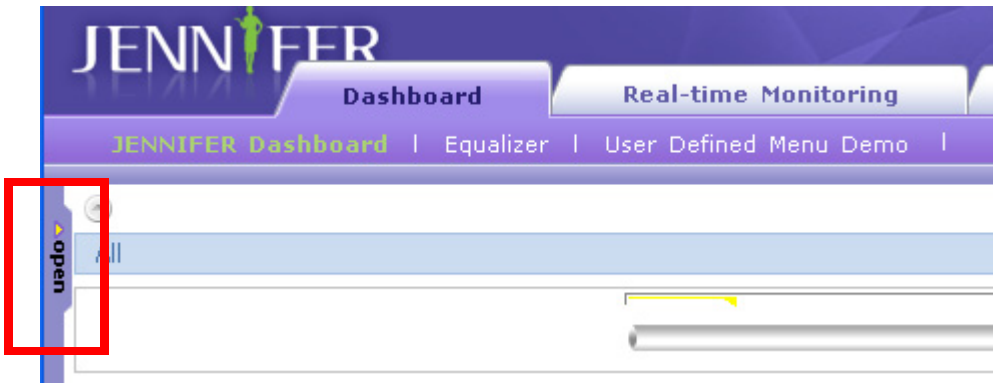
제니퍼 서버의 ui_hide_header 옵션을 true로 설정하면 로그인 시에 상단 영역이 숨겨진 상태로 표시된다.

```
ui_hide_header = true
```

5.3.2. 툴바 영역

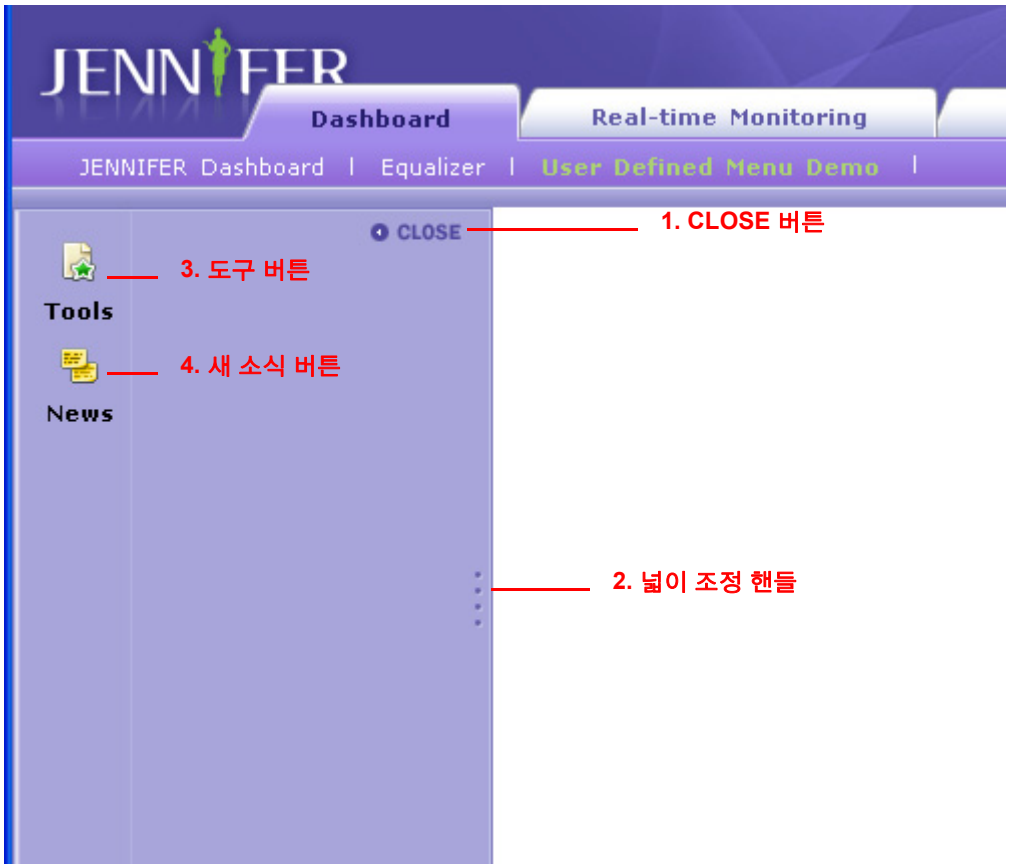
툴바 영역은 각종 도구와 새 소식 등을 제공한다.

그림 5-8: 툴바 영역 열기



[open] 버튼을 클릭하면 툴바 영역이 나타난다.

그림 5-9: 툴바 영역



1. CLOSE 버튼 - CLOSE 버튼을 클릭하면 툴바 영역이 사라진다.
2. 넓이 조정 핸들 - 툴바 영역의 넓이를 조정할 때 사용한다.
3. 도구 버튼 - 도구 버튼을 클릭하면 도구 목록이 나타난다.
4. 새 소식 버튼 - 새 소식 버튼을 클릭하면 제니퍼소프트가 제공하는 새 소식이 나타난다.

다음은 도구 버튼을 클릭하면 나타나는 기능에 대한 설명이다.

- Server Control Center - 네트워크 설정, 자바 힙 메모리 사용량, X-View 처리 상태, 대기 시간 기준 등의 제니퍼 서버의 주요 정보를 보여준다.
- 경보 목록 - 금일 경보 내역, 최근 경보 내역 등을 보여주고, 과거 경보 내역을 조회할 수 있는 기능을 제공한다.
- 쿼리 수행기 - 제니퍼 서버가 사용하는 데이터베이스에 대해서 SQL을 수행할 수 있는 기능을 제공한다.
- 쿼리 빌드 - 임의의 SQL에 대해서 바인딩 파라미터와 상수 파라미터를 결합할 수 있는 기능을 제공한다.

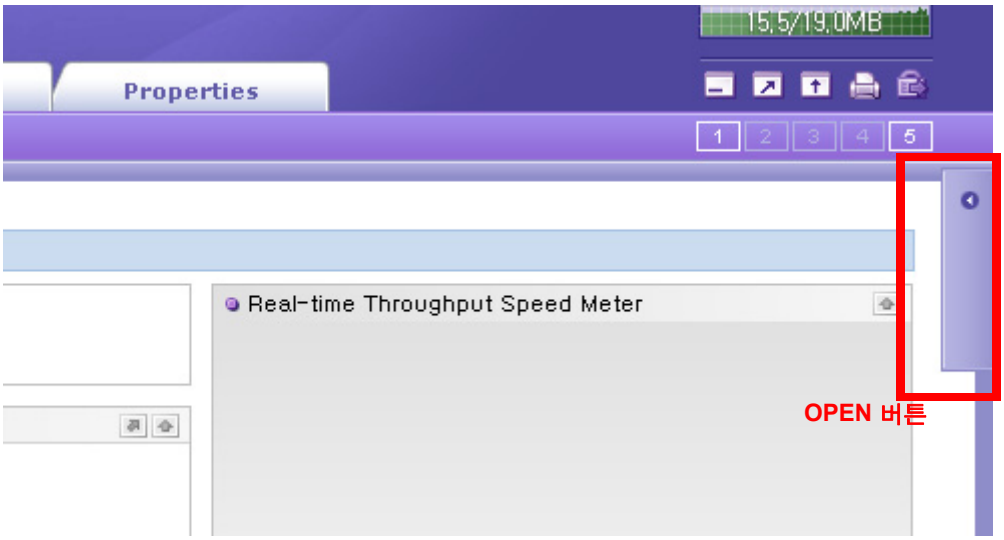
- REMON 목록 - REMON이 제니퍼 서버에 전송하는 REMON 데이터 목록을 보여준다.
- 테이블 스페이스 정리 - 제니퍼 서버의 데이터베이스로 아파치 더비를 사용하는 경우에만 제공한다. 이를 통해서 테이블 스페이스를 정리할 수 있다.
- 디버그 - 파일에 저장되어 있는 성능 데이터를 디버그 목적으로 조회할 수 있는 기능을 제공한다.
- 통계 데이터 요약 - 특정 날짜의 통계 데이터를 요약할 수 있는 기능을 제공한다.

5.3.3. 보드 영역

보드 영역은 실시간 경보 내역, 금일 서비스 현황, 실시간 서비스 현황 등의 정보를 제공한다. 보드 영역은 웹 브라우저 오른쪽에 숨어 있다가 장애가 발생하면 경고음과 함께 자동으로 나타난다. 보드 영역으로 모니터링하는 자바 애플리케이션의 전반적인 상황을 파악할 수 있다.

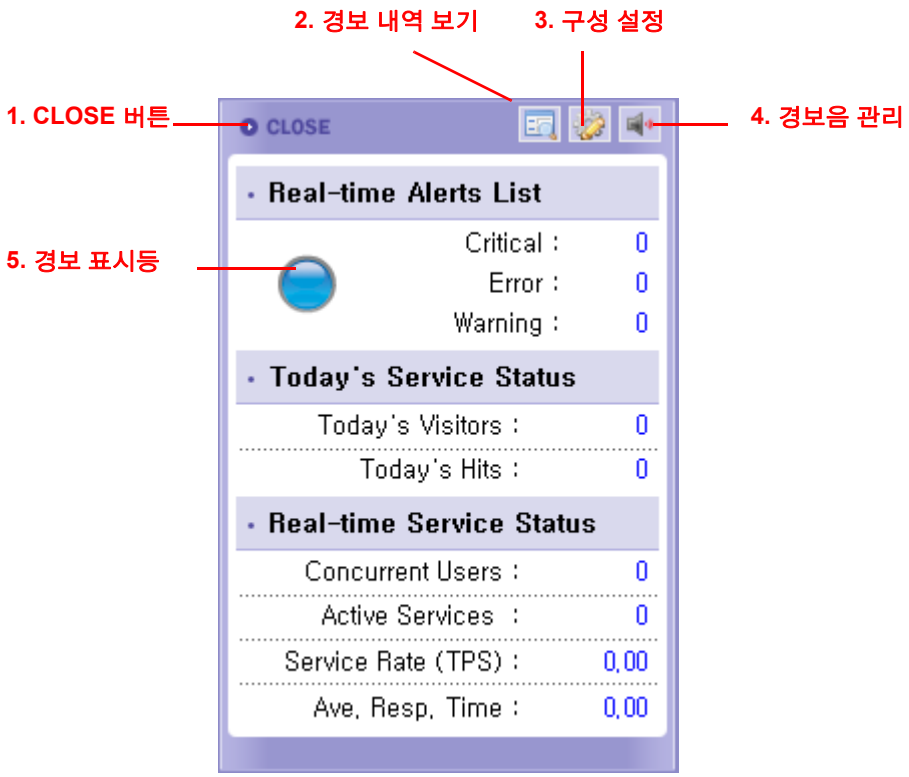
Notice: board 권한을 가지고 있는 그룹에 속한 사용자에게만 보드 영역이 보인다.

그림 5-10: 보드 영역 열기



OPEN 버튼을 클릭하면 웹 브라우저 오른쪽에 숨겨져 있는 보드 영역이 나타난다.

그림 5-11: 보드 영역



1. CLOSE 버튼 - CLOSE 버튼을 클릭하면 보드 영역이 사라진다.
2. 경보 내역 보기 - 상세한 경보 내역을 확인하려면 이 버튼을 클릭한다.
3. 구성 설정 - 경보는 심각, 에러, 경고 등의 유형으로 구분되는데 기본적으로 경보가 발생하면 경보 유형과 상관 없이 보드 영역이 자동으로 나타난다. 만약 특정 유형의 경보가 발생한 경우에만 보드 영역이 나타나게 하려면, 이 버튼을 클릭하거나 보드 영역에서 오른쪽 마우스 버튼을 클릭하면 나타나는 컨텍스트 메뉴를 이용해서 옵션을 변경한다.
4. 경보음 관리 - 이 버튼을 클릭하면 경보음이 들리지 않는다. 경보음을 들리도록 하려면 이 버튼을 다시 클릭한다.
5. 경보 표시등 - 정상적인 상황에서는 경보 표시등의 색은 푸른색이고 장애가 발생하면 경고 표시등의 색이 빨간색으로 변경된다. 경보 표시등의 색이 빨간색일 때 경보 표시등을 클릭하면 경보 표시등의 색이 푸른색으로 변경된다.

실시간 경보 내역은 다음과 같다.

표 5-2: 실시간 경보 내역

항목	설명
심각(Critical)	금일 발생한 심각 유형의 경보 건수

표 5-2: 실시간 경보 내역

항목	설명
에러 (Error)	금일 발생한 에러 유형의 경보 건수
경고(Warning)	금일 발생한 경고 유형의 경보 건수

금일 서비스 현황은 다음과 같다.

표 5-3: 금일 서비스 현황

항목	설명
금일 방문자 수	금일 방문자 수
금일 호출 건수	금일 호출 건수

실시간 서비스 현황은 다음과 같다.

표 5-4: 실시간 서비스 현황

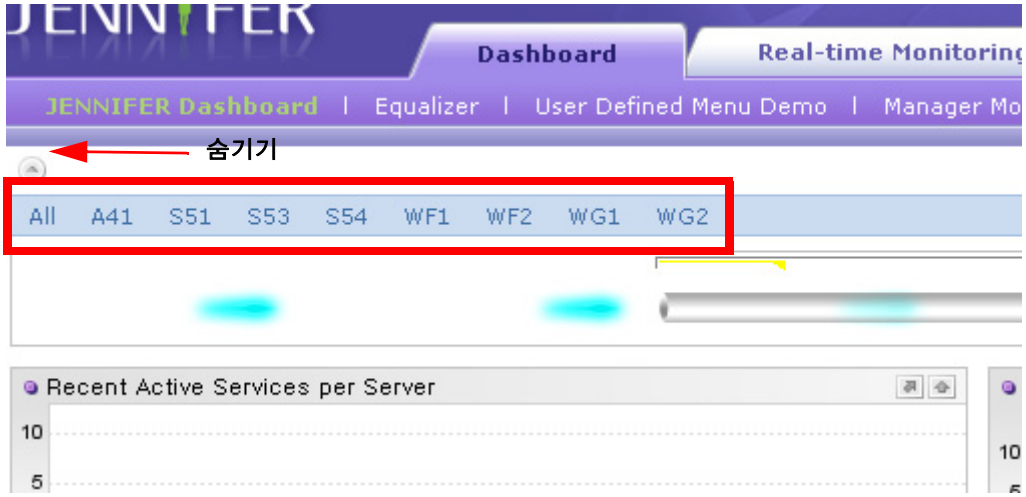
항목	설명
동시단말 사용자 수	실시간 동시단말 사용자 수
액티브 서비스 개수	실시간 액티브 서비스 개수
서비스 처리율	실시간 서비스 처리율
평균 응답 시간	실시간 평균 응답 시간

5.3.4. 메인 영역

다양한 모니터링 정보가 나타나는 영역으로 상단 영역의 메뉴를 클릭하면 메인 영역의 내용이 변경된다.

대부분의 화면 상단에 다음과 같이 제니퍼 에이전트 선택 영역이 존재한다.

그림 5-12: 제니퍼 에이전트 선택 영역



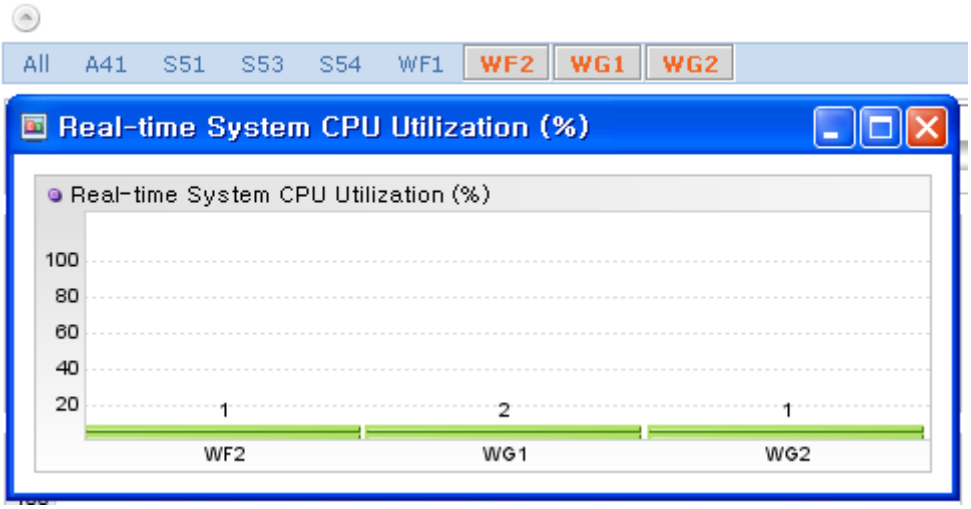
화면의 성격에 따라서 제니퍼 에이전트 선택 영역에 나타나는 항목이 달라진다.

- 모두 - 대시보드에 모든 제니퍼 에이전트를 표시하거나, 데이터를 조회할 때 특정 제니퍼 에이전트가 아닌 모든 제니퍼 에이전트들에 대해서 검색을 수행할 필요가 있는 경우에 제공한다.
- 서버 - **[구성 관리 | 구성 설정]** 메뉴에만 나타난다. 제니퍼 서버의 옵션을 변경하기 위해서 사용한다.
- 제니퍼 에이전트 - 제니퍼 에이전트를 의미한다. 화면에 따라서 여러 개의 제니퍼 에이전트를 동시에 선택할 수 있는 경우도 있고, 배타적으로 하나의 제니퍼 에이전트만을 선택해야 하는 경우도 있다. 제니퍼 에이전트가 정지된 경우에는 제니퍼 에이전트 아이디 옆에 [정지]가 표기되고, 현재는 모니터링하지 않지만 과거에 모니터링했던 제니퍼 에이전트가 있는 경우에는, **[통계 분석]** 메뉴 등에서 해당 제니퍼 에이전트가 제니퍼 에이전트 선택 영역에 나타나고, 제니퍼 에이전트 아이디 옆에 [미설정]으로 표기된다.

Notice: [숨기기] 아이콘을 통해서 제니퍼 에이전트 선택 영역을 토글할 수 있다.

차트를 보여주는 화면의 경우에, 제니퍼 에이전트 선택 영역에서 선택한 제니퍼 에이전트들만으로 차트가 나타난다. 이때 [모두]를 선택하면 선택한 제니퍼 에이전트들과 상관없이 모든 제니퍼 에이전트가 화면에 나타난다. 기본적으로 아무 것도 선택하지 않으면 모든 제니퍼 에이전트가 화면에 나타난다.

그림 5-13: 제니퍼 에이전트 선택



5.3.5. 팝업 창

팝업 창의 구성은 다음과 같다.

그림 5-14: 팝업 창

1. 팝업 타이틀 - 팝업 창의 제목을 보여준다.

2. 인쇄 버튼 - 인쇄 버튼을 클릭하면 팝업 창의 내용을 인쇄한다.
3. 닫기 버튼 - 닫기 버튼을 클릭하면 팝업 창을 닫는다.

5.4. 사용자 인터페이스 주요 기능

5.4.1. 스타일 변경

스타일은 제니퍼 사용자 인터페이스 테마를 의미한다. 사용자가 사용할 수 있는 스타일에는 BLACK, WHITE 등이 있으며 기본은 BLACK이다. 스타일은 다음과 같이 변경한다.

1. 상단 영역의 자바 플러그인 힙 메모리 차트를 클릭하면 컨텍스트 메뉴가 나타난다.
2. 컨텍스트 메뉴에서 스타일 변경을 선택한 후에 사용하고자 하는 스타일을 선택한다.

5.4.2. 다국어 처리

제니퍼 사용자 인터페이스는 영어, 한국어, 일본어, 중국어, 프랑스어 등의 다국어를 지원한다.

Notice: 제니퍼 서버와 웹 브라우저 사이의 인코딩은 UTF-8이며, 이를 변경할 수는 없다.

5.4.2.1. 옵션 설정

다국어 언어 목록은 제니퍼 서버의 supported_language_list 옵션으로 설정한다.

```
supported_language_list = en,ko,ja,zh,fr,es,pt
```

각 언어 코드를 콤마(,)로 구분하는데 특정 언어를 사용하지 않으려면 해당 언어 코드를 옵션에서 삭제한다.

표 5-5: 언어 및 언어 코드

언어	언어 코드
영어	en
한국어	ko
일본어	ja
중국어	zh

표 5-5: 언어 및 언어 코드

언어	언어 코드
프랑스어	fr
스페인어	es
포르투칼어	pt

5.4.2.2. 언어 변경

기본 언어는 웹브라우저의 언어 설정을 따른다. 임의의 언어로 변경하려면 다음과 같이 한다.

1. 상단 영역의 자바 플러그인 힙 메모리 차트를 클릭하면 컨텍스트 메뉴가 나타난다.
2. 컨텍스트 메뉴에서 언어 변경을 선택한 후에 사용하고자 하는 언어를 선택한다.

5.4.2.3. 메시지 변경

특정 메시지 자체를 임의로 변경할 수도 있다. 예를 들어, 첫번째 메뉴 **[대시보드]**를 **[메인 화면]**으로 변경하려면 다음과 같이 한다.

1. 상단 영역의 자바 플러그인 힙 메모리 차트를 클릭하면 컨텍스트 메뉴가 나타난다.
2. 컨텍스트 메뉴에서 언어 변경을 선택한 후에 키를 선택한다.
3. 변경하고자 하는 메시지의 키를 확인한다.예를 들어, **[대시보드]**에 대한 키는 [menu.dashboard]이다.
4. **[구성 관리 | 구성 설정 | 메시지 관리]** 메뉴로 이동한다.
5. 하단에 있는 추가 버튼을 클릭한다.
6. 키, 값, 언어를 입력한 후에 옆에 있는 저장 버튼을 클릭한다. 예를 들어, 키에는 [menu.dashboard]를, 값에는 **[메인 화면]**을 입력하고, 언어로는 [ko]를 선택한다.
7. 하단에 있는 새로 고침 버튼을 클릭한다.
8. 로그아웃 후에 다시 로그인하면 변경 내용을 확인할 수 있다.

5.4.3. 스크린 캡처 게시판에 저장하기

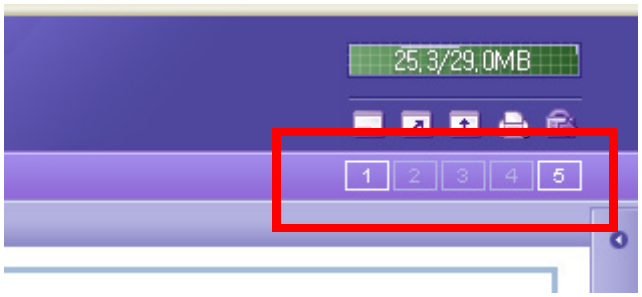
현재 보고 있는 화면을 게시판에 저장하려면 다음과 같이 한다.

1. 상단 영역의 자바 플러그인 힙 메모리 차트를 클릭하면 컨텍스트 메뉴가 나타난다.
2. 컨텍스트 메뉴에서 **[스크립 캡처를 게시판에 저장]** 메뉴를 클릭한다.
3. 저장된 이미지는 **[구성 관리 | 게시판]** 메뉴에서 확인할 수 있다.

5.4.4. 즐겨 찾기

자주 사용하는 메뉴를 5개까지 즐겨 찾기할 수 있다. 상단 영역 오른쪽 하단에 1, 2, 3, 4, 5로 표시된 부분이 즐겨 찾기 영역이다. 숫자 자체에는 의미가 없고 숫자가 흰색으로 표시되어 있으면 즐겨 찾기가 있음을 의미하고, 숫자가 회색으로 표시되어 있으면 즐겨 찾기가 없음을 의미한다. 흰색으로 표시된 숫자에 마우스 커서를 가져가면 즐겨 찾기된 메뉴를 확인할 수 있으며, 해당 번호를 클릭하면 즐겨 찾기된 메뉴로 이동한다.

그림 5-15: 즐겨 찾기



예를 들어, **[실시간 모니터링 | 프로파일]** 메뉴를 즐겨 찾기하려면 다음과 같이 한다.

1. 즐겨 찾기를 할 메뉴로 이동한다. 즉, **[실시간 모니터링 | 프로파일]** 메뉴로 이동한다.
2. 즐겨 찾기 영역에서 회색으로 표시된 임의의 숫자를 더블 클릭한다. 그러면 **[실시간 모니터링 | 프로파일]** 메뉴가 해당 숫자에 즐겨 찾기가 되고 해당 숫자는 흰색으로 표시된다.

즐겨 찾기를 제거하려면 흰색으로 표시된 번호를 더블 클릭한다.

5.4.5. 로고 및 타이틀 변경

[구성 관리 | 구성 설정 | 화면 관리] 메뉴를 통해서 로고 및 타이틀을 변경한다.

표 5-6: 로고 및 타이틀 변경

항목	설명
로고	상단 영역의 로고 이미지
로그인 로고	로그인 화면의 로고 이미지
타이틀	웹 브라우저의 타이틀 메시지
저작권	로그인 화면의 저작권 메시지

로고를 변경한 이후에 기본 로고를 다시 사용하려면 초기화 체크 박스를 선택한 후에 저장 버튼을 클릭한다.

Notice: 로고 이미지로는 png 파일과 gif 파일만을 사용할 수 있다.

5.4.6. 단축키 사용

사용자 편의성을 높이기 위해서 단축키 사용을 지원한다. 예를 들어, Ctrl 키와 왼쪽 방향키를 함께 누르면 화면이 축소된다. 그리고 Ctrl 키와 오른쪽 방향키를 함께 누르면 화면이 확대된다. 이는 상단 영역의 축소 버튼과 확대 버튼에 대한 단축키이다. 그리고 Ctrl 키와 위쪽 방향키를 함께 누르면 현재 화면이 팝업 창에 나타난다. 이는 상단 영역의 팝업창 열기 버튼에 대한 단축키이다. 팝업 창에서 Ctrl 키와 아래쪽 방향키를 함께 누르면 팝업 창이 닫힌다. 이는 팝업 창의 닫기 버튼에 대한 단축키이다.

메인 영역에 있는 버튼에 대한 단축키는 버튼 이름 옆에 나타난다. **(E)**로 표시되어 있는 버튼의 단축키는 Enter 키이다. 나머지의 경우에는 Ctrl 키와 Alt 키와 해당 키를 함께 누르면 된다. 표시가 없으면 단축키가 없음을 의미한다.

그림 5-16: 단축키 사용

Notice: HTML 텍스트 에어리어(TextArea)에 포커스가 있는 경우에는 Enter 키는 단축키가 아닌 줄 바꿈의 역할을 한다. 따라서 Enter 키를 통한 단축키를 사용하려면 HTML 텍스트 에어리어에서 다른 부분으로 포커스를 이동해야 한다.

5.4.7. 자바 플러그인 가비지 콜렉션

자바 플러그인 힙 메모리에 대한 가비지 콜렉션을 하려면 다음과 같이 한다.

1. 상단 영역의 자바 플러그인 힙 메모리 차트를 클릭하면 컨텍스트 메뉴가 나타난다.
2. 컨텍스트 메뉴에서 가비지 콜렉션을 선택한다.

5.4.8. 자바 플러그인 임시 파일 지우기

성능 향상을 위해서 애플리케이션 이름, SQL, 과거 일반 성능 데이터 등을 사용자 컴퓨터에 저장한다. 이 파일들을 삭제하려면 다음과 같이 한다.

1. 상단 영역의 자바 플러그인 힙 메모리 차트를 클릭하면 컨텍스트 메뉴가 나타난다.

2. 컨텍스트 메뉴에서 임시 파일 지우기를 선택한다.

5.5. 메뉴 구성

제니퍼 메뉴 체계는 5개의 상위 메뉴와 각 상위 메뉴에 대한 하위 메뉴로 구성된다.

5.5.1. 대시보드

다양한 성능 데이터를 통합적으로 모니터링하는데 적합한 제니퍼 대시보드와 이퀄라이저 메뉴를 제공한다. 사용자는 사용자 정의 대시보드를 통해서 임의의 대시보드를 구성할 수 있다.

5.5.2. 실시간 모니터링

실시간 X-View 메뉴, 프로파일을 동적으로 제어하는 메뉴, 최근 10분 동안의 애플리케이션 처리 현황 통계를 보여주는 메뉴, 그리고 다양한 모니터링 정보를 차트로 보여주는 실시간 현황 메뉴 등을 제공한다.

5.5.3. 통계 분석

과거 X-View 정보를 조회할 수 있는 메뉴, 과거 애플리케이션 처리 현황 통계를 보여주는 메뉴, 그리고 다양한 모니터링 정보를 차트로 보여주는 통계 현황 메뉴를 제공한다.

또한 일일, 주간, 월간 보고서 등을 제공하는 보고서 메뉴와 CRUD 메트릭스와 쿼리 수행기 메뉴도 제공한다.

5.5.4. 장애 진단

장애를 진단하고 해결하는데 필요한 메뉴를 제공한다. 예를 들어, 콜렉션 객체를 분석할 수 있는 메뉴, 파일과 소켓 사용 현황을 제공하는 메뉴, 그리고 HTTP 세션 객체의 정보를 제공하는 메뉴를 제공한다.

또한, 시스템 환경 변수와 로딩 클래스 목록을 확인할 수 있는 메뉴, 클래스 혹은 JAR 파일을 검색할 수 있는 메뉴, 그리고 수정된 파일이나 미사용 클래스를 검색할 수 있는 메뉴 등도 제공한다.

5.5.5. 구성 관리

제니퍼 서버와 제니퍼 에이전트 옵션을 설정할 수 있는 메뉴, 도메인과 노드를 구성하는 메뉴, 그리고 라이센스키를 관리할 수 있는 메뉴를 제공한다.

또한, 사용자, 권한, 메뉴를 관리할 수 있는 메뉴, 비즈니스 그룹을 설정하는 메뉴, 그리고 게시판 등의 메뉴 등도 제공한다.

6

서비스 및 사용자 모니터링

6.1. 자바 애플리케이션 서비스와 트랜잭션

자바 애플리케이션은 여러 개의 애플리케이션 서비스로 구성되어 있다. 애플리케이션 서비스는 고정된 개념이 아니고, 호출 건수와 응답 시간 등의 성능 데이터를 수집하는데 기준이되는 통계 단위이다.

Notice: 애플리케이션 서비스를 애플리케이션이라고도 한다. 그러나 이는 전체 프로그램을 의미하는 애플리케이션과는 다른 의미에서 사용된다. 그 차이는 전체 맥락 속에서 파악해야 한다.

일반적으로 애플리케이션 서비스는 업무를 기준으로 한다. 예를 들어, 쇼핑몰 애플리케이션에서는 제품 목록 보기, 제품 상세 내역 보기, 장바구니에 담기, 주문, 결제 등이 애플리케이션 서비스가 될 수 있다. 그런데 하나의 업무가 세분화될 수 있고, 성능 관리 측면에서는 세부 단계 별로 성능 데이터를 수집하는 것이 바람직하기 때문에 애플리케이션 서비스가 업무 구분보다 더 세분화되는 경향이 있다.

일차적으로 웹 애플리케이션에서는 URI가 애플리케이션 서비스를 지칭하는 대표적인 방법이다.

```
/view.jsp
/order.jsp
/pay.jsp
```

그러나 모든 요청의 URI가 동일해서 요청 파라미터로 애플리케이션 서비스를 구분해야 하는 경우도 있을 수 있다.

```
/service.do?action=view
/service.do?action=order
/service.do?action=pay
```

이 경우에 HTTP GET 방식이 아닌 HTTP POST 방식으로 파라미터가 전달될 수 있기 때문에 URI가 아닌 다른 방법으로 애플리케이션 서비스를 구분해야 한다.

또한 웹 애플리케이션이 아닌 경우에는 특정 클래스의 특정 메소드로 애플리케이션 서비스를 구분해야 한다.

```
example.ViewManager 클래스의 view 메소드
example.OrderManager 클래스의 order 메소드
example.PayManager 클래스의 pay 메소드
```

트랜잭션은 사용자가 애플리케이션 서비스를 요청하여 해당 애플리케이션 서비스가 자바 애플리케이션에서 수행된 것을 의미한다. 호출 건수, 평균 응답 시간 등은 트랜잭션 처리 결과를 바탕으로 수집하는 성능 데이터이고, 애플리케이션 서비스는 메타 데이터의 역할을 한다.

6.2. 애플리케이션 서비스 시작점

애플리케이션 서비스 시작점은 트랜잭션이 시작하는 위치이다. 엄밀하게 트랜잭션 시작점은 클라이언트 요청을 받아들이는 맨 앞단의 네트워크 모듈이 된다. 하지만 대부분의 요청을 동일한 포트 번호로 받아들이기 때문에, 이 지점에서는 애플리케이션 서비스를 구분할 수 없다. 트랜잭션과 관련한 성능 데이터는 애플리케이션 서비스를 기준으로 수집되기 때문에 애플리케이션 서비스 시작점은 애플리케이션 서비스를 구분할 수 있는 최초의 위치가 되어야 한다.

모니터링하는 자바 애플리케이션이 WAS이냐 아니냐에 따라서 애플리케이션 서비스 시작점을 설정하는 방법이 달라진다.

6.2.1. 웹 애플리케이션

WAS에 기반한 웹 애플리케이션의 애플리케이션 서비스 시작점은 javax.servlet.http.HttpServlet 클래스의 service(ServletRequest, ServletResponse) 메소드가 된다. 이 경우에는 별도의 설정이 필요없다.

그런데 이 메소드를 재정의한(override) 서블릿을 모니터링하려면 제니퍼 에이전트의 http_service_class 옵션으로 해당 서블릿을 설정해야 한다. 여러 개는 [;]을 구분자로 구분해서 설정한다. 예를 들어, example.AServlet과 example.BServlet 클래스가 javax.servlet.http.HttpServlet 클래스를 상속하고 service 메소드를 재정의했다면 다음과 같이 설정한다.

```
http_service_class = example.AServlet;example.BServlet
```

수정한 http_service_class 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다. http_service_class 옵션에 등록하지 않아도 WAS들이 제공하는 다음 서블릿은 자동으로 애플리케이션 서비스 시작점이 된다. 이 클래스들은 대부분 WAS 별 JSP의 상위 서블릿 클래스이다. 따라서 대부분의 WAS에서 JSP에 대한 별도의 설정을 할 필요가 없다.

```
javax.servlet.http.HttpServlet
com.evermind.server.http.JSPServlet
com.caucho.jsp.JavaPage
jrun.jsp.runtime.HttpJSPServlet
allaire.jrun.jsp.HttpJSPServlet
weblogic.servlet.jsp.JspBase
```

service가 아닌 ServletRequest과 ServletResponse 클래스를 파라미터로 하는 메소드를 애플리케이션 서비스 시작점으로 설정하려면 제니퍼 에이전트의 http_service_method 옵션을 사용한다. 수정한 http_service_method 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

```
http_service_method = doFilter
```

http_service_method 옵션에는 파라미터 개수와 상관없이 처음 2개의 파라미터가 ServletRequest와 ServletResponse인 메소드를 지정할 수 있다. 그러나 2개 이상의 메소드를 지정할 수는 없다. 그렇지만 service 메소드는 자동으로 등록되기 때문에 서블릿을 모니터링하는데는 문제가 없다.

javax.servlet.Filter 인터페이스를 구현한 필터 클래스를 애플리케이션 서비스 시작점으로 설정하려면 다음과 같이 한다. 다음의 예는 javax.servlet.Filter 인터페이스를 구현한 example.FrontFilter 클래스를 애플리케이션 서비스의 시작점이라고 가정한다.

```
http_service_class = example.FrontFilter
http_service_method = doFilter
```

그리고 성능 데이터를 수집할 필요가 없는 URI는 모니터링하지 않을 수 있다.

우선 제니퍼 에이전트의 ignore_url 옵션을 통해서 모니터링하지 않을 URI를 설정한다.

```
ignore_url = /check.jsp,/test.jsp
```

그리고 제니퍼 에이전트의 ignore_url_postfix 옵션을 통해서 특정 형태로 끝나는 URI를 모니터링하지 않을 수 있다.

```
ignore_url_postfix = .gif .GIF .jpg .JPG .zip .html .HTML .txt .css .js
.swf .htc .HTC .png .PNG
```

또한 제니퍼 에이전트의 ignore_url_prefix 옵션을 통해서 특정 형태로 시작하는 URI를 모니터링하지 않을 수 있다.

```
ignore_url_prefix = /admin,/console
```

6.2.2. 일반 자바 애플리케이션

WAS를 사용하지 않는 일반 자바 애플리케이션의 경우에는 제니퍼 에이전트의 tx_server로 시작하는 옵션을 통해서 애플리케이션 서비스 시작점을 설정한다. 수정한 tx_server로 시작하는 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

예를 들어, 주문을 처리하는 클래스가 example.OrderManager라고 가정하자. 이를 애플리케이션 서비스의 시작점으로 설정하려면 제니퍼 에이전트의 tx_server_class 옵션으로 다음과 같이 설정한다.

```
tx_server_class = example.OrderManager
```

그리고 결제를 처리하는 클래스가 example.PayManager라고 가정하자. 이 클래스도 애플리케이션 서비스 시작점으로 설정하려면 [;]을 구분자로 tx_server_class 옵션에 설정한

다. tx_server_class 옵션뿐만 아니라 애플리케이션 서비스 시작점과 관련한 모든 옵션에 2개 이상을 설정하려면 [;]을 구분자로 사용한다.

```
tx_server_class = example.OrderManager;example.PayManager
```

그런데 애플리케이션 서비스 시작점은 특정 클래스의 특정 메소드를 의미한다. 따라서 앞에서와 같이 설정하면 example.OrderManager 클래스의 모든 메소드가 애플리케이션 서비스 시작점으로 인식된다.

따라서 특정 메소드만을 애플리케이션 서비스 시작점으로 설정하려면 제니퍼 에이전트의 tx_server_target_method 옵션으로 설정한다.

```
tx_server_target_method = order
```

그런데 애플리케이션 서비스 시작점으로 설정한 클래스들 중에서 이름이 동일한 메소드가 있고, 이 중 특정 클래스의 특정 메소드만을 애플리케이션 서비스 시작점으로 설정하려면 다음과 같이 구체적으로 설정한다.

```
tx_server_target_method = example.OrderManager.order
```

반대로 특정 메소드만을 애플리케이션 서비스 시작점에서 제외하려면 제니퍼 에이전트의 tx_server_ignore_method 옵션으로 설정한다.

```
tx_server_ignore_method = example.OrderManager.someMethod
```

메소드의 접근자를 통해서도 애플리케이션 서비스 시작점을 설정할 수 있다. 예를 들어 public 접근자와 접근자가 없는 메소드를 애플리케이션 서비스 시작점으로 등록하려면 다음과 같이 설정한다.

```
tx_server_access_method = public;none
```

tx_server_access_method 옵션에 설정이 가능한 값은 다음과 같다.

- public - public 접근자
- protected - protected 접근자
- private - private 접근자
- none - 접근자가 없는 경우

그리고 특정 클래스를 상속한 모든 클래스가 애플리케이션 서비스 시작점인 경우에는 제니퍼 에이전트의 tx_server_super 옵션을 사용한다. 예를 들어, example.BaseAction을 상속한 모든 클래스를 애플리케이션 서비스 시작점으로 설정하려면 다음과 같이 설정한다.

```
tx_server_super = example.BaseAction
```

그러나 이 경우에 직접적으로 상속받은 클래스만이 애플리케이션 서비스 시작점이 된다. 예를 들어, A 클래스가 example.BaseAction 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스는 애플리케이션 서비스 시작점으로 인식하지만 B 클래스는 애플리케이션 서비스 시작점으로 인식하지 않는다.

그리고 특정 인터페이스를 구현한 모든 클래스가 애플리케이션 서비스 시적점인 경우에는 제니퍼 에이전트의 tx_server_interface 옵션을 사용한다. 예를 들어, example.IAction 인터페이스를 구현한 모든 클래스를 애플리케이션 서비스 시작점으로 설정하려면 다음과 같이 설정한다.

```
tx_server_interface = example.IAction
```

그러나 이 경우에 직접적으로 구현한 클래스만이 애플리케이션 서비스 시작점이 된다. 예를 들어, A 클래스가 example.IAction 인터페이스를 구현하고 B 클래스가 A 클래스를 상속했다면, A 클래스는 애플리케이션 서비스 시작점으로 인식하지만 B 클래스는 애플리케이션 서비스 시작점으로 인식하지 않는다.

그리고 클래스의 이름을 이용해서도 애플리케이션 서비스 시작점을 설정할 수 있다. 이 경우에는 제니퍼 에이전트의 tx_server_prefix, tx_server_postfix, tx_server_ignore_prefix 옵션을 사용한다. 예를 들어, 이름이 example.action으로 시작하는 모든 클래스를 애플리케이션 서비스 시작점으로 설정하려면 다음과 같이 한다.

```
tx_server_prefix = example.action
```

또한 이름이 Action으로 끝나는 모든 클래스를 애플리케이션 서비스 시작점으로 설정하려면 다음과 같이 한다.

```
tx_server_postfix = Action
```

그리고 특정 이름으로 시작하는 클래스를 애플리케이션 서비스 시작점에서 제외하려면 제니퍼 에이전트의 tx_server_ignore_prefix 옵션을 사용한다. 이 옵션은 다른 모든 옵션에 우선한다.

```
tx_server_ignore_prefix =
```

제니퍼 에이전트의 tx_server로 시작하는 옵션을 통해서 동일한 내용을 다양한 방법으로 설정할 수 있다. 애플리케이션 서비스 시작점을 설정하는데 있어서

tx_server_target_method 옵션을 통해서 실제로 애플리케이션 서비스 시작점인 메소드만을 설정하도록 한다.

예를 들어, pkg.ClassA와 pkg.ClassB 클래스가 있다. 여기서 ClassA 클래스의 run 메소드와 ClassB 클래스의 process 메소드가 애플리케이션 서비스 시작점이라고 가정한다. 그런데 ClassB 클래스에 run 메소드가 존재하고 이 메소드는 애플리케이션 서비스 시작점과는 상관이 없다면 다음과 같이 설정한다.

```
tx_server_class = pkg.ClassA;pkg.ClassB
tx_server_target_method = run;process
tx_server_ignore_method = pkg.ClassB.run
```

또는 다음과 같이 설정할 수도 있다.

```
tx_server_class = pkg.ClassA;pkg.ClassB
tx_server_target_method = pkg.ClassA.run;pkg.ClassB.process
```

6.3. 애플리케이션 서비스 네이밍

애플리케이션 서비스 이름은 APPLS 테이블에 저장된다. 기본적으로 웹 애플리케이션은 URI가 애플리케이션 서비스 이름이 되고, tx_server 옵션으로 설정한 경우에는 클래스 이름과 메소드 이름이 애플리케이션 서비스 이름이 된다. 그런데 요청 파라미터 값이나 특정 메소드의 파라미터 혹은 반환 값 등을 이용해서 애플리케이션 서비스 이름을 설정할 수도 있다.

이렇게 애플리케이션 서비스 이름을 다양한 방식으로 설정하는 것을 애플리케이션 서비스 네이밍이라고 한다.

6.3.1. 웹 애플리케이션

웹 애플리케이션은 요청 URI가 애플리케이션 서비스 이름이 된다. 그런데 웹 애플리케이션 프레임워크의 발전으로 애플리케이션 서비스가 URI만으로는 구분되지 않는 경우가 많아지고 있다. 이런 경우에는 요청 파라미터를 이용해서 애플리케이션 서비스 이름을 구체적으로 설정할 수 있다. 이를 위해서 제니퍼 에이전트의 url_additional_request_keys 옵션에 요청 파라미터 이름을 설정한다.

```
url_additional_request_keys = key1,key2
```

앞의 경우에는 다음 코드가 반환하는 파라미터 값을 사용해서 애플리케이션 서비스 이름을 결정한다.

```
request.getParameter(" key1" );
request.getParameter(" key2" );
```

예를 들어, 특정 요청에 대한 애플리케이션 서비스 이름은 다음과 같이 결정된다.

```
http://127.0.0.1:8080/app.jsp?key1=a&key2=b
   → /app.jsp+(key1=a&key2=b)
http://127.0.0.1:8080/app.jsp?key1=a
   → /app.jsp+(key1=a)
http://127.0.0.1:8080/app.jsp?key1=a&key2=
   → /app.jsp+(key1=a)
http://127.0.0.1:8080/app.jsp
   → /app.jsp
```

파라미터 값의 시작 일부만을 사용할 수도 있다.

```
url_additional_request_keys = key1:5
```

앞에서와 같이 설정하면 다음 코드가 반환하는 값을 애플리케이션 서비스 이름에 추가한다.

```
request.getParameter("key1").substring(0, 5)
```

일반적으로 URI와 파라미터 사이의 구분 문자는 [?]이다. 하지만 이동식 단말기를 위한 WAS 등에서는 다른 구분 문자를 사용하기도 한다. 제니퍼 에이전트의 uri_separator 옵션을 통해서 이를 명시적으로 설정할 수 있다.

```
uri_separator = ?
```

파일 업로드 요청(multipart/form-data)에는 url_additional_request_keys 옵션이 적용되지 않는다. 그리고 제니퍼 에이전트의 ignore_url_post_request_parsing_prefixes 옵션으로 특정 이름으로 시작하는 URI에 대해서 url_additional_request_keys 옵션이 적용되지 않도록 할 수 있다.

```
ignore_url_post_request_parsing_prefixes = /kesa.web,/insa/x_servlet
```

또한 파라미터의 값이 다국어이고, 서블릿 3.2 이상에서 javax.servlet.http.HttpServletRequest 객체의 setCharacterEncoding 메소드를 호출해서 다국어 처리를 하는 경우에는 애플리케이션 서비스 이름에 다국어가 올바르게 나타나지 않을 수 있다. 제니퍼 에이전트가

소스 코드에서 setCharacterEncoding 메소드가 호출되기 전에 파라미터 값을 획득하기 때문이다.

```
doGet(...) {
    request.setCharacterEncoding(" KSC5601" );
    ...
}
```

따라서 이 경우에는 제니퍼 에이전트의 request_set_character_encoding 옵션으로 인코딩을 설정하도록 한다.

```
request_set_character_encoding = KSC5601
```

그리고 아쿠아 로직 등의 솔루션은 URI에 사용자 정보 등이 포함된다. 다음은 아쿠아 로직의 URI 예제이다.

```
/x/gate/user_123_01_21/http;//ap1/xx
```

이 URI를 그대로 애플리케이션 서비스 이름으로 사용하면 애플리케이션 서비스 이름이 사용자 수에 비례해서 크게 증가한다. 예를 들어, 애플리케이션이 1,000개이고 사용자가 1,000명인 경우에 애플리케이션 서비스 이름은 100만개가 된다. 이 결과로 제니퍼 서버에서 java.lang.OutOfMemoryError 예외가 발생할 수 있다.

이를 해결하기 위해서 제니퍼 에이전트의 uri_starter 옵션으로 URI의 특정 패턴을 시작점으로 애플리케이션 서비스 이름을 도출할 수 있다.

```
uri_starter = /http
```

앞의 예에서, 제니퍼 에이전트의 uri_starter 옵션을 /http로 설정하면 애플리케이션 서비스 이름은 다음과 같이 된다.

```
/http;//ap1/xx
```

6.3.2. 일반 자바 애플리케이션

기본적으로 애플리케이션 서비스 이름은 클래스 이름과 메소드 이름으로 구성된다. 제니퍼 에이전트의 tx_server_ntype 옵션을 통해서 이를 다양하게 설정할 수 있다. 기본 값은 FULL이고, SIMPLE, CLASS, METHOD 등으로 설정할 수 있다. 수정한 tx_server_ntype 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

```
tx_server_ntype = FULL
```

예를 들어, pkg.ClassB 클래스의 process 메소드를 애플리케이션 서비스 시작점이라고 할 때 tx_server_ntype 옵션 설정에 따라서 애플리케이션 서비스 이름은 다음과 같이 결정된다.

- FULL - public void pkg.ClassB.process()
- SIMPLE - ClassB.process
- CLASS - ClassB
- METHOD - process

그리고 애플리케이션 서비스 시작점인 메소드의 파라미터 값을 애플리케이션 서비스 이름으로 사용할 수도 있다. 이를 위해서는 제니퍼 에이전트의 tx_server_using_param 옵션을 true로 설정한다. 수정한 tx_server_using_param 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

```
tx_server_using_param = true
```

tx_server_using_param 옵션을 true로 설정하면, 애플리케이션 서비스 시작점인 메소드의 여러 개의 파라미터 중에서 java.lang.String 타입의 파라미터 중에서 첫번째 파라미터가 애플리케이션 서비스 이름이 된다. 해당 사항이 없는 경우에는 tx_server_ntype 옵션에 따라서 애플리케이션 서비스 이름이 결정된다.

그리고 애플리케이션 서비스 이름으로 애플리케이션 서비스 시작 메소드 내부에서 호출되는 특정 클래스의 특정 메소드의 파라미터나 반환 값을 사용할 수도 있다.

애플리케이션 서비스 시작 메소드 내부에서의 특정 클래스의 특정 메소드의 파라미터를 애플리케이션 서비스 이름으로 사용하려면 다음과 같이 설정한다. 수정한 lwst_txserver_method_using_param 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

```
lwst_txserver_method_using_param = pkg.ClassC.f1(String)
```

이 경우에는 여러 개의 파라미터 중에서 java.lang.String 타입의 파라미터 중에서 첫번째 파라미터가 애플리케이션 서비스 이름이 된다.

그리고 애플리케이션 서비스 시작 메소드 내부에서의 특정 클래스의 특정 메소드의 반환 값을 애플리케이션 서비스 이름으로 사용하려면 다음과 같이 설정한다. 수정한 lwst_txserver_method_using_return 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

```
lwst_txserver_method_using_return = pkg.ClassC.f2(String)
```

단, 이 경우에는 반환되는 객체의 유형이 java.lang.String이어야 한다.

lwst_txserver_method_using_param과 lwst_txserver_method_using_return 옵션에도 2개 이상은 [;]을 구분자로 해서 설정한다.

여러 옵션이 설정된 경우에는 다음 옵션 순서대로 애플리케이션 서비스 이름이 결정된다.

- lwst_txserver_method_using_param 혹은 lwst_txserver_method_using_return
- tx_server_using_param
- tx_server_ntype

lwst_txserver_method_using_param과 lwst_txserver_method_using_return 옵션의 경우에는 실제 트랜잭션이 처리되는 과정에서 마지막으로 수행되는 메소드의 파리미터 혹은 반환 값이 애플리케이션 서비스 이름이 된다.

6.3.3. 로직 기반 네이밍

애플리케이션 서비스 시작점에서 그 이름을 결정할 수 없는 경우가 있다. 예를 들어, SOAP 프로토콜을 사용하는 경우에 애플리케이션 서비스 시작점에서 XML을 파싱하지 않고는 그 이름을 판별하지 못할 수 있다. 이 경우에는 트랜잭션이 수행되는 로직 흐름에서 tx_naming으로 시작하는 옵션으로 설정한 클래스와 메소드 이름을 애플리케이션 서비스 이름에 추가할 수 있다. 수정한 tx_naming으로 시작하는 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

예를 들어, 애플리케이션 서비스 이름을 결정하는데 중요한 특정 클래스가 example.BizAction이라고 가정하자. 이를 애플리케이션 서비스 이름에 추가하려면 제니퍼 에이전트의 tx_naming_class 옵션으로 설정한다.

```
tx_naming_class = example.BizAction
```

그리고 example.LogAction도 애플리케이션 서비스 이름을 결정하는데 중요한 클래스라고 가정하자. 이 클래스도 애플리케이션 서비스 이름에 추가하려면 [;]을 구분자로 tx_naming_class 옵션에 설정한다. tx_naming_class 옵션뿐만 아니라 애플리케이션 서비스 이름과 관련한 모든 옵션에 2개 이상을 설정하려면 [;]을 구분자로 사용한다.

```
tx_naming_class = example.BizAction;example.LogAction
```

그런데 애플리케이션 서비스 이름에 추가되는 것은 특정 클래스의 특정 메소드 이름이다. 물론 이는 제니퍼 에이전트의 tx_naming_ntype 옵션 설정에 영향을 받는다. 따라서 앞에서와 같이 설정하면 트랜잭션 처리 과정에서 호출된 example.BizAction 클래스의 모든 메소드가 애플리케이션 서비스 이름에 추가된다.

따라서 특정 메소드만을 애플리케이션 서비스 이름에 추가하려면 제니퍼 에이전트의 tx_naming_target_method 옵션으로 설정한다.

```
tx_naming_target_method = process
```

그런데 애플리케이션 서비스 이름에 추가할 클래스들 중에서 이름이 동일한 메소드가 있고, 이중 특정 클래스의 특정 메소드만을 애플리케이션 서비스 이름에 추가하려면 다음과 같이 구체적으로 설정한다.

```
tx_naming_target_method = example.BizAction.process
```

반대로 특정 메소드만을 애플리케이션 서비스 이름에 추가하지 않으려면 제니퍼 에이전트의 tx_naming_ignore_method 옵션으로 설정한다.

```
tx_naming_ignore_method = example.BizAction.someMethod
```

메소드의 접근자를 통해서도 애플리케이션 서비스 이름에 추가할 메소드를 설정할 수 있다. 예를 들어 public 접근자와 접근자가 없는 메소드를 애플리케이션 서비스 이름에 추가하려면 다음과 같이 설정한다.

```
tx_naming_access_method = public;none
```

tx_naming_access_method 옵션에 설정이 가능한 값은 다음과 같다.

- public - public 접근자
- protected - protected 접근자
- private - private 접근자
- none - 접근자가 없는 경우

그리고 특정 클래스를 상속한 모든 클래스를 애플리케이션 서비스 이름에 추가하려면 제니퍼 에이전트의 tx_naming_super 옵션을 사용한다. 예를 들어, example.BaseAction을 상속한 모든 클래스를 애플리케이션 서비스 이름에 추가하려면 다음과 같이 설정한다.

```
tx_naming_super = example.BaseAction
```

그러나 이 경우에 직접적으로 상속받은 클래스만이 애플리케이션 서비스 이름에 추가된다. 예를 들어, A 클래스가 example.BaseAction 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스는 애플리케이션 서비스 이름에 추가되지만 B 클래스는 애플리케이션 서비스 이름에 추가되지 않는다.

그리고 특정 인터페이스를 구현한 모든 클래스를 애플리케이션 서비스 이름에 추가하려면 제니퍼 에이전트의 tx_naming_interface 옵션을 사용한다. 예를 들어, example.IAction

인터페이스를 구현한 모든 클래스를 애플리케이션 서비스 이름에 추가하려면 다음과 같이 설정한다.

```
tx_naming_interface = example.IAction
```

그러나 이 경우에 직접적으로 구현한 클래스만이 애플리케이션 서비스 이름에 추가된다. 예를 들어, A 클래스가 example.IAction 인터페이스를 구현하고 B 클래스가 A 클래스를 상속했다면, A 클래스는 애플리케이션 서비스 이름에 추가되지만 B 클래스는 애플리케이션 서비스 이름에 추가되지 않는다.

그리고 클래스의 이름을 이용해서도 애플리케이션 서비스 이름에 추가될 클래스를 설정할 수 있다. 이 경우에는 제니퍼 에이전트의 tx_naming_prefix, tx_naming_postfix, tx_naming_ignore_prefix 옵션을 사용한다. 예를 들어, 이름이 example.action으로 시작하는 모든 클래스를 애플리케이션 서비스 이름에 추가하려면 다음과 같이 한다.

```
tx_naming_prefix = example.action
```

또한 이름이 Action으로 끝나는 모든 클래스를 애플리케이션 서비스 이름에 추가하려면 다음과 같이 한다.

```
tx_naming_postfix = Action
```

그리고 특정 이름으로 시작하는 클래스를 애플리케이션 서비스 이름에 추가하지 않으려면 제니퍼 에이전트의 tx_naming_ignore_prefix 옵션을 사용한다. 이 옵션은 다른 모든 옵션에 우선한다.

```
tx_naming_ignore_prefix =
```

기본적으로 tx_naming으로 시작하는 옵션 설정에 의해서 애플리케이션 서비스 이름에 추가되는 것은 클래스 이름과 메소드 이름이다. 제니퍼 에이전트의 tx_naming_ntype 옵션을 통해서 이를 다양하게 설정할 수 있다. 기본 값은 CLASS이고, FULL, SIMPLE, METHOD 등으로 설정할 수 있다.

```
tx_naming_ntype = CLASS
```

예를 들어, pkg.ClassB 클래스의 process 메소드를 애플리케이션 서비스 이름에 추가할 때 tx_naming_ntype 옵션 설정에 따라서 애플리케이션 서비스 이름에 다음 내용이 추가된다.

- FULL - public void pkg.ClassB.process()
- SIMPLE - ClassB.process
- CLASS - ClassB
- METHOD - process

그리고 클래스와 메소드 이름이 아닌 파라미터 값을 애플리케이션 서비스 이름에 추가할 수도 있다. 이를 위해서는 제니퍼 에이전트의 tx_naming_using_param 옵션을 true로 설정한다.

```
tx_naming_using_param = true
```

tx_naming_using_param 옵션을 true로 설정하면, 메소드의 여러 개의 파라미터 중에서 java.lang.String 타입의 파라미터 중에서 첫번째 파라미터가 애플리케이션 서비스 이름에 추가된다. 해당 사항이 없는 경우에는 tx_naming_ntype 옵션을 따른다.

또한 클래스와 메소드 이름이 아닌 반환 값을 애플리케이션 서비스 이름에 추가할 수도 있다. 이를 위해서는 제니퍼 에이전트의 tx_naming_using_return 옵션을 true로 설정한다.

```
tx_naming_using_return = true
```

tx_naming_using_return 옵션을 true로 설정하면, 반환 값이 java.lang.String 타입이면 해당 값을 애플리케이션 서비스 이름에 추가하고, 그렇지 않은 경우에는 tx_naming_ntype 옵션을 따른다.

예를 들어, txn.jsp에서 다음 코드를 수행하고 있다고 가정하자.

```
<%
    example.BizAction bizAction = new example.BizAction();
    bizAction.process(request, response);

    example.LogAction logAction = new example.LogAction();
    logAction.process(request, response);
%>
```

그리고 tx_naming을 다음과 같이 설정한다.

```
tx_naming_postfix = Action
tx_naming_ntype = CLASS
```

그러면 애플리케이션 서비스 이름은 다음과 같다.

```
/txn.jsp+BizAction+LogAction
```

6.4. 닷넷 애플리케이션 서비스와 트랜잭션

애플리케이션은 여러 개의 애플리케이션 서비스로 구성되어 있다. 애플리케이션 서비스는 고정된 개념이 아니고, 호출 건수와 응답 시간 등의 성능 데이터를 수집하는데 기준이 되는 통계 단위이다.

Notice: 애플리케이션 서비스를 애플리케이션이라고도 한다. 그러나 이는 전체 프로그램을 의미하는 애플리케이션과는 다른 의미에서 사용된다. 그 차이는 전체 맥락 속에서 파악해야 한다

일반적으로 애플리케이션 서비스는 업무를 기준으로 한다. 예를 들어, 쇼핑몰 애플리케이션에서는 제품 목록 보기, 제품 상세 내역 보기, 장바구니에 담기, 주문, 결제 등이 애플리케이션 서비스가 될 수 있다. 그런데 하나의 업무가 세분화될 수 있고, 성능 관리 측면에서는 세부 단계 별로 성능 데이터를 수집하는 것이 바람직하기 때문에 애플리케이션 서비스가 업무 구분보다 더 세분화되는 경향이 있다.

일차적으로 웹 애플리케이션에서는 URI가 애플리케이션 서비스를 지칭하는 대표적인 방법이다.

```
/view.aspx
/order.aspx
/pay.asmx
/submitOrder.svc
```

그러나 모든 요청의 URI가 동일해서 요청 파라미터로 애플리케이션 서비스를 구분해야 하는 경우도 있을 수 있다.

```
/service.aspx?action=view
/service.aspx?action=order
/service.aspx?action=pay
```

이 경우에 HTTP GET 방식이 아닌 HTTP POST 방식으로도 파라미터가 전달될 수 있기 때문에 URI 이외에 POST 데이터 까지 다뤄야만 애플리케이션 서비스를 구분하는 것이 가능하다.

또한 웹 애플리케이션이 아닌 경우에는 특정 클래스의 특정 메서드로 애플리케이션 서비스를 구분해야 한다.

```
example.ViewManager 클래스의 view 메서드
example.OrderManager 클래스의 order 메서드
example.PayManager 클래스의 pay 메서드
```

트랜잭션은 사용자가 애플리케이션 서비스를 요청하여 해당 애플리케이션 서비스가 애플리케이션에서 수행된 것을 의미한다. 호출 건수, 평균 응답 시간 등은 트랜잭션 처리 결과를 바탕으로 수집하는 성능 데이터이고, 애플리케이션 서비스는 메타 데이터의 역할을 한다.

6.4.1. . 애플리케이션 서비스 네이밍

애플리케이션 서비스 이름은 APPLS 테이블에 저장된다. 기본적으로 웹 애플리케이션은 URI가 애플리케이션 서비스 이름이 된다. 그런데 요청 파라미터 값이나 특정 메서드의 파라미터 혹은 반환 값 등을 이용해서 애플리케이션 서비스 이름을 설정할 수도 있다.

이렇게 애플리케이션 서비스 이름을 다양한 방식으로 설정하는 것을 애플리케이션 서비스 네이밍이라고 한다.

6.4.1.1. 웹 애플리케이션

웹 애플리케이션은 요청 URI가 애플리케이션 서비스 이름이 된다. 그런데 웹 애플리케이션 프레임워크의 발전으로 애플리케이션 서비스가 URI만으로는 구분되지 않는 경우가 많아지고 있다. 이런 경우에는 요청 파라미터를 이용해서 애플리케이션 서비스 이름을 구체적으로 설정할 수 있다. 이를 위해서 제니퍼 에이전트의 url_additional_request_keys 옵션에 요청 파라미터 이름을 설정한다.

```
url_additional_request_keys = key1,key2
```

앞의 경우에는 다음 코드가 반환하는 파라미터 값을 사용해서 애플리케이션 서비스 이름을 결정한다.

```
HttpContext.Current.Request.Params.Get( “key1” );
HttpContext.Current.Request.Params.Get( “key2” );
```

예를 들어, 특정 요청에 대한 애플리케이션 서비스 이름은 다음과 같이 결정된다.

```
http://127.0.0.1:8080/app.aspx?key1=a&key2=b
→ /app.aspx+(key1=a&key2=b)
http://127.0.0.1:8080/app.aspx?key1=a
→ /app.aspx +(key1=a)
http://127.0.0.1:8080/app.aspx?key1=a&key2=
→ /app.aspx +(key1=a)
http://127.0.0.1:8080/app.aspx
→ /app.aspx
```

파라미터 값의 시작 일부만을 사용할 수도 있다.

```
url_additional_request_keys = key1:5
```

앞에서와 같이 설정하면 다음 코드가 반환하는 값을 애플리케이션 서비스 이름에 추가한다.

```
HttpContext.Current.Request.Params.Get("key1").Substring(0, 5)
```

일반적으로 URI와 파라미터 사이의 구분 문자는 [?]이다. 하지만 이동식 단말기를 위한 WAS 등에서는 다른 구분 문자를 사용하기도 한다. 제니퍼 에이전트의 uri_separator 옵션을 통해서 이를 명시적으로 설정할 수 있다.

```
uri_separator = ?
```

파일 업로드 요청(multipart/form-data)에는 url_additional_request_keys 옵션이 적용되지 않는다. 그리고 제니퍼 에이전트의 ignore_url_post_request_parsing_prefixes 옵션으로 특정 이름으로 시작하는 URI에 대해서 url_additional_request_keys 옵션이 적용되지 않도록 할 수 있다.

```
ignore_url_post_request_parsing_prefixes = /kesa.web,/insa/controller
```

그리고, ASP.NET의 경우 Cookie를 사용하지 않는 세션 키를 지원하는 데 이를 사용하는 경우에는 요청 URI 형식에 세션 키 값이 포함된다. 다음은 세션 키가 포함된 URL 예제이다.

```
http://server/(session_key)/SessionState.aspx
예) http://localhost/(lit3py55t21z5v55vlm25s55)/Application/
SessionState.aspx
```

이 URI를 그대로 애플리케이션 서비스 이름으로 사용하면 애플리케이션 서비스 이름이 사용자 수에 비례해서 크게 증가한다. 예를 들어, 애플리케이션이 1,000개이고 사용자가 1,000명인 경우에 애플리케이션 서비스 이름은 100만개가 된다. 이 결과로 제니퍼 서버에서 java.lang.OutOfMemoryError 예외가 발생할 수 있다.

이를 해결하기 위해서 제니퍼 에이전트의 uri_starter 옵션으로 URI의 특정 패턴을 시작점으로 애플리케이션 서비스 이름을 도출할 수 있다. 앞의 예제를 위해서는 아래와 같이 설정하면,

```
uri_starter = )/
```

애플리케이션 서비스이름은 다음과 같이 된다.

```
)/Application/SessionState.aspx
```

6.5. 애플리케이션 서비스 모니터링

제니퍼는 자바 애플리케이션을 모니터링 할 때 2가지 관점에서 성능 데이터를 수집한다. 하나는 트랜잭션별 성능 모니터링이고 다른 하나는 서비스별 성능 모니터링이다.

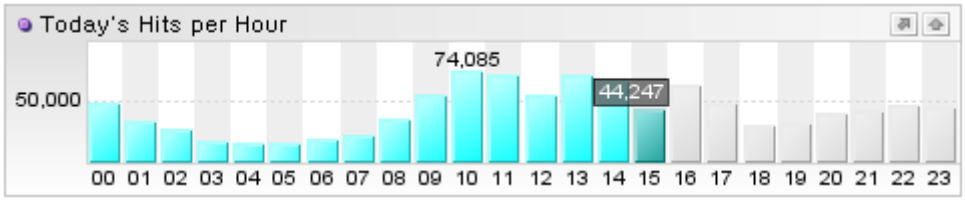
트랜잭션별 성능 모니터링은 클라이언트 요청에 대한 처리 현황을 각 트랜잭션별로 모니터링 하는 것으로 X-View를 이용한다. 자세한 사항은 X-View와 트렌잭션을 참조한다.

서비스별 성능 모니터링은 단위(10분) 시간 동안 수행되었던 트랜잭션들을 애플리케이션 서비스 단위로 통계화한 모니터링이다. 자세한 사항은 [실시간 모니터링 - 애플리케이션]과 [통계 분석 - 애플리케이션]을 참조한다.

6.5.1. 호출 건수

호출 건수는 자바 애플리케이션의 전체 트랜잭션 처리 건수를 의미한다. 호출 건수는 일반 성능 데이터로 5분 동안의 호출 건수가 PERF_X_01~31 테이블의 HIT 칼럼에 저장되고, 특정 날짜의 호출 건수 변화 추이는 막대 차트를 통해서 확인할 수 있다.

그림 6-1: 금일 시간당 호출 건수



Notice: 트랜잭션별 호출 건수, 평균 응답 시간 등의 성능 데이터를 애플리케이션 처리 현황 통계 데이터라고 한다. 자세한 사항은 [성능 현황과 보고서]를 참조한다. 그리고 개별 트랜잭션에 대한 자세한 분석은 X-View를 통해서 이루어진다. 자세한 사항은 [X-View와 프로파일링]를 참조한다.

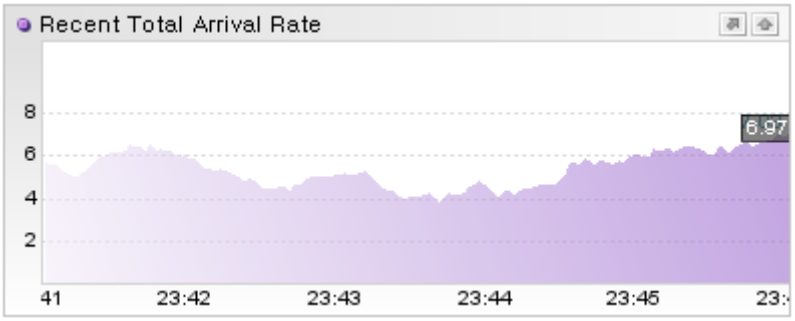
6.5.2. 서비스 요청률과 서비스 처리율

서비스 요청률은 자바 애플리케이션의 1초당 트랜잭션 시작 건수를 의미하고, 서비스 처리율은 자바 애플리케이션의 1초당 트랜잭션 종료 건수를 의미한다.

대부분의 경우에는 서비스 요청률과 서비스 처리율이 비슷한 수치를 보이지만 호출 건수가 많거나 성능 저하가 발생하는 경우에는 서비스 요청률과 서비스 처리율의 차이가 클 수 있다.

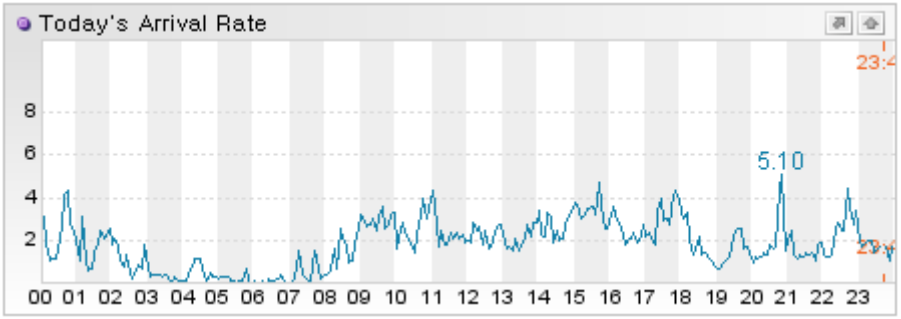
서비스 요청률과 서비스 처리율은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 30초 평균 값을 스피드 바, 스피드 미터, 런타임 라인 차트를 통해서 확인할 수 있다.

그림 6-2: 최근 서비스 요청률



또한 서비스 요청률과 서비스 처리율에 대한 5분 평균 값은 PERF_X_01~31 테이블의 ARRIVAL_RATE 칼럼과 SERVICE_RATE 칼럼에 저장되고, 특정 날짜의 서비스 요청률과 서비스 처리율의 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 6-3: 금일 서비스 요청률

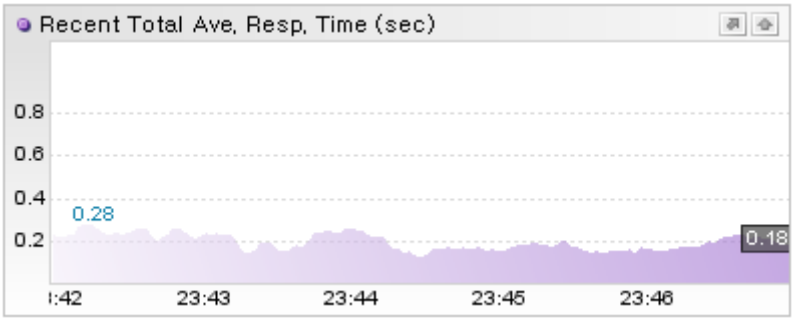


6.5.3. 평균 응답 시간

평균 응답 시간은 전체 트랜잭션의 응답 시간의 합을 호출 건수로 나눈 것으로 단위는 초이다.

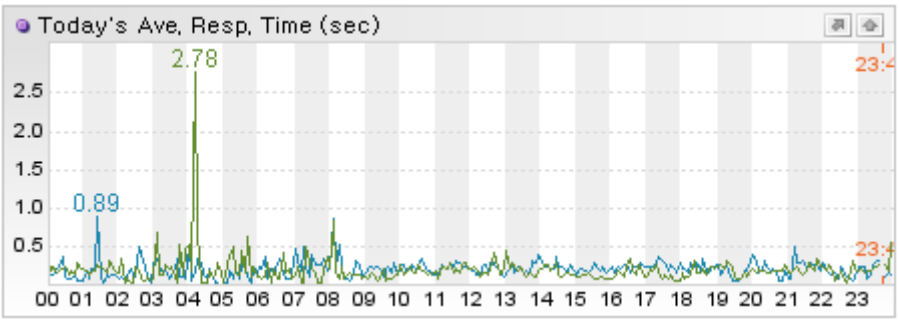
평균 응답 시간은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 30초 평균 값을 런타임 라인 차트를 통해서 확인할 수 있다.

그림 6-4: 최근 평균 응답 시간



또한 평균 응답 시간에 대한 5분 평균 값은 PERF_X_01~31 테이블의 RESPONSE_TIME 칼럼에 저장되고, 특정 날짜의 평균 응답 시간의 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 6-5: 금일 평균 응답 시간



6.5.4. 트랜잭션과 예외 처리

트랜잭션 처리 과정에서 발생한 예외를 감지하고 수집하는 것은 서비스 모니터링에서 중요한 영역을 차지한다. 그런데 트랜잭션이 수행하면서 발생한 모든 예외를 감지하고 추적하면 전체 애플리케이션 성능에 부정적인 영향을 미칠 수 있다. 따라서 제니퍼는 핵심적인 예외만을 감지하고 처리한다.

제니퍼가 예외를 감지하고 수집하는 경우는 다음과 같다.

- 트랜잭션 종료 시점, 즉 애플리케이션 서비스 시작점을 의미하는 메소드에서 예외가 발생하거나 트랜잭션을 처리하는 과정에서 발생한 예외가 이 메소드 까지 전달되는 경우
- 외부 트랜잭션 시작점을 의미하는 메소드에서 예외가 발생하거나 외부 트랜잭션을 처리하는 과정에서 발생한 예외가 이 메소드 까지 전달되는 경우
- JDBC를 처리하는 과정에서 예외가 발생한 경우

앞의 경우를 제외하고 트랜잭션 처리 과정에서 발생한 예외를 catch 문을 통해서 처리했다면 제니퍼는 이를 예외로 감지하지 않는다.

6.5.4.1. 트랜잭션 성공과 실패

트랜잭션 종료시에 다음 예외가 감지되면 트랜잭션이 실패한 것으로 간주한다.

- ERROR_RECURRSIVE_CALL
- ERROR_UNCAUGHT_EXCEPTION
- ERROR_HTTP_IO_EXCEPTION
- ERROR_OUTOFMEMORY
- ERROR_UNKNOWN_ERROR
- ERROR_PLC_REJECTED
- ERROR_JDBC_CONNECTION_FAIL

Notice: 단순한 응답 시간 지연으로 예외가 발생해도, 트랜잭션 자체는 성공한 것으로 간주한다. 또한 외부 트랜잭션과 DB 처리 과정에서 예외가 발생해도 코드 상에서 catch 문으로 처리를 했다면 트랜잭션 자체는 성공한 것으로 간주한다

6.5.4.2. 외부 트랜잭션과 예외

트랜잭션 처리 과정에서 감지할 수 있는 외부 트랜잭션 관련 예외는 다음과 같다.

- WARNING_TX_CALL_EXCEPTION
- WARNING_TX_BAD_RESPONSE

6.5.4.3. JDBC 처리와 예외

트랜잭션 처리 과정에서 감지할 수 있는 JDBC 관련 예외는 다음과 같다.

ERROR_JDBC_CONNECTION_FAIL

WARNING_JDBC_TOOMANY_RS_NEXT

WARNING_JDBC_STMT_EXCEPTION

WARNING_JDBC_PSTMT_EXCEPTION

WARNING_JDBC_BAD_RESPONSE

WARNING_JDBC_UN_COMMIT_ROLLBACK

WARNING_JDBC_CONN_ILLEGAL_ACCESS

6.5.4.4. 임의의 예외 감지

트랜잭션 처리 과정에서 발생한 임의의 예외가 catch 문으로 처리가 되더라도 CustomTrace 어뎁터를 이용해서 이를 감지할 수 있다. 자세한 사항은 Customtrace 어뎁터를 참조한다.

CustomTrace 어뎁터로 예외를 감지하더라도 해당 트랜잭션은 성공한 것으로 간주한다.

6.5.5. 서비스 모니터링과 예외 발생

6.5.5.1. 응답 시간 지연

트랜잭션의 응답 시간이 임계치를 초과하면 WARNING_APP_BAD_RESPONSE 예외가 발생한다. 임계치는 제니퍼 에이전트의 app_bad_responsetime 옵션으로 설정한다. 기본 값은 30000이고 단위는 밀리 세컨드이다.

```
app_bad_responsetime = 30000
```

6.5.5.2. 서비스 무한 재귀 호출 감지

자바 애플리케이션에서 서블릿 혹은 JSP가 임계치(기본 값은 50,000)를 초과하여 다른 서블릿 혹은 JSP를 계속 호출하면 ERROR_RECURRSIVE_CALL 예외가 발생한다. 제니퍼 에이전트의 recurrsive_call_max_count 옵션을 통해서 임계치를 설정한다.

```
recurrsive_call_max_count = 50000
```

서블릿 혹은 JSP가 다른 서블릿 혹은 JSP를 호출한다는 것은 자바 코드 상에서 javax.servelt.RequestDispatcher 객체의 forward 혹은 include 메소드를 호출하는 것을 의미한다.

이 예외의 원인을 파악하기 위해서 제니퍼 에이전트 로그 파일에 관련 스택트레이스를 기록하려면, 제니퍼 에이전트의 recurrsive_call_trace 옵션을 true로 지정한다. 기본 값은 false이다.

```
recurrsive_call_trace = true
```

기본적으로 제니퍼 에이전트 로그 파일에 기록되는 스택트레이스 글자수를 10,000자로 제한한다. 제니퍼 에이전트의 recurrsive_call_trace_size 옵션을 통해서 글자 수를 설정할 수 있다.

```
recurrsive_call_trace_size = 10000
```

Notice: 이 예외가 발생하면 해당 트랜잭션을 종료시킨다. 따라서 너무 작은 값을 recurrsive_call_max_count 옵션에 설정하면 서비스가 정상적으로 이루어지지 않을 수 있다.

6.6. 액티브 서비스 모니터링

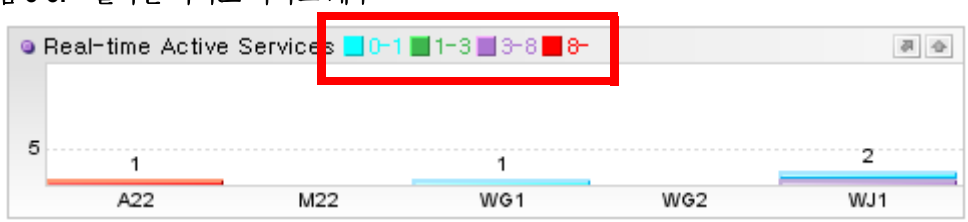
액티브 서비스는 자바 애플리케이션이 현재 처리 중인 트랜잭션을 의미한다. 즉 애플리케이션 시작점을 나타내는 메소드가 실행 중인 상태에 있는 트랜잭션을 액티브 서비스라고 한다.

6.6.1. 액티브 서비스 개수

액티브 서비스 개수는 자바 애플리케이션이 현재 처리 중인 트랜잭션의 개수를 의미한다.

액티브 서비스 개수는 제니퍼 에이전트가 수집하는 일반 성능 데이터로 현재 값은 이퀄라이저 차트로, 실시간 30초 평균 값은 런타임 라인 차트를 통해서 확인할 수 있다.

그림 6-6: 실시간 액티브 서비스 개수



액티브 서비스 개수는 경과 시간 구간으로 구분되고, 이퀄라이저 차트는 경과 시간 구간을 색상별로 구분하여 표시한다. 다음은 경과 시간 구간에 대한 설명이다.

표 6-1: 경과 시간 구간

경과 시간 구간	설명
0-1	경과 시간이 1초 미만인 경우

표 6-1: 경과 시간 구간

경과 시간 구간	설명
1-3	경과 시간이 1초 이상 3초 미만인 경우
3-8	경과 시간이 3초 이상 8초 미만인 경우
8-	경과 시간이 8초 이상인 경우

이 구간은 제니퍼 에이전트와 제니퍼 서버의 active_graph_interval 옵션으로 설정한다. 단위는 밀리 세컨드이다.

```
active_graph_interval = 0,1000,3000,8000
```

Notice: 이 옵션을 변경하려면 제니퍼 에이전트와 제니퍼 서버의 옵션을 함께 수정해야 한다.

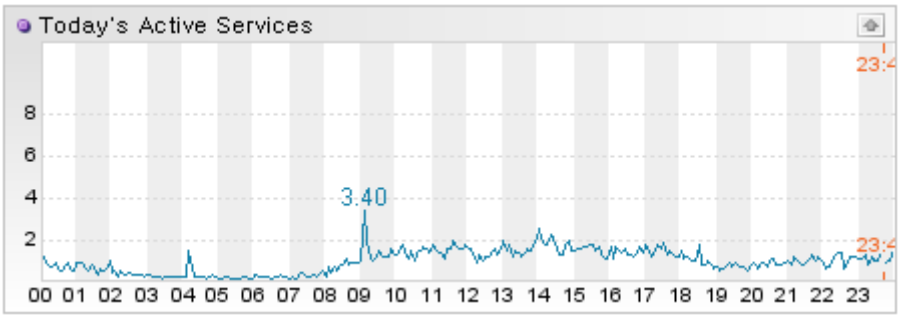
기본적으로 8초 이상인 경우가 이퀄라이저 차트의 하단에 나타난다. 이를 맨 상단에 표시하려면 제니퍼 서버의 ui_active_service_reverse 옵션을 true로 설정한다.

```
ui_active_service_reverse = true
```

이 옵션을 수정한 경우에는 로그인을 다시해야 한다.

또한 액티브 서비스 개수에 대한 5분 평균 값은 PERF_X_01~31 테이블의 ACTIVE_SERVICE 칼럼에 저장되고, 특정 날짜의 액티브 서비스 개수 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 6-7: 금일 액티브 서비스 개수



6.6.2. 액티브 서비스 모니터링과 경보 발령

제니퍼 서버가 모니터링하는 모든 자바 애플리케이션의 5 초 동안의 액티브 서비스 개수의 합이 임계치(기본 값은 70 개)를 초과하면 ERROR_SERVICE_QUEUING 경보를 발령한다. 임계치는 제니퍼 서버의 active_service_alert_limit 옵션을 통해서 설정한다.

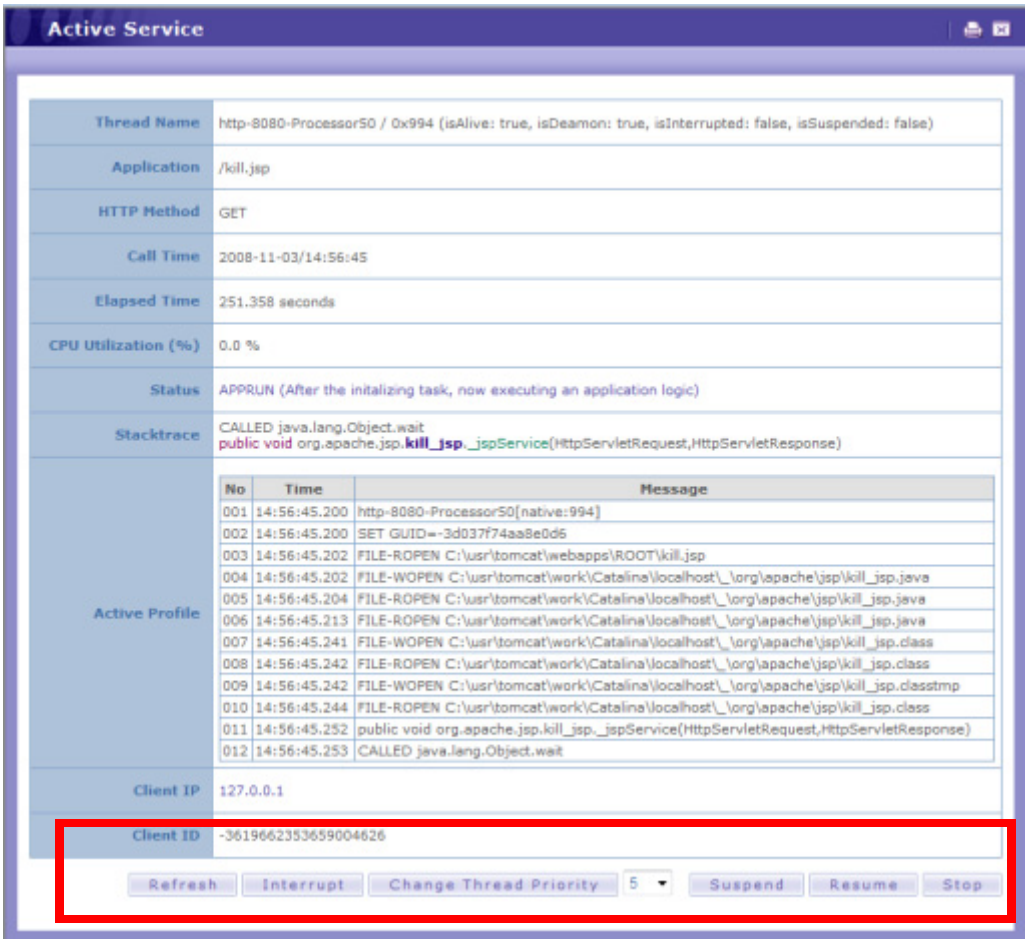
```
active_service_alert_limit = 70
```

6.6.3. 액티브 서비스 중단하기

제니퍼는 트랜잭션을 처리하는 자바 쓰레드가 비정상적인 상태(wait 혹은 sleep 등)에서 멈추어 있는 경우에 해당 쓰레드를 중단시키는 기능을 제공한다. 비정상적으로 장시간 수행되는 트랜잭션을 중단해야 하는 이유는 멈추어 있는 쓰레드가 점유한 한정된 자원을 해제하기 위함이다. 반대로 이야기 하면 특별히 부족한 자원이 없는 경우에는 해당 쓰레드를 강제로 종료할 필요가 없다.

액티브 서비스 목록에서 장시간 수행중인 트랜잭션의 애플리케이션 이름을 클릭하면 다음과 같은 팝업 창이 나타난다.

그림 6-8: 액티브 서비스 상세



바로 이 액티브 서비스 팝업 창에서 트랜잭션을 처리하는 쓰레드를 제어할 수 있다. 즉 쓰레드를 일시 정지(suspend) 혹은 정지(stop)시킬 수 있다. 사용자가 속한 그룹이 killtrhead 권한을 가지고 있는 경우에만 해당 버튼들이 나타난다.

Notice: 트랜잭션을 처리하는 쓰레드가 Socket Read 상태에서 멈추어 있으면 Socket Read 상태가 해제되기 전까지 쓰레드가 중단되지 않는다.

Notice: WAS에 따라서는 예측할 수 없는 부작용이 발생할 수 있다. 웹로직 8.x/9.x, 웹스피어4.x/5.x/6.x, 톰캣 5.x 등에서 테스트가 되었다.

6.6.3.1. 경과 시간이 지연된 액티브 서비스 자동 종료

경과 시간이 임계치를 초과해서 장시간 지연되는 경우에는 액티브 서비스를 자동으로 종료할 수 있다. 이를 위해서는 제니퍼 에이전트의 enable_long_running_thread_auto_kill 옵션을 true로 설정한다.

```
enable_long_running_thread_auto_kill = true
```

임계치는 제니퍼 에이전트의 long_running_thread_auto_kill_timeout 옵션으로 설정한다. 단위는 밀리 세컨드이다.

```
long_running_thread_auto_kill_timeout = 300000
```

6.6.4. 액티브 파라미터

[실시간 모니터링 | 애플리케이션] 메뉴에서 액티브 서비스 목록을 확인할 때 애플리케이션 서비스 이름에 HTTP 파라미터 값을 추가해서 확인할 수 있다.

Notice: 화면에서만 애플리케이션 서비스 이름에 HTTP 파라미터 값을 추가할 뿐 실제로 애플리케이션 서비스 이름을 변경하는 것은 아니다.

이를 위해서 애플리케이션 서비스 이름에 추가해서 보여줄 파라미터 이름을 제니퍼 에이전트의 http_post_request_tracking_list 옵션으로 설정한다.

```
http_post_request_tracking_list = txid
```

그리고 액티브 서비스 목록에 나타나는 애플리케이션 서비스 이름에 트랜잭션을 처리하는 과정에서 호출한 특정 클래스의 특정 메소드의 첫번째 java.lang.String 유형의 파라미터를 애플리케이션 서비스 이름에 추가해서 보여줄 수 있다. 수정한 active_param으로 시작하는 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

기본적으로 java.lang.String 유형의 첫번째 파라미터 값을 애플리케이션 서비스 이름에 추가한다. 그런데 byte[] 유형의 첫번째 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 제니퍼 에이전트의 active_param_type 옵션을 byte[]로 설정한다. 기본 값은 java.lang.String이다.

```
active_param_type = byte[]
```

예를 들어, 애플리케이션 서비스 이름에 example.BizAction 클래스의 모든 메소드의 파라미터 값을 추가하려면 제니퍼 에이전트의 active_param_class 옵션으로 설정한다.

```
active_param_class = example.BizAction
```

그리고 example.LogAction 클래스의 모든 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 [;]을 구분자로 active_param_class 옵션에 설정한다.
active_param_class 옵션뿐만 아니라 애플리케이션 서비스 이름과 관련한 모든 옵션에 2개 이상을 설정하려면 [;]을 구분자로 사용한다.

```
active_param_class = example.BizAction;example.LogAction
```

특정 메소드의 파라미터 값만을 애플리케이션 서비스 이름에 추가하려면 제니퍼 에이전트의 active_param_target_method 옵션으로 설정한다.

```
active_param_target_method = process
```

그런데 애플리케이션 서비스 이름에 추가할 클래스들 중에서 이름이 동일한 메소드가 있고, 이중 특정 클래스의 특정 메소드의 파라미터 값만을 애플리케이션 서비스 이름에 추가하려면 다음과 같이 구체적으로 설정한다.

```
active_param_target_method = example.BizAction.process
```

반대로 특정 메소드의 파라미터 값만을 애플리케이션 서비스 이름에 추가하지 않으려면 제니퍼 에이전트의 active_param_ignore_method 옵션으로 설정한다.

```
active_param_ignore_method = example.BizAction.someMethod
```

메소드의 접근자를 통해서도 애플리케이션 서비스 이름에 추가할 메소드의 파라미터 값을 설정할 수 있다. 예를 들어 public 접근자와 접근자가 없는 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 다음과 같이 설정한다.

```
active_param_access_method = public;none
```

active_param_access_method 옵션에 설정이 가능한 값은 다음과 같다.

- public - public 접근자
- protected - protected 접근자
- private - private 접근자
- none - 접근자가 없는 경우

그리고 특정 클래스를 상속한 모든 클래스의 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 제니퍼 에이전트의 active_param_super 옵션을 사용한다. 예를 들

어, example.BaseAction을 상속한 모든 클래스의 모든 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 다음과 같이 설정한다.

```
active_param_super = example.BaseAction
```

그러나 이 경우에 직접적으로 상속받은 클래스의 메소드의 파라미터 값만이 애플리케이션 서비스 이름에 추가된다. 예를 들어, A 클래스가 example.BaseAction 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스의 메소드의 파라미터 값은 애플리케이션 서비스 이름에 추가되지만 B 클래스의 메소드의 파라미터 값은 애플리케이션 서비스 이름에 추가되지 않는다.

그리고 특정 인터페이스를 구현한 모든 클래스의 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 제니퍼 에이전트의 active_param_interface 옵션을 사용한다. 예를 들어, example.IAction 인터페이스를 구현한 모든 클래스의 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 다음과 같이 설정한다.

```
active_param_interface = example.IAction
```

그러나 이 경우에 직접적으로 구현한 클래스의 메소드의 파라미터 값만이 애플리케이션 서비스 이름에 추가된다. 예를 들어, A 클래스가 example.IAction 인터페이스를 구현하고 B 클래스가 A 클래스를 상속했다면, A 클래스의 메소드의 파라미터 값은 애플리케이션 서비스 이름에 추가되지만 B 클래스의 메소드의 파라미터 값은 애플리케이션 서비스 이름에 추가되지 않는다.

그리고 클래스의 이름을 이용해서도 애플리케이션 서비스 이름에 추가될 클래스의 메소드의 파라미터 값을 설정할 수 있다. 이 경우에는 제니퍼 에이전트의 active_param_prefix, active_param_postfix, active_param_ignore_prefix 옵션을 사용한다. 예를 들어, 이름이 example.action으로 시작하는 모든 클래스의 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 다음과 같이 한다.

```
active_param_prefix = example.action
```

또한 이름이 Action으로 끝나는 모든 클래스의 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하려면 다음과 같이 한다.

```
active_param_postfix = Action
```

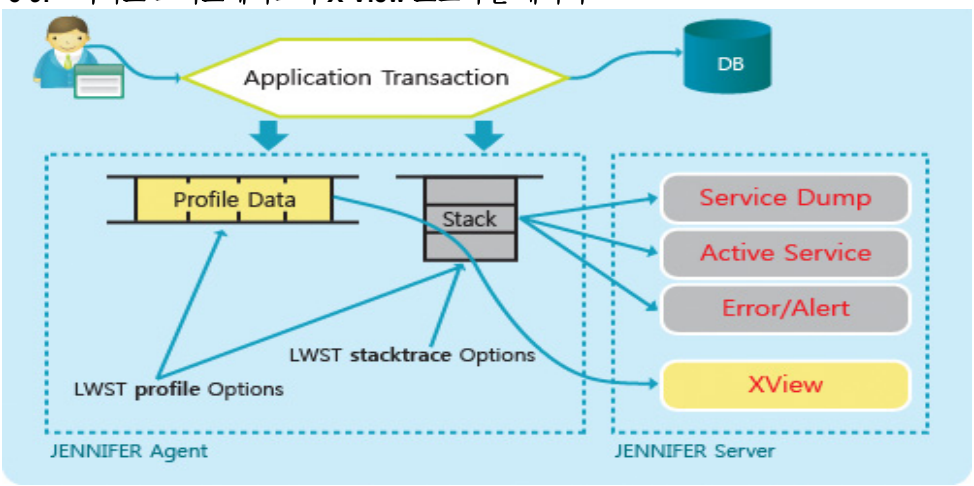
그리고 특정 이름으로 시작하는 클래스의 메소드의 파라미터 값을 애플리케이션 서비스 이름에 추가하지 않으려면 제니퍼 에이전트의 active_param_ignore_prefix 옵션을 사용한다. 이 옵션은 다른 모든 옵션에 우선한다.

```
active_param_ignore_prefix =
```

6.6.5. 액티브 스택트레이스

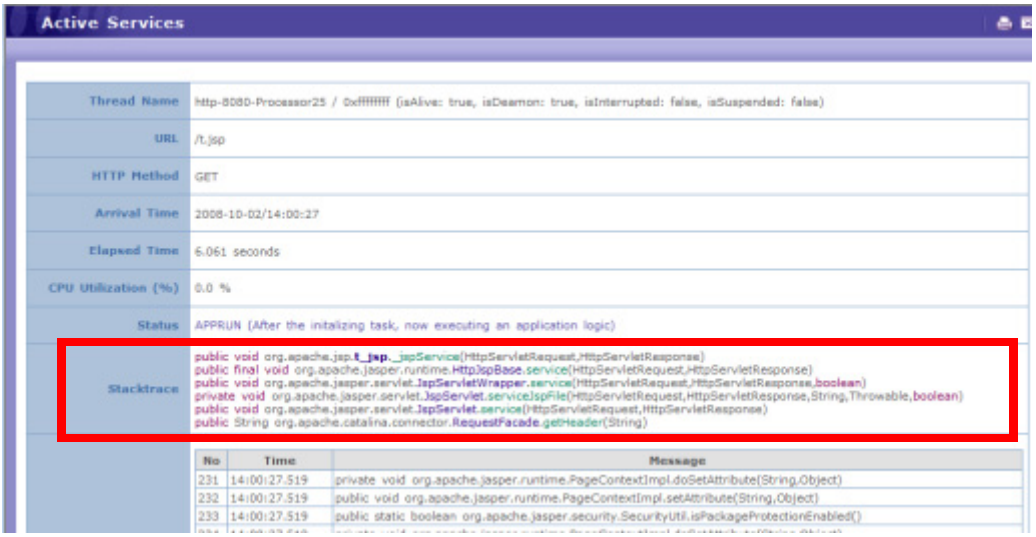
액티브 스택트레이스는 액티브 서비스의 스택트레이스를 추출하는 기능이다. X-View 프로파일 데이터와 유사하지만 X-View 프로파일 데이터는 영구적으로 저장되어 트랜잭션이 종료한 후에도 조회할 수 있는 반면에, 액티브 스택트레이스는 휘발성 데이터로 트랜잭션이 종료하면 사라진다.

그림 6-9: 액티브 스택트레이스와 X-View 프로파일 데이터



액티브 스택트레이스는 다음과 같이 액티브 서비스 상세 정보를 보여주는 팝업 창에서 확인할 수 있다.

그림 6-10: 액티브 스택트레이스



그러나 트랜잭션을 처리하는 과정에서 호출한 모든 메소드를 스택트레이스로 보여주지는 않는다. 제니퍼 에이전트의 stacktrace 시작하는 옵션으로 설정한 클래스의 메소드만이 스택트레이스에 포함된다. 수정한 stacktrace으로 시작하는 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

예를 들어, example.BizAction 클래스의 모든 메소드를 스택트레이스에 포함시키려면 제니퍼 에이전트의 stacktrace_class 옵션으로 설정한다.

```
stacktrace_class = example.BizAction
```

그리고 example.LogAction 클래스의 모든 메소드도 스택트레이스에 포함시키려면 [;]을 구분자로 stacktrace_class 옵션에 설정한다. stacktrace_class 옵션뿐만 아니라 스택트레이스와 관련한 모든 옵션에 2개 이상을 설정하려면 [;]을 구분자로 사용한다.

```
stacktrace_class = example.BizAction;example.LogAction
```

메소드의 접근자를 통해서도 스택트레이스에 포함시킬 메소드를 설정할 수 있다. 예를 들어, public 접근자와 접근자가 없는 메소드만을 스택트레이스에 포함시키려면 다음과 같이 설정한다.

```
stacktrace_access_method = public;none
```

stacktrace_access_method 옵션에 설정이 가능한 값은 다음과 같다.

- public - public 접근자
- protected - protected 접근자
- private - private 접근자
- none - 접근자가 없는 경우

그리고 특정 클래스를 상속한 모든 클래스의 메소드를 스택트레이스에 포함시키려면 제니퍼 에이전트의 stacktrace_super 옵션을 사용한다. 예를 들어, example.BaseAction을 상속한 모든 클래스의 메소드를 스택트레이스에 포함시키려면 다음과 같이 설정한다.

```
stacktrace_super = example.BaseAction
```

그러나 이 경우에 직접적으로 상속받은 클래스의 메소드만을 스택트레이스에 포함시킨다. 예를 들어, A 클래스가 example.BaseAction 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스의 메소드는 스택트레이스에 포함시키지만 B 클래스의 메소드는 스택트레이스에 포함시키지 않는다.

그리고 특정 인터페이스를 구현한 모든 클래스의 메소드를 스택트레이스에 포함시키려면 제니퍼 에이전트의 stacktrace_interface 옵션을 사용한다. 예를 들어, example.IAction

인터페이스를 구현한 모든 클래스의 메소드를 스택트레이스에 포함시키려면 다음과 같이 설정한다.

```
stacktrace_interface = example.IAction
```

그러나 이 경우에 직접적으로 구현한 클래스의 메소드만을 스택트레이스에 포함시킨다. 예를 들어, A 클래스가 example.IAction 인터페이스를 구현하고 B 클래스가 A 클래스를 상속했다면, A 클래스의 메소드는 스택트레이스에 포함시키지만 B 클래스의 메소드는 스택트레이스에 포함시키지 않는다.

그리고 클래스의 이름을 이용해서도 스택트레이스에 포함될 클래스의 메소드를 설정할 수 있다. 이 경우에는 제니퍼 에이전트의 stacktrace_prefix, stacktrace_postfix 옵션을 사용한다. 예를 들어, 이름이 example.action으로 시작하는 모든 클래스의 메소드를 스택트레이스에 추가시키려면 다음과 같이 한다.

```
stacktrace_prefix = example.action
```

또한 이름이 Action으로 끝나는 모든 클래스의 메소드를 스택트레이스에 추가시키려면 다음과 같이 한다.

```
stacktrace_postfix = Action
```

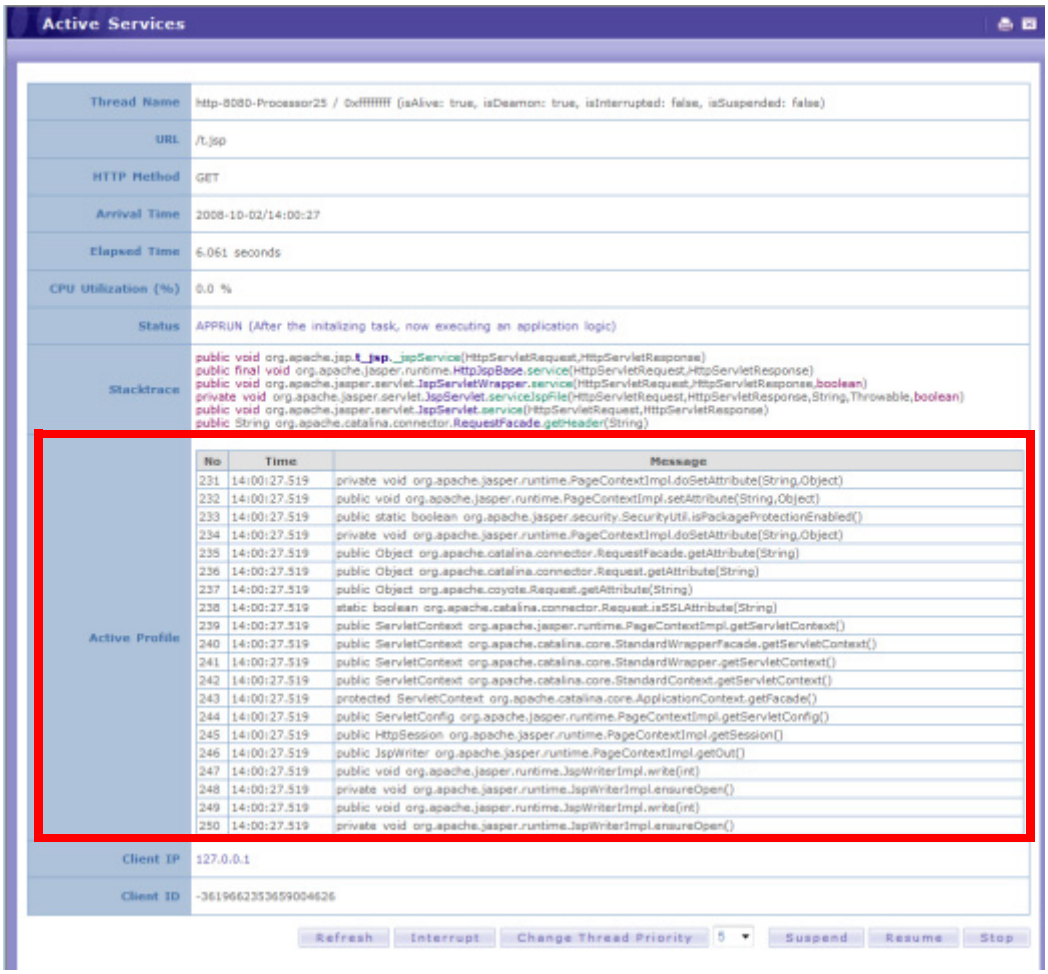
마지막으로 제니퍼 에이전트의 stacktrace_stacksize 옵션으로 스택트레이스에 포함시킬 메소드의 개수를 설정할 수 있다. 기본 값은 300이다.

```
stacktrace_stacksize = 300
```

6.6.6. 액티브 프로파일

액티브 서비스의 액티브한 X-View 프로파일 데이터를 다음과 같이 액티브 서비스 상세 정보를 보여주는 팝업 창에서 확인할 수 있다.

그림 6-11: 액티브 프로파일



화면에 나타날 X-View 프로파일 데이터의 프로파일 항목 개수를 제니퍼 서버의 active_profile_max_line 옵션으로 설정할 수 있다.

```
active_profile_max_line = 50
```

6.7. 부하량 제어(PLC)

PLC(Peak Load Control)는 액티브 서비스 개수와 비즈니스 중요 그룹 및 성능 저하 그룹을 기준으로 부하량 폭주를 조절하는 기능이다.

6.7.1. PLC 기본 설정

우선 PLC를 사용하려면 제니퍼 에이전트의 set_limit_active_service 옵션을 true로 설정한다.

```
set_limit_active_service = true
```

PLC를 사용하면 액티브 서비스 개수가 제니퍼 에이전트의 max_num_of_active_service 옵션으로 설정한 임계치를 초과하는 경우에 서비스 큐잉 현상을 방지하기 위해서 추가적인 요청을 거절한다.

```
max_num_of_active_service = 100
```

PLC에 의해 서비스 요청이 거절되면 부하량이 특정 임계치를 초과하여 추가적인 요청을 처리할 수 없음을 자바 애플리케이션 사용자에게 공지해야 한다. 공지 방법에는 두가지가 있으며 제니퍼 에이전트의 request_reject_type 옵션으로 공지 방법을 설정한다.

```
request_reject_type = [message|redirect]
```

request_reject_type 옵션으로 message를 설정하면 제니퍼 에이전트의 request_reject_message 옵션에 설정한 텍스트를 사용자에게 보여준다.

```
request_reject_message = Workload so high. Please, try again later!
```

request_reject_type 옵션으로 redirect를 설정하면 제니퍼 에이전트의 request_reject_redirect_url 옵션으로 설정한 페이지를 사용자에게 보여준다.

```
request_reject_redirect_url = /error.html
```

Notice: request_reject_redirect_url 옵션으로 지정한 페이지가 PLC에 의해서 재귀적으로 거절되는 현상을 방지하기 위해서 request_reject_redirect_url 옵션에 HTML 페이지를 지정하는 것을 권장한다.

6.7.2. PLC 그룹 설정

비즈니스 중요도 및 성능 저하에 미치는 영향을 기준으로, 애플리케이션을 분류하여 해당 애플리케이션 그룹 별로 PLC 설정을 다르게 적용할 수 있다.

6.7.2.1. 비즈니스 중요 그룹 지정

부하량이 증가할 때 자바 애플리케이션의 처리 역량을 업무 중요도에 따라서 다르게 분배할 수 있다. 즉, 비즈니스 중요 그룹 지정을 통해서 부하량이 증가할 때 중요하지 않은 업무보다 중요한 업무가 처리될 확률을 높일 수 있다.

비즈니스 중요 그룹은 다음과 같이 설정한다. 우선 제니퍼 에이전트의 biz_group.num 옵션으로 전체 비즈니스 중요 그룹에 대한 액티브 서비스 개수 임계치를 설정한다.

```
biz_group.num = 20
```

Notice: max_num_of_active_service 옵션은 전체 액티브 서비스 개수에 대한 임계치를 의미하고 biz_group.num 옵션은 전체 비즈니스 중요 그룹의 액티브 서비스 개수에 대한 임계치를 의미한다.

개별 비즈니스 중요 그룹은 제니퍼 에이전트의 biz_group.#.num 옵션과 biz_group.#.list 옵션으로 설정한다. # 대신에 임의의 비즈니스 중요 그룹 번호를 입력한다.

예를 들어, /biz1.jsp와 /biz.2.jsp를 비즈니스 중요 그룹으로 묶어 액티브 서비스 개수 임계치로 10을 지정하고, /biz3.jsp와 /biz4.jsp를 비즈니스 중요 그룹으로 묶어 액티브 서비스 개수 임계치로 15를 지정하려면 다음과 같이 설정한다.

```
biz_group.1.num = 10
biz_group.1.list = /biz1.jsp, /biz2.jsp
biz_group.2.num = 15
biz_group.2.list = /biz3.jsp, /biz4.jsp
```

Notice: 최대 20개까지의 비즈니스 중요 그룹을 설정할 수 있다.

비즈니스 중요 그룹에 의한 PLC 처리 정책은 다음과 같다.

- 전체 액티브 서비스 개수가 max_num_of_active_service 옵션으로 설정한 임계치를 초과하지 않으면 모든 요청을 처리한다.
- 전체 액티브 서비스 개수가 max_num_of_active_service 옵션으로 설정한 임계치를 초과해도 비즈니스 중요 그룹의 전체 액티브 서비스 개수가 biz_group.num 옵션으로 설정한 임계치를 초과하지 않으면 비즈니스 중요 그룹에 속하는 요청을 처리한다.
- 전체 액티브 서비스 개수가 max_num_of_active_service 옵션으로 설정한 임계치를 초과하고 비즈니스 중요 그룹의 전체 액티브 서비스 개수가 biz_group.num 옵션으로 설정한 임계치를 초과해도 해당 비즈니스 중요 그룹의 액티브 서비스 개수 임계치를 초과하지 않으면 해당 비즈니스 중요 그룹에 속하는 요청을 처리한다.

6.7.2.2. 성능 저하 그룹 지정

성능 저하에 영향을 미치는 애플리케이션들을 성능 저하 그룹으로 지정하고 각각의 성능 저하 그룹에 대한 액티브 서비스 개수 임계치를 설정하여 성능 저하를 유발하는 애플리케이션이 수행되는 개수를 제한할 수 있다.

성능 저하 그룹은 다음과 같이 설정한다. 우선 제니퍼 에이전트의 limit_group.num 옵션으로 전체 성능 저하 그룹에 대한 액티브 서비스 개수 임계치를 설정한다.

```
limit_group.num = 20
```

Notice: max_num_of_active_service 옵션은 전체 액티브 서비스 개수에 대한 임계치를 의미하고 limit_group.num 옵션은 전체 성능 저하 그룹의 액티브 서비스 개수에 대한 임계치를 의미한다.

개별 성능 저하 그룹은 제니퍼 에이전트의 limit_group.#.num 옵션과 limit_group.#.list 옵션으로 설정한다. # 대신에 임의의 성능 저하 그룹 번호를 입력한다. 예를 들어서 /bad1.jsp와 /bad.2.jsp를 성능 저하 그룹으로 묶어 액티브 서비스 개수 임계치로 10을 지정하고, /bad3.jsp와 /bad4.jsp를 성능 저하 그룹으로 묶어 액티브 서비스 개수 임계치로 15를 지정하려면 다음과 같이 설정한다.

```
limit_group.1.num = 10
limit_group.1.list = /bad1.jsp, /bad.jsp
limit_group.2.num = 15
limit_group.2.list = /bad.jsp, /bad.jsp
```

Notice: 최대 20개까지의 성능 저하 그룹을 설정할 수 있다.

성능 저하 그룹에 의한 PLC 처리 정책은 다음과 같다.

- 전체 액티브 서비스 개수가 max_num_of_active_service 옵션으로 설정한 임계치를 초과하면 성능 저하 그룹에 속하는 모든 요청을 거절한다.
- 성능 저하 그룹의 전체 액티브 서비스 개수가 limit_group.num 옵션으로 설정한 임계치를 초과하면 성능 저하 그룹에 속하는 모든 요청을 거절한다.
- limit_group.#.list 옵션으로 설정한 특정 성능 저하 그룹의 액티브 서비스 개수가 limit_group.#.num 옵션으로 설정한 해당 성능 저하 그룹 액티브 서비스 개수 임계치를 초과하면 해당 요청을 거절한다.

6.7.2.3. 애플리케이션 목록 설정

비즈니스 중요 그룹과 성능 저하 그룹에 속하는 애플리케이션 목록은 다음과 같이 애플리케이션 이름을 [,]로 구분하여 설정한다.

```
biz_group.1.list = /biz1.jsp, /biz2.jsp
```

그리고 애플리케이션 이름에 [*]를 사용할 수 있다.

표 6-2: 애플리케이션 이름 설정

애플리케이션 이름	설명
/biz*	애플리케이션 이름이 /biz로 시작하는 모든 애플리케이션이 해당 그룹에 속한다.
*biz.jsp	애플리케이션 이름이 biz.jsp로 끝나는 모든 애플리케이션이 해당 그룹에 속한다.
biz	애플리케이션 이름에 biz가 들어가는 모든 애플리케이션이 해당 그룹에 속한다.

Notice: PLC 애플리케이션 목록에 제니퍼 에이전트의 tx_naming 옵션으로 설정한 애플리케이션 이름을 설정할 수는 없다.

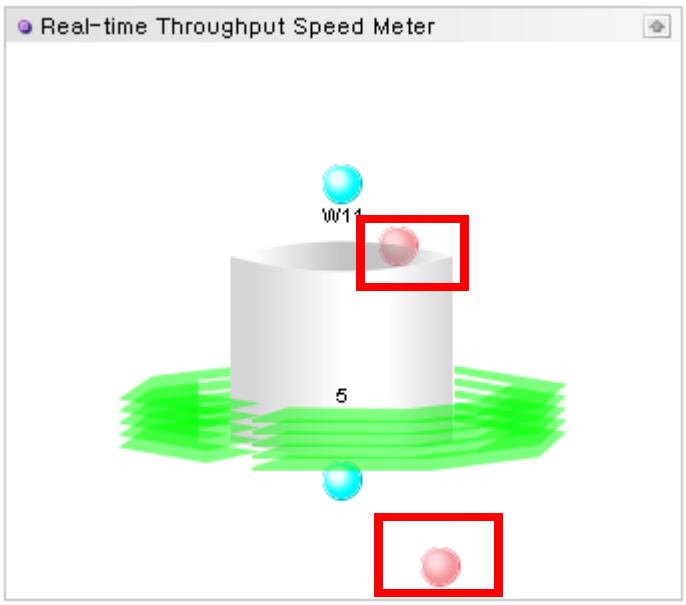
6.7.3. PLC 동작 확인

사용자는 PLC 동작 여부를 제니퍼 서버를 통해서 확인할 수 있다.

첫번째로 PLC에 의해서 요청이 거절되면 ERROR_PLC_REJECTED 예외가 발생한다.

두번째로 스피드 미터를 통해서 PLC 동작을 확인할 수 있다. 정상적인 상황에서는 트랜잭션이 제니퍼 에이전트 기둥 가운데로 지나가지만 PLC에 의해서 요청이 거절되면 에이전트 기둥의 오른쪽 옆을 따라서 흐른다.

그림 6-12: 스피드 미터를 통한 PLC 확인



6.7.4. CRUD 매트릭스

[통계 분석 | CRUD 매트릭스] 메뉴에서 애플리케이션 서비스와 데이터베이스 테이블 혹은 함수와의 상관 관계를 생성, 읽기, 삭제, 업데이트 등의 기준으로 확인할 수 있다.

프로젝트 분석 및 설계 단계에서 정의한 CRUD 매트릭스와 실제 운영 중에 나타난 CRUD 매트릭스의 비교 분석에 이 기능을 사용할 수 있다. 그래서 애플리케이션 품질을 관리하는데 이 기능을 유용하게 사용할 수 있다.

Notice: CRUD 매트릭스 조회는 다량의 데이터를 조회하기 때문에 시간이 많이 걸릴 수 있다.

그림 6-13: CRUD 조회 화면

- 노드 - 노드를 구성한 경우에는 제니퍼 에이전트 선택 영역에서 특정 노드를 선택하여, 해당 노드에 속하는 제니퍼 에이전트들에 대해서만 CRUD 매트릭스를 조회할 수 있다.
- 에이전트 - 제니퍼 에이전트 선택 영역에서 CRUD 매트릭스를 열람할 제니퍼 에이전트를 선택한다. 모든 제니퍼 에이전트들에 대해서 CRUD 매트릭스를 열람하려면 [모두]를 선택한다.
- 날짜 - CRUD 매트릭스를 검색할 날짜를 선택한다.
- 애플리케이션 - 특정 애플리케이션 서비스 이름에 대한 CRUD 매트릭스만을 조회할 수 있다.
- 테이블 - 특정 테이블에 대한 CRUD 매트릭스만을 조회할 수 있다. 테이블 이름 옆에 CRUD의 표기를 통해서 데이터베이스 작업의 성격도 명시해줄 수 있다. C는 신규 생성, R은 읽기, U는 수정, D는 삭제 작업을 의미한다. 예를 들어, ELEMENT 테이블에 대한 읽기와 수정에 대해서 조회하려면 ELEMENT(RU)라고 입력한다.
- 데이터베이스 함수 - 특정 데이터베이스 함수에 대한 CRUD 매트릭스만을 조회할 수 있다.

테이블 입력 필드에 입력할 수 있는 검색 조건의 형식은 다음과 같다.

```
* - 모든 테이블에 대해서 검색을 한다.
*BIZ - 테이블의 이름이 BIZ로 끝나는 테이블에 대해서 검색을 한다.
*BIZ* - 테이블의 이름에 BIZ가 들어가는 테이블에 대해서 검색을 한다
BIZ* - 테이블의 이름이 BIZ로 시작하는 테이블에 대해서 검색을 한다.
```

테이블 이름뿐만 아니라 CRUD 형식에 따른 검색도 가능하다.

```
*(*) - 모든 CRUD 형식에 대해서 검색을 한다.
*(R) - R 형식에 대해서 검색을 한다. 이 조건은 CR, CRUD, RU, RD 등을 모두 포함한다.
*(C|R) - C 혹은 R 형식에 대해서 검색을 한다. |를 생략하고 *(CR)로 입력할 수도 있다.
*(C&U) - C와 U가 모두 있는 형식에 대해서 검색을 한다.
*(C&!R&!U&!D) - 오직 C만 있는 형식에 대해서 검색을 한다.
```

테이블 이름과 CRUD 형식을 함께 사용한 검색도 가능하다.

```
BIZ*(CUD) - 테이블 이름이 BIZ로 시작하는 테이블 중에서 C 혹은 U 혹은 D 형식에 대해서 검색을 한다.
*BIZ(D) - 테이블 이름이 BIZ로 끝나는 테이블 중에서 D 형식에 대해서 검색을 한다.
*BIZ*(R&!C&!U&!D) - 테이블 이름에 BIZ가 들어가는 테이블 중에서 오직 R만 있는 형식에 대해서 검색을 한다.
```

그런데 제니퍼 서버의 disable_app_mapping_db 옵션을 true로 설정한 경우에는 CRUD 매트릭스를 조회할 수 없다.

```
disable_app_mapping_db = true
```

6.8. 서비스 덤프

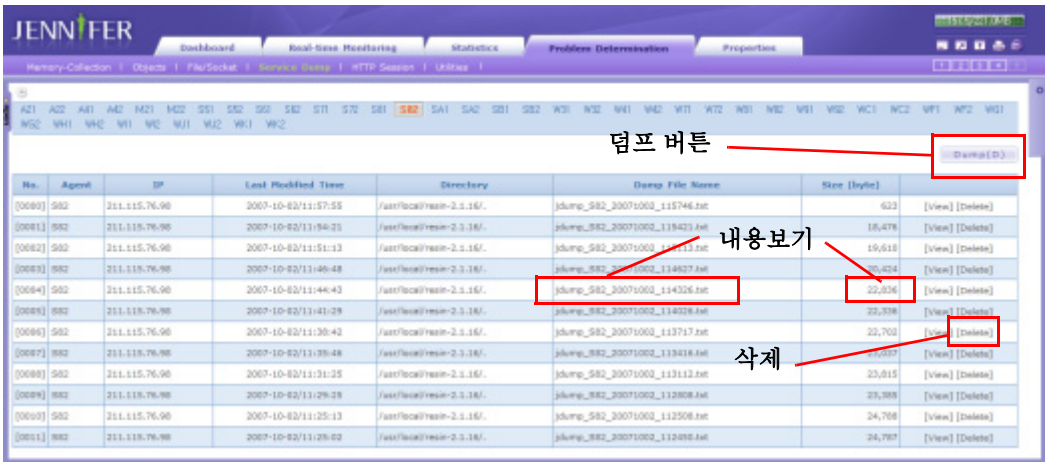
자바 애플리케이션에 대한 실시간 성능 데이터와 액티브 서비스 목록 등의 데이터를 파일에 기록할 수 있다. 이를 서비스 덤프라고 한다.

사용자는 **[장애진단 | 서비스 덤프]** 메뉴에서 특정 제니퍼 에이전트에 대한 서비스 덤프를 기록할 수 있다.

서비스 덤프 파일이 기록되는 디렉토리는 제니퍼 에이전트의 jdump_dir 옵션으로 설정한다. 기본 디렉토리는 작업 디렉토리이다.

```
jdump_dir = ./
```

그림 6-14: 서비스 덤프



- 덤프 버튼 - 해당 버튼을 클릭하면 현재 선택한 제니퍼 에이전트의 주요 데이터를 서비스 덤프 파일에 기록한다.

Notice: [덤프] 버튼을 클릭해도 서비스 덤프 파일 목록에 새로운 서비스 덤프 파일이 보이지 않을 수 있다. 이는 제니퍼 에이전트가 서비스 덤프 파일을 비동기적으로 기록하기 때문에 발생하는 현상이다. 따라서 몇 초 후에 서비스 덤프 파일 목록을 다시 확인하면 새로운 서비스 덤프 파일을 확인할 수 있을 것이다. 따라서 새로운 서비스 덤프 파일이 목록에 바로 나타나지 않는다고 **[덤프]** 버튼을 연속해서 클릭하지 않도록 한다.

- 삭제 - 서비스 덤프 파일을 삭제한다.
- 내용 보기 - 서비스 덤프 파일 이름이나 **[보기]** 링크를 클릭하면 파업 창에서 서비스 덤프 파일 내용을 확인할 수 있다.

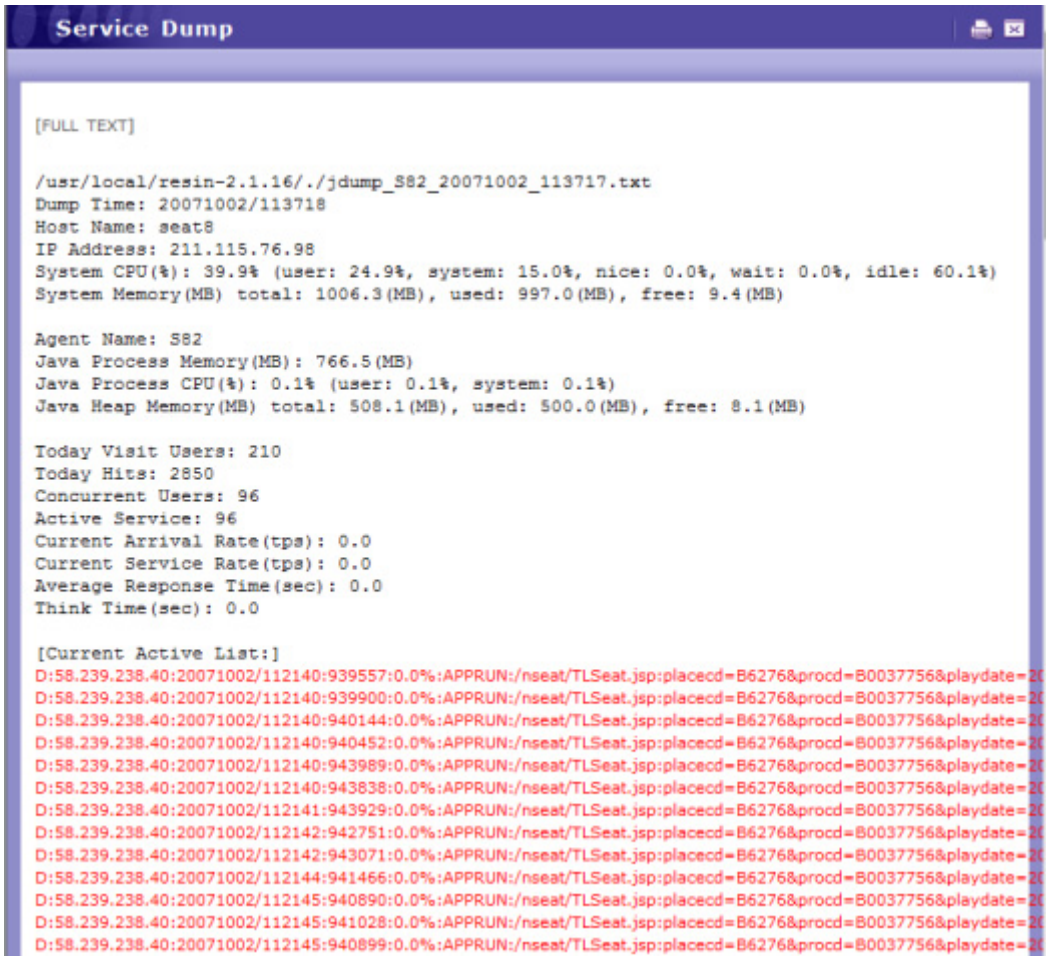
Notice: 서비스 덤프 파일은 제니퍼 서버가 아닌 제니퍼 에이전트의 작업 디렉토리에 기록된다.

6.8.1. 서비스 덤프 파일 내용 보기

서비스 덤프 파일에는 다음 내용들이 기록된다.

- 서비스 덤프 파일을 기록한 시간
- 덤프 시점의 시스템 CPU 사용률, 시스템 메모리 사용량, 자바 힙 메모리 사용량 등의 리소스 성능 데이터
- 덤프 시점의 액티브 서비스 개수, 동시단말 사용자 수, 서비스 요청률 등의 서비스 성능 데이터
- 자바 환경 변수 내역

그림 6-15: 서비스 덤프 파일 내용



6.8.2. 자동 서비스 덤프 기록

액티브 서비스 개수가 지정된 임계치를 초과하면 자동으로 서비스 덤프를 기록하도록 할 수 있다. 이를 위해서는 제니퍼 에이전트의 enable_dump_triggering 옵션을 true로 설정한다.

```
enable_dump_triggering = true
```

그리고 액티브 서비스 개수 임계치는 제니퍼 에이전트의 number_of_dump_trigger 옵션으로 설정한다.

```
number_of_dump_trigger = 90
```

빈번한 서비스 덤프 기록을 방지하기 위해서 자동 서비스 덤프 기록 간격을 제니퍼 에이전트의 dump_trigger_sleep_interval 옵션으로 설정한다. 단위는 밀리 세컨드이다.

```
dump_trigger_sleep_interval = 180000
```

6.9. 고급 서비스 모니터링

SOA(Service Oriented Architecture)와 같은 환경에서 서비스 모니터링을 하는 방법을 설명한다.

6.9.1. 글로벌 트랜잭션 연계 추적

기본적으로 제니퍼는 하나의 자바 쓰레드가 처리하는 트랜잭션을 추적한다. 즉, 하나의 요청이 자바 애플리케이션에 전달되면 해당 요청을 처리하기 위한 작업 쓰레드가 배정되고, 이 쓰레드가 트랜잭션을 처음부터 끝 까지 처리한다. 그런데 EAI(Enterprise Application Integration) 혹은 ESB(Enterprise Service Bus)와 같은 SOA 환경에서는 하나의 요청이 2개 이상의 자바 쓰레드에 의해서 처리되거나, 여러 개의 자바 애플리케이션이 XML 웹서비스를 이용해서 하나의 요청을 처리하기도 한다.

즉, 하나의 요청이 동일 애플리케이션 혹은 여러 애플리케이션에서 다수의 트랜잭션으로 처리되는 것이다. 따라서 서비스 모니터링을 위해서 하나의 요청을 처리한 트랜잭션들의 연관 관계를 제공해줄 필요가 있다. 이를 위해서 제니퍼는 GUID(Globally Unique Identifier)를 사용한다.

제니퍼는 애플리케이션이 처리하는 모든 트랜잭션에 UUID(Universally Unique Identifier)를 부여해서 트랜잭션을 구별한다. 그런데 동일한 요청을 처리하는 트랜잭션들에는 별도로 동일한 GUID를 부여해서 여러 개의 트랜잭션들의 상관 관계를 추적한다. 이를 글로벌 트랜잭션 연계 추적 혹은 GUID 추적이라고 한다.

GUID 추적을 하려면 제니퍼 에이전트의 trace_related_transaction 옵션을 true로 설정해야 한다.

```
trace_related_transaction = true
```

그런데 UUID는 제니퍼가 생성하여 트랜잭션에 부여하지만, GUID는 제니퍼가 아닌 애플리케이션이 관리하고 생성하는 값이기 때문에 이 값을 찾아내서 트랜잭션에 부여해야 한다. 제니퍼는 이를 위한 다양한 방법을 제공한다.

우선 HTTP 해더 정보에서 GUID를 추출해서 이를 트랜잭션에 부여할 수 있다. 이를 위해서는 제니퍼 에이전트의 relatx_guid_keyname 옵션으로 GUID를 나타내는 HTTP 해더 키 이름을 설정한다.

```
relatx_guid_keyname = _J_GUID_
```

앞에서와 같이 설정하면 다음과 같은 코드를 수행하면서 트랜잭션에 GUID와 선행 트랜잭션의 UUID를 부여한다.

```
String guid = request.getHeader( "_J_GUID_" );
```

그리고 트랜잭션을 처리하는 과정에서 호출되는 임의의 클래스의 임의의 메소드의 파라미터를 GUID로 트랜잭션에 부여할 수도 있다.

제니퍼 에이전트의 guid_param 옵션으로 GUID를 추출할 클래스의 메소드를 설정한다. 이 때 메소드의 파라미터 까지 기술해야 한다. 단, 파라미터는 패키지는 이름을 제외한 클래스 이름만으로 설정할 수 있다. 예를 들어, example.Action 클래스의 process 메소드의 파라미터 값을 GUID로 트랜잭션에 부여하려면 다음과 같이 설정한다. 수정한 guid_param 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

```
guid_param = example.Action.process(String, String)
```

이 때 파라미터의 유형은 java.lang.String이거나 byte[]이어야 한다.

기본적으로 첫번째 파라미터 값을 GUID로 트랜잭션에 부여하는데 제니퍼 에이전트의 guid_param_index 옵션으로 GUID로 사용할 파라미터 값을 지정할 수 있다. 0으로 설정하면 첫번째 파라미터의 값, 1로 설정하면 두번째 파라미터의 값을 GUID로 트랜잭션에 부여한다.

```
guid_param_index = 1
```

그리고 제니퍼 에이전트의 guid_return 옵션으로 지정한 메소드가 반환하는 값을 GUID로 트랜잭션에 부여할 수도 있다. 이 때 메소드의 파라미터 까지 기술해야 한다. 단, 파라미터는 패키지는 이름을 제외한 클래스 이름만으로 설정할 수 있다. 예를 들어, example.Action 클래스의 execute 메소드의 반환 값을 GUID로 트랜잭션에 부여하려면 다음과 같이 설정한다. 수정한 guid_return 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

```
guid_return = example.Action.execute(String, String)
```

또한 GUID가 없는 경우에는 현재 트랜잭션의 UUID를 GUID로 부여하도록 설정할 수 있다. 이를 위해서는 제니퍼 에이전트의 enable_guid_from_tuid 옵션을 true로 설정한다.

```
enable_guid_from_tuid = true
```

GUID 추적과 관련한 트랜잭션은 X-View의 GUID 뷰에서 모니터링할 수있다. 자세한 사항은 [GUID]를 참조한다. 그리고 상세한 분석을 하려면 [XVLog를 통한 X-View 트랜잭션 데이터 처리]을 참조한다. 앞에서 설명한 내용으로 GUID를 부여할 수 없다면, 애플리케이션 소스 코드를 수정해서 처리해야 한다. 이와 관련한 자세한 사항은 [ActiveTraceUtil]를 참조한다. .

Notice: 앞에서 설명한 내용으로 GUID를 부여할 수 없다면, 애플리케이션 소스 코드를 수정해서 처리해야 한다. 이와 관련한 자세한 사항은 Activetraceutil를 참조한다.

6.9.2. HTTP 해더와 파라미터 로깅

제니퍼 에이전트 로그 파일에 HTTP 해더와 파라미터 값을 기록하려면 제니퍼 에이전트의 dump_http_hide_all 옵션을 false로 설정한다. 기본 값은 true이다.

```
dump_http_hide_all = true
```

보안 등의 이유로 제니퍼 에이전트 로그 파일에 기록하지 않을 HTTP 해더와 파라미터 이름을 제니퍼 에이전트의 dump_http_hide_key 옵션으로 설정할 수 있다. 2개 이상은 콤마[,]를 구분자로 구분한다. 예를 들어, cookie를 이름으로 하는 HTTP 해더 값을 기록하지 않으려면 다음과 같이 설정한다.

```
dump_http_hide_key = cookie
```

HTTP 해더와 파라미터에 동시에 적용된다.

특정 URI에 대해서만 HTTP 해더를 제니퍼 에이전트 로그 파일에 기록하려면 제니퍼 에이전트의 dump_http_header_url_prefix 옵션으로 해당 URI를 설정한다. 다음과 같이 설정하면 모든 URI에 대해서 로그를 기록한다. 2개 이상은 콤마[,]를 구분자로 구분한다.

```
dump_http_header_url_prefix = /
```

파라미터에 대해서는 제니퍼 에이전트의 dump_http_parameter_url_prefix 옵션을 사용한다. 설정 방법은 dump_http_header_url_prefix 옵션과 동일하다.

6.10. 사용자 모니터링

WAS에 기반한 웹 애플리케이션을 모니터링할 때는 액티브 사용자 수, 동시단말 사용자 수, 방문자 수, 대기 시간 등의 데이터를 수집할 수 있다. 이를 사용자 모니터링이라고 한다.

6.10.1.쿠키를 이용한 사용자 모니터링

과거 클라이언트/서버 환경에서는 단순하게 TCP/IP 연결 개수를 접속자 수로 간주할 수 있었지만 웹 환경의 HTTP 프로토콜은 Connectionless이기 때문에 단순하게 TCP/IP 연결 개수를 접속자 수로 간주할 수 없다. 이 문제를 극복하기 위해서 제니퍼는 쿠키를 이용한 사용자 모니터링을 한다.

제니퍼 에이전트는 트랜잭션이 시작할 때 wmonid 쿠키의 존재 여부를 확인한다. 해당 쿠키가 존재하지 않으면 이 트랜잭션을 새로운 사용자에 의한 요청으로 간주하고 방문자 수를 증가시킨다. 그리고 임의의 값을 생성해서 wmonid 쿠키로 지정한다.

그리고 쿠키는 영구 쿠키이다. 그래서 웹 브라우저를 다시 시작해도 wmonid 쿠키가 유지되므로 방문자 수는 증가하지 않는다. 물론 방문자 수를 계산할 때 wmonid 뿐만 아니라 시간도 고려한다. 자세한 사항은 [방문자 수(128 페이지)]를 참조한다.

그런데 쿠키의 크기에는 제한이 있다. 따라서 웹 애플리케이션이 쿠키를 많이 사용하고 있다면, 제니퍼 에이전트가 wmonid 쿠키를 추가할 때 가능한 쿠키의 크기를 초과하면서 문제가 발생할 수 있다. 이 경우에는 제니퍼 에이전트의 hotfix_remote_address_for_wmonid 옵션을 true로 설정해서 쿠키를 이용한 사용자 모니터링을 비활성화시켜야 한다.

```
hotfix_remote_address_for_wmonid = true
```

hotfix_remote_address_for_wmonid 옵션을 true로 설정하면 쿠키가 아닌 클라이언트 IP로 사용자를 구분하기 때문에 동시단말 사용자 수와, 방문자 수 등의 데이터 정확도가 감소하게 된다.

그리고 도메인 이름으로 서비스하는 여러 개의 웹 애플리케이션을 모니터링할 때 고려해야 하는 것이 있다. 예를 들어, 2개의 웹 애플리케이션을 모니터링하고 있고, 각각의 도메인 이름은 다음과 같다고 가정한다.

```
http://www.jennifersoft.com
http://sales.jennifersoft.com
```

사용자가 http://www.jennifersoft.com 도메인 이름으로 서비스하는 웹 애플리케이션을 처음 사용하면 새로운 wmonid 쿠키 값이 배정된다. 그런데 이 사용자가 http://sales.jennifersoft.com 도메인 이름으로 서비스하는 웹 애플리케이션에 접근해도, 앞의 웹 애플리케이션에서 배정된 wmonid 쿠키 값을 읽을 수 없기 때문에 새로운 wmonid 쿠키 값이 배정된다. 따라서 이 사용자는 동일한 사용자 임에도 불구하고 이중으로 계산된다.

상황에 따라서 앞에서 설명한 방식대로 물리적으로 동일한 사용자라고 하더라도 도메인 이름별로 구분해서 계산하는 것이 적절할 수 있다. 그러나 이를 동일한 사용자로 계산하려면 제니퍼 에이전트의 default_cookie_domain 옵션을 사용한다.

```
default_cookie_domain = .jennifersoft.com
```

default_cookie_domain 옵션 값은 반드시 [.]로 시작해야 한다. 만약 이 옵션을 설정하면 도메인 이름이 아닌 IP 주소로 접속하는 사용자는 동시단말 사용자 수와 방문자 수에 영향을 미치지 못한다.

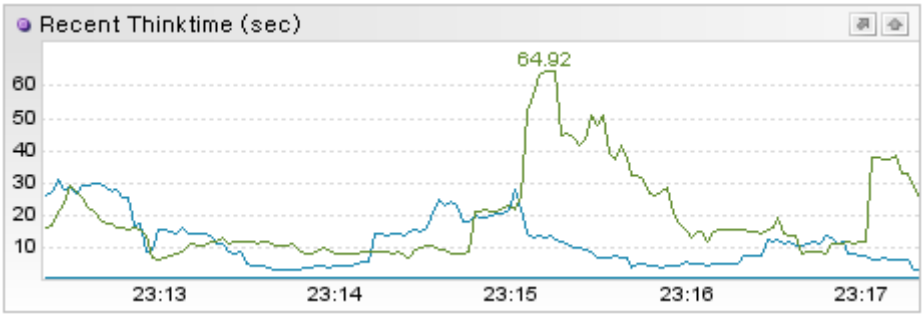
Notice: 앞에서 설명한 내용은 제니퍼 에이전트 별 방문자 수나 동시단말 사용자 수와 관련한 내용이 아니고, 모든 제니퍼 에이전트에 대한 전체 방문자 수와 전체 동시단말 사용자 수와 관련한 것이다.

6.10.2. 대기 시간

대기 시간은 wmonid 쿠키 값이 동일한 트랜잭션들의 평균 호출 간격을 의미한다.

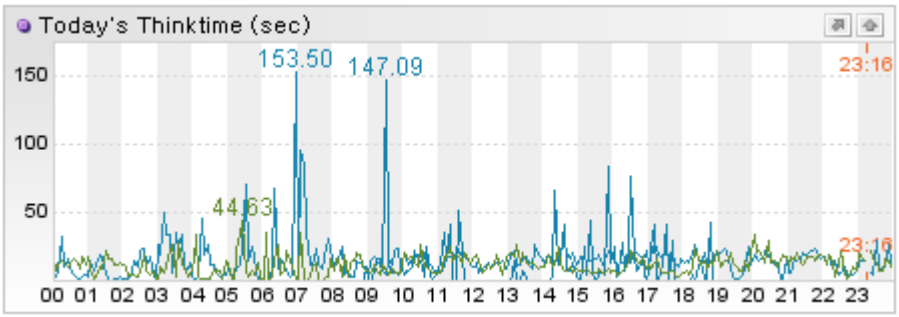
대기 시간은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 30초 평균 값을 런타임 라인 차트를 통해서 확인할 수 있다.

그림 6-16: 최근 대기 시간



또한 대기 시간에 대한 5분 평균 값은 PERF_X_01~31 테이블의 THINKTIME 칼럼에 저장되고, 특정 날짜의 대기 시간 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 6-17: 금일 대기 시간



6.10.3. 액티브 사용자 수

이론적으로는 HTML 프레임이나 AJAX 등의 개발 기법과 탭 브라우징 등의 웹 브라우저 인터페이스의 변화로 2개 이상의 액티브 서비스가 동일한 사용자와 연결될 수도 있다.

즉, 액티브 서비스 중에 wmonid 쿠키 값이 동일한 트랜잭션이 있을 수 있다. 액티브 사용자 수는 액티브 서비스 중에서 wmonid 쿠키 값이 동일한 트랜잭션들을 하나로 간주하여 계산한 것이다.

액티브 사용자 수에 대한 5분 평균 값은 PERF_X_01~31 테이블의 ACTIVE_USER 칼럼에 저장된다.

Notice: 액티브 사용자 수는 차트로 제공하지 않는다.

6.10.4. 방문자 수

방문자 수는 wmonid 쿠키 값의 존재 여부와 시간 구간을 기준으로 수집한다. 시간 구간은 날짜와 시간으로 구분된다. 날짜를 기준으로 수집한 것은 일일 방문자 수이고 시간을 기준으로 수집한 것은 시간당 방문자 수이다.

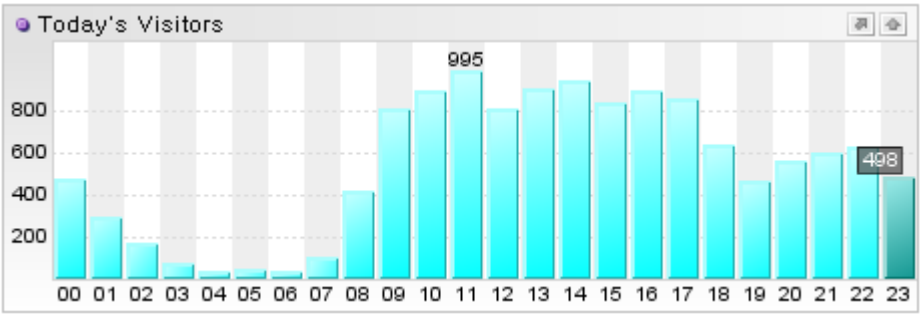
예를 들어, 10월 10일 01시 25분에 wmonid 쿠키가 없는 트랜잭션이 들어오면 일일 방문자 수가 1이 증가하고, 01시에 해당하는 시간당 방문자 수도 1이 증가한다.

그리고 10월 10일 02시 03분에 wmonid 쿠키가 있는 트랜잭션이 들오여면 일일 방문자 수는 증가하지 않지만, 02시에 해당하는 시간당 방문자 수는 1이 증가한다. 즉, 시간이 변경되면 wmonid 쿠키의 존재 여부와 상관없이 시간당 방문자 수가 증가한다.

그런데 10월 09일에 이미 방문한 사용자에 의해서 10월 10일 01시 01분에 트랜잭션이 들어오면 이 트랜잭션에는 wmonid 쿠키가 존재한다. 그럼에도 10월 10일 일일 방문자 수는 1이 증가한다. 즉, 날짜가 변경되면 wmonid 쿠키의 존재 여부와 상관없이 일일 방문자 수가 증가한다.

방문자 수는 일반 성능 데이터로 5분 동안 증가한 시간당 방문자 수는 PERF_X_01~31 테이블의 VISIT_HOUR 칼럼에 저장되고, 5분 동안 증가한 일일 방문자 수는 PERF_X_01~31 테이블의 VISIT_DAY 칼럼에 저장된다. 특정 날짜의 방문자 수 변화 추이는 막대 차트를 통해서 확인할 수 있다.

그림 6-18: 금일 시간당 방문자 수



시간을 고려해서 방문자 수를 계산할 때 주의할 것은 제니퍼 서버의 재시작이다. 제니퍼 서버는 방문자 수 계산을 위해서 모든 wmonid 쿠키 값을 자바 힙 메모리에 상주시킨다. 그런데 제니퍼 서버를 재시작하면 자바 힙 메모리에 상주되어 있던 wmonid 쿠키 값이 모두 사라지기 때문에 방문자 수가 정확하지 않을 수 있다.

이 문제를 해결하기 위해서 제니퍼 서버를 재시작할 때, 제니퍼 서버 재시작 전 까지 수집한 방문자 수를 제니퍼 서버를 재시작한 후에 새로 수집한 방문자 수에 합산할지 여부를 결정할 수 있다. 제니퍼 에이전트의 enable_visit_user_added_while_reloading 옵션을 true로 설정하면 합산을 하고, false로 설정하면 합산하지 않는다.

```
enable_visit_user_added_while_reloading = true
```

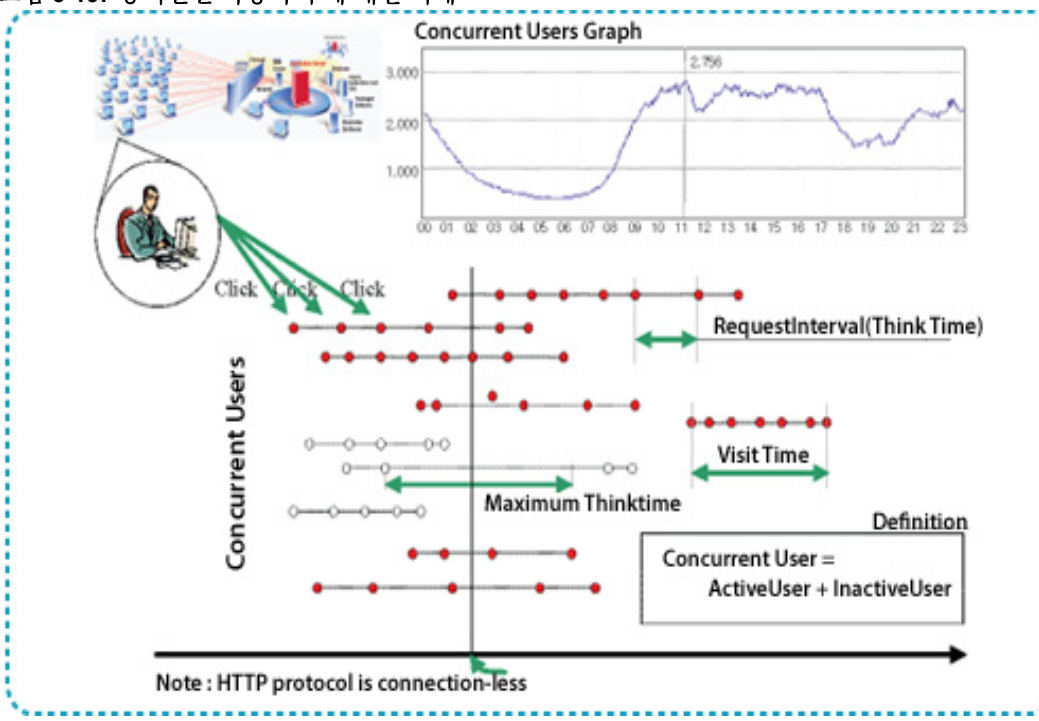
Notice: 일반적으로 쇼핑몰처럼 매번 사용자가 바뀌는 시스템은 true로 설정하고, 회사 내부 시스템처럼 사용자가 고정된 경우에는 false로 설정하는 것이 일반적인다. 기본값은 false이다.

6.10.5.동시단말 사용자 수

6.10.5.1. 동시단말 사용자 수에 대한 이해

다음 그림은 한 사용자가 웹 애플리케이션에 접속하여 최초로 클릭을 한 후, 그 다음 페이지로 이동하면서 클릭을 반복하고 이후 최종적으로 웹 애플리케이션을 떠나는 모습을 간략하게 형상화 한 것이다. 다른 사용자들 또한 서로 다른 시간에 해당 웹 애플리케이션에 방문하여 이러한 모습으로 서비스를 사용할 것이다.

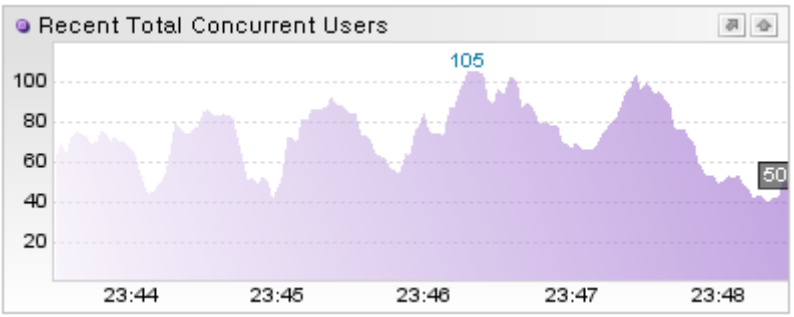
그림 6-19: 동시단말 사용자 수에 대한 이해



이때 클릭과 클릭 사이의 시간을 대기 시간으로 정의하고, 최초로 방문한 시간부터 마지막 클릭 까지의 시간을 방문 시간으로 정의한다. 이 상황에서 임의로 특정 시간대를 선으로 그어 해당 시각에서의 접속자 혹은 동시단말 사용자 수를 확인해 보면 6명(해당 그림에서)이 사용하고 있음을 확인할 수 있다. 즉, 동시단말 사용자 수란 특정 시점에 웹 애플리케이션에 접속하여 사용하고 있는 모든 사용자의 수를 의미한다.

동시단말 사용자 수는 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 30초 평균값을 런타임 라인 차트를 통해서 확인할 수 있다.

그림 6-20: 최근 동시단말 사용자 수



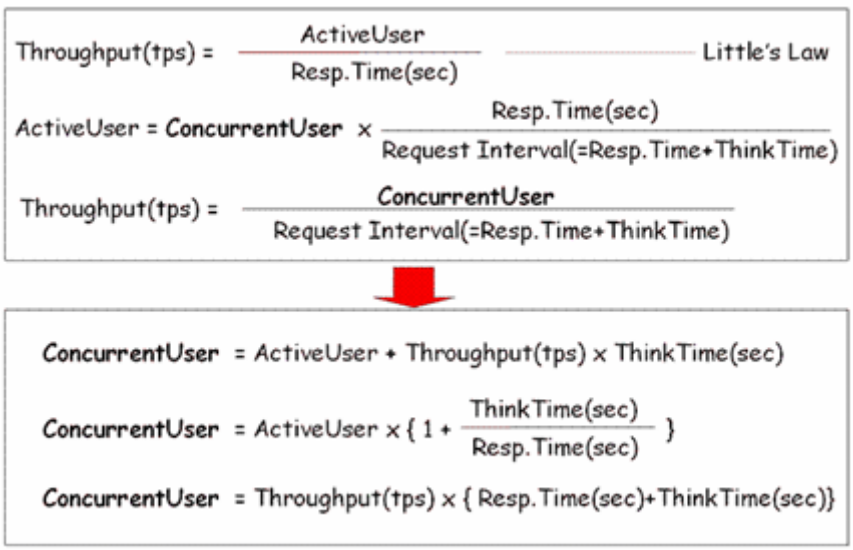
또한 동시단말 사용자 수에 대한 5분 평균 값은 PERF_X_01~31 테이블의 CONCURRENT_USER 칼럼에 저장되고, 특정 날짜의 동시단말 사용자 수 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 6-21: 금일 동시단말 사용자 수

6.10.5.2. 동시단말 사용자 수의 측정

제니퍼가 동시단말 사용자 수를 측정하는데 사용하는 알고리즘은 성능 이론에 기반을 두고 있으며, 현재 컴퓨터 앞에 앉아서 웹 애플리케이션을 사용하고 있는 사람의 수를 보다 정확하게 측정하기 위한 획기적인 방법이다. 과거 클라이언트/서버 방식의 시스템에서는 단순히 TCP/IP 연결 개수를 동시단말 사용자 수로 간주할 수 있었지만, HTTP 프로토콜은 사용자 컴퓨터와 서버 사이의 연결을 지속시키지 않고 사용자의 요청을 직접적으로 처리할 때만 연결을 맺는 방식이기 때문에 TCP/IP 연결 개수를 동시단말 사용자 수로 간주할 수는 없다. 그래서 제니퍼는 성능 이론을 토대로 하는 다음의 동시단말 사용자 수 측정 알고리즘으로 동시단말 사용자 수를 측정한다.

그림 6-22: 동시단말 사용자 수 측정 알고리즘



웹 애플리케이션의 부하량(Throughput)이 높을수록 정확한 수치에 수렴해 간다는 것이 수학적으로 증명되었고, 수십회의 웹 성능 테스트를 통한 시뮬레이션과 실제 운영 시스템

에 대한 모니터링 결과 이 알고리즘으로 측정한 동시단말 사용자 수 수치의 정확성은 검증되었다.

예를 들어, 큰 홀에 모인 500 명의 사용자가 동일한 시스템에 접속하여 사용하면 제니퍼가 측정한 시스템의 동시단말 사용자 수나 웹 애플리케이션 서버의 javax.servlet.http.HttpSession 객체의 개수는 모두 500이 된다. 그런데 500 명이 점식 식사를 위해서 모두 자리를 일어나 큰 홀을 떠나면 제니퍼가 측정한 동시단말 사용자 수는 곧바로 0이 되지만 웹 애플리케이션 서버의 javax.servlet.http.HttpSession 객체의 개수는 세션 타임아웃 기간 동안은 계속 500이 된다.

Notice: 간혹 동시단말 사용자 수가 급격하게 커지는 현상이 나타날 때가 있다. 이러한 현상은 TPS와 응답 시간, 대기 시간을 이용한 계산에 의한 문제로 기술적으로 발생할 수 있다. 이러한 경우에는 해당 수치를 무시하고 정상 상황에서의 값들을 기준으로 분석을 해야 한다.

6.10.5.3. 동시단말 사용자 수의 활용

동시단말 사용자 수는 성능적으로 매우 중요하지만, 기술적인 문제로 장애가 발생해서 동시단말 사용자 수가 증가한건지 동시단말 사용자 수가 증가해서 장애가 발생했는지에 대한 인과 관계를 파악하기 어려울 수 있다. 따라서 동시단말 사용자 수를 직접적인 장애 원인 분석 수단으로 활용하는 것은 권장하지 않는다.

그러나 웹 애플리케이션의 성능 테스트 단계에서 동시단말 사용자 수는 성능 테스트 환경과 운영 환경의 성능 지표간의 연관도를 파악하는데 중요한 역할을 한다.

즉, 가상 사용자를 지정하고 업무별 워크로드를 설계하여 부하 테스트를 하게 되는데, 이때 가상 사용자는 제니퍼의 동시 단말 사용자 수와 직접적으로 연관된다. 따라서 이것을 기준으로 부하 테스트의 워크로드가 실제 운영 환경의 워크로드과 유사한지 등을 파악할 수 있다.

6.10.6.전체 성능 데이터와 제니퍼 에이전트 그룹

하나의 제니퍼 서버가 모니터링하는 제니퍼 에이전트의 숫자가 많은 경우에 전체 방문자 수 혹은 전체 동시단말 사용자 수를 계산하는 방법을 설명한다.

예를 들어, 09시에서 10시 까지의 W11 제니퍼 에이전트의 시간당 방문자 수는 120명이고, W12 제니퍼 에이전트의 방문자 수는 140명이고, W13 제니퍼 에이전트의 방문자 수는 70명이라고 가정하자.

전체 방문자 수를 계산하는 가장 손쉬운 방법은 모든 제니퍼 에이전트의 방문자 수를 합하는 것이다. 따라서 이 경우에 전체 방문자 수는 330명이 된다.

그런데 동일한 사용자가 09시에서 10시 까지 W11, W12, W13 제니퍼 에이전트를 모두 방문했다면 전체 방문자 수가 중복 계산되어 실제 수치보다 클 수 있다. 이 문제를 해결하기

위해서 제니퍼 서버는 wmonid 쿠키 값을 활용해서 동일 사용자가 중복 계산되는 것을 방지한다.

그런데 경우에 따라서는 전체 제니퍼 에이전트가 아닌 특정 제니퍼 에이전트 그룹에 대한 전체 방문자 수나 전체 동시단말 사용자 수를 측정해야할 필요가 있다.

예를 들어, W13 제니퍼 에이전트를 제외한 W11 제니퍼 에이전트와 W12 제니퍼 에이전트에 대한 전체 방문자 수를 계산하려면 제니퍼 서버의 agent_group 옵션을 사용한다.

```
agent_group = 그룹아이디:에이전트목록
```

제니퍼 에이전트 그룹 아이디는 반드시 '@'로 시작하고 2자리의 숫자로 끝나야 한다. 따라서 제니퍼 에이전트 그룹 아이디로는 @01에서 @99 까지 사용할 수 있다.

그리고 제니퍼 에이전트 그룹에 속하는 제니퍼 에이전트 목록은 제니퍼 에이전트 아이디를 [,]를 구분자로 구분해서 설정한다. 따라서 앞의 예제는 다음과 같이 설정한다.

```
agent_group = @01:W11,W12
```

2개 이상의 제니퍼 에이전트 그룹은 [;]을 구분자로 구분한다.

```
agent_group = @01:W11,W12; @02:W13,W14
```

제니퍼 에이전트 그룹 아이디는 제니퍼 에이전트 아이디와 동일한 역할을 한다. 따라서 제니퍼 에이전트 그룹 아이디별로 PERF_X_01~31 테이블에 모든 성능 데이터가 저장된다.

Notice: 동일한 제니퍼 에이전트를 2개 이상의 제니퍼 에이전트 그룹에 설정할 수는 없다.

6.11. Batch JOB Monitoring

6.11.1.Batch JOB의 특징

Batch job 이란 실행되여 지정한 로직을 수행한후 종료되는 프로그램이다. Batch JOB 프로그램은 다음과 같은 특징이있다.

- 실행중에 사용자로 부터 요청을 받지 않는다.
- 지정한 시점에 시작되어 로직을 모두 수행하면 종료된다
- 하나의 시스템에서는 여러개의 Batch JOB이 선후관계를 가지고 실행될 수 있다.

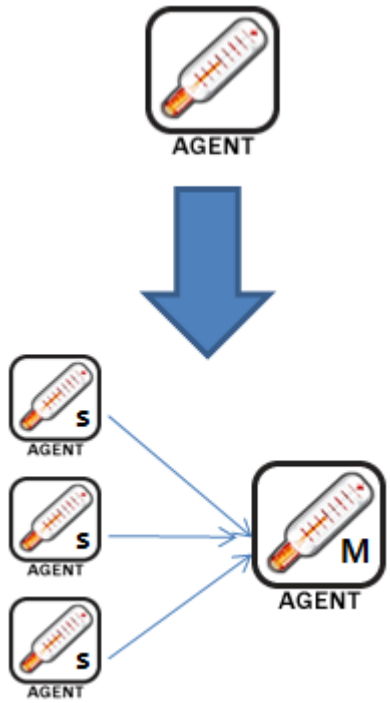
- 프로세스 내부에서 병행처리 로직을 가질 수 있다.

Batch Job 모니터링은 이러한 특징을 갖는 프로그램의 성능을 모니터링 하는 것을 말한다 .

6.11.2. 제니퍼의 Batch JOB 모니터링 기본 개념

배치잡은 짧은 시간동안 실행되고 정지 된다. 따라서 배치잡 하나를 온라인에서 서비스 형태로 모니터링한다.

그림 6-23: 제니퍼의 Batch JOB 모니터링 기본 개념



에이전트를 분리한다.

제니퍼 관점에서는 서비스는 에이전트 아이디를 가진 프로세스에 의해 수행된다. 따라서 별도의 가상의 마스터 에이전트를 두고 각각의 배치잡에서 직접 정보를 수집하기 위한 서브에이전트를 둔다.

UDP 중심 정보 수집

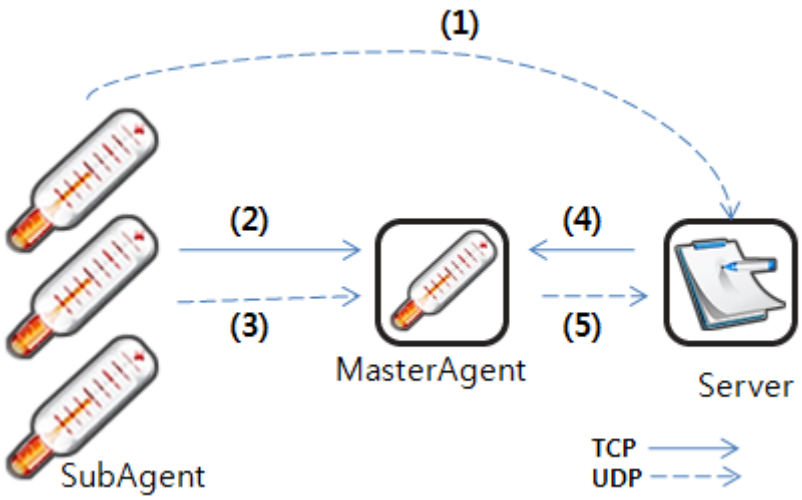
배치잡은 ?은 시간에 종료된다 따라서 TCP 세션을 유지하는 것이 큰잇점이 없다. 따라서 대부분의 정보를 UDP통해서 수집한다.

액티브스택을 위해 TCP세션을 사용한다.

주기적인 통계정보는 UDP통해서 수집하지만 현재 진행중인 배치잡의 액티브 스택이나 액티브 프로파일을 조회하기 위해선 여전히 TCP세션이 필요하다.

6.11.3.Batch JOB 모니터링을 위한 제니퍼 아키텍처

그림 6-24: Batch JOB 모니터링을 위한 제니퍼 아키텍처



1. Runtime Data(Transaction Start/End)와 프로파일 데이터가 전송된다 제니퍼서버입장에서는 어느 SubAgent인지 심지어 이데이터가 일반적인 제니퍼 에이전트에서 온것인지 구분하지 않는다.
2. SubAgent가 MasterAgent로 연결하는 TCP세션이다. 이곳으로는 MasterAgent로 부터 보내진 요청에 대한 응답만을 전송한다. 또한 이 세션이 종료되면 자동으로 다시 연결한다. 이곳에서는 액티브 서비스리스트나 각 쓰레드의 상세 액티브스택/액티브 프로파일 정보등이 전달된다.
3. 매초마다 보내는 Summary Data가 MasterAgent로 전송된다. 그리고 SubAgent에서 수집되는 텍스트 데이터(SQL, Application Name, Method Name) 등이 이곳으로 전송된다. 그리고 SubAgent는 애플리케이션 및 SQl수행 통계를 주기적으로 혹은 프로세스가 종료될때 (3)을 통해 전송한다.
4. 제니퍼 서버는 MasterAgent를 일반적인 제니퍼 Agent로 간주하고 통신한다. 특히 SubAgent가 (3)을 통해 전달한 통계 정보를 모아 두었다가 10분에 한번씩 (4)를 통해 제니퍼 서버로 전송한다.
5. MasterAgent는 (3)을 통해 SubAgent가 보내주는 SummaryData들 하나로 통합하여 매 초마다 제니퍼서버로 전송한다. MasterAgent에서는 직접적으로 서비스가 수행되지 않기 때문에 서비스 시작/종료나 프로파일 정보는 발생하지 않는다.

중요 특징은 데이터 수집을 위해 UDP를 사용하고 제어를 위해 TCP를 사용한다. TCP Listening은 마스터에이전트가 하고 서브에이전트는 실행과 동시에 하나의 TCP세션을 마스터 에이전트에 연결한다.

6.11.4.SubAgent설치

SubAgent를 설치할때는 일반적인 제니퍼 에이전트를 설치하는 방법과 동일하다. 다만 현재의 에이전트 실행모드가 SubAgent라는 것과 마스터 에이전트 그리고 제니퍼 서버의 네트웍 정보를 설정해야 한다.

- SubAgentMode라는 것을 자바 실행변수에 추가한다.

```
-Djennifer.subagent=true
```

- jennifer.conf(ex. w11.conf)에 Master에이전트 네트웍 정보를 설정한다.(아래 값들은 설정이 생략된 경우 기본값이다.

```
master_host_name = 127.0.0.1
master_udp_listen_port = 6705
```

- 기본 제니퍼 모드에서 실행될 때 설정하던 제니퍼서버에 대한 네트웍 정보는 동일하게 설정한다.
- 제니퍼 에이전트 명은 접속한 마스터에이전트에 설정된 에이전트명을 동일하게 지정해야 한다.

Notice: 따라서 SubAgent와 MasterAgent를 동일 머신에 설치하는 경우에는 설정파일은 공유하는것이 좋다.

6.11.5.MasterAgent설치

마스터 에이전트는 단순히 실행만 하면된다. 제니퍼 에이전트디렉토리에는 “masteragent.sh/bat” 파일이 포함되어있다. 이 파일을 실행한다.

```
java  -Xbootclasspath/p:./lwst.boot.jar   \
      -cp ./jennifer.jar                  \
      -Djava.library.path=.               \
      -Djennifer.config=w11.conf          \
      com.javaservice.jennifer.agent.master.MasterAgent
```

마스터 에이전트는 서브에이전트로 부터 UDP데이터를 수신한다. 따라서 리스닝 포트를 지정해야 한다. jennifer.conf 설정에 다음 설정을 추가하다 만약 지정하지 않으면 6705가 기본값이다.

```
master_udp_listen_port = 6705
```

제니퍼 에이전트 디렉토리로 이동하여 다음 파일을 실행한다.

```
masteragent.bat
```

6.11.5.1. 프로파일 설정

배치잡은 많은 수의 SQL이 실행될 가능성이 있다 만약 프로파일 데이터가 너무 많다면 중요도가 떨어지는 항목부터 제외해야 한다. 이것을 위해 몇가지 옵션들이 추가되었다.

```
profile_ignore_connection=false
profile_ignore_fetch=false
profile_sql_time=0
```

위 설정 예는 기본값이다

- profile_ignore_connection이 true설정되면 DB connection의 open과 close가 프로파일 정보에서 빠진다
- profile_ignore_fetch이 true로 설정되면 FETCH라는 프로파일 항목이 제외된다.
- profile_sql_time=10이라고 설정되면 10ms 미만의 속도로 수행된 SQL문은 프로파일 정보에서 제외된다.

6.11.6.마스터와 서브 에이전트 사이의 텍스트 케쉬제어

서브에이전트는 UDP를 통해서 전송된다. 이때 이미 전송된 목록을 기록해 두는 케쉬가 있는데 크기를 수정할 수 있다.

```
master_max_num_of_text = 1000
master_max_num_of_sql = 5000
```

APP, TX, ERR의 케쉬는 master_max_num_of_text를 따르고 SQL 수는 master_max_num_of_sql에서 설정된다.

서브 에이전트는 택스트 자체는 보관하지 않고 보냈다는 기록만을 보관하기 때문에 이값을 좀더 크게 설정해도 된다. 이값이 충분히 크면 텍스트는 서브에서 마스터로 단 한번만 전송되게 된다. 그러나 서브에이전트가 재기동되면 처음부터 다시 전송한다.

유사하게 마스터 에이전트는 서브에이전트로 부터 받은 텍스트를 보관하는 별도의 케쉬를 가지고 있다. 이 값또한 master_max_num_of_text, master_max_num_of_sql에 의해 크기가 정해진다.

6.12. 닷넷 배치 프로세스 모니터링

이 장에서는 닷넷 버전의 제니퍼 에이전트를 이용하여 배치(Batch) 작업을 모니터링하는 방법을 설명한다.

Notice: 제니퍼에서 구현되는 배치 모니터링의 개념 및 자바에서의 설치방법은 ” Batch JOB Monitoring” 문서를 참고한다.

6.12.1.MasterAgent 설치 및 실행

1. [conf 파일 설정]

” [에이전트 설치 폴더]\conf\app_pool.conf” 파일을 복사해서 “batchjob_master.conf” 파일을 생성한다. 제니퍼 서버와 연결되어야하므로, batchjob_master.conf 파일을 메모장에서 열고, 제니퍼 서버 주소를 설정한다.

```
udp_server_host = [제니퍼 서버 주소]
```

2. [마스터 에이전트 실행]

마스터 에이전트 실행파일은 JENNIFER와 함께 설치되므로 별도의 설치작업은 필요없고 단순히 아래의 경로에 제공되는 모듈을 실행하면 된다.

```
.NET 1.1: [에이전트 설치 폴더]\bin\MasterAgent.Clr10.exe
.NET 2.0 이상: [에이전트 설치 폴더]\bin\MasterAgent.exe
```

6.12.2.SubAgent 설치 및 실행

서브 에이전트는 모니터링 해야 할 대상을 의미하는데, 제니퍼 에이전트가 활성화 되는 배치 프로세스(exe)가 이에 해당한다.

1. [conf 파일 설정]

서브 에이전트가 종속될 마스터 에이전트의 conf 파일을 복사해서 새롭게 conf 파일을 생성하고 서브 에이전트 임을 명시하기 위해 다음의 설정을 추가한다.

```
[예: youragent.conf]

SUB_AGENT = true
```

2. [배치 프로세스에 conf 파일 연결]

모니터링하려는 배치 프로세스의 실행 파일 경로가 다음과 같다고 가정할 때,

```
C:\BatchJobs\DailyWorker.exe
```

해당 프로세스 명에 " .config" 파일을 연결하여 conf 폴더에 새롭게 파일을 생성한다.

```
[에이전트 설치 폴더]\conf\DailyWorker.exe.config
```

파일의 내용은 아래와 같이 이전에 서브 에이전트 용으로 만들어 둔 conf (예: youragent.conf) 를 지정한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <appSettings>
        <add key="Jennifer.FileName" value="youragent.conf" />
    </appSettings>
</configuration>
```

3. [사용자 정의 메서드 프로파일링 설정]

일반적으로 배치 프로세스에서는 사용자가 정의한 메서드를 트랜잭션 단위로 여기게 된다. 따라서, " 닷넷 메서드 프로파일링 "에서 설명한 방법에 따라 " [txserver]" 절을 별도로 구성해야 한다.

또한, 배치 프로세스의 경우 반드시 " [batchjob]" 절을 정의하고 해당 응용 프로그램의 Main 함수를 정의해 줘야 한다.

예를 들어 사용자 코드가 다음과 같이 정의된 경우,

```
namespace BatchJob1
{
   class Program
   {
      static void Main(string[] args)
      {
      }

      void DoBatchJob()
      {
      }
   }
}
```

만약, DoBatchJob 이 실제 배치 작업을 처리하는 메서드이고 txserver 로 모니터링해야 할 대상이라면 profiler.ini 파일에는 다음과 같이 설정해 주어야 한다.

```
[batchjob.프로세스명]
BatchJob1.Program.Main=1


[txserver.프로세스명]
BatchJob1.Program.DoBatchJob=1
```

4. [제니퍼 에이전트 환경변수 설정]

배치 프로세스가 프로파일링되기 위해서는 다음의 환경 변수를 프로세스 실행 전에 설정한다.

```
SET COR_ENABLE_PROFILING=1
SET COR_PROFILER={FF8C2B6C-DBB5-4AED-9876-2CED6FFAF4C9}
```

환경 변수는 여러가지 방법으로 설정이 가능한데, .bat 파일 또는 NT 서비스의 경우에는 레지스트리를 이용하는 방법은 ” 환경 변수 설정 “을 참고한다.

만약, 그 2가지 방법으로 설정할 수 없는 경우라면 전역적으로 환경 변수를 설정하는 것도 가능하다. 시스템 환경 변수로 등록되는 경우, 해당 운영체제에서 실행되는 모든 닷넷 프로세스는 제니퍼 에이전트의 프로파일링 작업이 이뤄지기 때문에 원치 않는 성능 저하를 야기할 수 있다. 이런 부작용을 해결하기 위해서 제니퍼 닷넷은
“JENNIFER_PROFILE” 이라는 별도의 환경 변수 옵션을 인식한다. 따라서, 사용자의

배치 프로세스 명이 “DailyWorker.exe”인 경우 다음과 같이 환경 변수를 등록해 줄 수 있다.

```
환경 변수 이름: JENNIFER_PROFILE
값: dailyworker.exe
```

Notice: 프로세스 명은 반드시 소문자로 지정해야 한다.

만약, 배치 프로세스가 여러 개인 경우 세미콜론(;)을 구분자로 해서 다음과 같이 지정하는 것이 가능하다.

```
환경 변수 이름: JENNIFER_PROFILE
값: dailyworker.exe;weeklyworker.exe;monthlyworker.exe
```

설정을 마치고, 해당 배치 프로세스를 실행하면 정상적으로 모니터링이 이뤄진다.

6.12.3.에이전트 모니터링 해제 / 제거

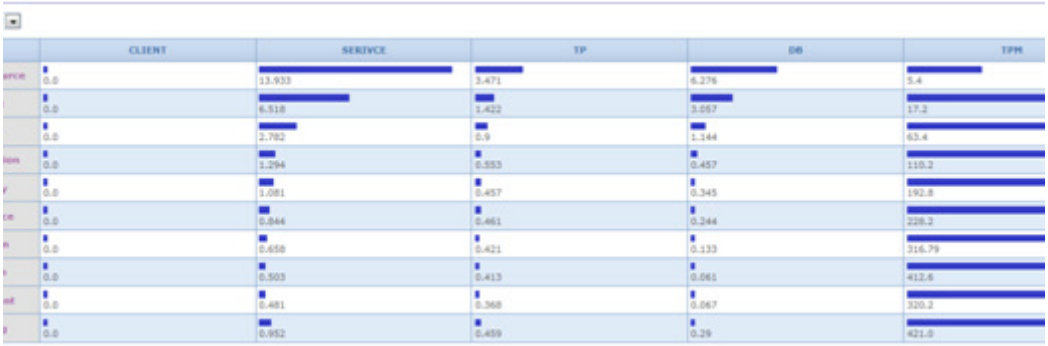
제니퍼 에이전트가 설치된 상태에서, 배치 프로세스에 대한 모니터링만을 해제하고 싶다면 환경 변수에서 “COR_ENABLE_PROFILING”값을 “0”으로 변경하면 된다. (또는 “JENNIFER_PROFILE”로 설정된 경우라면, 프로세스 명을 빼거나 JENNIFER_PROFILE 환경 변수 자체를 지우면 된다.)

만약, 제니퍼 에이전트 까지 모두 제거하고 싶다면 현재 모니터링 중인 배치 프로세스를 모두 종료하고 “[에이전트 설치 폴더]/uninstall_Jennifer.bat”을 관리자 권한으로 실행한 후, 설치 폴더를 수동으로 삭제해 준다.

6.13. 업무단위 모니터링

업무단위 모니터링은 JENNIFER® 4.5에서 강화된 기능이다 사용자는 업무단위(그룹)로 성능을 모니터링 할수 있다.

그림 6-1: 업무단위 모니터링



업무단위 모니터링의 최근 5분동안의 성능을 표현한다. 업무단위 성능에는 최종사용자 응답시간, 평균 서비스 응답시간, 외부트랜잭션 처리시간 SQL수행시간 및 패치시간, 초당 처리건수등이 포함된다.

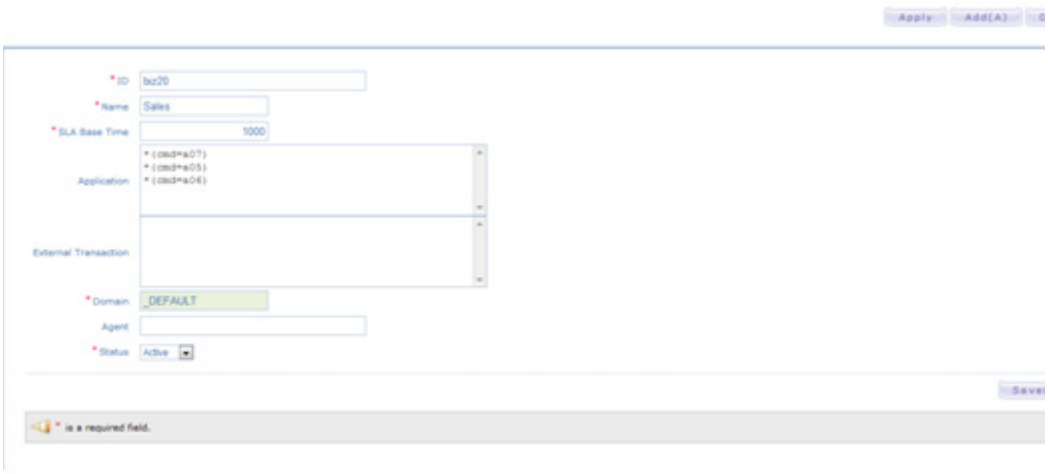
또한 업무단위 모니터링 화면 세부 업무그룹 이름을 클릭하면 각 서비스 처리별 성능을 X-View를 통해 확인할 수 있다.

그림 6-2: 업무단위 모니터링 화면 세부 업무그룹

6.13.1. 업무 단위 설정

업무 단위 모니터링을 [Properties | Business Group]에서 설정할 수 있다. 업무ID를 클릭하여 내용을 수정하거나 [Add] 버튼을 클릭하여 새로운 그룹을 추가할 수 있다.

그림 6-3: 업무 단위 설정



각 업무그룹은 유일한 "ID"를 사용해야 한다. 새 업무그룹에는 반드시 ID와 Name을 지정하고 SLA기준값을 설정한다. 해당 그룹의 서비스중에 SLA기준값을 초과하는 서비스가 발생하면 sla_fail로 카운팅된다. 수정된 그룹은 Status를 Active로 설정해야 적용된다.

애플리케이션에는 서비스명칭을 Prefix혹은 Postfix등을 이용하여 설정할 수 있다. 예를들어 *(cmd=a07)라고 설정하면 (cmd=a07)로 끝나는 서비스를 그룹으로 지정하는 것이다.

마지막으로 수정된 업무그룹이 적용되려면 [APPLY] 버튼을 클릭해야 한다.

7

차트와 대시보드

사용자는 성능 데이터를 다양한 차트로 직관적으로 모니터링할 수 있다. 그래서 제니퍼는 다양한 성능 데이터를 통합적으로 모니터링할 수 있는 대시보드를 제공하고, 사용자가 직접 대시보드를 구성할 수 있도록 사용자 정의 대시보드 기능을 제공한다.

7.1. 차트

7.1.1. 차트 일반

대부분의 차트는 다음과 같이 구성된다.

그림 7-1: 차트

1. 차트 이름 - 차트 이름은 해당 차트가 표시하는 성능 데이터의 종류와 성격을 설명해 준다. 런타임 라인 차트의 이름은 [최근]으로 시작하고, 이퀄라이저 차트의 이름은 [실시간]으로 시작한다. 그리고 막대 차트나 라인 차트가 오늘 날짜의 성능 데이터를 표시하면 이름이 [금일]로 시작한다.
2. 관련 내용 보기 버튼 - 이 버튼을 클릭하면 사용자는 해당 차트와 관련된 내용을 새로운 팝업 창에서 확인할 수 있다.
3. 새로운 창으로 보기 버튼 - 이 버튼을 클릭하면 해당 차트를 새로운 팝업 창에서 볼 수 있다. 사용자는 이 창을 통해서 차트의 크기를 조절할 수 있다.

Notice: 차트를 클릭한 상태에서 CTRL + C를 누르면 해당 차트 이미지가 클립 보드에 복사된다.

차트 이름 폰트 크기는 제니퍼 서버의 ui_chart_text_size 옵션으로 설정한다. 기본 값은 11이다. 이 옵션은 차트 이름뿐만 아니라 경보 차트 경보 메시지와 같이 텍스트 문자에 모두 적용된다.

```
ui_chart_text_size = 12
```

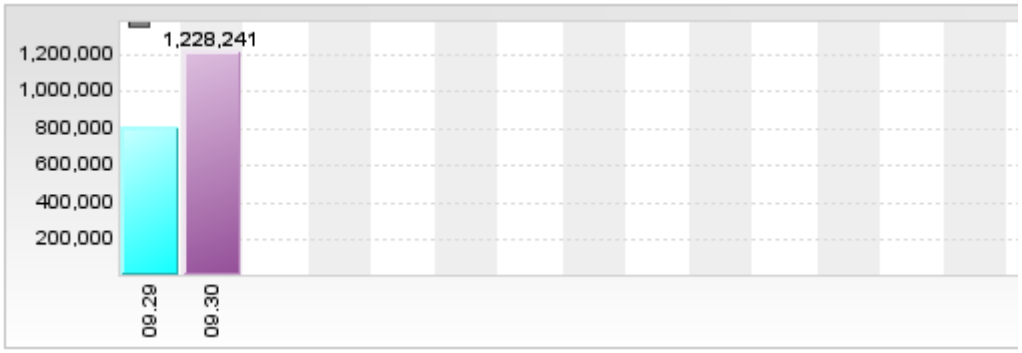
X축과 Y축 숫자 크기는 제니퍼 서버의 ui_chart_number_size 옵션으로 설정한다. 기본 값은 10이다.

```
ui_chart_number_size = 11
```

7.1.2. 막대 차트

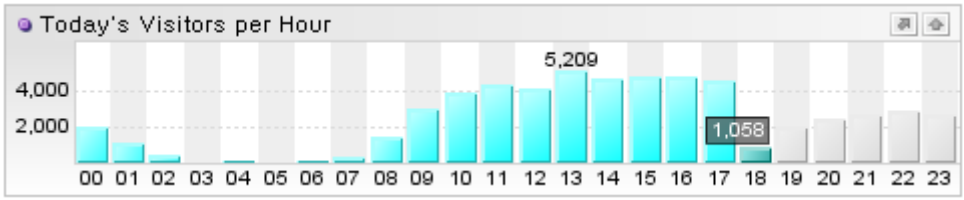
막대 차트는 시간 변화에 따른 정량적인 누적 값을 보여준다. 예를 들어, 다음 그림에서 막대 차트는 일자별 호출 건수를 표시한다. 이 경우에 X 축은 날짜가 된다.

그림 7-2: 일자별 호출 건수



또한 시간당 방문자 수와 같이 시간당 값을 표시할 때도 막대 차트를 사용한다. 이 경우에 X 축은 시간이 된다.

그림 7-3: 금일 시간당 방문자 수



Notice: 일반적으로 막대 차트는 주로 호출 건수와 방문자 수를 보여주는데 사용된다.

막대 차트에서 최대 값은 항상 해당 막대 위에 숫자로 표시된다. 그리고 막대 차트에서 특정 막대를 클릭하면 해당 막대가 표시하는 값이 해당 막대 위에 표시된다.

그리고 금일 시간당 방문자 수와 같이 차트의 이름이 금일로 시작하면 오늘 수집한 성능 데이터를 표시하고 있음을 의미한다. 이 경우에 현재 시간에 해당하는 막대 위에 현재 값이 박스 안에서 표시된다. 그리고 현재 시간에 해당하는 막대의 오른쪽에 있는 막대들은 어제의 성능 데이터를 표시한다. 반면에 현재 시간에 해당하는 막대의 왼쪽에 있는 막대들은 오늘의 성능 데이터를 표시한다. 구분을 위해서 오늘과 어제를 표시하는 막대들의 색상은 다르다.

막대 차트에서 오른쪽 마우스를 클릭하면 나타나는 컨텍스트 메뉴로 최대 값과 현재 값을 숨길 수 있다.

7.1.3. 라인 차트

라인 차트는 주로 하루 동안의 5분 평균 데이터의 변경 추이를 선으로 보여준다. 하루 24 시간을 5분으로 나누면 288이다. 따라서 라인 차트는 288개의 점을 선으로 연결한 것이다 .

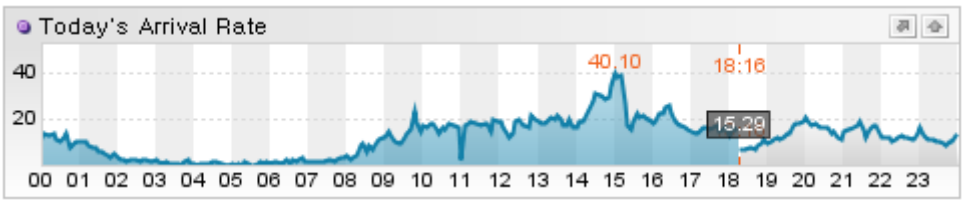
Notice: 하루 동안의 특정 애플리케이션의 평균 응답 시간은 10 분 주기로 수집된다. 따라서 이를 라인 차트로 보여주는 경우에는 144개의 점을 선으로 연결한 것이다. 애플리케이션, SQL, 외부 트랜잭션 등과 같이 10 분 주기로 성능 데이터를 수집하는 경우가 모두 이에 해당한다.

예를 들어, 다음 그림에서 라인 차트는 오늘 하루 동안의 서비스 요청률의 변경 추이를 선으로 보여준다.

그림 7-4: 금일 서비스 요청률

Notice: 에어리어 차트가 표시하는 것은 라인 차트와 동일하다. 단지, 선 아래 영역에 배경 색을 표시한 것에만 차이가 있다.

그림 7-5: 금일 서비스 요청률 - 에어리어 차트

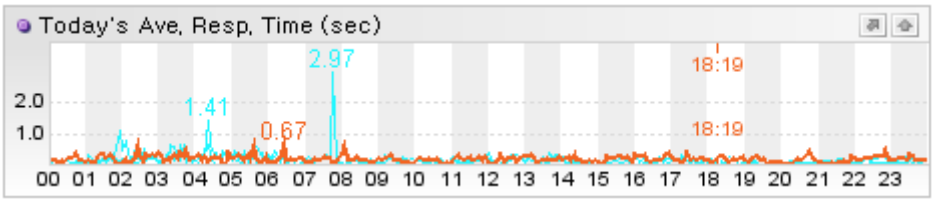


라인 차트에서 최대 값은 항상 좌표 위에 숫자로 표시된다.

그리고 금일 서비스 요청률과 같이 차트의 이름이 금일로 시작하면 오늘 수집한 성능 데이터를 표시하고 있음을 의미한다. 이 경우에 현재 시간에 해당하는 좌표 위에 현재 값이 박스 안에서 표시된다. 그리고 현재 시간에 해당하는 좌표의 오른쪽에 있는 라인은 어제의 성능 데이터를 표시한다. 반면에 현재 시간에 해당하는 좌표의 왼쪽에 있는 라인은 오늘의 성능 데이터를 표시한다. 구분을 위해서 오늘과 어제를 표시하는 라인의 색상은 다르게 표시된다.

에어리어 차트에서 오른쪽 마우스를 클릭하면 나타나는 컨텍스트 메뉴로 최대 값과 현재 값을 숨길 수 있다. 그런데 라인 차트에 여러 개의 선이 표시될 수 있다. 이 경우는 주로 제니퍼 에이전트 별 성능 데이터를 별도의 선으로 나타내는 것이다. 특정 선을 선택하면 해당 선만 강조된다.

그림 7-6: 금일 서버 별 평균 응답 시간



7.1.4. 런타임 라인 차트

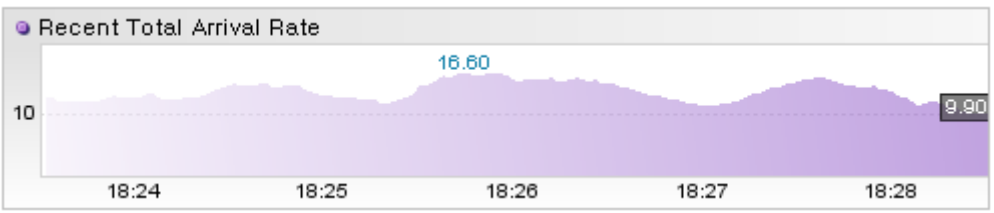
런타임 라인 차트는 최근 5분 동안의 30초 평균 데이터의 변경 추이를 선으로 보여준다.

예를 들어, 다음 그림에서 런타임 라인 차트는 최근 5분 동안의 30초 평균 서비스 요청률의 변경 추이를 선으로 보여준다.

그림 7-7: 최근 서비스 요청률

Notice: 런타임 에어리어 차트가 표시하는 것은 런타임 라인 차트와 동일하다. 단지 선 아래 영역에 배경 색을 표시한 것에만 차이가 있다.

그림 7-8: 최근 서비스 요청률 - 런타임 에어리어 차트

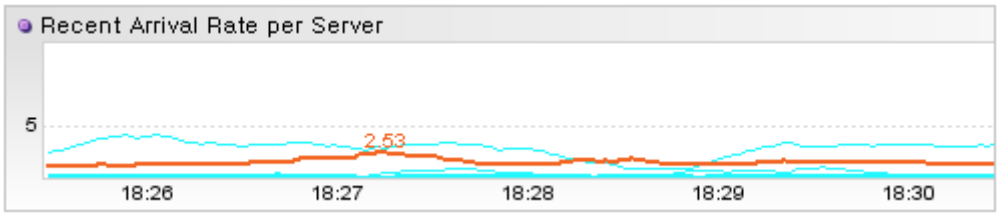


런타임 라인 차트에서 최대 값은 항상 좌표 위에 숫자로 표시된다.

런타임 에어리어 차트에서 오른쪽 마우스를 클릭하면 나타나는 컨텍스트 메뉴로 최대 값과 현재 값을 숨길 수 있다.

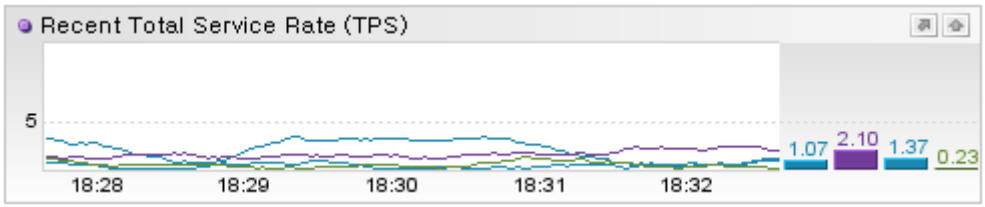
런타임 라인 차트에 여러 개의 선이 표시될 수 있다. 이 경우는 주로 제니퍼 에이전트 별 성능 데이터를 별도의 선으로 표시하는 것이다. 특정 선을 선택하면 해당 선이 강조된다.

그림 7-9: 최근 서버 별 평균 응답 시간



런타임 라인 차트에서는 현재 값을 막대로 오른쪽에 표시할 수 있다.

그림 7-10: 최근 서버 별 평균 응답 시간 - 현재 값 표시



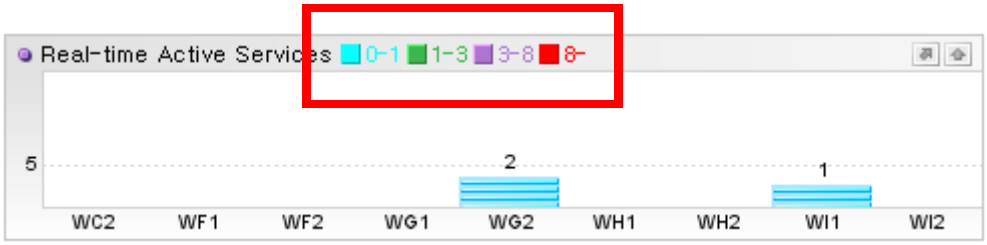
런타임 라인 차트와 런타임 에어리어 차트가 표시하는 데이터를 CSV 형식 파일로 저장할 수 있다. 차트를 클릭한 상태에서 CTRL + E를 누르면된다.

7.1.5. 이퀄라이저 차트

이퀄라이저 차트는 실시간 값을 이퀄라이저 형식으로 보여준다.

예를 들어, 다음 그림에서 이퀄라이저 차트는 실시간 액티브 서비스 개수를 보여준다.

그림 7-11: 실시간 액티브 서비스 개수



또한 이퀄라이저 차트는 데이터를 특정 기준에 맞추어 세분화해서 보여준다. 예를 들어, 특정 제니퍼 에이전트의 액티브 서비스 개수는 경과 시간 구간에 맞추어 세분화된다. 기본적으로 액티브 서비스 개수의 경과 시간 구간은 1초 미만, 1초 이상에서 3초 미만, 3초 이상에서 8초 미만, 8초 이상으로 이루어진다.

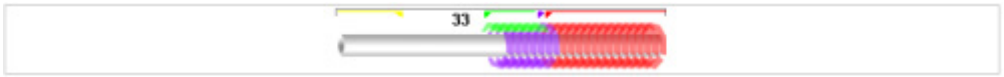
Notice: 성능 데이터가 특정 기준에 맞추어 세부적으로 분류되면 이퀄라이저 차트 이름 옆에 그 분류 기준이 표시된다.

액티브 서비스 개수는 경과 시간 구간을 기준으로 분류한다면 CPU 사용률은 CPU 사용 영역(WAIT, NICE, USER, SYS)에 따라서 분류된다. 그리고 실시간 JDBC 커넥션 개수는 JDBC 커넥션 상태(IDLE, ALLOCATED, ACTIVE)에 따라서 분류된다.

7.1.6. 스피드 바

스피드 바는 서비스 요청률, 서비스 처리율, 액티브 서비스 개수 등의 성능 데이터를 기초로 자바 애플리케이션의 부하량 상태를 직관적으로 보여준다.

그림 7-12: 스피드 바



스피드 바의 가운데에 있는 원통은 자바 애플리케이션을 의미하고, 그 원통을 감고 있는 링은 액티브 서비스 개수를 의미한다. 이퀄라이저 차트와 동일하게 경과 시간 구간으로 세분화해서 액티브 서비스 개수가 표시된다.

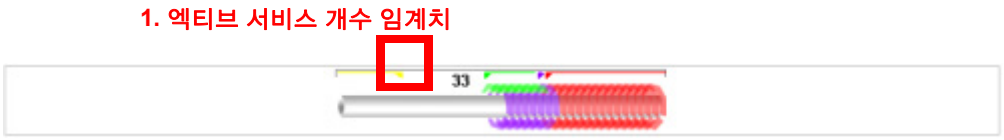
Notice: 링의 개수는 액티브 서비스 개수와 일치하지 않는다. 전체 액티브 서비스 개수에서 특정 구간의 액티브 서비스 개수가 차지하는 상대적인 비율에 해당하는 영역을 동일한 색상과 넓이의 링으로 채울뿐이다.

액티브 서비스 개수가 특정 임계치를 초과하면 경보(ERROR_SERVICE_QUEUING)가 발생하는데 임계치는 제니퍼 서버의 active_service_alert_limit 옵션으로 설정한다.

```
active_service_alert_limit = 70
```

액티브 서비스 개수가 임계치를 훨씬 초과해도 링은 스피드 바의 원통 밖으로 벗어나지는 않는다.

그림 7-13: 액티브 서비스 개수 임계치



- 액티브 서비스 개수 임계치 - 스피드 바의 액티브 서비스 개수의 임계치를 표시한다. 이 지점을 넘어서서 링이 원통을 감으면 ERROR_SERVICE_QUEUING 경보가 발생한다.

왼쪽에서 오른쪽으로 스피드 바를 지나가는 것은 트랜잭션을 의미한다. 지나가는 개별 트랜잭션의 개수는 스피드 바의 원통 왼쪽은 서비스 요청률, 스피드 바의 원통 오른쪽은 서비스 처리율을 기반으로 계산된다.

지나가는 트랜잭션의 개수에 따라서 스피드 바의 속도감이 달라지는데 자바 애플리케이션 별로 서비스 요청률에 따른 스피드 바의 속도감을 다르게 설정할 필요가 있을 수 있다. 예를 들어, A 자바 애플리케이션의 서비스 요청률 평균이 10이고 B 자바 애플리케이션의 서비스 요청률 평균이 30이라면, A 자바 애플리케이션에서는 서비스 요청률이 30일 때 스피드 바의 속도감이 높아야 하지만, B 자바 애플리케이션에서는 서비스 요청률이 30일 때 스피드 바의 속도감이 평균이어야 한다.

그래서 모니터링하는 자바 애플리케이션의 평균 서비스 요청률을 고려해서 스피드 바의 속도감을 조절해야 하는데, 이를 위해서 제니퍼 서버의 speed_threadhold 옵션을 사용한다. 기본 값은 1000이다.

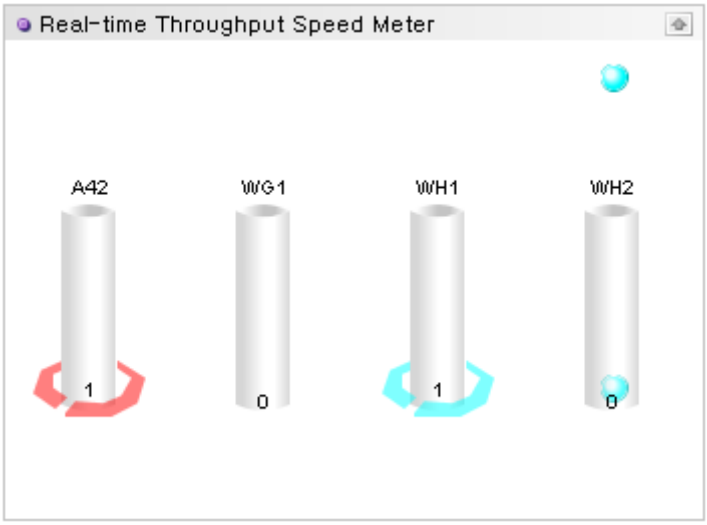
```
speed_threadhold = 1000
```

speed_threadhold 옵션을 1000보다 크게 설정하면 동일 서비스 요청률에 대해서 스피드 바의 속도감이 느려지고 1000보다 작게 설정하면 동일 서비스 요청률에 대해서 스피드 방의 속도감이 빨라진다. 이 옵션에 대한 변경 사항을 확인하려면 재로그인이 필요하다.

7.1.7. 스피드 미터

스피드 미터는 개념적으로 스피드 바와 동일한다. 단, 제니퍼 에이전트 별 부하량 상태를 동시에 표시해준다.

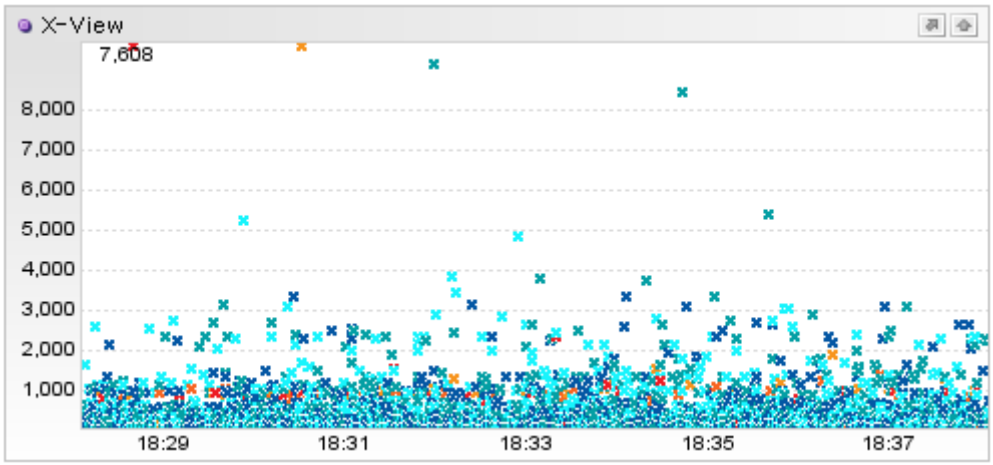
그림 7-14: 스피드 미터



7.1.8. X-View 차트

X-View 차트는 응답 시간 분포도로 모든 트랜잭션을 트랜잭션의 시작 시간과 응답 시간을 기준으로 점으로 표시한다.

그림 7-15: X-View 차트



X-View 차트에 대한 자세한 사항은 [X-View 차트] 설명서를 참조한다.

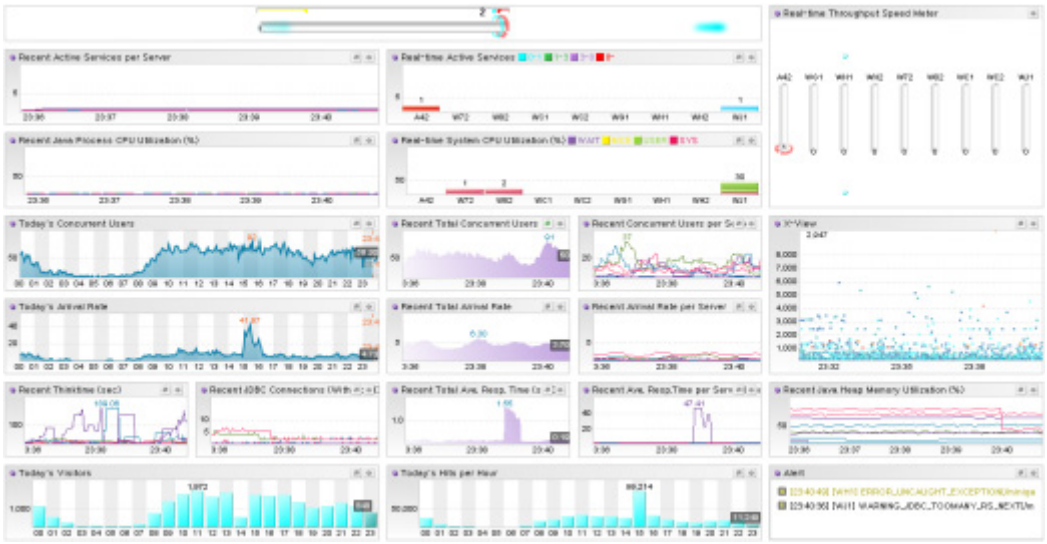
7.2. 대시보드

제니퍼에서 가장 일반적인 상황에 맞게 최적화된 모니터링 대시보드를 제공하고 있다.

7.2.1. 제니퍼 대시보드

[대시보드 | 제니퍼 대시보드] 메뉴에서 다양한 성능 데이터를 통합적으로 모니터링할 수 있다. 제니퍼 대시보드는 자바 애플리케이션을 효과적으로 모니터링하는데 초점을 맞추어 구성되어 있다.

그림 7-16: 제니퍼 대시보드



제니퍼 대시보드에서 액티브 서비스 개수와 CPU 사용률 이퀄라이저 차트의 크기를 제니퍼 에이전트의 숫자에 따라서 조정할 수 있다. 우선 제니퍼 에이전트의 숫자가 제니퍼 서버의 dashboard_equalyzer_min 옵션으로 설정한 값보다 크면 오른쪽에 있는 스피드 미터가 나타나지 않는다. 기본 값은 9이다.

```
dashboard_equalyzer_min = 9
```

그리고 제니퍼 에이전트의 숫자가 제니퍼 서버의 dashboard_equalyzer_max 옵션으로 설정한 값보다 크면 왼쪽에 있는 런타임 라인 차트가 나타나지 않는다. 기본 값은 29이다.

```
dashboard_equalyzer_max = 29
```

그림 7-17: 제니퍼 대시보드 - 제니퍼 에이전트가 10개 이상인 경우

상단 가운데에 있는 실시간 시스템 CPU 사용률 이퀄라이저 차트 대신에 WMOND로 수집한 시스템 CPU 사용률 이퀄라이저 차트를 사용할 수도 있다. WMOND로 수집한 시스템 CPU 사용률 이퀄라이저 차트를 사용하려면 제니퍼 서버의 dashboard_show_wmond 옵션을 true로 설정한다.

```
dashboard_show_wmond = true
```

그림 7-18: 제니퍼 대시보드 - WMOND로 수집한 시스템 CPU 사용률 이퀄라이저 차트의 사용

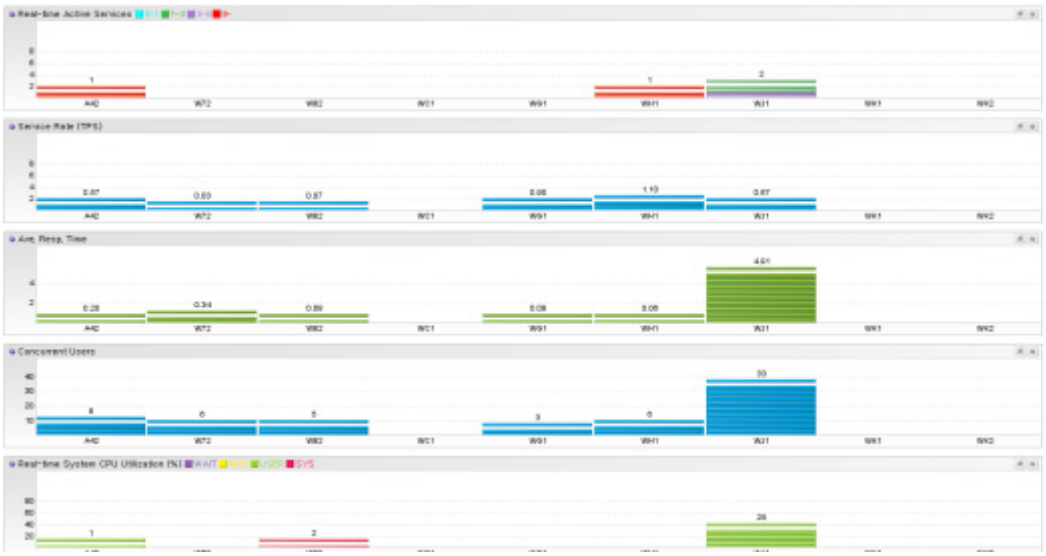
그리고 오른쪽 하단에 있는 최근 자바 힙 메모리 사용률 런타임 라인 차트 대신에 자바 힙 메모리 전체와 자바 힙 메모리 사용량을 모두 표시하는 최근 자바 힙 메모리 사용량 런타임 라인 차트를 사용할 수도 있다. 이 차트를 사용하려면 제니퍼 서버의 dashboard_show_heap_tot 옵션을 true로 설정한다.

```
dashboard_show_heap_tot = true
```

7.2.2. 이퀄라이저 대시보드

[대시보드 | 이퀄라이저] 메뉴에서 이퀄라이저 차트를 사용해 주요 성능 데이터를 모니터링할 수 있다. 이퀄라이저 대시보드는 제니퍼 에이전트 별로 다양한 성능 데이터를 실시간으로 모니터링하는데 최적화되어 있다.

그림 7-19: 이퀄라이저 대시보드



이퀄라이저 대시보드에서 제공하는 실시간 성능 데이터는 다음과 같다.

- 실시간 액티브 서비스 개수 - 제니퍼 에이전트 별 액티브 서비스 개수를 경과 시간 구간으로 세분화해서 보여준다.
- 서비스 처리율 - 제니퍼 에이전트 별 서비스 처리율을 보여준다.
- 평균 응답 시간 - 제니퍼 에이전트 별 평균 응답 시간을 보여준다.
- 동시단말 사용자 수 - 제니퍼 에이전트 별 동시단말 사용자 수를 보여준다.
- 실시간 시스템 CPU 사용률 - 제니퍼 에이전트 별 시스템 CPU 사용률을 CPU 사용 영역으로 세분화해서 보여준다.

- 실시간 프로세스 힙 메모리 사용률 - 제니퍼 에이전트 별 프로세스 힙 메모리 사용률, 즉 프로세스 힙 전체 메모리에 대한 프로세스 힙 메모리 사용량의 비율을 보여준다.
- 실시간 DB 커넥션 개수 - 제니퍼 에이전트 별 DB 커넥션 개수를 커넥션 상태로 세분화해서 보여준다.

7.3. 사용자 정의 대시보드

사용자는 제니퍼 에이전트와 REMON과 같은 제니퍼 독립 에이전트를 통해서 수집한 성능 데이터를 이용해서 드래그 앤 드랍 방식으로 임의의 대시보드를 구성할 수 있다. 이렇게 구성한 대시보드를 사용자 정의 대시보드라고 한다.

7.3.1. 사용자 정의 대시보드 메뉴 생성

사용자 정의 대시보드를 구성하려면 우선 사용자 정의 대시보드 유형의 메뉴를 추가해야 한다.

- **[구성 관리 | 메뉴 관리]** 메뉴로 이동한다.
- 하위 메뉴로 사용자 정의 대시보드 메뉴를 추가할 상위 메뉴 이름을 클릭한 후 컨텍스트 메뉴에서 **[하위 메뉴 추가]** 메뉴를 클릭한다.
- 오른쪽 메뉴 입력 폼에서 이름을 입력하고 유형으로 **[사용자 정의 대시보드]**를 선택한다.
- [저장] 버튼을 클릭한다.

사용자 정의 대시보드 메뉴를 추가한 후에 해당 메뉴로 이동한다. 새롭게 추가한 사용자 정의 대시보드에는 어떤 차트도 표시되지 않고 오른쪽 하단에 **[편집]** 버튼과 **[변경]** 버튼만이 존재한다.

Notice: 사용자 정의 대시보드 구성 기능을 테스트할 수 있는 **[대시보드 | 사용자 정의 대시보드 데모]** 메뉴를 기본으로 제공한다.

7.3.2. 사용자 정의 대시보드 편집

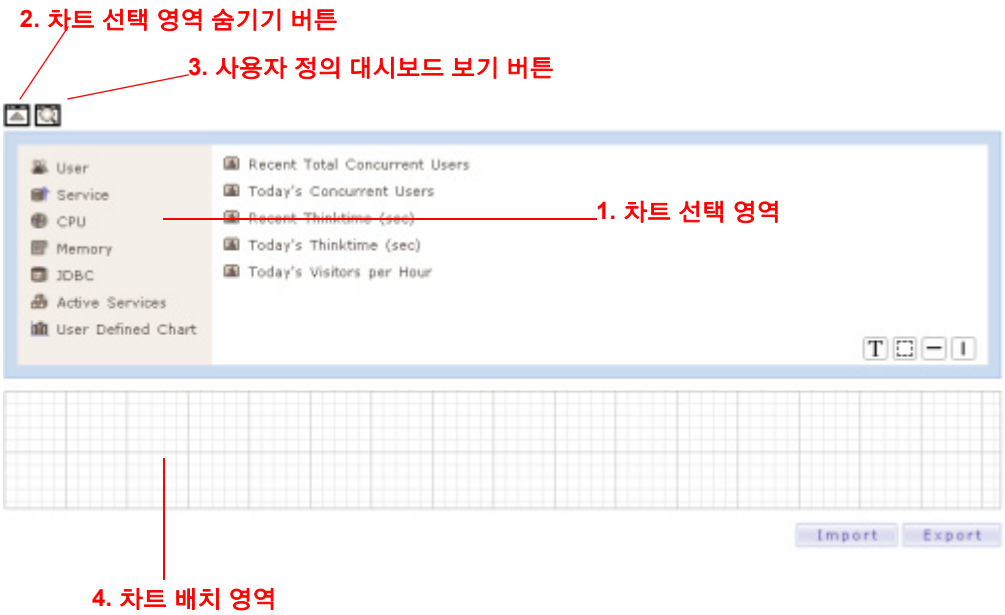
사용자 정의 대시보드를 편집하려면 오른쪽 아래에 있는 [편집] 버튼을 클릭한다.

Notice: pageedit 권한을 가지고 있는 그룹에 속한 사용자에게만 [편집] 버튼이 보인다.

7.3.2.1. 사용자 정의 대시보드 편집 화면 구성

사용자 정의 대시보드 편집 화면은 다음과 같이 구성된다.

그림 7-20: 사용자 정의 대시보드 편집 화면



1. 차트 선택 영역 - 사용자 정의 대시보드를 구성하는데 필요한 차트를 선택할 때 사용하는 영역이다. 맨 왼쪽에서 특정 그룹을 선택하면 해당 그룹에 속하는 모든 차트가 옆에 나타난다. 그 중에서 특정 차트를 선택하면 해당 차트가 오른쪽에 나타난다.
2. 차트 선택 영역 숨기기 버튼 - 해당 버튼을 클릭하면 차트 선택 영역이 사라진다. 반대로 차트 선택 영역이 숨겨진 상태에서 해당 버튼을 클릭하면 차트 선택 영역이 나타난다.
3. 사용자 정의 대시보드 보기 버튼 - 해당 버튼을 클릭하면 현재까지 구성한 사용자 정의 대시보드를 확인할 수 있다. 이는 상단 영역의 메뉴를 통해서 보는 것과 동일하다.
4. 차트 배치 영역 - 차트 선택 영역에서 선택한 차트를 드래그 앤 드랍을 통해서 차트 배치 영역에 놓는다. 차트 배치 영역 밖에 차트를 놓으면 차트가 정상적으로 표시되지 않는다.

7.3.2.2. 차트의 추가

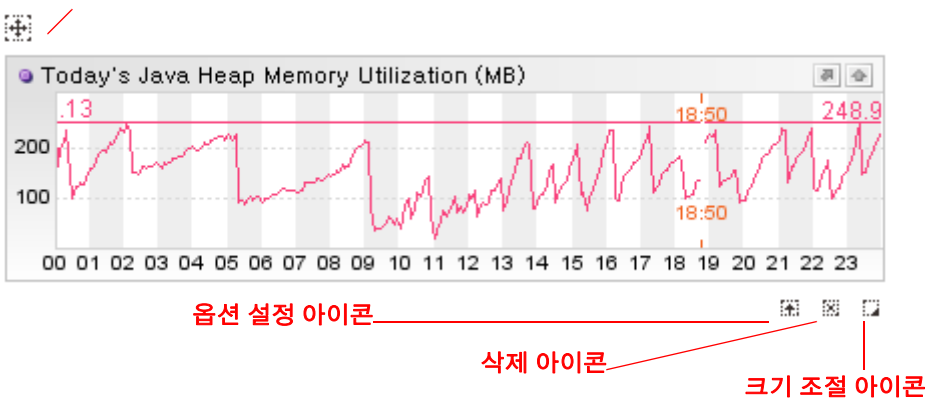
차트를 추가하려면 차트 선택 영역에서 사용자 정의 대시보드에 배치할 차트를 선택한다. 차트는 특성에 따라서 다음과 같이 몇 가지 그룹으로 구성된다.

표 7-1: 차트 그룹

그룹	설명
사용자	동시단말 사용자 수, 대기 시간, 시간당 방문자 수 등
서비스	서비스 요청률, 서비스 처리율, 평균 응답 시간, 시간당 호출 건수 등
CPU	시스템 CPU 사용률, 자바 프로세스 CPU 사용률 등
메모리	자바 힙 메모리 사용량, 시스템 메모리 사용량, 자바 프로세스 메모리 사용량 등
JDBC	JDBC 커넥션 개수 등
액티브 서비스	액티브 서비스 개수, 스피드 바, 스피드 미터, X-View 등
사용자 정의 차트	REMON과 같은 제니퍼 독립 에이전트로부터 수집한 데이터를 표시하는데 사용하는 차트

사용자 정의 대시보드 편집 화면에 놓여진 차트에는 왼쪽 상단과 오른쪽 하단에 몇 가지 아이콘이 함께 표시된다. 사용자는 이 아이콘을 통해서 차트의 위치를 이동하거나 차트의 크기를 조절할 수 있다.

그림 7-21: 사용자 정의 대시보드 편집 화면에서의 차트와 아이콘



사용자 정의 대시보드에 차트를 추가하는 방법은 다음과 같다.

- 차트 선택 영역에서 사용자 정의 대시보드에 추가할 차트를 선택한다. 그러면 오른쪽에 해당 차트가 나타난다.
- 해당 차트의 상단 이동 아이콘을 클릭한 후 드래그 앤 드랍으로 차트를 차트 배치 영역에 가져다 놓는다.

7.3.2.3. 차트의 이동

차트의 위치를 이동하는 방법은 다음과 같다.

- 위치를 이동할 차트의 상단 이동 아이콘을 클릭한다.
- 드래그 앤 드랍으로 해당 차트를 원하는 위치에 가져다 놓는다.

7.3.2.4. 차트의 크기 조절

차트의 크기를 조절하는 방법은 다음과 같다.

- 크기를 조절할 차트의 하단 크기 조절 아이콘을 클릭한다.
- 드래그 앤 드랍으로 해당 차트의 크기를 조절한다.

7.3.2.5. 차트의 삭제

차트를 삭제하는 방법은 다음과 같다.

- 삭제할 차트의 하단 삭제 아이콘을 클릭한다.

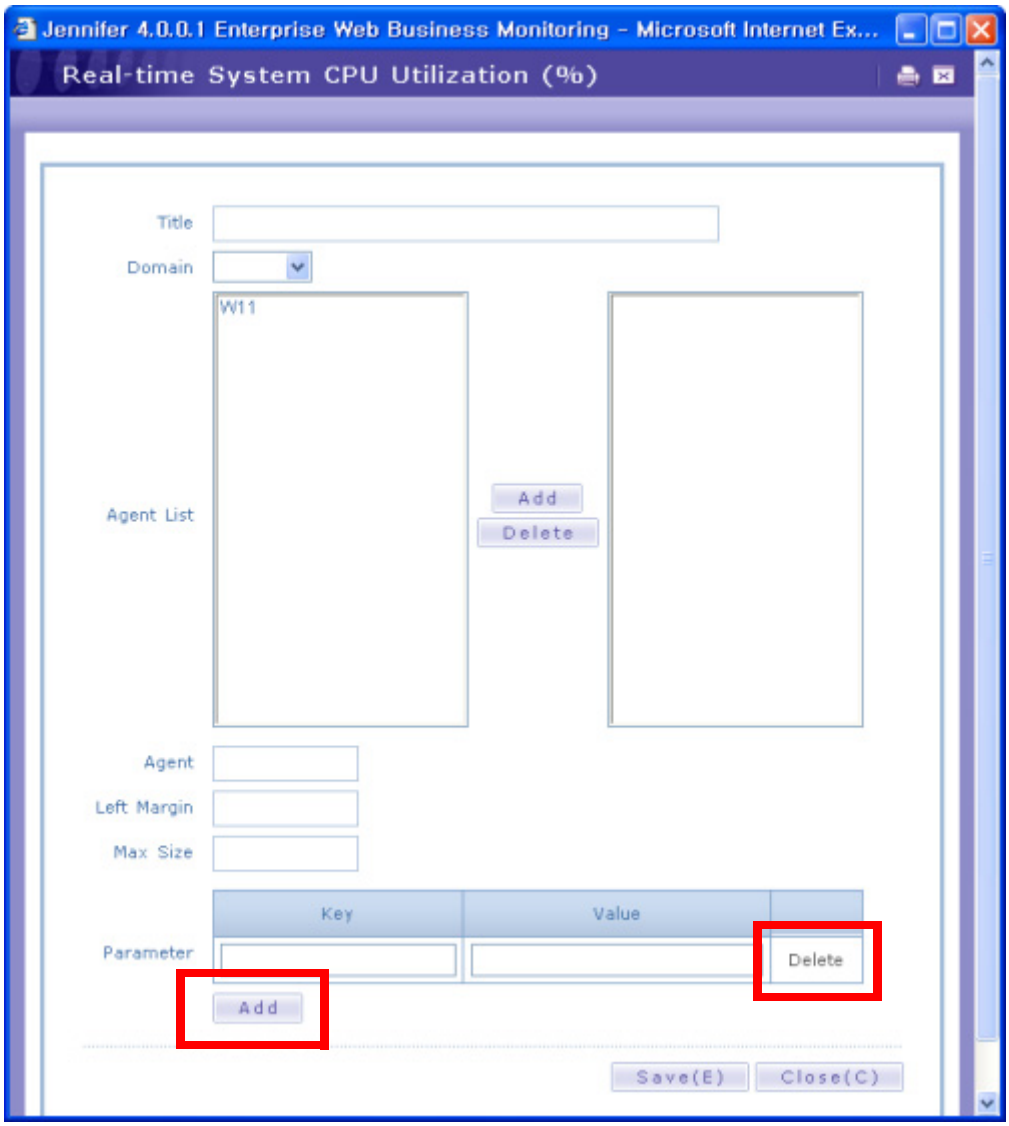
Notice: 차트는 자바 애플릿으로 구현되어 있고 아이콘은 HTML 태그로 구현되어 있는데, 기술적인 한계로 HTML 태그는 자바 애플릿 차트 위로 올라갈 수 없다. 따라서 2개 이상의 차트가 겹쳐지면 차트의 위치를 변경하거나 차트의 크기를 변경할 때 사용하는 아이콘이 다른 차트에 가려져서 보이지 않을 수 있다. 따라서 이를 고려해서 사용자 정의 대시보드를 구성하도록 한다.

7.3.3. 차트 옵션 설정

차트의 이름을 변경하거나 Y 축 최대 값을 지정하려면 차트의 옵션을 설정해야 한다. 차트의 옵션을 설정하는 방법은 다음과 같다.

- 옵션을 설정할 차트의 하단 옵션 설정 아이콘을 클릭한다.
- 옵션 설정 팝업 창에서 차트의 옵션을 설정한다.

그림 7-22: 옵션 설정 팝업 창



Notice: 차트가 차트 선택 영역이 아닌 차트 배치 영역에 있는 경우에만 옵션 설정 아이콘을 사용할 수 있다.

옵션은 제목, 차트 왼쪽 마진, 최대 값 등과 같이 입력 필드로 제공되는 필드 옵션과 키와 값 형식으로 입력하는 파라미터 옵션으로 나누어진다. 설정 가능한 필드 옵션은 차트의 성격에 따라서 달라진다.

파라미터 옵션을 설정하는 방법은 다음과 같다.

- 추가 버튼을 클릭하면 나타나는 입력 필드에 키와 값을 입력한다.

- 파라미터 옵션을 삭제하려면 [삭제] 링크를 클릭한다.

Notice: 옵션을 설정한 후에는 하단의 [저장] 버튼을 클릭해야만 설정 내용이 저장된다.

대부분의 차트에서 공통적으로 설정할 수 있는 필드 옵션은 다음과 같다.

표 7-2: 공통 필드 옵션

필드 옵션	설명
제목	차트의 이름으로 모든 차트에서 설정할 수 있는 옵션이다.
도메인	제니퍼 서버를 선택한다. 제니퍼 서버가 하나이면 [기본]으로 선택된다. 제니퍼 서버가 여러 개이면 도메인 선택에 따라서 제니퍼 에이전트가 달라진다. 모든 차트에서 설정할 수 있는 옵션이다.
에이전트	제니퍼 에이전트를 선택한다.
에이전트 목록	라인 차트, 런타임 라인 차트, 이퀄라이저 차트 등에서 여러 개의 제니퍼 에이전트를 표시하고자 할 때 선택한다. 단 에이전트 옵션으로 [모두]를 선택해야 한다.
오른쪽 막대 표시	런타임 라인 차트의 오른쪽에 막대로 현재 값을 표시하고자 할 때 사용한다.
오른쪽 막대 넓이	오른쪽 막대 표시 옵션을 선택한 후, 오른쪽 막대 영역의 넓이를 설정할 때 사용한다.
차트 왼쪽 마진	Y 축 좌표 영역의 넓이를 설정할 때 사용한다. 데이터의 최대 값에 따라서 Y 축 좌표가 차지하는 영역의 넓이가 달라진다. 이를 차트 왼쪽 마진을 통해서 임의의 값으로 고정할 수 있다.
최대 값	Y 축 좌표의 최대 값을 설정할 때 사용한다.

모든 차트에서 공통적으로 설정할 수 있는 파라미터 옵션은 다음과 같다.

표 7-3: 공통 파라미터 옵션

파라미터 옵션	설명
OMIT_TITLE	true로 설정하면 차트 제목 영역이 표시되지 않는다.
OMIT_BORDER	true로 설정하면 차트 보더가 표시되지 않는다.
ENABLE_POPUP	true로 설정하면 차트 팝업 창 열기 아이콘을 클릭했을 때 관련 화면이 팝업 창에서 열린다.
URL_LINK	ENABLE_POPUP 옵션을 true로 설정한 경우에 URL_LINK로 지정한 화면이 팝업 창에서 열린다. URL_LINK를 설정하지 않으면 /pop_remon_detail.jsp 화면이 열린다.

7.3.4. 일반 차트

7.3.4.1. 사용자

사용자 그룹에서 제공하는 차트는 다음과 같다.

- 실시간 동시단말 사용자 수 - 현재 동시단말 사용자 수를 이퀄라이저 차트로 표시한다.
- 최근 동시단말 사용자 수 - 최근 5분 동안의 동시단말 사용자 수를 런타임 라인 차트로 표시한다.
- 금일 동시단말 사용자 수 - 금일 동시단말 사용자 수를 라인 차트로 표시한다.
- 최근 대기 시간 - 최근 5분 동안의 대기 시간을 런타임 라인 차트로 표시한다.
- 금일 대기 시간 - 금일 대기 시간을 라인 차트로 표시한다.
- 금일 시간당 방문자 수 - 금일 시간당 방문자 수를 막대 차트로 표시한다.

7.3.4.2. 서비스

서비스 그룹에서 제공하는 차트는 다음과 같다.

- 최근 서비스 요청률 - 최근 5분 동안의 서비스 요청률을 런타임 라인 차트로 표시한다.
- 금일 서비스 요청률 - 금일 서비스 요청률을 라인 차트로 표시한다.
- 실시간 서비스 처리율 - 현재 서비스 처리율을 이퀄라이저 차트로 표시한다.
- 최근 서비스 처리율 - 최근 5분 동안의 서비스 처리율을 런타임 라인 차트로 표시한다.
- 금일 서비스 처리율 - 금일 서비스 처리율을 라인 차트로 표시한다.
- 실시간 평균 응답 시간 - 현재 평균 응답 시간을 이퀄라이저 차트로 표시한다.
- 최근 평균 응답 시간 - 최근 5분 동안의 평균 응답 시간을 런타임 라인 차트로 표시한다.
- 금일 평균 응답 시간 - 금일 평균 응답 시간을 라인 차트로 표시한다.
- 금일 시간당 호출 건수 - 금일 시간당 호출 건수를 막대 차트로 표시한다.

7.3.4.3. CPU

CPU 그룹에서 제공하는 차트는 다음과 같다.

- 실시간 시스템 CPU 사용률 - 현재 시스템 CPU 사용률을 이퀄라이저 차트로 표시한다.
- 최근 시스템 CPU 사용률 - 최근 5분 동안의 시스템 CPU 사용률을 런타임 라인 차트로 표시한다.
- 금일 시스템 CPU 사용률 - 금일 시스템 CPU 사용률을 라인 차트로 표시한다.

- 실시간 자바 프로세스 CPU 사용률 - 현재 자바 프로세스 CPU 사용률을 이퀄라이저 차트로 표시한다.
- 최근 자바 프로세스 CPU 사용률 - 최근 5분 동안의 자바 프로세스 CPU 사용률을 런타임 라인 차트로 표시한다.
- 금일 자바 프로세스 CPU 사용률 - 금일 자바 프로세스 CPU 사용률을 라인 차트로 표시한다.

7.3.4.4. 메모리

메모리 그룹에서 제공하는 차트는 다음과 같다.

- 실시간 자바 힙 메모리 사용률 - 현재 자바 힙 메모리 사용률을 이퀄라이저 차트로 표시한다.
- 실시간 자바 힙 메모리 사용량 - 현재 자바 힙 메모리 사용량을 이퀄라이저 차트로 표시한다.
- 최근 자바 힙 메모리 사용률 - 최근 5분 동안의 자바 힙 메모리 사용률을 런타임 라인 차트로 표시한다.
- 최근 자바 힙 메모리 사용량 - 최근 5분 동안의 자바 힙 메모리 사용량을 런타임 라인 차트로 표시한다.
- 금일 자바 힙 메모리 사용률 - 금일 자바 힙 메모리 사용률을 라인 차트로 표시한다.
- 금일 자바 힙 메모리 사용량 - 금일 자바 힙 메모리 사용량을 라인 차트로 표시한다.
- 최근 시스템 메모리 사용량 - 최근 5분 동안의 시스템 메모리 사용량을 런타임 라인 차트로 표시한다.
- 금일 시스템 메모리 사용량 - 금일 시스템 메모리 사용량을 라인 차트로 표시한다.
- 최근 자바 프로세스 메모리 사용량 - 최근 5분 동안의 자바 프로세스 메모리 사용량을 런타임 라인 차트로 표시한다.
- 금일 자바 프로세스 메모리 사용량 - 금일 자바 프로세스 메모리 사용량을 라인 차트로 표시한다.

7.3.4.5. DB

DB 그룹에서 제공하는 차트는 다음과 같다.

- 실시간 DB 커넥션 개수 - 현재 DB 커넥션 개수를 이퀄라이저 차트로 표시한다.
- 최근 DB 커넥션 개수 - 최근 5분 동안의 DB 커넥션 개수를 런타임 라인 차트로 표시한다.

7.3.4.6. 액티브 서비스

액티브 서비스 그룹에서 제공하는 차트는 다음과 같다.

- 실시간 액티브 서비스 개수 - 현재 액티브 서비스 개수를 이퀄라이저 차트로 표시한다.
- 최근 액티브 서비스 개수 - 최근 5분 동안의 액티브 서비스 개수를 런타임 라인 차트로 표시한다.
- 금일 액티브 서비스 개수 - 금일 액티브 서비스 개수를 라인 차트로 표시한다.
- 실시간 업무 처리량 스피드 바 - 전체 서비스 요청률, 서비스 처리율, 액티브 서비스 개수를 표시한다.
- 실시간 업무 처리량 스피드 미터 - 제니퍼 에이전트 별 서비스 요청률, 서비스 처리율, 액티브 서비스 개수를 표시한다.
- X-View - 트랜잭션 처리 현황을 표시한다.

X-View 차트의 경우에는 [SQL 파라미터 표시] 옵션을 통해서 프로파일 데이터에 나타나는 SQL 파라미터의 표시 여부를 설정할 수 있다.

7.3.5. 사용자 정의 차트

일반 차트는 사용자, 서비스, CPU, 메모리, DB, 액티브 서비스 등의 제니퍼 에이전트에서 수집한 성능 데이터를 이용해 사용자 정의 대시보드를 구성하는데 필요한 차트를 제공 한다. 반면에 사용자 정의 차트는 주로 REMON 등과 같은 제니퍼 독립 에이전트로부터 수집한 비정형 성능 데이터를 이용해 사용자 정의 대시보드를 구성하는데 필요한 차트를 제공한다..

Notice: 단, 사용자 정의 차트 그룹에 속해 있는 CPU, 노드, 노드 그룹, 경보 등의 차트는 REMON 과는 관련이 없다.

제니퍼 클라이언트는 제니퍼 서버로부터 일정한 시간 간격으로 REMON 데이터를 전송받는다. 만약 해당 시간 간격안에 새로운 REMON 데이터가 전송되지 않으면 해당 시점의 값은 0이 된다. 이를 위한 REMON 데이터 시간 간격은 제니퍼 클라이언트가 제니퍼 서버로부터 최근에 전송받은 2개의 데이터의 시간 차이로 정해진다.

그리고 LINE 등의 사용자 정의 차트와 같이 X 축이 존재하는 경우에는 버퍼 필드 옵션을 통해서 X 축 구간의 크기를 설정할 수 있다. X 축 구간의 크기는 버퍼 필드 옵션으로 설정한 값과 REMON 데이터 시간 간격의 곱으로 결정된다. 예를 들어, 버퍼가 30이고 REMON 데이터 시간 간격이 1분이면 X 축 구간의 크기는 30분이 된다.

버퍼를 설정하지 않으면 기본 값은 288이다. LINE, STACKED LINE , TABLE, BOX LINE, BOX BAR 등의 사용자 정의 차트와 같이 연속적인 값을 보여주는 경우에는 버퍼를 적절하게 설정한다.

자바 플러그인의 힙 메모리 사용량을 줄이기 위해서 EQUALIZER, HORIZONTAL BAR, PIE, ON/OFF CHECK, NUMBER, GAUGE 등의 사용자 정의 차트와 같이 현재 값만을 보여주는 경우에는 버퍼를 3으로 설정한다.

Notice: 현재 값만을 보여주는 경우에 버퍼를 1이 아닌 3으로 설정하는 이유는 REMON 데이터 시간 간격을 측정해서 데이터 수집에 실패했을 때 이를 0으로 표시하기 위해서이다.

7.3.5.1. AGENT SELECTOR

AGENT SELECTOR 사용자 정의 차트는 제니퍼 에이전트를 선택하여 다른 차트를 컨트롤할 수 있는 차트이다.

그림 7-23: AGENT SELECTOR

AGENT SELECTOR 차트에 반응하려면 해당 사용자 정의 차트에 다음 파라미터를 지정해야 한다.

```
AGENT_SELECTABLE = true
```

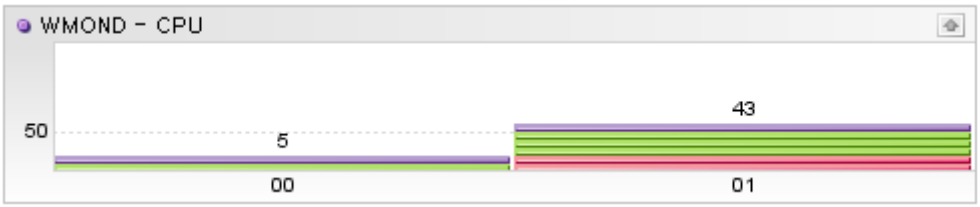
7.3.5.2. CPU

CPU 사용자 정의 차트는 WMOND를 통해서 수집한 CPU 사용률을 이퀄라이저 차트로 표시한다.

CPU 사용자 정의 차트를 사용하려면 차트 옵션 [에이전트]를 설정해야 한다.

Notice: WMOND는 제니퍼 에이전트와 상관이 없기 때문에 드랍 다운 리스트에서 에이전트를 선택할 수 없고, 텍스트 입력 필드에 WMOND 프로세스의 아이디를 직접 입력해야 한다.

그림 7-24: CPU 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-4: CPU 사용자 정의 차트 필드 옵션

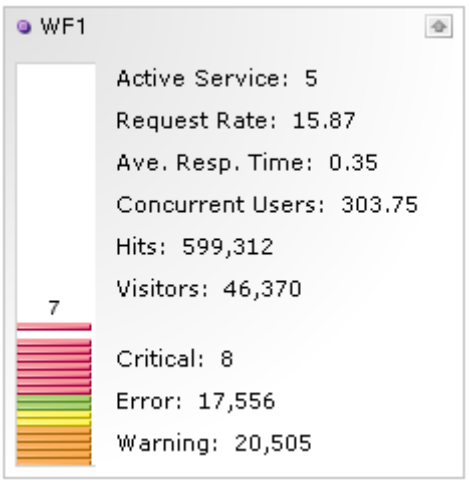
필드 옵션	설명
에이전트	WMOND 프로세스의 아이디를 입력한다. 제니퍼 에이전트와는 관련이 없다.

7.3.5.3. 노드

노드 사용자 정의 차트는 특정 제니퍼 에이전트에 대한 요약 정보를 표시한다.

노드 사용자 정의 차트를 사용하려면 차트 옵션 [에이전트]를 설정해야 한다.

그림 7-25: 노드 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-5: 노드 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	제니퍼 에이전트 아이디를 입력한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-6: 노드 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
FILTER_ITEM	특정 성능 데이터 항목만을 보기위해서는, 표시할 성능 데이터 항목 번호를 콤마 [,]를 구분자로 사용하여 입력한다.

표 7-6: 노드 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
SHOW_CHART	제니퍼 에이전트의 액티브 서비스 개수를 이퀄라이저로 보여준다. 액티브 서비스 개수 이퀄라이저를 표시하지 않으려면 파라미터 옵션 SHOW_CHART를 false로 설정한다.

성능 데이터 항목 번호는 다음 표를 참조한다.

표 7-7: 성능 데이터 항목 번호

항목 번호	설명
1	액티브 서비스 개수
2	서비스 요청률
3	평균 응답 시간
4	동시단말 사용자 수
5	호출 건수
6	방문자 수
7	심각
8	에러
9	경고

7.3.5.4. 노드 그룹

노드 그룹 사용자 정의 차트는 여러 개의 노드 차트를 동시에 표시한다.

그림 7-26: 노드 그룹 사용자 정의 차트

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-8: 노드 그룹 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
COL	행의 숫자를 설정한다.
ROW	열의 숫자를 설정한다.
NODE_LIST	노드 구성을 위한 제니퍼 에이전트 목록을 콤마[,]를 구분자로 설정한다. 도메인이 다른 경우에는 도메인 아이디를 제니퍼 에이전트 아이디 앞에 콜론[:]을 구분자로 설정한다.

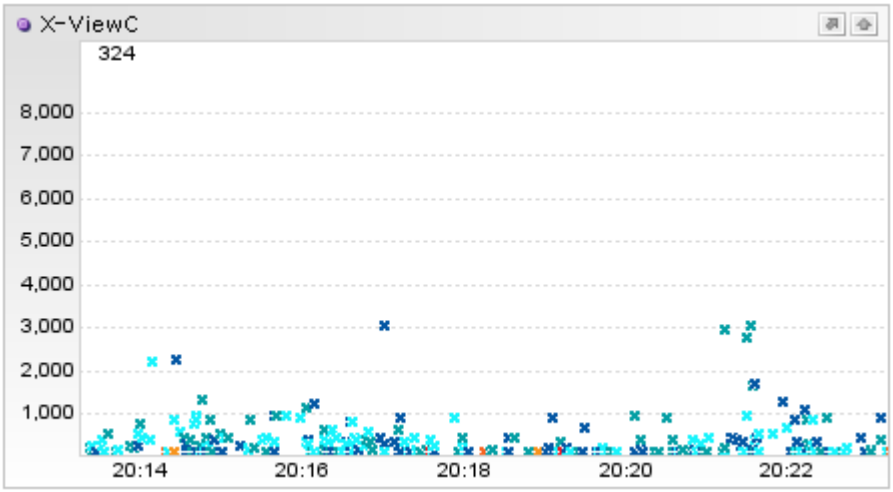
그리고 노드와 마찬가지로 FILTER_ITEM 옵션과 SHOW_CHART 옵션을 사용할 수 있다.

7.3.5.5. X-ViewC

REMON으로부터 수집한 데이터를 X-View 차트로 표시한다.

Notice: 모든 REMON 데이터를 X-ViewC 차트로 표시할 수는 없고, X-ViewC 차트에 맞게 설정한 REMON 데이터만을 사용할 수 있다.

그림 7-27: X-ViewC 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

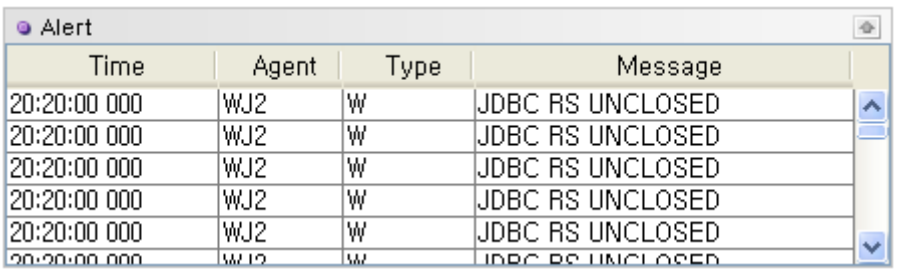
표 7-9: X-ViewC 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	에이전트를 설정하지 않으면 모든 X-ViewC 데이터가 표시된다. 특정 REMON 에이전트가 수집한 X-ViewC 데이터만을 표시하려면 이를 설정한다. 2개 이상의 경우에는 콤마[,]를 구분자로 구분하여 설정한다.

7.3.5.6. 경보

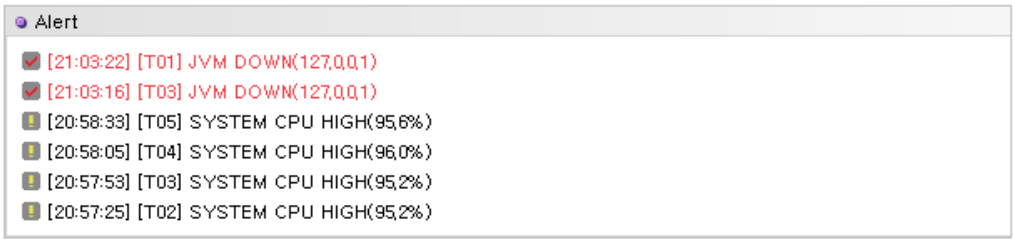
경보 사용자 정의 차트는 경보 메시지를 표시한다.

그림 7-28: 경보 사용자 정의 차트 - 그리드



파라미터 옵션 IS_TEXT를 true로 설정하면 그리드가 아닌 텍스트 형식으로 표시된다.

그림 7-29: 경보 사용자 정의 차트 - 텍스트



설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-10: 경보 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
IS_TEXT	true를 입력하면 경보 사용자 정의 차트가 텍스트 형식으로 표시된다.
VIEW_AGENTS	경보가 발생한 제니퍼 에이전트를 필터링하려면 제니퍼 에이전트 아이디를 콤마[,]를 구분자로 입력한다. 단, 그리드 형식에서만 사용 가능한 옵션이다.
FILTER_TYPE	경보 유형에 대한 필터링 조건을 설정할 때 사용한다. 심각은 C, 에러는 E, 경보는 W로 표시되는데 심각 유형만을 나타나게 하려면 FILTER_TYPE 옵션으로 C를 입력한다. 심각과 에러 유형을 나타나게 하려면 콤마[,]를 구분자로 사용하여 C,E를 입력한다. 단, 그리드 형식에서만 사용 가능한 옵션이다.
FILTER_NAME	경보 메시지에 대한 필터링 조건을 설정할 때 사용한다. LIKE 검색을 지원한다. 단, 그리드 형식에서만 사용 가능한 옵션이다.

7.3.5.7. LINE

LINE 사용자 정의 차트는 REMON으로부터 수집한 데이터를 런타임 라인 차트로 표시한다.

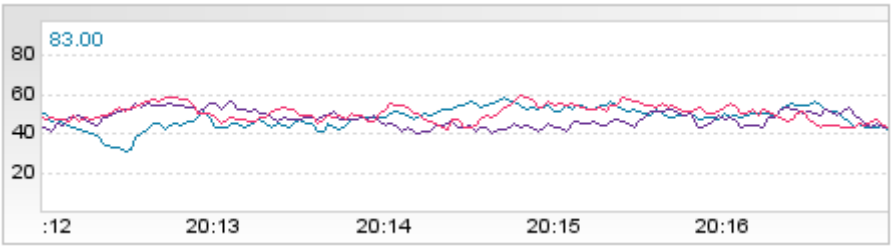


그림 7-30: LINE 사용자 정의 차트

설정이 가능한 필드 옵션은 다음과 같다.

표 7-11: LINE 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.
범례 표시	범례 표시 여부를 설정한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-12: LINE 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
LINE_WIDTH	라인의 두께를 설정할 때 사용한다.
COLORS	선 색상을 입력한다. 색상에 대한 포멧은 RGB(255,255,255)이고 2개 이상의 색상은 슬러쉬(/)를 구분자로 사용하여 입력한다. 예를 들어, 선 색상으로 빨간색과 파랑색을 사용하려면 255,0,0/0,0,255을 입력한다.
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.

7.3.5.8. TIME LINE

TIME LINE 사용자 정의 차트는 REMON으로부터 수집한 데이터를 24 시간 라인 차트로 표시한다.

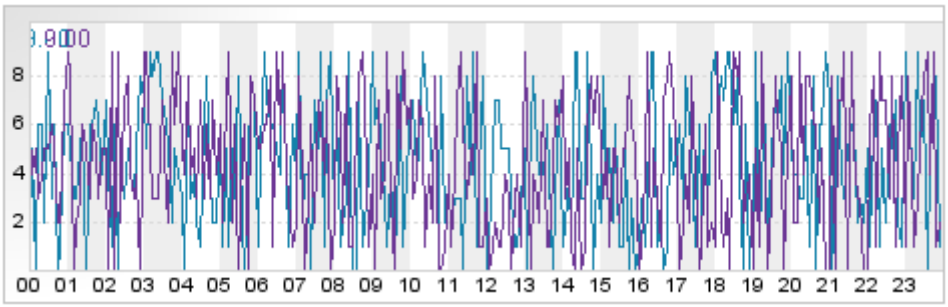


그림 7-31: TIME LINE 사용자 정의 차트

TIME LINE 사용자 정의 차트를 사용하려면 REMON을 다음과 같이 작성하여야 한다.

```
packet.addField("today", new INT(1)); // 필드이름만 사용. 데이터는 의미없음
packet.addField("yesterday", new INT(2));

for (int i = 0; i < 288; i++) {
  TYPE[] types = new TYPE[2];
  types[0] = new INT(random.nextInt(10)); // today
  types[1] = new INT(random.nextInt(10)); // yesterday
  packet.addAttachedRow(types);
}
```

설정이 가능한 필드 옵션은 다음과 같다.

표 7-13: TIME LINE 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.
범례 표시	범례 표시 여부를 설정한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

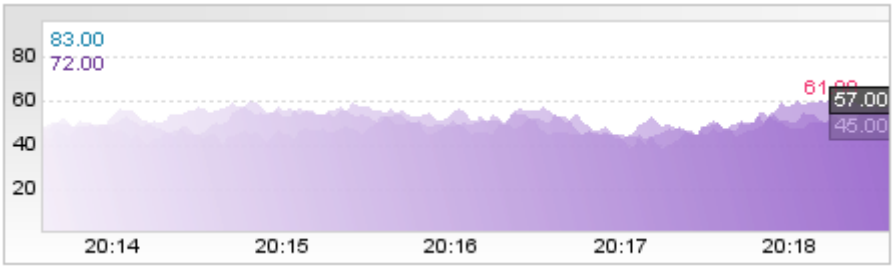
표 7-14: TIME LINE 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
LINE_WIDTH	라인의 두께를 설정할 때 사용한다.
COLORS	선 색상을 입력한다. 색상에 대한 포멧은 RGB(255,255,255)이고 2개 이상의 색상은 슬러쉬(/)를 구분자로 사용하여 입력한다. 예를 들어, 선 색상으로 빨간색과 파랑색을 사용하려면 255,0,0/0,0,255을 입력한다.
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.

7.3.5.9. STACKED LINE

STACKED LINE 사용자 정의 차트는 REMON으로부터 수집한 데이터를 런타임 에어리어 차트로 표시한다.

그림 7-32: STACKED LINE 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-15: STACKED LINE 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.
범례 표시	범례 표시 여부를 설정한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

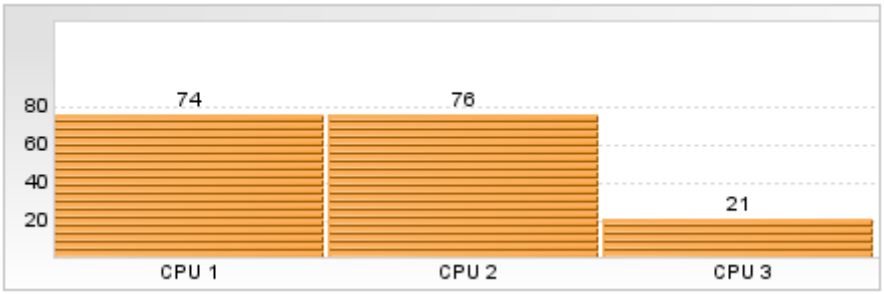
표 7-16: STACKED LINE 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.

7.3.5.10. EQUALIZER

EQUALIZER 사용자 정의 차트는 REMON으로부터 수집한 데이터를 이퀄라이저 차트로 표시한다.

그림 7-33: EQUALIZER 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-17: EQUALIZER 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

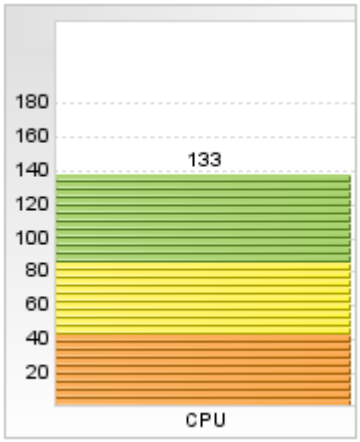
표 7-18: EQUALIZER 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
COLORS	이퀄라이저 색상을 입력한다. 색상에 대한 포멧은 RGB(255,255,255)이고 2개 이상의 색상은 슬러쉬(/)를 구분자로 입력한다. 예를 들어, 이퀄라이저 색상으로 빨간색과 파랑색을 사용하려면 255,0,0/0,0,255을 입력한다.
_COLORS	이퀄라이저가 비활성화되어 있을 때의 색상을 입력한다. 포멧은 COLORS와 동일하다.
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.

7.3.5.11. STACKED EQUALIZER

STACKED EQUALIZER 사용자 정의 차트는 REMON으로부터 수집한 데이터를 이퀄라이저 차트로 표시한다.

그림 7-34: STACKED EQUALIZER 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-19: STACKED EQUALIZER 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

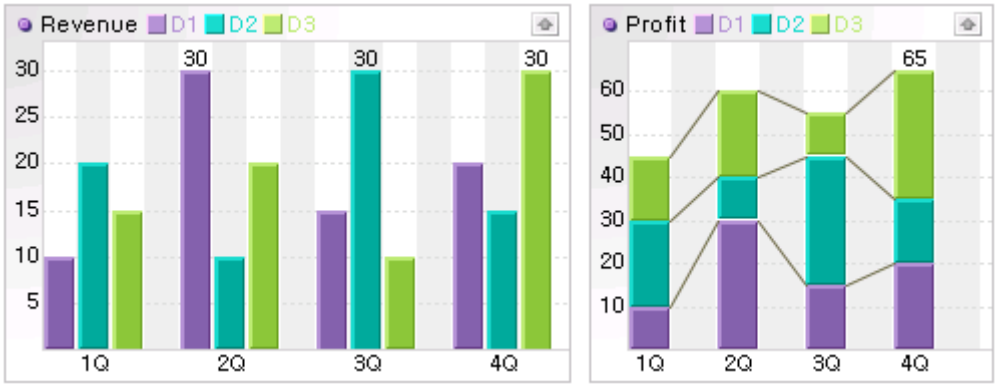
표 7-20: STACKED EQUALIZER 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.

7.3.5.12. BAR

BAR 사용자 정의 차트는 REMON으로부터 수집한 데이터를 막대 차트로 표시한다.

그림 7-35: BAR 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-21: BAR 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.
최대 값	최대 값을 입력한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-22: BAR 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
GROUP_SIZE	데이터를 그룹으로 묶어서 표현하려면 그룹 크기를 입력한다. 입력하지 않으면 데이터를 그룹으로 묶지 않는다.
ORIENTATION	데이터를 그룹으로 묶을 때, HORIZONTAL을 입력하면 막대를 수평으로 표시하고, VERTICAL을 입력하면 막대를 수직으로 표시한다. 기본 값은 HORIZONTAL이다.
GROUP_LABEL	그룹 항목에 대한 이름을 콤마를 구분자로 해서 설정한다. 제목을 입력하면 제목과 함께 범례로 이 이름을 표시한다.
SHOW_LINE	ORIENTATION을 VERTICAL로 입력한 경우에만 의미가 있다. true를 입력하면 막대를 선으로 연결한다.기본 값은 true이다.

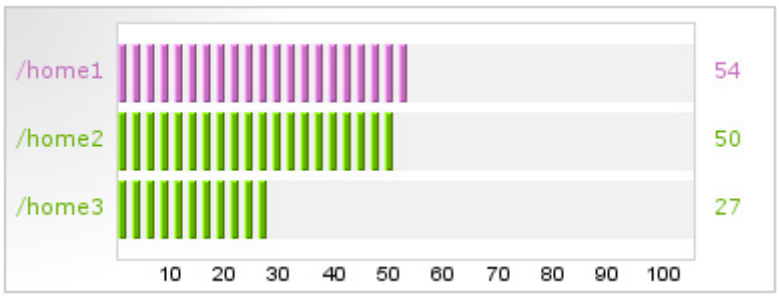
표 7-22: BAR 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
SHOW_VALUE	true를 입력하면 모든 막대 값이 표시된다. 기본 값은 false이다.

7.3.5.13. HORIZONTAL BAR

HORIZONTAL BAR 사용자 정의 차트는 REMON으로부터 수집한 데이터를 수평 막대 차트로 표시힌다. 주로 디스크 사용률을 모니터링하는데 사용한다.

그림 7-36: HORIZONTAL BAR 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-23: HORIZONTAL BAR 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.
최대 값	최대 값을 입력한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-24: HORIZONTAL BAR 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
IS_SIMPLE	true로 설정하면 단순한 막대로 표현된다.

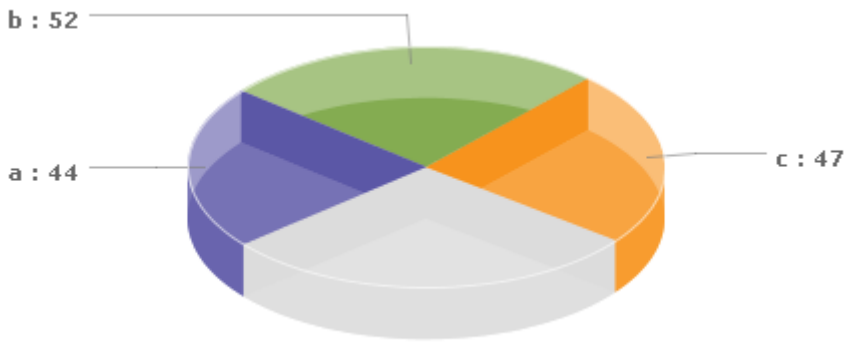
표 7-24: HORIZONTAL BAR 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
RANGES	값에 따라서 색상을 다르게 표현할 때 사용한다. 콤마[,]를 구분자로 사용하고, 80,60,40와 같이 큰 값부터 지정한다. 기본 값은 3.2 디스크 차트를 고려해서 90,70이다.
COLORS	값에 따라서 색상을 다르게 표현할 때 사용한다. RANGES 옵션보다 1개를 더 설정해야 한다. 슬러시[/]를 구분자로 RGB 색상 값을 255,0,0/0,255,0/0,0,255와 같이 지정한다. 기본 값은 3.2 디스크 차트를 고려해서 255,35,15/198,111,190/101,178,4 이다.
FORMAT	오른쪽에 나타나는 숫자의 포멧을 지정할 때 사용한다. 기본 값은 %,.0f이다. 3.2 디스크 차트와 같이 마지막에 %를 넣으려면 %,.0f%%로 설정한다.

7.3.5.14. PIE

PIE 사용자 정의 차트는 REMON으로부터 수집한 데이터를 파이 차트로 표시힌다. 기존의 METER 사용자 정의 차트는 이 차트로 통합되었다.

그림 7-37: PIE 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-25: PIE 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.

표 7-25: PIE 사용자 정의 차트 필드 옵션

필드 옵션	설명
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.
최대 값	최대 값을 입력한다. 이 값을 입력하지 않으면 전체 데이터의 합이 최대 값이 된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

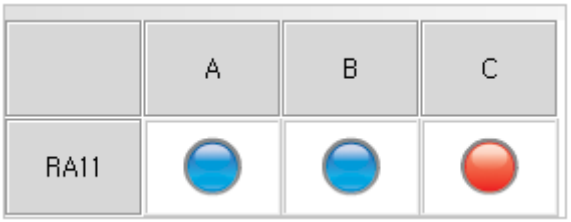
표 7-26: PIE 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
IS_3D	true로 설정하면 3D 형태로 나타난다. 기본 값은 false이다.
TOTAL_ANGLE	PIE의 전체 각도로 360, 180 등의 값을 설정한다.
START_ANGLE	첫번째 데이터가 표시되는 각도로 이 각도를 기준으로 시계 방향으로 데이터가 표시된다. 기본 값은 220이다. 아래쪽은 270, 왼쪽은 180, 위쪽은 90, 오른쪽은 0이다.
DONUT_RADIUS	PIE 차트를 도넛츠 형태로 표시하려면 0.1에서 0.9 사이의 값을 설정한다. 이는 전체 지름에 대한 도넛츠 크기의 비율을 의미한다.
IS_FILL	true로 설정하면 MAX_VALUE보다 작은 영역을 채운다. 기본 값은 false이다. 3D PIE 차트의 경우에는 DONUT_RADIUS를 설정하면 IS_FILL은 항상 true가 된다.
IS_FIXED	3D PIE 차트가 아닌 경우에 정원으로 하려면 true로 설정한다. 기본 값은 false이다.
IS_PERCENT	값을 최대 값에 대한 비율로 표시하려면 true로 설정한다. 기본 값은 false이다.
COLORS	각 PIE의 색상을 RGB 값으로 설정한다. 255,0,0/0,255,0/0,0,255 등의 형식으로 설정해야 한다.
FORMAT	데이터 포멧을 설정한다. 기본 값은 %2$s : %1$,.0f이다. 2$는 데이터 이름을, 1$ 데이터 값을 의미한다.
IS_SORT	true로 설정하면 데이터 크기를 기준으로 파이를 정렬하고, 데이터 크기가 0인 경우에는 파이를 차트에 표시하지 않는다. 기본 값은 false이다.

7.3.5.15. ON/OFF CHECK

ON/OFF CHECK 사용자 정의 차트는 REMON으로부터 수집한 데이터를 경고등으로 표시한다.

그림 7-38: ON/OFF CHECK 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-27: ON/OFF CHECK 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다. 콤마[,]를 구분자로 사용하여 여러 개의 차트를 입력할 수 있다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-28: ON/OFF CHECK 사용자 정의 차트 파라미터 옵션

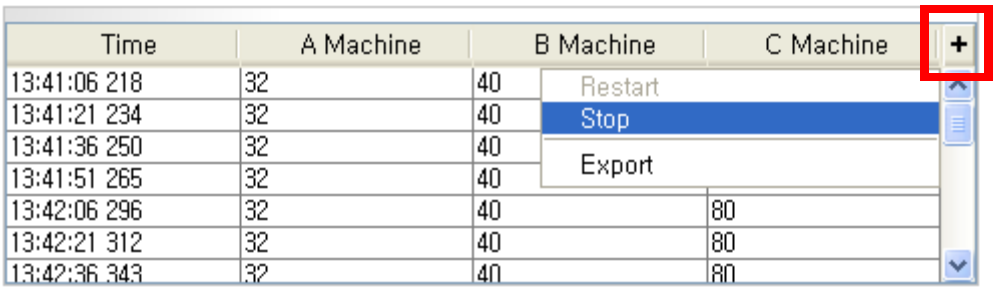
파라미터 옵션	설명
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.
SHOW_ICON	경고등 아이콘의 표시 여부를 설정한다.
SHOW_VALUE	값의 표시 여부를 설정한다.
PADDING	경고등 영역과 차트 보더와의 간격을 설정한다. 콤마[,]를 구분자로 사용하여 위쪽, 오른쪽, 아래쪽, 왼쪽 간격을 설정하면 된다.
WARNING_VALUE_OVER_THAN	해당 값보다 크면 경고등이 붉은 색으로 변경된다.
WARNING_VALUE_LOWER_THAN	해당 값보다 작으면 경고등이 붉은 색으로 변경된다.

Notice: WARNING_VALUE_OVER_THAN 옵션과 WARNING_VALUE_LOWER_THAN 옵션을 모두 설정하지 않으면 값이 0인 경우에 경고등이 붉은 색으로 변경된다.

7.3.5.16. TABLE

TABLE 사용자 정의 차트는 REMON으로부터 수집한 데이터를 그리드로 표시한다.

그림 7-39: TABLE 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-29: TABLE 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-30: TABLE 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.
SHOW_TIME	기본적으로 첫번째 칼럼에 해당 데이터가 수집된 시간이 표시된다. 이 옵션을 false로 설정하면 시간이 표시되지 않는다.
COLUMN_WIDTHS	콤마를 구분자로 해서 각 칼럼의 크기를 설정한다. 단위는 픽셀이다.
ADD_MODE	기본 값은 BOTTOM으로 새로운 데이터는 하단에 추가된다. 이 옵션을 TOP으로 설정하면 새로운 데이터는 상단에 추가된다.

TABLE 사용자 정의 차트 오른쪽 상단에 있는 [+] 아이콘을 통해서 다음 작업을 수행할 수 있다.

- 정지 - 새로 추가된 데이터가 자동으로 업데이트된다. 이 자동 업데이트를 정지할 수 있다.
- 재시작 - 자동 업데이트를 다시 시작한다.
- Export - 데이터를 CSV 형식 파일로 Export한다.

7.3.5.17. NUMBER

NUMBER 사용자 정의 차트는 REMON으로부터 수집한 데이터를 숫자로 표시한다.

그림 7-40: NUMBER 사용자 정의 차트

설정이 가능한 필드 옵션은 다음과 같다.

표 7-31: NUMBER 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-32: NUMBER 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.
FONT	폰트의 이름, 스타일, 크기를 설정한다. 포멧은 이름, 스타일, 크기를 콤마[,]를 구분자로 입력한다. 스타일의 경우에 0은 PLAIN을, 1은 BOLD를, 2는 ITALIC을 의미한다. 따라서 굴림,1,11로 입력하면 굴림 폰트를 사용하며 굵게 표시하고 크기는 11을 의미하게 된다. TYPE을 설정하지 않은 경우에만 의미가 있다.
COLOR	색상을 입력한다. 색상에 대한 포멧은 RGB(255,255,255)이다.

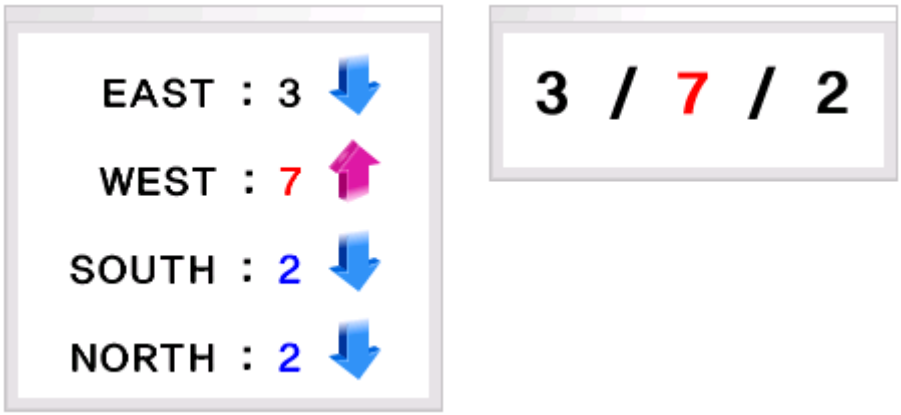
표 7-32: NUMBER 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
TYPE	digital 혹은 plain을 설정한다. EFFECT, EFFECT_COLOR, IS_REFLECTION 등의 옵션은 TYPE을 설정한 경우에만 의미를 갖는다.
TEXT_ALIGN	텍스트 좌우 정렬을 설정한다. left는 왼쪽, right는 오른쪽, center는 가운데를 의미한다. 기본 값은 center이다.
EFFECT	glow, shadow 등을 설정한다.
EFFECT_COLOR	효과와 관련한 색상을 RGB(255,0,0) 형식으로 설정한다.
IS_REFLECTION	true로 설정하면 반영 효과가 나타난다. 기본 값은 true이다.

7.3.5.18. TEXT

TEXT 사용자 정의 차트는 REMON으로부터 수집한 데이터를 텍스트로 표시한다.

그림 7-41: TEXT 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-33: TEXT 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-34: TEXT 사용자 정의 차트 파라미터 옵션

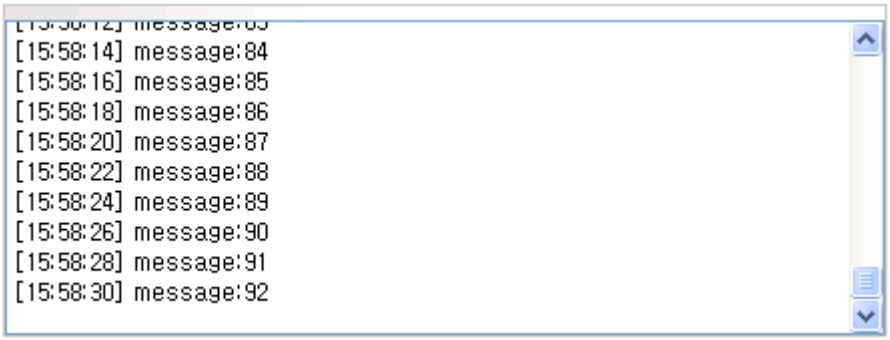
파라미터 옵션	설명
FORMAT	데이터 포멧을 설정한다. 기본 값은 %,.0f 이다.
TEXT_PATTERN	텍스트 포멧을 설정한다. 기본 값은 ${name} : ${value} 이다. 이 경우에는 모든 모든 데이터에 대해서 반복적으로 텍스트를 표시한다. ${} 밖에서는 임의의 문자열을 사용할 수 있다. 특정 필드만을 출력하려면 ${fieldname.name}과 ${fieldname.value}를 혼합해서 사용한다. 단, 이 경우에는 모든 데이터에 대해서 반복하지 않는다. 예를 들어, ${F1.value} / ${F2.value} / ${F3.value}와 같이 설정할 수 있다.
TEXT_ALIGN	텍스트 좌우 정렬을 설정한다. left는 왼쪽, right는 오른쪽, center는 가운데를 의미한다. 기본 값은 left이다.
LINE_HEIGHT	줄 간격으로 설정하지 않으면 폰트 크기에 맞추어 조정된다.
PADDING	왼쪽 혹은 오른쪽 여백을 설정한다. 단위는 픽셀이다.
FONT	폰트 이름, 스타일, 크기를 설정한다. 포멧은 이름, 스타일, 크기를 콤마[,]를 구분자로 입력한다. 스타일의 경우에 0은 PLAIN을, 1은 BOLD를, 2는 ITALIC을 의미한다. 따라서 굴림,1,11로 입력하면 굴림 폰트를 사용하며 굵게 표시하고 크기는 11을 의미하게 된다.
COLOR	텍스트 색상을 입력한다. 색상에 대한 포멧은 RGB(255,255,255)이다.
MAX_COLOR	최대 값에 대한 색상을 입력한다. 색상에 대한 포멧은 RGB(255,255,255)이다. 기본 값은 COLOR 옵션으로 설정한 색상이다.
MIN_COLOR	최소 값에 대한 색상을 입력한다. 색상에 대한 포멧은 RGB(255,255,255)이다. 기본 값은 COLOR 옵션으로 설정한 색상이다.
SHOW_FRAME	외각 프레임 표시 여부를 설정한다. 기본 값은 true이다.
SHOW_IMAGE	업/다운 이미지 출력 여부를 설정한다. 기본 값은 false이다.
DELIMIT	업/다운 이미지 경계 값을 설정한다. 기본 값은 0이다.

7.3.5.19. TEXT AREA

TEXT AREA 사용자 정의 차트는 REMON으로부터 수집한 데이터를 텍스트 에어리어에 표시한다.

Notice: REMON 데이터가 stream 유형인 경우에만 이 차트를 사용할 수 있다.

그림 7-42: TEXT AREA 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-35: TEXT AREA 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

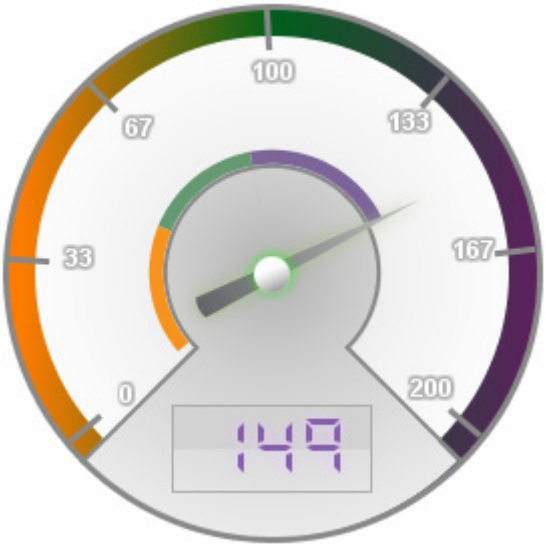
표 7-36: TEXT AREA 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
IS_TAIL	새로운 텍스트를 하단에 추가할지 상단에 추가할지를 설정한다. 기본 값은 true 이다.
SHOW_TIME	시간을 표시할지 여부를 설정한다. 기본 값은 true이다.
MAX_LENGTH	텍스트 글자 수를 제한한다. 기본 값은 5000이다.
COLOR	글자 색을 설정한다.
COLOR_PATTERN	메시지 단위로 글자 패턴을 추출하여 색상을 설정한다. 다음과 같이 설정한다. ERROR=255,0,0;WARN=0,0,255

7.3.5.20. GAUGE1

GAUGE1 사용자 정의 차트는 REMON으로부터 수집한 데이터를 계기판으로 표시한다.

그림 7-43: GAUGE1 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-37: GAUGE1 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
최대 값	최대 값을 입력한다. 이 값을 입력하지 않으면 전체 데이터의 합이 최대 값이 된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-38: GAUGE1 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.
IS_SHADOW	true로 설정하면 드랍 새도우 효과가 적용된다.
IS_PERCENT	값을 최대 값에 대한 비율로 표시하려면 true로 설정한다. 기본 값은 false이다.
RANGES	선택한 데이터가 하나인 경우에 경계 값을 설정할 수 있다. 경계 값을 1,3,5,9 처럼 콤마[,]를 구분자로 입력한다.

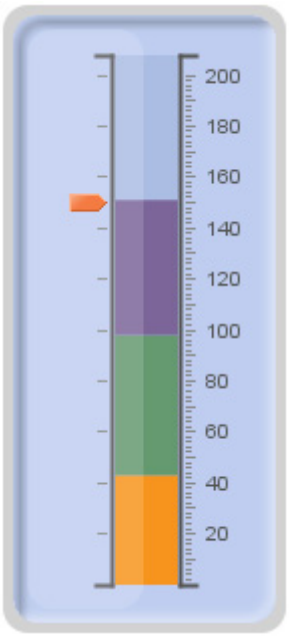
표 7-38: GAUGE1 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
COLORS	각 영역의 색상을 RGB 값으로 설정한다. 255,0,0/0,255,0/0,0,255 등의 형식으로 설정해야 한다.

7.3.5.21. GAUGE2

GAUGE2 사용자 정의 차트는 REMON으로부터 수집한 데이터를 계기판으로 표시한다.

그림 7-44: GAUGE2 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-39: GAUGE2 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
최대 값	최대 값을 입력한다. 이 값을 입력하지 않으면 전체 데이터의 합이 최대 값이 된다.

설정이 가능한 파라미터 옵션은 다음과 같다.

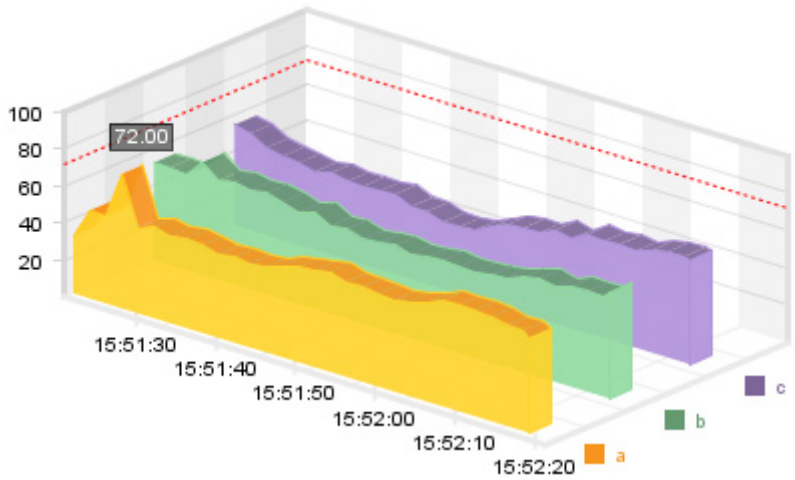
표 7-40: GAUGE2 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
FORMAT	데이터의 포멧을 입력한다. 기본 값은 %,.0f 이다.
IS_PERCENT	값을 최대 값에 대한 비율로 표시하려면 true로 설정한다. 기본 값은 false이다.
RANGES	선택한 데이터가 하나인 경우에 경계 값을 설정할 수 있다. 경계 값을 1,3,5,9 처럼 콤마[,]를 구분자로 입력한다.
COLORS	각 영역의 색상을 RGB 값으로 설정한다. 255,0,0/0,255,0/0,0,255 등의 형식으로 설정해야 한다.

7.3.5.22. BOX LINE

BOX LINE 사용자 정의 차트는 REMON으로부터 수집한 데이터를 3D 라인 차트로 표시한다.

그림 7-45: BOX LINE 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-41: BOX LINE 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.

표 7-41: BOX LINE 사용자 정의 차트 필드 옵션

필드 옵션	설명
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.
최대 값	최대 값을 입력한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-42: BOX LINE 사용자 정의 차트 파라미터 옵션

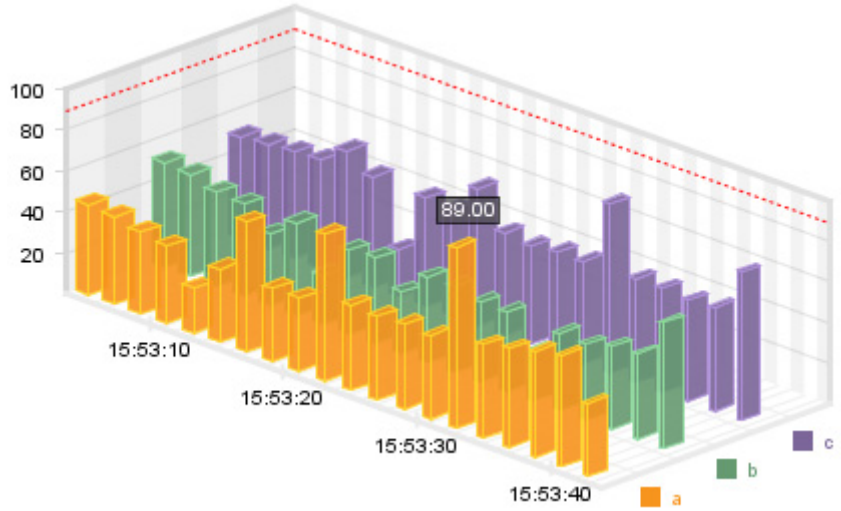
파라미터 옵션	설명
WIDTH_RATE	왼쪽 면의 넓이 비율을 설정한다. 0.1에서 0.9 사이의 값을 설정한다. 이는 전체 넓이에 대한 왼쪽 면의 비율을 의미한다.
HEIGHT_RATE	상단 면의 높이 비율을 설정한다. 0.1에서 0.9 사이의 값을 설정한다. 이는 전체 높이에 대한 상단 면의 높이의 비율을 의미한다.
START_ANGLE	박스의 각도를 설정한다. 10에서 90 사이의 값을 설정한다.
IS_FILL	true로 설정하면 라인의 하단 영역을 채운다. 기본 값은 false이다.
COLORS	각 라인 혹은 막대의 색상을 RGB 값으로 설정한다. 255,0,0/0,255,0/0,0,255 등의 형식으로 설정해야 한다.

BOX LINE 차트의 특정 영역을 클릭한 상태에서 마우스를 좌우로 이동한 후에 놓으면 START_ANGLE이 이동한 크기 만큼 변경된다. 또한 상단 꼭지점을 클릭한 상태에서 마우스를 좌우로 이동한 후에 놓으면 WIDTH_RATE가 이동한 크기 만큼 변경된다.

7.3.5.23. BOX BAR

BOX BAR 사용자 정의 차트는 REMON으로부터 수집한 데이터를 3D 막대 차트로 표시한다.

그림 7-46: BOX BAR 사용자 정의 차트



설정이 가능한 필드 옵션은 다음과 같다.

표 7-43: BOX BAR 사용자 정의 차트 필드 옵션

필드 옵션	설명
에이전트	REMON 에이전트를 입력한다.
스크립트	REMON 스크립트를 입력한다.
데이터	REMON으로부터 수집한 데이터의 유형을 입력한다. 콤마[,]를 구분자로 사용하여 데이터 유형을 입력한다. 단, 입력하지 않으면 모든 데이터 유형이 차트에 표시된다.
표시	데이터 유형에 대해서 임의의 이름을 부여할 때 사용한다. 임의의 이름을 콤마[,]를 구분자로 입력한다. 단, 입력하지 않으면 데이터 유형이 이름으로 사용된다.
최대 값	최대 값을 입력한다.

설정이 가능한 파라미터 옵션은 다음과 같다.

표 7-44: BOX BAR 사용자 정의 차트 파라미터 옵션

파라미터 옵션	설명
WIDTH_RATE	왼쪽 면의 넓이 비율을 설정한다. 0.1에서 0.9 사이의 값을 설정한다. 이는 전체 넓이에 대한 왼쪽 면의 비율을 의미한다.
HEIGHT_RATE	상단 면의 높이 비율을 설정한다. 0.1에서 0.9 사이의 값을 설정한다. 이는 전체 높이에 대한 상단 면의 높이의 비율을 의미한다.
START_ANGLE	박스의 각도를 설정한다. 10에서 90 사이의 값을 설정한다.

표 7-44: BOX BAR 사용자 정의 차트 파라미터 옵션

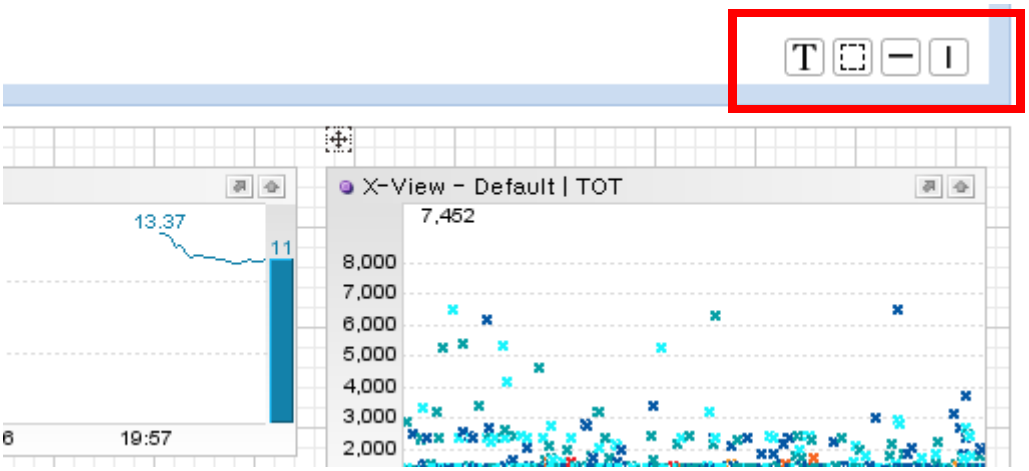
파라미터 옵션	설명
COLORS	각 라인 혹은 막대의 색상을 RGB 값으로 설정한다. 255,0,0/0,255,0/0,0,255 등의 형식으로 설정해야 한다.

BOX BAR 차트의 특정 영역을 클릭한 상태에서 마우스를 좌우로 이동한 후에 놓으면 START_ANGLE이 이동한 크기 만큼 변경된다. 또한 상단 꼭지점을 클릭한 상태에서 마우스를 좌우로 이동한 후에 놓으면 WIDTH_RATE가 이동한 크기 만큼 변경된다.

7.3.6. 텍스트, 박스, 선의 사용

7.3.6.1. 텍스트

그림 7-47: 텍스트, 박스, 선의 사용



Notice: 차트는 자바 애플릿으로 구현되어 있고 텍스트, 박스, 선 등은 HTML 태그로 구현되어 있는데, 기술적인 한계로 HTML 태그는 자바 애플릿 차트 위로 올라갈 수 없다. 따라서 텍스트, 박스, 선 등과 차트가 겹쳐지면 텍스트, 선, 박스 등의 위치를 변경하거나 크기를 변경할 때 사용하는 아이콘이 다른 자바 애플릿 차트에 가려져서 보이지 않을 수 있다. 따라서 이를 고려해서 사용자 정의 대시보드를 구성하도록 한다.

사용자 정의 대시보드에 텍스트를 추가하려면 차트 선택 영역 오른쪽 하단에 있는 텍스트 아이콘을 클릭한다. 그리고 텍스트를 사용자 정의 대시보드에 추가하는 방법은 일반 차트와 동일하다.

설정이 가능한 필드 옵션은 다음과 같다.

표 7-45: 텍스트 필드 옵션

필드 옵션	설명
텍스트	텍스트를 입력한다.
유형	title01, title02 등을 설정할 수 있다. 이 경우에는 정해진 양식으로 텍스트가 나타나며 아래 옵션들은 의미가 없다.
텍스트 정렬	텍스트의 좌우 정렬 방법을 설정한다.
글자 크기	텍스트의 글자 크기를 설정한다.
굵게	텍스트를 진하게 표시할지 여부를 설정한다.
글자 색	텍스트의 색상을 설정한다.
선 색	텍스트 테두리 선의 색상을 설정한다.
선 두께	텍스트 테두리 선의 굵기를 설정한다.
배경 색	텍스트의 배경 색을 설정한다.

7.3.6.2. 박스

사용자 정의 대시보드에 박스를 추가하려면 차트 선택 영역 오른쪽 하단에 있는 박스 아이콘을 클릭한다. 그리고 박스를 사용자 정의 대시보드에 추가하는 방법은 일반 차트와 동일하다.

설정이 가능한 필드 옵션은 다음과 같다.

표 7-46: 박스 필드 옵션

필드 옵션	설명
선 색	박스 테두리 선의 색상을 설정한다.
선 두께	박스 테두리 선의 굵기를 설정한다.
배경 색	박스의 배경 색을 설정한다.

7.3.6.3. 수평선

사용자 정의 대시보드에 수평선을 추가하려면 차트 선택 영역 오른쪽 하단에 있는 수평선 아이콘을 클릭한다. 그리고 수평선을 사용자 정의 대시보드에 추가하는 방법은 일반 차트와 동일하다.

설정이 가능한 필드 옵션은 다음과 같다.

표 7-47: 수평선 필드 옵션

필드 옵션	설명
선 색	수평선의 색상을 설정한다.
선 두께	수평선의 굵기를 설정한다.

7.3.6.4. 수직선

사용자 정의 대시보드에 수직선을 추가하려면 차트 선택 영역 오른쪽 하단에 있는 수직선 아이콘을 클릭한다. 그리고 수직선을 사용자 정의 대시보드에 추가하는 방법은 일반 차트와 동일하다.

설정이 가능한 필드 옵션은 다음과 같다.

표 7-48: 수직선 필드 옵션

필드 옵션	설명
선 색	수직선의 색상을 설정한다.
선 두깨	수직선의 굵기를 설정한다.

7.3.7. Import와 Export

사용자 정의 대시보드의 구성 내용을 XML 파일로 Export하고 Import할 수 있다.

사용자 정의 대시보드의 구성 내용을 XML 파일로 Export하려면 사용자 정의 대시보드 편집 화면 하단 오른쪽에 있는 **[Export]** 버튼을 클릭한다. XML 파일은 UTF-8로 인코딩되어 있고 구조는 다음과 같다.

최상위 태그는 charts 이고 charts 태그에는 제니퍼 서버 버전을 나타내는 version 속성과 Export를 한 시간을 나타내는 exportDate 속성이 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<charts version='4.0' exportDate='20080716/214809'>


</charts>
```

charts 태그는 개별 차트를 의미하는 여러 개의 하위 chart 태그를 가지고 있다.

```
<chart chartId='user.recent'
       left='358' top='9' width='300' height='80'>
</chart>
```

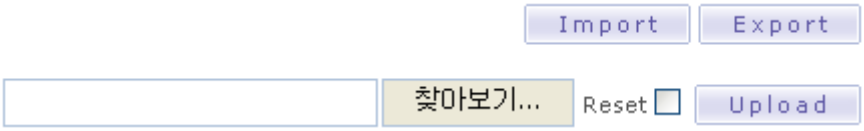
chart 태그의 속성의 의미는 다음과 같다.

- chartId - 차트 유형
- left - 차트의 X 축 좌표로 0에 가까울수록 차트가 왼쪽에 위치한다.
- top - 차트의 Y 축 좌표로 0에 가까울수록 차트가 위쪽에 위치한다.
- width - 차트의 넓이
- height - 차트의 높이
- chart 태그는 하위 config 태그를 통해서 개별 차트에 대해서 설정한 옵션을 기술한다. 이 때 (|)을 구분자로 사용하여 여러 개의 옵션을 구분한다

```
<chart>
    <config>SERVER=ALL_EXCEPT_TOT|MARGIN_LEFT=32|TITLE=메모리</config>
</chart>
```

XML 파일을 Import하려면 사용자 정의 대시보드 편집 화면 하단에 있는 [Import] 버튼을 누르면 나타나는 업로드 폼을 사용한다.

그림 7-48: Import 화면



Notice: 초기화 체크 박스를 선택하면 사용자 정의 대시보드 화면에 있는 기존 차트를 모두 삭제하고 XML 파일에 설정된 차트를 신규로 생성한다. 초기화 체크 박스를 선택하지 않으면 사용자 정의 대시보드 화면에 있는 기존 차트를 삭제하지 않고 XML 파일에 설정된 차트를 신규로 생성한다.

7.3.8. 배경 이미지 설정

사용자 정의 대시보드에 배경 이미지를 설정하려면 사용자 정의 대시보드 편집 화면 하단 오른쪽에 있는 **[Config]** 버튼을 누르면 나타나는 업로드 폼을 사용한다. 초기화 체크 박스를 선택한 후에 저장을 하면 배경 이미지가 제거된다.

7.3.9. 차트 위치와 크기 조절

사용자 정의 대시보드 화면에서 개별 차트의 위치를 이동하고 크기를 조절할 수도 있다. 우선 사용자 정의 대시보드 화면 오른쪽 하단에 있는 [변경] 버튼을 클릭하면 팝업 창이 나타난다.

Notice: 이 작업은 사용자 정의 대시보드 편집 화면이 아닌 사용자 정의 대시보드 화면에서 수행해야 한다.

그림 7-49: 차트 위치와 크기 변경 팝업 창

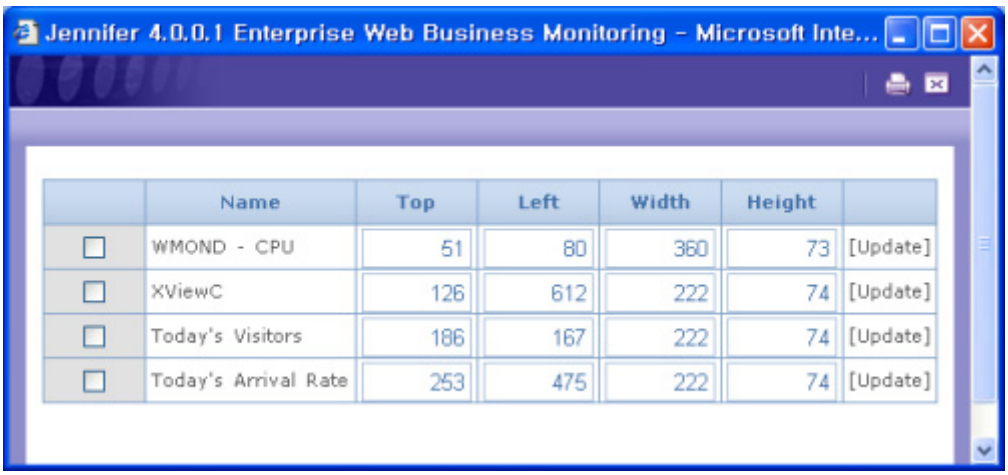


차트 위치와 크기 변경 팝업 창에는 사용자 정의 대시보드를 구성하는 모든 차트와 텍스트, 박스, 선 등의 위치와 크기 정보가 테이블에 표시된다. 특정 차트의 위치나 크기 정보를 변경한 후에, [수정] 버튼을 클릭하거나 엔터 키를 누르면 변경 사항이 반영된다.

표 7-49: 차트 위치와 크기 조절 필드

필드	설명
Top	차트의 Y 축 좌표로 0에 가까울수록 차트가 위쪽에 위치한다.
Left	차트의 X 축 좌표로 0에 가까울수록 차트가 왼쪽에 위치한다.
Width	차트의 넓이
Height	차트의 높이

차트 위치와 크기 조절 팝업 창에서 특정 차트에 대한 체크 박스를 선택하면 사용자 정의 대시보드 화면에 해당 차트가 점선으로 표시된다.

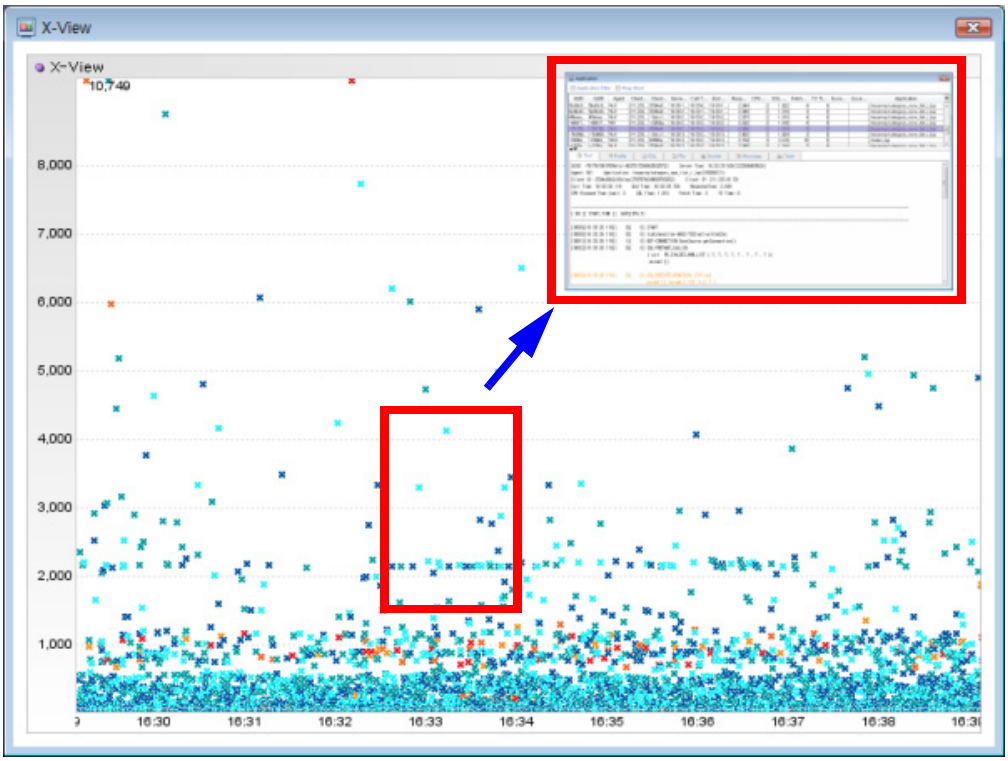
8

X-View와 프로파일링

8.1. 응답 시간 분포도와 X-View

응답 시간 분포도란 개별 트랜잭션의 종료 시간을 X 축으로, 응답 시간을 Y 축으로 한 점 차트를 말한다. 각 점은 트랜잭션을 의미하고 이 점을 선택해서 해당 트랜잭션에 대한 상세 수행 내역, 즉 프로파일을 조회할 수 있다. 제니퍼에서 응답시간 분포도를 X-View라고 부른다.

그림 8-1: 응답 시간 분포도 예



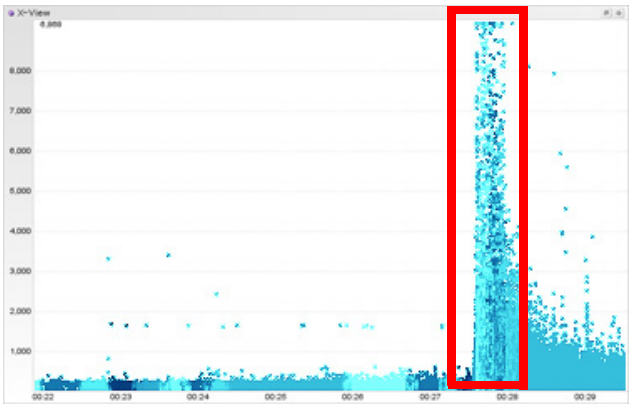
8.1.1. X-View 분포 패턴

성능 문제들은 비정상적인 응답 패턴으로 표출되는 경우가 있는데 그 중에서 알아두어야 할 X-View 분포 패턴들에 대해서 설명한다.

8.1.1.1. 단순 폭주 현상

단순 폭주 현상은 이벤트 행사 등의 원인으로 유명 상품 티켓 시스템에서 순간적으로 사용자가 증가할 때 관찰할 수 있는 패턴이다.

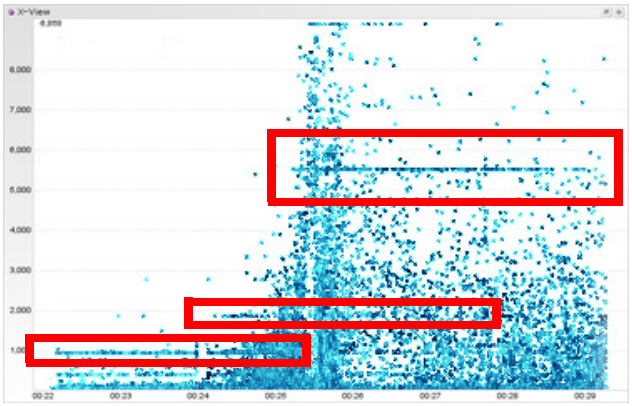
그림 8-2: 단순 폭주 현상



8.1.1.2. 시루떡 현상

시루떡 현상은 서비스 요청이 폭주한 후에 시스템이 정상화되지 않을 때 관찰할 수 있는 패턴이다. 단순 폭주 현상과는 폭주 후 응답 시간 분포도 패턴이 다르다. 차트 중간에 가로 선이 보이는데, 최초 폭주 직전부터 조금씩 가로 선의 개수가 많아지다가 폭주 후에는 확연하게 가로 선이 형성되는 것을 볼 수 있다. 이런 층 모양이 시루떡과 비슷해서 이 패턴을 시루떡 현상이라고 한다. 시루떡 현상에서는 선 여러 개가 계층적으로 형성되며 부하량이 많아질수록 가로 선 수가 늘어나는 특징이 있다.

그림 8-3: 시루떡 현상



Notice: 단순하게 한 두개의 가로 선이 있는 경우에는 시루떡 현상이라고 부르지 않는다.

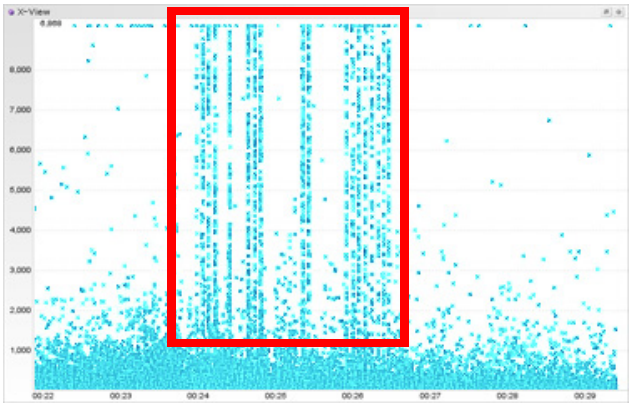
시루떡 현상은 트랜잭션이 특정 자원을 사용하는 과정에서 재시도와 기다림이 반복적으로 발생할 때 나타난다. 즉, 트랜잭션이 어떤 자원을 획득하고자 하는데 유효한 자원이 없어 일정 시간 기다리다가 다시 시도할 때 응답 시간의 다단계 층이 형성된다. 따라서 부하

량이 많아질수록 자원 부족 현상이 빈번해지고 기다리는 시간만큼을 간격으로 가로 선이 형성된다.

8.1.1.3. 폭포수 현상

폭포수 현상은 한계 상황에 민감하게 반응하는 자원이 존재할 때 관찰할 수 있는 패턴이다. 폭포수 현상에서는 반복적인 세로 선이 형성되는데, 특정 자원이 갑자기 한계 상황에 도달했다가 순간적으로 풀리면서 관련된 트랜잭션이 동시에 종료되는 현상이 반복될 때 발생할 수 있다.

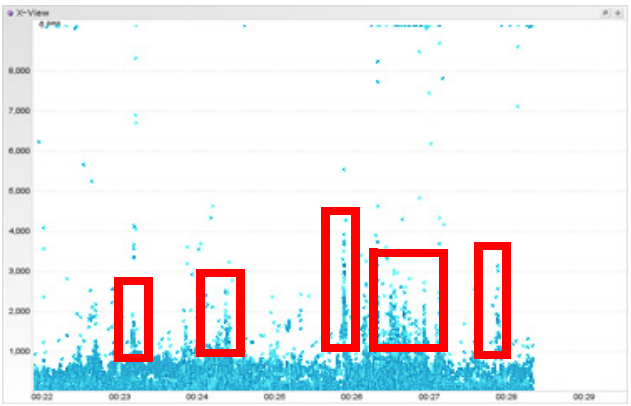
그림 8-4: 폭포수 현상



8.1.1.4. 물방울 현상

물방울 현상은 트랜잭션이 비 오는 날 바닥에서 물방울이 튀기듯이 형성되는 패턴을 지칭한다. 물방울 현상에서는 울퉁불퉁하고 짧은 세로 선이 여러 개 형성된다. 물방울 현상은 시스템에 병목이 존재하지만 부하량이 적어서 시스템 병목이 크게 표출되지 않을 때 발생할 수 있다. 따라서 부하량이 많아지면 해당 병목은 치명적인 시스템 장애 원인이 될 수 있다.

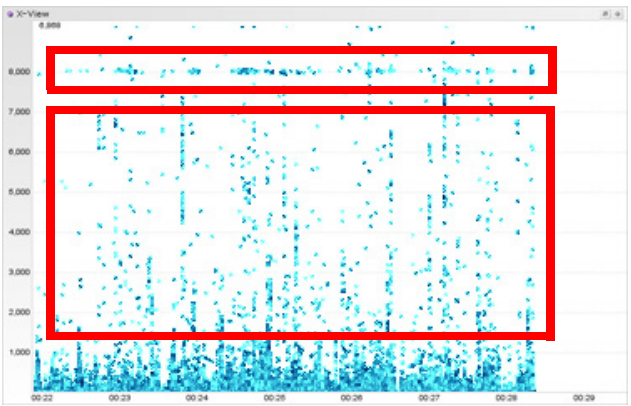
그림 8-5: 물방울 현상



8.1.1.5. 매트릭스 현상

매트릭스 현상은 영화 매트릭스 시작 화면처럼 점들 흘러 내리듯이 형성되는 패턴을 말한다. 매트릭스 현상은 짧고 빈번한 락 현상이 많은 트랜잭션에 걸쳐서 나타날 때 관찰된다. 하나의 락은 몇 개의 트랜잭션에만 영향을 미치고 이러한 락이 전체 애플리케이션 서비스에 불특정하게 나타나는 것이 특징이다. 예를 들어, 데이터베이스나 EJB 엔티티빈의 ISOLATION LEVEL이 높을 때 이 현상이 발생할 수 있다.

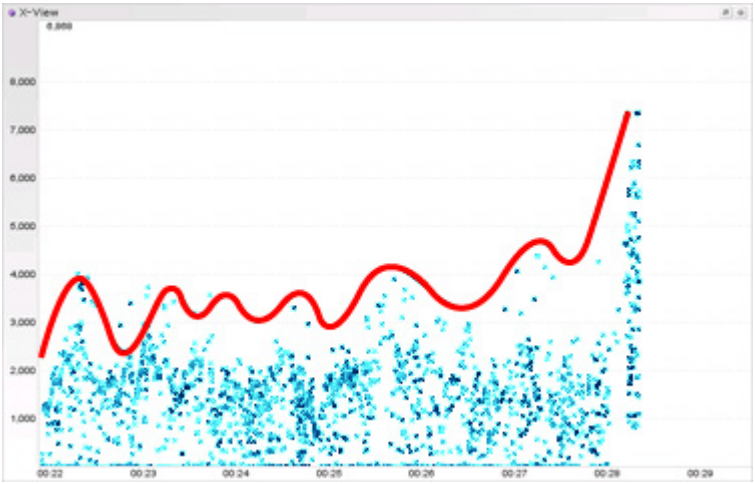
그림 8-6: 매트릭스 현상



8.1.1.6. 파도치기 현상

파도치기 현상은 트랜잭션이 물결처럼 출렁이듯이 형성되는 패턴을 말한다. 성능 문제와는 상관없이 서비스 요청이 많아졌다가 줄어들었다를 반복할 때 파도치기 현상이 발생할 수 있다. 그리고 트랜잭션이 사용하는 특정 자원이 부족할 때 발생할 가능성이 높다. 예를 들어, 모니터링하는 애플리케이션과 연동하는 외부 시스템의 CPU 사용률이 100%에 육박할 때 파도치기 현상이 발생할 수 있다.

그림 8-7: 파도치기 현상



8.2. X-View 데이터

8.2.1. X-View 데이터 구성

제니퍼 서버가 트랜잭션 처리 결과에 대해서 수집하는 데이터를 X-View 데이터라고 하고 , X-View 데이터는 트랜잭션 데이터와 프로파일 데이터로 이루어진다.

- 트랜잭션 데이터는 애플리케이션 이름과 호출 시간, 경과 시간 등으로 이루어진 데이터이다.
- 프로파일 데이터는 트랜잭션이 수행되는 과정에서 어떤 메소드를 호출했고, 어떤 SQL을 수행했고, 어떤 파일 혹은 소켓을 사용했는지를 순차적으로 보여주는 데이터이다.

트랜잭션 데이터는 다음과 같이 구성된다.

표 8-1: 트랜잭션 데이터

항목	설명
도메인	제니퍼 서버 도메인 이름
UUID	트랜잭션 고유 아이디
GUID	여러 트랜잭션을 하나의 그룹으로 묶을 때 사용하는 글로벌 키
에이전트	트랜잭션이 수행된 자바 애플리케이션의 제니퍼 에이전트 아이디
클라이언트 IP	트랜잭션을 요청한 클라이언트 컴퓨터의 IP 주소

표 8-1: 트랜잭션 데이터

항목	설명
클라이언트 아이디	HTTP 쿠키로 관리하는 클라이언트 고유 아이디
사용자 아이디	업무적인 의미를 갖는 사용자 아이디로 별도 설정이 필요하다.
서버 시간	트랜잭션 데이터를 제니퍼 서버가 수집한 시간으로 제니퍼 서버 시간을 기준으로 한다.
호출 시간	트랜잭션 시작 시간으로 제니퍼 에이전트를 설치한 자바 애플리케이션 시간을 기준으로 한다.
종료 시간	트랜잭션 종료 시간으로 제니퍼 에이전트를 설치한 자바 애플리케이션 시간을 기준으로 한다.
응답 시간	트랜잭션 종료 시간에서 트랜잭션 호출 시간을 뺀 것으로 트랜잭션이 수행되는데 걸린 시간을 의미한다. 단위는 밀리 세컨드이다.
CPU 시간	트랜잭션이 수행되는 과정에서 사용한 CPU 시간을 의미한다. 단위는 밀리 세컨드이다.
SQL 시간	트랜잭션이 수행되는 과정에서 SQL을 수행하는데 걸린 시간을 의미한다. 단위는 밀리 세컨드이다.
Fetch 시간	트랜잭션이 수행되는 과정에서 java.sql.ResultSet 객체의 next 메소드를 수행하는데 걸린 시간을 의미한다. 단위는 밀리 세컨드이다.
TX 시간	트랜잭션이 수행되는 과정에서 외부 트랜잭션을 수행하는데 걸린 시간을 의미한다. 단위는 밀리 세컨드이다.
클라이언트 응답 시간	웹 브라우저와 자바 애플리케이션 사이의 네트워크 시간으로 별도의 설정이 필요하다.
에러 항목	트랜잭션이 수행되는 과정에서 발생한 예외 유형(심각, 에러, 경고)
에러	트랜잭션이 수행되는 과정에서 발생한 예외 이름
애플리케이션	해당 트랜잭션의 애플리케이션 이름

프로파일 데이터는 여러 개 항목으로 구성된다. 프로파일 항목은 메소드 수행과 관련된 메소드 항목과 트랜잭션 처리 중에 발생한 특정 이벤트를 의미하는 메시지 항목으로 구성된다. 그리고 메시지 항목은 SQL, 파일, 소켓 등의 세부 항목으로 다시 구분된다.

표 8-2: 프로파일 데이터 항목

항목	설명
메소드 항목	트랜잭션 처리 과정에서 수행된 메소드를 의미하며 프로파일 데이터 대부분을 차지한다.
SQL 항목	트랜잭션 처리 과정에서 수행된 SQL 작업을 의미한다.
파일 항목	트랜잭션 처리 과정에서 수행된 파일 작업을 의미한다.
소켓 항목	트랜잭션 처리 과정에서 수행된 소켓 작업을 의미한다.

표 8-2: 프로파일 데이터 항목

항목	설명
메시지 항목	SQL, 파일, 소켓 항목을 제외한 메시지를 의미한다.

주요 메시지 항목 내용은 다음과 같다.

표 8-3: 주요 메시지 항목

메시지	설명
FILE-WOPEN 파일명	쓰기 모드로 파일을 오픈
FILE-ROPEN 파일명	읽기 모드로 파일을 오픈
SOCKET-ISTREAM 주소	소켓에서 읽기 스트림 오픈
SOCKET-OSTREAM 주소	소켓에서 쓰기 스트림 오픈
[cpu] GET-CONNECTION 메소드 혹은 데이타소스 [시간ms]	트랜잭션이 데이터베이스 커넥션을 커넥션 풀에서 획득하거나 새로운 연결을 맺음
[cpu] CLOSE-CONNECTION	데이터베이스 커넥션을 커넥션 풀에 반환하거나 CLOSE 함
[cpu] FETCH 패치 건수/누적 패치건수 [시간ms]	데이터베이스에서 데이터를 FETCH. FETCH 시간은 Gap 시간에 포함되고 FETCH 항목 CPU 누적 시간은 FETCH시 소요한 CPU 시간을 포함한다.
[cpu] TX-CALL 외부 트랜잭션 이름 [시간ms]	외부 트랜잭션 수행
THREAD-INIT 쓰레드명	새로운 서브 쓰레드 시작
PARAM 값	메소드 파라미터 값
RETURN 값	메소드 반환 값
[cpu] SQL-PREPARE_STMT*{n}*	PreparedStatement 객체 생성
[cpu] SQL-PREPARE_CALL*{n}*	CallableStatement 객체 생성
[cpu] SQL-EXECUTE-QUERY*{n}* [시간ms]	{n}의 executeQuery 메소드가 수행됨
[cpu] SQL-EXECUTE-UPDATE*{n}* [시간ms]	{n}의 executeUpdate 메소드가 수행됨
[cpu] SQL-EXECUTE*{n}* [시간ms]	{n}의 execute 메소드가 수행됨
[cpu] SQL-EXECUTE-QUERY [시간ms]	executeQuery(SQL) 메소드가 수행됨
[cpu] SQL-EXECUTE-UPDATE [시간ms]	executeUpdate(SQL) 메소드가 수행됨
[cpu] SQL-EXECUTE [시간ms]	(Prepared/Callable)Statement.execute(SQL) 혹은 excuteBatch 가호출됨

- 메시지 앞에 [cpu]가 표시된 경우에는 해당 메시지가 출력될 때 트랜잭션 누적 CPU가 출력된다.
- 메시지 뒤에 [시간ms]가 표시된 경우에는 해당 로직이 수행되는데 걸린 소요 시간이 출력된다.
- SQL 관련 예약어에서 {n}은 트랜잭션 내에서 JDBC 객체 구분자를 의미한다. PreparedStatement 객체가 여러 개 동시에 생성되어 사용될 수 있는데 이때 실제 execute가 어떤 객체에 해당 하는지를 알려준다.

8.2.2. X-View 데이터 전송

제니퍼 서버는 server_udp_runtime_port 옵션으로 설정한 포트로 제니퍼 에이전트가 보내는 트랜잭션 데이터를 수집하고, server_udp_lwst_call_stack_port 옵션으로 설정한 포트로 프로파일 데이터를 수집한다.

프로파일 데이터 크기는 프로파일 설정에 영향을 받는다. 그런데 UDP 방식으로 전송할 수 있는 데이터 크기가 64 KB를 넘을 수 없기 때문에 제니퍼 에이전트는 64 KB 보다 큰 프로파일 데이터에 대해서는 여러 개의 패킷으로 분리해서 전송한다. 패킷 크기는 제니퍼 에이전트의 xview_profile_udp_packet_size 옵션으로 설정하는데 기본 값은 32757이고 단위는 바이트이다.

```
xview_profile_udp_packet_size = 32757
```

Notice: 만약 데이터 크기가 제니퍼 에이전트의 xview_profile_udp_packet_size 옵션으로 지정한 크기를 초과하면 제니퍼 에이전트는 해당 데이터를 분할해서 제니퍼 서버에 전송한다. 따라서 xview_profile_udp_packet_size 옵션을 작게 설정하면 제니퍼 에이전트가 제니퍼 서버에 전송하는 패킷 개수가 많아지기 때문에 제니퍼 서버 성능 향상을 위해서 xview_profile_udp_packet_size 옵션을 가능한 크게 설정하는 것을 권장한다. 그리고 프로파일 데이터에 프로토콜 해더가 별도로 추가되므로 xview_profile_udp_packet_size 옵션은 최대 60000까지만 지정하도록 한다.

제니퍼 서버는 제니퍼 에이전트가 패킷으로 분리해서 전송한 프로파일 데이터를 임시 큐에 저장한 후에 모든 패킷이 도착하면 이를 결합한다. 임시 큐 크기는 제니퍼 서버의 xview_profile_multi_packet_queue_size 옵션을 통해서 설정한다. 기본 값은 303이다.

```
xview_profile_multi_packet_queue_size = 303
```

제니퍼 서버는 임계치를 초과해도 모든 패킷이 도착하지 않으면 임시 큐에 저장된 패킷만으로 프로파일 데이터를 구성하고 TIME OUT을 발생시킨다. TIME OUT이 일어나면 프로파일 데이터의 일부분이 유실되게 된다. 제니퍼 서버의

xview_profile_multi_packet_time_out 옵션을 통해서 임계치를 설정한다. 기본 값은 1000이고 단위는 밀리 세컨드이다.

```
xview_profile_multi_packet_time_out = 1000
```

제니퍼 에이전트는 트랜잭션의 응답 시간이 임계치 미만이면 해당 트랜잭션에 대한 프로파일 데이터를 제니퍼 서버에 전송하지 않는다. 제니퍼 에이전트의 xview_profile_ignore_resp_time 옵션으로 임계치를 설정한다. 기본 값은 0이고 단위는 밀리 세컨드이다.

```
xview_profile_ignore_resp_time = 0
```

Notice: xview_profile_ignore_resp_time 옵션은 제니퍼 에이전트와 제니퍼 서버에서 모두 사용할 수 있다. 에이전트에서는 서버로 전송할 기준 시간을 설정하기 위해 사용하고, 서버에서는 저장을 위한 기준 시간을 설정하기 위해 사용한다.

네크워크 설정과 관련한 자세한 사항은 네트워크 설정과 제니퍼서버 네트워크 구성을 참조한다.

Notice: UDP로 전송할 수 있는 데이터 크기(UDP Send Buffer Size)는 64 KB가 최대 값인데 선 마이크로시스템즈 솔라리스와 HP HP-UX 운영 체계는 64 KB가 기본 값이지만 IBM AIX 운영 체계는 64 KB보다 작은 수치가 기본 값이다. 제니퍼 에이전트가 UDP로 전송하는 데이터 크기가 UDP로 전송할 수 있는 데이터 크기보다 크면 데이터는 전송되지 않고 유실된다. 만약 운영 체계에 설정된 UDP 전송 데이터 크기를 넘는 경우에는 에러 메시지가 로그 파일에 기록되지만 제니퍼 에이전트와 제니퍼 서버 사이에 존재하는 네트워크 장비 버퍼 사이즈를 초과하는 경우에는 에러 메시지 기록 없이 패킷이 유실된다.

8.2.3. 대용량 프로파일 데이터를 안정적으로 수집하기

용량이 큰 프로파일 데이터를 여러 개의 패킷으로 분리해서 전송해도 UDP 특성으로 인한 손실은 불가피하다. 프로파일 데이터 크기가 작아도 UDP 특성으로 인한 손실은 발생할 수 있다.

그래서 제니퍼는 프로파일 데이터 크기나 트랜잭션 응답 시간을 기준으로 UDP 전송 손실을 방지할 수 있는 방법을 제공한다. 기준에 부합하는 프로파일 데이터를 제니퍼 서버로 전송하지 않고 제니퍼 에이전트를 설치한 하드웨어에 저장한 후에 요청 시 역방향 호출로 프로파일 데이터를 가져온다.

이 기능을 사용하려면 AgentDB를 사용해야 한다.

제니퍼 에이전트의 agent_db_enabled 옵션으로 AgentDB 사용 여부를 설정한다. 이 옵션을 true로 설정한다.

```
agent_db_enabled = true
```

AgentDB는 데이터를 비동기 방식으로 저장하는데, 제니퍼 에이전트의 agent_db_max_queue 옵션으로 저장 대기 큐 크기를 설정한다. 저장 대기 큐가 꽉 찬 경우에는 프로파일 데이터를 AgentDB에 저장하지 않고, 이전과 동일하게 제니퍼 서버에 전송한다. 그러나 기본 값이 10인 저장 대기 큐가 꽉 차는 경우는 많지 않고, 큐 크기를 크게 하는 경우에 메모리 사용량이 증가할 수 있기 때문에 이 옵션을 수정하지 않는 것을 권장한다.

```
agent_db_max_queue = 10
```

제니퍼 에이전트의 agent_db_rootpath 옵션으로 AgentDB가 사용하는 디렉토리 위치를 설정한다. 기본 값은 현재 디렉토리이다.

```
agent_db_rootpath = .
```

제니퍼 에이전트는 agent_db_rootpath 옵션으로 설정한 위치에 .db라는 디렉토리를 만들고 일자별로 하위 디렉토리를 만들어서 데이터를 저장한다. 이 옵션으로 설정한 디렉토리를 자동으로 생성하지 않기 때문에 임의의 디렉토리를 지정한 경우에는 해당 디렉토리가 존재해야 한다. 그리고 쓰기 권한도 있어야 한다. 여러 제니퍼 에이전트가 AgentDB 저장 디렉토리로 동일한 디렉토리를 사용해서는 안된다. 일시적으로 제니퍼 에이전트 아이디가 동일하면 파일이 훼손될 수 있다. 그리고 이 값을 변경하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다. 디렉토리를 변경한 경우에 이전 파일을 변경한 디렉토리로 단순하게 복사하면 과거 데이터를 계속 사용할 수 있다. 단, 자바 애플리케이션을 정지한 후에 복사를 해야 한다.

제니퍼 에이전트의 xview_profile_dump_entry 옵션으로 AgentDB에 저장할 프로파일 데이터 크기를 설정한다. 이 옵션으로 지정한 값보다 크기가 큰 프로파일 데이터를 AgentDB에 저장한다. 기본 값은 Integer 최대 값이고 단위는 바이트이다.

```
xview_profile_dump_entry = 10240000
```

제니퍼 에이전트의 xview_profile_dump_elapsed 옵션으로 AgentDB에 저장할 프로파일 데이터의 트랜잭션 응답 시간을 설정한다. 기본 값은 Integer 최대 값이고 단위는 밀리 세컨드이다. .

```
xview_profile_dump_elapsed = 300000
```

예를 들어, CPU를 많이 사용한 트랜잭션이 5분 이상 Hang 상태에 있고, 액티브 서비스 상세 화면에서 액티브 프로파일 데이터 크기가 큰 것을 확인하였다. 문제를 해결하기 위해서는 해당 트랜잭션에 대한 프로파일 데이터를 수집해야 하는데, UDP 전송에 의존하면

데이터가 유실될 수 있다. 이런 경우에 xview_profile_dump_elapsed 옵션을 300000으로 설정하면 해당 트랜잭션 프로파일 데이터가 AgentDB에 저장되기 때문에 안정적으로 확인할 수 있다.

Notice: xview_profile_dump_elapsed과 xview_profile_dump_entry 옵션은 OR 조건이다.

제니퍼 에이전트의 log_xview_profile_dump 옵션은 프로파일 데이터가 AgentDB에 저장되고 있는지를 확인하기 위한 디버그 옵션이다. 기본 값은 false이다.

```
log_xview_profile_dump = true
```

프로파일 데이터 항목 개수는 profile_buffer_size 옵션으로 설정한 개수를 초과할 수 없다. 따라서 항목이 많은 프로파일 데이터를 수집하려면 이 옵션을 충분하게 큰 값으로 설정해야 한다. 자세한 사항은 [프로파일링 제어]를 참조한다.

8.2.4. X-View 데이터 저장

제니퍼 서버는 X-View 데이터를 파일에 저장한다. 제니퍼 서버의 data_directory 옵션으로 X-View 데이터가 저장되는 디렉토리를 설정한다.

```
data_directory = ../../data/file/
```

데이터 파일 손실을 방지하기 위해서 이 옵션을 수정하려면 제니퍼 서버를 정지한 후에 설정 파일에서 직접 수정해야 한다.

8.2.4.1. 트랜잭션 데이터

제니퍼 서버는 최근에 수집한 트랜잭션 데이터를 메모리 큐에 담아놓고, 제니퍼 클라이언트는 2초 주기로 트랜잭션 데이터를 이 메모리 큐에서 읽어간다. 따라서 이 메모리 큐 크기는 최대 서비스 요청률의 2배 정도로 설정해야 한다. 제니퍼 서버의 xview_server_queue_size 옵션으로 이 큐 크기를 설정하는데 기본 값은 512이다. 이 옵션을 변경하면 제니퍼 서버를 재시작하여야 한다.

```
xview_server_queue_size = 512
```

제니퍼 서버의 xview_point_ignore_resp_time 옵션으로 설정한 값보다 응답 시간이 작은 트랜잭션 데이터는 큐에 담지 않고 저장도 하지 않는다. 기본 값은 0이고 단위는 밀리 세컨드이다.

```
xview_point_ignore_resp_time = 0
```

8.2.4.2. 프로파일 데이터

제니퍼 서버는 제니퍼 에이전트로부터 수집한 프로파일 데이터를 임시적으로 메모리 큐에 담아 놓은 후에 0.5초를 주기로 파일에 저장한다. 이 메모리 큐 크기는 제니퍼 서버의 xview_profile_cache_queue_size 옵션으로 설정하는데 프로파일 데이터가 파일에 저장되기 전에 이 큐가 전부 차면 프로파일 데이터의 유실이 발생할 수 있다. 따라서 저장 주기가 0.5초이기 때문에 최대 서비스 요청률의 50% 정도로 xview_profile_cache_queue_size 옵션을 설정해야 한다. 기본 값은 512이다. 이 옵션을 변경하면 제니퍼 서버를 재시작하여야 한다.

```
xview_profile_cache_queue_size = 512
```

제니퍼 서버는 프로파일 데이터를 실시간 X-View 프로파일 파일과 일자별 X-View 프로파일 파일에 이중으로 저장한다. 실시간 X-View 프로파일 파일에는 최근 수행된 트랜잭션에 대한 프로파일 데이터가 순환적으로 저장되는데, 그 기간은 실시간 X-View 프로파일 파일 크기와 서비스 요청률에 영향을 받는다. 일자별 X-View 프로파일 파일은 과거 프로파일 데이터 조회 목적을 위한 것이다.

실시간 X-View 프로파일 파일 크기는 제니퍼 서버의 xview_profile_isam_file_max_size 옵션으로 설정한다. 기본 값은 512mb이다. 이 옵션을 변경하면 제니퍼 서버를 재시작하여야 한다.

```
xview_profile_isam_file_max_size = 10mb
xview_profile_isam_file_max_size = 512mb
xview_profile_isam_file_max_size = 1gb
```

이 옵션은 mb(메가 바이트) 혹은 gb(기가 파이트) 등의 단위와 함께 설정한다. 제니퍼 서버를 처음 실행할 때 이 옵션으로 설정한 크기의 파일이 만들어진다.

제니퍼 서버의 xview_profile_ignore_resp_time 옵션으로 설정한 값보다 응답 시간이 작은 프로파일 데이터는 실시간 X-View 프로파일 파일에 저장되지 않는다. 기본 값은 0이고 단위는 밀리 세컨드이다.

```
xview_profile_ignore_resp_time = 100
```

이 옵션을 제니퍼 서버에 설정하는 것보다 제니퍼 에이전트에 설정하는 것이 효율적이다. 제니퍼 에이전트에 설정하면 프로파일 데이터가 네트워크로 전송도 되지 않기 때문이다. 실시간 X-View 프로파일 파일에는 프로파일 데이터를 저장하지만, 조건에 따라서 일자별 X-View 프로파일 파일에는 프로파일 데이터를 저장하지 않도록 설정할 수 있다. 첫번째로 제니퍼 서버의 xvdaily_ignore_resp_time 옵션으로 설정한 값보다 응답 시간이 작은 프로파일 데이터는 일자별 X-View 프로파일 파일에 저장되지 않는다. 기본 값은 500이고 단위는 밀리 세컨드이다.

```
xvdaily_ignore_resp_time = 500
```

두번째로 제니퍼 에이전트 별로 프로파일 데이터 저장 여부를 설정할 수 있다. 예를 들어, 클러스터링으로 구성된 동일한 애플리케이션들을 모니터링할 때 대표 제니퍼 에이전트에 대해서만 프로파일 데이터를 저장하면 파일 공간을 효율적으로 사용할 수 있다. 이는 제니퍼 서버의 xvdaily_agents 옵션으로 설정한다.

```
xvdaily_agents = W11,W12
```

값을 설정하지 않으면 모든 제니퍼 에이전트에 대한 프로파일 데이터를 일자별 X-View 프로파일 파일에 저장한다.

8.3. X-View 차트와 트랜잭션 리스트

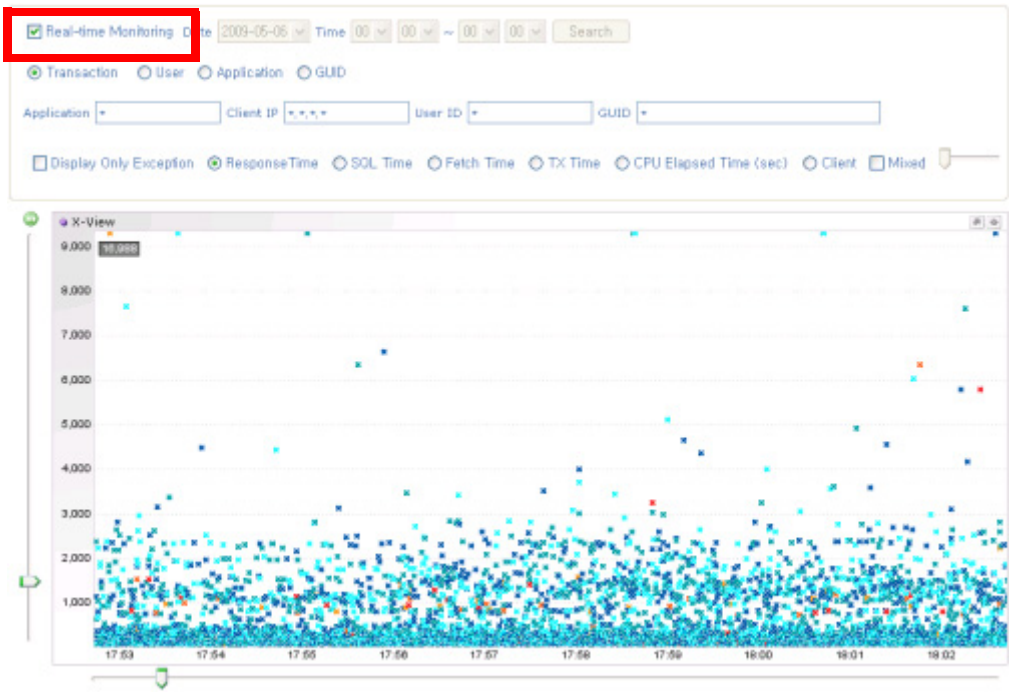
8.3.1. X-View 차트

사용자는 X-View 데이터를 X-View 차트를 통해서 확인할 수 있다. X-View 차트는 **[대시보드 | 제니퍼 대시보드]** 메뉴, **[실시간 모니터링| X-View]** 메뉴, **[통계 분석 | X-View]** 메뉴 등에 존재한다.

- **[대시보드 | 제니퍼 대시보드]** - 대시보드 상에 존재하는 X-View 차트로, 트랜잭션으로 유형이 고정되어 있고 애플리케이션이나 클라이언트 IP 등으로 검색을 할 수 없다. 이 차트는 트랜잭션 처리 현황을 실시간으로 보여준다.
- **[실시간 모니터링 | X-View]** - 트랜잭션 처리 현황을 실시간으로 보여주는 차트로, 다양한 유형을 선택할 수 있고, 애플리케이션이나 클라이언트 IP 등으로 검색을 할 수 있다.
- **[통계 분석 | X-View]** - 사용자가 시간대를 지정해서 트랜잭션 처리 현황을 검색할 수 있는 차트로, 다양한 유형을 선택할 수 있고, 애플리케이션이나 클라이언트 IP 등으로 검색을 할 수 있다. 검색이 가능한 날짜는 CleanActor 스케줄러로 설정한 데이터 보관 기간으로 결정된다.

Notice: [통계 분석 | X-View] 메뉴에서 트랜잭션 처리 현황을 검색할 때 최근에 수행된 트랜잭션 데이터도 검색할 수 있지만 10분이 경과하지 않은 최근에 처리된 트랜잭션에 대해서는 프로파일 데이터가 나타나지 않을 수 있다. 왜냐하면 10분 주기로 프로파일 인덱스를 생성하는데 아직 인덱스가 만들어지지 않은 트랜잭션의 프로파일 데이터는 검색할 수 없기 때문이다.

그림 8-8: X-View 차트



[실시간 모니터링| X-View] 메뉴와 **[통계 분석 | X-View]** 메뉴를 이동하지 않고, 상단에 있는 실시간 모니터링 체크박스를 통해서 실시간 X-View 데이터 조회와 일자별 X-View 데이터 조회를 선택할 수 있다.

8.3.1.1. Y 축

X-View 차트 Y 축은 기본적으로 트랜잭션의 수행 시간을 의미하지만, X-View 유형에 따라서 기준을 다르게 설정할 수 있다. 예를 들어, 트랜잭션 유형에서 SQL 시간이나 Fetch 시간 등으로 그 기준을 변경할 수 있다.

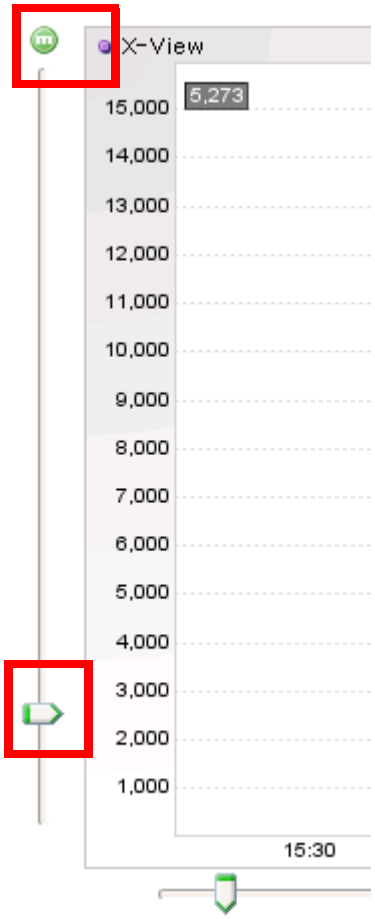
Y 축 전체 구간의 크기는 0초에서 9초까지로 설정되어 있는데 사용자가 전체 구간의 크기를 임의로 변경할 수 있다. 단위는 밀리 세컨드이다.

- 위쪽 방향키 - 위쪽 방향키를 누르면 Y 축 전체 구간 크기가 증가한다. 증가분은 Y 축 전체 구간 현재 크기에 영향을 받는다.
- 아래쪽 방향키 - 아래쪽 방향키를 누르면 Y 축 전체 구간 크기가 감소한다. 감소분은 Y 축 전체 구간 현재 크기에 영향을 받는다.

Shift 키와 함께 누르면 증감하는 크기가 10배가 된다.

Notice: 아래쪽 방향키를 사용하는 경우에도 Y 축 최대 값이 100 밀리 세컨드보다 작을 수는 없다.

그림 8-9: X-View 차트 Y 축 전체 구간 크기 설정



슬라이더를 사용하거나 임의의 값을 설정하여 Y 축 전체 구간 크기를 설정할 수 있다. Y 축 상단에 있는 아이콘을 클릭하면 임의의 값을 입력할 수 있는 필드가 나타난다. 수행 시간이 Y 축 최대 값을 초과하는 모든 트랜잭션은 X-View 차트 상단에 표시된다. 정확한 시간을 파악하려면 위쪽 방향키를 통해서 Y 축 전체 구간 크기를 증가시켜야 한다.

8.3.1.2. X 축

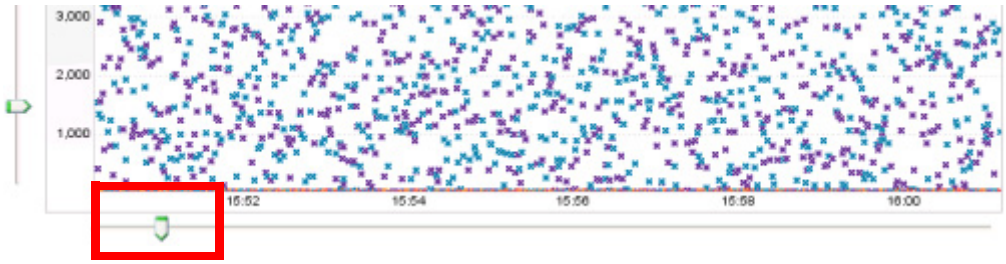
X-View 차트 X 축은 트랜잭션이 수행된 시간을 의미한다. **[통계 분석 | X-View]** 메뉴에서는 사용자가 시간 구간을 지정하여 검색하기 때문에 X 축 구간은 고정된다.

Notice: [통계 분석 | X-View] 메뉴에서 트랜잭션 처리 현황을 검색할 때 시간 구간을 1시간 이하로 지정하는 것을 권장한다. 서비스 처리율이 높은 자바 애플리케이션을 모니터링하는 경우에는 트랜잭션 데이터가 많아서 검색 처리 시간이 길어지고 자바 플러그인 힙 메모리가 부족할 수 있다.

Notice: [대시보드 | 제니퍼 대시보드] 메뉴와 **[실시간 모니터링 | X-View]** 메뉴에서는 X 축 구간은 기본적으로 10분으로 설정되어 있는데 사용자가 구간의 크기를 임으로 변경할 수 있다

- 오른쪽 방향키 - 오른쪽 방향키를 누르면 X 축 구간 크기가 1분 증가한다. 예를 들어, 현재 구간이 10시 10분에서 10시 20분까지인데 오른쪽 방향키를 누르면 구간이 10시 9분에서 10시 20분으로 변경된다. Shift 키와 오른쪽 방향키를 함께 누르면 X 축 구간 크기가 10분 증가한다. 예를 들어, 현재 구간이 10시 10분에서 10시 20분까지인데 Shift 키와 오른쪽 방향키를 함께 누르면 구간이 10시에서 10시 20분으로 변경된다.
- 왼쪽 방향키 - 왼쪽 방향키를 누르면 X 축 구간 크기가 1분 감소한다. 예를 들어, 현재 구간이 10시 9분에서 10시 20분까지인데 왼쪽 방향키를 누르면 구간이 10시 10분에서 10시 20분으로 변경된다. Shift 키와 왼쪽 방향키를 함께 누르면 X 축 구간 크기가 10분 감소한다. 예를 들어, 현재 구간이 10시에서 10시 20분까지인데 Shift 키와 왼쪽 방향키를 함께 누르면 구간이 10시 10분에서 10시 20분으로 변경된다.

그림 8-10: X-View 차트 X 축 전체 구간 크기 설정



슬라이더를 사용하여 X축 전체 구간 크기를 설정할 수 있다.

Notice: X-View 차트에 트랜잭션을 표시할 때 X 축 위치는 제니퍼 서버가 트랜잭션 정보를 수집한 시간을 기준으로 결정된다. 제니퍼 서버가 2개 이상의 자바 애플리케이션을 모니터링 할 수 있는데 각각의 자바 애플리케이션의 시스템 시간이 다를 수 있기 때문에 X-View 차트에 트랜잭션을 표시할 때 X 축 위치로 제니퍼 서버의 시간을 사용하는 것이다.

이전 버전에서는 마이너스 키와 플러스 키를 이용해서 X 축 구간 크기는 고정한 상태에서 구간대만을 변경할 수 있었다. 그러나 지금은 마이너스 키와 플러스 키는 왼쪽 방향키와 오른쪽 방향키와 동일하게 동작한다. 대신 X-View 차트 구간 선택 기능을 통해서 특정 구간에 대한 분석을 수행할 수 있다.

X-View 차트 구간 선택 기능을 사용하는 방법은 다음과 같다.

- X-View 차트에서 구간 시작으로 마우스 커서를 이동한 후에 더블 클릭을 한다. 그러면 그 위치에 점선이 나타난다.
- 다시 구간 끝으로 마우스 커서를 이동한 후에 더블 클릭을 한다. 그러면 그 위치에도 점선이 나타난다.
- 이 상태에서 Enter 키를 누르면 선택한 구간만이 확대되서 나타난다.
- 원래 상태로 돌아가려면 임의의 위치에서 더블 클릭을 하거나, ESC 키를 누른다.

8.3.2. X-View 차트 유형

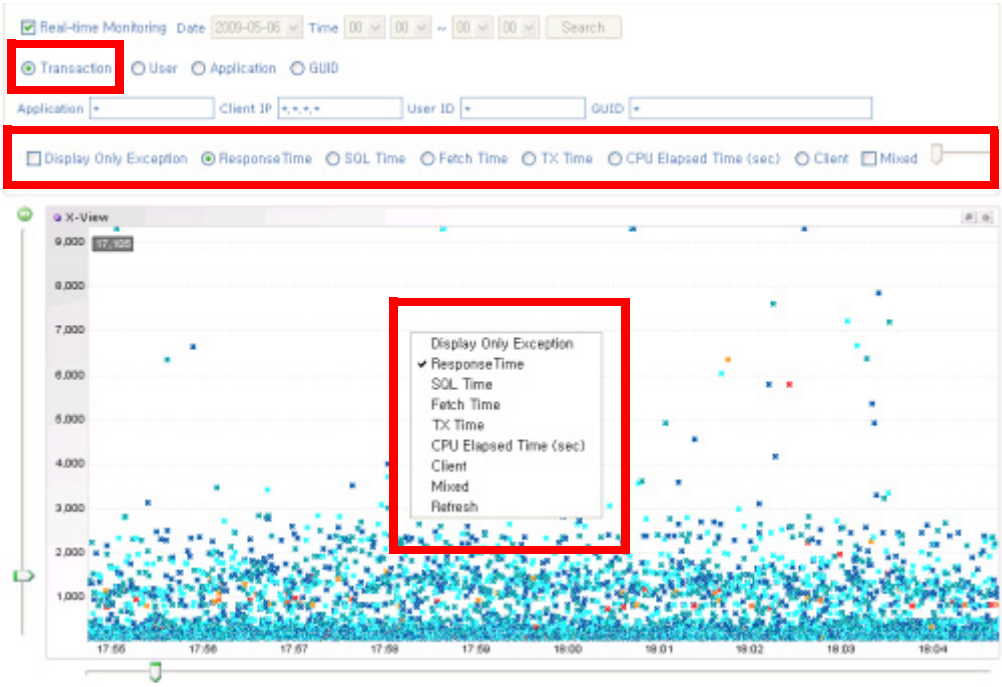
X-View 차트 유형은 트랜잭션을 표시하는 방식을 의미한다. 예를 들어, 트랜잭션 유형은 개별 트랜잭션을 X-View 차트에 모두 표시하는 방식이고, 사용자 유형은 트랜잭션을 클라이언트 아이디로 묶어서 X-View 차트에 표시하는 방식이다. **[실시간 모니터링 | X-View]** 메뉴와 **[통계 분석 | X-View]** 메뉴에서는 X-View 유형을 선택할 수 있다.

Notice: [대시보드 | 제니퍼 대시보드] 메뉴에서는 X-View 유형이 트랜잭션으로 고정되어 있다.

8.3.2.1. 트랜잭션

트랜잭션 유형은 X-View 차트에 X 축 구간에 해당하는 트랜잭션을 X 형태의 점으로 표시한다.

그림 8-11: 트랜잭션 유형 X-Vew 차트

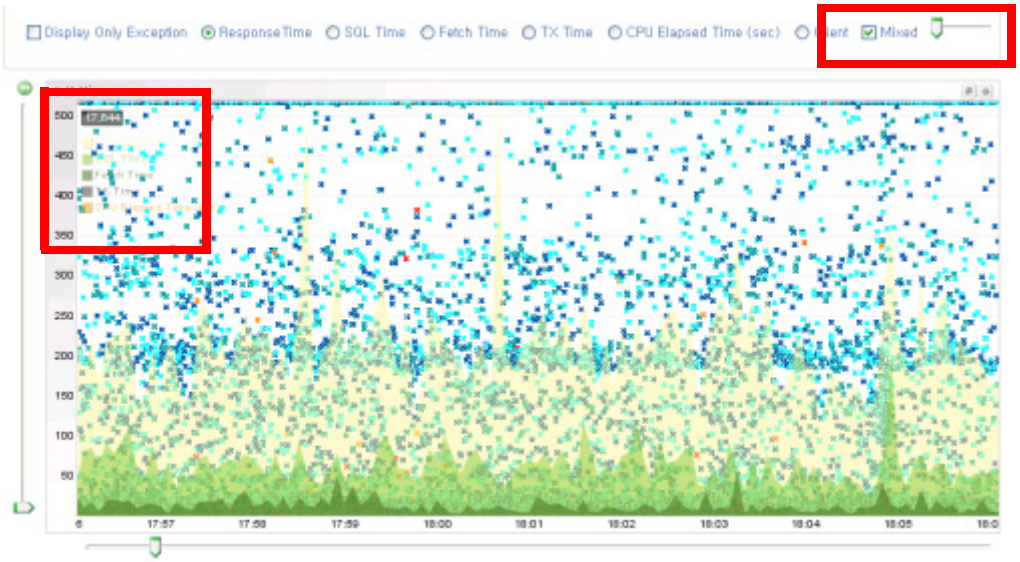


정상적인 트랜잭션은 푸른색 계열 색으로 표시되고 예외가 발생한 트랜잭션은 붉은색 계열 색으로 표시된다.

검색 조건이나 컨텍스트 메뉴를 통해서 다양한 설정을 할 수 있다. 컨텍스트 메뉴는 X-View 차트에서 오른쪽 마우스를 클릭하면 나타난다.

- 예외만 표시 - 예외가 발생한 트랜잭션만을 표시한다.
- 응답 시간 - Y 축을 응답 시간을 기준으로 표시한다. Y 축 기본 기준이 응답 시간이다.
- SQL 시간 - Y 축을 SQL 시간을 기준으로 표시한다.
- Fetch 시간 - Y 축을 Fetch 시간을 기준으로 표시한다.
- TX 시간 - Y 축을 외부 트랜잭션 시간을 기준으로 표시한다.
- CPU 시간 - Y 축을 CPU 시간을 기준으로 표시한다.
- 클라이언트 - Y 축을 클라이언트 응답 시간을 기준으로 표시한다.
- Mixed - Mixed를 선택하면 특정 단위 시간을 기준으로 경과 시간, SQL 시간, Fetch 시간, TX 시간, CPU 시간 등의 평균을 구해서 그 평균 값을 이용해서 영역을 표시한다. 단위 시간 값은 Mixed체크박스 우측의 슬라이더를 통해서 조정할 수 있다.
- 새로 고침 - 전체 X-View 트랜잭션 데이터를 다시 가져온다.

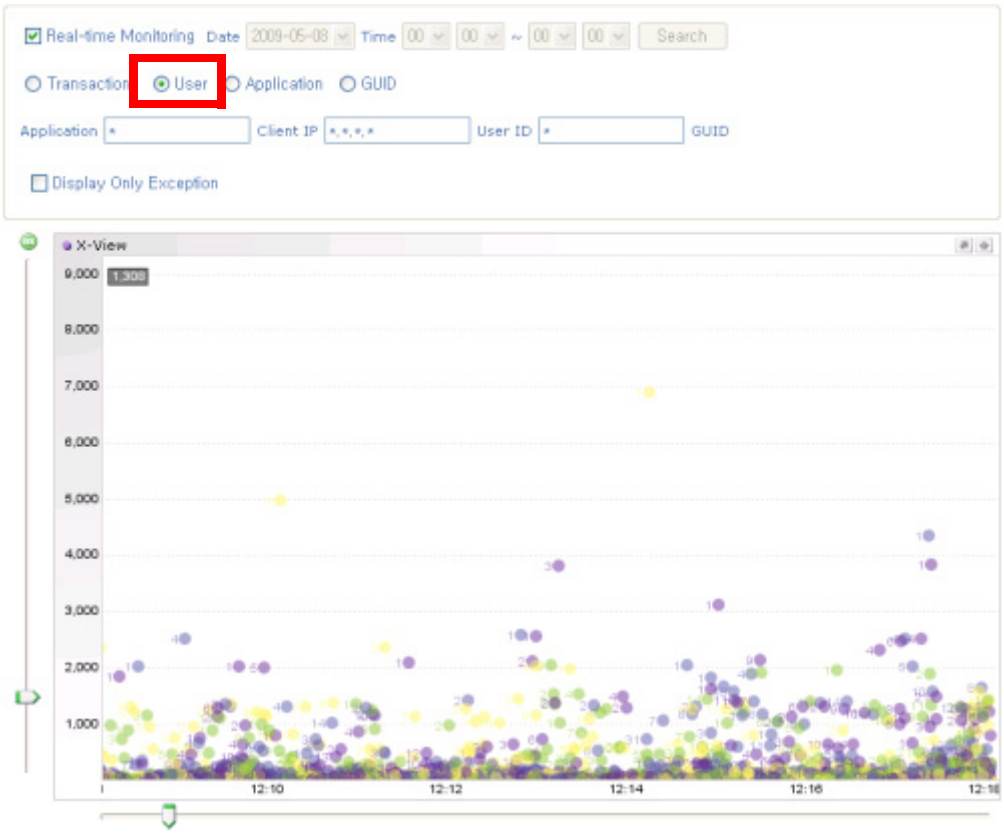
그림 8-12: 트랜잭션 유형 X-View 차트(Mixed)



8.3.2.2. 사용자

사용자 유형은 X-View 차트에 X 축 구간에 해당하는 트랜잭션을 동일한 클라이언트 아이디를 중심으로 묶어서 작은 원으로 표시한다. 원 색상은 아무런 의미가 없고 원 옆에 트랜잭션 숫자가 함께 표시된다.

그림 8-13: 사용자 유형 X-Vew 차트

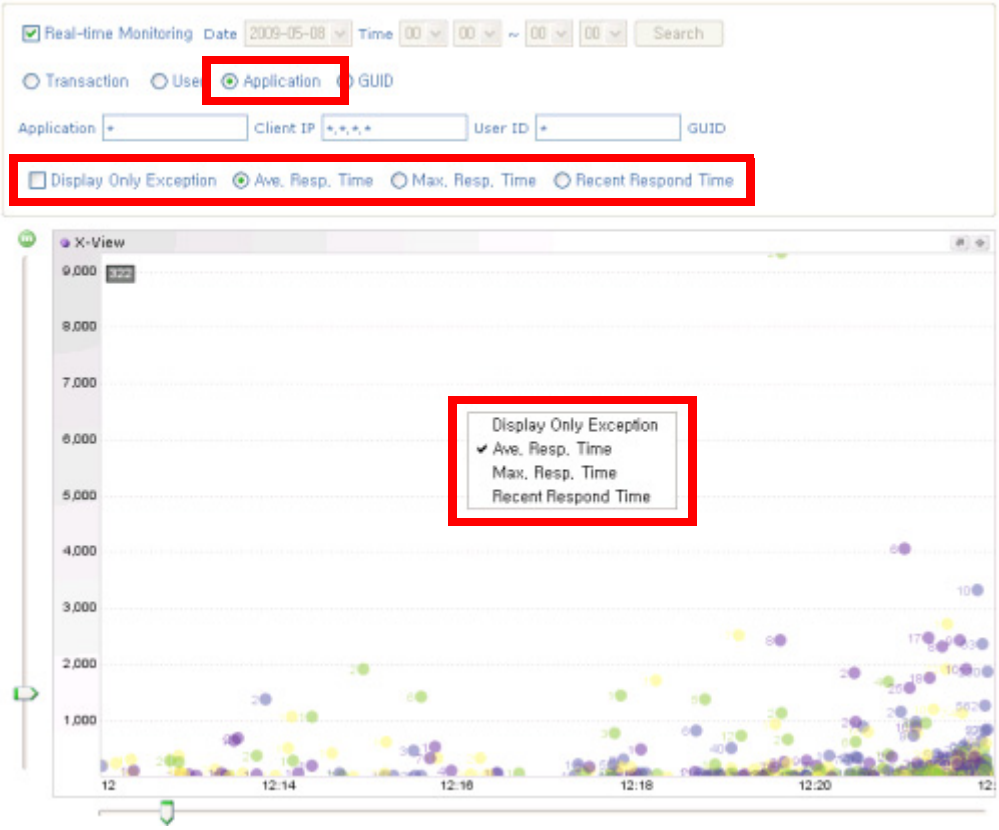


검색 조건이나 컨텍스트 메뉴를 통해서 다양한 설정을 할 수 있다. 컨텍스트 메뉴는 X-View 차트에서 오른쪽 마우스를 클릭하면 나타난다. 예외만 표시 - 예외가 발생한 트랜잭션만을 표시한다.

8.3.2.3. 애플리케이션

애플리케이션 유형은 X-View 차트에 X 축 구간에 해당하는 트랜잭션을 동일한 애플리케이션을 중심으로 묶어서 작은 원으로 표시한다. 원 색상은 아무런 의미가 없고 원 옆에 트랜잭션 숫자가 함께 표시된다.

그림 8-14: 애플리케이션 유형 X-Vew 차트



검색 조건이나 컨텍스트 메뉴를 통해서 다양한 설정을 할 수 있다. 컨텍스트 메뉴는 X-View 차트에서 오른쪽 마우스를 클릭하면 나타난다.

- 예외만 표시 - 예외가 발생한 트랜잭션만을 표시한다.
- 평균 응답 시간 - Y 축을 평균 응답 시간을 기준으로 표시한다. Y 축 기본 기준이 평균 응답 시간이다.
- 최대 응답 시간 - Y 축을 최대 응답 시간을 기준으로 표시한다.
- 최근 응답 시간 - Y 축을 최근 응답 시간을 기준으로 표시한다.

8.3.2.4. GUID

GUID 유형은 X-View 차트에 X 축 구간에 해당하는 트랜잭션을 동일한 GUID 키를 중심으로 묶어서 작은 원으로 표시한다. 원 색상은 아무런 의미가 없고 원 옆에 트랜잭션 숫자가 함께 표시된다.

Notice: 특정 원을 선택하면 동일한 GUID를 갖는 트랜잭션 데이터가 트랜잭션 리스트에 표시된다. 그러나 GUID가 동일해도 X 축 구간 밖에 있는 트랜잭션은 트랜잭션 리스트에 표시되지 않는다.

그림 8-15: GUID 유형 X-Vew 차트

GUID 응답 시간은 해당 GUID에 해당하는 모든 트랜잭션 중에서 마지막 종료 시간에서 최초 호출 시간을 뺀 값을 의미한다. Y 축을 GUID 응답 시간을 기준으로 표시한다.

8.3.3. 애플리케이션과 클라이언트 IP 등을 이용한 필터링

X-View 차트에 표시된 트랜잭션을 애플리케이션과 클라이언트 IP 등을 이용해서 필터링 할 수 있다.

애플리케이션은 정규 표현식을 이용해서 필터링할 수 있고, 클라이언트 IP에는 와일드 카드(*)를 사용할 수 있다.

```
192.168.0.1
192.*.0.1
192.*.*.*
```

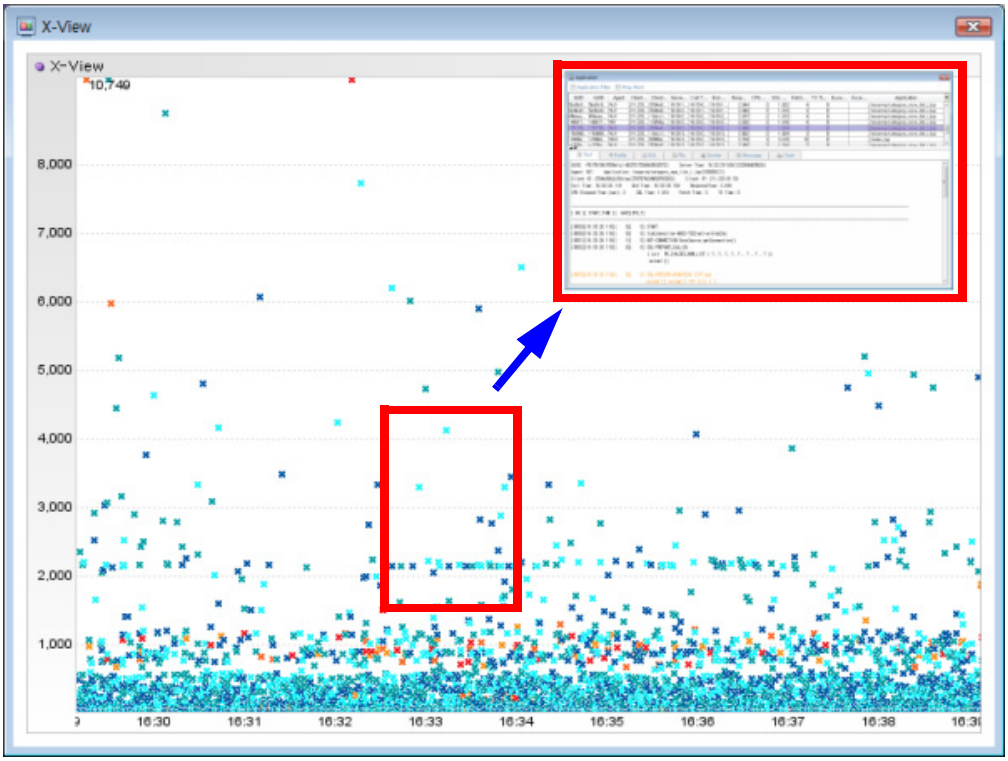
또한 사용자 아이디와 GUID를 이용해서도 필터링을 할 수 있다. GUID도 정규 표현식을 이용해서 필터링할 수 있다.

필터링은 애플리케이션과 클라이언트 IP 필드에 조건을 입력한 후에 엔터 키를 누르면 수행된다. 단, **[통계 분석 | X-View]** 메뉴에서 **[검색]** 버튼을 클릭하면 데이터를 가져올 뿐 필터링은 수행되지 않는다. 이 메뉴에서도 필터링을 수행하려면 애플리케이션과 클라이언트 IP 필드에 검색 조건을 입력한 후에 엔터 키를 눌러야 한다.

8.3.4. 트랜잭션 리스트

트랜잭션 데이터 상세 내용은 X-View 차트에서 임의의 점을 선택했을 때 확인할 수 있다. 다음 그림에서 팝업 창이 트랜잭션 리스트이다.

그림 8-16: 트랜잭션 리스트



트랜잭션 리스트 항목에 대한 설명은 [트랜잭션 데이터(204 페이지)]를 참조한다.

Notice: X-View 차트 유형이 트랜잭션인 경우에는 리스트가 하나로만 표시된다. 나머지 유형의 경우에는 2개의 리스트가 표시된다. 첫번째 리스트에는 해당 유형에 따른 데이터가 표시되고 두번째 리스트에는 첫번째 리스트에서 선택한 항목에 포함되어 있는 트랜잭션 목록이 표시된다.

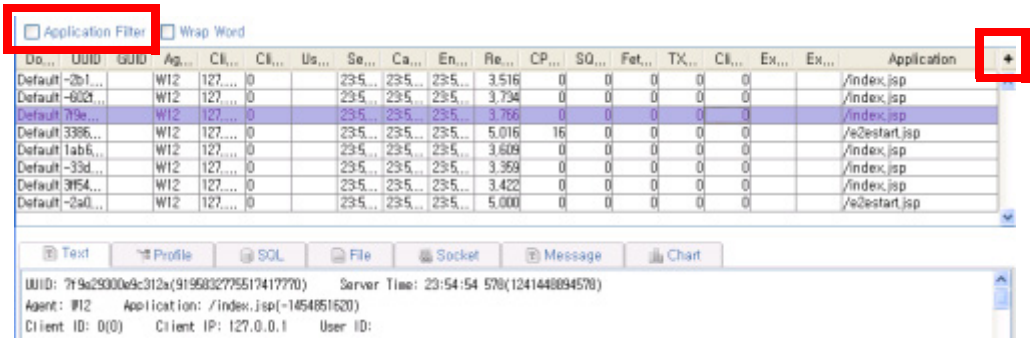
항상 트랜잭션 데이터 세부 항목이 의미가 있는 것은 아니다. 예를 들어, 제니퍼 에이전트를 설치한 대상이 자바 애플리케이션 서버가 아닌 경우에는 클라이언트 IP나 클라이언트 아이디가 0으로 표시된다. 이런 제약 조건이 존재하는 항목은 다음과 같다.

표 8-4: 트랜잭션 데이터 항목별 제약

항목	수집 조건
GUID	GUID 설정을 한 경우에만 표시
클라이언트 IP	모니터링 대상이 자바 애플리케이션 서버인 경우
클라이언트 아이디	모니터링 대상이 자바 애플리케이션 서버인 경우
사용자 아이디	모니터링 대상이 자바 애플리케이션 서버이고 사용자 아이디 추적 설정을 한 경우
CPU 시간	Native 모듈(jennifer20.so)을 올바르게 설치한 경우
SQL 시간	JDBC 자원 추적을 설정한 경우
Fetch 시간	JDBC 자원 추적을 설정한 경우
TX 시간	외부 트랜잭션 추적을 설정한 경우
클라이언트 응답 시간	모니터링 대상이 자바 애플리케이션 서버이고 클라이언트 응답 시간 추적 설정을 한 경우

트랜잭션 리스트 오른쪽 상단에 있는 [+] 아이콘을 통해서 트랜잭션 리스트 목록을 CSV 형식 파일이나 프로파일 데이터를 포함한 텍스트 파일로 Export할 수 있다.

그림 8-17: 트랜잭션 리스트 Export

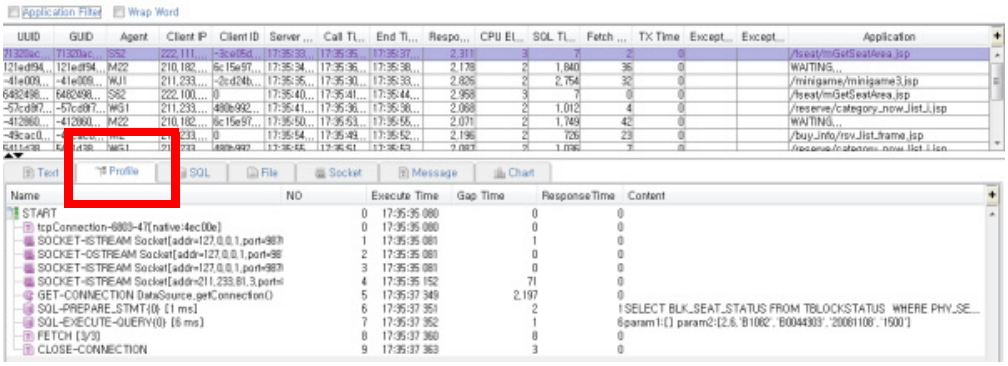


상단에 있는 애플리케이션 필터를 체크하면 트랜잭션 리스트에서 선택한 트랜잭션과 동일한 애플리케이션에 해당하는 트랜잭션들만이 X-View 차트에 표시된다.

8.3.5. 프로파일 탭 영역

트랜잭션 리스트에서 특정 트랜잭션 데이터를 선택하면 하단 프로파일 탭 영역에 해당 트랜잭션에 대한 프로파일 데이터가 표시된다.

그림 8-18: 프로파일 탭 영역



Notice: 해당 트랜잭션에 대한 프로파일 데이터가 없을 때는 텍스트 탭만이 활성화되고 다른 탭은 선택할 수 없게 된다.

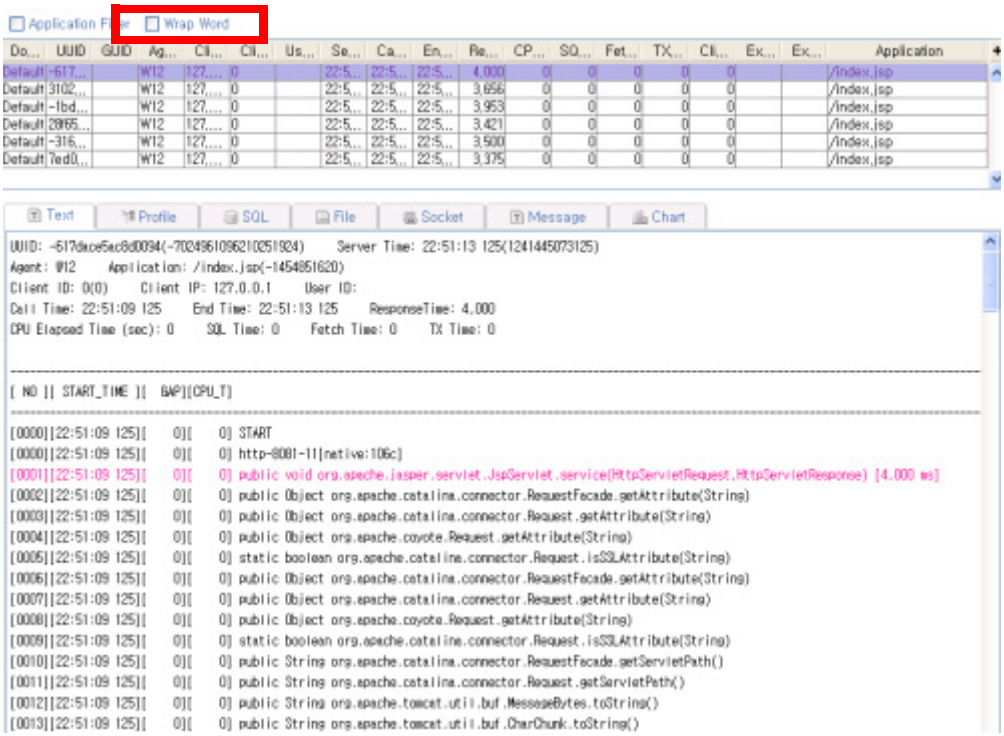
8.3.5.1. 텍스트

텍스트 탭에는 트랜잭션 리스트에서 선택한 트랜잭션에 대한 트랜잭션 데이터와 모든 프로파일 항목이 텍스트 형식으로 표시된다. 프로파일 항목 응답 시간에 따라서 텍스트 색상이 다르게 표현된다. 단위는 밀리 세컨드이다.

- 경과 시간이 1000 이상일 때 - 빨간색
- 경과 시간이 500 이상이고 1000 미만일 때 - 주황색
- 경과 시간이 100 이상이고 500 미만일 때 - 파랑색
- 경과 시간이 10 이상이고 100 미만일 때 - 녹색

좌우 스크롤이 텍스트 탭에 나타나지 않게 하려면 상단에 있는 [자동 줄바꿈]을 선택한다.

그림 8-19: 텍스트 탭



텍스트 탭에서 프로파일 데이터는 START로 시작하여 END로 끝나는 형태로 표시된다. START와 END는 시작과 종료를 표현하기 위해 임의로 추가한 안내 문자열이다.

텍스트 탭에서 오른쪽 마우스를 클릭하면 나타나는 컨텍스트 메뉴에서 **[프로파일을 게시판에 저장]** 메뉴를 클릭하면 해당 프로파일 데이터가 게시판에 저장된다.

8.3.5.2. 프로파일

프로파일 탭에는 트랜잭션 리스트에서 선택한 트랜잭션에 대한 모든 프로파일 항목이 트리 형태로 표시된다. 다음은 프로파일 리스트 칼럼에 대한 설명이다.

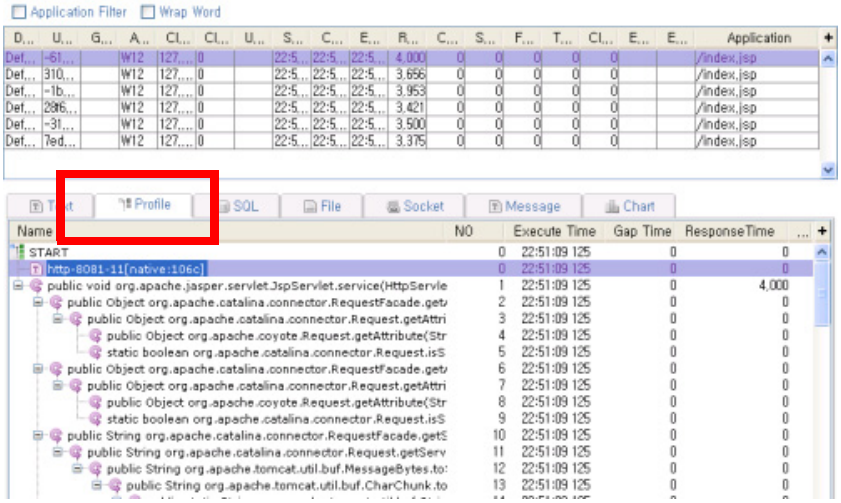
표 8-5: 프로파일 리스트 칼럼

칼럼	설명
이름	프로파일 항목 이름으로 메소드 이름이나 메시지 유형 이름이 표시된다.
NO	프로파일 항목 고유 번호
실행 시간	해당 프로파일 항목이 시작된 시간으로 제니퍼 에이전트를 설치한 자바 애플리케이션 시간을 기준으로 한다.
Gap 시간	해당 프로파일 항목과 바로 이전의 프로파일 항목 시작 시간의 차이
응답 시간	해당 프로파일 항목 수행에 소요된 시간

표 8-5: 프로파일 리스트 칼럼

칼럼	설명
내용	관련 내용

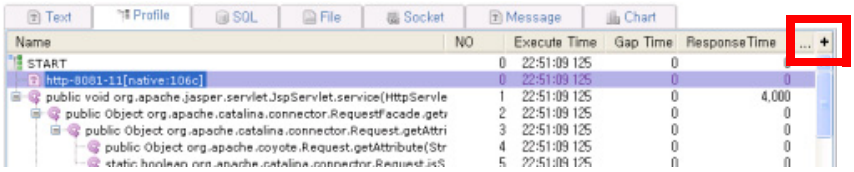
그림 8-20: 프로파일 탭



Notice: Gap 시간은 트랜잭션을 수행하는 과정에서 프로파일하지 않은 영역을 분석하는데 도움을 준다. 예를 들어, Gap 시간이 큰 프로파일 항목이 있다면 바로 전 프로파일 항목과 해당 프로파일 항목 사이에서 처리된 작업에 오랜 시간이 소요되었음을 의미한다. 따라서 그 사이에서 이루어진 작업을 프로파일링하도록 설정하면 상세한 내용을 분석할 수 있다.

이름이 [SQL-EXECUTE]로 시작하는 프로파일 항목을 선택한 후에 오른쪽 마우스를 클릭하면 SQL를 빌드하거나 복사할 수 있다. 자세한 내용은 [SQL(227 페이지)]을 참조한다. 그리고 프로파일 리스트 오른쪽 상단에 있는 [+] 아이콘을 통해서 전체 트리를 열고 닫을 수 있다.

그림 8-21: 프로파일 리스트 열고 닫기



프로파일 리스트에서 응답 시간이 큰 항목의 이름 칼럼을 선택한 후에 오른쪽 마우스를 클릭하면 나타나는 컨텍스트 메뉴에서 **[Critical Path]** 메뉴를 선택하면 해당 프로파일 항목의 하위 프로파일 항목 중에서 응답 시간이 가장 큰 항목을 쉽게 찾을 수 있다.

8.3.5.3. SQL

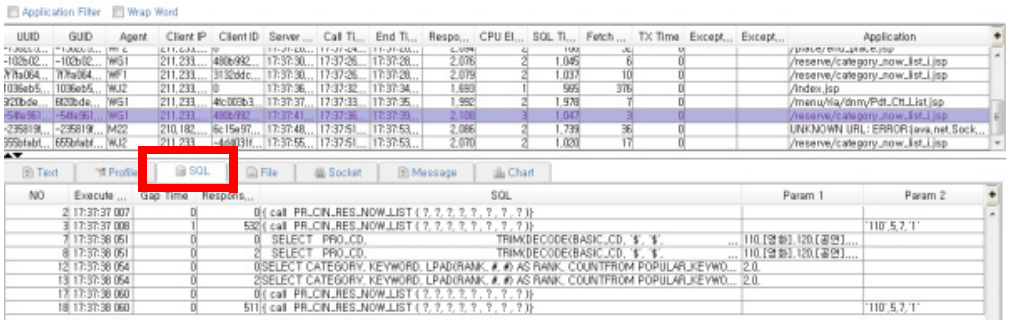
SQL 탭에는 트랜잭션 리스트에서 선택한 트랜잭션에 대한 SQL 메시지 유형 프로파일 항목이 표시된다. 이를 통해서 트랜잭션이 수행한 SQL 목록을 파악할 수 있다.

다음은 SQL 리스트 칼럼에 대한 설명이다.

표 8-6: SQL 리스트 칼럼

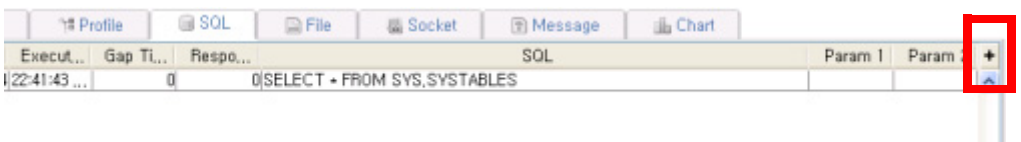
칼럼	설명
NO	프로파일 항목 고유 번호로 텍스트 탭이나 프로파일 탭에서 해당 프로파일 항목 위치를 찾는데 도움을 준다.
실행 시간	해당 프로파일 항목이 시작된 시간으로 제니퍼 에이전트를 설치한 자바 애플리케이션 시간을 기준으로 한다.
Gap 시간	해당 프로파일 항목과 바로 이전 프로파일 항목 시작 시간의 차이
응답 시간	해당 프로파일 항목 수행에 소요된 시간
SQL	SQL 쿼리
파라미터 1	SQL 쿼리 중복을 방지하기 위해서 SQL 쿼리 중에서 분리해낸 상수 파라미터
파라미터 2	SQL 쿼리에 대한 바인딩 파라미터

그림 8-22: SQL 탭



SQL 리스트 오른쪽 상단에 있는 [+] 아이콘을 통해서 전체 SQL를 열고 닫을 수 있다.

그림 8-23: SQL 리스트 열고 닫기



특정 SQL에 대한 실행 계획을 확인하는 방법은 다음과 같다.

- 특정 SQL를 선택한 후에 오른쪽 마우스를 클릭한다.
- 컨텍스트 메뉴에서 **[쿼리 빌드]** 메뉴를 선택하면 SQL 실행 계획 팝업 창이 나타난다.

SQL 실행 계획 팝업 창에 대한 상세한 정보는 SQL 실행계획을 참조한다.

또한 특정 SQL를 클릭보드에 복사하는 방법은 다음과 같다.

- 특정 SQL를 선택한 후에 오른쪽 마우스를 클릭한다.
- 컨텍스트 메뉴에서 **[복사]** 메뉴를 선택하면 SQL과 파라미터가 클립보드에 복사된다.

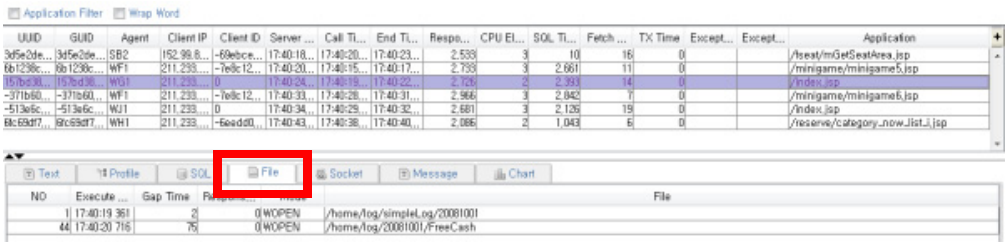
8.3.5.4. 파일

파일 탭에는 트랜잭션 리스트에서 선택한 트랜잭션에 대한 파일 메시지 유형 프로파일 항목이 표시된다. 이를 통해서 트랜잭션이 수행한 파일 IO 목록을 파악할 수 있다. 다음은 파일 리스트 칼럼에 대한 설명이다.

표 8-7: 파일 리스트 칼럼

칼럼	설명
NO	프로파일 항목 고유 번호로 텍스트 탭이나 프로파일 탭에서 해당 프로파일 항목 위치를 찾는데 도움을 준다.
실행 시간	해당 프로파일 항목이 시작된 시간으로 제니퍼 에이전트를 설치한 자바 애플리케이션 시간을 기준으로 한다.
Gap 시간	해당 프로파일 항목과 바로 이전 프로파일 항목 시작 시간의 차이
응답 시간	해당 프로파일 항목 수행에 소요된 시간
모드	[WOPEN]은 쓰기 모드를, [ROPEN]은 읽기 모드를 의미한다.
파일	파일 IO 대상이 된 파일의 경로와 이름이다.

그림 8-24: 파일 탭



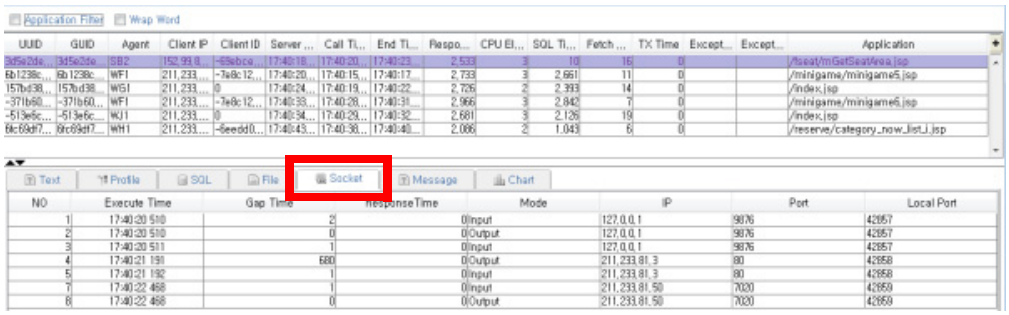
8.3.5.5. 소켓

소켓 탭에는 트랜잭션 리스트에서 선택한 트랜잭션에 대한 소켓 메시지 유형 프로파일 항목이 표시된다. 이를 통해서 트랜잭션이 수행한 소켓 IO 목록을 파악할 수 있다. 다음은 소켓 리스트 칼럼에 대한 설명이다.

표 8-8: 소켓 리스트 칼럼

칼럼	설명
NO	프로파일 항목 고유 번호로 텍스트 탭이나 프로파일 탭에서 해당 프로파일 항목 위치를 찾는데 도움을 준다.
실행 시간	해당 프로파일 항목이 시작된 시간으로 제니퍼 에이전트를 설치한 자바 애플리케이션 시간을 기준으로 한다.
Gap 시간	해당 프로파일 항목과 바로 이전 프로파일 항목 시작 시간의 차이
응답 시간	해당 프로파일 항목 수행에 소요된 시간
모드	[Output]은 쓰기 모드이고, [Input]은 읽기 모드를 의미한다.
IP	소켓 IO 대상이 되는 애플리케이션 IP 주소이다.
포트	소켓 IO 대상이 되는 애플리케이션 포트 번호이다.
로컬 포트	소켓 IO 대상이 되는 애플리케이션과 통신하는데 사용한 로컬 포트 번호이다.

그림 8-25: 소켓 탭



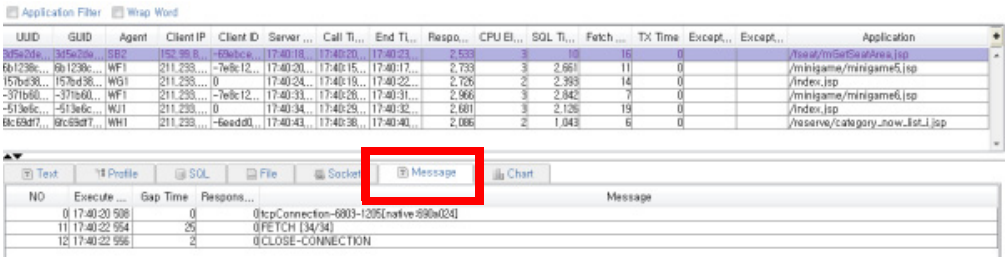
8.3.5.6. 메시지

메시지 탭에는 트랜잭션 리스트에서 선택한 트랜잭션에 대한 메시지 유형 프로파일 항목이 표시된다.

Notice: SQL, 파일, 소켓 등의 프로파일 항목도 메시지 유형에 해당하지만 메시지 탭에는 나타나지 않는다.

다음은 메시지 리스트 칼럼에 대한 설명이다.

그림 8-26: 메시지 탭



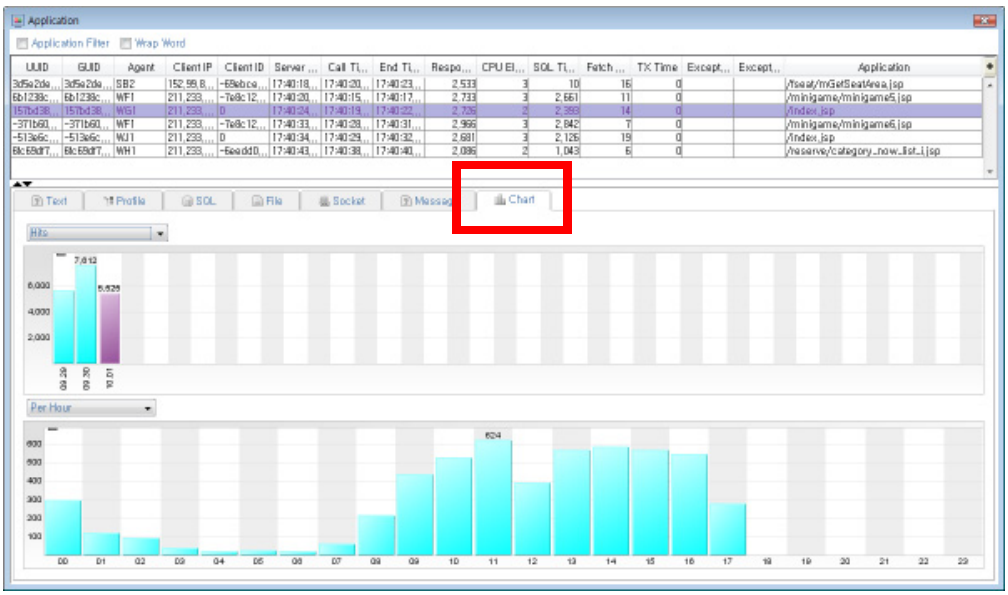
특정 메시지에 대한 상세 정보를 확인하는 방법은 다음과 같다.

- 특정 메시지를 선택한 후에 오른쪽 마우스를 클릭한다.
- 컨텍스트 메뉴에서 [상세 정보]를 선택하면 새로운 팝업 창에 메시지에 대한 상세 정보가 나타난다.

8.3.5.7. 차트

차트 탭에는 트랜잭션 리스트에서 선택한 트랜잭션의 애플리케이션에 대한 일자별 성능 데이터와 시간대별 성능 데이터가 표시된다.

그림 8-27: 차트 탭



- 일자별 차트 - 상단 차트는 일자별 성능 데이터를 보여준다. 드랍다운 박스를 통해서 성능 데이터 항목을 선택할 수 있다.

표 8-9: 성능 데이터 항목

항목	설명
호출 건수	호출 건수를 기준으로 차트를 표시한다.
실패 건수	실패 건수를 기준으로 차트를 표시한다.
응답 시간의 합	응답 시간의 합을 기준으로 차트를 표시한다.
평균 응답 시간	평균 응답 시간을 기준으로 차트를 표시한다.
최소 응답	최소 응답 시간을 기준으로 차트를 표시한다.
최대 응답	최대 응답 시간을 기준으로 차트를 표시한다.
CPU 시간	CPU 시간을 기준으로 차트를 표시한다.
CPU(tpmC)	CPU(tpmC)를 기준으로 차트를 표시한다.

- 일일 차트 - 상단 일자별 차트에서 특정 일을 선택하면 하단 차트에는 선택한 날짜에 해당하는 일일 성능 데이터가 표시된다. 드랍다운 박스에서 [1시간당]을 선택하면 1 시간 구간별 막대 차트가 표시되고, [단위시간당 (10분)]을 선택하면 10분 구간별 라인 차트가 표시된다.

8.3.6. X-View를 이용한 문제 해결

X-View에서 비정상적인 패턴이 발견되면 해당 트랜잭션을 선별적으로 분석할 수 있다는 것이 제니퍼의 강점이다. 관련 트랜잭션들에 대한 성능 데이터를 비교함으로써 문제가 어디서 발생했는지를 쉽게 판별할 수 있다.

트랜잭션 데이터를 분석할 때 다음 사항을 참고하도록 한다.

1. 문제가 있는 트랜잭션들의 제니퍼 에이전트가 동일하다면 자바 애플리케이션 내부의 문제일 가능성이 높고, 서로 다르다면 자바 애플리케이션 외부 문제일 가능성이 높다.
2. 문제가 있는 트랜잭션들의 애플리케이션 이름이 동일하면 특정 업무 문제일 가능성이 높다.
3. 문제가 있는 트랜잭션들의 응답 시간에서 SQL 혹은 Fetch 시간이 차지하는 비중이 크면 데이터베이스 문제일 가능성이 높고, TX 시간이 차지하는 비중이 크면 데이터베이스를 제외하면 외부 시스템 문제일 가능성이 높다.
4. JDBC 연결에 문제가 발생하면 SQL 혹은 Fetch 시간은 크지 않을 수 있다. JDBC 연결 문제는 데이터베이스나 자바 애플리케이션 서버 내부 문제에 기인할 수 있는데 이것은 문제가 있는 트랜잭션들의 제니퍼 에이전트가 동일한지 다른지로 구분할 수 있다.
5. 문제가 있는 트랜잭션들의 CPU 시간이 크다면 로직적 부하(문자열 제어, 루핑 로직 등)가 존재할 가능성이 높다.

6. 애플리케이션 이름이 동일한 JSP를 실행한 경우에 특정 트랜잭션만이 응답 시간이 큰 경우가 있다. 이 경우에 응답 시간에서 CPU 시간이 차지하는 비중이 큰 경우에는 JSP 컴파일로 발생했을 가능성이 있다. 이 경우에는 프로파일 데이터에서 FILE OPEN 메시지를 확인할 수 있을 것이다.

Notice: 프로파일 관련 설정이 정확하지 않으면 일부 정보가 사실과 다를 수 있기 때문에 정확성 여부를 확인해야 한다. 예를 들어, 운영 체계와 맞지 않는 Native 모듈(jennifer20.so/dll 파일)을 사용하면 기형적인 CPU 시간이 표시될 수 있다.

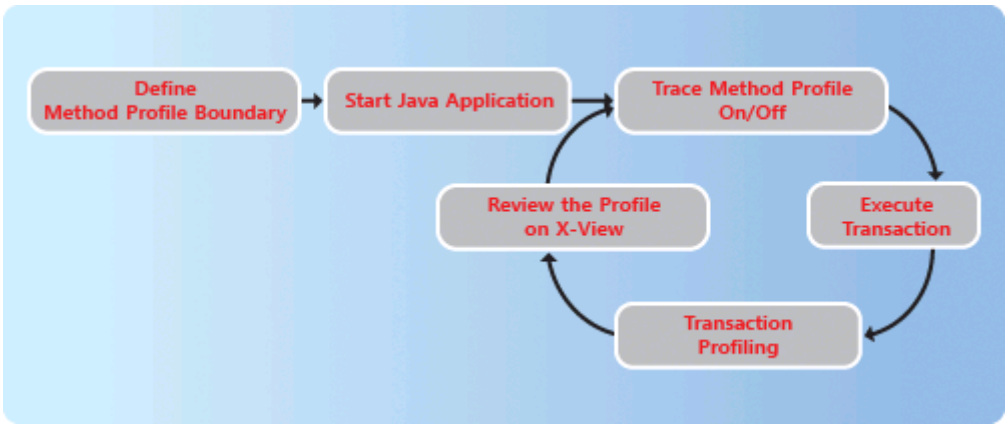
8.4. 자바 메소드 프로파일링

제니퍼는 프로파일링을 연계 프로파일링과 메소드 프로파일링으로 구분한다. DB 혹은 외부 트랜잭션 추적을 연계 프로파일링이라고 하고, 메소드 호출 응답 시간과 파라미터 혹은 반환 값 등의 추적을 메소드 프로파일링이라 한다.

Notice: 메소드 프로파일링은 연계 프로파일링에 비해서 우선 순위가 낮다. 따라서 연계 프로파일 설정이 메소드 프로파일 설정에 우선한다. 예를 들어, A 클래스가 연계 프로파일과 메소드 프로파일에 모두 설정되어 있다면 메소드 프로파일링은 이루어지지 않고, 연계 프로파일링만이 이루어진다.

메소드 프로파일링은 다음과 같은 과정으로 이루어진다.

그림 8-28: 메소드 프로파일링 과정



- 메소드 프로파일 범위 설정 - 모든 클래스의 모든 메소드를 프로파일링하면 성능 저하가 야기될 수 있다. 따라서 분석에 필요한 클래스의 메소드만을 프로파일링하는 것이 효율적이다. 메소드 프로파일 범위 설정은 어떤 클래스의 어떤 메소드를 프로파일링할 것인가를 결정하는 단계이다.

- 자바 애플리케이션 시작 - 변경한 메소드 프로파일 범위 설정을 반영하려면 자바 애플리케이션 재시작이 필요하다. 단, 제니퍼 에이전트를 javaagent로 설치한 경우에는 자바 애플리케이션 재시작없이 메소드 프로파일 범위 설정을 변경할 수 있다.
- 메소드 프로파일 제어 - 메소드 프로파일 범위에 포함되어 있는 클래스에 대한 메소드 프로파일 여부를 On/Off할 수 있다. 메소드 프로파일 기능이 Off되면 메소드 프로파일이 수행되지 않는다.
- 트랜잭션 수행과 프로파일링 - 트랜잭션이 수행되면 프로파일 데이터 수집도 함께 이루어진다. 이렇게 수집된 프로파일 데이터는 제니퍼 서버에 전송된다.
- 프로파일 데이터 확인 - X-View 차트를 통해서 프로파일 데이터를 확인할 수 있다.

8.4.1. 메소드 프로파일 범위 설정

제니퍼 에이전트의 profile로 시작하는 옵션들로 메소드 프로파일 범위를 설정한다. 수정한 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

Notice: java, org.jsn, org.apache.jennifer. com.javaservice 패키지는 프로파일링할 수 없다. 단, javax 패키지는 프로파일링할 수 있다.

example.BusinessManager 클래스의 모든 메소드를 프로파일링하려면 다음과 같이 설정한다.

```
profile_class = example.BusinessManager
```

추가로 example.DomainManager 클래스의 모든 메소드를 프로파일링하려면 세미콜론[;]을 구분자로 해서 profile_class 옵션에 설정한다. profile_class 옵션뿐만 아니라 메소드 프로파일 범위 설정과 관련한 모든 옵션에 2개 이상을 설정하려면 세미콜론[;]을 구분자로 사용한다.

```
profile_class = example.BusinessManager;example.DomainManager
```

메소드 프로파일 단위는 특정 클래스의 특정 메소드이다. 즉, 앞에서와 같이 클래스만 설정하면 example.BusinessManager 클래스의 모든 메소드가 메소드 프로파일 범위에 포함된다. 따라서 특정 메소드만을 메소드 프로파일 범위에 포함시키려면 제니퍼 에이전트의 profile_target_method 옵션으로 해당 메소드를 설정한다.

```
profile_target_method = execute
```

그런데 메소드 프로파일 범위로 설정한 클래스들 중에서 이름이 동일한 메소드가 있고, 이중 특정 클래스의 특정 메소드만을 메소드 프로파일 범위에 포함시키려면 다음과 같이 구체적으로 설정한다.

```
profile_target_method = example.BusinessManager.execute
```

반대로 특정 메소드만을 메소드 프로파일 범위에서 제외하려면 제니퍼 에이전트의 profile_ignore_method 옵션으로 설정한다.

```
profile_ignore_method = example.BusinessManager.someMethod
```

메소드의 접근자를 통해서도 메소드 프로파일 범위를 설정할 수 있다. 예를 들어, public 접근자와 접근자가 없는 메소드를 메소드 프로파일 범위에 포함시키려면 다음과 같이 설정한다.

```
profile_access_method = public;none
```

profile_access_method 옵션에 설정이 가능한 값은 다음과 같다.

- public - public 접근자
- protected - protected 접근자
- private - private 접근자
- none - 접근자가 없는 경우

그리고 특정 클래스를 상속한 모든 클래스를 메소드 프로파일 범위에 포함시키려면 제니퍼 에이전트의 profile_client_super 옵션을 사용한다. 예를 들어, example.ejb.BaseSessionBean을 상속한 모든 클래스를 메소드 프로파일 범위에 포함시키려면 다음과 같이 설정한다.

```
profile_super = example.ejb.BaseSessionBean
```

그러나 이 경우에 직접적으로 상속받은 클래스만이 메소드 프로파일 범위에 포함된다. A 클래스가 example.ejb.BaseSessionBean 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스는 메소드 프로파일 범위에 포함되지만 B 클래스는 메소드 프로파일 범위에 포함되지 않는다. 그리고 특정 인터페이스를 구현한 모든 클래스를 메소드 프로파일 범위에 포함시키려면 제니퍼 에이전트의 profile_interface 옵션을 사용한다. 예를 들어, example.eai.IBusinessManager 인터페이스를 구현한 모든 클래스를 메소드 프로파일 범위에 포함시키려면 다음과 같이 설정한다.

```
profile_interface = example.eai.IBusinessManager
```

그러나 이 경우에 직접적으로 구현한 클래스만이 메소드 프로파일 범위에 포함된다. A 클래스가 example.eai.IBusinessManager 인터페이스를 구현하고 B 클래스가 A 클래스를

상속했다면, A 클래스는 메소드 프로파일 범위에 포함되지만 B 클래스는 메소드 프로파일 범위에 포함되지 않는다. 그리고 클래스의 이름을 이용해서도 메소드 프로파일 범위를 설정할 수 있다. 이 경우에는 제니퍼 에이전트의 profile_prefix, profile_postfix, profile_ignore_prefix, profile_ignore_postfix 옵션을 사용한다. 이름이 example.biz로 시작하는 모든 클래스를 메소드 프로파일 범위에 포함시키려면 다음과 같이 설정한다.

```
profile_prefix = example.biz
```

또한 이름이 Manager로 끝나는 모든 클래스를 메소드 프로파일 범위에 포함시키려면 다음과 같이 설정한다.

```
profile_postfix = Manager
```

그리고 특정 이름으로 시작하는 클래스를 메소드 프로파일 범위에서 제외하려면 제니퍼 에이전트의 profile_ignore_prefix 옵션을 사용한다. 이 옵션은 다른 모든 옵션에 우선한다.

```
profile_ignore_prefix =
```

그리고 특정 이름으로 끝나는 클래스를 메소드 프로파일 범위에서 제외하려면 제니퍼 에이전트의 profile_ignore_postfix 옵션을 사용한다. 이 옵션은 다른 모든 옵션에 우선한다.

```
profile_ignore_postfix =
```

제니퍼 에이전트의 profile로 시작하는 옵션들로 동일한 내용을 다양한 방법으로 설정할 수 있다. 메소드 프로파일 범위를 설정하는데 있어서 profile_target_method 옵션으로 실제로 메소드 프로파일 범위에 포함시킬 메소드만을 설정하도록 한다.

예를 들어, pkg.ClassA와 pkg.ClassB 클래스가 있다. 여기서 ClassA 클래스의 run 메소드와 ClassB 클래스의 process 메소드를 메소드 프로파일 범위에 포함시켜야 한다고 가정한다. 그런데 ClassB 클래스에 run 메소드가 존재하고 이 메소드는 메소드 프로파일 범위에서 제외하려면 다음과 같이 설정한다.

```
profile_class = pkg.ClassA;pkg.ClassB
profile_target_method = run;process
profile_ignore_method = pkg.ClassB.run
```

또는 다음과 같이 설정할 수도 있다.

```
profile_class = pkg.ClassA;pkg.ClassB
profile_target_method = pkg.ClassA.run;pkg.ClassB.process
```

메소드 크기로 메소드 프로파일 범위를 설정할 수도 있다. 제니퍼 에이전트의 profile_method_max_byte 옵션으로 메소드 프로파일 범위에 포함시킬 메소드 최대 크기를 설정한다. 기본 값은 32000이고 단위는 바이트이다.

```
profile_method_max_byte = 32000
```

자바 메소드 크기가 64 KB를 넘을 수 없기 때문에 메소드 크기가 64 KB를 넘으면 에러가 발생한다. 메소드 프로파일링을 위해서 제니퍼가 해당 메소드에 추적 코드를 삽입하기 때문에 본래보다 메소드 크기가 증가한다. 따라서 이 옵션을 50 KB 이상으로 설정하지 않도록 한다.

그리고 제니퍼 에이전트의 profile_method_min_byte 옵션으로 메소드 프로파일 범위에 포함시킬 메소드 최소 크기를 설정한다. 기본 값은 0이고 단위는 바이트이다.

```
profile_method_min_byte = 0
```

Notice: getter와 setter와 같이 로직이 단순한 메소드를 메소드 크기에 기반해서 메소드 프로파일 범위에서 제외할 수 있다.

8.4.2. 프로파일링 제어

너무 많은 메소드를 프로파일링하면 제니퍼 서버가 수집하는 데이터 양이 크게 증가하는 문제가 있다. 이를 방지하기 위해서 한 트랜잭션이 수행되는 과정에서 프로파일되는 최대 항목 개수를 제니퍼 에이전트의 profile_buffer_size 옵션으로 설정할 수 있다. 기본 값은 1000이다.

```
profile_buffer_size = 1000
```

메소드 프로파일 범위를 설정해도 메소드 프로파일링은 이루어지지 않는다. 제니퍼 에이전트의 profile_default_on 옵션을 true로 설정해야 메소드 프로파일 범위에 포함된 메소드에 대한 메소드 프로파일링이 이루어진다. 기본 값은 false이다.

```
profile_default_on = true
```

메소드 프로파일링에 따른 성능 저하를 방지하기 위해서 이 옵션을 true로 설정하는 것은 권장하지 않는다.

profile_default_on 옵션을 false로 설정한 경우에도 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작하지 않고 특정 클래스에 대해서만 메소드 프로파일링 On/Off를 제어할 수 있다. 이를 위한 방법은 [다이나믹 프로파일링(237 페이지)]을 참조한다.

profile_default_on 옵션은 전체 메소드 프로파일 범위에 대한 On/Off를 제어한다. 그런데 특정 옵션으로 설정한 메소드 프로파일 범위에 대해서만 On/Off를 제어할 수 있다. 예를 들어, profile_class 옵션으로 설정한 메소드 프로파일 범위에 대한 On/Off를 제어하려면 제니퍼 에이전트의 profile_class_on 옵션을 사용한다. 기본 값은 false이다.

profile_super 옵션으로 설정한 메소드 프로파일 범위에 대한 On/Off를 제어하려면 제니퍼 에이전트의 profile_super_on 옵션을 사용한다. 기본 값은 false이다.

```
profile_super_on = true
```

profile_interface 옵션으로 설정한 메소드 프로파일 범위에 대한 On/Off를 제어하려면 제니퍼 에이전트의 profile_interface_on 옵션을 사용한다. 기본 값은 false이다.

```
profile_interface_on = true
```

8.4.3. Boot Class 프로파일링

제니퍼 에이전트를 javaagent로 설치한 경우에는 -Xbootclasspath에 설정된 클래스도 프로파일링할 수 있다. 단, 이 경우에도 java 패키지는 프로파일링할 수 없다. 이를 위해서는 제니퍼 에이전트의 enable_hooking_boot 옵션을 true로 설정한다. 기본 값은 false이다.

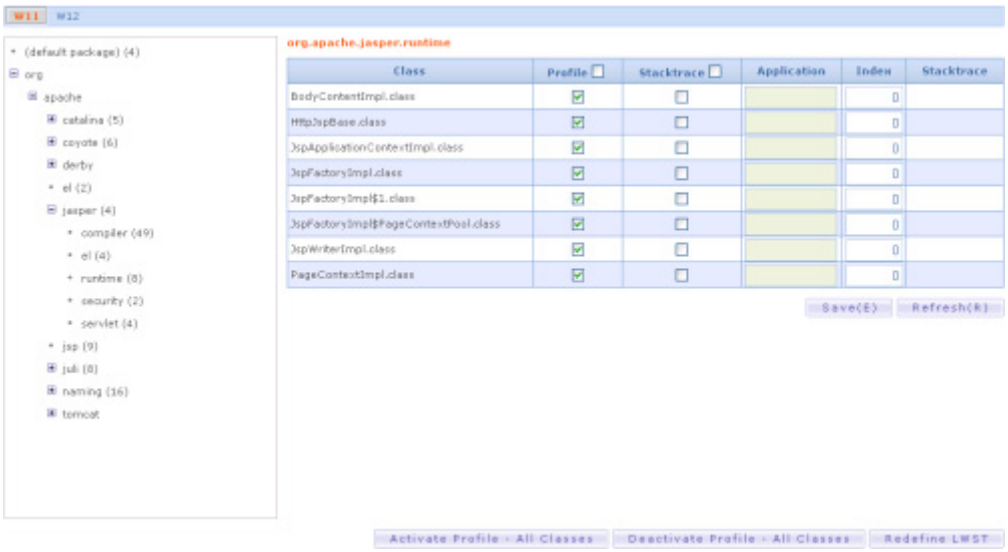
```
enable_hooking_boot = true
```

일반적으로 Boot Class 패스에는 시스템적인 성격의 클래스만을 설정한다. 따라서 업무 클래스가 Boot Class 패스에 설정된 경우에만 예외적으로 사용하는 것을 권장한다.

8.4.4. 다이나믹 프로파일링

제니퍼 에이전트를 설치한 자바 애플리케이션 재시작없이 동적으로 메소드 프로파일 범위 설정을 변경하거나 메소드 프로파일 범위에 포함된 클래스의 프로파일 On/Off를 제어하는 것을 다이나믹 프로파일링이라고 한다. 단, 제니퍼 에이전트를 javaagent로 설치한 경우만 자바 애플리케이션 재시작없이 메소드 프로파일 범위 설정을 변경할 수 있다. **[실시간 모니터링 | 프로파일]** 메뉴에서 메소드 프로파일 범위에 포함된 클래스에 대한 프로파일 On/Off를 제어할 수 있다.

그림 8-29: 다이나믹 프로파일링



메소드 프로파일 범위에 포함된 클래스를 확인하는 방법은 다음과 같다. 제니퍼 에이전트 선택 영역에서 메소드 프로파일 범위를 확인할 제니퍼 에이전트를 선택한다. 화면 왼쪽에 선택한 제니퍼 에이전트를 설치한 자바 애플리케이션 패키지 구조가 트리 형태로 나타난다. 패키지 트리에서 특정 패키지를 선택하면 화면 오른쪽에 해당 패키지에 있는 클래스 중에서 메소드 프로파일 범위에 포함된 클래스 목록이 나타난다. 클래스 목록에서 프로파일 칼럼이 체크되어 있는 클래스는 프로파일링이 수행되고 있음을 의미한다.

Notice: 해당 패키지에 있는 모든 클래스가 나타나는 것이 아니라 profile로 시작하는 옵션들로 설정한 메소드 프로파임 범위에 포함된 클래스들만이 나타난다.

특정 클래스에 대한 메소드 프로파일 On/Off는 다음과 같은 방법으로 변경한다.

- 패키지 트리에서 메소드 프로파일 On/Off를 변경할 클래스가 포함되어 있는 패키지를 선택한다.
- 클래스 목록에서 메소드 프로파일 On/Off를 변경한 후에 하단에 있는 [저장] 버튼을 클릭한다.

패키지 트리에서 default package를 선택하면 클래스 4개가 항상 표시된다. 제니퍼 에이전트 동작 방식을 제어하는데 이 클래스들을 사용할 수 있다.

표 8-10: 제니퍼 에이전트 제어를 위한 클래스

클래스	설명	기본 프로파일 상태
PROFILE_MESSAGE	프로파일을 Off로 변경하면 메시지 유형 프로파일 항목을 수집하지 않는다. 단, getConnection과 관련한 메시지는 영향을 받지 않는다.	true

표 8-10: 제니퍼 에이전트 제어를 위한 클래스

클래스	설명	기본 프로파일 상태
PROFILE_PARAM	프로파일을 Off로 변경하면 [파라미터와 반환 값 추적(242 페이지)]에 설명한 기능이 동작하지 않는다.	true
PROFILE_SQL	프로파일을 Off로 변경하면 SQL 유형 프로파일 항목을 수집하지 않는다.	true
PROFILE_USER_CONTROL	향후 기능 확정을 위한 것으로 현재 기능에는 영향을 미치지 않는다.	false

메소드 프로파일 범위에 포함되어 있는 모든 클래스들에 대한 프로파일 On/Off 변경은 화면 하단에 있는 **[전체 프로파일 설정]**과 **[전체 프로파일 해제]** 버튼을 이용한다.

그림 8-30: 전체 메소드 프로파일 범위 On/Off 제어

제니퍼 에이전트를 javaagent로 설치한 경우에는 자바 애플리케이션 재시작없이 메소드 프로파일 범위 설정을 변경할 수 있다. 이를 LWST 재설정이라고 한다.

LWST 재설정 방법은 다음과 같다.

- **[실시간 모니터링 | 프로파일]** 메뉴 하단에 있는 **[LWST 재설정]** 버튼을 클릭하면 LWST 재설정 팝업 창이 나타난다.
- LWST 재설정 팝업 창은 로딩 클래스 목록과 구성 설정 탭으로 이루어져 있다. 우선 구성 설정 탭에서 profile로 시작하는 옵션들로 메소드 프로파일 범위를 변경한다.
- LWST 재설정 팝업 창 로딩 클래스 목록 탭에서 프로파일 범위 재설정을 할 클래스를 선택한 후에 하단에 있는 **[적용]** 버튼을 클릭한다. 클래스 이름으로 검색을 할 수 있다.
- LWST 재설정 팝업 창을 닫고 **[실시간 모니터링 | 프로파일]** 메뉴에서 반영 여부를 확인한다.

LWST 재설정은 모든 클래스가 아닌 선택된 클래스에만 반영된다. 예를 들어, example.ClassA와 example.ClassB 클래스가 있는데 이 두 클래스는 메소드 프로파일 범위에

포함되어 있지 않다. LWST 재설정을 위해서 profile_class 옵션을 다음과 같이 수정하였다.

```
profile_class = example.ClassA;example.ClassB
```

LWST 재설정 팝업 창 로딩 클래스 목록 탭에서 example.CassA 클래스만을 선택한 후에 하단에 있는 **[적용]** 버튼을 클릭했다면, example.ClassA 클래스는 메소드 프로파일 범위에 포함되지만, example.ClassB 클래스는 메소드 포로파일 범위에 포함되지 않는다. 단, 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작하면 example.ClassB 클래스도 메소드 프로파일 범위에 포함된다.

8.4.5. 다이나믹 스택 트레이스

모든 클래스의 모든 메소드를 프로파일링은 하는 것은 부하가 크고, 수집한 프로파일 데이터 중에서 적은 일부만을 분석 목적으로 사용하기 때문에 효율적이지 못하다. 따라서 성능이나 기능 장애가 있는 애플리케이션에 직접적으로 관련된 클래스만을 프로파일링을 하는 것이 효율적이다. 그런데 외부 패키지나 직접 개발하지 않은 소스 코드를 사용하는 경우에는 특정 애플리케이션과 직접적으로 관련된 클래스들을 파악하는 것은 쉬운 일이 아니다. 바로 다이나믹 스택트레이스는 특정 애플리케이션과 직접적으로 관련된 클래스들을 찾는 기능이다. 메소드 프로파일 범위에 포함된 클래스 중에서 임의의 클래스를 기반으로 자바 스택을 추적함으로써 트랜잭션 흐름을 쉽게 파악하고, 프로파일 대상 클래스를 쉽게 선별할 수 있도록 도와 준다.

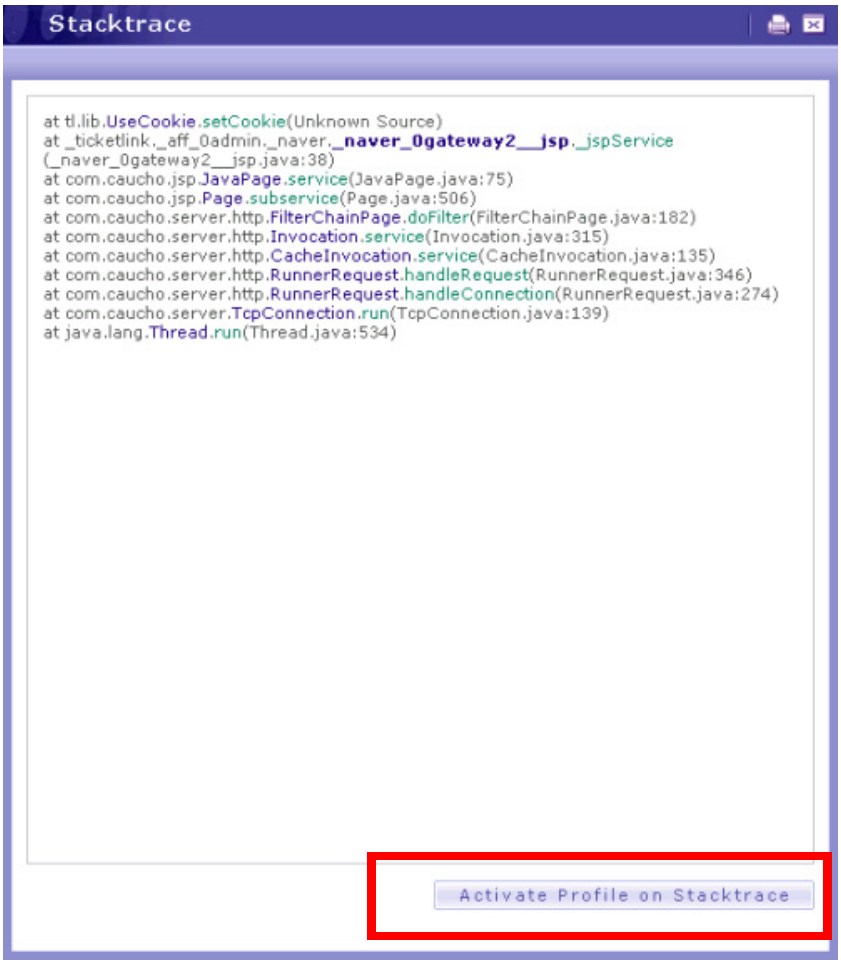
Notice: 다이나믹 스택트레이스 기능을 사용하기 위해서는 특정 애플리케이션에서 사용하고 있는 1개 이상의 클래스는 파악하고 있어야 한다.

다이나믹 스택트레이스 기능은 다음과 같이 사용한다.

- 성능이나 기능 장애가 있는 애플리케이션에서 사용하고 있는 임의의 클래스에 대한 메소드 프로파일을 On으로 변경한다.
- 해당 클래스의 스택트레이스 체크 박스를 선택한 후에 **[저장]** 버튼을 클릭한다.
- 자바 스택트레이스가 표시될 때까지 **[새로 고침]** 버튼을 반복적으로 클릭한다.
- 임의의 트랜잭션에 의해서 해당 클래스가 사용되면 스택트레이스가 기록되고, 스택트레이스 상태는 Off로 변한다.

스택트레이스에 포함되어 있는 모든 클래스에 대한 메소드 프로파일을 On으로 설정하는 방법은 다음과 같다.

그림 8-31: 스텍트레이스



기본적으로 해당 클래스를 사용한 임의의 첫번째 트랜잭션과 관련한 스택트레이스가 기록된다. 그런데 특정 애플리케이션에 의한 트랜잭션과 관련한 스택트레이스가 기록되게 할 수 있다. 이는 다음과 같은 방법으로 설정한다.

- 클래스 목록에서 스택트레이스를 수집할 클래스의 애플리케이션 필드를 클릭하면 애플리케이션 목록 팝업 창이 나타난다.
- 애플리케이션 목록 팝업 창에서 스택트레이스를 기록할 애플리케이션 이름을 클릭한다. 그러면 애플리케이션 필드에 선택한 애플리케이션 이름이 자동으로 입력된다.
- 클래스 목록 하단에 있는 **[저장]** 버튼을 클릭한다.

트랜잭션을 수행하는 자바 쓰레드가 아닌 백그라운드 쓰레드가 임의의 클래스를 사용하는 것에 대한 스택트레이스도 기록할 수 있다.

- 클래스 목록에서 스택트레이스를 수집할 클래스의 인덱스 필드에 -1을 입력한다.
- 클래스 목록 하단에 있는 **[저장]** 버튼을 클릭한다.

8.4.6. 파라미터와 반환 값 추적

메소드 프로파일을 수행할 때 특정 메소드의 파라미터 값이나 반환 값을 함께 수집할 수도 있다. 단, 메소드 파라미터나 반환 값의 유형이 java.lang.String이어야만 한다. 예를 들어, 다음 클래스에 대한 메소드 파라미터 값과 반환 값을 추적하는 방법은 다음과 같다.

```
package pkg;

public class ClassC {

    public void ptrace1(String value) {
        System.out.println("ClassC.process:" + value);
    }

    public void ptrace2(int x, String value) {
       System.out.println("ClassC.process:" + value);
    }

    public String rtrace() {
        return "It's the return value";
    }
}
```

ClassC 클래스의 ptrace1 메소드 파라미터 값을 추적하려면 제니퍼 에이전트의 lwst_profile_method_using_param 옵션을 다음과 같이 설정한다.

```
lwst_profile_method_using_param = pkg.ClassC.ptrace1(String)
```

java.lang.String 유형 파라미터가 여러 개인 메소드는 첫번째 String 유형 파라미터 값만이 수집된다.

클래스 이름과 메소드 이름을 점[.]으로 구분하여 설정하고 클래스 이름은 패키지 이름을 포함하여야 한다. 메소드는 파라미터 유형을 명시하여 지정해야 한다. 두개 이상의 메소드를 설정하려면 세미콜론[;]을 구분자로 구분한다. 예를 들어, ptrace2 메소드 파라미터 값도 추적하려면 다음과 같이 설정한다.

```
lwst_profile_method_using_param =
pkg.ClassC.ptrace1(String);pkg.ClassC.ptrace2(int,String)
```

메소드 반환 값은 제니퍼 에이전트의 lwst_profile_method_using_return 옵션으로 설정한다.

```
lwst_profile_method_using_return = pkg.ClassC.rtrace()
```

두개 이상의 메소드를 설정하려면 세미콜론[;]을 구분자로 구분한다. 이와 같이 설정하면 프로파일 데이터에서 다음 내용을 확인할 수 있다.

```
PARAM[test param] ClassC.ptrace1
PARAM[test param] ClassC.ptrace2
RETURN[It's the return value] ClassC.rtrace
```

8.4.7. 호출되는 메소드 추적

메소드 내부 로직이 복잡한 경우에 profile_call옵션을 이용해 메소드 내부를 보다 상세하게 분석할 수 있다. 예를 들어, 다음과 같은 클래스가 있다.

```
package pkg;

public class ClassD {

  public void process() {
       // 단계 1
       ClassE e = new ClassE();
       e.run();
       // 단계 2
       ClassF f = new ClassF();
       f.run();
  }
}
```

ClassD 클래스의 process 메소드는 다음과 같이 이루어져 있다.

1. 단계 1을 수행
2. ClassE 객체를 생성한 후에 run 메소드를 호출
3. 단계 2를 수행
4. ClassF 객체를 생성한 후에 run 메소드를 호출

단계 1과 2에서는 다양한 로직이 수행되고 여러 메소드가 호출될 수 있다. 이 때 개별 메소드 응답 시간이 아닌 단계 1과 2의 수행 시간 자체를 측정하라면 제니퍼 에이전트의 profile_call 옵션을 사용한다. 먼저 profile_class 옵션으로 ClassD를 메소드 프로파일 범위에 포함시킨다.

```
profile_class = pkg.ClassD
```

그리고 profile_call 옵션을 다음과 같이 설정한다. 2개 이상은 세미콜론[;]을 구분자로 구분한다.

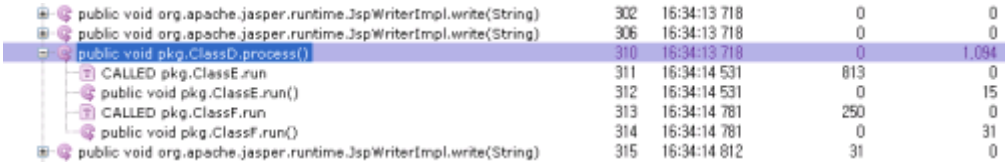
```
profile_call = pkg.ClassE.run;pkg.ClassF.run
```

메소드 이름이 같은 경우에는 별표[*]를 사용하여 설정할 수도 있다.

```
profile_call = *.run
```

이와 같이 설정하면 프로파일 데이터에서 다음 내용을 확인할 수 있다.

그림 8-32: 프로파일 데이터 - profile_call



profile_call 옵션에 설정된 메소드는 프로파일 데이터에 CALLED를 접두사로 표시된다. CALLED pkg.ClassE.run 프로파일 항목 Gap 시간이 단계 1의 수행 시간이고, CALLED pkg.ClassF.run 프로파일 항목 Gap 시간이 단계 2의 수행 시간이 된다. profile_call 옵션을 수정하면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다. 따라서 이 기능은 운영 시스템보다는 테스트 시스템에서 활용하는 것을 권장한다.

JSP 파일을 구간별로 추적할 때는 profile_call 옵션보다는 프로파일 API를 직접 사용하는 것을 권장한다. 자세한 사항은 [프로파일에 메세지 추가]를 참조한다.

8.4.8. 사용자 아이디 추출

특정 메소드의 파라미터 값이나 반환 값을 사용자 아이디로 추출할 수 있다.

특정 메소드의 파라미터 값을 사용자 아이디로 추출하려면 제니퍼 에이전트의 userid_param 옵션을 설정한다. 해당 메소드의 java.lang.String 유형의 첫번째 파라미터를 사용자 아이디로 추출한다.

```
userid_param = pkg.UserManager.setId(String)
```

특정 메소드의 반환 값을 사용자 아이디로 추출하려면 제니퍼 에이전트의 userid_return 옵션을 설정한다. 이 때 메소드의 반환 유형은 java.lang.String 이어야 한다.

```
userid_return = pkg.UserManager.getId()
```

이렇게 추출한 사용자 아이디는 X-View에서 확인할 수 있다.

8.5. 자바로딩 클래스 확인

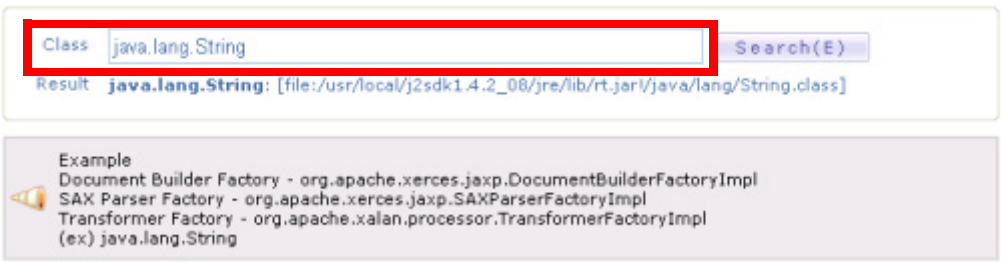
정확한 트랜잭션 프로파일링을 위해서는 클래스가 어느 JAR 파일에서 로딩되었는지와 다른 클래스 혹은 인터페이스와의 종속 관계가 어떠한지를 확인할 필요가 있다.

8.5.1. 클래스/Jar 체크

[장애 진단 | 유틸리티 | 클래스/Jar 검색] 메뉴에서 특정 클래스가 모니터링 대상 자바 애플리케이션의 어느 위치에 어떤 형태로 존재하는지를 검색할 수 있다. 클래스는 JAR 파일 안에 포함되어 있을 수도 있고 특정 디렉토리에 있을 수도 있다.

이름이 동일한 클래스가 여러 곳에 존재할 수 있는데 이 기능을 통해서 모니터링 대상 자바 애플리케이션이 실제로 사용하는 클래스 위치를 손쉽게 파악할 수 있다.

그림 8-33: 클래스/Jar 검색



클래스 필드에 자바 패키지 이름을 포함한 클래스 이름을 입력한 후에 **[검색]** 버튼을 누르면 모니터링 대상 자바 애플리케이션이 실제로 사용하고 있는 클래스 위치가 나타난다.

앞의 예제는 java.lang.String 클래스가 모니터링 대상 자바 애플리케이션의 /usr/local/j2sdk1.4.2_08/jre/lib 디렉토리에 위치한 rt.jar 파일 속에 있음을 보여준다.

8.5.2. 로딩 클래스 목록

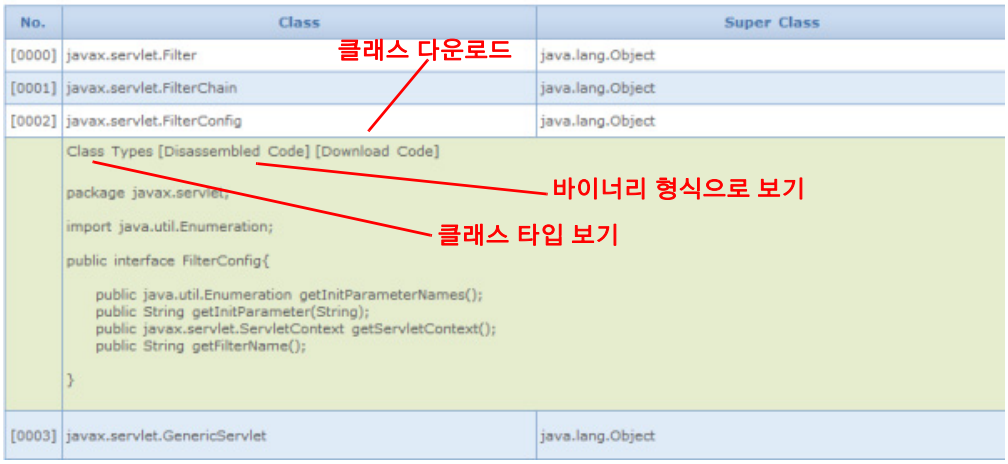
[장애 진단 | 유틸리티 | 로딩 클래스 목록] 메뉴에서 모니터링 대상 자바 애플리케이션에서 로딩된 모든 클래스 목록을 확인할 수 있다.

그림 8-34: 로딩 클래스 목록

- 클래스 이름 - 로딩된 클래스 이름을 보여준다. 해당 이름을 클릭하면 클래스에 대한 상세 정보를 확인할 수 있다.
- 상위 클래스 - 해당 클래스의 상위 클래스 이름을 보여준다. 상위 클래스 이름을 클릭하면 해당 클래스로 이동한다.
- 인터페이스 - 해당 클래스의 인터페이스 이름을 보여준다. 인터페이스 이름을 클릭하면 해당 인터페이스로 이동한다.

클래스 이름을 클릭하여 클래스 상세 정보를 확인할 수 있다.

그림 8-35: 로딩 클래스 상세 정보



- 클래스 타입 보기 - 해당 링크를 클릭하면 해당 클래스 기본 골격이 나타난다
- 바이너리 형식으로 보기 - 해당 링크를 클릭하면 해당 클래스를 바이너리 형식으로 보여준다.
- 클래스 다운로드 - 해당 링크를 클릭하면 해당 클래스 파일을 다운로드한다.

9

자바 시스템에서 리소스, 외부 트랜잭션, DB 연결 모니터링

CPU와 메모리 등의 시스템 자원을 모니터링하는 방법을 설명한다. 그리고 외부 애플리케이션과의 연동을 모니터링하는 외부 트랜잭션과 JDBC 모니터링을 설명한다.

9.1. CPU 모니터링

제니퍼 에이전트를 통해서 자바 애플리케이션이 동작하는 하드웨어의 시스템 CPU 사용률과 해당 자바 애플리케이션이 사용하는 자바 프로세스 CPU 사용률을 모니터링할 수 있다. 그리고 특정 트랜잭션의 처리 과정에서 사용한 트랜잭션 CPU 점유 시간도 모니터링할 수 있다.

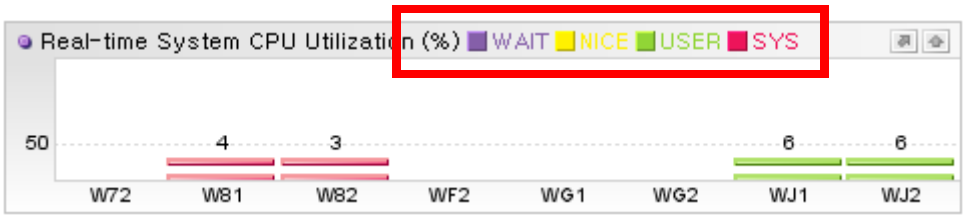
또한 제니퍼 에이전트의 설치와 상관없이 WMOND를 통해서 임의의 하드웨어의 CPU 개수당 CPU 사용률도 모니터링할 수 있다.

9.1.1. 시스템 CPU 사용률과 자바 프로세스 CPU 사용률

시스템 CPU 사용률은 시스템의 전체 CPU 사용률을 의미하고, 자바 프로세스 CPU 사용률은 제니퍼 에이전트를 설치한 자바 애플리케이션이 사용하는 CPU 사용률을 의미한다.

시스템 CPU 사용률과 자바 프로세스 CPU 사용률은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 값을 이퀄라이저 차트와 런타임 라인 차트를 통해서 확인할 수 있다.

그림 9-1: 실시간 시스템 CPU 사용률 차트



그런데 CPU 사용률은 CPU 사용 영역별로 구분되고, 이퀄라이저 차트는 CPU 사용률을 영역별로 색상을 구분하여 표시한다. 다음은 CPU 사용 영역에 대한 설명이다.

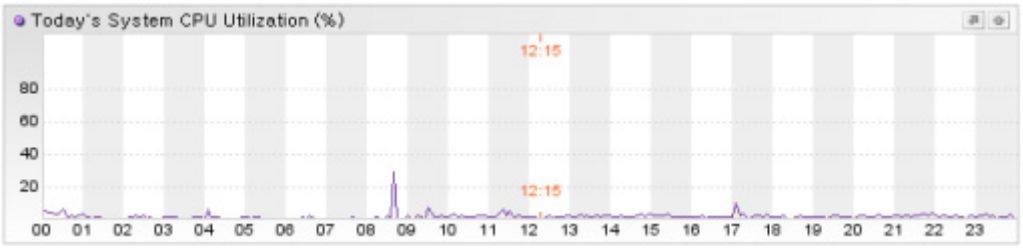
표 9-1: CPU 사용 영역

CPU 사용 영역	설명
WAIT	I/O 대기와 관련한 CPU 사용률을 의미한다.
NICE	NICE 우선 순위가 부여된 사용자 수준(애플리케이션)에서 사용하는 CPU 사용률을 의미한다.
USER	사용자 수준(애플리케이션)에서 사용하는 CPU 사용률을 의미한다.
SYS	시스템 수준(커널)에서 사용하는 CPU 사용률을 의미한다.

Notice: 이퀄라이저 차트는 자바 프로세스 CPU 사용률과 WMOND가 수집한 CPU 개수당 CPU 사용률도 CPU 사용 영역별로 색상을 구분하여 표시한다.

또한 시스템 CPU 사용률과 자바 프로세스 CPU 사용률에 대한 5분 평균 값은 PERF_X_01~31 테이블의 SYS_CPU 칼럼과 JVM_CPU 칼럼에 저장되고, 특정 날짜의 CPU 사용률 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 9-2: 금일 시스템 CPU 사용률



9.1.2. 트랜잭션 CPU Time

트랜잭션 CPU 점유 시간은 트랜잭션이 수행되는 과정에서 사용한 CPU 사용 시간을 의미한다. 이는 애플리케이션 처리 현황 통계 데이터와 X-View 프로파일 데이터로 수집된다.

그리고 tpmC를 통해서 트랜잭션 CPU 점유 시간을 하드웨어 별로 객관적으로 비교할 수 있다. tmpC는 TPC(Transaction Processing Performance Council, http://www.tpc.org)의 TPC-C 벤치마크 시나리오에 대한 1분당 최대 처리 건수를 나타내는 수치로써, 하드웨어(주로 CPU)의 성능을 측정하는 대표적인 방법이다.

하드웨어의 CPU 처리 용량이 다르기 때문에 트랜잭션 CPU 점유 시간만으로는 해당 트랜잭션이 성능에 미치는 영향을 객관적으로 파악할 수 없다. 예를 들어, CPU 처리 용량이 큰 하드웨어에서 처리된 트랜잭션 A의 CPU 점유 시간이 CPU 처리 용량이 작은 하드웨어에서 처리된 트랜잭션 B의 CPU 점유 시간보다 작은 경우에도 성능에 미치는 절대적인 영향은 트랜잭션 A가 더 높을 수도 있다. 즉 트랜잭션 A를 CPU 처리 용량이 작은 하드웨어에서 처리하면 트랜잭션 B보다 CPU 점유 시간이 클 수 있다는 것이다.

따라서 tpmC로 하드웨어의 CPU 처리 용량 변수를 통제한 상태에서 트랜잭션이 성능에 미치는 영향을 파악해야 한다.

이를 위해서 제니퍼 에이전트의 approximate_tpmc_on_this_system 옵션으로 제니퍼 에이전트를 설치한 해당 하드웨어의 tpmC 값을 설정하도록 한다.

```
approximate_tpmc_on_this_system = 30000
```

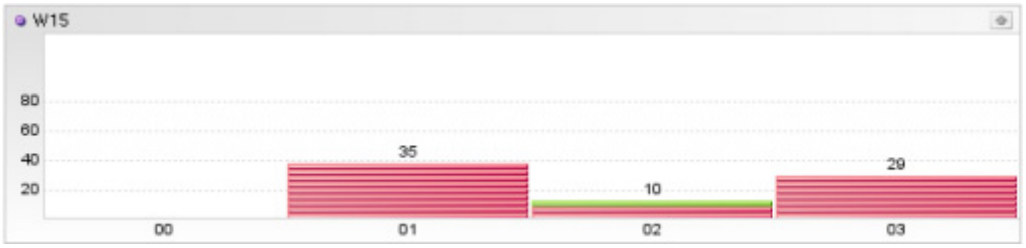
9.1.3. WMOND를 통한 CPU 개수당 CPU 사용률

WMOND는 임의의 하드웨어의 CPU 개수당 CPU 사용률 데이터를 수집하는 모듈이다. 이 모듈은 C 언어로 개발되어 있고 사용 방법이 간단한다. 제니퍼 에이전트 설치와는 상관이 없기 때문에 웹 서버나 데이터베이스 서버가 운영중인 하드웨어의 CPU 사용률을 모니터링하는데 적합하다.

Notice: WMOND는 하드웨어에 설치된 CPU 개수당 CPU 사용률을 수집하기 때문에 제니퍼 에이전트를 설치한 하드웨어에서도 WMOND를 사용할 필요가 있을 수 있다.

CPU 개수당 CPU 사용률은 이퀄라이저 차트를 통해서 확인할 수 있다. 다음 그림을 보면 WMOND로 모니터링하는 하드웨어에 4개의 CPU가 설치되어 있음을 확인할 수 있다.

그림 9-3: CPU 개수당 CPU 사용률



또한 CPU 개수당 CPU 사용률은 PERF_HOST 테이블에 CPU 사용 영역별로 구분되어서 저장된다. 기본 저장 주기는 5분이고, 제니퍼 서버의 perf_host_update_interval 옵션으로 변경할 수 있다. 단위는 밀리 세컨드이다.

```
perf_host_update_interval = 300000
```

Notice: 60000(1분)에서 300000(5분) 사이의 값만 가능하다.

9.1.3.1. WMOND의 설치와 실행

WMOND는 C 언어로 개발되었기 때문에 운영 체계에 적합한 것을 설치해야 한다. 현재 마이크로소프트 윈도우즈, 선 솔라리스, HP HP-UX, IBM AIX 등과 다양한 리눅스 운영 체계에서 WMOND를 사용할 수 있다.

WMOND를 사용하려면, 우선 JENNIFER_HOME/agent/wmond 디렉토리에 있는 파일 중에서 운영 체계에 적합한 wmond 파일을 CPU 사용률을 모니터링할 하드웨어에 복사한다.

그리고 운영 체계가 유닉스나 리눅스인 경우에는 복사한 wmond 파일에 실행 권한을 부여한다. 예를 들어, wmond 파일을 /usr/bin 디렉토리에 복사하였으면 다음처럼 실행 권한을 부여한다.

```
chmod 755 /usr/bin/wmond
```

WMOND를 실행하는 방법은 다음과 같다.

```
wmond [제니퍼 서버 IP] [포트 번호] [WMOND 아이디]
```

- 제니퍼 서버 IP - WMOND가 수집한 CPU 사용률 데이터를 전송할 제니퍼 서버 IP 주소를 지정한다.
- 포트 번호 - 제니퍼 서버의 server_udp_listen_port 옵션으로 설정한 UDP 포트 번호를 지정한다. 기본은 6902이다.

- WMOND 아이디 - WMOND를 구분하기 위한 고유한 아이디로 다른 WMOND 아이디 혹은 제니퍼 에이전트 아이디와 중복되서는 안된다. 그리고 제니퍼 에이전트 아이디와 동일하게 WMOND 아이디는 영어와 숫자만으로 구성되어야 하여, 글자 수는 반드시 3 자이어야 한다.

다음은 WMOND실행의 예이다.

```
nohup wmond 127.0.0.1 6902 DB1 &
```

유닉스나 리눅스에서 시스템이 시작할 때 WMOND를 자동으로 구동시키거나, WMOND가 정지되었을 때 자동으로 재시작하려면 /etc/inittab 파일에 다음 내용을 기술한다.

```
wmond:2:respawn:/usr/bin/wmond 127.0.0.1 6902 DB1 > /dev/null 2>&1
```

솔라리스에서는 /etc/rc2.d/S97wmond 파일을 추가한다. 파일 이름은 고유해야 한다.

```
/usr/bin/wmond 127.0.0.1 6902 DB1 > /dev/null 2>&1 &
```

마지막의 [&]를 생략해서는 안된다.

9.1.3.2. WMOND의 정지와 경보

제니퍼 서버의 agent_death_detection_time 옵션으로 설정한 기간동안 WMOND로부터 CPU 사용률 데이터가 전송되지 않으면, 제니퍼 서버는 WMOND를 설치한 시스템(머신)이 정지된 것으로 간주하고 ERROR_SYSTEM_DOWN 경보를 발령한다. 기본 값은 8000이고 단위는 밀리 세컨드이다.

```
agent_death_detection_time = 8000
```

9.1.4. CPU 모니터링과 경보 발령

시스템 CPU 사용률과 자바 프로세스 CPU 사용률이 임계치를 초과하면 제니퍼 서버는 경보를 발령한다. 자세한 사항은 경보와 예외 모니터링을 참조한다.

9.1.5. CPU 모니터링과 관련한 주의 사항

9.1.5.1. 실제 CPU 사용률과 제니퍼가 수집한 CPU 사용률의 차이가 큰

경우

제니퍼는 CPU 모니터링을 위해서 운영 체계가 제공하는 커널 API를 사용한다. 그런데 간혹 IBM AIX에서 CPU가 멀티코어인 경우에, 실제 CPU 사용률과 제니퍼가 수집한 CPU 사용률이 50% 이상의 차이를 보이는 현상이 나타날 수 있다. 이 경우에는 tech@jennifersoft.com 메일로 기술 지원을 요청하도록 한다.

9.1.5.2. 제니퍼 에이전트 Native 모듈 테스트

제니퍼 에이전트의 Native 모듈이 CPU 사용률과 메모리 사용량 등의 데이터를 수집한다. 제니퍼 클라이언트에 CPU 사용률과 메모리 사용량 등이 나타나지 않으면 Native 모듈이 올바르게 설치되지 않았음을 의미한다.

Native 모듈을 올바르게 설치했다면 제니퍼 에이전트의 로그 파일과 자바 애플리케이션 표준 출력 파일에 다음 메시지가 기록된다.

```
libjennifer20.so(sl) shared library loaded successfully.
```

만약 로그 파일에 앞의 메시지가 기록되지 않으면, Native 모듈이 잘못 설정되었거나 해당 시스템에 맞지 않는 Native 모듈을 사용하고 있음을 의미한다. 이런 경우에는 시스템에 맞는 Native 모듈을 사용해야 한다.

시스템에 적합한 Native 모듈을 찾기 위해서 제니퍼 에이전트와는 별도로 Native 모듈을 테스트할 수 있는 방법을 제공한다.

- Native 모듈은 JENNIFER_HOME/agent/jni 디렉토리에 운영 체계 별로 존재한다. 운영 체계에 맞는 디렉토리로 이동한다.
- 디렉토리에는 운영 체계 버전과 CPU에 따른 여러 개의 Native 모듈이 존재한다. 이 중에서 시스템에 적합한 모듈의 이름을 libjennifer20.so(sl)로 변경한다. 마이크로소프트 윈도우즈의 경우에는 jennifer20.dll로 변경한다.
- 유닉스 혹은 리눅스인 경우에는 Native 모듈에 실행 권한을 부여한다.
- test.sh 유틸리티를 이용해서 Native 모듈을 테스트한다. 이 유틸리티는 자바를 사용한다. 따라서 제니퍼 에이전트로 모니터링하는 자바 애플리케이션이 사용하는 JVM을 패스에 등록해야 한다. 그리고 Native 모듈과 test.sh 유틸리티가 동일 디렉토리에 있지 않으면 test.sh 파일을 열고 -Djava.library.path과 관련한 내용을 수정한다.

- test.sh 유틸리티를 실행하여 다음과 같은 메시지가 출력되면 Native 모듈이 정상적으로 동작하는 것이다.

```
jennifer@jennifer1:~/agent/jni/linux$ ./test.sh
get_nprocs_conf=2, get_nprocs(ncpu)=2, ncpu=2, _SC_CLK_TCK=100

Jennifer TimeZoneOffset:32400000
Jennifer4.0.0.2(2008-09-22) libjennifer20.so(sl) shared library loaded
successfully.
 USER   SYS  NICE  WAIT  IDLE USER   SYS  NICE  WAIT  IDLE
get_pid=30449
get_ppid=30447
get_ncpu=2
get_cpucluck=100
get_native_thread_id=-1209939056
get_current_thread_cpu_time=0
get_thread_cpu_time=0
get_total_mem=1059909632
get_free_mem=16814080
get_jvm_mem=217346048
  0.0   0.0   0.0   0.0 100.0   0.0   0.0   0.0   0.0 100.0
total_cpu: 19690350,2889545,42167,0,618107374,
jvm_total_cpu: 7,1,2021553758,0,-1,
get_current_thread_cpu_time=20
  0.0   0.0   0.0   0.0 100.0   0.0   0.0   0.0   0.0 100.0
total_cpu: 19690367,2889546,42167,0,618107561,
jvm_total_cpu: 8,1,2021553858,0,-1,
get_current_thread_cpu_time=30
  4.5   0.5   0.0   0.0  95.0   0.0   0.0   0.0   0.0 100.0
total_cpu: 19690376,2889547,42167,0,618107752,
jvm_total_cpu: 8,1,2021553958,0,-1,
get_current_thread_cpu_time=40
```

모든 파일을 테스트했는데도 정상적으로 동작하지 않는다면 Native 모듈을 해당 시스템에 맞게 재컴파일해야 한다. 이 경우에는 tech@jennifersoft.com 메일로 기술 지원을 요청하도록 한다.

9.2. 메모리 모니터링

제니퍼 에이전트를 통해서 자바 애플리케이션이 동작하는 하드웨어의 시스템 메모리 사용량과 해당 자바 애플리케이션이 사용하는 자바 프로세스 메모리 사용량을 모니터링할 수 있다. 그리고 자바 애플리케이션의 자바 힙 메모리 전체와 자바 힙 메모리 사용량도 모니터링할 수 있다.

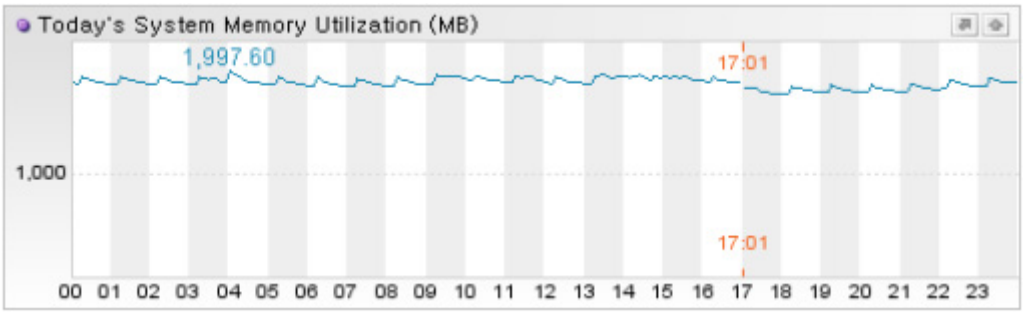
9.2.1. 시스템 메모리 사용량과 자바 프로세스 메모리 사용량

시스템 메모리 사용량은 시스템의 메모리 사용량을 의미하고, 자바 프로세스 메모리 사용량은 제니퍼 에이전트를 설치한 자바 애플리케이션이 사용하는 메모리 사용량을 의미한다. 단위는 MB이다.

시스템 메모리 사용량과 자바 프로세스 메모리 사용량은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 30초 평균 값을 런타임 라인 차트를 통해서 확인할 수 있다.

또한 시스템 메모리 사용량과 자바 프로세스 메모리 사용량에 대한 5분 평균 값은 PERF_X_01~31 테이블의 SYS_MEM_USED 칼럼과 JVM_NAT_MEM 칼럼에 저장되고, 특정 날짜의 메모리 사용량 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 9-4: 금일 시스템 메모리 사용량



Notice: 시스템 CPU 사용률과 동일하게 시스템 메모리 사용량과 자바 프로세스 메모리 사용량을 수집하려면 Native 모듈이 올바르게 설치되어 있어야 한다.

9.2.2. 자바 힙 메모리 전체와 자바 힙 메모리 사용량

자바 힙 메모리 전체는 다음 자바 코드를 통해서 얻을 수 있는 값을 의미한다.

```
Runtime runtime = Runtime.getRuntime();
long totalMemory = runtime.totalMemory();
```

그리고 자바 힙 메모리 사용량은 다음 자바 코드를 통해서 얻을 수 있는 값을 의미한다.

```
Runtime runtime = Runtime.getRuntime();
long usedMemory = runtime.totalMemory() - runtime.freeMemory();
```

앞의 코드를 통해서 수집한 값의 단위는 바이트이지만, 제니퍼는 이를 MB로 전환해서 사용한다.

자바 힙 메모리 전체와 자바 힙 메모리 사용량은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 30초 평균 값을 런타임 라인 차트를 통해서 확인할 수 있다.

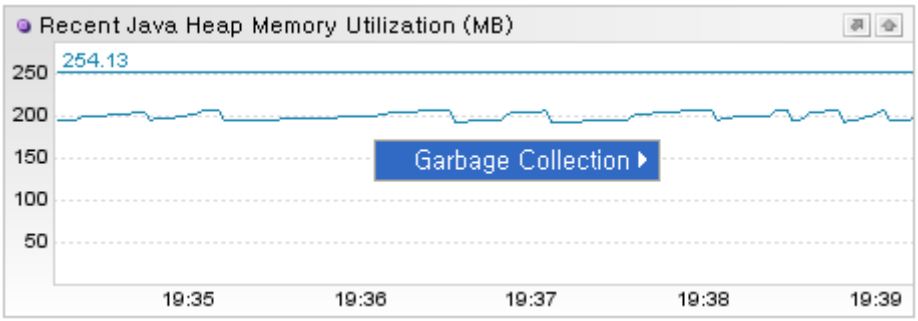
또한 자바 힙 메모리 전체와 자바 힙 메모리 사용량에 대한 5분 평균 값은 PERF_X_01~31 테이블의 HEAP_TOTAL 칼럼과 HEAP_USED 칼럼에 저장되고, 특정 날짜의 자바 힙 메모리 사용량 변화 추이는 라인 차트를 통해서 확인할 수 있다.

Notice: 자바 힙 메모리 사용률은 자바 힙 메모리 전체에 대한 자바 힙 메모리 사용량의 비율을 의미한다. 단위는 퍼센트이다. 그런데 자바 힙 메모리 전체가 거의 변하지 않기 때문에 라인 차트와 런타임 라인 차트에서 자바 힙 메모리 사용률과 자바 힙 메모리 사용량을 나타내는 선의 형태는 거의 유사하다.

자바 힙 메모리 전체, 자바 힙 메모리 사용량, 자바 힙 메모리 사용률 등을 나타내는 런타임 라인 차트에서는 특정 제니퍼 에이전트에 대한 가비지 콜렉션을 수행할 수 있다.

- 런타임 라인 차트를 클릭한 후에 오른쪽 마우스를 클릭하면 컨텍스트 메뉴가 나타난다.
- 컨텍스트 메뉴에서 **[가비지 콜렉션]** 메뉴를 선택한 후에 가비지 콜렉션을 수행할 제니퍼 에이전트를 선택하면 해당 제니퍼 에이전트에 대한 가비지 콜력션을 수행한다.

그림 9-5: 자바 힙 메모리 가비지 콜렉션



단, 사용자가 속한 그룹이 gc 권한을 가지고 있어야 컨텍스트 메뉴가 나타난다.

9.2.3. 메모리 모니터링과 경보 발령

제니퍼 서버는 임의의 기간 동안의 자바 힙 메모리 사용률이 임계치를 초과하면 WARNING_JVM_HEAP_MEM_HIGH 경보를 발령한다.

9.3. 메모리-콜렉션 모니터링

자바 애플리케이션에서 자바 힙 메모리 누수는 성능 저하를 유발하는 대표적인 현상이다. 제니퍼는 이 문제를 해결하기 위한 도구로 메모리-콜렉션 모니터링을 제공한다.

9.3.1. 메모리-콜렉션 모니터링에 대한 이해

메모리-콜렉션 모니터링은 java.util.Collection과 java.util.Map 인터페이스를 구현한 자바 콜렉션 객체 중에서 임계치 이상의 객체를 담고 있는 것을 추적하고 스텍트레이스를 기록하는 기능이다. 임계치는 제니퍼 에이전트의 lwst_collection_minimum_monitoring_size 옵션으로 설정한다. 기본 값은 3000이다.

예를 들어, 특정 트랜잭션에 의해 다음 자바 코드가 실행된다고 가정한다.

```
java.util.List books = new java.util.ArrayList();
for (int i = 0; i < 4000; i++) {
    books.add(new Book(i));
}
```

앞의 코드에 의하면 books 자바 콜렉션 객체에 4000개의 객체가 담겨지기 때문에, 제니퍼 에이전트는 books 자바 콜렉션 객체를 추적한다.

사용자는 이렇게 추적된 콜렉션 객체 목록을 **[장애 진단 | 메모리-콜렉션]** 메뉴에서 확인할 수 있다.

9.3.2. 메모리-콜렉션 모니터링 활성화

기본적으로 메모리-콜렉션 모니터링은 동작하지 않는다. 이 기능을 사용하려면 build_collection 패치 옵션을 true로 설정해서 LWST 빌드를 다시 해야 한다. 자세한 사항은 LWST빌드와 설치를 참조한다.

기본으로 메모리-콜렉션 모니터링을 사용하지 않는 이유는, 다른 모니터링 기능에 비해서 이 모니터링이 야기하는 성능 저하가 클 수 있기 때문이다. 따라서 자바 힙 메모리 누수와 관련한 문제를 해결할 때만 사용하고, 그 이후에는 build_collection 패치 옵션을 false로 설정해서 LWST 빌드를 다시 하도록 한다.

9.3.3. 메모리-콜렉션 모니터링 활용

모든 메모리-콜렉션 모니터링은 **[장애 진단 | 메모리-콜렉션]** 메뉴에서 활용한다. 이 메뉴에서 자바 콜렉션 객체에 담겨 있는 객체 개수의 변화량을 확인하고, 자바 콜렉션 객체를 사용하는 트랜잭션의 스택트레이스를 추출할 수 있다.

먼저 제니퍼 에이전트 선택 영역에서 특정 제니퍼 에이전트를 선택하면, 제니퍼 에이전트의 lwst_collection_minimum_monitoring_size 옵션으로 설정한 크기보다 많은 객체를 담고 있는 자바 콜렉션 객체 목록이 나타난다. 기본 값은 3000이다.

```
lwst_collection_minimum_monitoring_size = 3000
```

그리고 lwst_collection_minimum_monitoring_size 옵션을 직접 수정하지 않고, [콜렉션 최소 모니터링 크기] 검색 조건에서 임계치를 임시적으로 수정할 수 있다. [콜렉션 최소 모니터링 크기] 검색 조건에 임의의 값을 입력한 후에 **[검색]** 버튼을 클릭하면 된다.

그림 9-6: 메모리-콜렉션 모니터링 화면

다음은 메모리-콜렉션 모니터링 항목에 대한 설명이다.

표 9-2: 메모리 - 콜렉션 모니터링 항목

항목	설명
번호	일련 번호
생성 시간	자바 콜렉션 객체의 생성 시간
전체	자바 콜렉션 객체에 담겨 있는 객체의 개수로, java.util.Collection이나 java.util.Map 인터페이스의 size 메소드가 반환하는 값이다.
델타	자바 콜렉션 객체에 담겨 있는 객체 개수의 변화량

표 9-2: 메모리 - 콜렉션 모니터링 항목

항목	설명
콜렉션 이름	자바 콜렉션 클래스의 이름
해시 코드	자바 콜렉션 객체의 해시 코드
스택트레이스	자바 콜렉션 객체를 사용하는 트랜잭션에서 추출한 스택트레이스
애플리케이션	자바 콜렉션 객체를 사용하는 트랜잭션의 애플리케이션 이름

특정 자바 콜렉션 객체에 담겨 있는 객체 개수의 변화량을 확인하려면 다음과 같이 한다.

- 우선 **[델타 값 초기화]** 버튼을 클릭한다.
- 그리고 반복적으로 **[검색]** 버튼을 누르면서 자바 콜렉션 객체에 담겨 있는 객체 개수의 변화량을 델타 항목을 통해서 확인한다.

그리고 자바 콜렉션 객체를 사용하는 트랜잭션의 스택트레이스를 추출할 수 있다. 하나의 자바 콜렉션 객체에 대한 스택트레이스를 추출하려면 다음 사항을 참고한다.

- 기본적으로 자바 콜렉션 객체의 스택트레이스 항목에는 **[받기]** 링크가 나타난다.
- 스택트레이스를 기록하려면 이 **[받기]** 링크를 클릭한다. 그러면 **[받기]** 링크가 **[대기]** 링크로 변한다.
- **[대기]** 링크는 스택트레이스를 추출하기 위해서 기다리고 있다는 뜻이다. 이렇게 기다리는 이유는 특정 트랜잭션이 해당 자바 콜렉션 객체를 사용해야지만 스택트레이스가 기록되기 때문이다.
- 즉, 특정 트랜잭션이 자바 콜렉션 객체를 사용하지 않으면 스택트레이스가 추출되지 않는다. 따라서 특정 트랜잭션이 자바 콜렉션 객체를 사용하여 스택트레이스가 추출될 때까지 **[검색]** 버튼을 반복적으로 클릭한다.
- 스택트레이스가 추출되면 스택트레이스 항목에 **[지우기]** 링크와 **[다시 받기]** 링크가 나타나고 하단에는 스택트레이스 정보가 나타난다.
- **[지우기]** 링크를 클릭하면 추출한 스택트레이스가 삭제되고, **[받기]** 링크가 나타난다.
- **[다시 받기]** 링크를 클릭하면 추출한 스택트레이스가 삭제되고, **[대기]** 링크가 나타난다. 즉 새로운 트랜잭션에 의한 스택트레이스 추출을 기다리게 된다.

Notice: 특정 트랜잭션에 의해서 스택트레이스가 추출되면, 이후에 다른 트랜잭션이 해당 자바 콜렉션 객체를 사용해도 스택트레이스가 다시 추출되지는 않는다. 따라서 새로운 스택트레이스를 기록하려면 **[다시 받기]** 링크를 클릭한다.

모든 자바 콜렉션 객체에 대한 스택트레이스를 추출하려면 다음 사항을 참고한다.

- 모든 자바 콜렉션 객체에 대한 스텍트레이스를 추출하려면 상단에 있는 **[모든 스택 받기]** 버튼을 클릭한 후에, 반복적으로 **[검색]** 버튼을 누르면서 스택트레이스 추출 여부를 확인한다.

- 추출한 모든 스택트레이스를 삭제하려면 **[스택트레이스 정리]** 버튼을 클릭한다.

Notice: 추출된 스택트레이스는 제니퍼 에이전트를 설치한 자바 애플리케이션의 자바 힙 메모리에 상주한다. 따라서 분석을 완료한 후에는 모든 스택트레이스를 삭제하는 것을 권장한다.

그런데 사용자가 조작을 하지 않아도, 자바 콜렉션 객체에 담겨 있는 객체의 개수가 임계치를 초과하면 자동으로 스택트레이스를 추출한다. 임계치는 제니퍼 에이전트의 lwst_collection_auto_stacktrace_size 옵션으로 설정한다. 기본 값은 100000이다.

```
lwst_collection_auto_stacktrace_size = 100000
```

그리고 lwst_collection_auto_stacktrace_size 옵션을 직접 수정하지 않고, [콜렉션 자동 스택트레이스 크기] 검색 조건에서 임계치를 임시적으로 수정할 수 있다. [콜렉션 자동 스택트레이스 크기] 검색 조건에 임의의 값을 입력한 후에 **[검색]** 버튼을 클릭하면 된다.

9.4. 파일/소켓 모니터링

자바 애플리케이션에서 IO는 성능 저하를 유발하는 원인이 될 수 있다. 제니퍼는 이 문제를 해결하기 위한 도구의 하나로 파일/소켓 모니터링을 제공한다.

9.4.1. 파일 모니터링

[장애 진단 | 파일/소켓] 메뉴를 통해서 액티브한 자바 애플리케이션의 파일 IO 현황을 모니터링할 수 있다. 파일 IO 현황을 모니터링하려면 build_file 패치 옵션을 true로 설정해서 LWST 빌드를 해야 한다. 기본이 true이다. 자세한 사항은 LWST빌트와 설치를 참조한다.

그림 9-7: 파일 IO 목록

No.	Last Modified Time	File Name	Size (byte)	Mode	Concurrent Access Count
[0000]	2008-10-01/04:04:25	/home/log/webt.log	0	WRITE	1
[0001]	2007-07-16/13:28:04	DBAgentManager.log	0	WRITE	24
[0002]	2007-07-16/13:28:05	ErrorMessage.log	0	WRITE	25
[0003]	2002-08-31/08:31:36	/dev/random	0	READ	1
[0004]	2008-09-02/14:08:51	TLConnectionManager.log	16,978,582	WRITE	25

다음은 파일 IO 목록 항목에 대한 설명이다.

표 9-3: 파일 IO 목록 항목

칼럼 이름	설명
번호	일련 번호
마지막 수정 시간	파일의 마지막 수정 시간
파일명	파일의 절대 경로
크기	파일의 크기로 단위는 바이트이다.
모드	작업 상태를 나타내는 코드로, READ는 읽기를, WRITE는 쓰기를 의미한다.
동시 접근 수	파일의 동시 접근 수

파일 IO 모니터링을 통해서 자바 애플리케이션이 불필요한 파일 IO를 수행하고 있는지를 확인 할 수 있다.

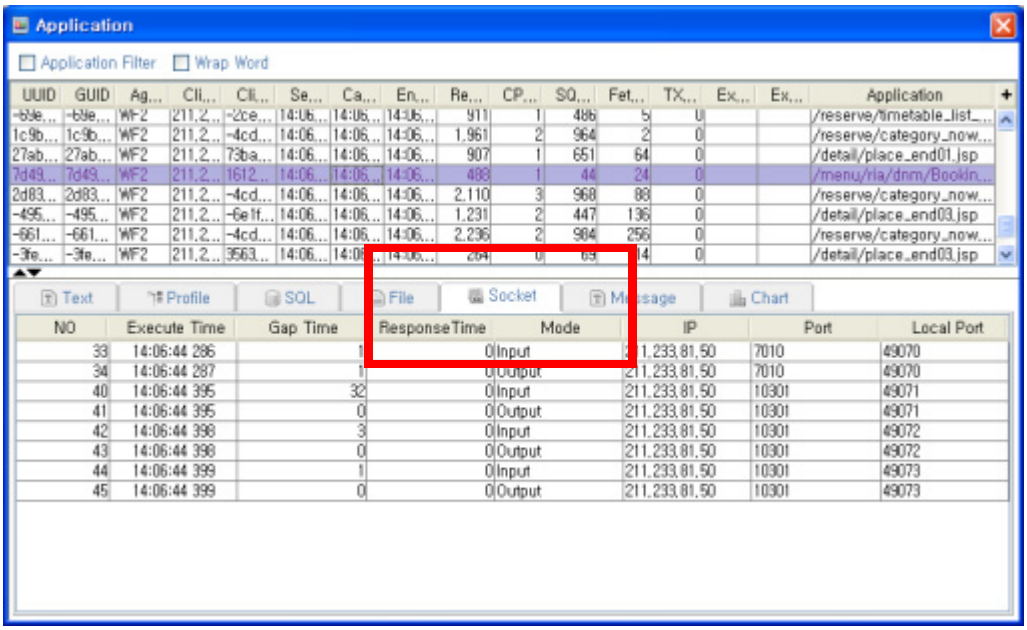
Notice: 파일 IO 모니터링은 java.io.FileInputStream과 java.io.FileOutputStream 클래스로 접근한 파일만을 감지한다. 따라서 JNI를 이용하여 C 프로그램으로 파일을 열면, 해당 파일은 IO 목록에 나타나지 않는다.

9.4.2. 소켓 모니터링

소케 모니터링은 자바 애플리케이션의 java.net.Socket 클래스를 통한 소켓 IO 현황을 제공한다. 소켓 IO 현황을 모니터링하려면 build_socket 패치 옵션을 true로 설정해서 LWST 빌드를 해야 한다. 기본이 true이다. 자세한 사항은 LWST빌드와 설치를 참조한다.

기본적으로 특정 트랜잭션이 수행한 소켓 IO 작업 정보가 X-View 프로파일 데이터에 나타난다.

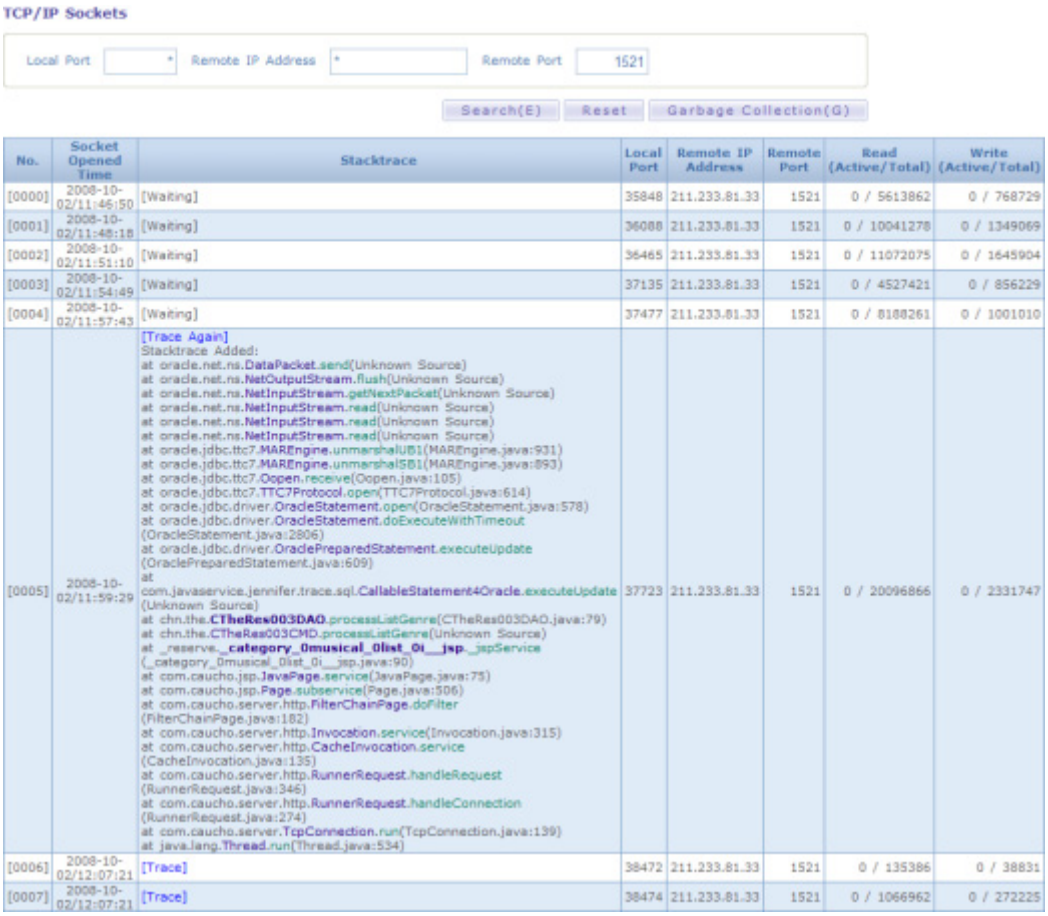
그림 9-8: X-View 프로파일 - 소켓



그리고 제니퍼 에이전트의 socket_simple_trace 옵션을 false로 하면 **[장애 진단 | 파일/소켓]** 메뉴를 통해서 액티브한 자바 애플리케이션의 소켓 IO 현황을 모니터링할 수 있다. 기본 값은 true이다.

```
socket_simple_trace = false
```

그림 9-9: 소켓 IO 목록



No.	Socket Opened Time	Stacktrace	Local Port	Remote IP Address	Remote Port	Read (Active/Total)	Write (Active/Total)
[0000]	2008-10-02/11:46:50	[Waiting]	35848	211.233.81.33	1521	0 / 5613862	0 / 768729
[0001]	2008-10-02/11:48:18	[Waiting]	36088	211.233.81.33	1521	0 / 10041278	0 / 1349069
[0002]	2008-10-02/11:51:10	[Waiting]	36465	211.233.81.33	1521	0 / 11072075	0 / 1645904
[0003]	2008-10-02/11:54:49	[Waiting]	37135	211.233.81.33	1521	0 / 4527421	0 / 856229
[0004]	2008-10-02/11:57:43	[Waiting]	37477	211.233.81.33	1521	0 / 8188261	0 / 1001010
[0005]	2008-10-02/11:59:29	[Trace Again] Stacktrace Added: at oracle.net.ns.DataPacket.send(Unknown Source) at oracle.net.ns.NetOutputStream.flush(Unknown Source) at oracle.net.ns.NetInputStream.getNextPacket(Unknown Source) at oracle.net.ns.NetInputStream.read(Unknown Source) at oracle.net.ns.NetInputStream.read(Unknown Source) at oracle.net.ns.NetInputStream.read(Unknown Source) at oracle.jdbc.ttc7.MAREngine.unmarshalUB1(MAREngine.java:931) at oracle.jdbc.ttc7.MAREngine.unmarshalSB1(MAREngine.java:893) at oracle.jdbc.ttc7.Oopen.receive(Oopen.java:105) at oracle.jdbc.ttc7.TTC7Protocol.open(TTC7Protocol.java:614) at oracle.jdbc.driver.OracleStatement.open(OracleStatement.java:578) at oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout (OracleStatement.java:2806) at oracle.jdbc.driver.OraclePreparedStatement.executeUpdate (OraclePreparedStatement.java:609) at com.javaservice.jennifer.trace.sql.CallableStatement4Oracle.executeUpdate (Unknown Source) at chn.the.CTheRes003DAO.processListGenre(CTheRes003DAO.java:79) at chn.the.CTheRes003CMD.processListGenre(Unknown Source) at _reserve._category_0musical_0list_0i__jsp._jspService (_category_0musical_0list_0i__jsp.java:90) at com.caucho.jsp.JavaPage.service(JavaPage.java:75) at com.caucho.jsp.Page.subservice(Page.java:506) at com.caucho.server.http.FilterChainPage.doFilter (FilterChainPage.java:182) at com.caucho.server.http.Invocation.service(Invocation.java:315) at com.caucho.server.http.CacheInvocation.service (CacheInvocation.java:135) at com.caucho.server.http.RunnerRequest.handleRequest (RunnerRequest.java:346) at com.caucho.server.http.RunnerRequest.handleConnection (RunnerRequest.java:274) at com.caucho.server.TcpConnection.run(TcpConnection.java:139) at java.lang.Thread.run(Thread.java:534)	37723	211.233.81.33	1521	0 / 20096866	0 / 2331747
[0006]	2008-10-02/12:07:21	[Trace]	38472	211.233.81.33	1521	0 / 135386	0 / 38831
[0007]	2008-10-02/12:07:21	[Trace]	38474	211.233.81.33	1521	0 / 1066962	0 / 272225

다음은 소켓 IO 목록 항목에 대한 설명이다.

표 9-4: 소켓 IO 목록 항목

항목	설명
번호	일련 번호
소켓이 열린 시간	소켓이 열린 시간
스택트레이스	소켓을 사용하는 트랜잭션에서 추출한 스택트레이스
로컬 포트	소켓의 로컬 포트 번호
원격 IP 주소	소켓의 원격 서버 IP 주소
원격 포트	소켓의 원격 서버 포트 번호
읽기(활성/전체)	소켓을 통해서 읽은 데이터의 크기를 표시한다. 단위는 바이트이다. 활성은 최근에 읽은 데이터의 크기를, 전체는 소켓을 통해서 읽은 누적된 모든 데이터의 크기를 의미한다.

표 9-4: 소켓 IO 목록 항목

항목	설명
쓰기(활성/전체)	소켓을 통해서 쓴 데이터의 크기를 표시한다. 단위는 바이트이다. 활성은 최근에 쓴 데이터의 크기를, 전체는 소켓을 통해서 쓴 누적된 모든 데이터의 크기를 의미한다.

소켓 IO 목록에 표시된 소켓 IO 현황이 많은 경우에는 [로컬 포트], [원격 IP 주소] 그리고 [원격 포트] 등의 조건으로 검색을 할 수 있다. **[검색]** 버튼 옆의 **[초기화]** 버튼을 클릭하면 검색 조건이 초기화된다.

그리고 소켓을 사용하는 트랜잭션의 스택트레이스를 추출할 수 있다. 하나의 소켓에 대한 스택트레이스를 추출하려면 다음 사항을 참고한다.

- 기본적으로 소켓의 스택트레이스 항목에는 **[받기]** 링크가 나타난다.
- 스택트레이스를 기록하려면 이 **[받기]** 링크를 클릭한다. 그러면 **[받기]** 링크가 **[대기]** 링크로 변한다.
- **[대기]** 링크는 스택트레이스를 추출하기 위해서 기다리고 있다는 뜻이다. 이렇게 기다리는 이유는 특정 트랜잭션이 해당 소켓을 사용해야지만 스택트레이스가 기록되기 때문이다.
- 즉, 특정 트랜잭션이 소켓을 사용하지 않으면 스택트레이스가 추출되지 않는다. 따라서 특정 트랜잭션이 소켓을 사용하여 스택트레이스가 추출될 때까지 **[검색]** 버튼을 반복적으로 클릭한다.
- 스택트레이스가 추출되면 스택트레이스 항목에 **[지우기]** 링크와 **[다시 받기]** 링크가 나타나고 하단에는 스택트레이스 정보가 나타난다.
- **[지우기]** 링크를 클릭하면 추출한 스택트레이스가 삭제되고, **[받기]** 링크가 나타난다.
- **[다시 받기]** 링크를 클릭하면 추출한 스택트레이스가 삭제되고, **[대기]** 링크가 나타난다. 즉 새로운 트랜잭션에 의한 스택트레이스 추출을 기다리게 된다.

Notice: 특정 트랜잭션에 의해서 스택트레이스가 추출되면, 이후에 다른 트랜잭션이 해당 소켓을 사용해도 스택트레이스가 다시 추출되지는 않는다. 따라서 새로운 스택트레이스를 기록하려면 **[다시 받기]** 링크를 클릭한다.

소켓 모니터링을 제니퍼 에이전트의 JDBC 커넥션 추적용 설정정보를 획득하기 위한 수단으로도 사용할 수 있다. JDBC 모니터링을 하려면 java.sql.Connection 객체를 제공하는 클래스를 찾아야 한다. 그런데 이를 명확하게 알 수 없다면 데이터베이스 서버의 원격 IP 주소와 원격 포트 번호를 이용해서 데이터베이스 연결을 담당하는 소켓을 찾아내서, 해당 소켓을 사용하는 트랜잭션의 스택트레이스를 추출하면 어느 클래스에서 java.sql.Connection 객체를 반환하는지를 파악할 수 있다.

사용자가 조작을 하지 않아도, 제니퍼 에이전트의 lwst_trace_remote_port 옵션으로 설정한 원격 포트에 대해서는 자동으로 스택트레이스를 추출한다. 예를 들어, 외부 EAI 솔루션

의 2300 포트와 소켓 통신을 하는 자바 애플리케이션의 소켓에 대한 스택트레이스를 자동으로 기록하려면 다음과 같이 설정한다.

```
lwst_trace_remote_port = 2300
```

이 옵션으로 특정 원격 포트 번호를 설정했다면 socket_simple_trace 옵션을 true로 설정해도 해당 원격 포트를 사용하는 소켓에 대해서는 스택트레이스가 추출된다.

Notice: 제니퍼 에이전트의 lwst_trace_remote_port 옵션을 변경해도 자바 애플리케이션을 재시작할 필요는 없다. 그러나 이 옵션으로 설정한 원격 포트 번호를 사용하는 소켓 객체가 이미 만들어져서 풀링된 상태라면 해당 소켓 객체에 대한 스택트레이스를 추출할 수는 없다. 따라서 이 경우에는 자바 애플리케이션을 재시작해야 한다.

9.5. 라이브 오브젝트 모니터링

라이브 오브젝트 모니터링은 특정 클래스의 객체 개수를 파악하는 기능이다.

9.5.1. 라이브 오브젝트 모니터링 설정

불필요한 성능 저하를 방지하기 위해서 모든 클래스의 객체 개수를 파악하지는 않고, 제니퍼 에이전트의 liveobject로 시작하는 옵션을 통해서 설정한 클래스들에 대해서만 라이브 오브젝트 모니터링을 한다. 수정한 liveobject로 시작하는 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

예를 들어, java.util.ArrayList 클래스의 객체 개수를 파악하려면 다음과 같이 설정한다.

```
liveobject_class = java.util.ArrayList
```

두 개 이상의 클래스는 세미콜론[;]을 구분자로 구분한다. liveobject로 시작하는 모든 옵션도 마찬가지 이다. 따라서 java.util.HashMap 클래스의 객체 개수도 파악하려면 다음과 같이 설정한다.

```
liveobject_class = java.util.ArrayList;java.util.HashMap
```

특정 클래스를 상속한 모든 클래스의 객체 개수를 파악하려면 제니퍼 에이전트의 liveobject_super 옵션을 사용한다. 예를 들어, java.lang.Thread 클래스의 하위 클래스의 객체 개수를 파악하려면 다음과 같이 설정한다.

```
liveobject_super = java.lang.Thread
```

그러나 이 경우에 직접적으로 상속받은 클래스의 객체 개수만을 모니터링한다. 예를 들어, A 클래스가 java.lang.Thread 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스의 객체 개수는 모니터링하지만 B 클래스의 객체 개수는 모니터링하지 않는다.

특정 인터페이스를 구현한 모든 클래스의 객체 개수를 파악하려면 제니퍼 에이전트의 liveobject_interface 옵션을 사용한다. 예를 들어, java.lang.Runnable과 java.sql.Connection 인터페이스를 구현한 클래스의 객체 개수를 파악하려면 다음과 같이 설정한다.

```
liveobject_interface = java.lang.Runnable;java.sql.Connection
```

그러나 이 경우에 직접적으로 상속받은 클래스의 객체 개수만을 모니터링한다. 예를 들어, A 클래스가 java.lang.Thread 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스의 객체 개수는 모니터링하지만 B 클래스의 객체 개수는 모니터링하지 않는다. 특정 인터페이스를 구현한 모든 클래스의 객체 개수를 파악하려면 제니퍼 에이전트의 liveobject_interface 옵션을 사용한다. 예를 들어, java.lang.Runnable과 java.sql.Connection 인터페이스를 구현한 클래스의 객체 개수를 파악하려면 다음과 같이 설정한다.

```
liveobject_prefix = java.util
```

특정 이름으로 끝나는 클래스들의 객체 개수만을 모니터링할 수 있다. 예를 들어, Factory로 끝나는 모든 클래스의 객체 개수를 파악하려면 다음과 같이 설정한다.

```
liveobject_postfix = Factory
```

9.6. HTTP 세션 모니터링

HTTP 세션은 웹 애플리케이션에서 사용자 상태를 유지하기 위한 방법이다. 자바에서 HTTP 세션은 javax.servlet.http.HttpSession 인터페이스에 대한 구현 클래스로 WAS 마다 상이하다.

9.6.1. Dummy HTTP 세션 추적

제니퍼 에이전트는 page 속성 session을 설정하지 않았거나 true로 설정한 JSP가 수행되는 과정에서 HTTP 세션 객체가 만들어지면 WARNING_DUMMY_HTTPSESSION_CREATED 예외를 발생시킨다. 대부분의 환경에서는 이 예외가 적합하지 않기 때문에 기본적으로 이 예외는 발생하지 않는다. 이 예외를 발생시키려면 제니퍼 에이전트의 enable_dummy_httpsession_trace 옵션을 true로 설정한다. 기본 값은 false이다.

```
enable_dummy_httpsession_trace = true
```

9.6.2. HTTP 세션 덤프

[장애 진단 | HTTP 세션] 메뉴에서 WAS의 액티브한 HTTP 세션 객체를 확인할 수 있다. 이를 HTTP 세션 덤프라고 한다. HTTP 세션 덤프를 기록하려면 WAS 별로 별도의 설정을 해야 한다. 이는 WAS 별로 javax.servlet.http.HttpSession 인터페이스를 다르게 구현하기 때문이다.

현재 제니퍼가 HTTP 세션 덤프를 지원하는 WAS는 다음과 같다.

- 아파치 톰캣 4.x, 5.x, 6.x
- BEA 웹로직 8.x, 9.x, 10.x
- IBM 웹 스피어 5.x, 6.0, 6.1, 7.0

각 WAS 별로 HTTP 세션 덤프를 설정하는 방법은 다음과 같다. 설정을 완료한 후에 WAS를 재시작하여야 한다.

아파치 톰캣 4.x, 5.x에서는 다음과 같이 설정한다.

- JENNIFER_HOME/agent/jennifer_session.jar 파일을 TOMCAT_HOME/server/lib 디렉토리에 복사한다.
- 제니퍼 에이전트의 session_class 옵션을 tomcat으로 설정한다.

```
session_class = tomcat
```

- 정상적으로 설정된 경우에는 제니퍼 에이전트 로그 파일에 다음과 같은 메시지가 기록된다.

```
Catalina session-trace ok
```

아파치 톰캣 6.x에서는 다음과 같이 설정한다.

- JENNIFER_HOME/agent/jennifer_session.jar 파일을 TOMCAT_HOME/lib 디렉토리에 복사한다.
- 제니퍼 에이전트의 session_class 옵션을 tomcat으로 설정한다.

```
session_class = tomcat
```

- 정상적으로 설정된 경우에는 제니퍼 에이전트 로그 파일에 다음과 같은 메시지가 기록된다.

```
Catalina session-trace ok
```

오라클 웹로직 8.x, 9,x, 10,x에서는 다음과 같이 설정한다.

- JENNIFER_HOME/agent/jennifer_session.jar 파일을 jennifer.jar 파일 뒤에 설정한다.
- 제니퍼 에이전트의 session_class 옵션을 weblogic으로 설정한다.

```
session_class = weblogic
```

- 정상적으로 설정된 경우에는 제니퍼 에이전트 로그 파일에 다음과 같은 메시지가 기록된다.

```
Weblogic session-trace ok
```

IBM 웹스피어 5.x, 6,0에서는 다음과 같이 설정한다.

- JENNIFER_HOME/agent/jennifer_session.jar 파일을 WEBSPHERE_HOME/lib 디렉토리에 복사한다.
- 제니퍼 에이전트의 session_class 옵션을 websphere로 설정한다.

```
session_class = websphere
```

- 정상적으로 설정된 경우에는 제니퍼 에이전트 로그 파일에 다음과 같은 메시지가 기록된다.

```
WebSphere session-trace ok
```

IBM 웹스피어 6.1에서는 다음과 같이 설정한다.

Notice: IBM 웹스피어 6.1에서 HTTP 세션 덤프를 하려면 해당 라이브러리 자체를 변경해야 한다. 따라서 반드시 필요한 경우가 아니라면 이 기능을 사용하지 않도록 한다.

- WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer_2.0.0.jar 파일을 임시 디렉토리에 백업한다.
- JENNIFER_HOME/agent/jennifer_session.jar 파일에 있는 클래스들을 WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer_2.0.0.jar 파일에 포함시킨다.
- 제니퍼 에이전트의 session_class 옵션을 websphere로 설정한다.

```
session_class = websphere
```

정상적으로 설정된 경우에는 제니퍼 에이전트 로그 파일에 다음과 같은 메시지가 기록된다.

```
WebSphere session-trace ok
```

IBM 웹스피어 7.x에서는 다음과 같이 설정한다.

Notice: IBM 웹스피어 7.x에서 HTTP 세션 덤프를 하려면 해당 라이브러리 자체를 변경해야 한다. 따라서 반드시 필요한 경우가 아니라면 이 기능을 사용하지 않도록 한다.

- WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer.jar 파일을 임시 디렉토리에 백업한다.
- JENNIFER_HOME/agent/jennifer_session.jar 파일에 있는 클래스들을 WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer.jar 파일에 포함시킨다.
- 제니퍼 에이전트의 session_class 옵션을 websphere로 설정한다.

```
session_class = websphere7
```

정상적으로 설정된 경우에는 제니퍼 에이전트 로그 파일에 다음과 같은 메시지가 기록된다.

```
WebSphere session-trace ok
```

IBM 웹스피어 7.x에서는 다음과 같이 설정한다..

Notice: IBM 웹스피어 7.x에서 HTTP 세션 덤프를 하려면 해당 라이브러리 자체를 변경해야 한다. 따라서 반드시 필요한 경우가 아니라면 이 기능을 사용하지 않도록 한다.

- WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer.jar 파일을 임시 디렉토리에 백업한다.
- JENNIFER_HOME/agent/jennifer_session.jar 파일에 있는 클래스들을 WEBSPHERE_HOME/plugins/com.ibm.ws.webcontainer.jar 파일에 포함시킨다.

- 제니퍼 에이전트의 session_class 옵션을 websphere로 설정한다

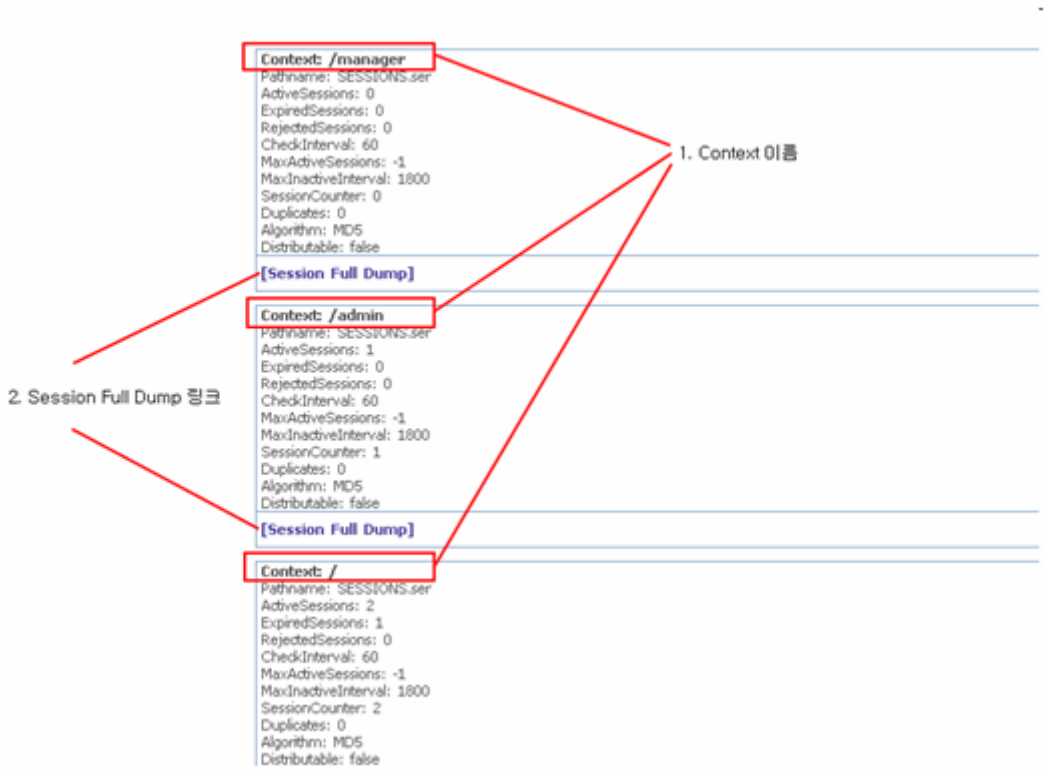
```
session_class = websphere7
```

정상적으로 설정된 경우에는 제니퍼 에이전트 로그 파일에 다음과 같은 메시지가 기록된다.

```
WebSphere session-trace ok
```

HTTP 세션 덤프를 설정했으면 **[장애 진단 | HTTP 세션]** 메뉴에서 액티브한 HTTP 세션 객체 현황을 확인할 수 있다.

그림 9-10: HTTP 세션 목록



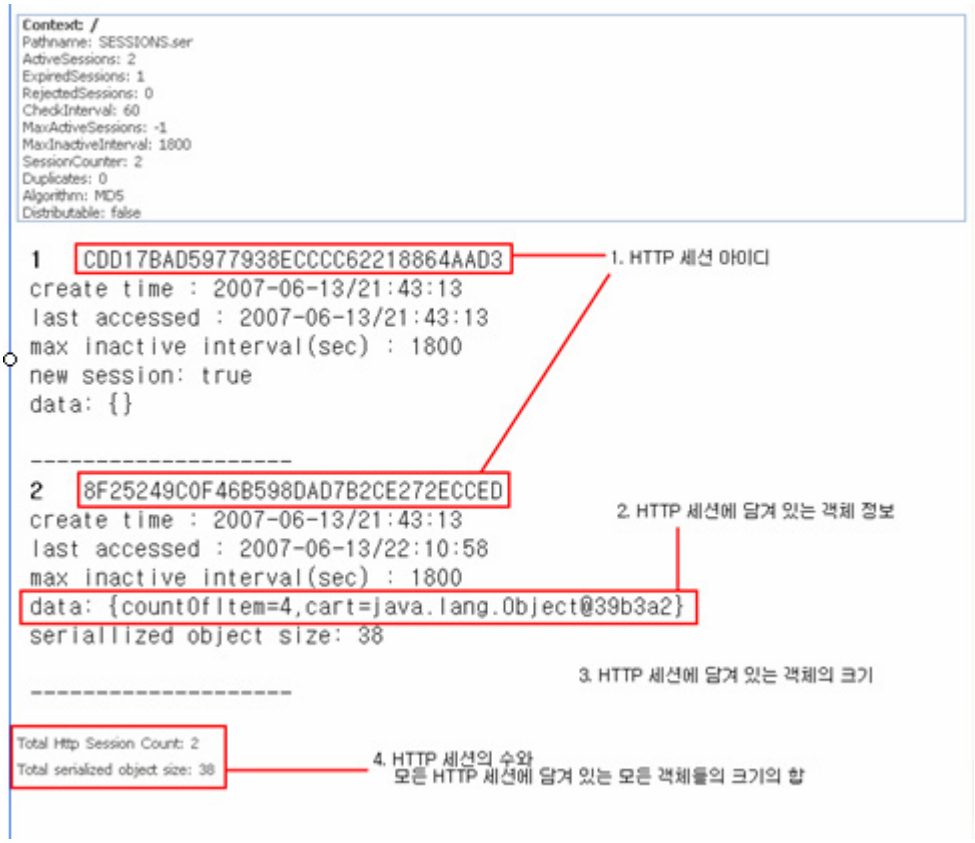
- Context 이름 - 제니퍼는 HTTP 세션에 대한 통계 정보를 웹 애플리케이션의 Context로 구분하여 제공한다. 따라서 Context 이름을 통해서 WAS에 배치되어 있는 애플리케이션 별로 HTTP 세션의 상태를 확인할 수 있다.
- Session Full Dump 링크 - 해당 링크를 클릭하면 특정 애플리케이션과 관련한 모든 HTTP 세션 객체에 대한 기본 정보를 확인할 수 있다.

다음은 아파치 톰켓을 기준으로 HTTP 세션 덤프의 주요 정보에 대한 설명이다. HTTP 세션 덤프 정보는 WAS에 따라 다르다.

표 9-5: HTTP 세션 주요 정보 (톰켓)

항목	설명
Pathname	WAS가 정지되는 순간에 액티브한 HTTP 세션 객체를 파일에 저장한 후에 WAS가 재시작할 때 해당 파일에서 액티브한 HTTP 세션 객체를 로딩하는 경우가 있다. Pathname은 이 파일의 경로를 의미한다. 상대 경로일 경우에는 자바 javax.servlet.context.tempdir 속성으로 설정한 임시 디렉토리를 기준으로 한다.
ActiveSessions	액티브한 HTTP 세션 객체 개수
ExpiredSessions	만료된 HTTP 세션 객체 개수
RejectedSessions	WAS가 생성할 수 있는 최대 액티브한 HTTP 세션 객체 개수가 초과되어서 HTTP 세션 객체를 생성하지 못한 횟수
CheckInterval	HTTP 세션 객체가 만료되었는지를 체크하는 주기로 단위는 초이다.
MaxActiveSessions	WAS가 생성할 수 있는 최대 액티브한 HTTP 세션 객체 개수로 -1은 제한이 없음을 의미한다.
MaxInactiveInterval	사용자가 요청을 하지 않으면 WAS가 HTTP 세션 객체를 소멸시키는 최대 시간 주기로 단위는 초이다. 0보다 작은 값은 HTTP 세션 객체가 소멸하지 않음을 의미한다.
SessionCounter	WAS가 생성한 HTTP 세션 객체 수
Duplicates	HTTP 세션 아이디가 중복된 개수로, 0 보다 크면 문제가 있다는 의미이다.
Algorithm	HTTP 세션 아이디를 생성하는 알고리즘으로, java.security.MessageDigest 클래스가 지원해야 한다.
Distributable	분산 환경에서 HTTP 세션 객체를 복제하는지에 대한 여부로, true인 경우에는 HTTP 세션 객체에 추가되는 모든 객체는 java.io.Serializable 인터페이스를 구현해야 한다.

그림 9-11: 개별 애플리케이션의 HTTP 세션 정보



- HTTP 세션 아이디 - HTTP 세션 아이디로 javax.servlet.http.HttpSession 객체의 getId 메소드가 반환하는 값
- HTTP 세션에 담겨 있는 객체 정보 - HTTP 세션에 담겨 있는 모든 객체의 정보를 보여준다. 특정 객체를 HTTP 세션에 저장할 때 사용했던 이름과 그 객체의 toString 메소드가 반환하는 값으로 구성된다.
- HTTP 세션에 담겨 있는 객체의 크기 - HTTP 세션에 담겨 있는 모든 객체들의 크기의 합을 보여준다. 단, transient로 선언된 필드에 할당된 객체 혹은 변수의 값은 그 크기에서 제외된다는 점을 유의한다. 단위는 바이트이다.
- HTTP 세션의 수와 모든 HTTP 세션에 담겨 있는 모든 객체들의 크기의 합 - 특정 애플리케이션에 대한 HTTP 세션의 수와 그 HTTP 세션에 담겨 있는 모든 객체들의 크기의 합을 보여준다. 단위는 바이트이다.

create time, last accessed, max inactive interval 등은 HttpSession 객체의 getCreationTime, getLastAccessedTime, getMaxInactiveInterval 등의 메소드가 반환하는 값을 의미한다.

9.7. 외부 트랜잭션 모니터링

외부 트랜잭션은 자바 애플리케이션이 다른 애플리케이션과 연동하는 것을 의미한다. 외부 트랜잭션 처리 현황과 응답 시간은 전체 자바 애플리케이션 성능에 중요한 영향을 미친다.

따라서 제니퍼는 외부 트랜잭션을 모니터링하여 외부 트랜잭션 처리 현황 통계 데이터와 외부 트랜잭션 처리 내용을 포함한 X-View 프로파일 데이터 등을 수집한다.

일반적으로 외부 트랜잭션은 데이터베이스, TP(Transaction Processing) 모니터, LDAP(Lightweight Directory Access Protocol) 서버 등 모든 외부 애플리케이션과의 연계를 의미하지만, 데이터베이스는 그 중요성과 특수성으로 인하여 별도의 JDBC 모니터링으로 취급한다. 즉, 제니퍼에서 외부 트랜잭션 모니터링은 데이터베이스를 제외한 모든 외부 애플리케이션과의 연동을 모니터링하는 것이다.

9.7.1. 외부 트랜잭션 시작점 설정

외부 트랜잭션 모니터링은 다른 애플리케이션과의 연동을 담당하는 자바 애플리케이션 내부 모듈(특정 클래스의 특정 메소드)을 모니터링하는 것이다. 예를 들어, 메일 서버와 연동하는 클래스가 example.MailManager이고 메일을 보내는 메소드는 sendMail이라면, 외부 트랜잭션 모니터링은 example.MailManager 클래스의 sendMail 메소드의 호출 건수와 응답 시간 등의 데이터를 수집하는 것을 의미한다.

이를 위해서는 어떤 클래스의 어떤 메소드가 외부 트랜잭션 시작점인지를 설정해야 한다. 이를 외부 트랜잭션 시작점 설정이라고 한다.

제니퍼는 아래의 외부 애플리케이션과의 연동은 별도의 설정없이 외부 트랜잭션 시작점으로 자동 인식한다.

- BEA JOLT 2.0, 2.1, 2.5, 3.0
- BEA WTC(Weblogic Tuxedo Connector)
- 티맥스소프트 WebT 3.x, 5.x

IBM CICS(Customer Information Control System)에서 CTG 5.x를 사용하는 경우, JENNIFER_HOME/agent/3rdparty/ctg51_jennifer40.jar파일을 CTG(CICS Transaction Gateway)를 위한 JAR 파일보다 앞쪽으로 클래스 패스에 추가하면, 외부 트랜잭션 시작점으로 자동 인식된다. CTG 5.x이외의 버전에서는 외부 트랜잭션 시작점으로 자동인식이 되지 않는다.

Notice: CTG 5.x 버전이 아닌 경우에는 tx_client 옵션으로 외부 트랜잭션 시작점을 설정해야 한다.

위에 기술된 것들 이외의 외부 트랜잭션의 경우, 제니퍼 에이전트의 tx_client로 시작하는 옵션을 통해서 외부 트랜잭션 시작점을 설정한다. 수정한 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

다음은 메일 서버와 연동하는 클래스가 example.MailManager일 경우의 외부 트랜잭션 시작점 설정의 예이다.

```
tx_client_class = example.MailManager
```

추가로 LDAP서버와 연동하는 클래스 example.LdapManager를 외부 트랜잭션 시작점으로 설정하는 경우에는 세미콜론[;]을 구분자로 해서 tx_client_class옵션에 설정한다. tx_client_class 옵션뿐만 아니라 외부 트랜잭션 시작점과 관련한 모든 옵션에 2개 이상을 설정하려면 세미콜론[;]을 구분자로 사용한다.

```
tx_client_class = example.MailManager;example.LdapManager
```

외부 트랜잭션 시작점은 특정 클래스의 특정 메소드를 의미한다. 즉, 앞에서와 같이 클래스만 설정하면 example.MailManager 클래스의 모든 메소드가 외부 트랜잭션 시작점으로 인식된다.

따라서 특정 메소드만을 외부 트랜잭션 시작점으로 설정하려면 제니퍼 에이전트의 tx_client_target_method 옵션으로 해당 메소드를 설정한다.

```
tx_client_target_method = sendMail
```

그런데 외부 트랜잭션 시작점으로 설정한 클래스들 중에서 이름이 동일한 메소드가 있고, 이중 특정 클래스의 특정 메소드만을 외부 트랜잭션 시작점으로 설정하려면 다음과 같이 구체적으로 설정한다.

```
tx_client_target_method = example.MailManager.sendMail
```

반대로 특정 메소드만을 외부 트랜잭션 시작점에서 제외하려면 제니퍼 에이전트의 tx_client_ignore_method 옵션으로 설정한다.

```
tx_client_ignore_method = example.MailManager.someMethod
```

메소드의 접근자를 통해서도 외부 트랜잭션 시작점을 설정할 수 있다. 예를 들어 public 접근자와 접근자가 없는 메소드를 외부 트랜잭션 시작점으로 등록하려면 다음과 같이 설정한다.

```
tx_client_access_method = public;none
```

tx_client_access_method 옵션에 설정이 가능한 값은 다음과 같다.

- public - public 접근자
- protected - protected 접근자
- private - private 접근자
- none - 접근자가 없는 경우

그리고 특정 클래스를 상속한 모든 클래스가 외부 트랜잭션 시작점인 경우에는 제니퍼 에이전트의 tx_client_super 옵션을 사용한다. 예를 들어, example.ejb.BaseSessionBean을 상속한 모든 클래스를 외부 트랜잭션으로 설정하려면 다음과 같이 설정한다.

```
tx_client_super = example.ejb.BaseSessionBean
```

그러나 이 경우에 직접적으로 상속받은 클래스만이 외부 트랜잭션 시작점이 된다. A 클래스가 example.ejb.BaseSessionBean 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스는 외부 트랜잭션 시작점으로 인식하지만 B 클래스는 외부 트랜잭션 시작점으로 인식하지 않는다.

그리고 특정 인터페이스를 구현한 모든 클래스가 외부 트랜잭션 시적점인 경우에는 제니퍼 에이전트의 tx_client_interface 옵션을 사용한다. 예를 들어, example.eai.ITransaction-Manager 인터페이스를 구현한 모든 클래스를 외부 트랜잭션으로 설정하려면 다음과 같이 설정한다.

```
tx_client_interface = example.eai.ITransactionManager
```

그러나 이 경우에 직접적으로 구현한 클래스만이 외부 트랜잭션 시작점이 된다. A 클래스가 example.eai.ITransactionManager 인터페이스를 구현하고 B 클래스가 A 클래스를 상속했다면, A 클래스는 외부 트랜잭션 시작점으로 인식하지만 B 클래스는 외부 트랜잭션 시작점으로 인식하지 않는다.

그리고 클래스의 이름을 이용해서도 외부 트랜잭션 시작점을 설정할 수 있다. 이 경우에는 제니퍼 에이전트의 tx_client_prefix, tx_client_postfix, tx_client_ignore_prefix 옵션을 사용한다. 이름이 example.eai로 시작하는 모든 클래스를 외부 트랜잭션 시작점으로 하기 위해서는 다음과 같이 설정한다.

```
tx_client_prefix = example.eai
```

또한 이름이 TpMonitor로 끝나는 모든 클래스를 외부 트랜잭션 시작점으로 하기 위해서는 다음과 같이 설정한다.

```
tx_client_postfix = TpMonitor
```

그리고 특정 이름으로 시작하는 클래스를 외부 트랜잭션 시작점에서 제외하려면 제니퍼 에이전트의 tx_client_ignore_prefix 옵션을 사용한다. 이 옵션은 다른 모든 옵션에 우선한다.

```
tx_client_ignore_prefix =
```

제니퍼 에이전트의 tx_client로 시작하는 옵션을 통해서 동일한 내용을 다양한 방법으로 설정할 수 있다. 외부 트랜잭션 시작점을 설정하는데 있어서 tx_client_target_method 옵션을 통해서 실제로 외부 트랜잭션 시작점인 메소드만을 설정하도록 한다.

예를 들어, pkg.ClassA와 pkg.ClassB 클래스가 있다. 여기서 ClassA 클래스의 run 메소드와 ClassB 클래스의 process 메소드가 외부 트랜잭션 시작점이라고 가정한다. 그런데 ClassB 클래스에 run 메소드가 존재하고 이 메소드는 외부 트랜잭션과는 상관이 없다면 다음과 같이 설정한다.

```
tx_client_class = pkg.ClassA;pkg.ClassB
tx_client_target_method = run;process
tx_client_ignore_method = pkg.ClassB.run
```

또는 다음과 같이 설정할 수도 있다.

```
tx_client_class = pkg.ClassA;pkg.ClassB
tx_client_target_method = pkg.ClassA.run;pkg.ClassB.process
```

9.7.2. 외부 트랜잭션 네이밍

외부 트랜잭션 이름은 TXNAMES 테이블에 저장된다. 기본적으로 외부 트랜잭션 이름은 클래스 이름과 메소드 이름으로 구성된다. 그런데 BEA 텍시도를 호출하는 경우에는 텍시도 서비스 이름이, IBM 웹스피어 MQ를 호출하는 경우에는 큐 이름이 외부 트랜잭션 이름에 포함되어야 좀더 정확한 모니터링이 가능하다.

이를 위해서 외부 트랜잭션 이름을 다양한 방식으로 설정하는 것을 외부 트랜잭션 네이밍이라고 한다. 수정한 외부 트랜잭션 네이밍과 관련한 옵션을 반영하려면 제니퍼 에이전트를 설치한 자바 애플리케이션을 재시작해야 한다.

기본적으로 외부 트랜잭션 이름은 클래스 이름과 메소드 이름으로 구성된다. 제니퍼 에이전트의 tx_client_ntype 옵션을 통해서 이를 다양하게 설정할 수 있다. 기본 값은 FULL이고, SIMPLE, CLASS, METHOD 등으로 설정할 수 있다.

```
tx_client_ntype = FULL
```

예를 들어, pkg.ClassB 클래스의 process 메소드를 외부 트랜잭션 시작점이라고 할 때 tx_client_ntype 옵션 설정에 따라서 외부 트랜잭션 이름은 다음과 같이 결정된다.

- FULL - public void pkg.ClassB.process()
- SIMPLE - ClassB.process
- CLASS - ClassB
- METHOD - process

그리고 외부 트랜잭션 시작점인 메소드의 파라미터 값을 외부 트랜잭션 이름으로 사용할 수도 있다. 이를 위해서는 제니퍼 에이전트의 tx_client_using_param 옵션을 true로 설정한다.

```
tx_client_using_param = true
```

tx_client_using_param 옵션을 true로 설정하면, 외부 트랜잭션 시작점인 메소드의 여러 개의 파라미터 중에서 java.lang.String 타입의 파라미터 중에서 첫번째 파라미터가 외부 트랜잭션 이름이 된다. 해당 사항이 없는 경우에는 tx_client_ntype 옵션에 따라서 외부 트랜잭션 이름이 결정된다.

그리고 외부 트랜잭션 이름으로 외부 트랜잭션 시작 메소드 내부에서 호출되는 특정 클래스의 특정 메소드의 파라미터나 반환 값을 사용할 수도 있다.

외부 트랜잭션 시작 메소드 내부에서 호출되는 특정 클래스의 특정 메소드의 파라미터를 외부 트랜잭션 이름으로 사용하려면 다음과 같이 설정한다.

```
lwst_txclient_method_using_param = pkg.ClassC.f1(String)
```

이 경우에는 여러 개의 파라미터 중에서 java.lang.String 타입의 파라미터 중에서 첫번째 파라미터가 외부 트랜잭션 이름이 된다.

그리고 외부 트랜잭션 시작 메소드 내부에서 호출되는 특정 클래스의 특정 메소드의 반환 값을 외부 트랜잭션 이름으로 사용하려면 다음과 같이 설정한다.

```
lwst_txclient_method_using_return = pkg.ClassC.f2(String)
```

단, 이 경우에는 반환되는 객체의 유형이 java.lang.String이어야 한다.

lwst_txclient_method_using_param과 lwst_txclient_method_using_return 옵션에도 2개 이상은 세미콜론[;]을 구분자로 해서 설정한다.

여러 옵션이 설정된 경우에는 다음 옵션 순서대로 외부 트랜잭션 이름이 결정된다.

- lwst_txclient_method_using_param 혹은 lwst_txclient_method_using_return
- tx_client_using_param

- tx_client_ntype

lwst_txclient_method_using_param과 lwst_txclient_method_using_return 옵션의 경우에는 실제 트랜잭션이 처리되는 과정에서 마지막으로 수행되는 메소드의 파리미터 혹은 반환 값이 외부 트랜잭션 이름이 된다.

9.7.3. 외부 트랜잭션 모니터링과 예외 발생

외부 트랜잭션의 응답 속도가 임계치를 초과하면 WARNING_TX_BAD_RESPONSE 예외가 발생한다. 임계치는 제니퍼 에이전트의 tx_bad_responsetime 옵션으로 설정한다. 기본 값은 10000이고 단위는 밀리 세컨드이다.

```
tx_bad_responsetime = 10000
```

9.8. DB 연결 모니터링

데이터베이스 연동은 엔터프라이즈 애플리케이션 성능에 많은 영향을 미친다. 따라서 제니퍼는 이에 대한 성능 분석 및 조치를 위한 DB 모니터링을 제공한다. DB 모니터링으로 수집하는 데이터는 다음과 같다.

- 애플리케이션이 수행하는 SQL 및 파라미터
- SQL 응답 시간 및 Fetch 건수
- DB 커넥션 개수
- DB 연결 객체 미반환 등의 예외

9.8.1. DB 모니터링을 위한 기본 설정

DB 사용을 모니터링을 하려면 build_jdbc 패치 옵션을 true로 설정해서 LWST 빌드를 해야 한다. 기본 값은 true이다. 자세한 사항은 [LWST 빌드와 설치(61 페이지)]를 참조한다.

그리고 제니퍼 에이전트의 enable_jdbc_sql_trace 옵션을 true로 설정해야 한다. 기본 값은 true이다. 이 옵션을 false로 설정하면 JDBC 모니터링이 이루어지지 않는다.

```
enable_jdbc_sql_trace = true
```

또한 세부 항목별로 JDBC 모니터링을 비활성화할 수 있다. 기본 값은 모두 true이다.

```
enable_jdbc_callablestatement_trace = true
enable_jdbc_preparedstatement_trace = true
enable_jdbc_statement_trace = true
enable_jdbc_resultset_trace = true
enable_jdbc_databasemetadata_trace = true
```

JDBC 모니터링 세부 항목을 비활성화하면 관련 데이터가 수집되지 않는다. 예를 들어, enable_jdbc_resultset_trace 옵션을 false로 하면 Fetch 건수가 수집되지 않고, enable_jdbc_preparedstatement_trace 옵션을 false로 하면 java.sql.PreparedStatement 객체로 수행한 SQL의 응답 시간이 수집되지 않는다.

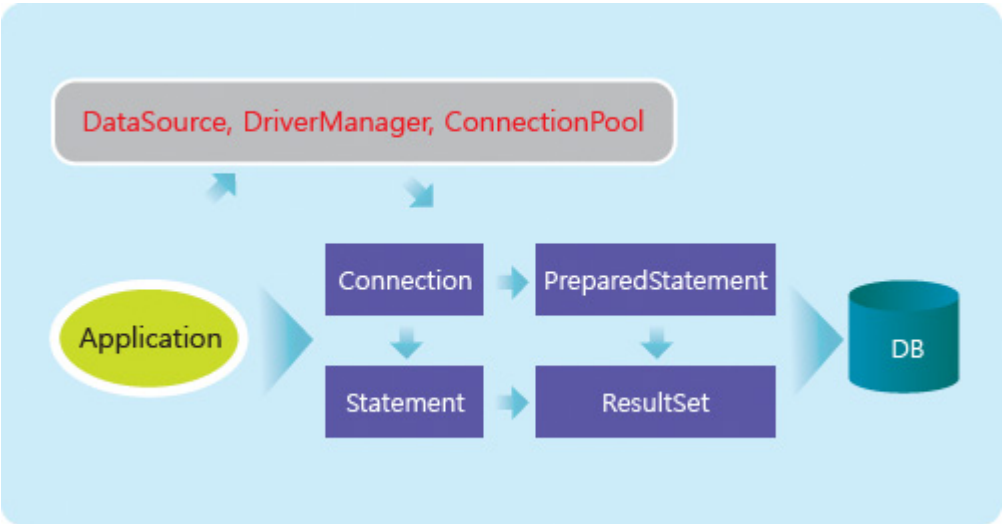
비표준 방식으로 데이터베이스와 연동하는 자바 애플리케이션에 제니퍼를 설치하면 충돌이 발생할 수 있다. 이런 경우에는 세부 항목별로 JDBC 모니터링을 비활성화하여 원인을 파악하도록 한다.

9.8.2. 자바 DB 커넥션 추적 설정

자바 애플리케이션이 JDBC API를 통해서 데이터베이스와 연동하는 방법은 다음과 같다.

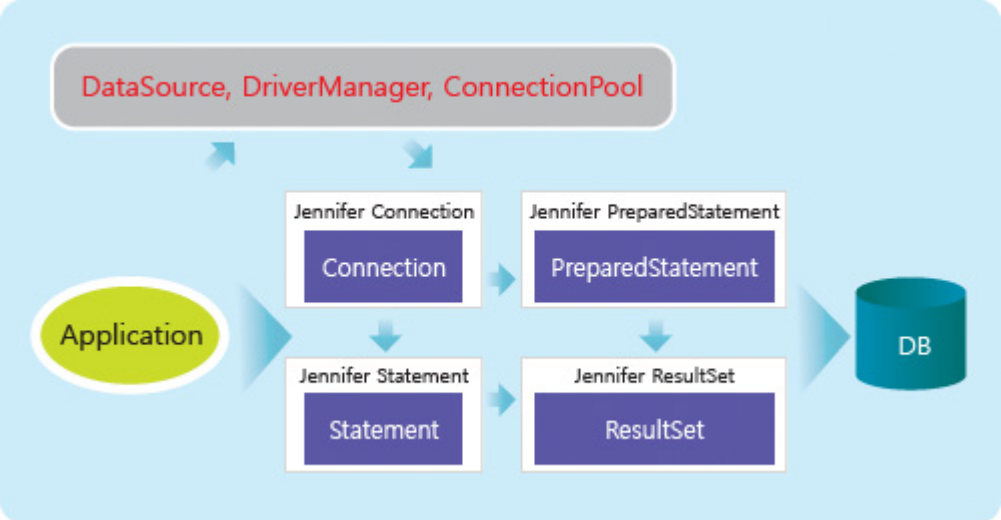
- java.sql.DriverManager 클래스 혹은 javax.sql.DataSource 객체를 통해서 java.sql.Connection 객체를 획득한다.
- java.sql.Connection 객체를 중심으로 java.sql.Statement, java.sql.PreparedStatement, java.sql.ResultSet 등의 객체를 획득하여 데이터베이스 작업을 수행한다.

그림 9-12: JDBC API 구조



그리고 JDBC 모니터링을 위해서 사용하는 방법은 Class Wrapping이다.

그림 9-13: JDBC 모니터링을 위한 구조



이를 위해서 자바 애플리케이션이 java.sql.Connection 객체를 획득하는 시점에 해당 객체를 JenniferConnection 객체로 Wrapping해야 한다. java.sql.Statement, java.sql.ResultSet 등의 객체는 JenniferConnection 객체에 의해서 자동으로 Wrapping된다.

그래서 java.sql.Connection 객체를 JenniferConnection 객체로 Wrapping하는 것을 JDBC 커넥션 추적 설정이라고 한다. 일반적으로 애플리케이션이 java.sql.Connection 객체를 획득하는데에는 3가지 유형이 있고 제니퍼는 이 유형들에 맞는 JDBC 커넥션 추적 설정 방법을 제공한다.

- 유형 1 - JNDI와 javax.sql.DataSource 클래스를 사용
- 유형 2 - java.sql.DriverManager 클래스를 사용
- 유형 3 - 임의의 클래스를 사용

그런데 java.sql.Connection 객체가 만들어지는 시점의 스택트레이스를 제니퍼 에이전트 로그 파일에 기록할 수 있다. 이를 위해서는 제니퍼 에이전트의 debug_connection_open 옵션을 true로 설정한다. 기본 값은 false이다.

```
debug_connection_open = true
```

이를 통해서 JDBC 커넥션 추적 설정에 필요한 정보를 확보할 수 있다. 그러나 java.sql.Connection 객체를 생성할 때 마다 java.lang.Throwable 객체의 printStackTrace 메소드로 스택트레이스를 기록함으로 부하가 발생한다. 따라서 디버깅을 위한 목적으로만 사용하고 일반적인 경우에는 false로 설정한다.

9.8.2.1. 유형 1 - JNDI & DataSource

자바 애플리케이션에서 JNDI(Java Naming and Directory Interface)를 이용해서 javax.sql.DataSource 객체를 획득하고, 이 객체로부터 java.sql.Connection 객체를 획득하는 경우를 유형 1이라고 한다.

```
javax.naming.Context jndiContext = new javax.naming.InitialContext();
javax.sql.DataSource ds = (javax.sql.DataSource)
                    jndiContext.lookup( “java:comp/env/jdbc/AppDS” );
java.sql.Connection con = ds.getConnection();
```

유형 1은 추가적인 설정없이 JDBC 모니터링이 가능하다. 단, 제니퍼 에이전트의 enable_jdbc_datasource_trace 옵션을 true로 설정해야 한다. 기본 값은 true이다.

```
enable_jdbc_datasource_trace = true
```

그런데 javax.naming.InitialContext 객체로부터 javax.sql.DataSource 객체를 직접적으로 획득하지 않고, 다른 javax.naming.Context 객체를 획득한 후에 그 객체에서 javax.sql.DataSource 객체를 획득하는 경우에는 별도의 설정이 필요하다.

```
javax.naming.Context jndiContext = new javax.naming.InitialContext();
javax.naming.Context jdbcContext = (javax.naming.Context)
                          jndiContext.lookup( “java:comp/env/jdbc” );
javax.sql.DataSource ds = (javax.sql.DataSource)
                jdbcContext.lookup( “AppDS” );
java.sql.Connection con = ds.getConnection();
```

이런 경우에는 제니퍼 에이전트의 enable_wrap_context_jdbc_trace 옵션을 true로 설정해야 한다.

```
enable_wrap_context_jdbc_trace = true
```

단, enable_wrap_context_jdbc_trace 옵션을 true로 설정하면 EJB를 사용하는 경우에 java.lang.ClassCastException이 발생할 수 있다. 이 경우에는 유형 3으로 JDBC 커넥션을 추적해야 한다.

그리고 유형 1로 JDBC 커넥션을 추적하는 경우에 트랜잭션을 처리하는 자바 쓰레드가 다른 자바 쓰레드를 만들어서 JDBC 처리를 하면 해당 자바 쓰레드가 작업한 JDBC 내용이 모니터링되지 않는다. 이 경우에는 제니퍼 에이전트의 enable_non_servlet_thread_jdbc_trace 옵션을 true로 설정한다.

```
enable_non_servlet_thread_jdbc_trace = true
```

그리고 유형 1로 JDBC 커넥션을 추적하는 경우에 자바 애플리케이션이 데이터소스를 백그라운드 자바 쓰레드에서 초기화하면 JDBC 모니터링이 수행되지 않는다. 이 경우에도 제니퍼 에이전트의 enable_non_servlet_thread_jdbc_trace 옵션을 true로 설정한다. EJB 엔티티 빈을 사용하는 경우에 장애가 발생할 수도 있다. 이 경우에는 유형 1이 아닌 유형 3으로 JDBC 커넥션을 추적해야 한다.

9.8.2.2. 유형 2 - DriverManager

자바 애플리케이션에서 java.sql.DriverManager 클래스를 통해서 java.sql.Connection 객체를 획득하는 경우를 유형 2라고 한다. 이 경우에는 JDBC 커넥션을 풀링하지 않기 때문에 JDBC 커넥션 개수를 체크하지 못한다.

Notice: Warning: 한 트랜잭션이 처리하는 SQL 개수가 많거나 커넥션에 대한 획득과 반환이 매우 빈번한 경우에 제니퍼를 설치하면 자원 사용량이 높아질 수 있다.

유형 2는 제니퍼 에이전트의 user_defined_jdbc_connectionpool_prefixes 옵션을 통해서 JDBC 커넥션 추적 설정을 한다. 사용하는 데이터베이스와 JDBC 드라이버에 따라서 설정이 달라진다.

다음 코드는 오라클 데이터베이스를 사용하는 경우이다.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
java.sql.Connection con = java.sql.DriverManager.getConnection
( "jdbc:oracle:thin:@127.0.0.1:1521:ORACL","scott", "tiget");
```

제니퍼 에이전트의 user_defined_jdbc_connectionpool_prefixes 옵션에 java.sql.DriverManager 클래스의 getConnection 메소드의 첫번째 파라미터 값의 일부를 시작 부분을 포함해서 설정한다.

```
user_defined_jdbc_connectionpool_prefixes = jdbc
```

그런데 여러 개의 데이터베이스를 사용하고 이 중 특정 데이터베이스와의 연동만을 모니터링하려면 user_defined_jdbc_connectionpool_prefixes 옵션을 좀더 구체적으로 설정한다.

```
user_defined_jdbc_connectionpool_prefixes = jdbc:oracle:thin:@local
```

이 옵션을 사용하는 경우에 JDBC 커넥션 객체의 미반환 여부를 체크하려면 제니퍼 에이전트의 user_defined_jdbc_ignore_close 옵션을 추가한 후, false로 설정해야 한다.

```
user_defined_jdbc_ignore_close = false
```

9.8.2.3. 유형 3 - 임의의 클래스

자바 애플리케이션이 커넥션 풀의 역할을 담당하는 임의의 클래스를 통해서 java.sql.Connection 객체를 획득하는 경우를 유형 3이라고 한다. 아파치 DBCP와 같은 커넥션 풀 라이브러리를 사용하거나 레드햇 Hibernate 혹은 아파치 iBATIS 등의 프레임워크를 사용하는 경우가 이에 해당한다. 엄밀하게는 JDBC 커넥션 풀링 여부와는 상관없이 특정 클래스의 메소드를 통해서 java.sql.Connection 객체를 획득하고 반환하는 경우에 유형 3으로 설정한다. 따라서 유형 3으로 설정하려면 애플리케이션 소스 코드를 이해하고 있어야 한다.

예를 들어, 자바 애플리케이션에서 example.ConnectionPool 클래스의 getConnection 메소드를 통해서 java.sql.Connection 객체를 획득한다면 다음과 같이 설정한다.

```
jdbc_connection_get = example.ConnectionPool.getConnection()
```

만약 해당 메소드에 파라미터가 있다면 파라미터 유형까지 기술해야 한다.

```
jdbc_connection_get = example.ConnectionPool.getConnection(String)
```

그리고 example.ConnectionPool 클래스의 releaseConnection 메소드로 java.sql.Connection 객체를 커넥션 풀에 반환하는 경우에는 다음과 같이 설정한다.

```
jdbc_connection_close =
example.ConnectionPool.releaseConnection(Connection)
```

이 경우에는 메소드의 파라미터 중에 java.sql.Connection 유형이 꼭 존재해야 한다. 그런데 java.sql.Connnection 객체의 close 메소드로 반환을 한다면 제니퍼 에이전트의 jdbc_connection_close 옵션을 설정할 필요가 없다. 그리고 java.sql.Connection 객체를 반환하는 방법을 알수 없는 경우에는 제니퍼 에이전트의 jdbc_connection_justget 옵션으로 설정한다.

```
jdbc_connection_juestget =
example.ConnectionPool.getConnection(String)
```

제니퍼 에이전트의 jdbc_connection_juestget 옵션을 사용하면 JDBC 커넥션 개수와 JDBC 커넥션 객체의 미반환 여부를 체크하지 못한다.

유형 3으로 설정할 때 다음을 주의한다.

- 패키지 이름을 포함한 클래스 이름을 사용해야 한다.
- 단, 메소드의 파라미터는 패키지 이름을 제외한 클래스 이름만을 사용할 수 있다.
- jdbc_connection_get 혹은 jdbc_connection_justget 옵션으로 설정한 메소드의 반환 값은 java.sql.Connection 유형이어야 한다.

- jdbc_connection_close 옵션으로 설정한 메소드의 파라미터 중에 반드시 하나는 java.sql.Connection 유형이어야 한다. 단, 파라미터 위치는 상관이 없다.
- 모든 옵션에 세미콜론[;]을 구분자로 두개 이상의 값을 설정할 수 있다.

레드햇 Hibernate를 사용하는 경우에는 다음과 같이 설정한다.

```
jdbc_connection_justget =
org.hibernate.jdbc.ConnectionManager.getConnection()
```

아파치 iBATIS를 사용하는 경우에는 다음과 같이 설정한다.

```
jdbc_connection_get =
com.ibatis.sqlmap.engine.transaction.jdbc.JdbcTransaction.getConnectio
n()
```

아파치 DBCP를 사용하는 경우에는 다음과 같이 설정한다.

```
jdbc_connection_get =
org.apache.commons.dbcp.PoolingDataSource.getConnection()
```

9.8.3. DB 커넥션 개수 모니터링

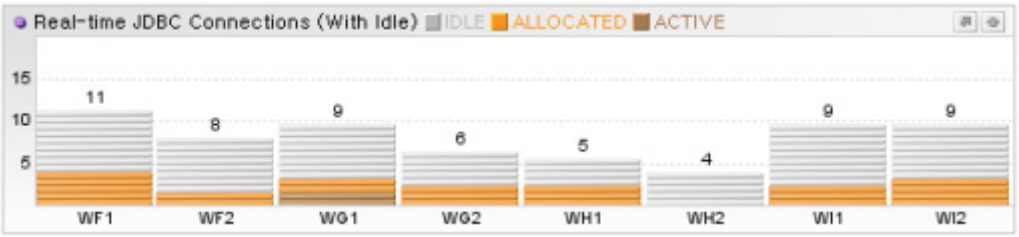
유형 1 혹은 유형 3으로 DB 커넥션을 추적하는 경우에는 DB 커넥션 개수를 모니터링할 수 있다. DB 커넥션 유형은 다음과 같다.

표 9-6: DB 커넥션 유형

유형	설명
대기 중인 JDBC 커넥션 개수(IDLE)	DB 커넥션 풀에서 대기 중인 DB 커넥션 개수
할당된 DB 커넥션 개수(ALLOCATED)	자바 쓰레드가 DB 커넥션 풀로부터 할당받은DB 커넥션 중에서 SQL 쿼리를 수행하고 있지 않은 DB 커넥션 개수
사용 중인 DB 커넥션 개수(ACTIVE)	자바 쓰레드가 DB커넥션 풀로부터 할당받은 DB 커넥션 중에서 SQL 쿼리를 수행하고 있는 DB 커넥션 개수

DB 커넥션 개수는 제니퍼 에이전트가 수집하는 일반 성능 데이터로 해당 시점의 값을 이 퀄라이저 차트와 런타임 라인 차트를 통해서 확인할 수 있다.

그림 9-14: 실시간 JDBC 커넥션 개수



또한 DB 커넥션 개수에 대한 5분 평균 값은 PERF_X_01~31 테이블의 JDBC_IDLE, JDBC_ALLOC, JDBC_ACTIVE 칼럼에 저장된다.

• 자바 애플리케이션에서 DB 커넥션 개수가 모니터링되지 않는 경우

제니퍼 내부에는 각 데이터베이스와 JDBC 드라이버별로 키가 되는 커넥션 목록이 등록되어 있다. 만약 제니퍼에 등록되지 않은 JDBC 드라이버가 사용되면 JDBC 커넥션 개수가 모니터링 되지 않는다.

이 경우 해당 JDBC 드라이버의 키 커넥션을 찾아 제니퍼 에이전트의 connection_trace_class 옵션에 설정해주어야 한다. 2개 이상은 세미콜론[;]을 구분자로 구분한다.

```
connection_trace_class = com.ibm.db2.jcc.b.bb;com.ibm.db2.jcc.b.o
```

특히 DB2 JDBC 드라이버는 버전에 따라 키 커넥션 클래스가 변경되기 때문에 이 옵션으로 설정해주어야 한다. 키 커넥션은 데이터베이스 연결을 위한 TCP 포트 번호에 대한 스택트레이스를 추적하는 방법으로 찾을 수 있다.

9.8.4. SQL 데이터 수집 방법

JDBC 모니터링은 다른 자원을 모니터링하는 것에 비해 상당히 많은 데이터를 수집해야 한다. 자바 애플리케이션이 수행하는 모든 SQL과 파라미터를 수집해야지만 문제가 있는 SQL에 대해서 튜닝 등의 작업을 할 수 있기 때문이다.

그래서 데이터 전송량을 줄이기 위해서 제니퍼 에이전트가 제니퍼 서버에 보내는 데이터에서는 SQL이 아닌 해당 SQL에 대한 해시 값을 사용한다. 그리고 별도의 방법을 통해서 해시 값에 대한 SQL을 수집해서 제니퍼 서버의 SQLS 테이블에 저장한다.

그리고 SQLS 테이블에 저장되는 SQL의 개수를 줄이기 위해서, SQL에 있는 상수를 파라미터로 처리하여 저장한다. 예를 들어, 자바 애플리케이션이 처리한 다음 SQL은 별도의 SQL이다.

```
SELECT * FROM TABLE WHERE ID = ? AND AGE = 32 AND NAME =  'KIM'
SELECT * FROM TABLE WHERE ID = ? AND AGE = 27 AND NAME =  'LEE'
```

그러나 앞의 코드는 상수를 제외한 기본적인 구조는 동일하다. 이를 그대로 SQLS 테이블에 저장하면 데이터의 양이 많아진다. 예를 들어, 다음 코드가 수행된다면 10,000개의 SQL이 SQLS 테이블에 저장되어야 한다.

```
Statement stmt = ...;
ResultSet rs = null;
for (int i = 0; i < 10000; i++) {
  rs = st.executeQuery("SELECT * FROM USERS WHERE ID = " + i);
  ...
}
```

이를 해결하기 위해서 상수 중에서 문자는 [$]로, 숫자는 [#]으로 변경하여 SQLS 테이블에 저장한다.

따라서 SQLS 테이블에는 다음 SQL 만이 저장된다.

```
SELECT * FROM TABLE WHERE ID = ? AND AGE = # AND NAME =  '$'
```

그리고 X-View 프로파일 데이터 등에서는 상수 파라미터는 파라미터 1로 나타나고, 바인딩 파라미터는 파라미터 2로 표시된다. 바인딩 파라미터는 java.sql.PreparedStatement 객체로 수행하는 SQL에서 [?]로 표시되는 파라미터를 의미한다.

그런데 제니퍼 서버가 수집한 바인딩 파라미터가 정상적으로 보이지 않는 경우가 있다. 이 경우에는 제니퍼 에이전트의 sqlparam_encoding 옵션을 추가하여 java.sql.PreparedStatement 객체의 SQL 바인딩 파라미터에 대한 인코딩을 설정한다.

```
sqlparam_encoding = 8859_1
```

예를 들어, 데이터베이스 인코딩이 애플리케이션이 사용하는 인코딩과 달라서, 다른 프로파일 데이터는 정상적으로 보이는데 SQL 바인딩 파라미터만 정상적으로 보이지 않는 경우가 있다. 이런 경우는 애플리케이션이 SQL 파라미터를 설정하기 전에 명시적으로 인코딩을 변경한 것이며, 이 때 이 옵션을 사용하여 문제를 해결하도록 한다.

9.8.5. 오라클 Dependency

오라클 데이터베이스를 사용하는 자바 애플리케이션을 모니터링할 때, 표준이 아닌 오라클 의존적인 코드를 사용하면 java.lang.ClassCastException이 발생할 수 있다. 예를 들어, CLOB, BLOB 등을 다음과 같이 처리한 경우가 이에 해당한다.

```
Statement stmt = ...;
ResultSet rs =
stmt.executeQuery("SELECT TEXT FROM TEST_CLOB WHERE ID =1 FOR UPDATE");
if (rs. next()) {
    oracle.sql.CLOB cl =
           ((oracle.jdbc.OracleResultSet) rs).getCLOB("TEXT");
    ....
}
```

이 경우에는 제니퍼 에이전트의 enable_jdbc_oracle_dependency_used 옵션을 true로 설정한다.

```
enable_jdbc_oracle_dependency_used = true
```

오라클 데이터베이스가 아닌 다른 데이터베이스를 사용하는 경우에 이 옵션을 true로 설정해도 문제가 되지 않는다.

그런데 enable_jdbc_oracle_dependency_used 옵션을 true로 설정한 후에 JDBC 모니터링이 되지 않고 X-View 프로파일 데이터에 다음과 같은 메세지가 나타날 수 있다.

```
enable_jdbc_oracle_dependency_used = true, but Oracle Driver not found
```

이 경우는 오라클 JDBC 드라이버가 자바 애플리케이션 클래스 패스에 없음에도 해당 옵션을 true로 설정하였거나, jennifer.jar 파일에서 오라클 JDBC 드라이버 파일을 참조할 수 없음을 의미한다.

enable_jdbc_oracle_dependency_used 옵션을 true로 설정한 상태에서 내부 객체가 오라클 JDBC 객체가 아닌 경우 내부 객체를 unwrap할 수 있다.

WAS가 JDBC 커넥션 풀을 구현할 때 Wrapper 클래스를 사용한다. 따라서 unwrap이란 이 Wrapper 객체로 부터 원래의 JDBC 객체를 꺼내는 것을 말한다.

unwrap_method로 끝나는 제니퍼 에이전트의 옵션으로 unwrap을 위한 메소드명을 설정한다.

```
jdbc_connection_unwrap_method = getVendorObj
jdbc_statement_unwrap_method = getVendorObj
jdbc_preparedstatement_unwrap_method = getVendorObj
jdbc_callablestatement_unwrap_method = getVendorObj
jdbc_resultset_unwrap_method = getVendorObj
```

각 JDBC 객체 별로 unwrap 메소드를 설정할 수 있다. 만약 모든 JDBC 객체를 위한 unwrap 메소드 이름이 동일하다면 제니퍼 에이전트의 jdbc_unwrap_method 옵션만으로 설정할 수 있다.

```
jdbc_unwrap_method = getVendorObj
```

unwrap은 특별한 경우에만 사용한다. 웹로직 JDBC 커넥션 풀 Wrapper 클래스들은 오라클 JDBC 드라이버 API를 구현하고 있다. 따라서 웹로직을 사용하는 경우에는 unwrap이 필요하지 않다.

9.8.6. JDBC Vendor Dependency

WAS에 의존적인 코드를 사용하는 자바 애플리케이션에 제니퍼를 설치하면 에러가 발생할 수 있다. 왜냐하면 제니퍼가 JDBC 모니터링을 위해서 사용하는 Wrapper 클래스와 WAS가 JDBC 커넥션 풀 구현을 위해서 사용하는 Wrapper 클래스가 호환되지 않기 때문이다.

이 문제를 해결하려면 JDBC 모니터링을 위해 사용하는 Wrapper 클래스가 WAS가 JDBC 커넥션 풀 구현을 위해서 사용하는 Wrapper 클래스와 호환되도록 해야 한다.

우선 제니퍼 에이전트의 enable_jdbc_vendor_wrap 옵션을 true로 설정한다.

```
enable_jdbc_vendor_wrap = true
```

이 옵션을 설정하면 enable_jdbc_oracle_dependency_used 옵션은 무시된다.

WAS 종류와 버전, 그리고 사용하는 데이터베이스에 따라서 JDBC 커넥션 풀 구현을 위해서 사용하는 Wrapper 클래스가 상이하다. 따라서 Wrapper 클래스를 명시적으로 설정해 주어야 한다. 제니퍼가 생성하는 Wrapper 클래스는 Wrapped 클래스 이름에 JWP가 붙는다. 다음은 주요 환경을 위한 설정 예제이다.

• 웹로직 9.x + 데이터소스 + 오라클

```
weblogic.jdbc.wrapper.PoolConnection_oracle_jdbc_driver_T4CConnectionJWP(S) =
    weblogic.jdbc.wrapper.PoolConnection_oracle_jdbc_driver_T4CConnection
weblogic.jdbc.wrapper.Statement_oracle_jdbc_driver_T4CStatementJWP(S) =
    weblogic.jdbc.wrapper.Statement_oracle_jdbc_driver_T4CStatement
weblogic.jdbc.wrapper.PreparedStatement_oracle_jdbc_driver_T4CPreparedStatementJWP(S) =
    weblogic.jdbc.wrapper.PreparedStatement_oracle_jdbc_driver_T4CPreparedStatement
weblogic.jdbc.wrapper.CallableStatement_oracle_jdbc_driver_T4CCallableStatementJWP(S) =
    weblogic.jdbc.wrapper.CallableStatement_oracle_jdbc_driver_T4CCallableStatement
weblogic.jdbc.wrapper.ResultSet_oracle_jdbc_driver_OracleResultSetImplJWP(S) =
    weblogic.jdbc.wrapper.ResultSet_oracle_jdbc_driver_OracleResultSetImpl
```

Notice: 재시작 필요

• 웹로직 6.x + 데이터소스

```
weblogic.jdbc.pool.ConnectionJWP(S) = weblogic.jdbc.pool.Connection
weblogic.jdbc.pool.StatementJWP(S) = weblogic.jdbc.pool.Statement
weblogic.jdbc.pool.PreparedStatementJWP(S) = weblogic.jdbc.pool.PreparedStatement
weblogic.jdbc.pool.CallableStatementJWP(S) = weblogic.jdbc.pool.CallableStatement
weblogic.jdbc.pool.ResultSetJWP(S) = weblogic.jdbc.pool.ResultSet
```

Notice: 재시작 필요

• DriverManager + 오라클

```
oracle.jdbc.driver.T4CConnectionJWP(S) = oracle.jdbc.driver.OracleConnection
oracle.jdbc.driver.T4CStatementJWP(S) = oracle.jdbc.driver.T4CStatement
oracle.jdbc.driver.T4CPreparedStatementJWP(S) = oracle.jdbc.driver.T4CPreparedStatement
oracle.jdbc.driver.T4CCallableStatementJWP(S) = oracle.jdbc.driver.T4CCallableStatement
oracle.jdbc.driver.OracleResultSetImplJWP(S) = oracle.jdbc.driver.OracleResultSetImpl
```

Notice: 재시작 필요

• 톰캣 6.x + 데이터소스

```
org.apache.tomcat.dbcp.dbcp.PoolingDataSource$PoolGuardConnectionWrapperJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingConnection
org.apache.tomcat.dbcp.dbcp.DelegatingStatementJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingStatement
org.apache.tomcat.dbcp.dbcp.DelegatingPreparedStatementJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingPreparedStatement
org.apache.tomcat.dbcp.dbcp.DelegatingCallableStatementJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingCallableStatement
org.apache.tomcat.dbcp.dbcp.DelegatingResultSetJWP(S) =
    org.apache.tomcat.dbcp.dbcp.DelegatingResultSet
```

Notice: 재시작 필요

• **웹스피어 5.x + 데이터소스**

```
com.ibm.ws.rsadapter.jdbc.WSJdbcConnectionJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcConnection
com.ibm.ws.rsadapter.jdbc.WSJdbcStatementJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcStatement
com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatementJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatement
com.ibm.ws.rsadapter.jdbc.WSJdbcCallableStatementJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcCallableStatement
com.ibm.ws.rsadapter.jdbc.WSJdbcResultSetJWP(S) =
    com.ibm.ws.rsadapter.jdbc.WSJdbcResultSet
```

• **오직 오라클**

그런데 만약 애플리케이션의 JDBC드라이버에 대한 의존성이 명확한 경우에는 다음과 같이 간단하게 설정할 수 있다.

```
java.sql.Connection(S) = oracle.jdbc.driver.OracleConnection
java.sql.Statement(S) = oracle.jdbc.driver.OracleStatement
java.sql.PreparedStatement(S) =
oracle.jdbc.driver.OraclePreparedStatement
java.sql.CallableStatement(S) =
oracle.jdbc.driver.OracleCallableStatement
java.sql.ResultSet(S) = oracle.jdbc.driver.OracleResultSet
```

위의 설정은 enable_jdbc_vendor_wrap = false하고 enable_jdbc_oracle_dependency_used=true로 설정한 것과 동일한 기능을 한다.

9.8.7. 오라클 SID 확인 기능

[실시간 모니터링 | 애플리케이션] 메뉴의 JDBC 탭에서 오라클 SID를 확인할 수 있다. 오라클 SID를 확인하기 위해서는이를 위한 설정이 필요하며 그 방법은 데이터베이스 버전에 따라서 상이하다.

오라클 9i 데이터베이스는 제니퍼 에이전트의 dbsession_query 옵션에 다음과 같이 설정한다.

```
dbsession_query = \
oracle.jdbc.driver.OracleConnection: \
select SYS_CONTEXT ('USERENV','SESSIONID') sid from dual;
```

오라클 10g 데이터베이스는 제니퍼 에이전트의 dbsession_query 옵션에 다음과 같이 설정한다.

```
dbsession_query = \
oracle.jdbc.driver.PhysicalConnection: \
select SYS_CONTEXT ('USERENV','SESSIONID') sid from dual;
```

9.8.8. SQL 예외 로그 기록

자바 애플리케이션에서 java.sql.SQLException이 발생하면, 관련 내용을 제니퍼 에이전트 로그 파일에 기록할 수 있다. 이 내용을 기록하려면 제니퍼 에이전트의 enable_sql_error_trace 옵션을 true로 설정한다. 기본 값은 false 이다.

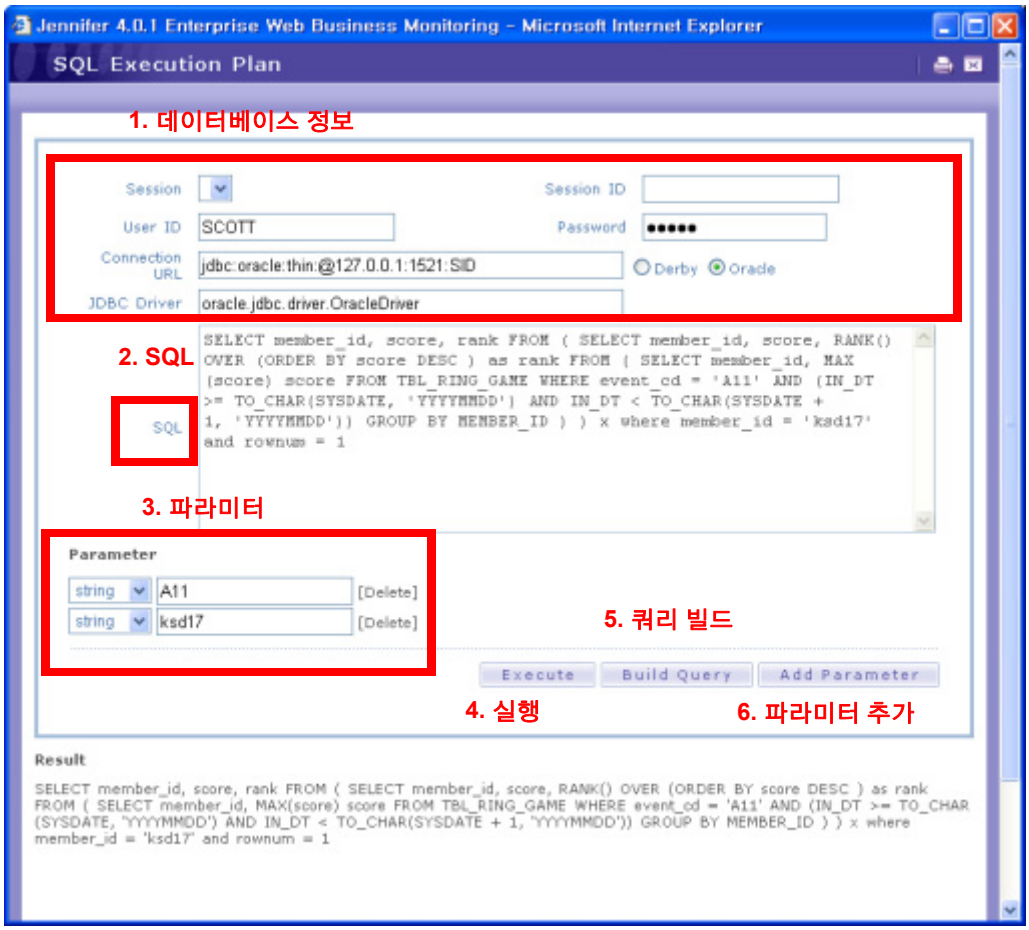
```
enable_sql_error_trace = true
```

단, 정상적인 상황에서도 비즈니스 로직에 종속적으로 java.sql.SQLException이 발생할 수 있다. 따라서 java.sql.SQLException을 오류 혹은 장애로 단정지을 수는 없다. 또한 반복적인 java.sql.SQLException의 발생으로 로그를 빈번하게 기록하면 성능 저하가 발생할 수 있으므로, 필요시에만 이 옵션을 true로 지정하고 평상시에는 false로 지정하는 것을 권장한다.

9.8.9. SQL 실행 계획

X-View 프로파일의 SQL 탭에서 오른쪽 마우스를 클릭하면 나타나는 컨텍스트 메뉴에서 **[쿼리 빌드]** 메뉴를 선택하면, SQL 실행 계획을 확인할 수 있는 팝업 창이 나타난다.

그림 9-15: SQL 실행 계획



1. 데이터베이스 정보 - SQL 실행 계획을 확인할 데이터베이스 정보를 입력한다.
2. SQL - X-View 프로파일 탭에서 선택한 SQL이 자동으로 입력된다. 파라미터 1의 값은 자동으로 치환되고, 파라미터 2로 표시되는 바인딩 파라미터는 하단에 별도로 나타난다.
3. 파라미터 - 파라미터 2에 해당하는 바인딩 파라미터가 나타난다. 파라미터 값을 직접 수정할 수 있다.
4. 실행 - 해당 버튼을 누르면 SQL 실행 계획이 하단 결과 영역에 나타난다.
5. 쿼리 빌드 - 해당 버튼을 누르면 바인딩 파라미터를 바인딩한 SQL이 하단 결과 영역에 나타난다.
6. 파라미터 추가 -해당 버튼을 누르면 파라미터가 추가된다.

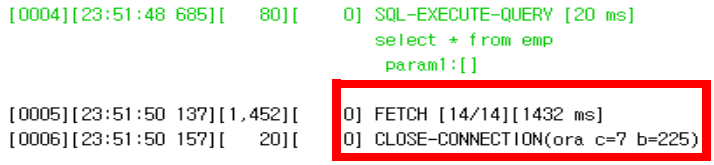
Notice: SQL 실행 계획을 확인하려면 해당 데이터베이스에 대한 JDBC 드라이버를 JENNIFER_HOME/server/common/lib 디렉토리에 복사한 후에 제니퍼 서버를 재시작해야 한다.

9.8.10.SQL에 대한 세션별 데이터베이스 자원 사용량 확인

제니퍼 에이전트의 enable_dbstat 옵션을 true로 설정하면, 해당 SQL에 대한 세션별 데이터베이스 자원 사용량을 X-View 프로파일 데이터의 CLOSE-CONNECTION 메시지에서 확인할 수 있다.

```
enable_dbstat = true
```

그림 9-16: SQL에 대한 세션별 데이터베이스 자원 사용량



여기서 ora는 오라클을 지칭하고, c는 CPU 사용량(1/100초)을 의미하고, b는 오라클 read block(Logical/Physical IO Block)을 의미한다.

9.8.11.DB 모니터링과 예외 발생

DB 모니터링과 관련해서 발생하는 예외는 다음과 같다.

9.8.11.1. SQL 응답 속도의 지연

SQL 응답 속도가 임계치를 초과하면 WARNING_DB_BAD_RESPONSE 예외가 발생한다. 임계치는 제니퍼 에이전트의 sql_bad_responsetime 옵션으로 설정한다. 기본 값은 20000이고 단위는 밀리 세컨드이다.

```
sql_bad_responsetime = 20000
```

9.8.11.2. Fetch 건수 초과

Fetch는 java.sql.ResultSet 객체의 next 메소드를 호출하는 것을 의미한다. 따라서 Fetch 건수는 next 메소드의 호출 건수를 의미한다. 트랜잭션이 수행되는 과정에서 동일한 java.sql.ResultSet 객체에 대한 Fetch 건수가 임계치를 초과하면 WARNING_JDBC_TOOMANY_RS_NEXT 예외가 발생한다. 임계치는 제니퍼 에이전트의 jdbc_resultset_warning_fetch_count 옵션으로 설정한다. 기본 값은 10000 이다.

```
jdbc_resultset_warning_fetch_count = 10000
```

9.8.11.3. DB 리소스 미반환 추적

java.sql.Connection, java.sql.Statement, java.sql.ResultSet 객체 등을 DB 연결 자원이라고 한다. 이 객체들을 사용한 후에 해당 객체의 close 메소드를 호출하지 않으면 DB 연결 자원 미반환 예외가 발생한다.

자바 애플리케이션에서 DB 연결 자원 미반환을 감지하는 시점의 차이가 존재한다. 제니퍼는 DB 연결 자원이 가비지 콜렉션되는 순간에 close 메소드의 수행 여부를 체크하여 DB 연결 자원 미반환을 감지한다. 따라서 가비지 콜렉션이 수행되지 않으면 DB 연결 자원 미반환 예외가 발생하지 않을 수 있고, 가비지 콜렉션이 수행되어도 트랜잭션 종료 시점과 DB 연결 자원 미반환 예외 발생 사이에 시차가 존재하게 된다. 즉 예외가 실제보다 지연되서 발생할 수 있다.

그림 9-17: JDBC 자원 미반환 추적

Notice: Java Application에서는 트랜잭션이 종료되는 시점에는 DB 연결 자원 미반환 여부를 알 수 없기 때문에 X-View 트랜잭션 데이터나 애플리케이션 처리 현황 통계 데이터에는 DB 연결 자원과 관련한 예외가 발생하지 않은 것으로 나타난다.

DB 연결 자원 미반환과 관련한 예외는 다음과 같다.

- **WARNING_DB_CONN_UNCLOSED**

애플리케이션에서 DB Connection 객체를 사용한 후에 close 메소드를 호출하지 않으면 WARNING_DB_CONN_UNCLOSED 예외가 발생한다.

- **WARNING_JDBC_STMT_UNCLOSED**

자바 애플리케이션에서 java.sql.Statement 객체를 사용한 후에 close 메소드를 호출하지 않으면 WARNING_JDBC_STMT_UNCLOSED 예외가 발생한다.

• WARNING_JDBC_PSTMT_UNCLOSED

자바 애플리케이션에서 java.sql.PreparedStatement 객체를 사용한 후에 close 메소드를 호출하지 않으면 WARNING_JDBC_PSTMT_UNCLOSED 예외가 발생한다.

• WARNING_JDBC_CSTMT_UNCLOSED

자바 애플리케이션에서 java.sql.CallableStatement 객체를 사용한 후에 close 메소드를 호출하지 않으면 WARNING_JDBC_CSTMT_UNCLOSED 예외가 발생한다.

• WARNING_JDBC_RS_UNCLOSED

자바 애플리케이션에서 java.sql.ResultSet 객체를 사용한 후에 close 메소드를 호출하지 않으면 WARNING_JDBC_RS_UNCLOSED 예외가 발생한다.

JDBC 자원 미반환 문제를 해결하기 위한 스택트레이스를 제공한다. 일반적으로 이것만으로 충분하지만, 좀더 상세한 스택트레이스가 필요할 수도 있다. 이 경우에는 제니퍼 에이전트의 enable_jdbc_XXXX_fullstack_trace 옵션을 true로 설정한다. 기본 값은 모두 false이다.

```
enable_jdbc_connection_fullstack_trace = true
enable_jdbc_callablestatement_fullstack_trace = true
enable_jdbc_preparedstatement_fullstack_trace = true
enable_jdbc_statement_fullstack_trace = true
enable_jdbc_resultset_fullstack_trace = true
```

Notice: 상세한 스택트레이스를 기록하는 것은 부하가 클 수 있다. 따라서 문제를 해결하는데 필요한 기간에만 일시적으로 적용하는 것을 권장한다.

미반환 JDBC 자원을 제니퍼 에이전트가 임의적으로 반환하도록 설정할 수 있다. 제니퍼 에이전트의 enable_auto_XXXX_close 옵션을 통해서 이를 설정할 수 있다. 기본 값은 모두 false이다.

```
enable_auto_connection_close = true
enable_auto_statement_close = true
enable_auto_preparedstatement_close = true
enable_auto_callablestatement_close = true
enable_auto_resultset_close = true
```

그러나 ApplicationServer 혹은 커넥션 풀 라이브러리에서 JDBC자원을 관리하고 있는 경우, java.sql.Connection 객체를 자동으로 반환하는 것은 위험할 수 있다.

9.8.11.4. 오라클 XML DB XMLType 누수 추적

오라클 XML DB를 사용하는 경우에 oracle.xdb.XMLType 객체를 사용한 후에 close 메소드를 호출하지 않으면 WARNING_ORACLE_XMLTYPE_UNCLOSED 예외가 발생한다. 단, 이 예외는 제니퍼 에이전트의 enable_oracle_xdb_xmltype_trace 옵션을 true로 설정한 경우에만 발생한다. 기본 값은 true이다.

```
enable_oracle_xdb_xmltype_trace = true
```

9.8.11.5. SQL수행 후 Uncommit/Rollback 추적

애플리케이션에서 DB 트랜잭션을 시작(false를 파라미터로 java.sql.Connection 객체의 setAutoCommit 메소드를 호출)하고 java.sql.Statement 객체의 execute, executeBatch, executeUpdate 등의 메소드를 호출한 후에, 명시적으로 DB 트랜잭션을 종료(rollback 혹은 commit 메소드를 호출)하지 않으면 WARNING_DB_UN_COMMIT_ROLLBACK 예외가 발생한다. 이런 경우에 제니퍼 에이전트의 enable_rollback_uncommited_close 옵션을 통해서 관련 java.sql.Connection 객체를 자동으로 rollback할 수 있다. 기본 값은 false이다.

```
enable_rollback_uncommited_close = true
```

Notice: 검증된 시스템이 아닌 경우에 이 옵션을 true로 설정하는 것을 권장하지 않는다. java.sql.Connection 객체의 close 메소드가 호출되는 시점의 자바 쓰레드와는 다른 자바 쓰레드에서 rollback 메소드가 호출되면 장애가 발생할 수 있다.

```
ignore_rollback_uncommited_error = true
```

장기적으로는 SELECT 문이 exeucteQuery 메소드로 처리되도록 해당 소스 코드를 수정하는 것을 권장한다.

9.8.11.6. DB 커넥션 객체의 중복 할당

동일한 DB Connection 객체가 여러 개의 서비스 쓰레드에 중복으로 할당되면 애플리케이션이 정지되는 것과 같은 치명적인 문제를 야기할 수 있다. 이런 현상이 나타나면 제니퍼는 WARNING_DB_CONN_ILLEGAL_ACCESS 예외를 발생시킨다. 이런 중복 할당이 일어나는 원인은 다음과 같다.

- 멤버 필드나 static 변수로 DB Connection 객체를 선언한 경우

- 프로그래밍 상의 실수로 한 트랜잭션에서 동일한 DB Connection 객체를 두번 이상 반환하는 경우에, 커넥션 풀이 반환된 DB Connection 객체를 별개로 인식하여 풀링하는 경우

• DB 중복 풀링 구조로 인한 중복 할당 예외

그런데 중복 할당이 아님에도 불구하고 애플리케이션 내부 구조로 인해서 이 경보가 발생할 수도 있다. 이런 현상을 유발하는 애플리케이션 내부 구조를 중복 풀링 구조라고 한다.

예를 들어, 제니퍼가 DataSource(유형1)나 DriverManager(유형2)로 JDBC 모니터링을 하는데 애플리케이션 내부에 또다른 커넥션 풀이 존재하면, 제니퍼는 하나의 커넥션이 반환되지 않고 여러 쓰레드에서 사용되고 있다고 인식하고 WARNING_JDBC_CONN_ILLEGAL_ACCESS 예외를 발생시킨다.

따라서 WARNING_JDBC_CONN_ILLEGAL_ACCESS 예외가 발생하면 먼저 중복 풀링 구조인지를 확인하고 JDBC 커넥션 추적 설정을 변경하도록 한다.

Notice: 중복 풀링 구조 때문에 WARNING_JDBC_CONN_ILLEGAL_ACCESS 예외가 발생하면 그 빈도가 매우 높고 거의 대부분의 트랜잭션에서 발생하는 특징이 있으므로 프로그램 오류에 의한 중복 할당과 쉽게 구분할 수 있다. 또한 중복 풀링 구조는 불필요한 이중 작업이 수행되기 때문에 애플리케이션 구조 개선을 검토할 필요도 있다.

9.9. 사용자 정의형 리소스 모니터링

JDBC 자원뿐만 아니라 임의의 리소스에 대한 누수 현상도 모니터링할 수 있다. 예를 들어, 네트워크 성능 향상을 위해서 java.net.Socket 객체를 풀링하는 경우에 누수 현상을 모니터링할 수 있다.

그러나 모든 리소스 누수 현상을 모니터링할 수 있는 것은 아니고, 리소스를 획득하는 메소드와 반환하는 메소드가 명확한 경우에만 이 기능을 사용할 수 있다. 트랜잭션을 처리하는 과정에서 리소스를 획득하는 메소드의 호출 건수와 리소스를 반환하는 메소드의 호출 건수가 동일하지 않으면 WARNING_RESOURCE_LEAK 예외가 발생한다.

사용자 정의형 자원 모니터링은 제니퍼 에이전트의 leakcheck_XXXX_get과 leakcheck_XXXX_close 옵션을 통해서 설정한다.

예를 들어, mypool과 yourpool이라는 리소스 풀이 있다면 제니퍼 에이전트의 leakcheck_XXXX_get 옵션을 통해서 리소스 획득 메소드를 설정한다. XXXX 대신에 리소스 풀 이름을 사용한다.

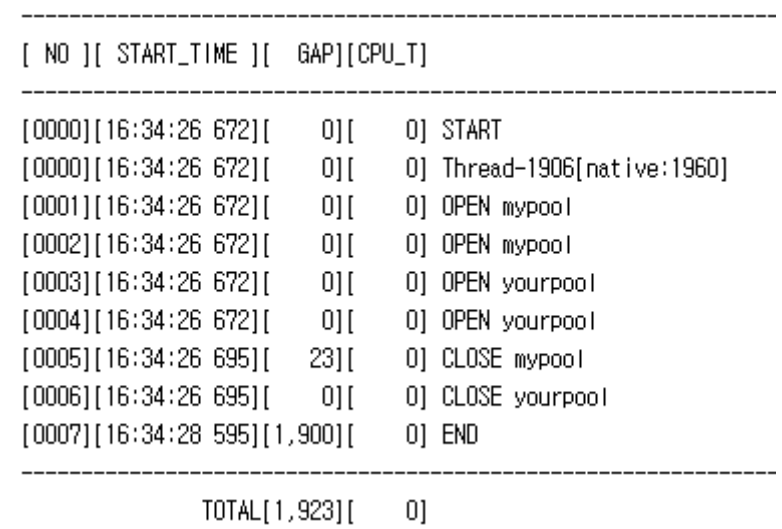
```
leakcheck_mypool_get = example.Leak.get();
leakcheck_yourpool_get = example.Leak2.get();
```

그리고 제니퍼 에이전트의 leakcheck_XXXX_close 옵션을 통해서 리소스 반환 메소드를 설정한다. XXXX 대신에 리소스 풀 이름을 사용한다.

```
leakcheck_mypool_close = example.Leak.release(Object);
leakcheck_yourpool_close = example.Leak2.release(Object);
```

사용자 정의 리소스 풀 사용 내역을 X-View 프로파일 데이터에서 확인할 수 있다.

그림 9-18: 사용자 정의 리소스 풀에 대한 X-View 프로파일



9.10. 쓰레드 모니터링(CPU튜닝을 위한 HOW TO)

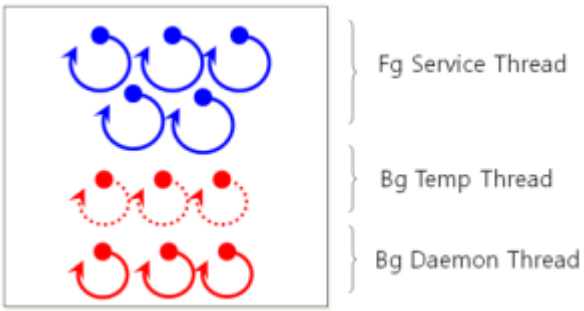
제니퍼는 Java1.5환경에서 JMX를 이용한 쓰레드 상세 모니터링 기능을 제공한다. 제니퍼는 서비스 중심적인 모니터링 솔루션이다. 즉 사용자의 요청을 처리하는 쓰레드를 중심으로 서비스 성능을 추적한다.

그런데 간혹 프로세스의 CPU사용량을 튜닝해야하는 상황이 있다. 이런 경우에는 쓰레드 중심적인 모니터링을 수행해야 한다. 즉 어떤 쓰레드가 CPU를 많이 사용하는지를 판별하고 해당 쓰레드를 튜닝해야 하는데 이 쓰레드는 크게 서비스 쓰레드와 백그라운드 쓰레드로 구분할 수 있다.

웹 애플리케이션 서버에서 서비스 쓰레드는 일반적으로 정형화된 쓰레드 이름을 가지고 있다. 따라서 우리는 쓰레드 이름을 통해 서비스쓰레드와 백그라운드 쓰레드를 구분할 수 있다.

이렇게 구분하여 서비스 쓰레드가 CPU를 많이 사용하는지 백그라운드 쓰레드가 CPU를 많이 사용하는지를 확인고 튜닝해야 한다.

그림 9-19: 쓰레드 모니터링



제니퍼는 프로세스 내부의 쓰레드를 이름으로 구분하고 그들의 CPU사용량을 모니터링 함으로써 CPU튜닝을 위한 기반정보를 제공한다.

9.10.1. 설정 및 환경

JENNIFER AGENT 설정

쓰레드 상세 모니터링은 제니퍼 에이전트 환경이 Java 1.5 이상에서만 사용할 수 있다.

```
extra_enable=true
extra_agent_class=jennifer.extra.ThreadMon
extra_agent_classpath=/jennifer/agent/lwst40.custom.jar
extra_data_interval=3000
xtmon_fg_name=http-8080
xtmon_topn=20
xtmon_file_save=true
```

제니퍼에이전트의 모니터링 확장 기능인 ExtraAgent기능을 사용한다.

- extra_enable : Extra Agent기능을 활성화한다.

- extra_agent_class : 쓰레드 정보를 수집할 Extra Agent 어댑터이다.
- extra_agent_classpath : Extra Agent 모듈를 포함하고 있는 Jar파일
- extra_data_interval : Extra Agent가 데이터를 수집하는 주기
- xtmon_file_save : 쓰레드 모니터링 데이터 중에서 주요 데이터를 파일에 저장
- xtmon_topn : 쓰레드 중에서 상위 20위 내의 CPU사용량을 보이는 쓰레드만 리스트로 보여줌
- xtmon_fg_name : 서비스 쓰레드 이름 규칙 ';'를 사용하여 여러개를 설정할 수 있다.

JENNIFER SERVER 설정

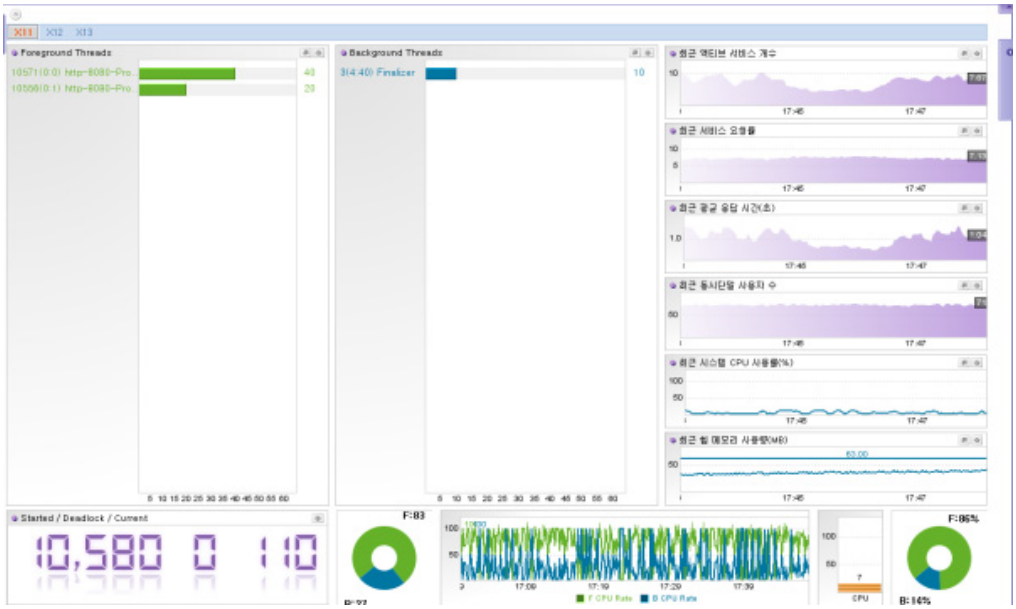
제니퍼 서버에는 메모리 상세 정보를 모니터링 하기 위한 화면을 등록해야 한다. 제니퍼 4.2 서버를 처음 기동했다면 자동으로 [실시간 모니터링| 쓰레드]에 등록되어 있고 권한만 빠져 있는 상태가 된다. 만약 다른 버전이거나 메뉴가 존재하지 않으면 다음 URL을 메뉴에 등록한다.

```
real_thread.jsp
```

9.10.1.1. 쓰레드 대시보드와 해석

쓰레드 상세 모니터링의 대시보드는 아래와 같다.

그림 9-20: 쓰레드 대시보드



Warning: 화면 상단에서 에이전트를 선택해야만 그래프가 보인다.

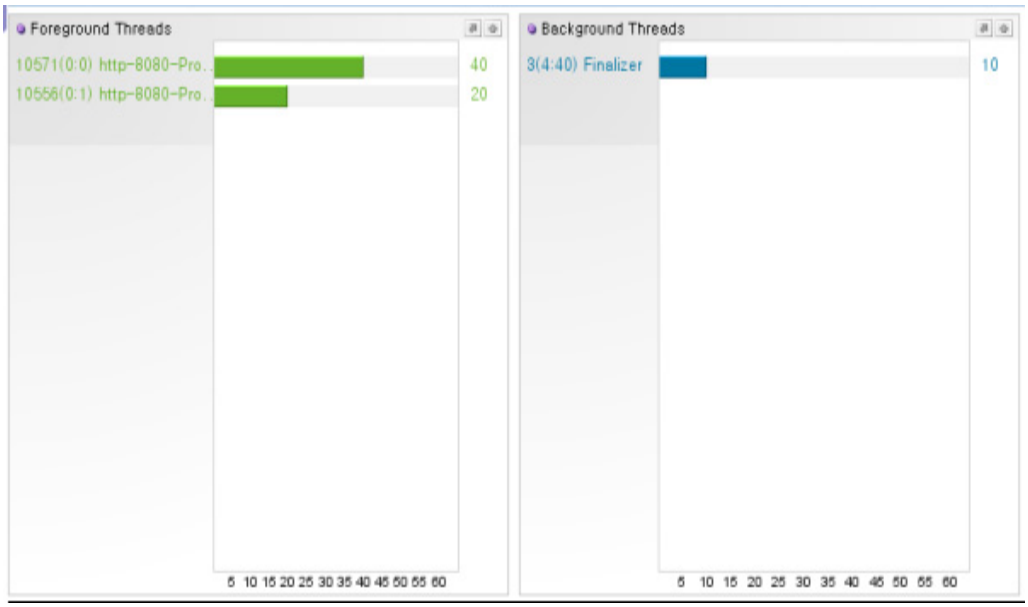
중요 그래프

주요 그래프 내용은 다음과 같다

Forground/Background Threads

서비스(Forground)쓰레드와 백그라운드 쓰레드들 중에서 단위시간 (기본 3000ms) 동안 사용한 CPU시간이 긴 순서데로 리스트로 보여줌 어떤 특정 쓰레드의 CPU사용량이 지속적으로 많으면 원인을 찾아봐야 한다.

그림 9-21: Forground/Background Threads



서비스(Forground)쓰레드와 백그라운드 쓰레드들 중에서 단위시간 (기본 3000ms) 동안 사용한 CPU시간이 긴 순서데로 리스트로 보여줌 어떤 특정 쓰레드의 CPU사용량이 지속적으로 많으면 원인을 찾아봐야 한다.

Thread Count

그림 9-22: Thread Count

대시보드의 하단 그래프 중에서 왼쪽은 몇개의 쓰레드가 실행되었는지를 보여준다.

- Started : 자바 프로세스가 기동된 후에 실행된 쓰레드 총갯수
- Deadlock : Deadlock에 빠진 쓰레드수 (0이 아니면 무조건 문제임)
- Current : 현재 수행중인 쓰레드

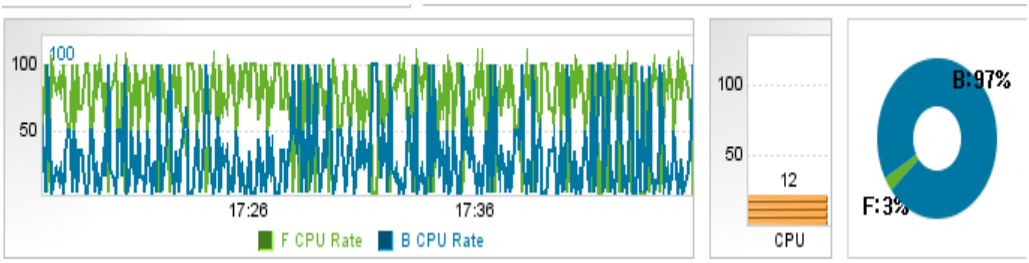
두번에 F:99, B:99를 표시하는 그래프는 현재 수행중이 쓰레드중에서 Forground 쓰레드 수와 Background 쓰레드 수의 비율을 나타낸다.

CPU 사용량 비율

백그라운드 쓰레드가 CPU를 많이 사용하는지 포그라운드 쓰레드사 CPU를 많이 사용하는지를 나타내는 그래프이다.

시스템 CPU사용량을 보고 CPU사용비율을 확인한후 어떻게 CPU사용을 튜닝할지를 연구해야 한다.

그림 9-23: CPU 사용량 비율



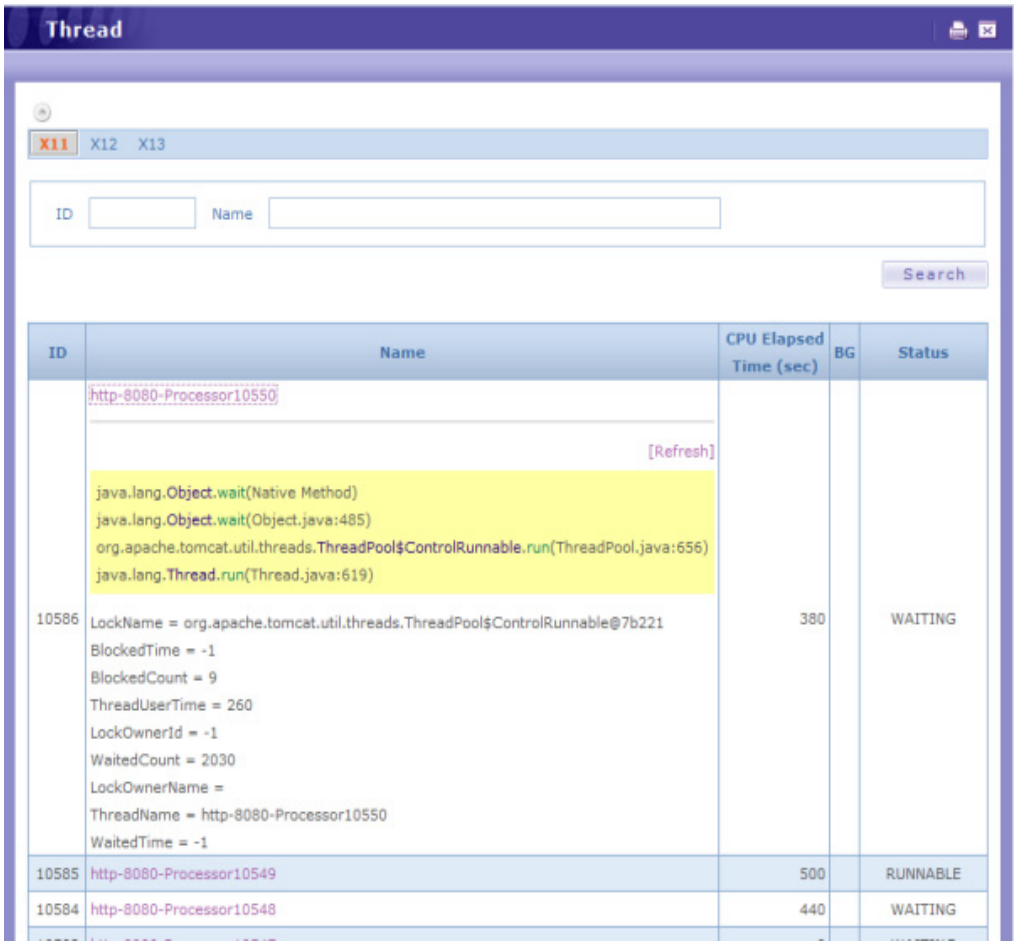
백그라운드 쓰레드가 CPU를 많이 사용하는지 포그라운드 쓰레드사 CPU를 많이 사용하는지를 나타내는 그래프이다.

시스템 CPU사용량을 보고 CPU사용비율을 확인한후 어떻게 CPU사용을 튜닝할지를 연구해야 한다.

실시간 상세 조회

수행중인 쓰레드의 Stack 정보를 상세하게 조회할 수있다.

그림 9-24: 실시간 상세 조회



수행중인 쓰레드의 Stack 정보를 상세하게 조회할 수있다.

언제 시작되었고 지금까지 얼마의 CPU를 사용했는지 등을 알 수있다. 또한 해당 쓰레드의 콜 스택을 조회할 수 있다.

각 쓰레드별 CPU Time은 쓰레드가 실행되어 지금까지 사용한 CPU점유시간이다. 이 값을 보고 어느쓰레드가 CPU를 많이 사용하는지를 판별할 수 있다.

9.11. 자바 메모리 상세 모니터링

자바 프로세스의 성능을 튜닝할때 메모리 튜닝을 빼놓을 수는 없다. 강력한 힙 메모리 관리 모델을 제시함으로써 이전보다 편리하고안정적인 프로그램이 가능해 졌지만 여전히 메모리는 관리영역중에 1순위이다. 힙메모리를 통한 엔진레벨의 관리는 메모리를 안정화 시켰지만 CPU 과도한 사용을 야기했다. 더이상 메모리 부족은 메모리의 문제로 나타나는 것이 아니라 필요이상의 CPU사용으로 나타나기 때문에 여전히 메모리 관리와 튜닝은 쉽지 않은 영역이다.

그런데 Java5 부터는 이전보다 강력한 메모리 모니터링 수단을 제공한다. 이것을 이용하여 힙메모리 혹은 NON-HEAP 상태를 모니터링 하거나 메모리를 해제하기 위한 GC의 움직임이나 오버헤드를 실시간 관찰할 수 있다.

제니퍼에서는 이것을 이용하여 보다 운영자 직관적으로 메모리 상태를 모니터링 할 수 있도록 기능을 제공하고 있다.

9.11.1. 설정 및 환경

9.11.1.1. JENNIFER AGENT 설정

메모리 상세 모니터링은 제니퍼 에이전트 환경이 Java 1.5 이상에서만 사용할 수 있다.

```
extra_enable=true
extra_agent_class=jennifer.extra.MemoryMon
extra_agent_classpath=/jennifer/agent/lwst40.custom.jar
extra_data_interval=3000
xmmon_file_save=true
```

제니퍼에이전트의 모니터링 확장 기능인 ExtraAgent기능을 사용한다.

- extra_enable : Extra Agent기능을 활성화한다.
- extra_agent_class : 메모리 정보를 수집할 Extra Agent 어댑터이다.
- extra_agent_classpath : Extra Agent 모듈를 포함하고 있는 Jar파일
- extra_data_interval : Extra Agent가 데이터를 수집하는 주기
- xmmon_file_save : 메모리 모니터링 데이터 중에서 주요데이터를 파일에 저장

9.11.1.2. JENNIFER SERVER 설정

제니퍼 서버에는 메모리 상세 정보를 모니터링 하기 위한 화면을 등록해야 한다. 제니퍼 4.2 서버를 처음 기동했다면 자동으로 [실시간 모니터링| 메모리]에 등록되어 있고 권한

만 빠져 있는 상태가 된다. 만약 다른 버전이거나 메뉴가 존재하지 않으면 다음 URL을 메뉴에 등록한다.

```
real_memory.jsp
```

9.11.2. 메모리 대시보드와 해석

메모리 상세 모니터링의 대시보드는 아래와 같다

그림 9-25: 메모리 대시보드

Notice: 화면 상단에서 에이전트를 선택해야만 그래프가 보인다.

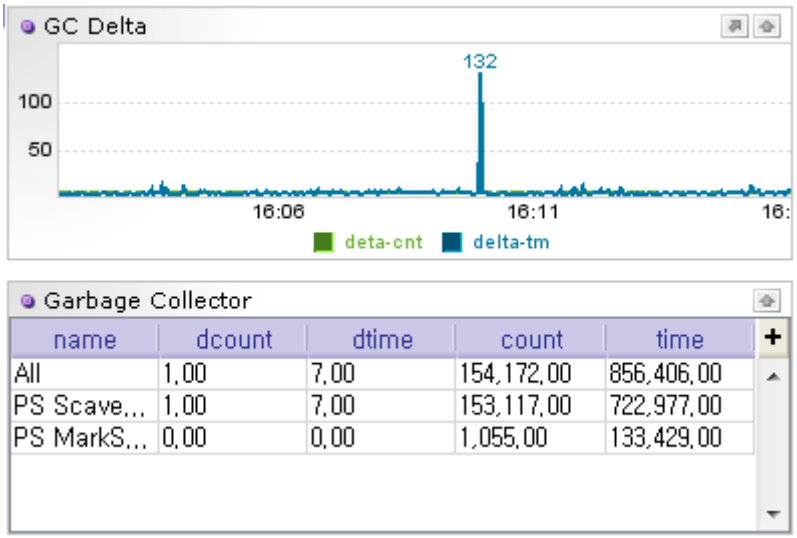
9.11.3. 중요 그래프

주요 그래프 내용은 다음과 같다

9.11.3.1. GC Delta

단위 시간(기본 3000ms) 동안 GC가 발생한 횟수와 그것을 위해 사용한 시간을 라인 그래프로 보여진다. GC가 발생할때 걸린 시간을 실시간으로 모니터링 할 수있다.

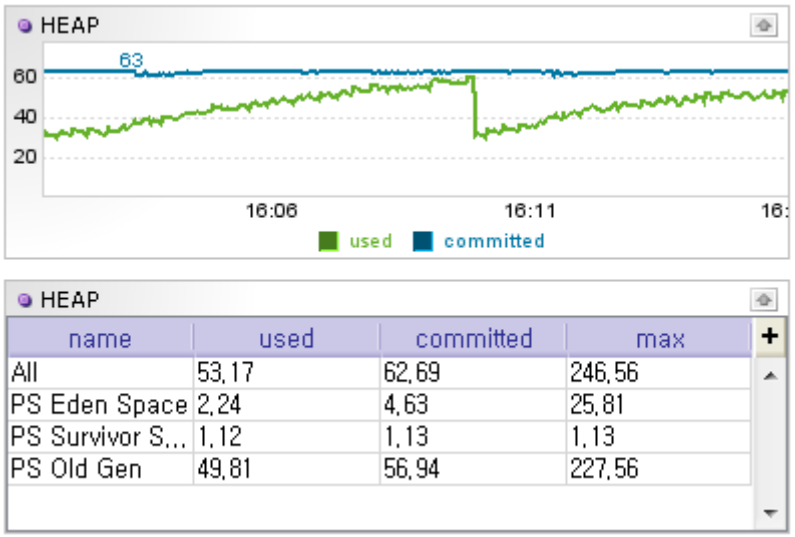
그림 9-26: GC Delta



9.11.3.2. 영역별 HEAP 사용량

힙메모리 사용량의 변화를 모니터링 한다.

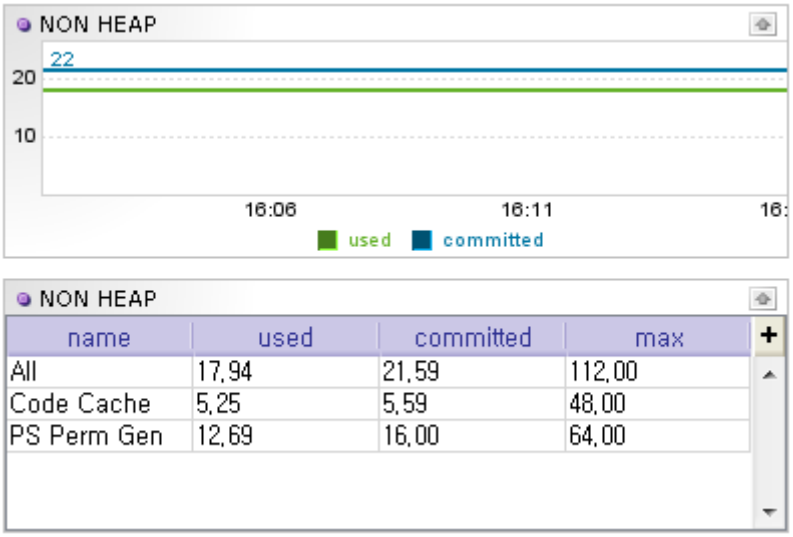
그림 9-27: 영역별 HEAP 사용량



9.11.3.3. 영역별 NON-HEAP 사용량

NON-HEAP 메모리 사용량의 변화를 모니터링 한다.

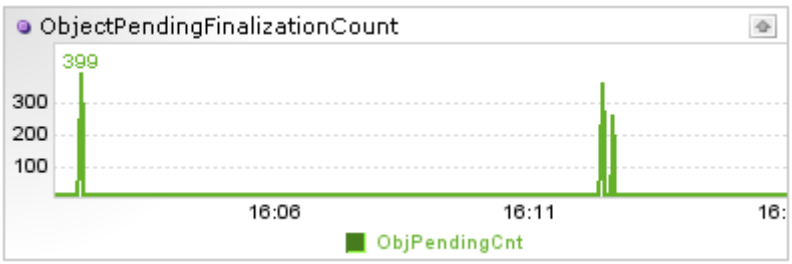
그림 9-28: 영역별 NON-HEAP 사용량



9.11.3.4. Object Pending Finalization Count

finalize()메소드가 호출되기를 기다리를 객체 수 로써 만약 이값이 높다면 finalze()메소드에 많은 로직이 존재할 수 있음으로 분석이 필요할 수 있다.

그림 9-29: Object Pending Finalization Count



9.11.3.5. GC Total

프로세스가 기동되고 수행된 GC의 총수와 그로인해 사용되었던 시간정보를 나타내기 위한 화면이다.

그림 9-30: GC Total

기타 시스템 그래프

모니터링중인 에이전트의 액티브서비스, 시스템 CPU, TPS, 응답시간은 GC로 영향받는 에이전트의 상황을 비교하기 위해 포함되어있다.

9.11.4. 데이터 조회

메모리 상세 모니터링은 메모리 현황을 모니터링하고 튜닝하여 최적의 실행환경을 만들기 위한 목적이다. 따라서 다른 성능 데이터들 처럼 장시간 보관하거나 관리할 필요가 없다.

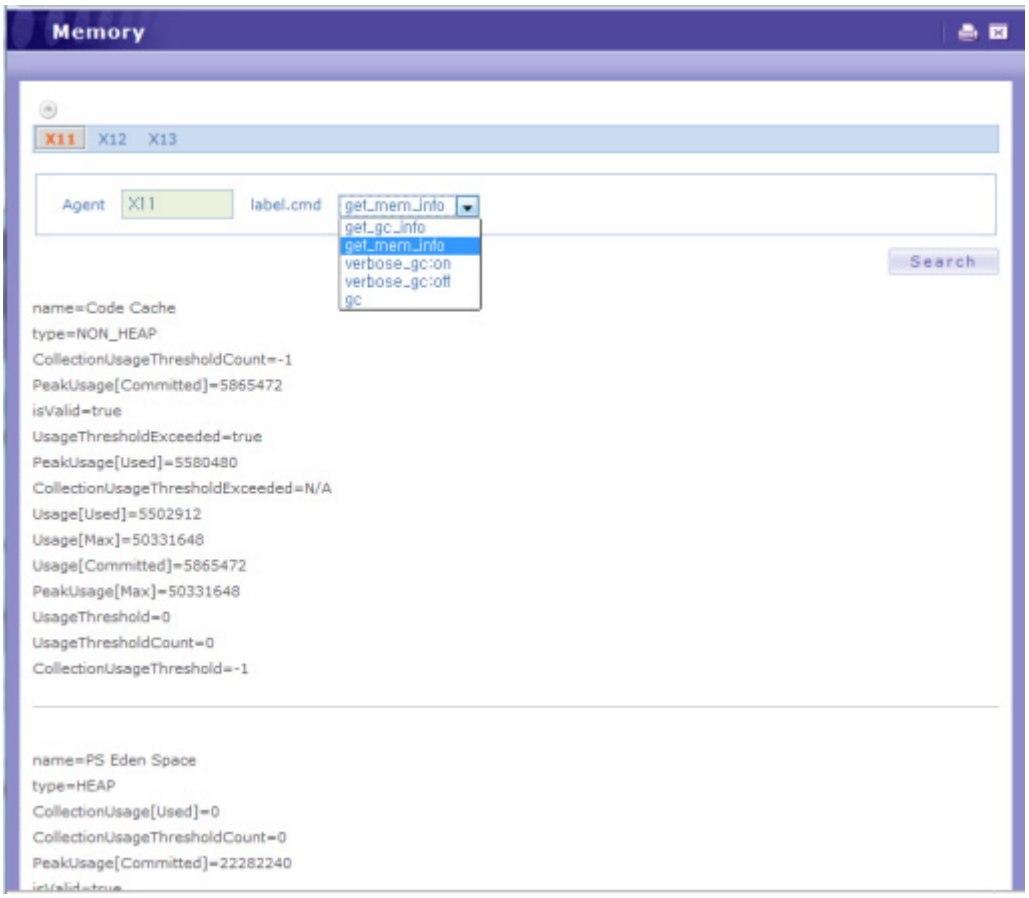
이에 따라 제니퍼는 현재의 프로세스 메모리 현황을 조회하거나 간단한 제어를 할 수있는 기능과 상대적으로 단기간의 메모리 상세데이터를 엑셀로 다운받아 이차가공하는 기능을 제공한다.

9.11.4.1. 실시간 데이터 조회

GC Delta(왼쪽 상단) 그래프의 좌측 버튼을 클릭하면 현재 에이전트의 메모리 현황을 상세하게 조회할 수 있는 기능이 제공된다. 사용할 수있는 중요 기능은 아래와 같다.

- 메모리 상세 현황조회
- VERBOSE:GC on/off
- 강제 GC

그림 9-31: 실시간 데이터 조회



Notice: 메모리 상세 조회시 각 속성의 의미는 JVM 설명서를 참조해야 한다.

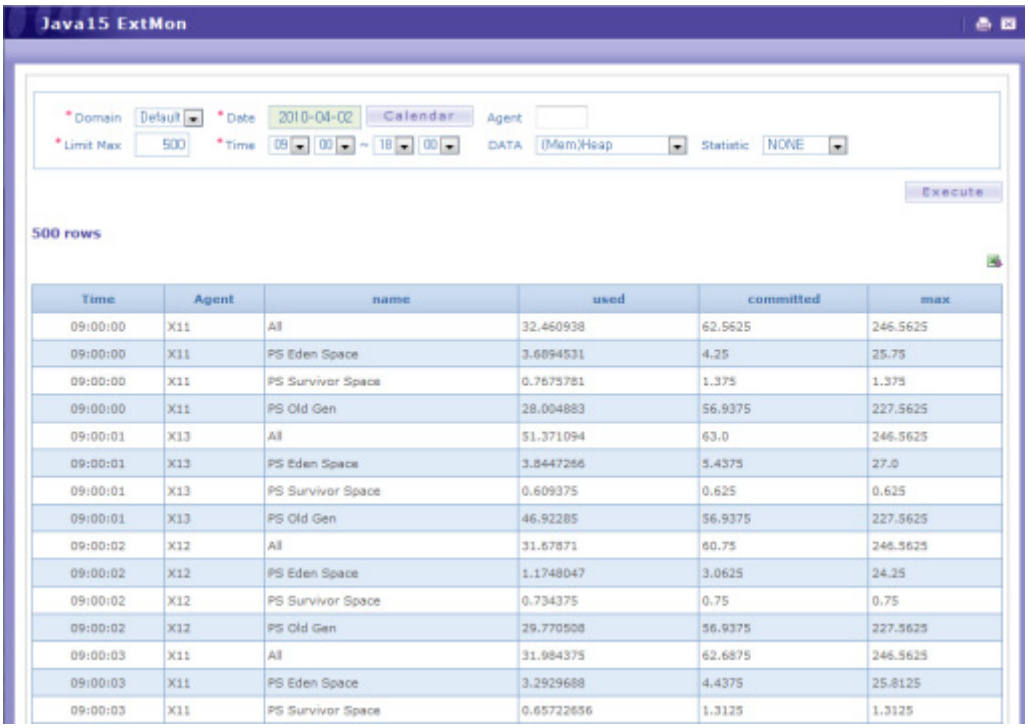
9.11.4.2. 저장 데이터 조회

에이전트에 메모리 모니터링 기능을 설정할때 제니퍼 서버에서 주요 데이터를 저장하도록 할 수 있다.

```
xmmon_file_save = true
```

그러면 메모리 관련 상세 정보가 REMON포멧으로 저장된다. 이데이터는 화면 우측 하단의 [검색]버튼을 클릭하여 조회할 수 있다.

그림 9-32: 저장 데이터 조회



Time	Agent	name	used	committed	max
09:00:00	X11	All	32.460938	62.5625	246.5625
09:00:00	X11	PS Eden Space	3.6894531	4.25	25.75
09:00:00	X11	PS Survivor Space	0.7675781	1.375	1.375
09:00:00	X11	PS Old Gen	28.004883	56.9375	227.5625
09:00:01	X13	All	51.371094	63.0	246.5625
09:00:01	X13	PS Eden Space	3.8447266	5.4375	27.0
09:00:01	X13	PS Survivor Space	0.609375	0.625	0.625
09:00:01	X13	PS Old Gen	46.92285	56.9375	227.5625
09:00:02	X12	All	31.67871	60.75	246.5625
09:00:02	X12	PS Eden Space	1.1748047	3.0625	24.25
09:00:02	X12	PS Survivor Space	0.734375	0.75	0.75
09:00:02	X12	PS Old Gen	29.770508	56.9375	227.5625
09:00:03	X11	All	31.984375	62.6875	246.5625
09:00:03	X11	PS Eden Space	3.2929688	4.4375	25.8125
09:00:03	X11	PS Survivor Space	0.65722656	1.3125	1.3125

10

닷넷 시스템에서 리소스, DB 연결 모니터링

CPU와 메모리 등의 시스템 자원 및 DB 모니터링 하는 방법을 설명한다.

10.1. CPU 모니터링

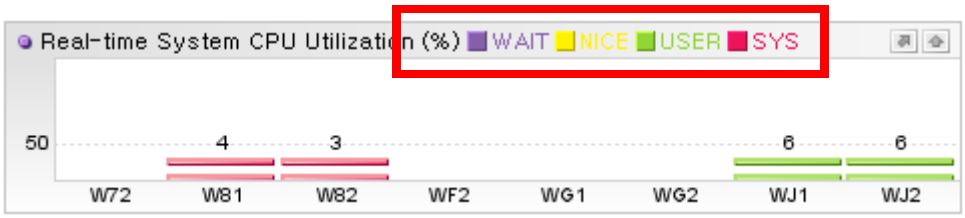
제니퍼 에이전트를 통해서 애플리케이션이 동작하는 하드웨어의 시스템 CPU 사용률과 해당 애플리케이션이 사용하는 프로세스 CPU 사용률을 모니터링 할 수 있다. 그리고 특정 트랜잭션의 처리 과정에서 사용한 트랜잭션 CPU 점유 시간도 모니터링 할 수 있다. 또한 제니퍼 에이전트의 설치와 상관없이 WMOND를 통해서 임의의 하드웨어의 CPU 개수당 CPU 사용률도 모니터링 할 수 있다.

10.1.1.시스템 CPU 사용률과 프로세스 CPU 사용률

시스템 CPU 사용률은 시스템의 전체 CPU 사용률을 의미하고, 프로세스 CPU 사용률은 제니퍼 에이전트를 설치한 애플리케이션이 사용하는 CPU 사용률을 의미한다.

시스템 CPU 사용률과 프로세스 CPU 사용률은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 값을 이퀄라이저 차트와 런타임 라인 차트를 통해서 확인할 수 있다.

그림 10-1: 실시간 시스템 CPU 사용률 차트



그런데 CPU 사용률은 CPU 사용 영역별로 구분되고, 이퀄라이저 차트는 CPU 사용률을 영역별로 색상을 구분하여 표시한다. 다음은 CPU 사용 영역에 대한 설명이다.

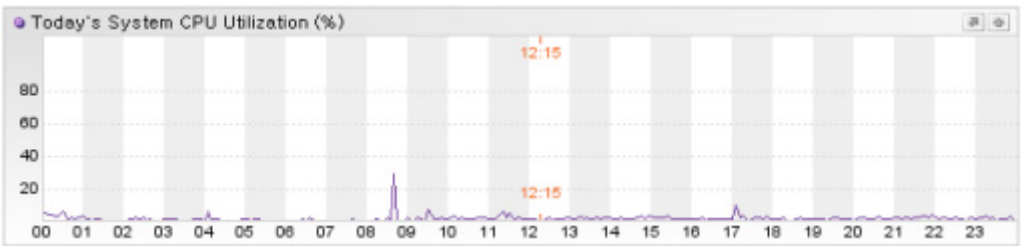
표 10-1: CPU 사용 영역

CPU 사용 영역	설명
WAIT	(닷넷 에이전트에서는 사용 안함)
NICE	(닷넷 에이전트에서는 사용 안함)
USER	사용자 수준(애플리케이션)에서 사용하는 CPU 사용률을 의미한다.
SYS	시스템 수준(커널)에서 사용하는 CPU 사용률을 의미한다.

Notice: 이퀄라이저 차트는 프로세스 CPU 사용률과 에이전트가 수집한 CPU 개수당 CPU 사용률도 CPU 사용 영역별로 색상을 구분하여 표시한다.

또한 시스템 CPU 사용률과 자바 프로세스 CPU 사용률에 대한 5분 평균 값은 PERF_X_01~31 테이블의 SYS_CPU 칼럼과 JVM_CPU 칼럼에 저장되고, 특정 날짜의 CPU 사용률 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 10-2: 금일 시스템 CPU 사용률



10.1.2.트랜잭션 CPU 점유 시간

트랜잭션 CPU 점유 시간은 트랜잭션이 수행되는 과정에서 사용한 CPU 사용 시간을 의미한다. 이는 애플리케이션 처리 현황 통계 데이터와 X-View 프로파일 데이터로 수집된다.

그리고 tpmC를 통해서 트랜잭션 CPU 점유 시간을 하드웨어 별로 객관적으로 비교할 수 있다. tmpC는 TPC(Transaction Processing Performance Council, http://www.tpc.org)의 TPC-C 벤치마크 시나리오에 대한 1분당 최대 처리 건수를 나타내는 수치로써, 하드웨어(주로 CPU)의 성능을 측정하는 대표적인 방법이다.

하드웨어의 CPU 처리 용량이 다르기 때문에 트랜잭션 CPU 점유 시간만으로는 해당 트랜잭션이 성능에 미치는 영향을 객관적으로 파악할 수 없다. 예를 들어, CPU 처리 용량이 큰 하드웨어에서 처리된 트랜잭션 A의 CPU 점유 시간이 CPU 처리 용량이 작은 하드웨어에서 처리된 트랜잭션 B의 CPU 점유 시간보다 작은 경우에도 성능에 미치는 절대적인 영향은 트랜잭션 A가 더 높을 수도 있다. 즉 트랜잭션 A를 CPU 처리 용량이 작은 하드웨어에서 처리하면 트랜잭션 B보다 CPU 점유 시간이 클 수 있다는 것이다.

따라서 tpmC로 하드웨어의 CPU 처리 용량 변수를 통제한 상태에서 트랜잭션이 성능에 미치는 영향을 파악해야 한다.

이를 위해서 제니퍼 에이전트의 approximate_tpmc_on_this_system 옵션으로 제니퍼 에이전트를 설치한 해당 하드웨어의 tpmC 값을 설정하도록 한다.

```
approximate_tpmc_on_this_system = 30000
```

10.1.3.CPU 모니터링과 경보 발령

시스템 CPU 사용률과 프로세스 CPU 사용률이 임계치를 초과하면 제니퍼 서버는 경보를 발령한다. 자세한 사항은 경보와 예외 모니터링을 참조한다.

10.2. 메모리 모니터링

제니퍼 에이전트를 통해서 자바 애플리케이션이 동작하는 하드웨어의 시스템 메모리 사용량과 해당 자바 애플리케이션이 사용하는 자바 프로세스 메모리 사용량을 모니터링할 수 있다. 그리고 자바 애플리케이션의 자바 힙 메모리 전체와 자바 힙 메모리 사용량도 모니터링할 수 있다.

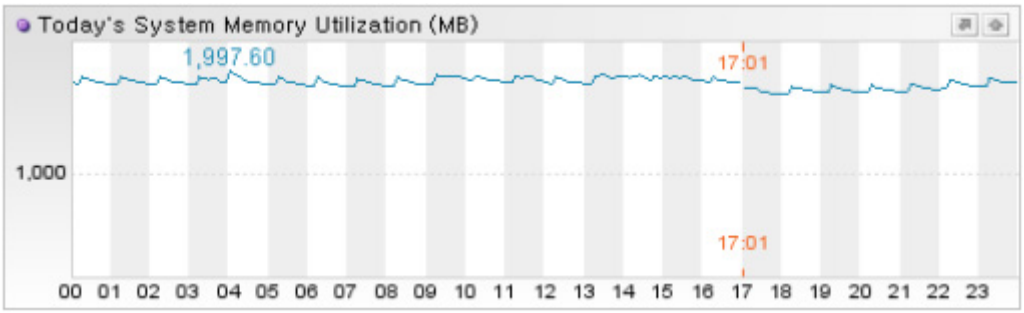
10.2.1.시스템 메모리 사용량과 자바 프로세스 메모리 사용량

시스템 메모리 사용량은 시스템의 메모리 사용량을 의미하고, 자바 프로세스 메모리 사용량은 제니퍼 에이전트를 설치한 자바 애플리케이션이 사용하는 메모리 사용량을 의미한다. 단위는 MB이다.

시스템 메모리 사용량과 자바 프로세스 메모리 사용량은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 30초 평균 값을 런타임 라인 차트를 통해서 확인할 수 있다.

또한 시스템 메모리 사용량과 자바 프로세스 메모리 사용량에 대한 5분 평균 값은 PERF_X_01~31 테이블의 SYS_MEM_USED 칼럼과 JVM_NAT_MEM 칼럼에 저장되고, 특정 날짜의 메모리 사용량 변화 추이는 라인 차트를 통해서 확인할 수 있다.

그림 10-3: 금일 시스템 메모리 사용량



Notice: 시스템 CPU 사용률과 동일하게 시스템 메모리 사용량과 자바 프로세스 메모리 사용량을 수집하려면 Native 모듈이 올바르게 설치되어 있어야 한다.

10.2.2.CLR 힙 메모리 사용량

CLR 힙 메모리 사용량은 다음의 코드로 얻어진 값을 의미한다.

```
_jvmmem.used = GC.GetTotalMemory(false);
```

앞의 코드를 통해서 수집한 값의 단위는 바이트이지만, 제니퍼는 이를 MB로 전환해서 사용한다.

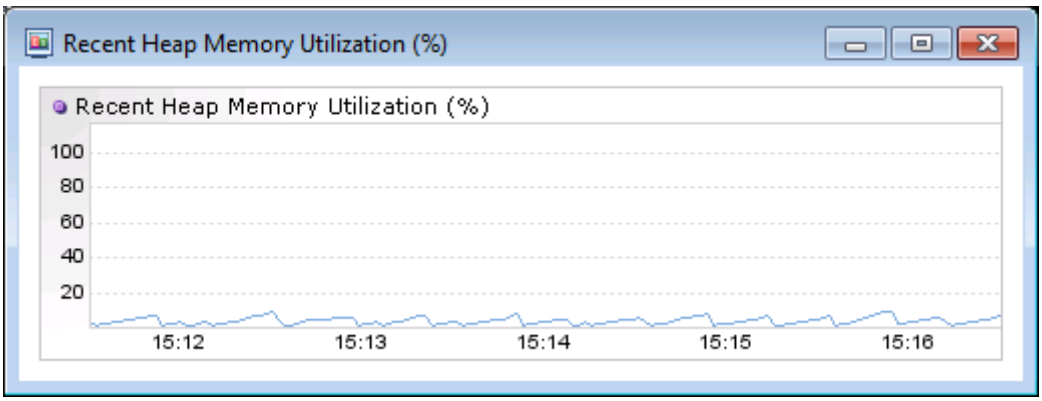
힙 메모리 사용량은 제니퍼 에이전트가 수집하는 일반 성능 데이터로 실시간 30초 평균 값을 런타임 라인 차트를 통해서 확인할 수 있다. 또한 힙 메모리 사용량에 대한 5분 평균

값은 PERF_X_01~31 테이블의 HEAP_TOTAL 칼럼과 HEAP_USED 칼럼에 저장되고, 특정 날짜의 힙 메모리 사용량 변화 추이는 라인 차트를 통해서 확인할 수 있다. .

Notice: 힙 메모리 사용률은 시스템의 물리 메모리 전체에 대한 힙 메모리 사용량의 비율을 의미한다. 단위는 퍼센트이다.

힙 메모리 사용량, 힙 메모리 사용률 등을 나타내는 런타임 라인 차트에서는 특정 제니퍼 에이전트에 대한 가비지 콜렉션을 수행할 수 있다.

그림 10-4: CLR 힙 메모리 사용량



10.3. DB 연결 모니터링

데이터베이스 연동은 엔터프라이즈 애플리케이션 성능에 많은 영향을 미친다. 따라서 제니퍼는 이에 대한 성능 분석 및 조치를 위한 DB 모니터링을 제공한다. DB 모니터링으로 수집하는 데이터는 다음과 같다.

- 애플리케이션이 수행하는 SQL 문 및 파라미터
- SQL 응답 시간 및 Fetch 건수
- DB 커넥션 개수
- SqlException

10.3.1.DB 모니터링을 위한 기본 설정

DB 사용을 모니터링을 하려면 제니퍼 에이전트의 enable_db_sql_trace 옵션을 true로 설정해야 한다. 기본 값은 true이다. 이 옵션을 false로 설정하면 DB 모니터링이 이루어지지 않는다.

```
enable_db_sql_trace = true
```

10.3.2.DB 커넥션 개수 모니터링

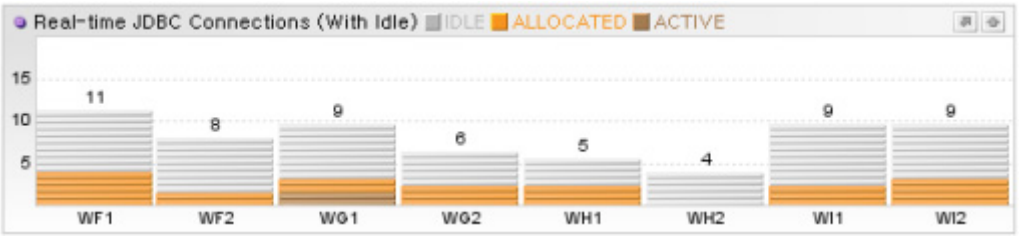
추적 가능한 DB 커넥션 상태는 다음과 같다.

표 10-2: DB 커넥션 유형

유형	설명
대기 중인 DB 커넥션 개수(IDLE)	DB 커넥션 풀에서 대기 중인 DB 커넥션 개수
할당된 DB 커넥션 개수(ALLOCATED)	서비스 쓰레드가 DB 커넥션 풀로부터 할당받은 DB 커넥션 개수
사용 중인 DB 커넥션 개수(ACTIVE)	(닷넷 버전에서는 지원하지 않음)

DB 커넥션 개수는 제니퍼 에이전트가 수집하는 일반 성능 데이터로 해당 시점의 값을 이퀄라이저 차트와 런타임 라인 차트를 통해서 확인할 수 있다.

그림 10-5: 실시간 JDBC 커넥션 개수



또한 DB 커넥션 개수에 대한 5분 평균 값은 PERF_X_01~31 테이블의 JDBC_IDLE, JDBC_ALLOC, JDBC_ACTIVE 칼럼에 저장된다.

10.3.3.SQL 데이터 수집 방법

DB 모니터링은 다른 자원을 모니터링 하는 것에 비해 상당히 많은 데이터를 수집해야 한다. 애플리케이션이 수행하는 모든 SQL과 파라미터를 수집해야지만 문제가 있는 SQL에 대해서 튜닝 등의 작업을 할 수 있기 때문이다.

그래서 데이터 전송량을 줄이기 위해서 제니퍼 에이전트가 제니퍼 서버에 보내는 데이터에서는 SQL이 아닌 해당 SQL에 대한 해시 값을 사용한다. 그리고 별도의 방법을 통해서 해시 값에 대한 SQL을 수집해서 제니퍼 서버의 SQLS 테이블에 저장한다.

그리고 SQLS 테이블에 저장되는 SQL의 개수를 줄이기 위해서, SQL에 있는 상수를 파라미터로 처리하여 저장한다. 예를 들어, 자바 애플리케이션이 처리한 다음 SQL은 별도의 SQL이다.

```
SELECT * FROM TABLE WHERE ID = ? AND AGE = 32 AND NAME =  'KIM'
SELECT * FROM TABLE WHERE ID = ? AND AGE = 27 AND NAME =  'LEE'
```

그러나 앞의 코드는 상수를 제외한 기본적인 구조는 동일하다. 이를 그대로 SQLS 테이블에 저장하면 데이터의 양이 많아진다. 예를 들어, 다음 코드가 수행된다면 10,000개의 SQL이 SQLS 테이블에 저장되어야 한다.

```
IDbCommand command = ...;
IDataReader dr = null;
for (int i = 0; i < 10000; i++)
{
  command.CommandText = "SELECT * FROM USERS WHERE ID = " + I;
  dr = command.ExceuteReader();
...
}
```

이를 해결하기 위해서 상수 중에서 문자는 [$]로, 숫자는 [#]으로 변경하여 SQLS 테이블에 저장한다.

따라서 SQLS 테이블에는 다음 SQL 만이 저장된다.

```
SELECT * FROM TABLE WHERE ID = ? AND AGE = # AND NAME =  '$'
```

그리고 X-View 프로파일 데이터 등에서는 상수 파라미터는 파라미터 1로 나타나고, 바인딩 파라미터는 파라미터 2로 표시된다. 바인딩 파라미터는 java.sql.PreparedStatement 객체로 수행하는 SQL에서 [?]로 표시되는 파라미터를 의미한다.

10.3.4.SQL 예외 로그 기록

애플리케이션에서 SqlException이 발생하면, 관련 내용을 제니퍼 에이전트 로그 파일에 기록할 수 있다. 이 내용을 기록하려면 제니퍼 에이전트의 enable_sql_error_trace 옵션을 true로 설정한다. 기본 값은 false 이다.

```
enable_sql_error_trace = true
```

단, 정상적인 상황에서도 비즈니스 로직에 종속적으로 SqlException이 발생할 수 있다. 따라서 SqlException을 오류 혹은 장애로 단정지을 수는 없다. 또한 반복적인 SqlException의 발생으로 로그를 빈번하게 기록하면 성능 저하가 발생할 수 있으므로, 필요 시에만 이 옵션을 true로 지정하고 평상시에는 false로 지정하는 것을 권장한다.

10.3.5.DB 모니터링과 예외 발생

DB 모니터링과 관련해서 발생하는 예외는 다음과 같다.

10.3.5.1. SQL 응답 속도의 지연

SQL 응답 속도가 임계치를 초과하면 WARNING_DB_BAD_RESPONSE 예외가 발생한다. 임계치는 제니퍼 에이전트의 sql_bad_responsetime 옵션으로 설정한다. 기본 값은 20000이고 단위는 밀리 세컨드이다.

```
sql_bad_responsetime = 20000
```

10.3.5.2. Fetch 건수 초과

Fetch는 java.sql.ResultSet 객체의 next 메소드를 호출하는 것을 의미한다. 따라서 Fetch 건수는 next 메소드의 호출 건수를 의미한다. 트랜잭션이 수행되는 과정에서 동일한 java.sql.ResultSet 객체에 대한 Fetch 건수가 임계치를 초과하면 WARNING_JDBC_TOOMANY_RS_NEXT 예외가 발생한다. 임계치는 제니퍼 에이전트의 jdbc_resultset_warning_fetch_count 옵션으로 설정한다. 기본 값은 10000 이다.

```
db_reader_warning_fetch_count = 10000
```

10.3.5.3. DB 리소스 미반환 추적

SqlConnection 객체 등을 DB 자원이라고 한다. 이 객체들을 사용한 후에 해당 객체의 Close (또는 Dispose) 메서드를 호출하지 않으면 DB 자원 미반환 예외가 발생한다.

그런데 문제가 되는 것은 DB 자원 미반환을 감지하는 시점이다. 웹 응용 프로그램의 경우 ProcessRequest 처리 완료 후에 해당 요청에서 사용된 DB 자원에 대한 미반환을 체크하고 일반적인 응용 프로그램의 경우에는 메서드 단위의 실행마다 미반환을 체크한다.

DB 미반환과 관련한 예외는 다음과 같다.

- **WARNING_DB_CONN_UNCLOSED**

애플리케이션에서 IDbConnection 객체를 사용한 후에 Close (또는 Dispose) 메서드를 호출하지 않으면 WARNING_DB_CONN_UNCLOSED 경고가 발생한다.

11

경보와 예외 모니터링

제니퍼는 자바 애플리케이션에서 발생하는 예외(Exception)를 감지하고 자바 애플리케이션 상태를 분석하여 적절한 경보(Alert)를 발령한다. 경보는 심각(Critical), 에러(Error), 경고(Warning) 등으로 구분되는데, 사용자는 제니퍼 사용자 인터페이스를 통해서 실시간으로 경보 발령을 확인하고 과거 경보 내역을 분석할 수 있다.

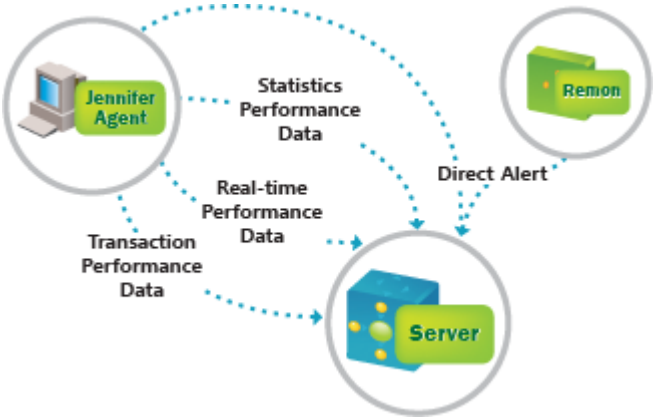
11.1. 경보와 예외의 이해

예외는 제니퍼 에이전트를 설치한 자바 애플리케이션이 트랜잭션을 처리하는 과정에서 java.lang.Throwable이 발생하거나, 응답 시간이 지연되는 것과 같은 특정 상태가 될 때 발생된다. 반면 경보는 제니퍼 서버가 제니퍼 에이전트로부터 수집한 다양한 성능 데이터를 분석하여 도출해내는 것이다.

Notice: 제니퍼 서버가 발령하는 경보는 자바 애플리케이션에서 발생한 예외 및 다양한 성능 데이터를 근거로 한다.

제니퍼에서 예외와 경보를 이해하기 위해서는 어떠한 방식으로 이들이 발생하거나 생성되는지를 이해해야 한다.

그림 11-1: 경보와 예외



11.1.1. 애플리케이션 예외와 경보

모든 경보는 제니퍼 에이전트가 아닌 제니퍼 서버가 발령한다. 즉, 예외를 포함해 제니퍼 에이전트로부터 수집한 성능 데이터를 기반으로 제니퍼 서버가 경보를 발령한다.

제니퍼 에이전트는 매 트랜잭션마다 트랜잭션 성능 데이터를 제니퍼 서버에 전송하는데 트랜잭션 수행 도중에 예외가 발생하면, 트랜잭션 성능 데이터에 예외 코드를 포함시켜 전송한다.

트랜잭션 성능 데이터는 전송 빈도가 높기 때문에 작은 양의 데이터만을 제니퍼 서버에 전송해야 한다. 따라서 제니퍼 에이전트는 하나의 트랜잭션에서 발생한 예외 숫자와 상관없이, 하나의 예외 코드만을 트랜잭션 성능 데이터에 포함시켜서 제니퍼 서버에 전송한다. 즉, 하나의 트랜잭션에서 두 개 이상의 예외가 발생해도, 제니퍼 서버는 제니퍼 에이전트로부터 전송받은 하나의 예외만을 경보로 발령한다.

그러나 제니퍼 서버가 10분마다 수집하는 예외 현황 통계 데이터에는 하나의 트랜잭션에서 발생한 모든 예외가 포함된다.

예를 들어, 트랜잭션을 처리하는 과정에서 DB 연결에 실패하면 제니퍼 에이전트는 이를 ERROR_JDBC_CONNECTION_FAIL 예외로 감지한다. 그런데 애플리케이션 코드에서 DB 연결 실패에 대한 예외 처리를 하지 않았으면 서비스 시작점에서 또 다시 예외가 발생하고, 제니퍼 에이전트는 이를 ERROR_UNCAUGHT_EXCEPTION 예외로 감지한다. 그러면 제니퍼 에이전트는 트랜잭션 성능 데이터에 ERROR_UNCAUGHT_EXCEPTION 예외 코드만을 포함하여 제니퍼 서버에 전송한다. 그러나 10분마다 제니퍼 서버가 수집하는 예외 현황 통계 데이터에는 ERROR_JDBC_CONNECTION_FAIL 예외와 ERROR_UNCAUGHT_EXCEPTION 예외가 모두 포함된다.

즉, 제니퍼 서버는 해당 트랜잭션에서 ERROR_UNCAUGHT_EXCEPTION 예외가 발생했다고 인지하고, ERROR_UNCAUGHT_EXCEPTION 경보를 발령한다. 그리고

ERROR_JDBC_CONNECTION_FAIL 예외와 ERROR_UNCAUGHT_EXCEPTION 예외는 **[실시간 모니터링 | 애플리케이션]** 메뉴에서 확인할 수 있다.

Notice: 예외 유형과 경보 유형은 동일한 코드를 사용한다.

11.1.2. 트랜잭션에서 복수의 예외가 발생하는 경우와 경보 발령

앞에서 설명한 것처럼 하나의 트랜잭션에서 여러 개의 예외가 발생하면 그 중 하나의 예외만 제니퍼 서버로 전송되고, 제니퍼 서버는 이를 경보로 발령한다. 여러 개의 예외 중에서 제니퍼 서버로 전송할 예외를 결정하는 기준은 다음과 같다.

우선 먼저 발생한 예외가 제니퍼 서버로 전달된다. 예를 들어, 외부 트랜잭션 처리 과정에서 응답이 지연되어 WARNING_TX_BAD_RESPONSE 예외가 발생한 후에, SQL 처리 과정에서 응답이 지연되어 WARNING_JDBC_BAD_RESPONSE 예외가 발생하면 WARNING_TX_BAD_RESPONSE 예외가 제니퍼 서버에 전송된다.

단, 트랜잭션이 끝나는 시점에 다음과 같은 예외가 발생하면 다른 예외에 우선하여 해당 예외를 제니퍼 서버에 전달한다.

- ERROR_RECURRSIVE_CALL
- ERROR_UNCAUGHT_EXCEPTION
- ERROR_HTTP_IO_EXCEPTION
- ERROR_OUTOFMEMORY
- ERROR_UNKNOWN_ERROR
- ERROR_PLC_REJECTED

11.1.3. 실시간 성능 데이터와 경보

제니퍼 에이전트는 1초마다 응답 시간, 업무 처리량, 동시단말 사용자 수 등과 같은 성능 데이터를 제니퍼 서버에 전송한다. 그러면 제니퍼 서버는 이 정보를 기반으로 다양한 경보를 발령한다. 예를 들어, CPU 사용률이 높으면 제니퍼 서버는 WARNING_SYSTEM_CPU_HIGH 경보를 발령한다.

11.1.4. 사용자 정의 경보

제니퍼 서버가 아닌 제니퍼 에이전트나 REMON 등에서 직접 경보를 발령하여 제니퍼 서버에 전송할 수도 있다. 이렇게 발령하는 경보 유형은 주로 USER_DEFINED_FATAL,

USER_DEFINED_ERROR, USER_DEFINED_WARNING, 그리고 USER_DEFINED_MESSAGE 등이다.

Notice: ExtraAgent와 같은 제니퍼 에이전트 어뎁터 혹은 REMON의 AlertFilter 필터를 이용해서 사용자 정의 경보를 발령할 수 있다. 자세한 사항은 [ExtraAgent 어뎁터]와 [ALERT]를 참조한다.

ExtraAgent에서 1초 동안 버퍼링 할 수 있는 최대 경고 개수를 제니퍼 에이전트의 max_num_of_direct_alert 옵션으로 설정할 수 있다.기본 값은 10이다. 1초 동안 설정된 값 이상의 사용자 정의 경보가 제니퍼 에이전트에서 발생하면 마지막 10개만 제니퍼 서버로 전달된다.

```
max_num_of_direct_alert = 10
```

11.1.5. 경보 제어

특정 경보가 발령되지 않도록 제어할 수 있다. 즉 애플리케이션에서 특정 예외가 발생해도, 제니퍼 서버가 이를 경보로 발령하지 않도록 제어할 수 있다.

이는 발령된 경보 데이터를 다른 애플리케이션과 연동하는 것과는 다른 개념이다.

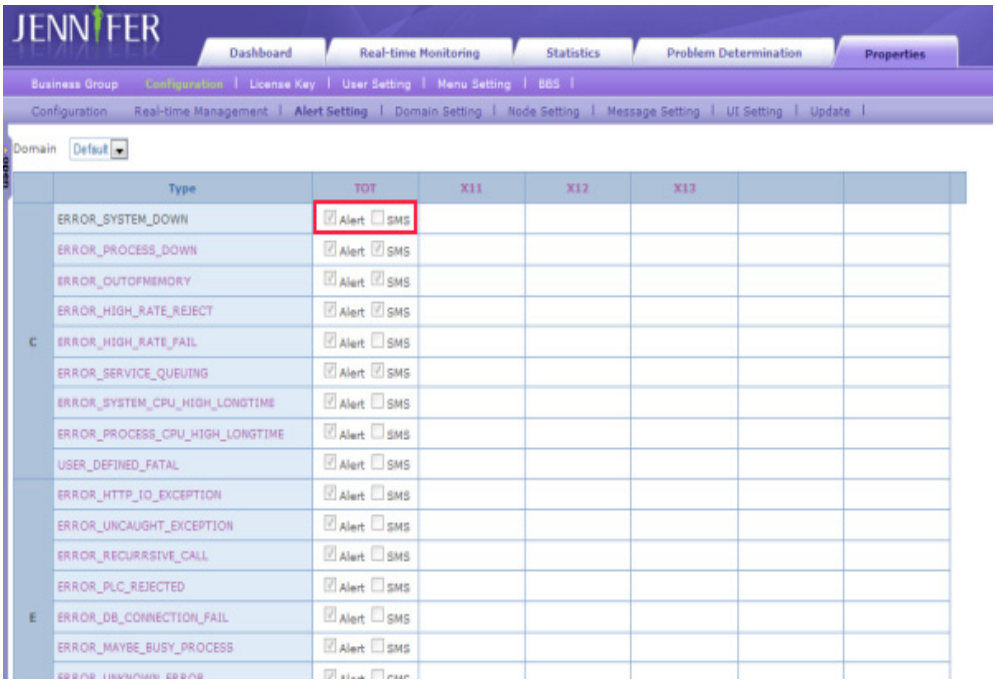
이를 위해서는 경보 발령과 다른 애플리케이션과의 경보 데이터 연동을 구분해야 한다. 경보 데이터 연동과 관련한 자세한 사항은 [경보 데이터 연동(343 페이지)]을 참조한다.

그림 11-2: 경보 발령 흐름

- 경보 발령 - 제니퍼 서버가 Alert 모듈을 통해서 경보를 발령한다.
- 경보 데이터 연동 - 제니퍼 서버가 SMS 모듈을 통해서 발령한 경보 데이터를 다른 애플리케이션에 전송한다.

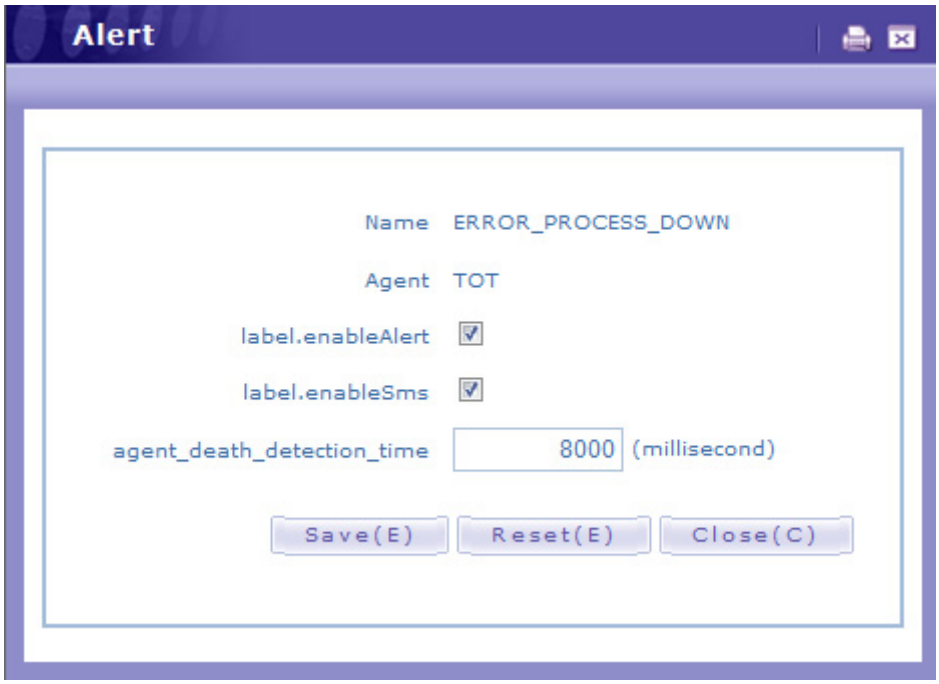
경보 발령이나 경보 데이터 연동에 대한 제어는 [Congigure | Alert]메뉴에서 설정한다.

그림 11-3: 경보 데이터 연동 제어 선택



각 셀을 클릭하면 각 에이전트별로 경보를 제어할 수 있다. 또한 주요 임계값들도 설정된다.

그림 11-4: 경보 데이타 연동 제어 설정확인



11.2. 경보와 예외 유형

제니퍼는 경보 유형을 심각(Critical), 에러(Error), 경고(Warning) 등 3 가지로 분류한다.

11.2.1. 심각(Critical)

심각(Critical)으로 분류되는 경보 유형은 다음과 같다.

- **ERROR_SYSTEM_DOWN**

제니퍼 서버는 CPU 사용률을 모니터링하는 WMOND 프로세스가 정지하면 이 경보를 발령한다.

Notice: 이 경보는 제니퍼 에이전트와는 관련이 없다.

- **ERROR_PROCESS_DOWN**

제니퍼 서버는 애플리케이션이 다운되면 이 경보를 발령한다. 제니퍼 서버는 임계치(기본 값은 8초)를 초과한 후에도 제니퍼 에이전트로부터 데이터를 받지 못하면, 해당 제니퍼 에이전트의 기동 여부를 체크하기 위해서 TCP 연결을 시도한다. 이 TCP 연결 시도가 실패하면 제니퍼 서버는 해당 제니퍼 에이전트를 설치한 애플리케이션이 정지된 것으로 간주한다.

Notice: 애플리케이션이 정지되지 않았음에도 해당 애플리케이션이 장시간의 가비지 콜렉션을 수행하는 경우에는, 제니퍼 에이전트가 제니퍼 서버로 데이터를 보내지 못하고 제니퍼 서버의 TCP 연결 시도 역시 실패할 수 있다. 따라서 이 경우에도 ERROR_PROCESS_DOWN 경보가 발령될 수 있다.

임계치는 제니퍼 서버의 agent_death_detection_time 옵션을 통해서 설정한다. 단위는 밀리 세컨드이다.

```
agent_death_detection_time = 8000
```

이 경보와 상관없이 자바 애플리케이션이 정상적인 Shutdown 명령이 아닌 java.lang.System 클래스의 exit 메소드 호출과 같은 방법으로 종료되면 제니퍼 에이전트 로그 파일에 관련 내용이 기록된다. 로그 메시지는 다음과 같다.

```
20081024/150638:[B012] System.exit() detected.
20081024/150638:[B013] But it would be abnormal System.exit() on
Servlet/JSP.
   at java.lang.Runtime.exit(Runtime.java:90)
   at java.lang.System.exit(System.java:906)
   at org.apache.jsp.exit_jsp._jspService(exit_jsp.java:45)
   at org.apache.jasper.runtime.HttpJspBase( ...
```

[B013] 코드는 서블릿이나 JSP가 수행되는 과정에서 java.lang.System 클래스의 exit 메소드가 호출되었다는 경고 번호이다. 스택트레이스를 통해서 애플리케이션 이름을 확인할 수 있다.

• ERROR_OUTOFMEMORY

제니퍼 서버는 자바 애플리케이션에서는 java.lang.OutOfMemoryError, 닷넷 애플리케이션에서는 System.OutOfMemoryException 예외가 발생하면 이 경보를 발령한다. .

제니퍼 서버는 애플리케이션의 5 초 동안의 평균 서비스 처리률(Service Rate) 대비 PLC에 의한 부하량 제어률(Service Reject Rate)의 비율이 임계치를 초과하는 경우에 이 경보를 발령한다. 제니퍼 서버의 high_rate_reject_alert_limit 옵션을 통해서 임계치를 설정한다. 기본 값은 50이고 단위는 퍼센트이다.

```
high_rate_reject_alert_limit = 50
```

- **ERROR_HIGH_RATE_FAIL**

제니퍼 서버는 애플리케이션의 5 초 동안의 평균 서비스 처리율(Service Rate) 대비 실패율(Failed Rate)의 비율이 임계치를 넘은 경우에 이 경보를 발령한다. 제니퍼 서버의 high_rate_failed_alert_limit 옵션을 통해서 임계치를 설정한다. 기본 값은 50이고 단위는 퍼센트이다.

```
high_rate_failed_alert_limit = 50
```

- **ERROR_SERVICE_QUEUING**

제니퍼 서버는 애플리케이션의 5 초 동안의 평균 액티브 서비스 개수가 임계치를 초과하면 이 경보를 발령한다. 임계치는 제니퍼 서버의 active_service_alert_limit 옵션을 통해서 설정한다. 기본 값은 70이다.

```
active_service_alert_limit = 70
```

- **ERROR_SYSTEM_CPU_HIGH_LONGTIME**

제니퍼 서버는 30 초 동안의 평균 시스템 CPU 사용률이 임계치를 초과하면 이 경보를 발령한다. 제니퍼 서버의 sys_cpu_alert_limit 옵션을 통해서 임계치를 설정한다. 기본 값은 95이고 단위는 퍼센트이다.

```
sys_cpu_alert_limit = 95
```

- **ERROR_PROCESS_CPU_HIGH_LONGTIME**

제니퍼 서버는 30 초 동안의 평균 프로세스 CPU 사용률이 임계치를 초과하면 이 경보를 발령한다. 기본 값은 90이고 단위는 퍼센트이다.

```
process_cpu_alert_limit = 90
```

- **USER_DEFINED_FATAL**

사용자가 REMON을 통해서 발령할 수 있는 심각 유형의 사용자 정의 경보이다.

11.2.2. 에러(Error)

에러(Error)로 분류되는 경보 유형은 다음과 같다.

- **ERROR_HTTP_IO_EXCEPTION**

제니퍼 서버는 자바 애플리케이션에서 java.io.IOException, 닷넷 애플리케이션에서 System.IO.IOException 타입에 해당하는 예외가 발생하면 이 경보를 발령한다. 예를 들어, 사용자가 서비스를 요청하고 결과를 받기 전에 웹 브라우저를 닫는 경우가 이에 해당한다.

• ERROR_UNCAUGHT_EXCEPTION

제니퍼 서버는 자바 애플리케이션에서 다른 경보 유형에 해당하지 않는 java.lang.Exception, 닷넷 애플리케이션에서 System.Exception 타입의 예외가 발생하면 이 경보를 발령한다.

Notice: 애플리케이션 시작점까지 전달된 예외만 이 경보에 해당한다. 만약 애플리케이션 처리 과정에서 발생한 예외를 catch 문을 통해서 처리했다면 이 경보가 발령되지 않는다.

• ERROR_RECURRSIVE_CALL

제니퍼 서버는 애플리케이션에서 임계치를 초과하여 다른 서블릿/JSP/ASPX를 계속 호출하면 이 경보를 발령한다. 제니퍼 에이전트의 recurrsive_call_max_count 옵션을 통해서 임계치를 설정한다.

[에이전트옵션] 기본 값은 50000이다.

```
recurrsive_call_max_count = 50000
```

서블릿 혹은 JSP가 다른 서블릿 혹은 JSP를 호출한다는 것은 자바 코드 상에서 javax.servelt.RequestDispatcher 객체의 forward 혹은 include 메소드를 호출하는 것을 의미한다.

이 경보의 원인을 파악하기 위해서 제니퍼 에이전트 로그 파일에 관련 스택트레이스를 기록하려면, 제니퍼 에이전트의 recurrsive_call_trace 옵션을 true로 지정한다.

[에이전트옵션] 기본 값은 false이다.

```
recurrsive_call_trace = true
```

기본적으로 제니퍼 에이전트 로그 파일에 기록되는 스택트레이스 글자수를 10,000자로 제한한다. 제니퍼 에이전트의 recurrsive_call_trace_size 옵션을 통해서 글자 수를 설정할 수 있다.

[에이전트옵션]

```
recurrsive_call_trace_size = 10000
```

• ERROR_PLC_REJECTED

제니퍼 서버는 자바 애플리케이션에서 PLC에 의해서 요청이 거절되면 이 경보를 발령한다.

• ERROR_DB_CONNECTION_FAIL

제니퍼 서버는 애플리케이션에서 DB 연결 객체를 얻지 못하면(데이터베이스 연결에 실패하는 경우) 이 경보를 발령한다.

• ERROR_MAYBE_BUSY_PROCESS

제니퍼 서버는 임계치(기본 값은 8초)를 초과한 후에도 제니퍼 에이전트로부터 데이터를 받지 못하면, 해당 제니퍼 에이전트의 기동 여부를 체크하기 위해 TCP 연결을 시도한다. TCP 연결 시도가 성공하면 제니퍼 에이전트를 설치한 애플리케이션에서 데이터를 받지 못한 시간 동안 가비지 콜렉션이 수행된 것으로 간주한다.

Notice: 제니퍼 에이전트가 제니퍼 서버로 데이터를 전송하지 못하는데는 가비지 콜렉션 수행 이외의 다른 이유가 있을 수 있다. 따라서 이 경보를 100% 가비지 콜렉션에 의한 지연 현상으로 받아들여서는 안된다.

제니퍼 서버의 agent_death_detection_time 옵션을 통해서 임계치를 설정한다. 기본 값은 8000이고 단위는 밀리 세컨드이다.

```
agent_death_detection_time = 8000
```

• ERROR_UNKNOWN_ERROR

제니퍼 서버는 애플리케이션에서 다른 경보 유형에 해당하지 않는 java.lang.Error 타입의 예외가 발생하면 이 경보를 발령한다. 애플리케이션 시작점까지 전달된 예외만이 이 경보에 해당한다. 만약 애플리케이션 처리 과정에서 발생한 예외를 catch 문을 통해서 처리했다면 이 경보가 발령되지 않는다.

• USER_DEFINED_ERROR

사용자가 REMON을 통해서 발령할 수 있는 에러 유형의 사용자 정의 경보이다.

• ERROR_LOGICAL_PROCESS

제니퍼 서버는 ALSB(AquaLogic Service Bus) 혹은 WLI(WebLogic Integration) 등에서 논리적 에러가 발생한 경우에 이 경보를 발령한다.

11.2.3. 경고(Warning)

경고(Warning)로 분류되는 경보 유형은 다음과 같다.

• WARNING_DB_CONN_UNCLOSED

제니퍼 서버는 애플리케이션에서 DB연결 객체를 사용한 후에 close 메소드를 호출하지 않으면 이 경보를 발령한다.

• WARNING_JDBC_STMT_UNCLOSED

제니퍼 서버는 자바 애플리케이션에서 java.sql.Statement 객체를 사용한 후에 close 메소드를 호출하지 않으면 이 경보를 발령한다..

• WARNING_JDBC_PSTMT_UNCLOSED

제니퍼 서버는 자바 애플리케이션에서 java.sql.PreparedStatement 객체를 사용한 후에 close 메소드를 호출하지 않으면 이 경보를 발령한다.

• WARNING_JDBC_CSTMT_UNCLOSED

제니퍼 서버는 자바 애플리케이션에서 java.sql.CallableStatement 객체를 사용한 후에 close 메소드를 호출하지 않으면 이 경보가 발령한다.

• WARNING_JDBC_RS_UNCLOSED

제니퍼 서버는 자바 애플리케이션에서 java.sql.ResultSet 객체를 사용한 후에 close 메소드를 호출하지 않으면 이 경보를 발령한다..

Notice: DB 연결 자원 미반환 경보 java.sql.Connection, java.sql.Statement, java.sql.ResultSet 객체 등을 DB 연결 자원이라고 한다. 제니퍼 서버는 이 객체들을 사용한 후에 해당 객체의 close 메소드를 호출하지 않으면 DB 연결 자원 미반환 경보를 발령한다. 자바 환경에서는 자원 미반환 경보가 실제보다 지연되어 발령된다.

• WARNING_DB_TOOMANY_FETCH

제니퍼 서버는 자바 애플리케이션에서 동일한 java.sql.ResultSet 객체에 대해서 임계치를 초과하여 패치(Fatch)를 하면 이 경보를 발령한다.

Notice: 자바 환경에서 페치는 java.sql.ResultSet 객체의 next 메소드를 호출하는 것을 의미하고 닷넷 환경에서 페치는 System.Data.IDataReader 객체의 Read 메서드를 호출하는 것을 의미한다.

[자바] 제니퍼 에이전트의 jdbc_resultset_warning_fetch_count 옵션을 통해서 임계치를 설정한다. 기본 값은 5000이다. .

```
jdbc_resultset_warning_fetch_count = 5000
```

• WARNING_JDBC_STMT_EXCEPTION

제니퍼 서버는 자바 애플리케이션에서 java.sql.Statement 객체의 execute, executeBatch, executeQuery, executeUpdate 등의 메소드를 통해서 쿼리를 수행하다가 java.sql.SQLException 예외가 발생하면 이 경보를 발령한다.

• WARNING_JDBC_PSTMT_EXCEPTION

제니퍼 서버는 자바 애플리케이션에서 java.sql.PreparedStatement 객체의 execute, executeBatch, executeQuery, executeUpdate 등의 메소드를 통해 쿼리를 수행하다가 java.sql.SQLException 예외가 발생하면 이 경보를 발령한다.

• WARNING_DB_BAD_RESPONSE

제니퍼 서버는 애플리케이션에서 SQL 수행 시간이 임계치를 초과하면 이 경보를 발령한다. 제니퍼 에이전트의 sql_bad_responsetime 옵션을 통해서 임계치를 설정한다. 기본 값은 20000이고 단위는 밀리 세컨드이다..

```
sql_bad_responsetime = 20000
```

• WARNING_TX_CALL_EXCEPTION

제니퍼 서버는 자바 애플리케이션에서 외부 트랜잭션을 수행하다가 예외가 발생하면 이 경보를 발령한다.

• WARNING_TX_BAD_RESPONSE

제니퍼 서버는 자바 애플리케이션에서 외부 트랜잭션 수행 시간이 임계치를 초과하면 이 경보를 발령한다. 제니퍼 에이전트의 tx_bad_responsetime 옵션을 통해서 임계치를 설정한다. 기본 값은 10000이고 단위는 밀리 세컨드이다.

```
tx_bad_responsetime = 10000
```

• WARNING_APP_BAD_RESPONSE

제니퍼 서버는 애플리케이션에서 애플리케이션 트랜잭션 수행 시간이 임계치를 초과하면 이 경보를 발령한다. 제니퍼 에이전트의 app_bad_responsetime 옵션을 통해서 임계치를 설정한다. 기본 값은 10000이고 단위는 밀리 세컨드이다. .

```
app_bad_responsetime = 10000
```

• WARNING_DB_UN_COMMIT_ROLLBACK

제니퍼 서버는 자바 애플리케이션에서 JDBC 트랜잭션을 시작(false를 파라미터로 java.sql.Connection 객체의 setAutoCommit 메소드를 호출)하고 java.sql.Statement 객체의 execute, executeBatch, executeUpdate 등의 메소드를 호출한 후에, 명시적으로 JDBC 트랜잭션을 종료(java.sql.Connection 객체의 rollback 혹은 commit 메소드를 호출)하지 않으면 이 경보를 발령한다.

그런데 java.sql.Statement 객체의 exeucte 메소드를 통해 SELECT 문을 수행하는 경우에는 이 경보가 실제 상황을 올바르게 설명하지 못한다. 이 경우에는 제니퍼 에이전트의 ignore_rollback_uncommited_error 옵션을 통해서 발령되지 않도록 설정할 수 있다. .

```
ignore_rollback_uncommited_error = true
```

Notice: 장기적으로는 SELECT 문이 exeucteQuery 메소드로 처리되도록 해당 소스 코드를 수정하는 것을 권장한다. .

• WARNING_DUMMY_HTTPSESSION_CREATED

다음과 같이 JSP의 page 속성 중에 session을 명시적으로 false로 지정하지 않으면 JSP 수정 과정에서 javax.servlet.http.HttpSession 객체가 만들어진다.

```
<%@ page session="false" %>
```

이렇게 session 속성을 명시적으로 false로 지정하지 않고 코드 상에서도 javax.servlet.http.HttpSession 객체를 사용하지 않으면 제니퍼 서버는 이 경보를 발령한다. 대부분의 환경에서는 이 경보가 적합하지 않기 때문에 기본적으로 이 경보는 발령되지 않는다. 이 경보를 발령시키려면 제니퍼 에이전트의 enable_dummy_httpsession_trace 옵션을 true로 설정한다.기본 값은 false이다.

```
enable_dummy_httpsession_trace = true
```

• WARNING_DB_CONN_ILLEGAL_ACCESS

제니퍼 서버는 애플리케이션에서 2개 이상의 쓰레드가 동시에 동일한 DB연결 객체를 사용하면 이 경보를 발령한다.

• WARNING_ORACLE_XMLTYPE_UNCLOSED

제니퍼 서버는 애플리케이션에서 오라클 데이터베이스를 사용하면서 oracle.xdb.XMLType 객체의 close 메소드를 호출하지 않으면 이 경보를 발령한다.

• USER_DEFINED_WARNING

사용자가 REMON을 통해서 발령할 수 있는 경고 유형의 사용자 정의 경보이다.

• WARNING_CUSTOM_RUNTIME_EXCEPTION

Custom Trace 어뎁터를 이용해서 로직에서 발생한 java.lang.RuntimeException을 처리한 경우에 발령되는 경보이다.

• WARNING_CUSTOM_EXCEPTION

Custom Trace 어뎁터를 이용해서 로직에서 발생한 java.lang.Exception을 처리한 경우에 발령되는 경보이다.

• WARNING_CUSTOM_THROWABLE

Custom Trace 어뎁터를 이용해서 로직에서 발생한 java.lang.Throwable을 처리한 경우에 발령되는 경보이다.

• WARNING_RESOURCE_LEAK

제니퍼 서버는 사용자 정의 리소스 릭이 발생한 경우에 이 경보를 발령한다.

• WARNING_JVM_HEAP_MEM_HIGH

제니퍼 서버는 애플리케이션이 동작하는 시스템의 임의의 기간 동안의 평균 힙 메모리 사용률이 임계치를 초과하면 이 경보를 발령한다. 제니퍼 서버의 jvm_heap_warning_limit 옵션을 통해서 임계치를 설정한다. 기본 값은 95이고 단위는 퍼센트이다.

```
jvm_heap_warning_limit = 95
```

그리고 임의의 기간은 제니퍼 서버의 jvm_heap_check_time 옵션으로 설정한다. 기본 값은 300이고 단위는 초이다.

```
jvm_heap_check_time = 300
```

그리고 이 경보가 발령되는 간격은 제니퍼 서버의 jvm_heap_warning_interval 옵션으로 설정한다. 기본 값은 3600이고 단위는 초이다.

```
jvm_heap_warning_interval = 3600
```

• WARNING_SYSTEM_CPU_HIGH

제니퍼 서버는 자바 애플리케이션이 동작하는 시스템의 30 초 동안의 평균 시스템 CPU 사용률이 임계치를 초과하면 이 경보를 발령한다. 제니퍼 서버의 sys_cpu_warning_limit 옵션을 통해서 임계치를 설정한다. 기본 값은 80이고 단위는 퍼센트이다.

```
sys_cpu_warning_limit = 80
```

ERROR_SYSTEM_CPU_HIGH_LONGTIME 경보가 우선한다.

• WARNING_PROCESS_CPU_HIGH

제니퍼 서버는 애플리케이션의 30 초 동안의 평균 프로세스 CPU 사용률이 임계치(기본 값은 70 %)를 초과하면 이 경보를 발령한다. 제니퍼 에이전트의 process_cpu_warning_limit 옵션을 통해서 임계치를 설정한다. 기본 값은 70이고 단위는 퍼센트이다.

```
process_cpu_warning_limit = 700
```

ERROR_PROCESS_CPU_HIGH_LONGTIME 경보가 우선한다.

11.2.4. 알림 메세지

• SYSTEM_MESSAGE

제니퍼 서버가 제니퍼 에이전트의 기동과 같은 상황을 알릴 때 사용하는 경보이다.

• USER_DEFINED_MESSAGE

제니퍼 에이전트나 REMON에서 사용할 수 있는 사용자 정의 경보 유형에 해당하는 알림 메시지 경보이다.

11.3. 경보와 예외 데이터의 저장

경보와 예외 데이터는 성능 데이터베이스에 저장된다. 경보는 ALERT_01~31 테이블에 저장되고 예외는 ERRORS_10M_01~31 테이블에 저장된다. 그러나 해당 테이블들에 경보와 예외 데이터가 저장되는 방식에는 차이가 있다.

Notice: 모든 예외는 경보에 포함되기 때문에 ALERT_01~31 테이블에는 예외 정보도 저장된다.

우선 ALERT_01~31 테이블에는 모든 경보가 저장된다. 예를 들어, ERROR_SERVICE_QUEUING 유형의 경보가 1건, WARNING_JDBC_STMT_EXCEPTION 유형의 경보가 2건이 발생하면 ALERT_01~31 테이블에는 3건의 경보 데이터가 저장된다.

그러나 ERRORS_10M_01~31 테이블에는 10분 구간을 기준으로 동일 예외에 대해서 1건만이 저장되고 CNT 칼럼에는 발생 건수가 저장된다. 따라서 위의 예에서

WARNING_JDBC_STMT_EXCEPTION 예외가 동일 10분 구간에서 발생했다면 ERRORS_10M_01~31 테이블에는 1건만이 저장되고 CNT 칼럼에는 2가 발생 건수로 저장된다.

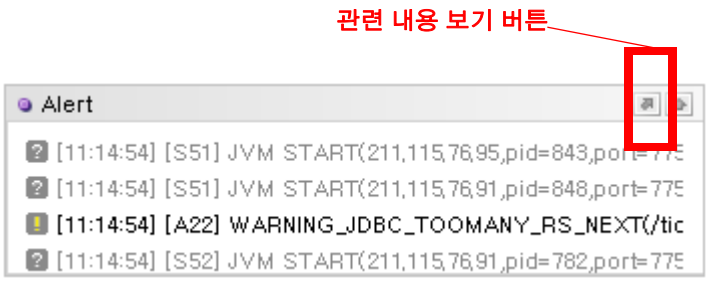
11.4. 경보와 예외 조회

제니퍼 사용자 인터페이스를 통해서 경보와 예외를 감지하고 분석하는 방법을 설명한다.

11.4.1. 경보 차트와 보드 영역

제니퍼 대시보드 하단에 있는 경보 차트에 경보 내역이 실시간으로 표시된다.

그림 11-5: 경보 차트



경보 차트에는 차트 영역에 표시할 수 있는 개수의 경보 내역만이 나타난다. 그래서 상세 내역은 경보 차트의 [관련 내용 보기] 버튼을 클릭하면 나타나는 경보 팝업 창에서 확인한다.

경보 차트에 있는 경보 내역을 지우려면 다음과 같이 한다.

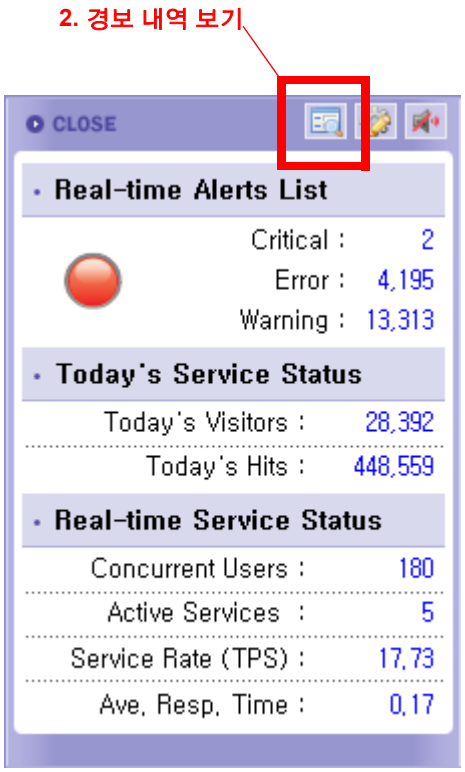
- 경보 차트안으로 마우스 커서를 이동시킨 후에 오른쪽 버튼을 클릭한다.
- 컨텍스트 메뉴에서 [지우기] 메뉴를 선택한다.

Notice: 경보 차트에서 경보 내역을 지워도 실제 경보 내역은 삭제되지 않고 성능 데이터베이스에 기록된다.

경보가 발생하면 오른쪽에 숨겨져 있던 보드 영역이 나타난다. 사용자는 보드 영역의 실시간 경보 내역을 통해서 금일 발생한 심각, 에러, 경고 유형의 발생 건수를 확인할 수 있다. 그러나 보드 영역만으로는 어떤 경보가 발생했는지를 확인할 수는 없다. 발생한 경보

의 상세 내역은 보드 영역에서 [경보 내역 보기] 버튼을 클릭하면 나타나는 경보 내역 팝업 창에서 확인한다.

그림 11-6: 보드 영역



11.4.2. 경보 내역 팝업 창

경보 내역 팝업 창은 3개의 탭으로 구성되어 있다.

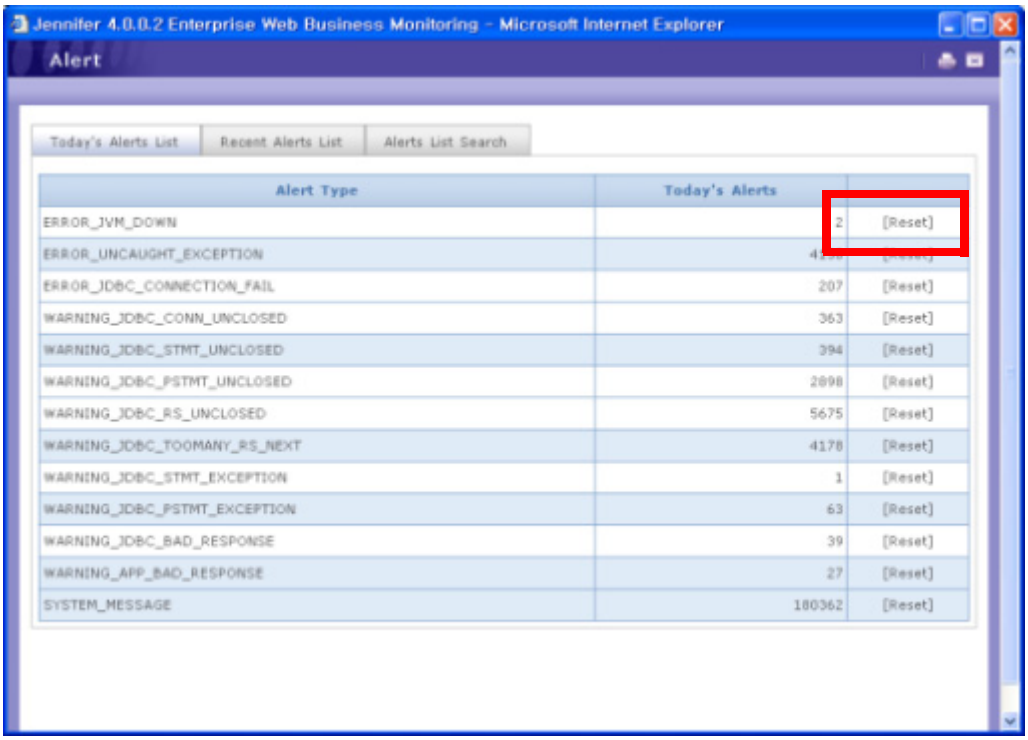
표 11-1: 경보 내역 탭

경보 내역 탭	설명
금일 경보 내역	금일 발생한 모든 경보 내역을 보여준다.
최근 경보 내역	최근 발생한 경보 내역을 보여준다. 경보가 발생한 시간과 상관없이 심각, 에러, 경고 유형별로 150개까지의 경보 내역을 보여준다.
경보 내역 조회	과거에 발생한 모든 경보 내역을 검색할 수 있다.

11.4.2.1. 금일 경보 내역

경보 유형을 기준으로 금일 발생한 경보를 보여준다. 이 탭을 통해서 특정 경보 유형이 금일 몇 건 발생했는지 확인할 수 있다.

그림 11-7: 금일 경보 내역



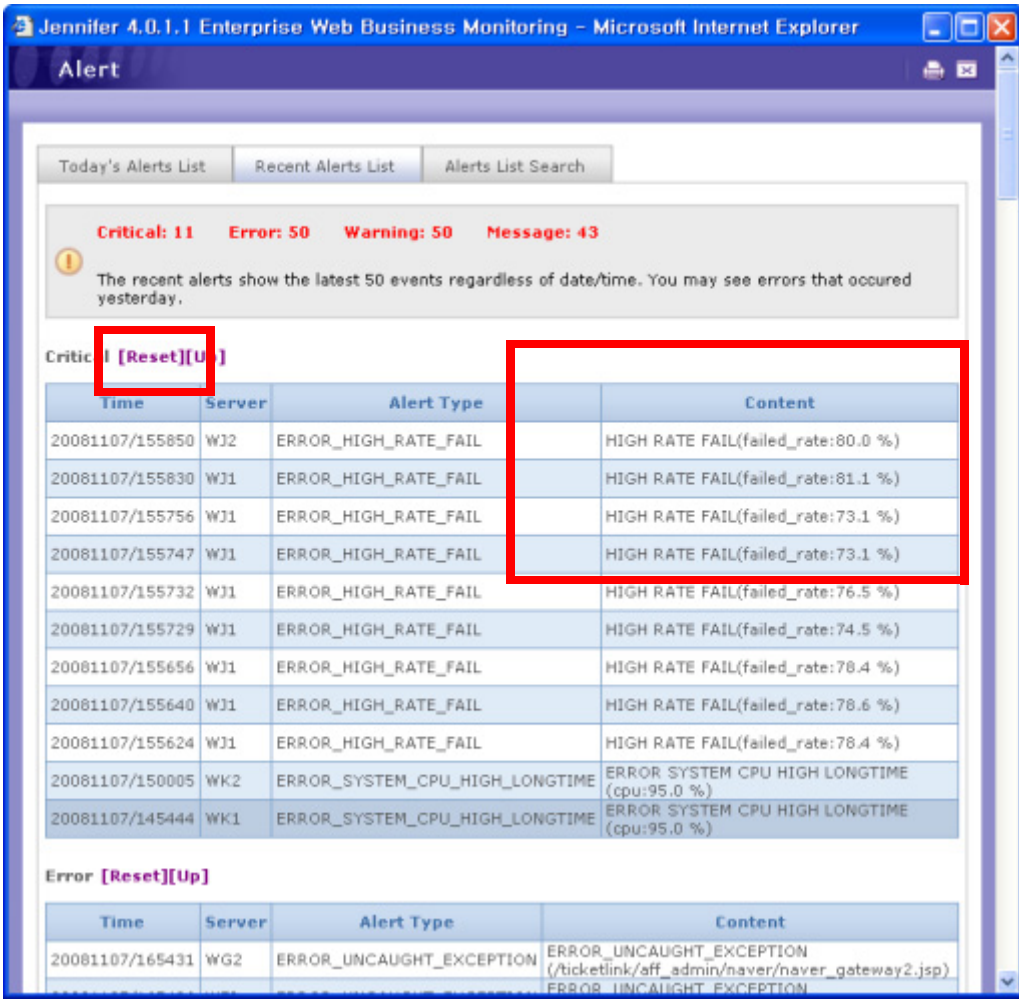
- 초기화 링크 - 특정 경보 유형의 [초기화] 링크를 클릭하면 해당 경보 유형의 금일 발생 건수가 0이 되고 해당 경보 유형이 금일 경보 내역 목록에서 제거된다.

Notice: 초기화 링크를 클릭해도 실제 경보 내역은 삭제되지 않고 성능 데이터베이스에 기록된다. 그리고 초기화 링크를 클릭하면 보드 영역의 실시간 경보 내역도 초기화된다. 따라서 이 경우에는 경보 내역 조회 탭에서 조회한 내용과 화면에 나타나는 내역 사이에 불일치가 발생한다.

11.4.2.2. 최근 경보 내역

최근 발생한 경보 내역을 심각, 에러, 경고, 메시지 유형 별로 150 건까지 보여준다. 금일 경보 내역이 특정 경보 유형의 금일 발생 건수를 보여준다면, 최근 경보 내역은 개별 경보 내역을 모두 보여준다.

그림 11-8: 최근 경보 내역



- 초기화 링크 - [초기화] 링크를 클릭하면 심각, 에러, 경고 유형 중에서 현재 보고 있는 유형의 최근 경보 내역이 제거된다.

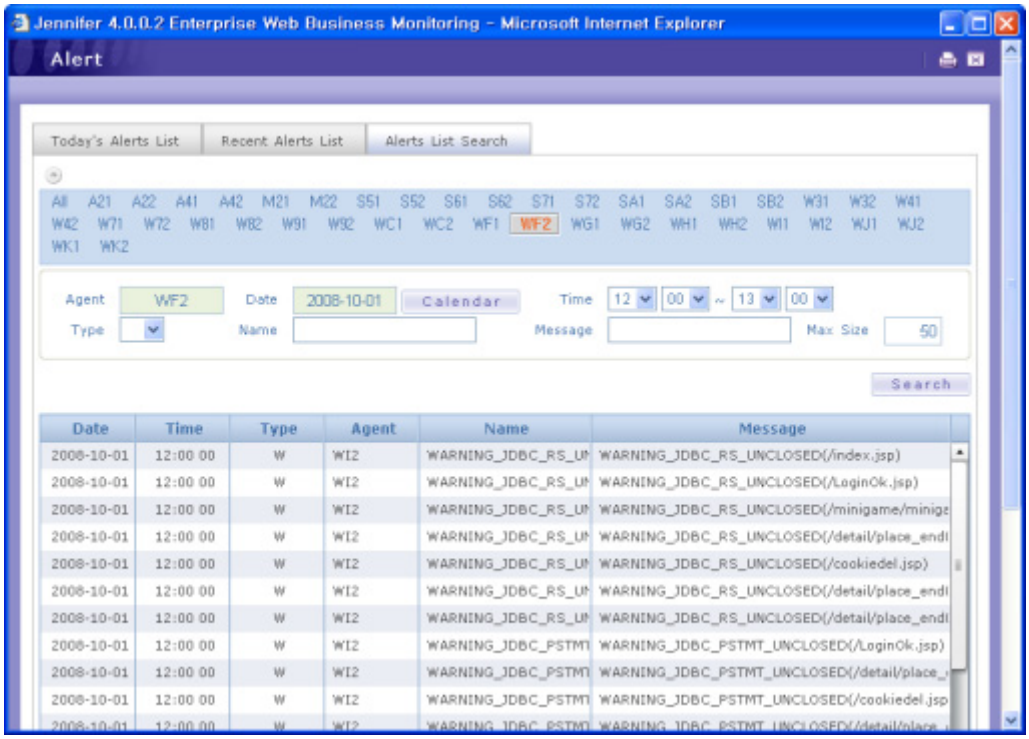
Notice: 초기화 링크를 클릭해도 실제 경보 내역은 삭제되지 않고 성능 데이터베이스에 기록된다.

- 내용 - 해당 경보가 발생한 애플리케이션 이름이나 관련 내용을 보여준다.

11.4.2.3. 경보 내역 조회

과거에 발생한 경보 내역을 조회할 수 있다. 과거는 최근 10분 구간 이전을 의미한다.

그림 11-9: 경보 내역 조회



검색 조건은 다음과 같다.

표 11-2: 경보 내역 검색 조건

검색 조건	설명
에이전트	상단 제니퍼 에이전트 목록에서 특정 제니퍼 에이전트를 선택한다. 모든 제니퍼 에이전트에 대해서 검색을 하려면 [모두]를 선택한다.
날짜	조회할 경보가 발생한 날짜를 입력한다.
시간	조회할 경보가 발생한 시간을 분단위까지 입력한다.
유형	심각, 에러, 경고 등의 경보 유형을 선택한다.
이름	조회할 경보 유형 이름을 입력한다.
메시지	조회할 경보 메시지를 입력한다.
최대 크기	검색 결과로 반환되는 경보 내역의 최대 건수를 입력한다. 기본은 50개이다.

11.4.3. 실시간 애플리케이션 목록

[실시간 모니터링 | 애플리케이션] 메뉴를 통해서 최근 10분 구간에서 발생한 예외 내역을 확인할 수 있다.

Notice: 이 목록에는 경보가 아닌 애플리케이션 트랜잭션 처리시에 발생한 예외만이 표시된다.

11.4.4. 통계 분석 애플리케이션 목록

[통계 분석 | 애플리케이션] 메뉴를 통해서 과거에 발생한 예외 내역을 확인할 수 있다. 과거는 최근 10분 구간 이전을 의미한다.

Notice: 이 목록에는 경보가 아닌 애플리케이션 트랜잭션 처리시에 발생한 예외만이 표시된다.

11.4.5. X-View 차트

예외가 발생한 애플리케이션 트랜잭션은 X-View 차트에 붉은 색 계열로 표시된다.

Notice: DB 자원 미반환과 관련한 예외는 X-View 차트에 붉은 색으로 표시되지 않는다.

11.5. 경보 데이터 연동

제니퍼 서버는 SMS 어뎁터로 경보 내용을 다른 애플리케이션에 전달할 수 있다.

11.5.1. 경보 데이터 연동을 위한 기본 설정

SMS 어뎁터가 경보 데이터 연동을 담당하는데, SMS 어뎁터 구현 클래스는 제니퍼 서버의 sms_adapter_class_name 옵션으로 설정한다.

```
sms_adapter_class_name = com.javaservice.jennifer.server.SMSExample
```

두 개 이상의 SMS 어뎁터를 사용하려면 세미콜론[;]을 구분자로 구분하여, SMS 어뎁터를 제니퍼 서버의 sms_adapter_class_name 옵션으로 설정한다.

```
sms_adapter_class_name =
com.javaservice.jennifer.server.SMSExample;com.javaservice.jennifer.se
rver.AlertToLogFile
```

새로운 SMS 어뎁터를 개발하는 방법은 경보데이터 연동을 참조한다.

그리고 짧은 시간에 제니퍼 서버가 많은 경보를 발령하면, SMS 어뎁터가 빈번하게 해당 경보를 다른 애플리케이션에게 전송한다. 이렇게 동일한 경보를 중복해서 전송하는 것을 방지하려면, 제니퍼 서버의 sms_alert_minimal_interval 옵션을 이용해서 동일한 경보 유형에 대한 SMS 전송 간격을 조정해준다. 기본 값은 1000이고 단위는 밀리 세컨드이다.

```
sms_alert_minimal_interval = 1000
```

그리고 SMS 어뎁터가 특정 경보 유형에 대해서만 다른 애플리케이션에 전송하도록 설정할 수 있다. SMS 어뎁터로 전송하기를 원하는 경보 유형을 이름으로 사용하고 그 값을 true로 해서 제니퍼 서버의 옵션으로 지정하면 된다.

예를 들어, ERROR_PLC_REJECTED 경보 유형이 발생할 때 이를 SMS 어뎁터로 다른 애플리케이션으로 전송하려면 제니퍼 서버에 다음 옵션을 추가한다.

```
ERROR_PLC_REJECTED = true
```

기본적으로 SMS 어뎁터가 다른 애플리케이션으로 전송하도록 설정된 경보 유형은 다음과 같다.

- ERROR_JVM_DOWN
- ERROR_OUTOFMEMORY
- ERROR_HIGH_RATE_REJECT
- ERROR_SERVICE_QUEUING
- USER_DEFINED_FATAL
- SYSTEM_MESSAGE
- USER_DEFINED_MESSAGE

11.5.2. 제니퍼가 제공하는 SMS 어뎁터

제니퍼가 기본적으로 제공하는 SMS 어뎁터에 대해서 설명한다. 새로운 SMS 어뎁터를 개발하는 방법은 경보데이터 연동을 참조한다.

11.5.2.1. 경보 로깅

com.javaservice.jennifer.server.AlertToLogFile SMS 어뎁터는 경보를 제니퍼 서버 로그 파일에 기록한다.

우선 제니퍼 서버의 sms_adapter_class_name 옵션으로 이 SMS 어뎁터를 설정한다.

```
sms_adapter_class_name =
com.javaservice.jennifer.server.AlertToLogFile
```

제니퍼 서버 로그 파일에 기록되는 로그 메시지 포멧은 다음과 같다

```
ALERT:SERVER_DOWN:W11:20060323/123350:000
ALERT:ERR_SYSTEM_CPU:W11:20060323/123350:000:96.6%
```

11.5.2.2. 메일 전송

com.javaservice.jennifer.server.AlertMail SMS 어뎁터는 경보를 메일로 전송한다. 우선 제니퍼 서버의 sms_adapter_class_name 옵션으로 이 SMS 어뎁터를 설정한다.

```
sms_adapter_class_name = com.javaservice.jennifer.server.AlertMail
```

그리고 default_sms로 시작하는 옵션들로 메일을 발송할 메일 서버 정보를 설정한다.

```
default_sms.mail.smtp.host = jennifersoft.com
default_sms.mail.smtp.port = 25
default_sms.emailToList = tech@jennifersoft.com,admin@jennifersoft.com
default_sms.emailFrom = tech@jennifersoft.com
```

그리고 메일 전송을 위한 자바 라이브러리 activation.jar 파일과 mail.jar 파일을 JENNIFER_HOME/server/common/lib 디렉토리에 복사한다.

Notice: AlertMail SMS 어뎁터는 메일 서버가 SMTP Relay를 허용하는 경우에만 사용할 수 있다.

11.5.2.3. SNMP TRAP

com.javaservice.jennifer.server.SnmpTrap SMS 어뎁터는 경보를 SNMP Trap을 이용하여 전송한다.

Notice: 다음 설명은 SNMP TRAP을 이해하고 있다고 가정한다.

우선 제니퍼 서버의 sms_adapter_class_name 옵션으로 이 SMS 어뎁터를 설정한다.

```
sms_adapter_class_name = com.javaservice.jennifer.server.SnmpTrap
```

그리고 snmp_trap으로 시작하는 옵션들로 SNMP TRAP 정보를 지정한다.

```
# Private Enterprise Number(PEN) for JenniferSoft is 27767.
snmp_trap_oid =  1.3.6.1.4.1.27767.1.1
snmp_trap_target_address = 127.0.0.1/162
snmp_trap_target_community = public
```

테스트를 위해서 JENNIFER_HOME/server/doc/snmp-trap/trapview.bat를 실행하면 제니퍼 서버가 전송하는 Trap 메세지를 콘솔에서 확인할 수 있다. 관련 메시지는 다음과 같다.

```
TRAP[requestID=506943247, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1.
27767.1.1 = 10:46:19,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
TRAP[requestID=942904425, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1.
27767.1.1 = 10:46:21,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
TRAP[requestID=270685152, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1.
27767.1.1 = 10:46:27,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
TRAP[requestID=1772678772, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1
.27767.1.1 = 10:46:27,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
TRAP[requestID=1562294832, errorStatus=Success(0), errorIndex=0,
VBS[1.3.6.1.4.1
.27767.1.1 = 10:46:27,E,UNCAUGHT,T11,UNCAUGHT EXCEPTION]]
```

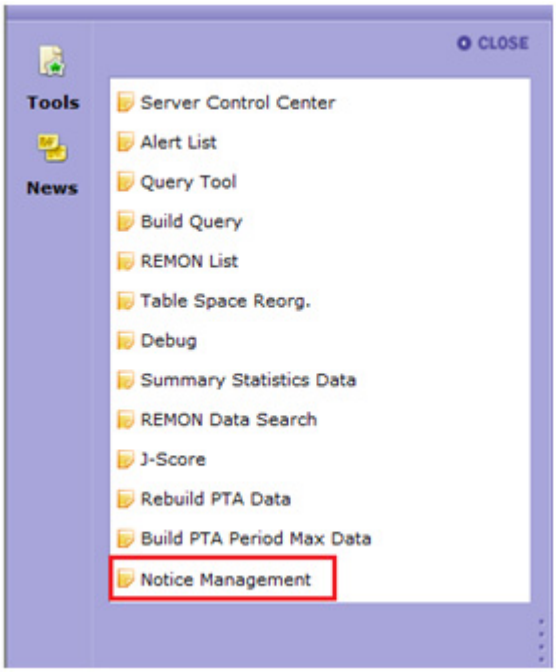
11.5.2.4. 단문 서비스

경보를 단문 서비스로 전송하려면 외부의 SMS 서비스를 사용해야 한다. 제니퍼 라이센스에 외부 SMS 서비스에 대한 사용 권한이 포함되어 있지는 않다. 따라서 경보를 단문 서비스로 전송하려면 외부 SMS 서비스 제공 업체와 사용 계약을 체결하고, 해당 SMS 서비스 업체가 제공하는 API를 기반으로 SMS 어뎁터를 개발해야 한다.

11.6. 공지기능

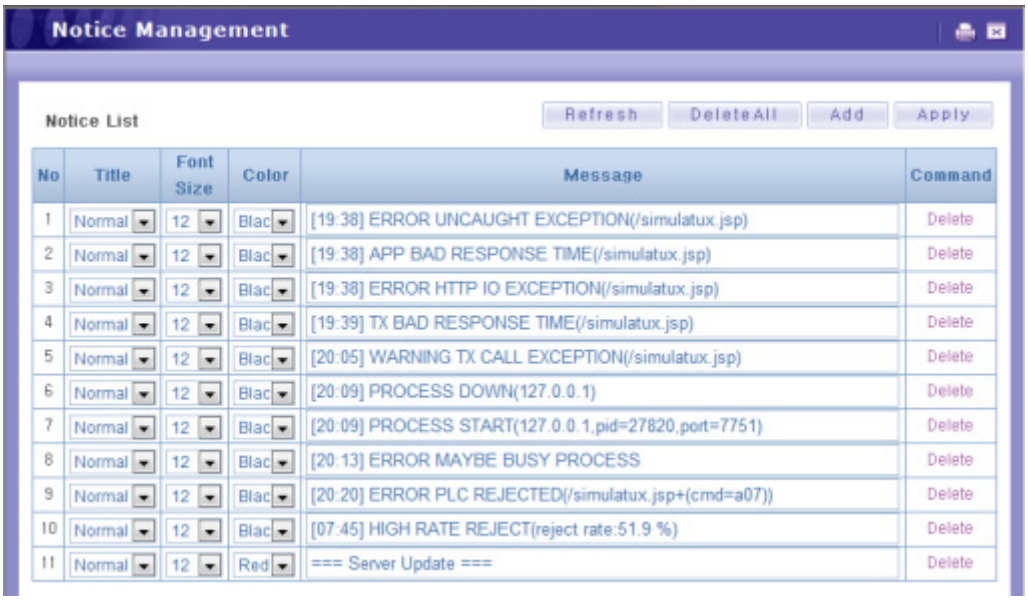
공지 기능은 제니퍼 4.5에 새로 추가된 기능이다. 시스템 관리자는 제니퍼의 “공지” 기능을 통해서 주요 정보를 제니퍼 화면을 보고 있는 사람들에게 공지 할수 있다.

그림 11-10:공지기능



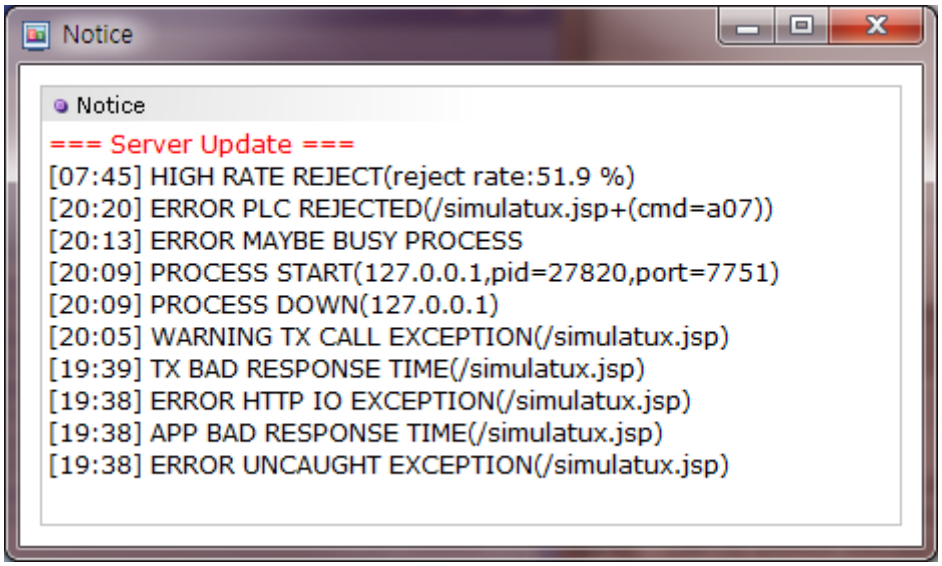
메세지를 추가하기 위해서는 간단하게 [Add]버튼을 클릭하고 추가한다 혹은 기존 메세지를 수정할 수 있다.

그림 11-11:공지기능 관리



수정된 내용은 [Apply] 버튼을 클릭함으로써 모두에게 전파된다.

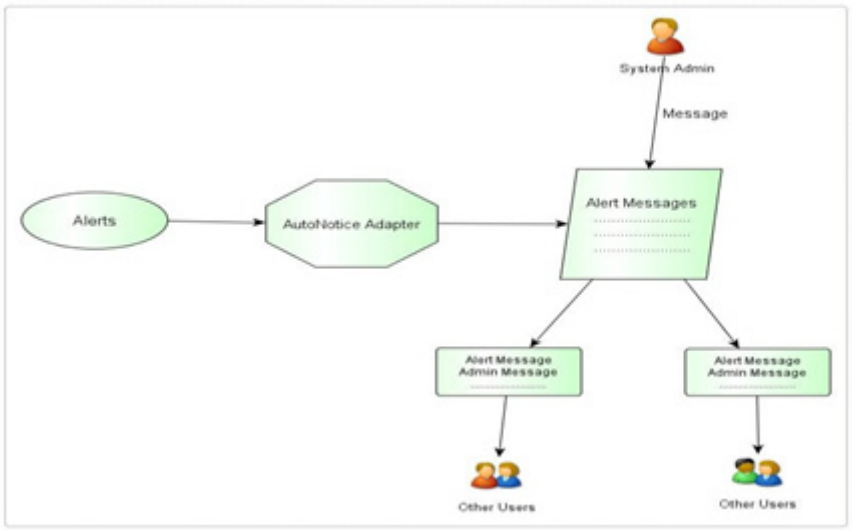
그림 11-12:공지 수정



11.6.1.Auto Notice

경보 내용을 자동으로 사용자에게 공지하는 기능을 자동공지(AutoNotice)라 한다.

그림 11-13:자동 공지



자동 공지는 sms Adapter를 등록함으로써 사용할 수 있다.

```
sms_adapter_class_name = com.javaservice.jennifer.server.AutoNotice
```

11.6.2. 공지기능 제거

이전 버전 처럼 공지기능을 사용하지 않으려면 제니퍼 서버 옵션에 다음과 같이 설정한다.

```
ui_dashboard_notice=True
```

12

제니퍼 서버 운영 관리

12.1. 제니퍼 서버 시작과 중지

제니퍼 서버를 시작하려면 JENNIFER_HOME/server/bin 디렉토리에 있는 startup.sh를 실행한다.

```
./startup.sh
```

Notice: 단, 닷넷 에이전트만을 전용으로 모니터링을 할 경우에는 startup.net.bat를 사용하여 제니퍼 서버를 기동해야 한다.

제니퍼 서버를 기동하게 위해서는 JDK가 필요하다. 제니퍼 서버는 JDK1.5이후 버전의 사용을 권장한다. JDK 1.4.2를 사용하는 경우, 다음 작업을 수행한다.

- JENNIFER_HOME/server/doc/jdk142/bin/jmx.jar 파일을 JENNIFER_HOME/server/bin 디렉토리로 복사한다.
- 그리고 JENNIFER_HOME/server/doc/jdk142/common/endorsed 디렉토리를 JENNIFER_HOME/server/common 디렉토리 밑으로 복사한다.
- 제니퍼 서버를 시작한다.

표준 출력 로그는 JENNIFER_HOME/server/logs/catalina.out 파일에서 확인할 수 있다.

제니퍼 서버를 정지하려면 JENNIFER_HOME/server/bin 디렉토리에 있는 shutdown.sh 를 실행한다.

```
./shutdown.sh
```

12.2. 제니퍼 서버 설정 파일

모니터링 수준과 방법, 네트워크 포트 번호 지정 등의 모든 설정은 제니퍼 서버 옵션을 통해서 이루어진다. JENNIFER_HOME/server/bin/jennifer.properties 파일이 제니퍼 서버 설정 파일이다.

12.2.1. 설정 파일 형식

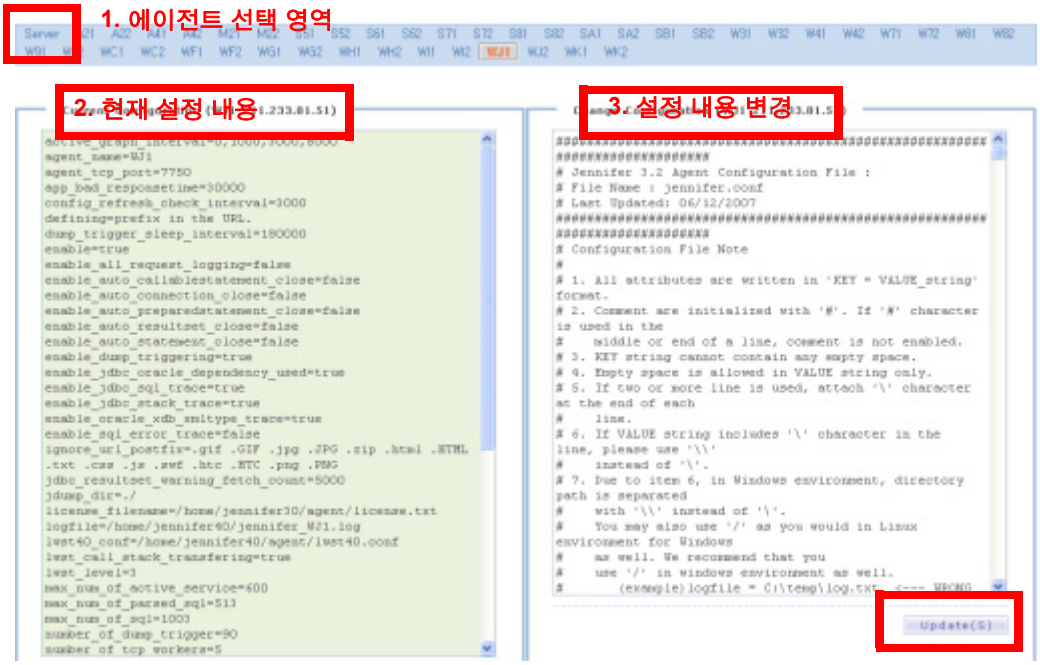
제니퍼 서버 옵션은 텍스트 형식의 설정 파일에 기록된다. 이 파일은 다음과 같은 특징을 갖는다.

- 키 = 값 형식으로 기술한다. 키는 제니퍼 서버의 옵션을 의미한다.
- [#]으로 시작하는 줄은 주석이다. 그러나 문자열 중간에 있는 [#] 기호는 주석으로 인식되지 않는다.
- [=] 대신 '공백'을 키와 값의 구분자로 사용할 수 있다.
- 앞의 이유로 인하여 키에 공백을 사용하는 것을 허용하지 않는다.
- 그러나 값에는 공백을 사용할 수 있다.
- 두 줄 이상을 사용하려면 라인의 끝에 [\] 문자를 사용한다. 만약, [\] 문자 자체가 필요한 경우에는, [\]대신 [\\]을 사용한다.
- 앞의 이유로 인하여 마이크로소프트 윈도우즈 환경에서 디렉토리 구분은 [\]가 아니라 [\\]로 해야 한다. 윈도우즈 환경에서 유닉스처럼 [/]을 사용할 수 있기 때문에 윈도우즈 환경에서도 [/]을 사용하는 것을 권장한다.
- 일부 옵션을 제외한 대부분의 옵션들은 수정과 함께 실시간으로 반영된다. 수정된 옵션을 반영하기 위해 제니퍼 서버를 재시작해야 하는 경우에는 해당 옵션을 설명할 때 이를 명기하였다.
- 설정 파일에 기술되어 있지 않은 옵션들이 존재한다. 이 옵션들을 사용하는 경우에는 해당 옵션을 새로 추가한다.

12.2.2. 설정 변경

설정 파일은 에디터를 통해서 직접 수정하거나, **[구성 관리 | 구성 설정]** 메뉴에서 수정한다. 단, 제니퍼 클리이언트에서는 관리자 그룹에 속한 사용자만이 옵션을 수정할 수 있다.

그림 12-1: 제니퍼 옵션 설정 화면



1. 에이전트 선택 영역 - 제니퍼 서버 옵션 내용을 확인하거나 수정하려면 [서버]를 선택한다.
2. 현재 설정 내용 - 왼쪽에 있는 현재 설정 내용에서 제니퍼 서버 옵션을 확인한다.
3. 설정 내용 변경 - 오른쪽에 있는 설정 내용 변경 입력 폼에서 기존 옵션을 수정하거나 새로운 옵션을 추가한 후에, 하단에 있는 **[수정]** 버튼을 클릭한다.

제니퍼 서버 설정 파일은 제니퍼 서버의 config_refresh_check_interval 옵션으로 설정한 시간마다 변경 여부가 체크된다.

```
config_refresh_check_interval = 3000
```

12.3. 제니퍼 서버 네트워크 구성

제니퍼 클라이언트 혹은 제니퍼 에이전트와 제니퍼 서버 사이에 일어나는 다양한 네트워크 통신을 처리하기 위한, 제니퍼 서버의 네트워크 구성에 대해서 설명한다. 주로 포트 번호 설정과 관련된 것으로, 기본 포트 번호를 다른 애플리케이션에서 이미 사용하고 있다면 이를 수정해야 한다. 또한 동일한 하드웨어에 복수의 제니퍼 서버를 설치한 경우에도 해당 포트 번호가 중복되지 않도록 설정해야 한다.

12.3.1.클라이언트를 위한 설정

제니퍼 서버와 제니퍼 클라이언트는 웹 기반 사용자 인터페이스 제공을 위해서 HTTP 프로토콜을 사용한다. 이 때 사용하는 기본 HTTP 포트 번호는 7900이다. 그리고 제니퍼 서버 정지를 위한 기본 포트 번호는 7999이다. 이 포트 번호를 변경하려면 JENNIFER_HOME/server/bin/catalina.sh(bat) 파일을 수정한다.

유닉스 혹은 리눅스의 경우에는 catalina.sh 파일의 다음 부분에서 포트 번호를 설정한다.

```
JAVA_HOME="$JAVA_HOME"

if [ -z "${STARTUP_PORT}" ]
then
    export STARTUP_PORT="7900"
fi

if [ -z "${SHUTDOWN_PORT}" ]
then
    export SHUTDOWN_PORT="7999"
fi
```

윈도우즈의 경우에는 catalina.bat 파일의 다음 부분에서 포트 번호를 설정한다.

```
set JAVA_HOME=%JAVA_HOME%

if "%STARTUP_PORT%" == "" SET STARTUP_PORT=7900
if "%SHUTDOWN_PORT%" == "" SET SHUTDOWN_PORT=7999
```

자바 애플릿은 차트 구성에 필요한 데이터를 제니퍼 서버로부터 TCP 통신을 통해서 획득한다. 이 때 사용되는 제니퍼 서버의 기본 TCP 포트 번호는 제니퍼 서버의 server_tcp_port 옵션으로 설정하고, 기본 포트 번호는 6701이다.

```
server_tcp_port = 6701
```

12.3.2.제니퍼 에이전트를 위한 설정

제니퍼 에이전트는 성능 데이터를 UDP 방식으로 제니퍼 서버에 전송한다. 제니퍼 서버는 제니퍼 에이전트가 보내는 성능 데이터를 3개의 UDP 포트로 분리해서 받아들인다. 각각의 포트가 받아들이는 성능 데이터는 상이하다.

제니퍼 에이전트는 제니퍼 서버의 server_udp_runtime_port 옵션으로 설정한 UDP 포트로 모든 트랜잭션의 시작과 종료와 관련한 데이터를 전송한다. 이 데이터는 크기는 매우 작으며 주로 X-View 차트를 표현하는데 사용된다. 기본 포트 번호는 6901이다.

```
server_udp_runtime_port = 6901
```

제니퍼 에이전트는 제니퍼 서버의 server_udp_listen_port 옵션으로 설정한 UDP 포트로 1초마다 반복적으로 서비스 요청률, 평균 응답 시간 등의 일반 성능 데이터를 전송한다. 기본 포트 번호는 6902이다.

```
server_udp_listen_port = 6902
```

제니퍼 에이전트는 제니퍼 서버의 server_udp_lwst_call_stack_port 옵션으로 설정한 UDP 포트로 X-View 트랜잭션 프로파일 데이터를 전송한다. 기본 포트 번호는 6703이다.

```
server_udp_lwst_call_stack_port = 6703
```

Notice: 앞에서 설명한 UDP 포트 번호를 수정할 때는 제니퍼 서버 뿐만 아니라 제니퍼 에이전트의 동일한 옵션도 함께 수정해야 한다.

그리고 제니퍼 에이전트가 UDP 방식으로 보내는 성능 데이터를 제니퍼 서버가 받을 때 사용할 IP 바인딩 주소를 지정할 필요가 있다. 이는 자바 TCP 소켓 프로그래밍에서 new java.net.DatagramSocket(port, ip) 생성자의 두번째 파라미터 IP에 해당하는 값이다. 하드웨어에 둘 이상의 네트워크 카드가 있다면, 특정 네트워크 카드로 들어오는 요청만 바인딩하고자 할 때 설정한다. udp_server_host 속성을 "0.0.0.0"으로 지정하면 모든 네트워크 카드에서 들어오는 패킷을 받을 수 있다.

```
udp_server_host = 0.0.0.0
```

제니퍼 서버에서 제니퍼 에이전트로의 TCP 역방향 연결에 대한 타임 아웃 시간을 설정할 수 있다. 소켓 연결에 대한 타임 아웃 시간은 제니퍼 서버의 agent_tcp_connect_timeout 옵션으로 설정한다. 기본 값은 3000이고 단위는 밀리 세컨드이다.

```
agent_tcp_connect_timeout = 3000
```

소켓 연결 후 데이터를 읽는 작업과 관련한 타임 아웃 시간은 제니퍼 서버의 agent_tcp_io_timeout 옵션으로 설정한다. 기본 값은 5000이고 단위는 밀리 세컨드이다.

```
agent_tcp_io_timeout = 5000
```

Notice: 만약 네트워크 연결이 지연되는 현상이 발생하면, 제니퍼 서버 로그 파일에 에러가 기록된다. 이때는 옵션 값을 바꾸기 전에 먼저 제니퍼 에이전트와 서버 사이의 네트워크 환경을 튜닝하는 것을 권고한다.

12.3.3.제니퍼 사용을 위한 방화벽 설정

제니퍼 서버와 제니퍼 에이전트 혹은 제니퍼 클라이언트 사이에 방화벽이 존재하면, 앞에서 설정한 포트 번호가 방화벽을 정상적으로 통과하도록 설정해야 한다.

- 사용자 컴퓨터(제니퍼 클라이언트)에서 제니퍼 서버로 HTTP 7900 포트와 TCP 6701 포트를 통해서 접근할 수 있어야 한다.
- 제니퍼 에이전트에서 제니퍼 서버로 UDP 6901, 6902, 6703 포트를 통해서 접근할 수 있어야 한다.
- 제니퍼 서버에서 제니퍼 에이전트로 TCP 7750 포트를 통해서 접근할 수 있어야 한다.

Reference: UDP 네트워크에 대한 방화벽 테스트는 [네트워크 테스트]를 참조한다.

12.4. 서버 로그 관리

제니퍼 서버의 로그 파일은 제니퍼 서버의 logfile 옵션으로 설정한다.

```
logfile = ../logs/jennifer.log
```

로그 파일을 일자별로 기록하려면 제니퍼 서버의 enable_logfile_daily_rotation 옵션을 true로 설정한다. 기본 값은 true이다.

```
enable_logfile_daily_rotation = true
```

기본 설정에 따르면 JENNIFER_HOME/server/logs/jennifer.YYYY-MM-DD.log 파일에 로그가 기록된다.

Notice: logfile 옵션을 변경하려면 제니퍼 서버를 재시작하여야 한다.

제니퍼 서버는 아파치 톰캣을 사용하고 톰캣 로그는 JENNIFER_HOME/server/logs 디렉토리에 일자별로 분리되어 기록된다.

표 12-1: 톰캣 로그 파일

로그 파일	설명
admin.YYYY-MM-DD.log	톰캣이 내부적으로 사용하는 로그 파일로 중요 메시지가 기록되지는 않는다.
catalina.YYYY-MM-DD.log	콘솔에 출력되는 메시지가 기록된다.
host-manager.YYYY-MM-DD.log	톰캣이 내부적으로 사용하는 로그 파일로 중요 메시지가 기록되지는 않는다.
localhost.YYYY-MM-DD.log	에러 및 예외 메시지가 기록된다.
manager.YYYY-MM-DD.log	톰캣이 내부적으로 사용하는 로그 파일로 중요 메시지가 기록되지는 않는다.

Notice: 톰캣 서버와 관련한 로그는 catalina.YYYY-MM-DD.log 파일과 localhost.YYYY-MM-DD.log 파일을 주로 참고한다.

12.5. 동일한 하드웨어에서 복수의 제니퍼 서버 운영하기

동일한 하드웨어에서 복수의 제니퍼 서버를 운영하는 방법을 설명한다. 운영하려는 제니퍼 서버의 개수와 상관없이 하나의 제니퍼 서버만을 설치한다. 그리고 운영하려는 개수에 해당하는 시작/정지 스크립트를 작성한다. 시작/정지 스크립트에 제니퍼 서버가 사용하는 HTTP 포트 번호와 제니퍼 서버 정지 포트 번호, 그리고 제니퍼 서버 설정 파일 경로를 설정한다.

유닉스 혹은 리눅스의 경우에는 JENNIFER_HOME/server/bin/start_server_01.sh 파일을 다음과 같이 작성한다. 임의의 파일 이름을 사용할 수 있다.

```
export JAVA_HOME=/usr/java/jdk1.6.0_11

export STARTUP_PORT=7901
export SHUTDOWN_PORT=7991

export JAVA_OPTS=-Djennifer.config=/jennifer/data/conf/
jennifer_01.properties

./startup.sh
```

그리고 JENNIFER_HOME/server/bin/shutdown_server_01.sh 파일을 다음과 같이 작성한다. 임의의 파일 이름을 사용할 수 있다.

```
export JAVA_HOME=/usr/java/jdk1.6.0_11

export STARTUP_PORT=7901
export SHUTDOWN_PORT=7991

export JAVA_OPTS=-Djennifer.config=/jennifer/data/conf/
jennifer_01.properties

./shutdown.sh
```

윈도우즈의 경우에는 JENNIFER_HOME/server/bin/start_server_01.bat 파일을 다음과 같이 작성한다. 임의의 파일 이름을 사용할 수 있다.

```
set JAVA_HOME=C:\Program Files\Java\jdk1.6.0_11

SET STARTUP_PORT=7901
SET SHUTDOWN_PORT=7991

set JAVA_OPTS=-Djennifer.config=C:/jennifer/data/conf/
jennifer_01.properties

startup run
```

그리고 JENNIFER_HOME/server/bin/shutdown_server_01.bat 파일을 다음과 같이 작성한다. 임의의 파일 이름을 사용할 수 있다.

```
set JAVA_HOME=C:\Program Files\Java\jdk1.6.0_11

SET STARTUP_PORT=7901
SET SHUTDOWN_PORT=7991
set JAVA_OPTS=-Djennifer.config=C:/jennifer/data/conf/
jennifer_01.properties

shutdown run
```

각 제니퍼 서버별로 다르게 설정해야 하는 제니퍼 서버의 옵션은 다음과 같다.

- domain_name
- server_tcp_port
- logfile
- upload_directory
- data_directory
- system.derby.system.home
- backup_root

자바 임시 디렉토리도 다르게 하려면 시작 스크립트에서 JAVA_OPTS 환경 변수로 java.io.tmpdir를 설정한다.

12.6. 제니퍼 서버 디버깅

12.6.1. 이벤트 로그 기록

사용자 및 권한 등과 관련한 주요 이벤트 로그를 관리자 데이터베이스 EVENT_LOG 테이블에 기록할 수 있다. 이벤트 로그를 기록하려면 제니퍼 서버의 enable_event_log 옵션을 true로 설정한다. 기본 값은 false이다.

```
enable_event_log = true
```

주요 이벤트 로그에 대한 설명은 다음과 같다.

표 12-2: 주요 이벤트 로그

이벤트 로그 아이디	설명
JENNIFER_SERVER_START	제니퍼 서버의 시작 이벤트 로그
JENNIFER_SERVER_STOP	제니퍼 서버의 종료 이벤트 로그
ADD_USER	사용자 추가 이벤트 로그(사용자에게 할당된 그룹 정보 포함)
CHANGE_USER	사용자 변경 이벤트 로그(사용자에게 할당된 그룹 정보 포함)
REMOVE_USER	사용자 삭제 이벤트 로그
ADD_GROUP	그룹 추가 이벤트 로그(제니퍼 에이전트 목록 정보 포함)
CHANGE_GROUP	그룹 변경 이벤트 로그(제니퍼 에이전트 목록 정보 포함)
REMOVE_GROUP	그룹 삭제 이벤트 로그
ADD_ROLE	권한 추가 이벤트 로그(권한에 할당된 그룹 정보 포함)
CHANGE_ROLE	권한 변경 이벤트 로그(권한에 할당된 그룹 정보 포함)
REMOVE_ROLE	권한 삭제 이벤트 로그
CHANGE_PASSWORD	패스워드 변경 이벤트 로그
CHANGE_PASSWORD_BY_ADMIN	권리자 그룹에 속한 사용자에 의한 패스워드 변경 이벤트 로그
LOGIN_SUCCESS	로그인 성공 이벤트 로그
LOGIN_FAILURE	로그인 실패 이벤트 로그로 로그인 실패 사유도 함께 기록된다.
LOGOUT_SUCCESS	로그아웃 성공 이벤트 로그로 사용자에 의한 로그아웃과 세션 타임아웃에 의한 로그아웃을 구분한다.
LOGOUT_FAILURE	로그아웃 실패 이벤트 로그
INVALID_LOGIN_LOCK	사용자가 정해진 임계치 이상으로 패스워드를 틀리면 해당 사용자 계정을 사용할 수 없게 된다. 이 내용을 기록하는 이벤트 로그이다.

12.6.2.제니퍼 서버 SQL 로깅

제니퍼 서버 튜닝을 위해서 제니퍼 서버가 사용하고 있는 내부 SQL과 에러 여부를 임의의 로그 파일에 기록할 수 있다. 이를 위해서는 제니퍼 서버의 enable_server_trace 옵션을 true로 설정한다. 기본 값은 false이다.

```
enable_server_trace = true
```

그리고 로그를 기록할 파일을 제니퍼 서버의 server_trace_filename 옵션으로 설정한다. 기본 값은 servertrace.log이다.

```
server_trace_filename = servertrace.log
```

그리고 SQL 실행 시간이 제니퍼 서버의 server_jdbc_trace_overthan 옵션으로 설정한 시간 이상인 경우에만 SQL을 로그 파일에 기록한다. 기본 값은 10000이고 단위는 밀리 세컨드이다.

```
server_jdbc_trace_overthan = 10000
```

제니퍼 클라이언트 TCP 연결 로깅 제니퍼 서버를 사용하는 사용자가 많은 경우에 Client TCP Worker 숫자가 부족하여 예외가 발생할 수 있다. 이 때 제니퍼 서버의 number_of_tcp_pooled_workers 옵션으로 Client TCP Worker 숫자를 늘려주기 전에 실제 유효한 사용자가 정상적으로 TCP 연결을 사용하는지를 테스트하려면 제니퍼 서버의 debug_tcp 옵션을 true로 설정한다. 기본 값은 false이다. 이를 true로 설정하면 제니퍼 서버 로그 파일에 제니퍼 클라이언트 TCP 요청 내역이 기록된다.

```
debug_tcp = true
```

12.7. 제니퍼 스케줄러

제니퍼 서버는 데이터 관리 혹은 임의의 작업을 수행하기 위한 스케줄러를 지원한다. 제니퍼에서는 이런 스케줄러를 TimeActor라고 한다. 제니퍼는 몇 가지 기본 스케줄러를 제공하며, 필요시 임의의 스케줄러를 작성하여 추가할 수 있다.

12.7.1. 기본 스케줄러

제니퍼는 CleanerActor, SummaryActor, ReportActor, FileCleanerActor 등의 기본 제니퍼 스케줄러를 제공한다.

- CleannerActor - 파일과 데이터베이스에 저장된 성능 데이터를 주기적으로 삭제한다.
- SummaryActor - 일자별 테이블에 대한 조회 성능 개선을 위한 통계 성능 데이터를 생성한다.
- ReportActor - 보고서를 자동 생성한다.
- FileCleanerActor - 제니퍼 서버의 로그 파일과 임시 파일을 삭제한다. 임시 파일은 자바 java.io.tmpdir 환경 변수로 설정한 디렉토리에 존재하는 파일을 의미한다.

- BackupActor - 제니퍼 서버가 수집한 성능 데이터를 백업한다.

FileCleanerActor는 다음과 같이 설정한다.

```
time_actor_14 =
com.javaservice.jennifer.server.timeactor.FileCleanerActor 02 30
```

첫번째 파라미터 02는 수행 시간을 의미하고, 두번째 파라미터 30은 기간을 의미한다. 따라서 기본적으로 매일 02시에 30일이 지난 로그 파일과 임시 파일을 삭제한다. 단, 확장자가 log와 png로 끝나는 파일만을 삭제한다.

Notice: 나머지 스케줄러를 설정하는 방법은 해당 스케줄러와 관련된 부분에서 설명한다.

12.7.2.제니퍼 스케줄러 작성

임의의 스케줄러를 작성하는 방법은 다음과 같다.

12.7.2.1. 스케줄러 클래스 작성

모든 스케줄러 클래스는 com.javaservice.jennifer.server.TimeActor 클래스를 상속해야 한다. TimeActor 클래스는 JENNIFER_HOME/server/common/lib/jenniferserver.jar 파일에 포함되어 있다. 따라서, 스케줄러 클래스를 작성하기 위해서는 jenniferserver.jar파일을 클래스 패스에 등록하여야 한다.

제니퍼 스케줄러 클래스는 process 메소드를 구현해야 한다.

```
package sample;

import com.javaservice.jennifer.server.TimeActor;

public class SampleActor extends TimeActor {
    public void process(String[] args) { }
}
```

process 메소드의 파라미터는 스케줄러를 제니퍼 서버에 등록할 때 설정한 파라미터 목록이다. 예를 들어, 다음과 같이 스케줄러를 등록하면, process 메소드의 파라미터는 {"03", "sample"}이 된다. 파라미터의 의미는 스케줄러마다 다르지만, 첫번째 파라미터로 해당 스케줄러가 수행될 시간을 지정하는 것을 권장한다.

```
time_actor_32 = sample.SampleActor 03 sample
```

스케줄러의 process 메소드는 1분마다 반복적으로 호출된다. 따라서 특정 시점에 해당 스케줄러가 임의의 작업을 수행해야 할지에 대한 판단을 제니퍼 서버가 아닌 스케줄러가 스스로 결정해야 한다.

다음은 TimeActor 클래스가 제공하는 주요 메소드 API에 대한 설명이다.

표 12-3: TimeActor 클래스 API

메소드	설명
public String getYYYYMMDD()	현재 시간을 yyyyMMdd 포멧으로 반환한다.
public static String toString(Calendar day)	파라미터 java.util.Calendar 객체의 시간을 yyyyMMdd 포멧으로 반환한다.
public String getYYYYMMDD(int type, int n)	현재를 기준으로 이전이나 이후의 일자를 yyyyMMdd 포멧으로 반환한다. 첫번째 파라미터 type에는 Calendar.YEAR나 Calender.DATE나 Calender.HOUR 등을 지정할 수 있다. 두번째 파라미터 n에는 변동 크기를 지정한다. 음수 값은 과거를, 양수 값은 미래를 의미한다. 예를 들어 getYYYYMMDD(Calender.MONTH, -1)은 한달 전의 날짜를 반환한다.
public String getHOUR()	현재 시간을 HH 포멧으로 반환한다.
Connection getAdmConn() throws SQLException	제니퍼 관리자 데이터베이스에 대한 java.sql.Connection 객체를 반환한다.
public Connection getDataConn() throws SQLException	제니퍼 성능 데이터베이스에 대한 java.sql.Connection 객체를 반환한다.
public void log(String message)	제니퍼 서버 로그 파일에 로그 메시지를 기록한다.

Notice: 스케줄러로 제니퍼 데이터베이스에 대한 작업을 수행한 후에는 반드시 사용한 JDBC 자원(java.sql.Connection, java.sql.Statement, java.sql.ResultSet)을 반환(close)한다. 그렇지 않으면 제니퍼 서버에서 장애가 발생할 수 있다.

12.7.2.2. 스케줄러 클래스 패키지 및 배포

컴파일한 스케줄러 클래스를 임의의 JAR 파일로 패키지한 후에, 해당 파일을 JENNIFER_HOME/server/common/lib 디렉토리에 복사한다.

12.7.2.3. 스케줄러 설정

새로운 스케줄러는 제니퍼 서버의 time_actor 옵션을 사용해서 등록한다.

```
time_actor_32 = sample.SampleActor 03 sample
```

스케줄러를 등록할 때 사용하는 제니퍼 서버의 옵션은 time_actor_로 시작하고, 다른 스케줄러와 구분하기 위한 2자의 고유한 숫자로 끝나야 한다.

Notice: 새로운 스케줄러의 추가나 기존 스케줄러의 변경은 제니퍼 서버의 재시작이 필요하다.

12.8. 사용자, 권한 그리고 메뉴

사용자와 그룹의 관계를 설명하고 권한을 그룹에 할당하는 방법을 설명한다. 그리고 메뉴를 관리하는 방법과 사용자 혹은 그룹 별로 메뉴를 다르게 설정하는 방법 등을 설명한다.

12.8.1.사용자 관리

사용자는 제니퍼 서버에 로그인할 수 있는 계정을 의미한다. 사용자 별로 접근 가능한 메뉴를 다르게 설정할 수 있으며 권한의 수준을 다르게 지정할 수 있다.

제니퍼 서버를 설치하면 admin 사용자가 만들어진다. 기본적으로 admin 사용자는 모든 메뉴에 접근할 수 있고 모든 권한을 가진다. 사용자를 추가하거나 그 정보를 변경하려면 **[구성 관리 | 사용자 관리 | 사용자 관리]** 메뉴를 사용한다.

12.8.1.1. 사용자와 그룹

사용자에 따라서 메뉴 구성을 다르게 하거나 권한을 다르게 부여할 수 있다. 사용자가 많아지면 이와 관련한 설정 작업이 불편할 수 있다. 제니퍼 서버는 동일한 유형의 사용자를 그룹으로 묶고, 그 그룹에 대해서 메뉴 구성이나 권한을 설정할 수 있다. 즉, 그룹은 동일한 메뉴에 접근하고 동일한 권한을 가지고 있는 사용자의 집합을 의미한다.

12.8.1.2. admin 사용자와 admin 그룹

제니퍼 서버를 설치하면 admin 사용자와 admin 그룹이 기본적으로 만들어진다. admin 사용자와 admin 그룹은 임의로 삭제할 수 없다. 또한 admin 사용자는 자신이 속한 그룹(admin)을 변경할 수 없다.

사용자와 관련한 정보는 다음과 같다.

표 12-4: 사용자 정보

항목	설명
아이디	로그인에 사용하는 계정 아이디로, 신규 생성 후에는 변경할 수 없다.

표 12-4: 사용자 정보

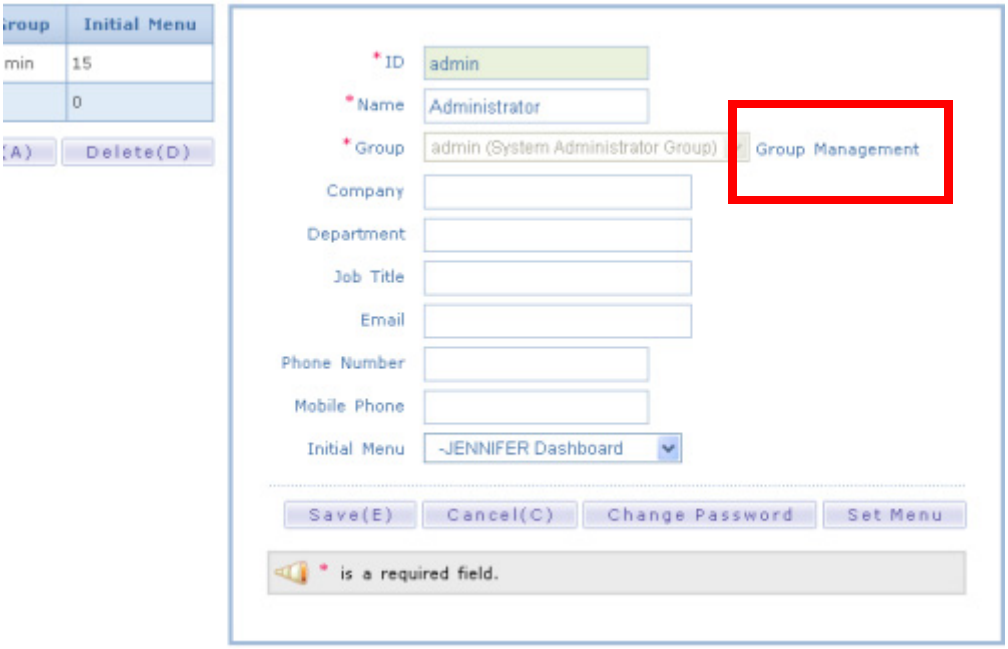
항목	설명
이름	사용자 이름
그룹	사용자가 속한 그룹으로 사용자는 하나의 그룹에만 속할 수 있다. 단 여러 사용자는 동일한 그룹에 속할 수 있다.
회사	사용자가 속한 회사
부서	사용자가 속한 부서
직책	사용자 직책
이메일	사용자 이메일 주소
전화번호	사용자 전화번호
핸드폰	사용자 이동통신 전화번호
초기 메뉴	로그인 후에 이동하는 첫번째 메뉴로 사용자 혹은 사용자가 속한 그룹이 해당 메뉴에 접근할 수 있어야 한다.

Notice: 회사, 부서, 직책, 이메일, 전화번호, 핸드폰 등은 부가 정보로서 제니퍼 서버 사용에 영향을 주지 않는다.

12.8.1.3. 그룹 관리

사용자 정보 편집 화면에서 그룹을 선택하는 드랍 다운 박스 옆에 있는 **[그룹 관리]** 링크를 통해서 그룹 정보를 관리한다.

그림 12-2: 그룹 관리 링크



그룹과 관련한 정보는 다음과 같다.

표 12-5: 그룹 정보

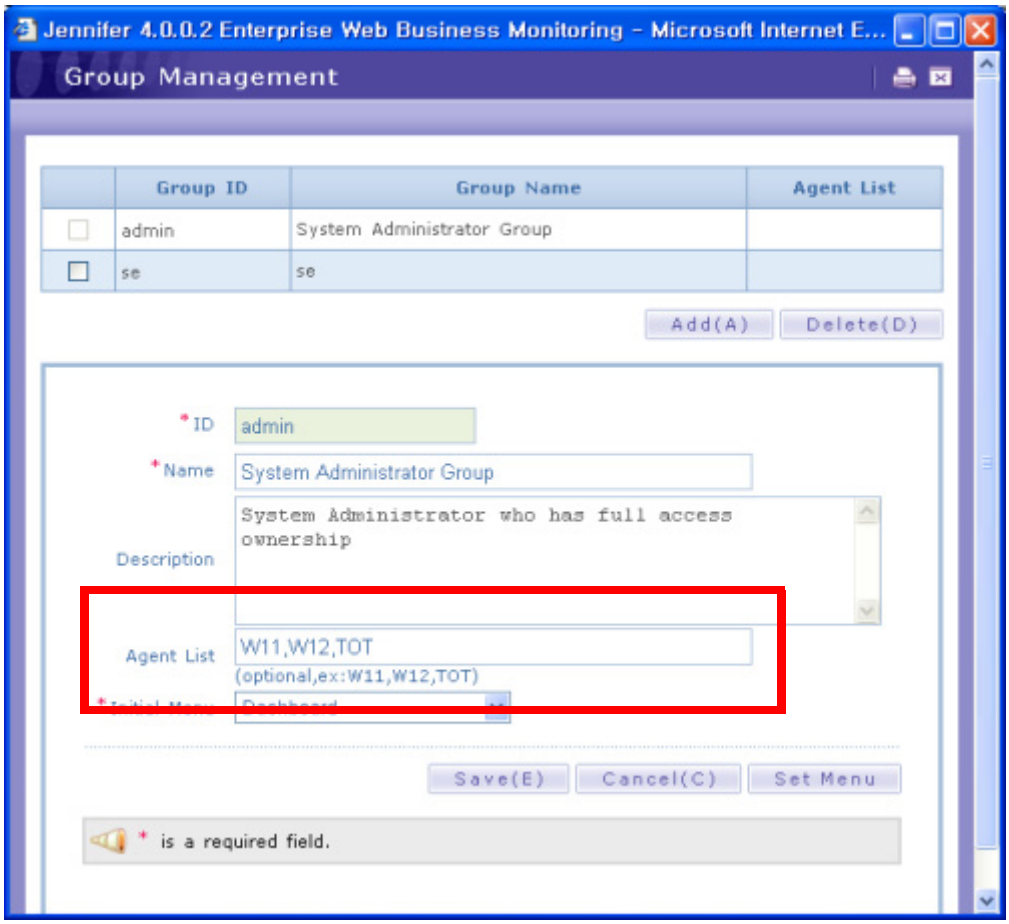
항목	설명
아이디	그룹 아이디
이름	그룹 이름
설명	그룹에 대한 설명
에이전트 목록	그룹 별 에이전트 지정하기를 참조한다.
초기 메뉴	해당 그룹에 속한 사용자가 로그인 후에 이동하는 첫번째 메뉴로 그룹이 해당 메뉴에 접근할 수 있어야 한다. 사용자 정보에 설정한 초기 메뉴가 그룹 정보에 설정한 초기 메뉴에 우선한다. 설정하지 않은 경우에는 **[대시보드 | 제니퍼 대시보드]** 메뉴가 초기 메뉴가 된다.

12.8.1.4. 그룹 별 에이전트 지정하기

사용자 별로는 모니터링하는 에이전트를 다르게 지정할 수 없지만, 그룹 별로는 모니터링하는 에이전트를 다르게 지정할 수 있다. 예를 들어, 제니퍼 서버를 통해서 W11, W12, W13 등의 에이전트를 모니터링하고 있는 상황에서, A 그룹에게는 W11, W12만을, B 그룹에게는 W12, W13만을 모니터링하게 할 수 있다.

그룹 정보 편집 화면에서 특정 그룹이 모니터링할 에이전트 이름을 콤마(,)를 구분자로 하여 에이전트 목록 필드에 입력한다. 전체를 의미하는 가상 에이전트 TOT를 입력하지 않으면 대시보드 등에서 전체와 관련한 정보를 확인할 수 없다.

그림 12-3: 에이전트 목록 설정 화면



12.8.1.5. 사용자 숫자 제한하기

기본적으로 제니퍼 서버를 동시에 사용할 수 있는 사용자의 숫자를 제한하지 않는다. 동시 사용자 수를 제한하고자 하는 경우, 제니퍼 서버의 userlogin_count_limit 옵션을 통해서 사용자 숫자를 제한할 수 있다.

```
userlogin_count_limit = 10
```

제니퍼 서버와 제니퍼 클라이언트(애플릿)는 주기적으로 TCP 통신을 한다. 그런데 제니퍼 클라이언트와의 TCP 통신을 담당하는 제니퍼 서버의 소켓 수는 제한되어 있다. 이는 제니퍼 서버의 number_of_tcp_pooled_workers 옵션으로 설정하는데 기본 값은 80이다. 평균적으로 제니퍼 클라이언트 당 약 3개의 소켓을 사용하는데, 사용자가 증가하면 TCP

연결에 필요한 소켓을 할당받지 못해서 제니퍼 클라이언트가 정상적으로 동작하지 못할 수 있다. 따라서 이 경우에는 userlogin_count_limit 옵션을 통해서 사용자 숫자를 제한할 수 있다.

12.8.1.6. 동일 계정으로 로그인하는 것을 방지하기

기본적으로 여러 사용자가 동일한 사용자 계정으로 제니퍼 서버를 동시에 사용할 수 있다. 예를 들어, 사용자 A가 admin 사용자 계정으로 제니퍼 서버를 사용하고 있는 상태에서, 사용자 B가 동일한 admin 사용자 계정으로 제니퍼 서버에 로그인해도 어떠한 제약이 없다.

다른 사용자가 동일한 사용자 계정으로 제니퍼 서버를 사용하는 것을 방지하기 위해서는, 제니퍼 서버의 prevent_duplicated_login 옵션을 true로 지정한다.

```
prevent_duplicated_login = true
```

중복 로그인 방지는 나중에 로그인한 사용자가 먼저 로그인한 사용자를 밀어내는 방식으로 동작한다. 예를 들어, 사용자 A가 admin 사용자 계정으로 제니퍼 서버를 사용하고 있는 상태에서, 사용자 B가 동일한 admin 사용자 계정으로 제니퍼 서버에 로그인을 시도하면, 제니퍼 서버는 사용자 B에게 이미 admin 계정으로 로그인한 사용자가 있음을 알려준다.

그림 12-4: 중복 로그인 확인 화면

이 때 사용자 B는 먼저 로그인한 사용자 A를 밀어내고 제니퍼 서버에 로그인할 수 있는 선택권을 갖는다. 사용자 B가 사용자 A를 밀어내면 사용자 A는 자동으로 로그아웃된다.

Notice: prevent_duplicated_login 옵션의 사용 여부와 상관없이 실제 사용자가 다른 경우에는 사용자 아이디도 다르게 가져가는 것을 권장한다.

12.8.1.7. 로그인한 사용자 확인

[구성 관리 | 사용자 관리 | 로그인 사용자] 메뉴에서 제니퍼 서버에 로그인해 있는 사용자를 확인할 수 있다. 로그인한 사용자 목록에서 글자 하단에 줄이 있는 열은 사용자 자기 자신을 나타낸다.

표 12-6: 로그인한 사용자 목록

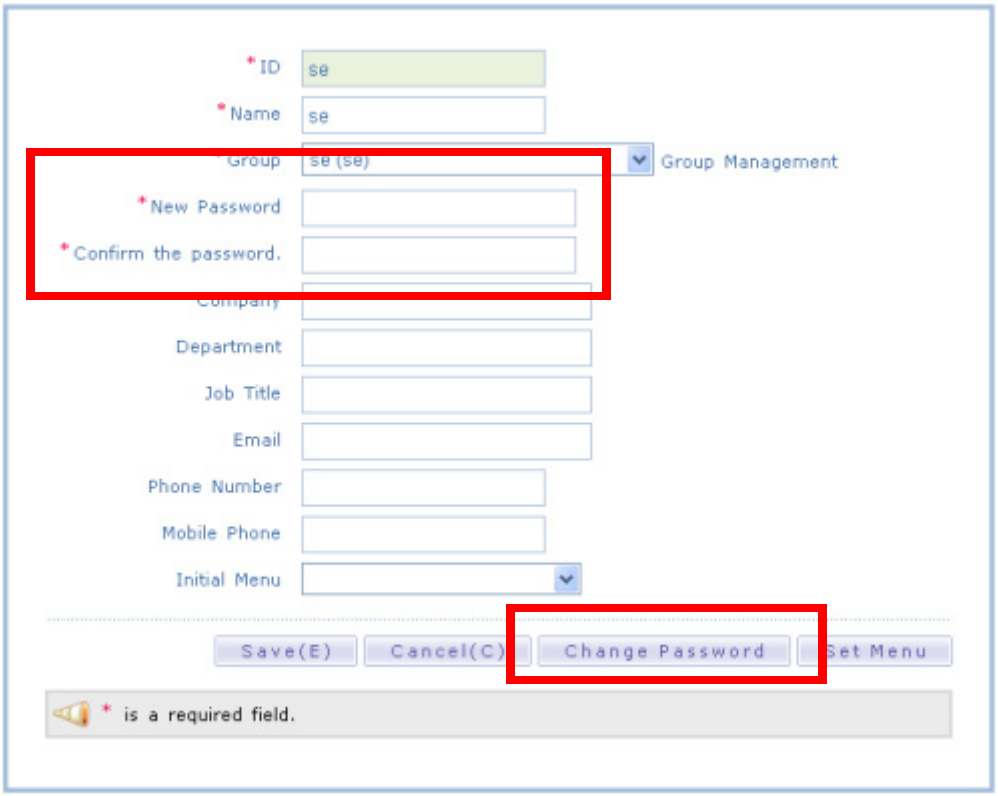
항목	설명
아이디	HTTP 세션 아이디
사용자 아이디	사용자 아이디
사용자 IP	사용자 IP
로그인 시간	제니퍼 서버에 로그인한 시간
마지막 접근 시간	마지막으로 제니퍼 서버에 접근한 시간
언어	사용자 언어

12.8.1.8. 패스워드 변경 및 패스워드 분실 조치

모든 사용자는 **[구성 관리 | 사용자 관리 | 패스워드 변경]** 메뉴에서 본인의 패스워드를 변경할 수 있다.

admin 그룹에 속한 사용자는 **[구성 관리 | 사용자 관리 | 사용자 관리]** 메뉴에서 본인을 포함한 모든 사용자의 패스워드를 변경할 수 있다. 타 사용자 패스워드는 다음과 같이 변경한다.

그림 12-5: 타 사용자 패스워드 변경



- 사용자 목록에서 패스워드를 변경할 사용자 아이디를 클릭한다.
- 사용자 정보 편집 화면 하단에 있는 **[패스워드 변경]** 버튼을 클릭하면 패스워드 입력 필드가 나타난다.
- 패스워드 입력 필드에 패스워드를 입력한 후에 **[저장]** 버튼을 클릭하면 해당 사용자의 패스워드가 변경된다.

admin 그룹에 속한 모든 사용자의 패스워드를 분실한 경우에는 다음과 같이 처리한다.

- 제니퍼 서버를 정지한다.
- JENNIFER_HOME/server/webapps/ROOT/WEB-INF/web.xml 파일에서 컨텍스트 파라미터 enable_password_check를 false로 변경한다. 이 값을 false로 설정하면 로그인을 할 때 패스워드를 체크하지 않는다.
- 제니퍼 서버를 시작한다.
- 로그인 화면에서 admin 그룹에 속하는 사용자 아이디와 임의의 패스워드를 입력하여 로그인한다.
- **[구성 관리 | 사용자 관리 | 사용자 관리]** 메뉴에서 admin 그룹에 속하는 사용자의 패스워드를 변경한다.

- 제니퍼 서버를 정지한다.
- JENNIFER_HOME/server/webapps/ROOT/WEB-INF/web.xml 파일에서 컨텍스트 파라미터 enable_password_check를 true로 변경한다.
- 제니퍼 서버를 시작한다.

12.8.2. 권한 관리

사용자는 제니퍼 서버를 통해서 자바 애플리케이션에 대한 가비지 콜렉션을 수행하거나 제니퍼 서버 사용자와 그룹 정보를 편집할 수 있다. 이와 같이 사용자가 수행할 수 있는 주요 행위를 권한이라고 하며 사용자가 속한 그룹 별로 권한을 다르게 부여할 수 있다.

예를 들어, gc 권한은 자바 애플리케이션에 대한 가비지 컬렉션 수행 권한을 의미하는데 A 그룹에 gc 권한을 부여하고 B 그룹에는 gc 권한을 부여하지 않으면, A 그룹에 속한 사용자는 자바 애플리케이션에 대한 가비지 컬렉션을 수행할 수 있지만 B 그룹에 속한 사용자는 자바 애플리케이션에 대한 가비지 컬렉션을 수행할 수 없다.

[구성 관리 | 사용자 관리 | 권한 관리] 메뉴에서 권한 목록을 확인하고 특정 권한을 특정 그룹에 부여한다.

표 12-7: 권한 목록

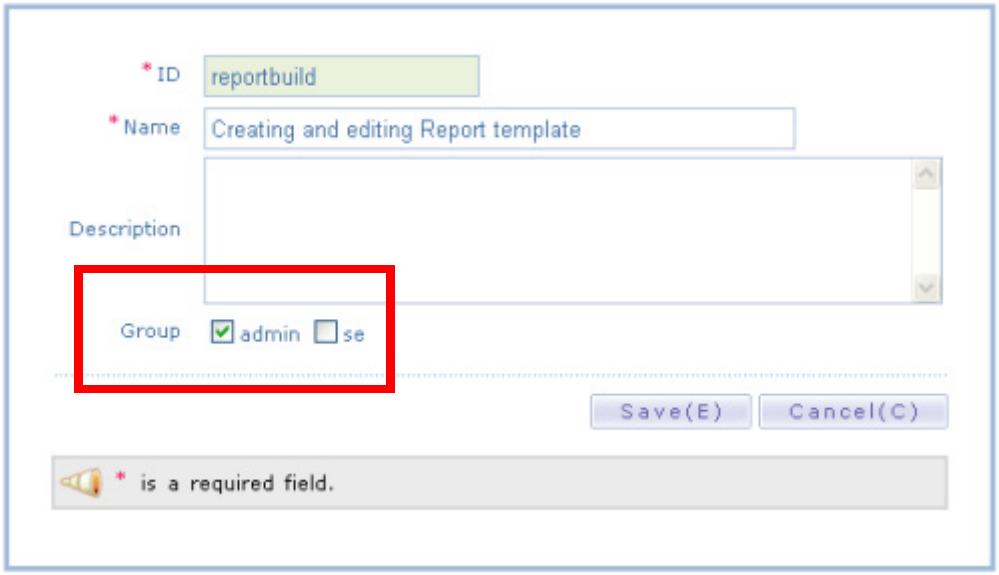
권한 아이디	설명
codedisassemble	자바 클래스 코드를 바이너리 포멧으로 볼 수 있는 권한
gc	자바 애플리케이션에 대해서 가비지 컬렉션을 수행할 수 있는 권한
groupedit	그룹을 편집할 수 있는 권한
killthread	자바 애플리케이션의 자바 쓰레드를 정지시킬 수 있는 권한
menuedit	메뉴를 편집할 수 있는 권한
pageedit	사용자 정의 대시보드를 구성할 수 있는 권한
reportbuild	보고서 템플릿을 작성할 수 있는 권한
sqlparam	SQL 파라미터를 볼 수 있는 권한
svcdump	자바 애플리케이션의 주요 정보를 덤프 파일에 기록할 수 있는 권한
useredit	사용자를 편집할 수 있는 권한
board	보드 영역 표시 여부와 관련한 권한

권한을 특정 그룹에 할당하려면 다음과 같이 한다.

- 권한 목록에서 그룹을 할당할 권한 아이디를 클릭한다.

- 권한 정보 편집 화면의 그룹 필드에서 권한을 할당하고자 하는 그룹을 체크하고 저장한다.

그림 12-6: 권한 편집 화면



12.8.3. 메뉴 관리

[구성 관리 | 메뉴 관리] 메뉴에서 메뉴를 관리한다. 제니퍼 서버는 3단계 메뉴로 구성된다.

그림 12-7: 메뉴 목록 화면

1. 메뉴 트리 노드 - 메뉴는 3단계 구조로 되어 있고 메뉴 트리 노드를 통해서 하위 메뉴를 열고 닫는다.
2. 컨텍스트 메뉴 - 메뉴 아이디 / 이름 칼럼을 클릭하면 컨텍스트 메뉴 목록이 나타난다.
3. 메뉴 수정 - 컨텍스트 메뉴 목록에서 **[수정]** 메뉴를 클릭하면 화면 오른 쪽에 메뉴를 수정할 수 있는 입력폼이 나타난다.
4. 하위 메뉴 추가 - 컨텍스트 메뉴 목록에서 **[하위 메뉴 추가]** 메뉴를 클릭하면 화면 오른 쪽에 하위 메뉴를 입력할 수 있는 입력폼이 나타난다. 단 메뉴는 3 단계까지만 입력이 가능하기 때문에 3 단계 메뉴의 경우에는 이 메뉴는 비활성화되어 있다.
5. 메뉴 삭제 - 컨텍스트 메뉴 목록에서 **[삭제]** 메뉴를 클릭하면 메뉴가 삭제된다.
6. 1 단계 메뉴 추가 - 메뉴 목록 오른쪽 하단에 있는 **[추가]** 버튼을 클릭하여 1 단계 메뉴를 추가한다.

새로운 메뉴를 생성할 때 메뉴 아이디는 제니퍼 서버에 의해서 자동으로 만들어진다.

메뉴 기본 정보는 다음과 같다.

표 12-8: 메뉴 정보

항목	설명
이름 코드	다국어 지원을 위한 메시지 키를 입력하는 필드이다.
상위 메뉴	해당 메뉴의 위치를 수정하려면 상위 메뉴 필드의 값을 수정한다.

표 12-8: 메뉴 정보

항목	설명
기본 하위 메뉴	사용자가 상단 영역에서 특정 메뉴를 클릭했을 때 해당 메뉴의 URL 등이 지정되어 있지 않은 경우에 기본 하위 메뉴로 설정한 메뉴의 URL로 이동한다.
순서	상위 메뉴가 같은 동일 단계의 메뉴들 간의 순서를 결정하는데 사용하는 필드이다. 필드의 값이 작은 메뉴를 먼저 보여준다.
그룹	해당 메뉴에 접근할 수 있는 그룹을 지정한다. 특정 메뉴에 대해서 접근 권한이 없는 그룹에 속한 사용자가 로그인을 하면 상단 영역에 해당 메뉴가 나타나지 않는다.

메뉴는 몇 가지 유형으로 구분되며 해당 유형에 따라서 입력할 수 있는 내용이 달라진다.

- URL 링크 메뉴 - 특정 URL을 링크하는 메뉴로 가장 일반적이다. 메뉴 유형으로 이 유형을 선택하면 URL 필드가 나타난다.
- URL 팝업 메뉴 - 특정 URL을 새로운 웹 브라우저 창에서 연다. 메뉴 유형으로 이 유형을 선택하면 URL 필드가 나타난다.
- 사용자 정의 대시보드 메뉴 - 드래그 & 드랍으로 사용자가 직접 화면을 구성하는 메뉴로 상단 영역에서 해당 메뉴를 클릭하면 메인 영역의 오른 쪽 하단에 편집 버튼이 나타난다. 이 버튼을 누르면 사용자 정의 대시보드 편집 화면으로 이동한다.
- 리포트 메뉴 - 보고서를 보여주는 메뉴이다. 메뉴 유형으로 이 유형을 선택하면 보고서 템플릿을 선택할 수 있는 드랍 다운 박스가 나타난다.

12.8.3.1. 사용자 혹은 그룹에 따른 메뉴 설정

메뉴 정보 편집 화면에서 특정 메뉴에 접근할 수 있는 그룹을 지정할 수 있다. 특정 사용자 혹은 그룹에 대해서 메뉴를 설정하는 것은 사용자 관리 혹은 그룹 관리에서 하는 것이 편리하다. 사용자에 대해서 메뉴를 설정하는 방법은 다음과 같다.

- **[구성 관리 | 사용자 관리 | 사용자 관리]** 메뉴의 사용자 목록에서 메뉴를 설정할 사용자 아이디를 클릭한다.
- 사용자 정보 편집 화면 하단에 있는 **[메뉴 설정]** 버튼을 클릭하면 팝업 창에 메뉴 목록이 나타난다.
- 해당 사용자가 접근할 수 있는 메뉴를 체크한 후에 하단에 있는 **[저장]** 버튼을 누른다.

그룹에 대해서 메뉴를 설정하는 방법은 다음과 같다.

- **[구성 관리 | 사용자 관리 | 사용자 관리]** 메뉴의 사용자 목록에서 임의의 사용자 아이디를 클릭한다.

- 사용자 정보 편집 화면에서 그룹을 지정하는 드랍 다운 박스 옆에 있는 **[그룹 관리]** 링크를 클릭하면 팝업 창에 그룹 목록이 나타난다.
- 그룹 목록에서 메뉴를 설정할 그룹 아이디를 클릭한다.
- 그룹 정보 편집 화면 하단에 있는 **[메뉴 설정]** 버튼을 클릭하면 팝업 창에 메뉴 목록이 나타난다.
- 해당 그룹이 접근할 수 있는 메뉴를 체크한 후에 하단에 있는 **[저장]** 버튼을 누른다.

Notice: 상위 메뉴와 하위 메뉴를 독립적으로 선택해야 한다. 상위 메뉴를 선택했을 때 자동으로 하위 메뉴가 선택되지 않는다. 그리고 상위 메뉴를 선택하지 않은 상태에서 하위 메뉴만을 선택하면 해당 하위 메뉴는 화면에 나타나지 않는다.

사용자에 대한 메뉴 설정이 그룹에 대한 메뉴 설정에 우선한다.

12.9. 패스워드 관리

12.9.1.로그인과 관련한 설정

보안 강화를 위해서 로그인과 관련한 다양한 정책을 설정할 수 있다. 우선 사용자가 제니퍼 서버에 처음 로그인을 한 후에 패스워드를 변경하도록 강제할 수 있다. 이 기능을 사용하려면 제니퍼 서버의 enable_initial_password_change 옵션을 true로 설정한다. 기본 값은 false이다.

```
enable_initial_password_change = true
```

이 옵션을 true로 설정하면 처음 로그인한 사용자는 패스워드 변경 화면에서 패스워드를 변경해야 다른 메뉴로 이동할 수 있다.

또한, 정해진 기간이 지나면 사용자가 패스워드를 변경하도록 강제할 수 있다. 이 기능을 사용하려면 제니퍼 서버의 enable_password_expiration_check 옵션을 true로 설정한다. 기본 값은 false이다.

```
enable_password_expiration_check = true
```

이 옵션을 true로 설정하면 일반 사용자는 60일이 지나면 패스워드를 변경해야 하고 관리자 그룹에 속하는 사용자는 30일이 지나면 패스워드를 변경해야 한다. 제니퍼 서버의 password_expiration_days 옵션을 통해서 패스워드의 유효 기간을 설정할 수 있다.

```
password_expiration_days = 60
```

이 옵션은 일반 사용자와 관리자 그룹에 속한 사용자에게 모두 적용된다. 관리자 그룹에 속한 사용자의 패스워드 유효 기간을 설정하려면 제니퍼 서버의 admin_password_expiration_days 옵션을 사용한다.

```
admin_password_expiration_days = 30
```

로그인 시에 사용자가 정해진 임계치 이상으로 패스워드를 틀리면 해당 사용자 계정을 사용하지 못하도록 설정할 수 있다. 이 기능을 사용하려면 제니퍼 서버의 enable_invalid_login_lock 옵션을 true로 설정한다. 기본 값은 false이다.

```
enable_invalid_login_lock = true
```

기본적으로 3회 이상 패스워드를 틀리면 해당 사용자 계정을 사용할 수 없는데 제니퍼 서버의 invalid_login_count 옵션을 통해서 임계치를 설정할 수 있다.

```
invalid_login_count = 3
```

위와 같은 이유로 사용할 수 없게 된 사용자 계정은 관리자 그룹에 속한 사용자가 **[구성 관리 | 사용자 관리]** 메뉴에서 해당 사용자의 패스워드를 변경해주면 사용할 수 있게 된다.

12.9.2.패스워드 변경과 관련한 설정

악의적인 사용자에 의해서 패스워드가 유추되는 것을 방지하기 위해서 패스워드에 대한 다양한 정책을 설정할 수 있다.

우선 패스워드의 길이를 제한할 수 있다. 제니퍼 서버의 password_length_min 옵션을 통해서 패스워드의 최소 길이를, password_length_max 옵션을 통해서 패스워드의 최대 길이를 설정할 수 있다.

```
password_length_min = 3
password_length_max = 10
```

그리고 패스워드에 대문자, 소문자, 숫자, 특수 문자 등을 몇 개 이상 포함시키도록 설정할 수 있다. 예를 들어서 제니퍼 서버의 password_uppercase_count 옵션을 통해서 패스워드에 포함시켜야 하는 대문자의 최소 숫자를 설정할 수 있다.

```
password_uppercase_count = 3
```

동일하게 소문자는 password_lowercase_count 옵션으로, 숫자는 password_number_count 옵션으로, 특수 문자는 password_specialcase_count 옵션으로 설정하면 된다.

```
password_lowercase_count = 2
password_number_count = 2
password_specialcase_count = 2
```

다음은 패스워드에 사용 가능한 특수 문자이다.

```
`~!@#$%^&*()-_=+[{]}\|;:'",<.>/?
```

제니퍼 서버의 password_specialcase_list 옵션을 통해서 패스워드에 사용 가능한 특수 문자를 임의로 설정할 수 있다.

```
password_specialcase_list = !@#$%?
```

또한 과거에 사용한 패스워드의 재사용을 방지할 수도 있다. 제니퍼 서버의 password_history_count 옵션을 통해서 재사용할 수 없는 과거 패스워드의 숫자를 설정하면 최근 날짜를 기준으로 해당 숫자 만큼의 과거 패스워드는 재사용할 수 없다.

```
password_history_count = 6
```

그리고 패스워드에 동일한 문자를 사용하거나 연속적으로 이어지는 문자를 사용하는 것을 방지할 수 있다. 동일한 문자를 사용하는 것을 방지하려면 제니퍼 서버의 password_enable_repetive_charactor_check 옵션을 true로 설정하고 연속적으로 이어지는 문자를 사용하는 것을 방지하려면 제니퍼 서버의 password_enable_consecutive_charactor_check 옵션을 true로 설정한다.

```
password_enable_repetive_charactor_check = true
password_enable_consecutive_charactor_check = true
```

마지막으로 사용자 아이디나 지정된 단어를 패스워드에 사용하는 것을 방지할 수 있다. 이 기능을 사용하려면 제니퍼 서버의 password_enable_common_word 옵션을 true로 설정한다.

```
password_enable_common_word = true
```

이 옵션을 true로 설정하면 기본적으로 패스워드에 사용자 아이디를 사용하는 것을 방지한다. 지정된 단어의 사용까지 방지하려면 JENNIFER_HOME/server/bin 디렉토리에 password_common_word.txt 파일을 만든 후에 사용할 수 없는 단어를 입력한다. 단어는 줄바꿈으로 구분한다. 관리자 그룹에 속한 사용자가 [구성 관리 | 사용자 관리 | 사용자 관

리] 메뉴에서 다른 사용자의 패스워드를 변경할 때는 위에서 설명한 내용이 적용되지 않는다.

12.10. 게시판 관리

[구성 관리 | 게시판] 메뉴에서 게시판 기능을 사용한다.

12.10.1. 게시판 유형

게시판 유형은 제니퍼 서버의 bbs_type_list 옵션으로 설정한다. 게시판 유형은 [게시판 유형 코드:게시판 유형 이름] 형식을 따르고, 각 게시판 유형은 세미 콜론[;]으로 구분한다.

```
bbs_type_list = biz:Business;rpt:Report;chat:Personal
```

기본으로 제공하는 게시판 유형은 다음과 같다.

표 12-9: 게시판 유형

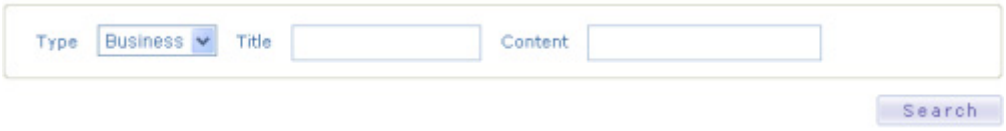
게시판 유형 코드	게시판 유형 이름	설명
biz	Business	업무 관련 게시판
rpt	Report	보고서 관련 게시판
chat	Personal	사적 게시판

새로운 게시판 유형은 제니퍼 서버 bbs_type_list 옵션을 수정하여 추가한다.

12.10.2. 게시판 검색

등록된 게시물은 게시판 유형, 제목, 내용 등을 조건으로 검색할 수 있다. 제목과 내용은 LIKE 검색을 지원한다.

그림 12-8: 게시판 검색 조건



12.10.3.게시물 올리기

새로운 게시물은 다음과 같이 등록한다.

1. **[구성 관리 | 게시판]** 메뉴로 이동한다.
2. 게시물 목록 하단에 있는 **[추가]** 버튼을 클릭한다.
3. 게시판 입력 폼에서 내용을 입력한 후에 하단에 있는 **[쓰기]** 버튼을 클릭한다.

표 12-10: 게시판 입력 필드

필드	설명
게시판 유형	게시판 유형을 선택한다. 게시물을 올린 이후에는 수정할 수 없다.
제목	게시물 제목
처리 담당자	해당 게시물에 대한 처리가 필요한 경우에는 처리 담당자를 지정한다.
처리 요망일	해당 게시물에 대한 처리가 필요한 경우에는 처리 요망일을 지정한다.
상태	[관련 없음], [처리 요청], [진행 중], [처리 완료] 중에서 하나를 선택하여 해당 게시물의 상태를 지정한다 .
내용	게시물 내용
첨부 파일	첨부할 파일을 선택한다. 첨부할 수 있는 파일의 크기와 형식에는 제한이 없다.

Notice: 첨부 파일이 제니퍼 서버에 저장되는 위치는 제니퍼 서버의 upload_directory 옵션으로 설정한다. 기본 디렉토리는 JENNIFER_HOME/data/upload 디렉토리이다.

게시물 변경은 다음과 같이 한다.

1. **[구성 관리 | 게시판]** 메뉴로 이동한다.
2. 게시물 목록에서 수정할 게시물 제목을 클릭한다.
3. 게시물 보기 화면에서 **[수정]** 링크를 클릭한다.
4. 게시판 입력 폼에서 내용을 수정한 후에 하단에 있는 **[쓰기]** 버튼을 클릭한다.

게시물 삭제는 다음과 같이 한다.

1. **[구성 관리 | 게시판]** 메뉴로 이동한다.
2. 게시물 목록에서 삭제할 게시물 제목을 클릭한다.

3. 게시물 보기 화면에서 **[삭제]** 링크를 클릭한다.

Notice: 답글이 있는 경우에는 게시물을 삭제할 수 없다.

12.10.4. 답글 올리기

게시물에 대한 답글 올리기는 다음과 같이 한다.

1. **[구성 관리 | 게시판]** 메뉴로 이동한다.
2. 게시물 목록에서 답글을 올릴 게시물 제목을 클릭한다.
3. 게시물 보기 화면에서 하단에 있는 **[답글 올리기]** 버튼을 클릭한다.
4. 답글 입력 폼에서 내용을 입력한 후에 하단에 있는 **[저장]** 버튼을 클릭한다.

답글 변경은 다음과 같이 한다.

1. **[구성 관리 | 게시판]** 메뉴로 이동한다.
2. 게시물 목록에서 답글을 올릴 게시물 제목을 클릭한다.
3. 게시물 보기 화면에서 수정할 답글 하단에 있는 **[수정]** 링크를 클릭한다.
4. 답글 입력 폼에서 내용을 수정한 후에 하단에 있는 **[저장]** 버튼을 클릭한다.

답글 삭제는 다음과 같이 한다.

1. **[구성 관리 | 게시판]** 메뉴로 이동한다.
2. 게시물 목록에서 답글을 올릴 게시물 제목을 클릭한다.

게시물 보기 화면에서 삭제할 답글 하단에 있는 **[삭제]** 링크를 클릭한다.

12.11. 도메인 구성

하나의 제니퍼 서버가 감당할 수 있는 제니퍼 에이전트의 숫자와 업무 처리량에는 한계가 있다. 시스템 환경이 한계를 초과하는 경우에는 여러 개의 제니퍼 서버를 운영해서 부하를 분산시켜야 한다. 이를 위한 것이 도메인 구성이다.

또한 도메인 구성 여부와 상관없이, 제니퍼 에이전트를 업무 종류에 따라 계층적으로 분류해서 모니터링할 필요성도 있다. 이를 위한 것이 노드 구성이다.

12.11.1.도메인이란?

하나의 제니퍼 서버가 감당할 수 있는 제니퍼 에이전트의 숫자와 업무 처리량에는 한계가 있다. 이 한계를 초과하는 경우에는 여러 개의 제니퍼 서버를 운영해서 부하를 분산시켜야 한다.

Notice: 하나의 제니퍼 서버가 감당할 수 있는 제니퍼 에이전트의 숫자는 50개 정도이고 TPS는 약 1000이다.

그러나 여러 개의 제니퍼 서버를 운영하면 관리 비용은 증가하고 사용자 편의성은 감소하는 단점이 있다. 이런 단점을 해결하기 위해서 제니퍼 4.0에서는 도메인 구성을 통해서 여러 개의 제니퍼 서버를 통합해서 관리하는 통합 사용자 인터페이스를 제공한다.

여기서 도메인은 개별 제니퍼 서버를 의미하고, 도메인 구성은 통합 서버를 통해서 여러 개의 제니퍼 서버를 하나로 통합하는 것을 의미한다.

Notice: 제니퍼 에어전트로 모니터링하는 자바 애플리케이션을 특정 제니퍼 서버에 할당할 때는 우선 업무의 유사성을 고려한다. 그러나 업무의 유사성만을 기준으로 제니퍼 에이전트를 할당하면 제니퍼 서버가 많아질 수 있다. 그러므로 업무의 유사성이 낮더라도 TPS가 높지 않은 제니퍼 에이전트들을 하나의 제니퍼 서버에 모두 할당하는 것을 권장한다.

12.11.2.통합 서버

통합 서버는 여러 개의 제니퍼 서버 중에서 사용자가 웹브라우저를 통해서 접근하는 유일한 서버이다. 따라서 통합 서버도 제니퍼 서버이다. 사용자는 이 통합 서버를 통해서 도메인 구성을 하고 사용자, 권한, 메뉴 등의 설정을 한다.

여러 개의 제니퍼 서버 중에서 통합 서버를 선택할 때는 다음 사항을 고려한다.

- 특정 제니퍼 서버가 자바 애플리케이션을 모니터링하는 역할과 통합 서버의 역할을 동시에 수행 할 수 있다. 통합 서버의 역할만을 하는 제니퍼 서버를 별도로 운영하는 것은 권장하지 않는다.
- 여러 개의 제니퍼 서버에 제니퍼 에이전트와 TPS를 적절히 분배해서 제니퍼 서버의 숫자를 최소한으로 억제한다.
- 통합 서버에는 성능 데이터 수집과 분석 이외에 사용자 인터페이스 처리와 관련한 추가적인 부하가 발생하기 때문에, 부하가 가장 적은 제니퍼 서버를 통합 서버로 선택하는 것을 권장한다.
- REMON을 통해서 많은 데이터를 수집하는 경우에는 REMON 데이터를 여러 개의 제니퍼 서버에서 별도로 수집하는 것보다 하나의 제니퍼 서버에서 수집하는 것을 권장한다. 또한 이런 경우에는 REMON 데이터만을 수집하는 제니퍼 서버를 두고 이를 통합 서버로 사용하는 것을 권장한다.

12.11.3.도메인 관리

도메인 구성과 관련한 작업은 통합 서버로 선택한 제니퍼 서버에서 이루어진다.

도메인을 추가하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 도메인 관리]** 메뉴로 이동한다.
- 도메인 목록 오른쪽 하단의 **[추가]** 버튼을 클릭한다.
- 도메인 입력 폼에 내용을 입력한 후에 하단의 **[저장]** 버튼을 클릭한다.

다음은 도메인 입력 폼의 필드에 대한 설명이다.

표 12-11: 도메인 입력 폼

필드	설명
아이디	해당 제니퍼 서버를 지칭하는 고유한 아이디로, 띄어쓰기 없이 영어와 숫자만을 사용하여 입력한다.
이름	해당 제니퍼 서버의 이름
IP	해당 제니퍼 서버의 IP 주소
HTTP 포트	해당 제니퍼 서버의 HTTP 포트 번호
TCP 포트	해당 제니퍼 서버의 server_tcp_port 옵션으로 설정한 포트 번호
버전	해당 제니퍼 서버의 버전으로 4.0을 입력한다. 제니퍼 4.0 이하의 버전은 도메인을 통해서 통합 할 수 없다.
상태	Inactive로 설정한 제니퍼 서버는 사용하지 않는다.

도메인 추가시 제니퍼 서버 외부 IP를 제니퍼 서버가 설치된 하드웨어에서는 호출할 수 없는 경우가 있다. 이 문제를 해결하려면 IP를 설정할 때 제니퍼 서버 외부 IP와 내부 IP를 슬래시[/]를 구분자로 입력하도록 한다.

```
External_Jennifer_Server_IP/Internal_Jennifer_Server_IP
```

도메인을 추가한 경우에는 싱글 사인온을 위해서 해당 제니퍼 서버의 domain_url 옵션에 통합 서버의 HTTP 주소를 설정한다.

```
domain_url = http://통합서버IP:포트번호
```

Notice: 외부 IP와 내부 IP가 분리되어 있는 경우에는 외부 IP를 사용한다. 통합 서버가 싱글 사인온 서버의 역할도 수행하기 때문에, 통합 서버에 로그인한 사용자는 통합 서버에 도메인으로 등록된 제니퍼 서버가 제공하는 서비스를 별도의 로그인 없이 사용할 수 있다.

Notice: 이 옵션을 설정하기 전에는 도메인으로 등록한 해당 제니퍼 서버가 통합 서버와 싱글 사인온으로 연결되어 있지 않기 때문에 해당 제니퍼 서버에 로그인하여 옵션을 수정하거나 해당 제니퍼 서버의 설정 파일을 직접 수정해야 한다.

도메인을 수정하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 도메인 관리]** 메뉴로 이동한다.
- 도메인 목록 중에서 수정할 도메인의 아이디를 클릭한다.
- 도메인 입력 폼의 내용을 수정한 후에 하단의 **[저장]** 버튼을 클릭한다.

도메인을 삭제하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 도메인 관리]** 메뉴로 이동한다.
- 도메인 목록 중에서 삭제할 도메인의 체크 박스를 선택한다.
- 도메인 목록 하단의 **[삭제]** 버튼을 클릭한다.

도메인의 순서를 변경하는 방법은 다음과 같다.

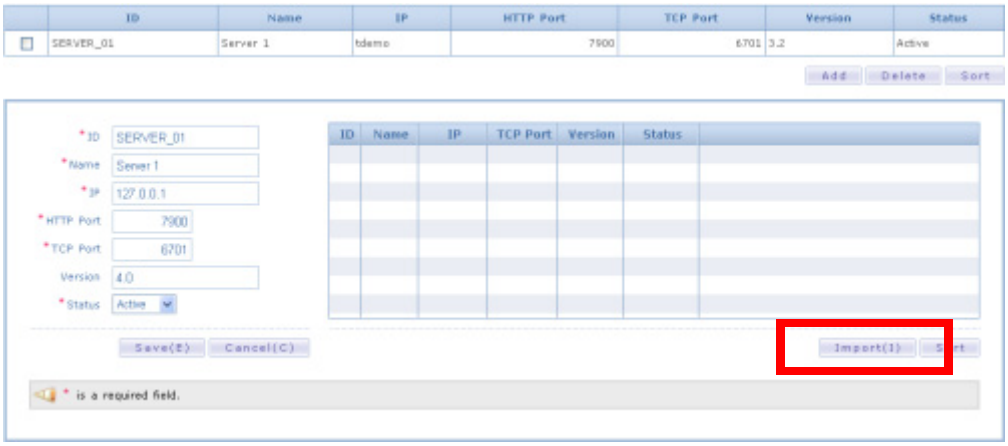
- 통합 서버의 **[구성 관리 | 도메인 관리]** 메뉴로 이동한다.
- 도메인 목록 오른쪽 하단의 **[정렬]** 버튼을 클릭한다.
- 정렬 팝업 창에서 도메인의 순서를 변경한다.

Notice: 도메인 구성을 변경한 후에 통합 서버에 다시 로그인을 해야 변경 사항이 반영된다.

12.11.4.제니퍼 에이전트 불러오기

노드를 구성하거나 제니퍼 에이전트에 이름을 부여하려면 도메인을 구성한 후에 제니퍼 에이전트 불러오기를 수행해야 한다.

그림 12-9: 제니퍼 에이전트 불러오기



제니퍼 에이전트를 불러오는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 도메인 관리]** 메뉴로 이동한다.
- 도메인 목록 중에서 제니퍼 에이전트를 불러 올 도메인의 아이디를 클릭한다.
- 오른쪽 하단 제니퍼 에이전트 목록 아래에 있는 **[불러오기]** 버튼을 클릭한다.

Notice: 도메인 입력폼의 **[취소]** 버튼과 상관없이 **[불러오기]** 버튼을 클릭하는 순간에 관련 데이터가 저장된다.

특정 제니퍼 에이전트의 이름을 변경하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 도메인 관리]** 메뉴로 이동한다.
- 도메인 목록 중에서 이름을 변경할 제니퍼 에이전트가 속해 있는 도메인의 아이디를 클릭한다.
- 오른쪽 하단 제니퍼 에이전트 목록에서 이름을 수정할 제니퍼 에이전트를 선택한 후에 이름 칼럼에 이름을 입력한다.
- 선택한 제니퍼 에이전트 열에 있는 **[저장]** 링크를 클릭한다.

더 이상 사용하지 않는 특정 제니퍼 에이전트를 삭제하는 방법은 다음과 같다.

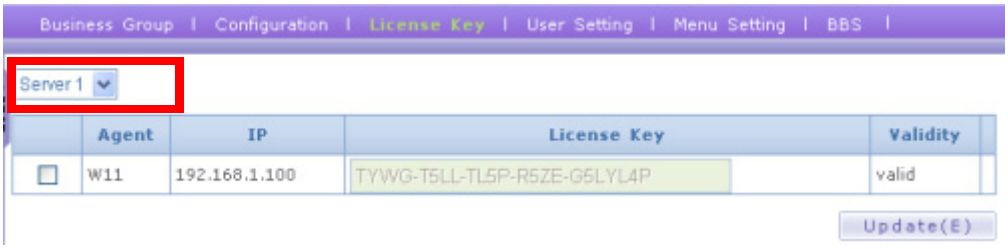
- 통합 서버의 **[구성 관리 | 도메인 관리]** 메뉴로 이동한다.
- 도메인 목록 중에서 삭제할 제니퍼 에이전트가 속해있는 도메인의 아이디를 클릭한다.
- 오른쪽 하단 제니퍼 에이전트 목록에서 삭제할 제니퍼 에이전트를 선택한 후에 해당 열에 있는 **[삭제]** 링크를 클릭한다.

12.11.5.도메인 구성에 따른 사용자 인터페이스의 변화

도메인을 구성한 경우에는 많은 사용자 인터페이스 화면에서 도메인을 명시적으로 선택해 주어야 한다.

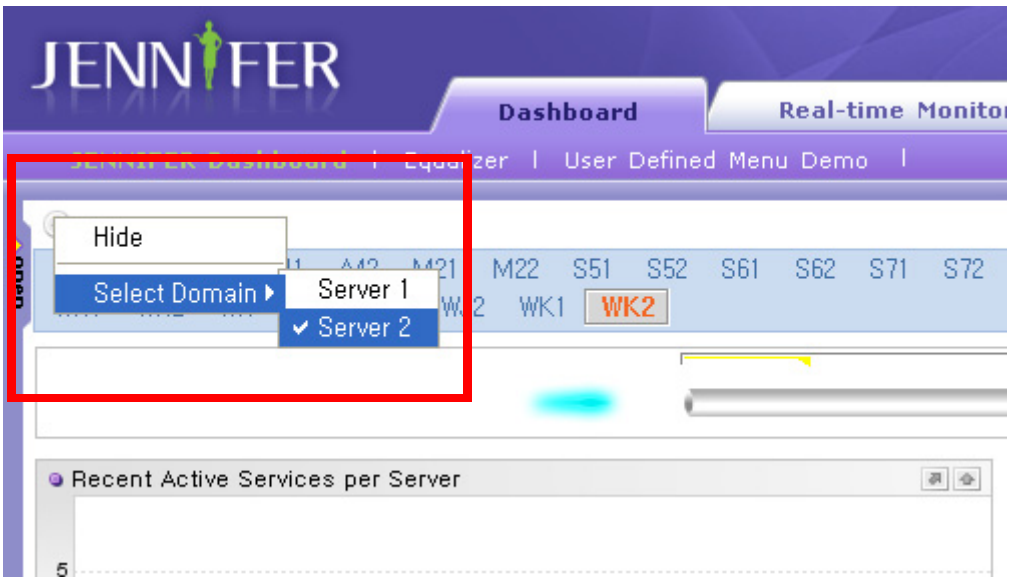
예를 들어, **[구성 관리 | 라이센스키 관리]** 메뉴에서 도메인을 드랍다운 박스를 통해서 선택해야 한다.

그림 12-10:라인센스키 관리 - 도메인 선택



제니퍼 에이전트 선택 영역은 도메인을 기준으로 한다. 다른 도메인에 속하는 제니퍼 에이전트 목록을 확인하는 방법은 다음과 같다.

그림 12-11:제니퍼 에이전트 선택 영역 - 도메인의 선택



- 옵션 아이콘을 클릭한다.
- 컨텍스트 메뉴에서 **[도메인 선택]** 메뉴를 선택하여 특정 도메인을 클릭한다.

12.11.6.노드 구성

12.11.6.1.노드란?

제니퍼 서버가 모니터링하는 제니퍼 에이전트가 많으면, 업무 종류에 따라 트리 형태로 제니퍼 에이전트를 묶어 체계적으로 분류하여 모니터링할 수 있다. 여기서 노드는 일부 제니퍼 에이전트의 묶음이고 노드 구성은 노드를 이용해서 트리 형태로 전체 제니퍼 에이전트를 분류하는 것이다.

예를 들어, 특정 제니퍼 서버가 모니터링하는 제니퍼 에이전트가 W11에서 W19까지 있을 때, 업무 성격을 기준으로 W11에서 W15까지로 [그룹웨어] 노드를 구성하고 W16에서 W19까지로 [HR] 노드를 구성 할 수 있다. 그리고 [그룹웨어] 노드와 [HR] 노드로 [사내시스템] 노드를 구성할 수 있다.

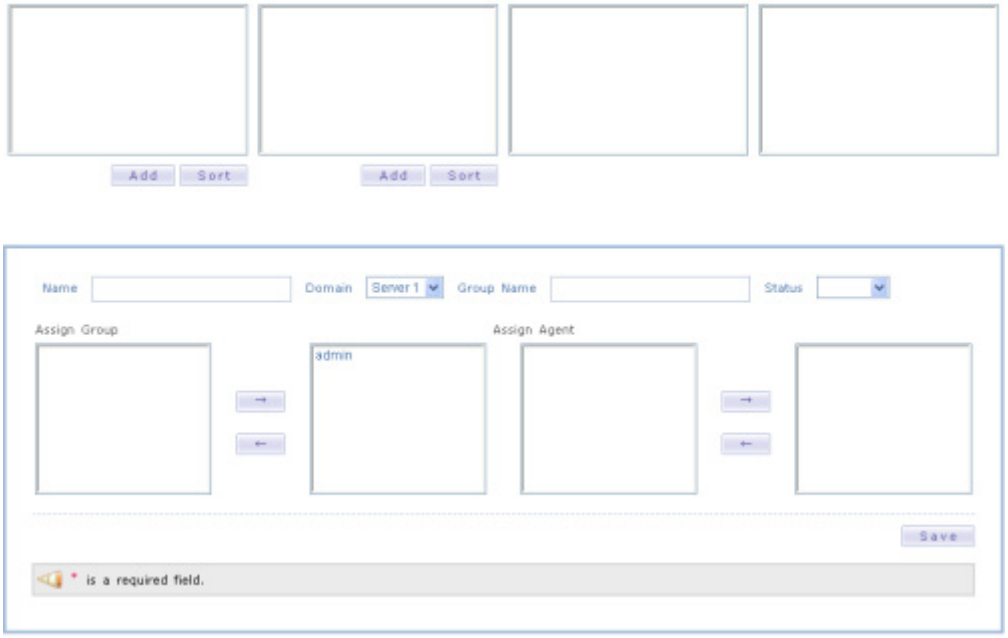
Notice: 노드의 단계는 4단계까지로 제한된다.

12.11.6.2.노드 관리

노드를 구성하려면 우선 도메인을 등록한 후에 제니퍼 에이전트 불러오기를 수행해야 한다.

Notice: 노드를 구성하려면 제니퍼 서버가 하나인 경우에도 도메인을 등록해야 한다. 여기서 등록하는 도메인은 현재 사용하고 있는 제니퍼 서버이다.

그림 12-12:노드 구성 화면



노드를 추가하려면 우선 최상위 노드를 입력해야 한다. 최상위 노드를 입력하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 노드 관리]** 메뉴로 이동한다.
- 첫번째 노드 그룹 박스 아래에 있는 **[추가]** 버튼을 클릭한다.
- 노드 입력 폼에 내용을 입력한 후에 하단의 **[저장]** 버튼을 클릭한다.

다음은 노드 입력 폼의 필드에 대한 설명이다.

표 12-12: 노드 입력 필드

필드	설명
이름	노드 이름
도메인	노드에 할당할 제니퍼 에이전트가 속하는 도메인으로, 최상위 노드와 모든 하위 노드는 동일한 도메인에 속하는 제니퍼 에이전트만으로 설정해야 한다. 도메인은 노드를 생성한 후에는 수정할 수 없다.
그룹명	정확한 동시단말 사용자 수와 방문자 수를 모니터링하려면 해당 노드에 속하는 제니퍼 에이전트들의 제니퍼 에이전트 그룹 아이디를 입력한다.
상태	Inactive로 설정된 노드는 사용하지 않는다.

표 12-12: 노드 입력 필드

필드	설명
그룹 할당	해당 노드를 사용하는 그룹으로, 해당 그룹에 속한 사용자들에게만 해당 노드가 나타난다. 상위 노드에 특정 그룹이 할당되어 있지 않으면, 하위 노드에 해당 그룹이 할당되어 있어도 해당 그룹에 속한 사용자에게 하위 노드가 나타나지 않는다.
에이전트 할당	노드에 속하는 제니퍼 에이전트를 할당한다. 노드가 하위 노드들을 묶는 역할만을 하는 경우에는 제니퍼 에이전트를 할당하지 않아도 된다.

특정 최상위 노드가 제니퍼 서버를 통해서 모니터링하는 모든 제니퍼 에이전트를 포함하고 있다면 그룹명에 TOT를 입력한다. 그리고 일반 노드에 전체 동시단말 사용자 수와 전체 방문자 수가 HTTP 쿠키에 기반한 정확한 값으로 표시되기를 원하면 제니퍼 서버의 agent_group 옵션으로 설정한 제니퍼 에이전트 그룹 아이디를 노드의 그룹명으로 입력한다. 그룹명을 지정하지 않으면 전체 동시단말 사용자 수와 전체 방문자 수는 개별 제니퍼 에이전트의 동시단말 사용자 수와 방문자 수의 단순 합으로 표시된다.

```
agent_group = @01:W11,W12; @02:W13,W14
```

제니퍼 에이전트 그룹 아이디는 @로 시작하고 2자리 숫자로 끝나야 한다. 여러 개의 제니퍼 에이전트 그룹은 세미 콜론(;)을 구분자로 구분하고, 제니퍼 에이전트 그룹 아이디와 해당 제니퍼 에이전트 그룹에 속하는 제니퍼 에이전트 목록은 콜론(:)을 구분자로 구분한다. 그리고 동일한 제니퍼 에이전트 그룹에 속하는 제니퍼 에이전트들은 콤마(,)를 구분자로 구분한다. 이 옵션을 수정하면 제니퍼 서버를 재시작하여야 한다.

Notice: 2개 이상의 제니퍼 서버로 도메인을 구성한 경우에는 제니퍼 서버의 agent_group 옵션은 해당 제니퍼 에이전트가 존재하는 제니퍼 서버에 설정해야 한다. 제니퍼 서버가 다른 제니퍼 에이전트들로 agent_group 옵션을 설정할 수는 없다.

하위 노드를 추가하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 노드 관리]** 메뉴로 이동한다.
- 노드 그룹 박스를 통해서 하위 노드를 추가할 상위 노드를 선택한다.
- 상위 노드 그룹 박스 오른쪽 옆에 있는 노드 그룹 박스의 아래에 있는 **[추가]** 버튼을 클릭한다.
- 노드 입력 폼에 내용을 입력한 후에 하단의 **[저장]** 버튼을 클릭한다.

Notice: 하위 노드에는 상위 노드와 동일한 도메인에 속하는 제니퍼 에이전트만을 할당할 수 있다. 그리고 이미 다른 노드에 할당한 제니퍼 에이전트를 중복 할당할 수는 없다.

노드를 수정하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 노드 관리]** 메뉴로 이동한다.
- 노드 그룹 박스를 통해서 수정할 노드를 선택한다.

- 노드 입력 폼의 내용을 수정한 후에 하단의 **[저장]** 버튼을 클릭한다.

노드를 삭제하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 노드 관리]** 메뉴로 이동한다.
- 노드 그룹 박스를 통해서 삭제할 노드를 선택한다.
- 노드 입력 폼에서 하단의 **[삭제]** 버튼을 클릭한다.

동일한 상위 노드에 속하는 하위 노드들의 순서를 변경하는 방법은 다음과 같다.

- 통합 서버의 **[구성 관리 | 노드 관리]** 메뉴로 이동한다.
- 노드 그룹 박스를 통해서 순서를 변경할 노드들의 상위 노드를 선택한다.
- 순서를 변경할 노드들의 노드 그룹 박스 아래에 있는 **[정렬]** 버튼을 클릭한다.
- 정렬 팝업 창에서 노드의 순서를 변경한다.

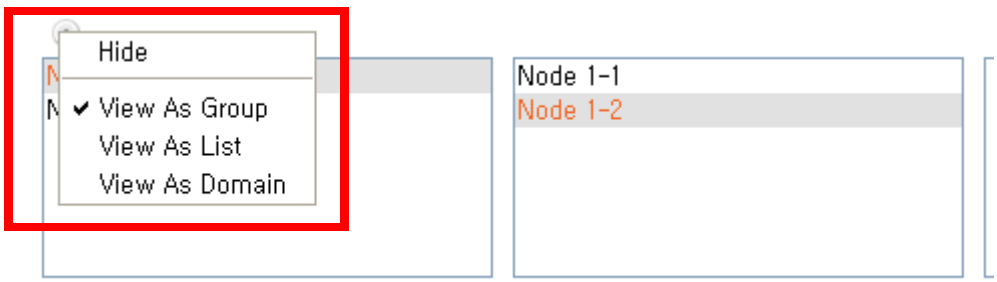
Notice: 노드 구성을 변경한 후에 제니퍼 서버에 다시 로그인을 해야 변경 사항이 반영된다.

12.11.6.3.노드 구성에 따른 사용자 인터페이스의 변화

노드를 구성하면 대부분의 사용자 인터페이스 화면 상단에 존재하는 제니퍼 에이전트 선택 영역이 달라진다.

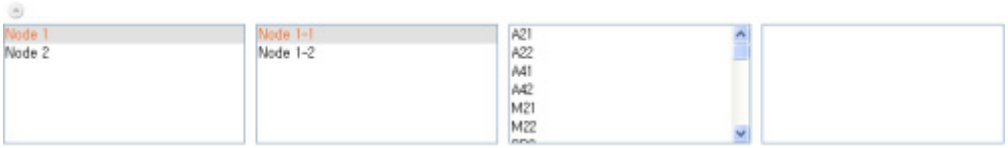
일단 노드를 구성하면 제니퍼 에이전트 선택 영역의 뷰를 다음 3가지 중에서 선택할 수 있다.

그림 12-13:제니퍼 에이전트 선택 영역 - 노드의 선택



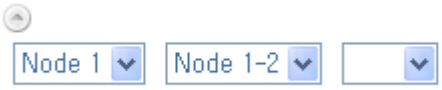
- 그룹으로 보기 - 노드 구성이 노드 그룹 영역으로 나타나는 방식이다.

그림 12-14:제니퍼 에이전트 선택 영역 - 그룹으로 보기



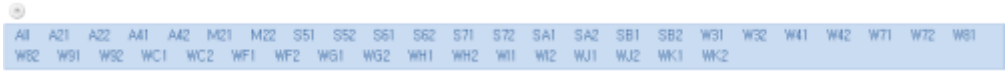
- 리스트로 보기 - 노드 구성이 드랍다운 박스로 나타나는 방식이다.

그림 12-15:제니퍼 에이전트 선택 영역 - 리스트로 보기



- 도메인 별 보기 - 노드 구성과는 상관없이 도메인을 기준으로 특정 도메인에 속한 제니퍼 에이전트를 표시하는 방식이다.

그림 12-16:제니퍼 에이전트 선택 영역 - 도메인 별 보기



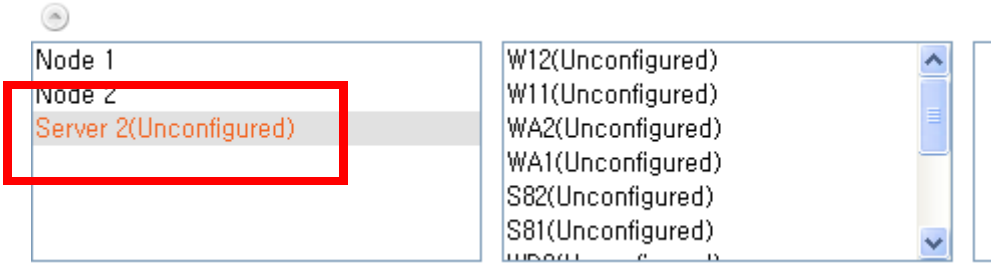
제니퍼 에이전트 선택 영역의 뷰를 변경하는 방법은 다음과 같다.

- 옵션 아이콘을 클릭한다.
- 컨텍스트 메뉴에서 특정 뷰를 선택한다.

노드 그룹 영역이나 드랍다운 박스에서 특정 노드를 선택하면 해당 노드와 모든 하위 노드에 할당한 제니퍼 에이전트를 기준으로 화면이 구성된다.

그런데 새로운 제니퍼 에이전트를 추가한 경우나 실수로 특정 제니퍼 에이전트를 노드에 할당하지 않은 경우와 같이, 노드에 할당되지 않는 제니퍼 에이전트가 있을 수 있다. 이 경우에 admin 그룹에 속한 사용자에게는 [도메인 이름(미설정)]을 이름으로 하는 최상위 노드가 만들어지고, 그 노드에 어떤 노드에도 할당되지 않은 제니퍼 에이전트가 할당된다.

그림 12-17:미할당 제니퍼 에이전트가 있는 경우



12.12. 데이터 관리

제니퍼 서버는 제니퍼 에이전트와 제니퍼 독립 에이전트 등으로부터 수집한 데이터를 파일과 데이터베이스에 저장한다.

12.12.1. 데이터 파일

제니퍼 서버는 데이터베이스에 저장하기에 적합하지 않은 X-View 트랜잭션과 프로파일 데이터 등을 파일에 저장한다.

데이터 파일이 저장되는 디렉토리 위치는 제니퍼 서버의 data_directory 옵션으로 설정한다

```
data_directory = ../../data/file/
```

기본으로 데이터 파일이 저장되는 디렉토리는 JENNIFER_HOME/data/file이며, 아래에 일자별로 하위 디렉토리가 만들어지고 그 일자별 디렉토리에 데이터 파일이 저장된다. 디렉토리의 구조 및 파일의 형식은 임의로 변경될 수 있다.

Notice: 파일 손상이 발생할 수 있기 때문에, 제니퍼 서버의 운영 중에는 data_directory 옵션을 변경하지 않는다. 해당 옵션을 변경하려면 제니퍼 서버를 정지한 후에, 제니퍼 서버의 설정 파일에서 직접 변경하도록 한다.

12.12.2. 데이터베이스

제니퍼 서버는 대부분의 성능 데이터와 구성 데이터 등을 데이터베이스에 저장한다. 그리고 데이터베이스는 성능 데이터를 저장하는 성능 데이터베이스와 구성 데이터를 저장하는 관리자 데이터베이스로 구분된다.

Notice: 기본적으로 제니퍼 서버는 아파치 더비를 데이터베이스로 사용한다.

제니퍼 에이전트 개수가 많거나 서비스 요청률이 높을 때 상용 DBMS를 사용하면 성능 개선 효과를 볼 수 있다. 성능 데이터베이스와 관리자 데이터베이스로 서로 다른 DBMS를 사용할 수 있는데 관리자 데이터베이스가 성능에 미치는 영향은 없기 때문에 성능 데이터베이스에만 상용 DBMS를 사용하고 관리자 데이터베이스에는 아파치 더비를 사용하는 것을 권장한다.

12.12.2.1. 아파치 더비의 사용

아파치 더비 데이터베이스의 위치는 제니퍼 서버의 system.derby.system.home 옵션으로 설정한다.

```
system.derby.system.home = ../../data/database
```

기본적으로 데이터베이스가 저장되는 디렉토리는 JENNIFER_HOME/data/datab ase이다.

제니퍼 서버에서 아파치 더비를 사용하기 위한 JDBC 연결은 JENNIFER_HOME/server/conf/Catalina/localhost/ROOT.xml 파일에 설정되어 있다.

다음은 성능 데이터베이스에 대한 JDBC 설정이다.

```
<Resource name="jdbc/Jennifer" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="org.apache.derby.jdbc.EmbeddedDriver"
          url="jdbc:derby:jennifer;create=true" />
```

다음은 관리자 데이터베이스에 대한 JDBC 설정이다.

```
<Resource name="jdbc/JenniferAdmin" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="org.apache.derby.jdbc.EmbeddedDriver"
          url="jdbc:derby:jenniferadm;create=true" />
```

별도의 자바 애플리케이션에서 아파치 더비 데이터베이스에 접근하려면, 제니퍼 서버에 다음 옵션들을 추가한 후에 제니퍼 서버를 재시작한다.

```
system.derby.drda.startNetworkServer = true
system.derby.drda.portNumber = 1527
system.derby.drda.host = 192.168.0.1
```

Notice: system.derby.drda.host 옵션에는 127.0.0.1이나 localhost가 아닌 실제 IP 주소를 입력해야 한다.

아파치 더비를 사용하는 경우에는 Reorg 스케줄러를 이용해서 테이블 스페이스의 낭비를 최소화하고 있다.

```
time_actor_2 = com.javaservice.jennifer.server.timeactor.ReorgActor 03
```

기본적으로 매일 오전 3시에 테이블 스페이스를 정리한다.

Notice: 기본적으로 데이터베이스로 아파치 더비를 사용하도록 설정되어 있기 때문에, 아파치 더비를 사용하는 경우에는 설정을 변경할 필요가 없다.

12.12.2.2.오라클의 사용

오라클을 사용하는 경우에는 오라클 9i 이상을 사용해야 하고 JDBC 드라이버는 오라클 10g 이상을 사용해야 한다. 그리고 데이터베이스는 UTF-8 인코딩 형식으로 만들어져야 한다.

제니퍼 서버에서 오라클을 사용하기 위한 JDBC 연결은 JENNIFER_HOME/server/conf/Catalina/localhost/ROOT.xml 파일에 설정되어 있다.

다음은 성능 데이터베이스에 대한 JDBC 설정이다.

```
<Resource name="jdbc/Jennifer" auth="Container"
            type="javax.sql.DataSource" maxActive="100" maxIdle="30"
            maxWait="10000" username="jennifer" password="jennifer"
            driverClassName="com.ibm.db2.jcc.DB2Driver"
            url="jdbc:db2://120.0.0.1:50000/jennifer" />
```

다음은 관리자 데이터베이스에 대한 JDBC 설정이다.

```
<Resource name="jdbc/JenniferAdmin" auth="Container"
            type="javax.sql.DataSource" maxActive="100" maxIdle="30"
            maxWait="10000" username="jennifer" password="jennifer"
            driverClassName="com.ibm.db2.jcc.DB2Driver"
            url="jdbc:db2://120.0.0.1:50000/jenniferadm" />
```

username, password, url 등의 속성은 사용할 오라클 데이터베이스에 맞게 변경한다.

그리고 JDBC 드라이버 라이브러리(ojdbc14.jar)를 JENNIFER_HOME/server/common/lib 디렉토리에 복사한다.

Notice: 앞의 설정 변경을 적용하기 위해서는 제니퍼 서버의 재시작이 필요하다.

12.12.2.3.IBM DB2의 사용

IBM DB2를 사용하는 경우에는 DB2 9.5 이상을 사용해야 하고 데이터베이스는 UTF-8 인코딩 형식으로 만들어져야 한다. 그리고 페이지 사이즈를 32K 이상으로 만들어야 한다.

제니퍼 서버에서 IBM DB2를 사용하기 위한 JDBC 연결은 JENNIFER_HOME/server/conf/Catalina/localhost/ROOT.xml 파일에 설정되어 있다.

다음은 성능 데이터베이스에 대한 JDBC 설정이다.

```
<Resource name="jdbc/Jennifer" auth="Container"
           type="javax.sql.DataSource" maxActive="100" maxIdle="30"
           maxWait="10000" username="jennifer" password="jennifer"
           driverClassName="com.ibm.db2.jcc.DB2Driver"
           url="jdbc:db2://120.0.0.1:50000/jennifer" />
```

다음은 관리자 데이터베이스에 대한 JDBC 설정이다.

```
<Resource name="jdbc/JenniferAdmin" auth="Container"
          type="javax.sql.DataSource" maxActive="100" maxIdle="30"
          maxWait="10000" username="jennifer" password="jennifer"
          driverClassName="com.ibm.db2.jcc.DB2Driver"
          url="jdbc:db2://120.0.0.1:50000/jenniferadm" />
```

username, password, url 등의 속성은 사용할 IBM DB2에 맞게 변경한다.

그리고 JDBC 드라이버 라이브러리(db2jcc.jar, db2jcc_license_cu.jar)를 JENNIFER_HOME/server/common/lib 디렉토리에 복사한다.

Notice: 앞의 설정 변경을 적용하기 위해서는 제니퍼 서버의 재시작이 필요하다.

12.12.3.쿼리 수행기

[통계 분석 | 쿼리 수행기] 메뉴를 통해서 제니퍼 데이터베이스에 대해 임의의 SQL을 실행할 수 있다.

그림 12-18:쿼리 수행기

1. 테이블 탐색기 - 왼쪽의 테이블 탐색기를 통해서 성능 데이터베이스와 관리자 데이터베이스 테이블 스키마를 확인할 수 있다.

2. 테이블 탐색기 새로 고침 - 테이블 탐색기 내용을 새로 고치려면 해당 버튼을 클릭한다.

제니퍼 데이터베이스에 대해서 임의의 SQL을 수행하는 방법은 다음과 같다.

- SQL 필드에 쿼리를 입력한다.
- 최대 결과 수 제한, 도메인, 관리자 데이터베이스 여부, 결과 형식 등을 설정한다.
- 오른쪽 하단에 있는 [실행] 버튼을 클릭한다.

SQL의 결과는 하단에 나타나며 각 필드에 대한 설명은 다음과 같다.

표 12-13: SQL 수행기 필드

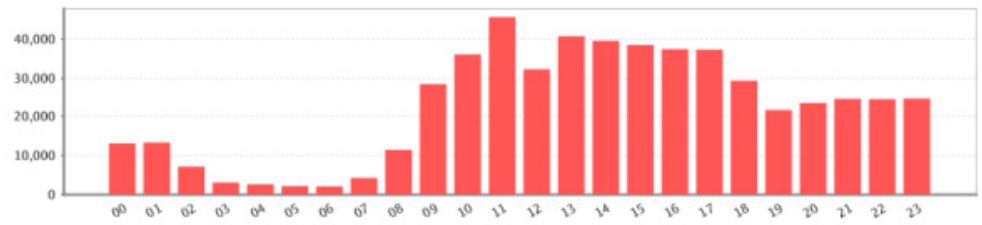
필드	설명
SQL	여러 개의 SQL을 동시에 수행하려면 세미 콜론[;]을 구분자 사용하여 여러 SQL을 입력한다. --로 시작하는 줄은 주석으로 간주된다.
최대 결과 수 제한	SELECT 절을 실행했을 때 반환되는 레코드 숫자의 최대 크기를 설정한다.
도메인	도메인을 선택한다.
관리자 데이터베이스 여부	해당 체크 박스를 선택하면 관리자 데이터베이스에 대해서 SQL을 수행하고, 선택하지 않으면 성능 데이터베이스에 대해서 SQL을 수행한다.
결과 형식	SQL의 결과 형식을 선택한다.

결과 형식으로 테이블을 선택하면 HTML 테이블로 데이터가 표시되고, 엑셀을 선택하면 CSV 형식의 파일로 데이터가 다운로드된다.

결과 형식으로 막대 차트를 선택하면 막대 차트가 표시된다. 예를 들어, 다음은 시간당 방문자 수를 구하는 SQL이다. SELECT 절의 첫번째 칼럼이 X 좌표의 값이 되고 두번째 칼럼부터는 Y 좌표의 값이 된다. 그리고 칼럼의 이름을 범례로 사용한다.

```
SELECT LOG_HH, SUM(VISIT_HOUR) AS VISIT
  FROM PERF_X_01
 WHERE LOG_DT = '20081001'
   AND AGENT_ID = 'TOT'
 GROUP BY LOG_HH
```

그림 12-19:시간당 방문자 수



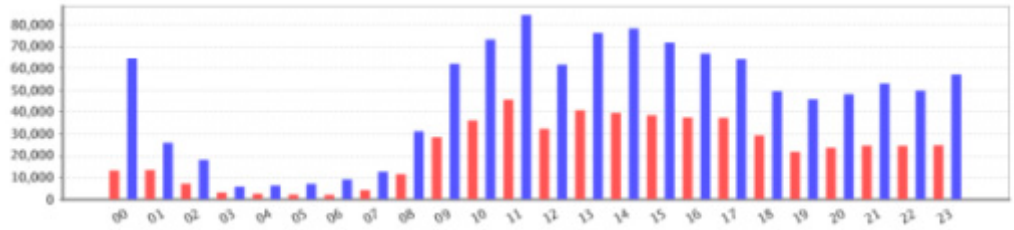
차트의 크기는 제니퍼 서버의 report_image_width 옵션과 report_image_height 옵션으로 설정한다.

```
report_image_width = 800
report_image_height = 200
```

동일 시간대에 대해서 여러 데이터를 막대 차트로 표시하려면 SELECT 절에 칼럼을 추가한다. 예를 들어, 방문자 수와 호출 건수를 막대 차트에 표시하려면 다음 SQL을 사용한다.

```
SELECT LOG_HH, SUM(VISIT_HOUR) AS VISIT, SUM(HIT) AS HIT
  FROM PERF_X_01
 WHERE LOG_DT = '20081001'
   AND AGENT_ID = 'TOT'
 GROUP BY LOG_HH
```

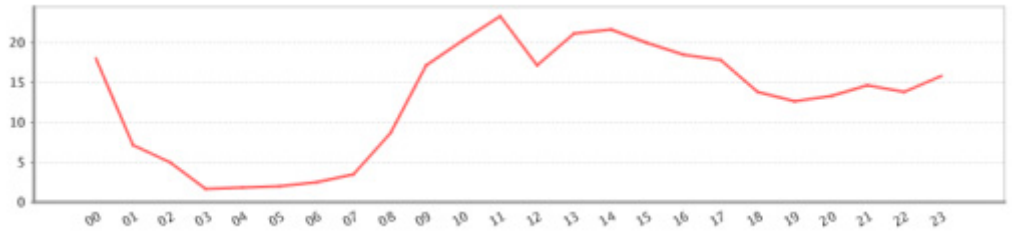
그림 12-20:시간당 방문자 수와 호출 건 수



결과 형식으로 라인 차트를 선택하면 라인 차트가 표시된다. 예를 들어, 다음은 시간당 서비스 요청률을 구하는 SQL이다. SELECT 절의 첫번째 칼럼이 X 좌표의 값이 되고 두번째 칼럼부터는 Y 좌표의 값이 된다. 그리고 칼럼의 이름을 범례로 사용한다.

```
SELECT LOG_HH, AVG(ARRIVAL_RATE) AS ARRIVAL_RATE
  FROM PERF_X_01
 WHERE LOG_DT = '20081001'
   AND AGENT_ID = 'TOT'
 GROUP BY LOG_HH
```

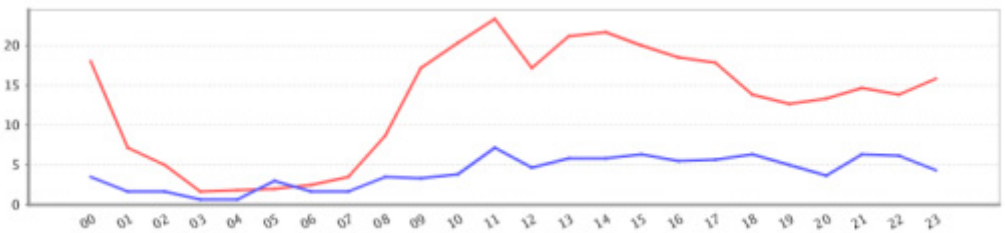
그림 12-21:시간당 서비스 요청률



동일 시간대에 대해서 여러 데이터를 라인 차트로 표시하려면 SELECT 절에 칼럼을 추가한다. 예를 들어, 서비스 요청률과 액티브 서비스를 라인 차트에 표시하려면 다음 SQL을 사용한다.

```
SELECT LOG_HH, AVG(ARRIVAL_RATE) AS ARRIVAL_RATE,
        AVG(ACTIVE_SERVICE) AS ACTIVE_SERVICE
  FROM PERF_X_01
 WHERE LOG_DT = '20081001'
   AND AGENT_ID = 'TOT'
 GROUP BY LOG_HH
```

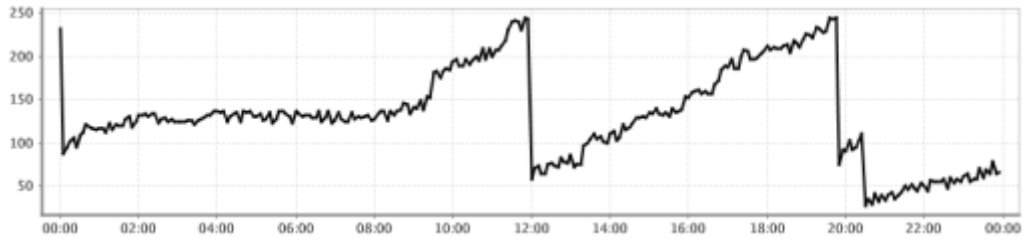
그림 12-22:시간당 서비스 요청률과 액티브 서비스



결과 형식으로 시계열 차트를 선택하면 시계열 차트가 표시된다. 예를 들어, 다음은 5분당 자바 힙 메모리 사용량을 구하는 SQL이다. SELECT 절의 첫번째 칼럼이 X 좌표의 값이 되고 두번째 칼럼부터는 Y 좌표의 값이 된다. 그리고 칼럼의 이름을 범례로 사용한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM, HEAP_USED
  FROM PERF_X_01
 WHERE LOG_DT ='20081001'
   AND AGENT_ID = 'A21'
```

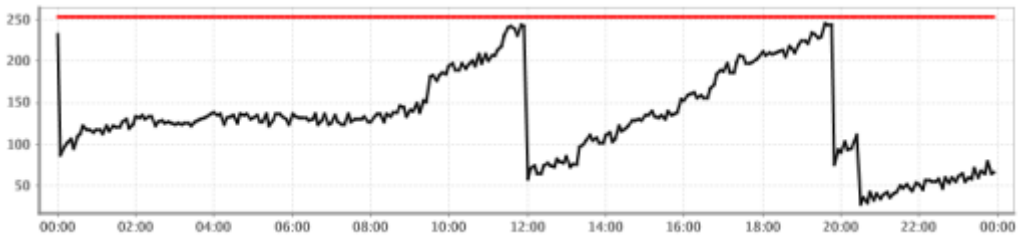
그림 12-23:5분당 자바 힙 메모리 사용량



동일 시간대에 대해서 여러 데이터를 시계열 차트로 표시하려면 SELECT 절에 칼럼을 추가한다. 예를 들어, 자바 힙 메모리 사용량과 자바 힙 전체 메모리를 시계열 차트에 표시하려면 다음 SQL을 사용한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM, HEAP_USED, HEAP_TOTAL
  FROM PERF_X_01
 WHERE LOG_DT ='20080801'
   AND AGENT_ID = 'A21'
```

그림 12-24:5분당 자바 힙 메모리 사용량과 자바 힙 전체 메모리



Notice: 데이터의 숫자가 적은 경우(시간당으로 표시하는 경우)에는 결과 형식으로 막대 차트나 라인 차트를 선택하고, 데이터의 숫자가 많은 경우(5분 단위로 표시하는 경우)에는 결과 형식으로 시계열 차트를 선택한다.

12.12.4. 데이터 보관 및 삭제

데이터베이스에 대한 검색 시간 향상을 위해서 제니퍼는 SummaryActor 스케줄러를 이용해서 성능 데이터에 대한 요약 데이터를 매일 주기적으로 생성한다. 제니퍼 서버의 SummaryActor에 대한 설정은 다음과 같다.

```
time_actor_1=com.javaservice.jennifer.server.timeactor.SummaryActor 01
```

첫번째 파라미터 01이 지칭하는 것처럼 SummaryActor는 매일 오전 1시에 수행된다. SummaryActor가 정상적으로 수행되지 않은 경우에는 **[툴바 영역 | 통계 데이터 요약]**을 통해서 특정 날짜에 대해서 SummaryActor를 실행 할 수도 있다.

Notice: [툴바 영역 | 통계 데이터 요약]을 통해서 특정 날짜에 대해서 SummaryActor를 반복적으로 수행해도 데이터가 중복으로 계산되지는 않는다.

제니퍼 에이전트의 숫자와 업무 처리량에 비례해서 제니퍼가 수집하는 성능 데이터의 양이 증가한다. 그래서 디스크 사용을 최적화하기 위해서 기본적으로 30일 이전의 데이터는

CleanerActor를 통해서 삭제한다. 제니퍼 서버의 CleanerActor에 대한 설정은 다음과 같다. .

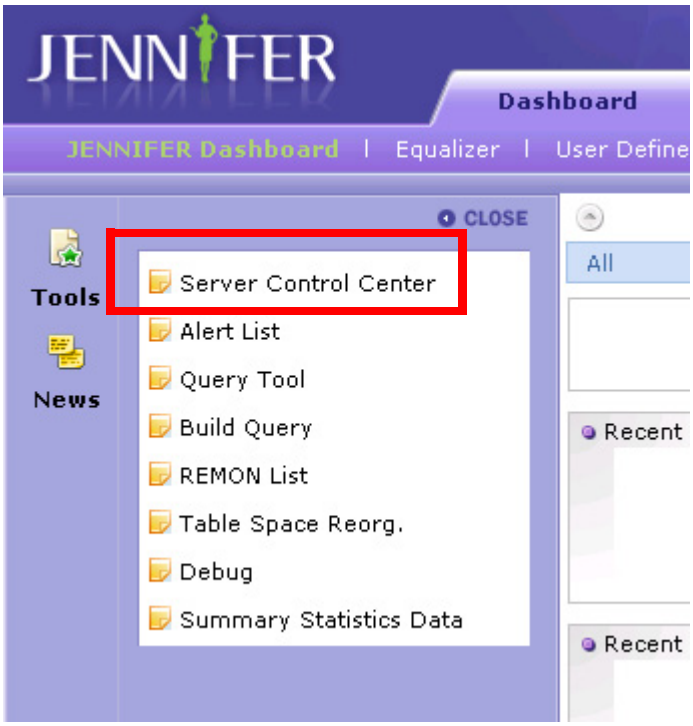
```
time_actor_03
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 ALERT MONTH 1
time_actor_04
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 ERROR  MONTH 1
time_actor_05
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 APPL MONTH 1
time_actor_06
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 SQL MONTH 1
time_actor_07
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 TX MONTH 1
time_actor_08
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 PERF MONTH 1
time_actor_09
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 FILE MONTH 1
time_actor_11
  =com.javaservice.jennifer.server.timeactor.CleanerActor
   02 REMON DAY 7
```

기본 설정에 따르면 ALERT_01~31, ERRORS_10M_01~31, APPL_10M_01~31, SQLS_10M_01~31, TX_10M_01~31, PERF_X_01~31 등의 테이블에서는 한달 전의 데이터를 삭제하고 REMON으로부터 수집한 데이터를 저장한 테이블에서는 7일 이전의 데이터를 삭제한다. 그리고 한달 전의 데이터 파일을 삭제한다. 이 작업은 오전 2시에 순차적으로 이루어진다.

12.12.5.데이터 백업 및 복구

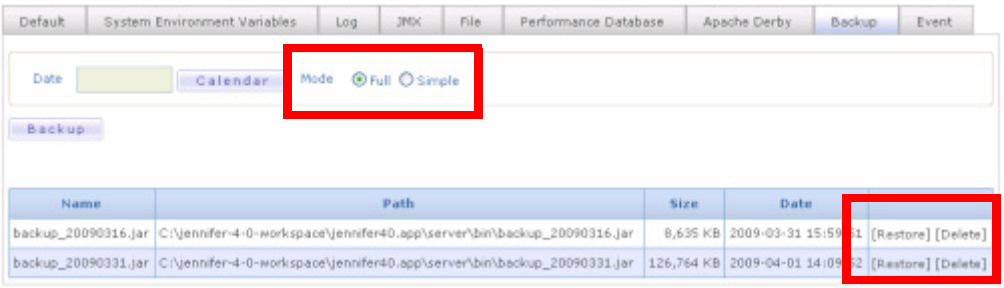
데이터 파일과 데이터베이스 내용을 일자별로 백업하고 복구할 수 있다. 도구 영역에서 **[도구 | Server Control Center]** 메뉴를 클릭하면 팝업 창이 나타난다.

그림 12-25:Server Control Center



팝업 창에서 백업 탭을 선택하면 나타나는 화면에서 데이터 백업과 복구 작업을 수행할 수 있다.

그림 12-26:데이터 백업과 복구



데이터 백업 파일 위치는 제니퍼 서버의 backup_root 옵션으로 설정한다. 기본으로는 해당 옵션이 설정되어 있지 않은데 이 경우에는 JENNIFER_HOME/server/bin 디렉토리에 데이터 백업 파일이 저장된다.

```
backup_root = /home/jennifer/backup
```

Notice: 동일한 하드웨어에서 복수의 제니퍼 서버를 운영하는 경우에는 각 제니퍼 서버별로 backup_root 옵션을 명시적으로 지정해서 백업 디렉토리를 다르게 설정해야 한다.

데이터를 백업하는 방법은 다음과 같다.

1. 데이터를 백업할 날짜를 선택한다. 금일 데이터는 백업할 수 없다.
2. 모드를 선택한다. Full 모드를 선택하면 데이터 파일과 데이터베이스 내용을 모두 백업한다. 반면에 Simple 모드를 선택하면 데이터베이스 내용만을 백업한다.
3. **[백업]** 버튼을 클릭하면 데이터 백업이 진행된다. 데이터 양에 따라서 긴 시간이 소요될 수 있다. 데이터 백업이 완료되면 하단 데이터 백업 파일 목록에 해당 날짜에 대한 데이터 백업 파일이 추가된다.

데이터를 복원하는 방법은 다음과 같다.

1. 모드를 선택한다. Full 모드를 선택하면 데이터 파일과 데이터베이스 내용을 모두 복원한다. 반면에 Simple 모드를 선택하면 데이터베이스 내용만을 복원한다.
2. 데이터 백업 파일 목록에서 데이터를 복원할 날짜에 해당하는 **[복원]** 링크를 클릭하면 데이터 복원이 진행된다. 데이터 양에 따라서 긴 시간이 소요될 수 있다.

Notice: 제니퍼 서버 CPU 사용률 증가를 최소화하기 위해서 동시에 백업 혹은 복원 작업을 수행할 수 없다. 예를 들어, 2009-04-02 날짜 데이터를 백업하고 있는 중간에는 다른 날짜 데이터를 백업하거나 복원할 수 없다. 마찬가지로 2009-04-04 날짜 데이터를 복원하고 있는 중간에는 다른 날짜 데이터를 백업하거나 복원할 수 없다.

데이터 복원을 완료한 후에 복원된 데이터를 정상적으로 확인하려면 다음 작업을 수행한다.

1. [자바 플러그인 임시 파일 지우기]를 참조하여 임시 파일을 삭제한다.
2. 재로그인을 한다.

데이터 백업 파일을 삭제하는 방법은 다음과 같다.

1. 데이터 백업 파일 목록에서 데이터 백업 파일을 삭제할 날짜에 해당하는 **[삭제]** 링크를 클릭하면 해당 데이터 백업 파일이 삭제된다.
2. 데이터 백업 파일 삭제가 완료되면 하단 데이터 백업 파일 목록에 해당 날짜에 대한 데이터 백업 파일이 삭제된다.

그리고 BackupActor 스케줄러를 이용해서 주기적으로 데이터를 백업할 수도 있다. 제니퍼 서버의 BackupActor에 대한 설정은 다음과 같다.

```
time_actor_15 = com.javaservice.jennifer.server.timeactor.BackupActor
0220 FULL
```

Notice: 기본적으로 BackupActor 스케줄러는 설정되어 있지 않다.

이 설정에 따르면 02시 20분에 FULL 모드로 데이터 백업이 수행된다. FULL 대신에 SIMPLE을 모드로 지정할 수 있다.

Notice: 데이터 백업과 복원은 제니퍼 서버 CPU 사용률을 증가시킨다. 따라서 동일한 하드웨어에서 복수의 제니퍼 서버를 운영하는 경우에는 BackupActor 스케줄러가 동작하는 시간을 다르게 설정하는 것을 권장한다.

12.13. 테이블 스키마

테이블 칼럼의 데이터 유형은 아파치 더비를 기준으로 작성하였다.

12.13.1. 성능 데이터베이스

12.13.1.1. AGENT

제니퍼 에이전트 정보를 저장하는 테이블이다.

표 12-14: AGENT 테이블

칼럼	유형	제약 조건	설명
ID	VARCHAR(20)	PK, NOT NULL	제니퍼 에이전트 아이디
NAME	VARCHAR(50)		제니퍼 에이전트 이름
IP	VARCHAR(50)	NOT NULL	IP 주소
TCP_PORT	INTEGER	NOT NULL	제니퍼 에이전트의 agent_tcp_port 옵션으로 설정한 포트 번호
DOMAIN_ID	VARCHAR(10)	NOT NULL	도메인 아이디
NODE_ID	BIGINT		노드 아이디
VERSION	VARCHAR(50)		제니퍼 에이전트 버전
TYPE	INTEGER		유형
STATUS	INTEGER	NOT NULL	상태를 구분하는 코드로, 1은 사용을, 0은 비사용을 의미한다.
SORT	INTEGER		정렬 기준

12.13.1.2.ALERT_01~31

경보 데이터를 저장하는 일자별 테이블이다.

Notice: 데이터베이스에 저장되는 성능 데이터 중에서 양이 많은 경우에는 일자별로 구분하여 저장한다. 이 때 사용하는 테이블을 일자별 테이블이라고 한다. 따라서 10월 3일에 수집한 경보 데이터는 ALERT_03 테이블에 저장되고, 10월 17일에 수집한 경보 데이터는 ALERT_17 테이블에 저장된다. ALERT_01~31은 일자별 테이블을 지칭하기 위한 개념적인 이름으로 해당 테이블이 실제로 존재하지는 않는다.

표 12-15: ALERT_01~31 테이블

칼럼	유형	제약 조건	설명
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
LOG_SS	CHAR(2)		초(SS)
TYPE	CHAR(1)		경보 유형을 구분하는 코드로, C는 심각을, E는 에러를, W는 경고를 의미한다.
ALERT_NM	VARCHAR(100)		경보 이름
ALERT_MSG	VARCHAR(32672)		경보 메시지
AGENT_ID	VARCHAR(20)		경보가 발생한 제니퍼 에이전트 아이디

12.13.1.3.APPLS

애플리케이션 이름을 저장하는 테이블이다.

표 12-16: APPLS 테이블

칼럼	유형	제약 조건	설명
HASH	INTEGER	PK, NOT NULL	애플리케이션 이름 해쉬 값
NAME	VARCHAR(512)		애플리케이션 이름
FLAG	CHAR(1)		데이터 정상 여부를 구분하는 코드로, T는 정상을, F는 비정상을 의미한다.

12.13.1.4.APPL_10M_01~31

10분 동안의 애플리케이션 처리 현황 통계 데이터를 저장하는 일자별 테이블이다.

표 12-17: APPL_10M_01~31 테이블

칼럼	유형	제약 조건	설명
APPL_HASH	INTEGER		애플리케이션 이름 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
CNT	INTEGER		호출 건수
FAIL_CNT	INTEGER		실패 건수
ELAPSED_SUM	BIGINT		응답 시간의 합
ELAPSED2_SUM	BIGINT		표준 편차 계산을 위한 응답 시간 제곱의 합
ELAPSED_MIN	BIGINT		최소 응답 시간
ELAPSED_MAX	BIGINT		최대 응답 시간
CPU_SUM	BIGINT		CPU 사용 시간의 합
TPMC_SUM	BIGINT		TPMC 합

12.13.1.5.APPL_SQLTX_01~31

10분 동안의 애플리케이션과 SQL 혹은 외부 트랜잭션 사이의 매핑 데이터를 저장하는 일자별 테이블이다.

표 12-18: APPL_SQLTX_01~31 테이블

칼럼	유형	제약 조건	설명
APPL_HASH	INTEGER		애플리케이션 이름 해쉬 값
SQLTX_HASH	INTEGER		SQL 혹은 외부 트랜잭션 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
TYPE	CHAR(1)		유형를 구분하는 코드로, S는 SQL을, T는 외부 트랜잭션을 의미한다.
CNT	INTEGER		실행 건수
ELAPSED_SUM	BIGINT		응답 시간의 합

표 12-18: APPL_SQLTX_01~31 테이블

칼럼	유형	제약 조건	설명
ELAPSED2_SUM	BIGINT		표준 편차 계산을 위한 응답 시간 제곱의 합
ELAPSED_MIN	BIGINT		최소 응답 시간
ELAPSED_MAX	BIGINT		최대 응답 시간

12.13.1.6.BIZ_GROUP

애플리케이션 비즈니스 그룹 데이터를 저장하는 테이블이다.

표 12-19: BIZ_GROUP 테이블

칼럼	유형	제약 조건	설명
UUID	BIGINT	PK, NOT NULL	고유 아이디로 자동 생성된다.
ID	VARCHAR(256)	NOT NULL	아이디
DIV	VARCHAR(100)	NOT NULL	구분명
NAME	VARCHAR(100)	NOT NULL	비지니스 그룹 이름
DOMAIN_ID	VARCHAR(10)		도메인 아이디
AGENT_ID	VARCHAR(300)		제니퍼 에이전트 아이디
APP	VARCHAR(1000)		애플리케이션 목록으로 줄바꿈(\n)을 구분자로 구분한다.
TX	VARCHAR(1000)		외부 트랜잭션 목록으로 줄바꿈(\n)을 구분자로 구분한다.
TYPE	INTEGER		유형
STATUS	INTEGER		상태로 1은 Active를, 0은 Inactive를 의미한다.
SORT	INTEGER		정렬 기준

12.13.1.7.DOMAIN

도메인 데이터를 저장하는 테이블이다.

표 12-20: DOMAIN 테이블

칼럼	유형	제약 조건	설명
ID	VARCHAR(10)	PK, NOT NULL	도메인 아이디
NAME	VARCHAR(20)	NOT NULL	도메인 이름

표 12-20: DOMAIN 테이블

칼럼	유형	제약 조건	설명
IP	VARCHAR(50)	NOT NULL	제니퍼 서버의 IP 주소
HTTP_PORT	INTEGER	NOT NULL	제니퍼 서버의 HTTP 포트 번호
TCP_PORT	INTEGER	NOT NULL	제니퍼 서버의 server_tcp_port 옵션으로 설정한 포트 번호
VERSION	VARCHAR(50)		제니퍼 서버 버전
TYPE	INTEGER		유형
STATUS	INTEGER	NOT NULL	상태를 구분하는 코드로, 1은 Active를, 0은 Inactive를 의미한다.
SORT	INTEGER		정렬 기준

12.13.1.8.ERRORS

예외의 이름을 저장하는 테이블이다.

표 12-21: ERRORS 테이블

칼럼	유형	제약 조건	설명
HASH	INTEGER	PK, NOT NULL	예외 해쉬 값
NAME	VARCHAR(32672)		예외 이름
FLAG	CHAR(1)		데이터 정상 여부를 구분하는 코드로, T는 정상을, F는 비정상을 의미한다.

12.13.1.9.ERRORS_10M_01~31

10분 동안의 예외 현황 통계 데이터를 저장하는 일자별 테이블이다.

표 12-22: ERRORS_10M_01~31 테이블

칼럼	유형	제약 조건	설명
ERR_HASH	INTEGER		예외 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)

표 12-22: ERRORS_10M_01~31 테이블

칼럼	유형	제약 조건	설명
TYPE	CHAR(1)		경보 유형를 구분하는 코드로, C는 심각을, E는 에러를, W는 경고를 의미한다.
CNT	INTEGER		예외 건수
ERROR_CD	INTEGER		예외 코드
ERROR_NM	VARCHAR(100)		예외 이름

12.13.1.10.ERR_APPL_01~31

예외와 애플리케이션 사이의 매핑 데이터를 저장하는 테이블이다.

표 12-23: ERR_APPL_01~31 테이블

칼럼	유형	제약 조건	설명
ERR_HASH	INTEGER		예외 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
CNT	INTEGER		예외 건수
APPL_HASH	INTEGER		예외가 발생한 애플리케이션 이름 해쉬 값
MSG_HASH	INTEGER		예외 메시지 해쉬 값
TX_UUID	BIGINT		X-View 트랜잭션 아이디

12.13.1.11.GROUP_NODE

그룹과 노드 사이의 매핑 정보를 저장하는 테이블이다.

표 12-24: GROUP_NODE 테이블

칼럼	유형	제약 조건	설명
GROUP_ID	VARCHAR(20)	PK, NOT NULL	그룹 아이디
NODE_ID	BIGINT	PK, NOT NULL	노드 아이디

12.13.1.12.HOST

WMOND가 실행되고 있는 컴퓨터의 정보를 저장하는 테이블이다.

표 12-25: HOST 테이블

칼럼	유형	제약 조건	설명
HOST_ID	VARCHAR(20)		호스트 아이디
HOST_NM	VARCHAR(50)		호스트 이름
IPADDR	VARCHAR(20)	NOT NULL	호스트 IP 주소

12.13.1.13.METHODS

메소드의 이름을 저장하는 테이블이다.

표 12-26: METHODS 테이블

칼럼	유형	제약 조건	설명
HASH	INTEGER	PK, NOT NULL	메소드 이름 해쉬 값
NAME	VARCHAR(32672)		메소드 이름
FLAG	CHAR(1)		데이터 정상 여부를 구분하는 코드로, T는 정상을, F는 비정상을 의미한다.

12.13.1.14.NODE

노드 데이터를 저장하는 테이블이다.

표 12-27: NODE 테이블

칼럼	유형	제약 조건	설명
ID	BIGINT	PK, NOT NULL	노드 아이디
NAME	VARCHAR(50)	NOT NULL	노드 이름
GROUP_ID	VARCHAR(2)		제니퍼 서버의 agent_group 옵션으로 설정한 그룹 아이디
PARENT_ID	BIGINT		상위 노드 아이디
DOMAIN_ID	VARCHAR(10)		도메인 아이디
TYPE	INTEGER		유형
STATUS	INTEGER	NOT NULL	상태를 구분하는 코드로, 1은 Active를, 0은 Inactive를 의미한다.

표 12-27: NODE 테이블

칼럼	유형	제약 조건	설명
SORT	INTEGER		정렬 기준

12.13.1.15.PERF_HOST

WMOND를 통해서 수집한 CPU 데이터를 저장하는 테이블이다.

제니퍼 서버의 perf_host_update_interval 옵션을 통해서 저장 주기를 변경할 수 있다. 기본 값은 300000이고 단위는 밀리 세컨드이다.

```
perf_host_update_interval = 300000
```

표 12-28: PERF_HOST 테이블

칼럼	유형	제약 조건	설명
IPADDR	VARCHAR(20)		호스트 IP 주소
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
LAST_TM	CHAR(6)		마지막 시간(HHMM)
CPU_USER	REAL		사용자 애플리케이션 CPU 사용률
CPU_SYS	REAL		커널 CPU 사용률
CPU_NICE	REAL		NICE CPU 사용률
CPU_IOWAIT	REAL		IO CPU 사용률
CPU_IDLE	REAL		IDLE CPU 사용률

12.13.1.16.PERF_X_01~31

5분 동안의 일반 성능 데이터를 저장하는 일자별 테이블이다.

제니퍼 서버의 perf_x_update_interval 옵션을 통해서 저장 주기를 변경할 수 있다. 기본 값은 300000이고 단위는 밀리 세컨드이다.

```
perf_x_update_interval = 300000
```

표 12-29: PERF_X_01~31 테이블

칼럼	유형	제약 조건	설명
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
LAST_TM	CHAR(6)		마지막 시간(HHMM)
CONCURRENT_USER	REAL		동시단말 사용자 수
ACTIVE_USER	REAL		액티브 사용자 수
ACTIVE_SERVICE	REAL		액티브 서비스 개수
ARRIVAL_RATE	REAL		서비스 요청률
SERVICE_RATE	REAL		서비스 처리율
RESPONSE_TIME	REAL		평균 응답 시간
THINKTIME	REAL		대기 시간
HEAP_TOTAL	REAL		자바 힙 전체 메모리
HEAP_USED	REAL		자바 힙 메모리 사용량
SYS_CPU	REAL		시스템 CPU 사용률
JVM_CPU	REAL		자바 프로세스 CPU 사용률
SYS_MEM_USED	REAL		시스템 메모리 사용량
JVM_NAT_MEM	REAL		자바 프로세스 메모리 사용량
JDBC_ACTIVE	REAL		사용 중인 JDBC 커넥션 개수
JDBC_ALLOC	REAL		할당된 JDBC 커넥션 개수
JDBC_IDLE	REAL		대기 중인 JDBC 커넥션 개수
HIT	INTEGER		호출 건수
VISIT_DAY	INTEGER		방문자 수(일 기준)
VISIT_HOUR	INTEGER		방문자 수(시간 기준)

12.13.1.17.SQLS

SQL을 저장하는 테이블이다.

표 12-30: SQLS 테이블

칼럼	유형	제약 조건	설명
HASH	INTEGER	PK, NOT NULL	SQL 해쉬 값
NAME	CLOB		SQL
FLAG	CHAR(1)		데이터 정상 여부를 구분하는 코드로, T는 정상을, F는 비정상을 의미한다.

12.13.1.18.SQLS_10M_01~31

10분 동안의 SQL의 처리 현황 통계 데이터를 저장하는 일자별 테이블이다.

표 12-31: SQLS_10M_01~31 테이블

칼럼	유형	제약 조건	설명
SQL_HASH	INTEGER		SQL 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
CNT	INTEGER		실행 건수
ELAPSED_SUM	BIGINT		응답 시간의 합
ELAPSED2_SUM	BIGINT		표준 편차 계산을 위한 응답 시간 제곱의 합
ELAPSED_MIN	BIGINT		최소 응답 시간
ELAPSED_MAX	BIGINT		최대 응답 시간
PARAM1	VARCHAR(32672)		상수 파라미터
PARAM2	VARCHAR(32672)		바인딩 파라미터

12.13.1.19.S_ALERT

1시간 동안의 경보 현황 통계 데이터를 저장하는 통계 테이블이다.

표 12-32: S_ALERT

칼럼	유형	제약 조건	설명
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)

표 12-32: S_ALERT

칼럼	유형	제약 조건	설명
TYPE	CHAR(1)		경보 유형을 구분하는 코드로, C는 심각을, E는 에러를, W는 경고를 의미한다.
ALERT_NM	VARCHAR(100)		경보 이름
AGENT_ID	VARCHAR(20)		경보가 발생한 제니퍼 에이전트 아이디
CNT	INTEGER		경보 건수

12.13.1.20.S_APPL

일일 동안의 애플리케이션 처리 현황 통계 데이터를 저장하는 통계 테이블이다.

Notice: 이름이 S_로 시작하는 테이블은 SummaryActor를 통해서 관리되는 통계 테이블이다.

표 12-33: S_APPL 테이블

칼럼	유형	제약 조건	설명
APPL_HASH	INTEGER		애플리케이션 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
CNT	INTEGER		호출 건수
FAIL_CNT	INTEGER		실패 건수
ELAPSED_SUM	BIGINT		응답 시간의 합
ELAPSED2_SUM	BIGINT		표준 편차 계산을 위한 응답 시간 제곱의 합
ELAPSED_MIN	BIGINT		최소 응답 시간
ELAPSED_MAX	BIGINT		최대 응답 시간
CPU_SUM	BIGINT		CPU 사용 시간의 합
TPMC_SUM	BIGINT		TPMC 합

12.13.1.21.S_ERRORS

1시간 동안의 예외 현황 통계 데이터를 저장하는 통계 테이블이다.

표 12-34: S_ERRORS 테이블

칼럼	유형	제약 조건	설명
ERR_HASH	INTEGER		예외 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
TYPE	CHAR(1)		경보 유형을 구분하는 코드로, C는 심각을, E는 에러를, W는 경고를 의미한다.
ERROR_CD	INTEGER		예외 코드
ERROR_NM	VARCHAR(100)		예외 이름
CNT	INTEGER		에외 건수

12.13.1.22.S_PERF_X

1시간 동안의 일반 성능 데이터를 저장하는 통계 테이블이다.

표 12-35: S_PERF_X 테이블

칼럼	유형	제약 조건	설명
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
CONCURRENT_USER	REAL		동시단말 사용자 수
ACTIVE_USER	REAL		액티브 사용자 수
ACTIVE_SERVICE	REAL		액티브 서비스 개수
ARRIVAL_RATE	REAL		서비스 요청률
SERVICE_RATE	REAL		서비스 처리율
RESPONSE_TIME	REAL		평균 응답 시간
THINKTIME	REAL		대기 시간
HEAP_TOTAL	REAL		자바 힙 전체 메모리
HEAP_USED	REAL		자바 힙 메모리 사용량
SYS_CPU	REAL		시스템 CPU 사용율
JVM_CPU	REAL		자바 프로세스 CPU 사용율
SYS_MEM_USED	REAL		시스템 메모리 사용량

표 12-35: S_PERF_X 테이블

칼럼	유형	제약 조건	설명
JVM_NAT_MEM	REAL		자바 프로세스 메모리 사용량
JDBC_ACTIVE	REAL		작업 중인 JDBC 커넥션 개수
JDBC_ALLOC	REAL		할당된 JDBC 커넥션 개수
JDBC_IDLE	REAL		대기 중인 JDBC 커넥션 개수
HIT	INTEGER		호출 건수
VISIT_DAY	INTEGER		방문자 수(일 기준)
VISIT_HOUR	INTEGER		방문자 수(시간 기준)

12.13.1.23.S_SQLS

1시간 동안의 SQL 처리 현황 통계 데이터를 저장하는 통계 테이블이다.

표 12-36: S_SQLS 테이블

칼럼	유형	제약 조건	설명
SQL_HASH	INTEGER		SQL 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
CNT	INTEGER		실행 건수
ELAPSED_SUM	BIGINT		응답 시간의 합
ELAPSED2_SUM	BIGINT		표준 편차 계산을 위한 응답 시간 제곱의 합
ELAPSED_MIN	BIGINT		최소 응답 시간
ELAPSED_MAX	BIGINT		최대 응답 시간

12.13.1.24.S_TX

일일 동안의 외부 트랜잭션의 처리 현황 통계 데이터를 저장하는 통계 테이블이다.

표 12-37: S_TX 테이블

칼럼	유형	제약 조건	설명
TX_HASH	INTEGER		외부 트랜잭션 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)

표 12-37: S_TX 테이블

칼럼	유형	제약 조건	설명
CNT	INTEGER		실행 건수
ELAPSED_SUM	BIGINT		응답 시간의 합
ELAPSED2_SUM	BIGINT		표준 편차 계산을 위한 응답 시간 제곱의 합
ELAPSED_MIN	BIGINT		최소 응답 시간
ELAPSED_MAX	BIGINT		최대 응답 시간

12.13.1.25.TXNAMES

외부 트랜잭션 이름을 저장하는 테이블이다.

표 12-38: TXNAMES 테이블

칼럼	유형	제약 조건	설명
HASH	INTEGER	PK, NOT NULL	외부 트랜잭션 해쉬 값
NAME	VARCHAR(32672)		외부 트랜잭션 이름
FLAG	CHAR(1)		데이터 정상 여부를 구분하는 코드로, T는 정상을, F는 비정상을 의미한다.

12.13.1.26.TX_10M_01~31

10분 동안의 외부 트랜잭션의 처리 현황 데이터를 저장하는 일자별 테이블이다.

표 12-39: TX_10M_01~31 테이블

칼럼	유형	제약 조건	설명
TX_HASH	INTEGER		외부 트랜잭션 해쉬 값
AGENT_ID	VARCHAR(20)		제니퍼 에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
CNT	INTEGER		실행 건수
ELAPSED_SUM	BIGINT		응답 시간의 합
ELAPSED2_SUM	BIGINT		표준 편차 계산을 위한 응답 시간 제곱의 합
ELAPSED_MIN	BIGINT		최소 응답 시간

표 12-39: TX_10M_01~31 테이블

칼럼	유형	제약 조건	설명
ELAPSED_MAX	BIGINT		최대 응답 시간

12.13.2.관리자 데이터베이스

12.13.2.1.ADF

사용자 정의 대시보드를 구성하는 차트와 관련한 데이터를 저장하는 테이블이다.

표 12-40: ADF 테이블

칼럼	유형	제약 조건	설명
UUID	VARCHAR(50)	PK, NOT NULL	고유 아이디로 자동 생성된다.
VIEW_ID	VARCHAR(10)	NOT NULL	사용자 정의 대시보드의 메뉴 아이디
CHART_ID	VARCHAR(50)	NOT NULL	차트 아이디
LEFT_POSITION	INTEGER		차트의 X 축 좌표로, 0에 가까울수록 차트가 왼쪽에 위치한다.
TOP_POSITION	INTEGER		차트의 Y 축 좌표로, 0에 가까울수록 차트가 위쪽에 위치한다.
WIDTH	INTEGER		차트의 넓이
HEIGHT	INTEGER		차트의 높이
CONFIG	VARCHAR(1000)		차트 옵션

12.13.2.2.AUTH

권한 데이터를 저장하는 테이블이다.

표 12-41: AUTH 테이블

칼럼	유형	제약 조건	설명
AUTH_ID	VARCHAR(20)	PK, NOT NULL	권한 아이디
AUTH_NM	VARCHAR(200)		권한 이름
AUTH_DESC	VARCHAR(2000)		권한에 대한 설명
CONFIG	VARCHAR(1000)		권한 설정 정보
IS_FIXED	CHAR(1)		사용하지 않는 칼럼

12.13.2.3.CONTENTS

게시판 데이터를 저장하는 테이블이다.

표 12-42: CONTENTS 테이블

칼럼	유형	제약 조건	설명
BBS_ID	VARCHAR(20)	NOT NULL	게시판 유형
THREAD_ID	INTEGER	NOT NULL	게시물 아이디
PARENT_ID	INTEGER		답변을 단 게시물 아이디
TITLE	VARCHAR(500)		제목
BODY	CLOB		내용
DOC_TYPE	CHAR(1)		사용하지 않는 칼럼
HIT_CNT	INTEGER		조회 건수
USER_ID	VARCHAR(20)		게시물을 등록한 사용자 아이디
USER_NM	VARCHAR(50)		사용하지 않는 칼럼
PASSWORD	VARCHAR(20)		사용하지 않는 칼럼
STATUS_CD	CHAR(1)		상태를 구분하는 코드로, 0은 관련 없음, 1은 처리 요청, 2는 진행 중, 3은 처리 완료를 의미한다.
ASSIGNED_USER	VARCHAR(50)		처리 담당자
NEXT_CHECK_DT	VARCHAR(10)		처리 요망일
CREATE_DT	VARCHAR(10)		게시물 생성 날짜
CREATE_TM	VARCHAR(8)		게시물 생성 시간
UPDATE_DT	VARCHAR(10)		게시물 수정 날짜
UPDATE_TM	VARCHAR(8)		게시물 수정 시간

12.13.2.4.EVENT_LOG

이벤트 로그 데이터를 저장하는 테이블이다.

표 12-43: EVENT_LOG 테이블

칼럼	유형	제약 조건	설명
UUID	BIGINT	PK, NOT NULL	고유 아이디로 자동 생성된다.
ID	VARCHAR(20)	NOT NULL	이벤트 로그 아이디
EVENT_LOG_DESC	VARCHAR(2000)		이벤트 로그 설명

표 12-43: EVENT_LOG 테이블

칼럼	유형	제약 조건	설명
USER_ID	VARCHAR(20)		사용자 아이디
CLIENT_IP	VARCHAR(50)		클라이언트 IP
CREATE_DT	CHAR(10)		생성 날짜(YYYYMMDD)
CREATE_TM	CHAR(8)		생성 시간(HHMMSS)

12.13.2.5.GROUP_AUTH

그룹과 권한 사이의 매핑 데이터를 저장하는 테이블이다.

표 12-44: GROUP_AUTH 테이블

칼럼	유형	제약 조건	설명
AUTH_ID	VARCHAR(20)	PK, NOT NULL	권한 아이디
GROUP_ID	VARCHAR(20)	PK, NOT NULL	그룹 아이디

12.13.2.6.GROUP_MENU

그룹과 메뉴 사이의 매핑 데이터를 저장하는 테이블이다.

표 12-45: GROUP_MENU 테이블

칼럼	유형	제약 조건	설명
GROUP_ID	VARCHAR(20)	PK, NOT NULL	그룹 아이디
MENU_ID	INTEGER	PK, NOT NULL	메뉴 아이디

12.13.2.7.LOGIN_USER

사용자 데이터를 저장하는 테이블이다.

표 12-46: LOGIN_USER 테이블

칼럼	유형	제약 조건	설명
USER_ID	VARCHAR(20)	PK, NOT NULL	사용자 아이디
GROUP_ID	VARCHAR(20)	NOT NULL	사용자가 속한 그룹 아이디
PASSWD	VARCHAR(50)		패스워드
NAME	VARCHAR(100)		사용자 이름

표 12-46: LOGIN_USER 테이블

칼럼	유형	제약 조건	설명
EMAIL	VARCHAR(100)		이메일
COMPANY	VARCHAR(100)		회사
DEPT	VARCHAR(100)		부서
JOBTITLE	VARCHAR(100)		직책
PHONE	VARCHAR(50)		전화 번호
CELLPHONE	VARCHAR(50)		핸드폰
LAST_IP	VARCHAR(20)		마지막에 로그인한 IP 주소
LAST_TM	CHAR(8)		마지막 로그인 시간
LAST_DT	CHAR(10)		마지막 로그인 날짜
LOGIN_CNT	INTEGER		로그인 횟수
DEFAULT_MENU_ID	INTEGER		초기 메뉴 아이디
CONFIG	VARCHAR(1000)		사용자 옵션

12.13.2.8.MENU

메뉴 데이터를 저장하는 테이블이다.

표 12-47: MENU 테이블

칼럼	유형	제약 조건	설명
MENU_ID	INTEGER	PK, NOT NULL	메뉴 아이디
PARENT_ID	INTEGER	NOT NULL	상위 메뉴 아이디
MENU_NM_CD	VARCHAR(1000)		메뉴 다국어 이름 코드
MENU_MN	VARCHAR(100)		메뉴 이름
MENU_ORDER	INTEGER		메뉴 순서
TYPE	CHAR(1)		메뉴 유형을 구분하는 코드로, U는 URL 링크 메뉴를, O는 URL 팝업 메뉴를, P는 사용자 정의 대시보드 메뉴를, S는 리포트 메뉴를 의미한다.
IS_FIXED	CHAR(1)		사용하지 않는 칼럼
MENU_PROP	VARCHAR(200)		URL
DEFAULT_CHILD_ID	INTEGER		기본 하위 메뉴
COLUMN_CNT	INTEGER		사용하지 않는 칼럼

표 12-47: MENU 테이블

칼럼	유형	제약 조건	설명
DEFAULT_HEIGHT	INTEGER		사용하지 않는 칼럼
DEFAULT_WIDTH	INTEGER		사용하지 않는 칼럼
CONFIG	VARCHAR(1000)		메뉴 옵션

12.13.2.9.MESSAGE

다국어 메시지 데이터를 저장하는 테이블이다.

표 12-48: MESSAGE 테이블

칼럼	유형	제약 조건	설명
ID	BIGINT	PK, NOT NULL	고유한 아이디로 자동 생성된다.
MSG_KEY	VARCHAR(50)	NOT NULL	메시지 키
VALUE	VARCHAR(1000)	NOT NULL	메시지 값
LOCALE	VARCHAR(5)	NOT NULL	언어
STATUS	INTEGER		상태를 구분하는 코드로, 1은 사용을 0은 비사용을 의미한다.

12.13.2.10.PASSWORD_HISTORY

패스워드 이력 데이터를 저장하는 테이블이다.

표 12-49: PASSWORD_HISTORY 테이블

칼럼	유형	제약 조건	설명
UUID	BIGINT	PK, NOT NULL	고유한 아이디로 자동 생성된다.
PASSWD	VARCHAR(50)	NOT NULL	패스워드
USER_ID	VARCHAR(20)	NOT NULL	사용자 아이디
CREATE_DT	CHAR(10)		생성 날짜(YYYYMMDD)
CREATE_TM	CHAR(8)		생성 시간(HHMMSS)

12.13.2.11.REPORT_TMPL

리포트 템플릿 데이터를 저장하는 테이블이다.

표 12-50: REPORT_TMPL 테이블

칼럼	유형	제약 조건	설명
TMPL_ID	INTEGER	PK, NOT NULL	리포트 템플릿 아이디
TMPL_NM	VARCHAR(200)		리포트 템플릿 제목
USER_ID	VARCHAR(20)		사용자 아이디
USER_NM	VARCHAR(50)		사용자 이름
DEFAULT_PARAM	VARCHAR(2000)		기본 파라미터
HEADER	VARCHAR(1000)		머리글
FOOTER	VARCHAR(1000)		바닥글
COPYRIGHT	VARCHAR(1000)		저작권
CREATE_DT	CHAR(8)		생성 날짜
CREATE_TM	CHAR(6)		생성 시간

12.13.2.12.TMPL_ITEM

리포트 템플릿 아이템 데이터를 저장하는 테이블이다.

표 12-51: TMPL_ITEM 테이블

칼럼	유형	제약 조건	설명
ITEM_ID	INTEGER	NOT NULL	아이템 아이디
ITEM_NM	VARCHAR(500)	NOT NULL	아이템 이름
ITEM_TYPE	VARCHAR(20)		아이템 유형
ITEM_PROP	VARCHAR(32672)		아이템 파라미터
TMPL_ID	INTEGER	NOT NULL	리포트 템플릿 아이디
CHAPTER	INTEGER		Chapter
SECTION	INTEGER		Section
TITLE	VARCHAR(500)		ITEM_NM 칼럼과 동일한 데이터가 입력됨
HEADING_TEXT	VARCHAR(32672)		머리글
TAIL_TEXT	CLOB		사용하지 않는 칼럼
LOOPING_KEY	VARCHAR(32672)		반복 파라미터
RESERVED_PROP	VARCHAR(32672)		사용하지 않는 칼럼
SQL_QUERY	VARCHAR(32672)		SQL 쿼리
SQL_PARAM	VARCHAR(2000)		사용하지 않는 칼럼

12.13.2.13.UPLOAD_FILE

게시판에서 업로드한 파일 데이터를 저장하는 테이블이다.

표 12-52: UPLOAD_FILE 테이블

칼럼	유형	제약 조건	설명
FILENAME	VARCHAR(500)	NOT NULL	업로드한 파일의 이름
THREAD_ID	INTEGER	NOT NULL	게시판 아이디
SUBDIR	VARCHAR(20)		업로드한 파일이 저장된 디렉토리
FILE_SZ	BIGINT		업로드한 파일의 크기

12.13.2.14.USER_AUTH

사용자와 권한 사이의 매핑 데이터를 저장하는 테이블이다.

표 12-53: USER_AUTH 테이블

칼럼	유형	제약 조건	설명
AUTH_ID	VARCHAR(20)	PK, NOT NULL	권한 아이디
USER_ID	VARCHAR(20)	PK, NOT NULL	사용자 아이디

12.13.2.15.USER_GROUP

그룹 데이터를 저장하는 테이블이다.

표 12-54: USER_GROUP 테이블

칼럼	유형	제약 조건	설명
GROUP_ID	VARCHAR(20)	PK, NOT NULL	그룹 아이디
GROUP_NM	VARCHAR(100)		그룹 이름
DEFAULT_MENU_ID	INTEGER		그룹 초기 메뉴 아이디
GROUP_DESC	VARCHAR(2000)		그룹에 대한 설명
AGENT_LIST	VARCHAR(2000)		해당 그룹에 속한 사용자가 모니터링할 수 있는 제니퍼 에이전트 아이디로 콤마(,)를 구분자로 구분한다.

12.13.2.16.USER_MENU

사용자와 메뉴 사이의 매핑 데이터를 저장하는 테이블이다.

표 12-55: USER_MENU 테이블

칼럼	유형	제약 조건	설명
USER_ID	VARCHAR(20)	PK, NOT NULL	사용자 아이디
MENU_ID	INTEGER	PK, NOT NULL	메뉴 아이디

12.13.2.17.USER_PREFERENCE

사용자별 설정 데이터를 저장하는 테이블이다.

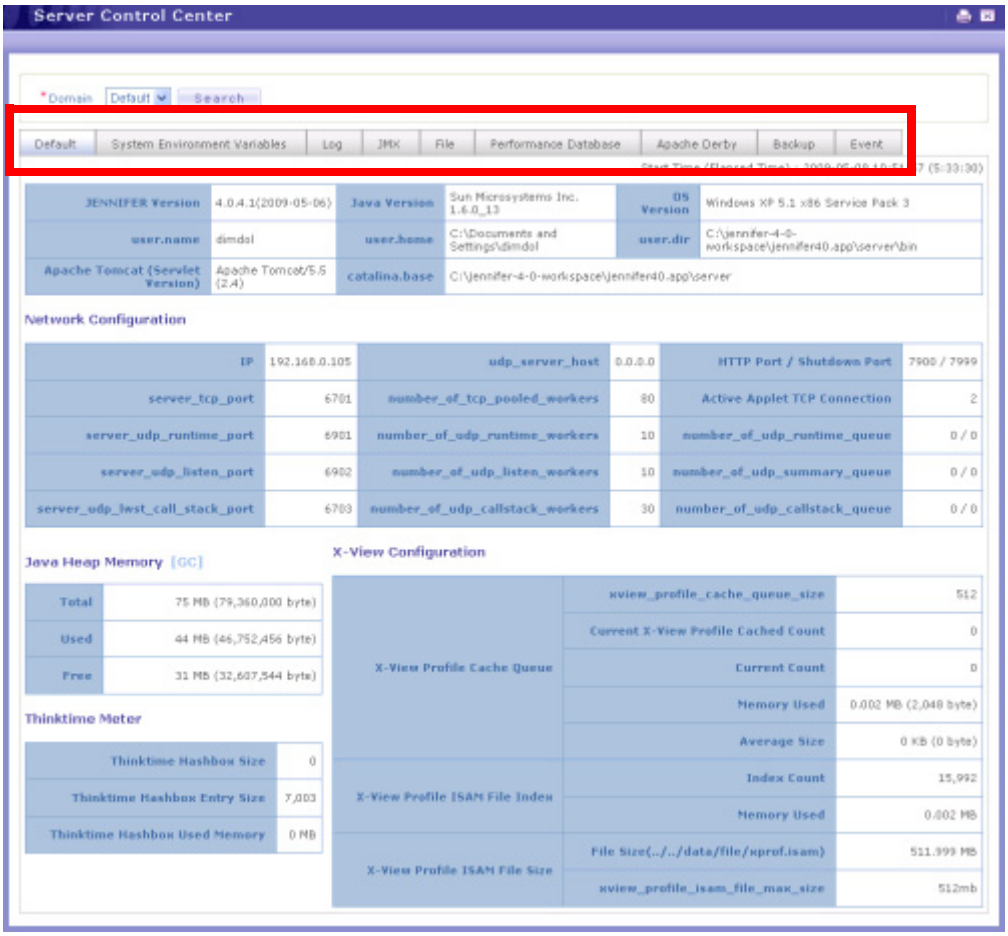
표 12-56: USER_PREFERENCE 테이블

칼럼	유형	제약 조건	설명
NAME	VARCHAR(50)	PK, NOT NULL	설정 이름
VALUE	VARCHAR(255)	NOT NULL	설정 값
USER_ID	VARCHAR(20)	PK, NOT NULL	사용자 아이디

12.14.Server Control Center

Server Control Center는 제니퍼 서버 상태를 확인하고 관리 작업을 수행할 수 있는 팝업 창으로 **[툴바 영역 | 도구 | Server Control Center]** 메뉴를 통해서 접근한다.

그림 12-27:Server Control Center



Server Control Center는 9개 탭으로 구성되어 있다.

12.14.1.기본

기본 탭에서 확인할 수 있는 정보는 다음과 같다.

- 제니퍼 서버, 자바, 운영 체계 등에 대한 기본 정보를 확인할 수 있다.
- 제니퍼 서버 네트워크 설정을 확인할 수 있다.
- 제니퍼 서버 자바 힙 메모리 사용량을 확인할 수 있다. 여기서 제니퍼 서버에 대한 자바 가비지 콜렉션을 수행하려면 **[GC]** 버튼을 클릭한다.
- 대기 시간을 계산하는데 사용하는 정보를 확인할 수 있다.
- X-View 관련 설정 및 처리 현황을 확인할 수 있다.

12.14.2.시스템 환경 변수

시스템 환경 변수 탭에서 확인할 수 있는 정보는 다음과 같다.

- 자바 환경 변수
- 제니퍼 서버 옵션
- 서블릿 컨텍스트 정보

12.14.3.로그

로그 탭에는 제니퍼 서버 로그 파일 목록이 나타난다. 이 목록에서 로그 파일을 다운로드 하거나 삭제할 수 있다.

Notice: 오늘 날짜 로그 파일은 삭제할 수 없다.

12.14.4.JMX

JMX 탭에서 확인할 수 있는 정보는 다음과 같다.

- 자바 JMX 데이터
- 아파치 톰켓 JMX 데이터

목록에서 JMX 이름을 클릭하면 상세 정보를 확인할 수 있다.

Notice: 자바 1.5 이상으로 제니퍼 서버를 설치한 경우에만 이 탭이 표시된다.

12.14.5.파일

파일 탭에서 확인할 수 있는 정보는 다음과 같다.

- 데이터 파일 관련 제니퍼 서버의 옵션 설정
- 디렉토리 별 파일 목록 및 크기
- 화면 맨 하단에서 전체 파일 크기를 확인할 수 있다.

12.14.6.성능 데이터베이스

성능 데이터베이스 탭에서 확인할 수 있는 정보는 다음과 같다.

- 성능 데이터베이스 관련 DBMS와 JDBC 드라이버 정보
- 반환 값이 있는 java.sql.Connection 클래스 메소드 이름과 해당 메소드 반환 값

12.14.7.Apache Derby

Apache Derby 탭에서 확인할 수 있는 정보는 다음과 같다.

- 아파치 더비 관련 주요 설정 정보와 데이터베이스 파일 디렉토리 위치
- 시스템 테이블을 제외한 성능 데이터베이스에 있는 모든 테이블 크기와 데이터 건수를 확인할 수 있다. 단, 성능 저하를 방지하기 위해서 일자별 테이블은 어제 날짜 테이블에 대해서만 건수를 보여준다. 특정 일자별 테이블 데이터 건수를 확인하려면 **[보기]** 링크를 클릭한다.
- 화면 맨 하단에서 전체 테이블 크기를 확인할 수 있다.

Notice: 성능 데이터베이스로 아파치 더비를 사용하는 경우에만 이 탭이 표시된다.

12.14.8.백업

백업 탭에서는 데이터를 백업하고 복원할 수 있다. 자세한 사항은 [데이터 백업 및 복구]를 참조한다.

12.14.9.이벤트

이벤트 탭에서는 이벤트 내용을 조회할 수 있다. 이벤트에 대한 상세한 내용은 [이벤트 로그 기록]을 참조한다. .

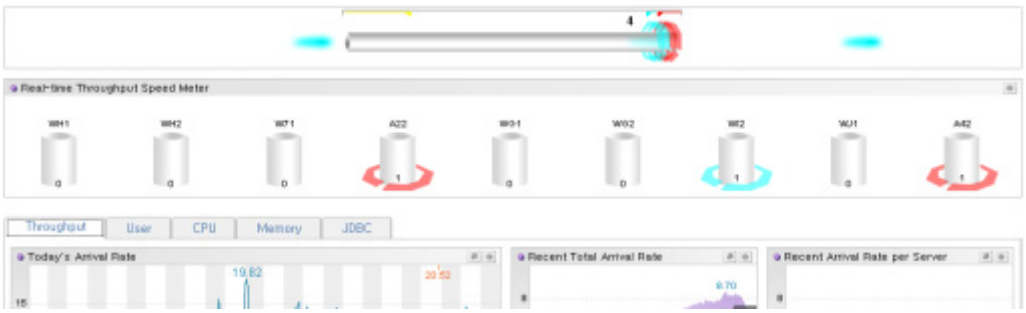
13

성능 현황과 보고서

13.1. 실시간 모니터링 - 실시간 현황

[실시간 모니터링 | 실시간 현황] 메뉴에서 오늘 날짜 일반 성능 데이터를 다양한 차트로 모니터링할 수 있다.

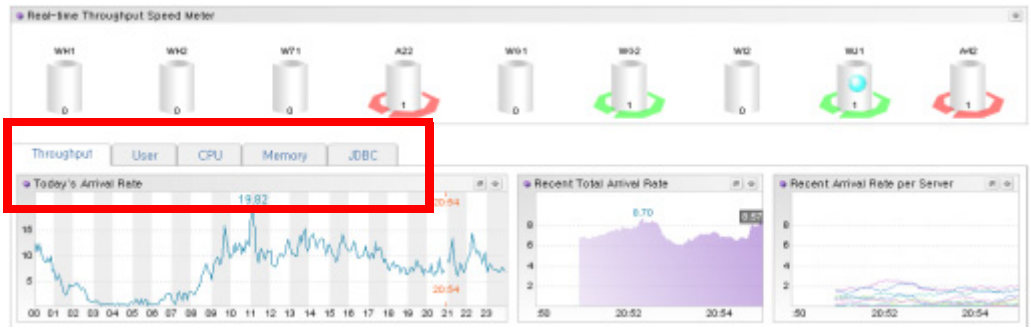
그림 13-1: 실시간 현황(1)



우선 화면 상단에 있는 스피드 바와 스피드 미터를 통해서 자바 애플리케이션의 실시간 부하량을 직관적으로 모니터링할 수 있다.

그리고 화면 중간에 있는 여러 탭을 통해서 업무 처리량, 사용자, CPU, 메모리 그리고 DB 등으로 분류된 오늘 날짜 일반 성능 데이터를 확인할 수 있다.

그림 13-2: 실시간 현황(2)

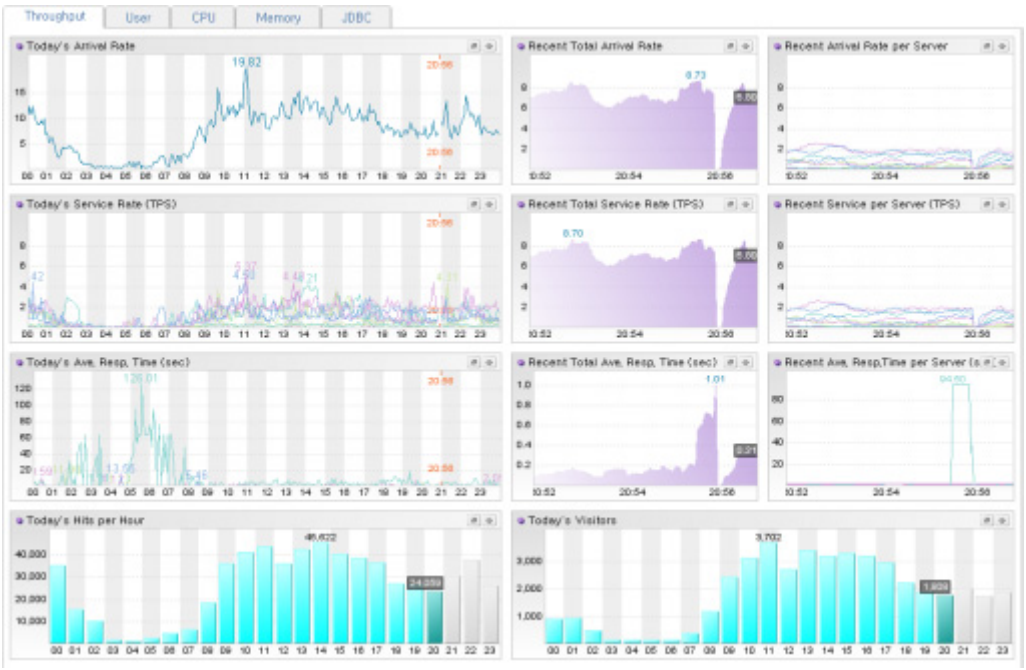


실시간 현황에서는 이퀄라이저 차트와 런타임 라인 차트 등을 주로 사용한다.

13.1.1. 업무 처리량

업무 처리량 탭에서는 오늘 날짜 서비스 요청률, 서비스 처리율, 평균 응답 시간, 시간당 호출 건수, 시간당 방문자 수 등을 런타임 라인 차트, 라인 차트, 그리고 막대 차트 등을 통해서 확인할 수 있다.

그림 13-3: 실시간 현황 - 업무 처리량 탭

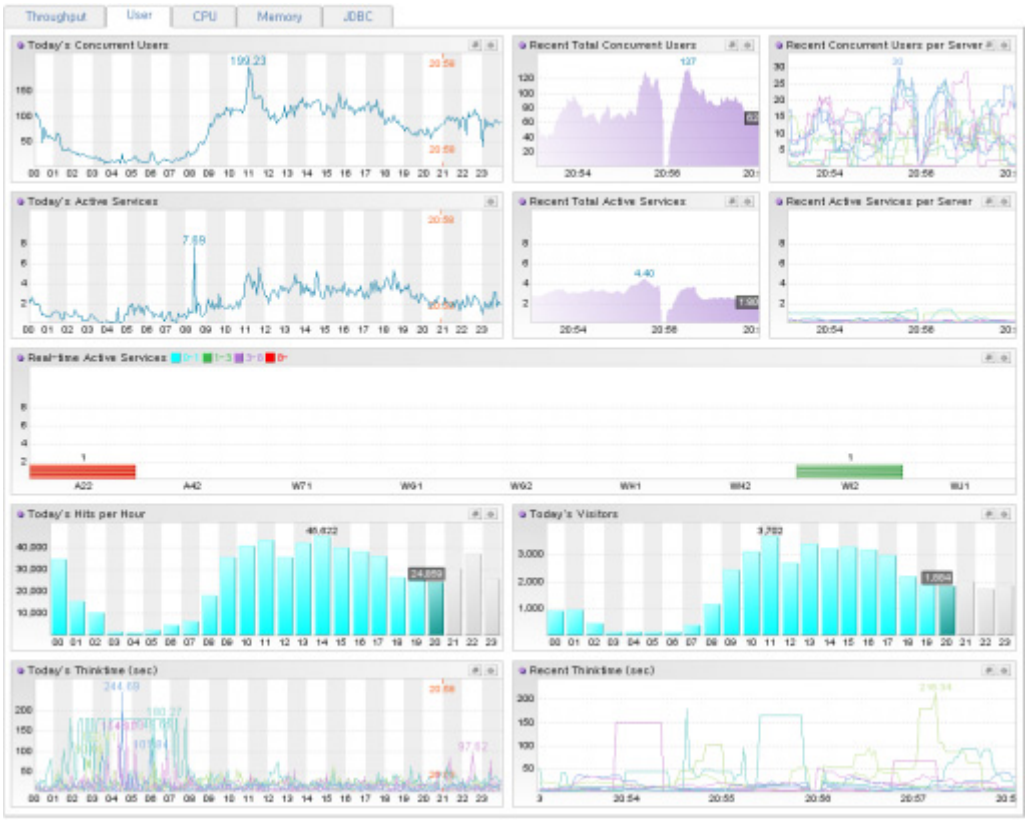


13.1.2.사용자

사용자 탭에서는 오늘 날짜 동시단말 사용자 수, 액티브 서비스 개수, 대기 시간, 시간당 방문자 수, 호출 건수 등을 이퀄라이저 차트, 런타임 라인 차트, 라인 차트 그리고 막대 차트 등을 통해서 확인할 수 있다.

Notice: 시간당 호출 건수는 업무 처리량과 관련이 있고 시간당 방문자 수는 사용자와 관련이 있다. 그런데 막대 차트의 균형적인 좌우 배치를 위해서 시간당 호출 건수와 시간당 방문자 수를 표시하는 막대 차트를 업무 처리량 탭과 사용자 탭에 모두 표시하였다.

그림 13-4: 실시간 현황 - 사용자 탭

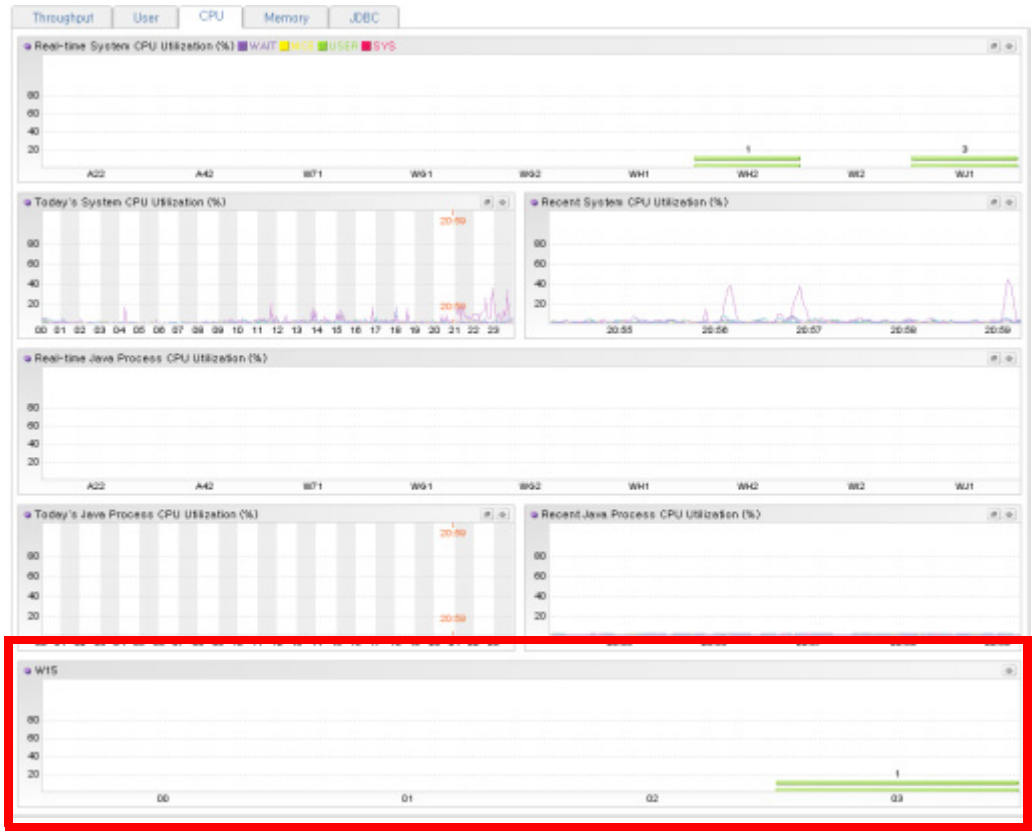


13.1.3.CPU

CPU 탭에서는 오늘 날짜 시스템 CPU 사용률, 자바 프로세스 CPU 사용률 등을 이퀄라이저 차트, 런타임 라인 차트, 그리고 라인 차트 등을 통해서 확인할 수 있다.

그리고 WMOND를 사용하면 CPU 탭 하단에 WMOND가 수집한 시스템 CPU 사용률이 이퀄라이저 차트로 나타난다.

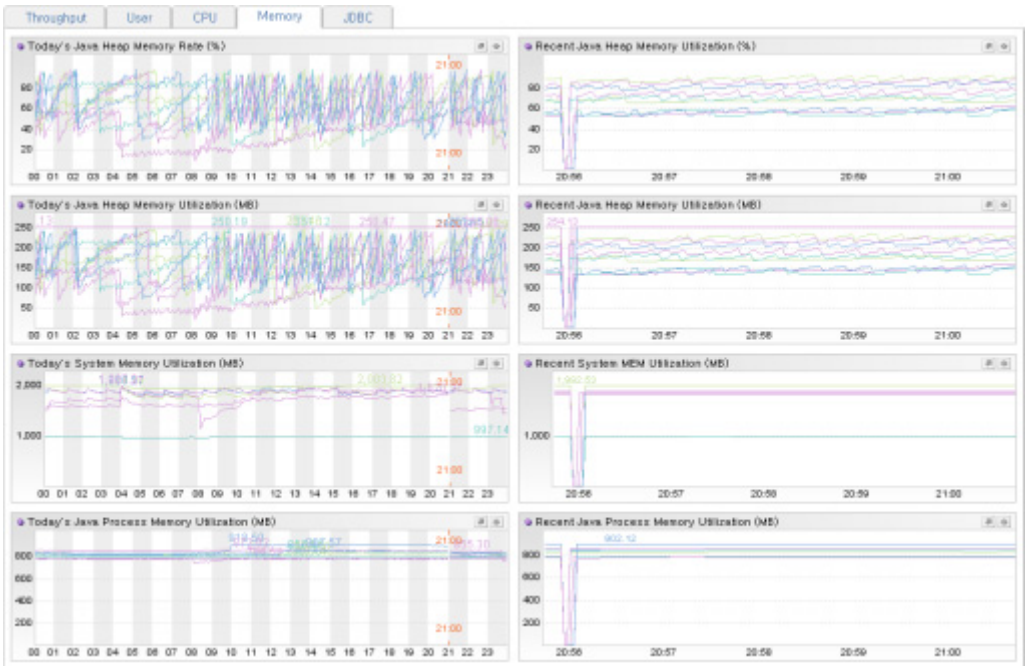
그림 13-5: 실시간 현황 - CPU 탭



13.1.4. 메모리

메모리 탭에서는 오늘 날짜 자바 힙 메모리 사용률, 자바 힙 메모리 사용량, 시스템 메모리 사용량, 자바 프로세스 메모리 사용량 등을 런타임 라인 차트와 라인 차트 등을 통해서 확인할 수 있다.

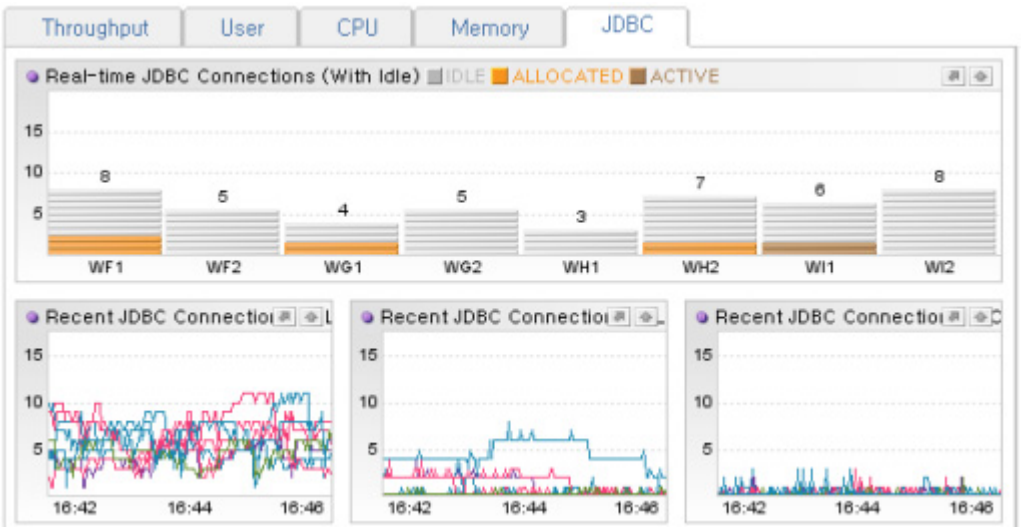
그림 13-6: 실시간 현황 - 메모리 탭



13.1.5.DB

DB 탭에서는 오늘 날짜 DB 커넥션 상태를 이퀄라이저 차트와 런타임 라인 차트 등을 통해서 확인할 수 있다.

그림 13-7: 실시간 현황 - JDBC 탭



13.2. 실시간 모니터링 - 애플리케이션

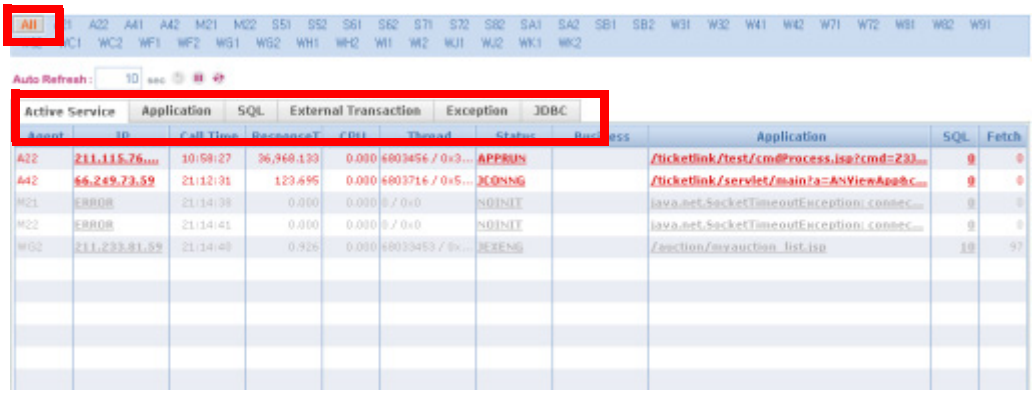
제니퍼 에이전트는 액티브 서비스, 애플리케이션, SQL, 외부 트랜잭션, 예외, DB 등의 처리 현황 통계 데이터를 수집한다. 사용자는 **[실시간 모니터링 | 애플리케이션]** 메뉴를 통해서 실시간 애플리케이션 처리 현황 통계 데이터를 열람할 수 있다. 내부적으로는 사용자 요청에 의해 제니퍼 서버가 제니퍼 에이전트로의 TCP 역방향 호출을 통해서 해당 데이터를 가져온다.

Notice: 액티브 서비스와 DB 처리 현황 통계 데이터는 실시간으로만 조회할 수 있다.

여기서 제공하는 데이터는 10분 단위로 요약한 통계 값이다. 예를 들어, 10분 동안의 어떤(ex login.jsp/login.aspx) 애플리케이션에 대한 전체 호출 건수와 평균 응답 시간 등은 제공하지만 어떤(ex login.jsp/login.aspx) 애플리케이션의 개별 트랜잭션에 대한 요청 시간과 응답 시간 등은 제공하지 않는다. 개별 트랜잭션을 분석하려면 [X-View와 프로파일링]을 참조한다.

Notice: 정확하게 설명하면, 실시간 처리 현황 통계 데이터는 현재 시간을 기준으로 한 10분을 의미하지 않는다. 00분, 10분, 20분과 같이 10분 단위를 기준으로 나누어지는 시간을 의미한다. 따라서 현재 시간이 09시 18분이라면 실시간 데이터에서는 09시 10분에서 09시 20분까지의 처리 현황 통계 데이터를 보여준다.

그림 13-8: 실시간 모니터링 | 애플리케이션



애플리케이션, SQL, 외부 트랜잭션, 예외, JDBC 등은 특정 제니퍼 에이전트에 대해서만 확인할 수 있다. 따라서 제니퍼 에이전트 선택 영역에서 [모두]를 선택하면 애플리케이션, SQL, 외부 트랜잭션, 예외, JDBC 등의 탭은 비활성화된다. 그러나 액티브 서비스에 대해서는 특정 제니퍼 에이전트 뿐만 아니라 모든 제니퍼 에이전트에 대해서도 확인할 수 있다. 단, 제니퍼 에이전트가 많은 경우에는 액티브 서비스 목록을 조회하는데 오랜 시간이 걸릴 수 있다.

13.2.1.액티브 서비스 처리 현황 통계

다음은 액티브 서비스 목록에 대한 설명이다.

표 13-1: 액티브 서비스 목록

항목	설명
에이전트	제니퍼 에이전트 아이디
IP	서비스를 요청한 사용자 IP 주소로 해당 칼럼을 클릭하면, WHOIS 기능을 통해서 IP에 대한 자세한 정보를 확인할 수 있다.
도착 시간	서비스 요청이 도착한 시간
경과 시간	현재까지 서비스 요청을 처리하는데 소요된 시간
CPU	현재까지 서비스 요청을 처리하는데 사용한 CPU 사용 시간
쓰레드	서비스 요청을 처리하고 있는 자바 쓰레드 아이디
상태	액티브 서비스 상태를 표시한다. 해당 칼럼을 클릭하면 액티브 서비스 상태 정보를 확인할 수 있다.
업무명	애플리케이션이 처리하는 업무 이름
애플리케이션	애플리케이션 이름으로 해당 칼럼을 클릭하면 액티브 서비스 상세 내역을 확인할 수 있다.
SQL	현재까지 액티브 서비스에서 수행된 SQL 수로 해당 칼럼을 클릭하면 해당 액티브 서비스에서 수행된 SQL을 확인할 수 있다.
Fetch	처리된 SQL Fetch 수

다음은 액티브 서비스 상태 코드에 대한 설명이다.

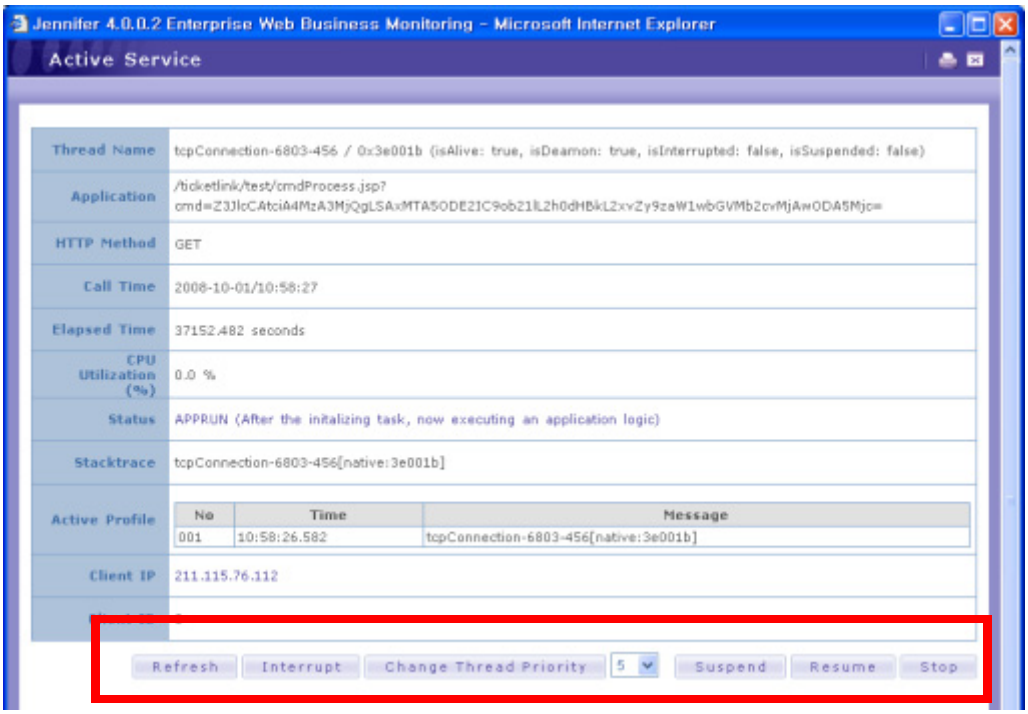
표 13-2: 액티브 서비스 상태 코드

상태 코드	설명
NOINIT	초기화 과정 중, URI, 원격 IP 주소, 자바 쓰레드 아이디 등 기본 정보를 추출하고 있는 상태
INITZG	액티브 서비스 등록 및 호출 건수를 증가시키고 있는 상태
REQST1	쿠키에서 기본 정보(사용자 아이디)를 추출하고 있는 상태
ARRSND	서비스 요청에 대한 데이터를 UDP 방식으로 제니퍼 서버에 전송하고 있는 상태
REQST2	캐릭터 셋(Character Set) 설정 및 최초의 HTTP GET/POST 데이터를 처리하고 있는 상태
REQEND	HTTP GET/POST 처리 후, 대기 시간, 방문자 수 등에 대한 계산을 처리하고 있는 상태

표 13-2: 액티브 서비스 상태 코드

상태 코드	설명
INITED	초기화 과정을 끝내고 PLC(Peak Load Control) 사용 여부를 체크하고 있는 상태
RJCTNG	PLC에 의한 폭주 중 메시지가 사용자에게 전달되고 있는 상태
RJCTED	PLC에 의한 폭주 중 메시지 전송이 완료된 상태
APPRUN	초기화 과정을 모두 끝내고 애플리케이션 로직을 수행하고 있는 상태
JCONNG	DB 연결(ds.getConection())을 수행하고 있는 상태
JCONND	DB 연결이 된 후, 그 다음 애플리케이션 로직을 수행하고 있는 상태
JPRENG	con.prepareStatement("...") 코드를 수행하고 있는 상태
JPREPD	con.prepareStatement("...") 코드 수행 후, 그 다음 로직을 수행하고 있는 상태
JEXENG	DB SQL 쿼리를 수행하고 있는 상태
JEXETD	DB SQL 쿼리를 수행한 후에, 그 다음 로직을 진행하고 있는 상태
JRSNXT	rs.next() 코드를 반복적으로 수행하고 있는 상태
JRSFLS	rs.next() 코드 수행 결과가 false가 된 후, 그 다음 로직을 수행하고 있는 상태
JRSCLD	rs.close() 코드가 수행된 후, 그 다음 로직을 수행하고 있는 상태
JSTCLD	stat.close()와 pstmt.close() 코드가 수행된 후, 그 다음 로직을 수행하고 있는 상태
JCONCD	con.close() 코드가 수행된 후, 그 다음 로직을 수행하고 있는 상태
TXEXEC	CTG/Jolt/JLink/WebT 등과의 외부 트랜잭션을 수행하고 있는 상태
TXEXED	외부 트랜잭션 수행을 끝내고, 그 다음 로직을 수행하고 있는 상태
ENDAPP	애플리케이션 로직을 모두 끝낸 상태
ENDLNG	사용자 요청을 처리하는데 소요된 응답 시간이 지정된 값 보다 오래 걸린 경우의 상태
ERREXP	애플리케이션 로직 수행 중에 처리되지 않은 예외가 발생한 상태
ERRIOE	예외 사항이 java.io.IOException인 경우의 상태
ERRRCS	반복적인 서블릿 객체의 service 메소드 호출을 감지하여 강제 중단시킨 상태

그림 13-9: 액티브 서비스 상세 내역



액티브 서비스 상세 내역 화면 하단에 있는 버튼을 통해서 액티브 서비스를 처리하고 있는 자바 쓰레드를 컨트롤할 수 있다.

- 인터럽트 - 해당 자바 쓰레드 객체의 interrupt 메소드를 호출한다.
- 쓰레드 우선순위 변경 - 해당 자바 쓰레드 객체의 setPriority 메소드를 호출하여 쓰레드의 우선 순위를 변경한다.
- 일시 정지 - 해당 자바 쓰레드 객체의 suspend 메소드를 호출하여 자바 쓰레드를 일시 중지시킨다.
- 재시작 - 해당 자바 쓰레드 객체의 resume 메소드를 호출하여 자바 쓰레드를 다시 시작시킨다.
- 정지 - 해당 자바 쓰레드 객체의 stop 메소드를 호출하여 자바 쓰레드를 정지시킨다.

DB 커넥션이 열려 있는 상태에서 정지 버튼을 클릭하면 DB 커넥션 릭이 발생할 수도 있다.

또한 WAS는 일반적으로 성능 향상을 위해서 쓰레드 풀을 사용하여 쓰레드를 관리한다. WAS가 애플리케이션에 쓰레드를 할당하기 전에 쓰레드가 정상적인 상태에 있는지를 체크한다면, 제니퍼를 통해서 특정 쓰레드를 중지시키는 것이 큰 문제가 되지는 않는다. 하지만 WAS가 이를 확인하지 않는다면 비정상적인 쓰레드가 애플리케이션에 할당되어 다른 서비스를 처리하는 과정에서 알 수 없는 문제가 발생할 수 있다. 따라서 완료될 가능성

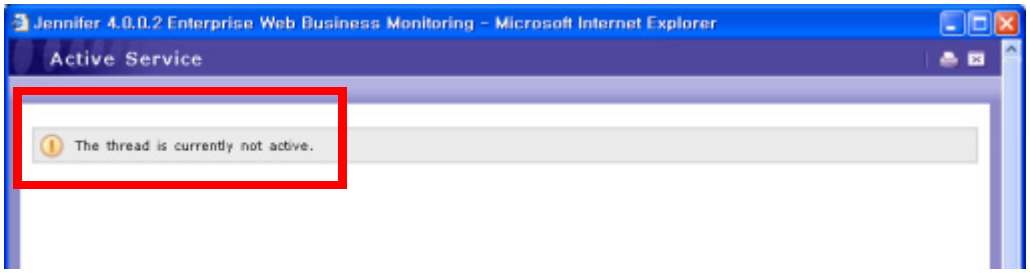
이 없으면서 CPU를 과도하게 사용하는 액티브 서비스가 있다면 정지보다는 일시 정지하는 것을 권장한다.

제니퍼 에이전트의 enable_active_thread_kill 옵션을 통해서 자바 쓰레드 중지를 방지할 수 있다.

```
enable_active_thread_kill = false
```

액티브였던 서비스가 더 이상 실행 상태가 아니면 파업 창에 다음과 같은 메시지가 나타난다.

그림 13-10:액티브 서비스가 종료된 경우의 상세 내역



13.2.2.애플리케이션 처리 현황 통계

다음은 애플리케이션 목록에 대한 설명이다.

표 13-3: 애플리케이션 목록

항목	설명
호출 건수	애플리케이션 수행 건수
성공 건수	애플리케이션 수행이 성공한 건수
실패 건수	애플리케이션 수행이 실패한 건수
응답 시간의 합	애플리케이션 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	애플리케이션 수행에 소요된 응답 시간의 표준 편차
최소 응답	애플리케이션 수행에 소요된 최소 응답 시간
최대 응답	애플리케이션 수행에 소요된 최대 응답 시간
평균 CPU	애플리케이션 수행에 사용된 평균 CPU 사용 시간
CPU(tpmC)	애플리케이션 수행에 사용된 CPU(tpmC) 값

표 13-3: 애플리케이션 목록

항목	설명
CPU 합	애플리케이션 수행에 사용된 CPU 사용 시간의 합
애플리케이션	애플리케이션 이름으로 일반적으로 URI를 의미한다.

애플리케이션 목록에서 특정 애플리케이션을 선택하면 하단에 해당 애플리케이션에서 수행된 SQL과 외부 트랜잭션 목록이 표시된다. 다음은 이에 대한 설명이다.

표 13-4: 애플리케이션과 관련한 상세 목록

항목	설명
유형	SQL은 SQL을 의미하고 TX는 외부 트랜잭션을 의미한다.
애플리케이션 수행 점유 비율	SQL 혹은 외부 트랜잭션의 응답 시간이 해당 애플리케이션 응답 시간에서 차지하는 점유 비율로 단위는 퍼센트이다.
호출 건수	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행 건수
응답 시간의 합	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행에 소요된 응답 시간의 표준 편차
최소 응답	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행에 소요된 최소 응답 시간
최대 응답	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행에 소요된 최대 응답 시간
SQL/트랜잭션	SQL 혹은 외부 트랜잭션 이름

13.2.3.SQL 처리 현황 통계

다음은 SQL 목록에 대한 설명이다.

표 13-5: SQL 목록

항목	설명
호출 건수	SQL 수행 건수
응답 시간의 합	SQL 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	SQL 수행에 소요된 응답 시간의 표준 편차

표 13-5: SQL 목록

항목	설명
최소 응답	SQL 수행에 소요된 최소 응답 시간
최대 응답	SQL 수행에 소요된 최대 응답 시간
SQL	SQL

SQL 목록에서 특정 SQL을 선택하면 하단에 해당 SQL을 수행한 애플리케이션 목록이 표시된다. 다음은 이에 대한 설명이다.

표 13-6: SQL 과 관련한 상세 목록

항목	설명
애플리케이션 수행 점유 비율	SQL의 응답 시간이 해당 애플리케이션 응답 시간에서 차지하는 점유 비율로 단위는 퍼센트이다.
호출 건수	해당 애플리케이션에서 호출한 SQL 수행 건수
응답 시간의 합	해당 애플리케이션에서 호출한 SQL 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	해당 애플리케이션에서 호출한 SQL 수행에 소요된 응답 시간의 표준 편차
최소 응답	해당 애플리케이션에서 호출한 SQL 수행에 소요된 최소 응답 시간
최대 응답	해당 애플리케이션에서 호출한 SQL 수행에 소요된 최대 응답 시간
애플리케이션	애플리케이션 이름

13.2.4.외부 트랜잭션 처리 현황 통계

다음은 외부 트랜잭션 목록에 대한 설명이다.

표 13-7: 외부 트랜잭션

항목	설명
호출 건수	외부 트랜잭션 수행 건수
응답 시간의 합	외부 트랜잭션 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	외부 트랜잭션 수행에 소요된 응답 시간의 표준 편차
최소 응답	외부 트랜잭션 수행에 소요된 최소 응답 시간
최대 응답	외부 트랜잭션 수행에 소요된 최대 응답 시간
외부 트랜잭션	외부 트랜잭션 이름

외부 트랜잭션 목록에서 특정 외부 트랜잭션을 선택하면 하단에 해당 외부 트랜잭션을 수행한 애플리케이션 목록이 표시된다. 다음은 이에 대한 설명이다.

표 13-8: 외부 트랜잭션과 관련한 상세 목록

항목	설명
애플리케이션 수행 점유 비율	외부 트랜잭션의 응답 시간이 해당 애플리케이션 응답 시간에서 차지하는 점유 비율로 단위는 퍼센트이다.
호출 건수	해당 애플리케이션에서 호출한 외부 트랜잭션 수행 건수
응답 시간의 합	해당 애플리케이션에서 호출한 외부 트랜잭션 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	해당 애플리케이션에서 호출한 외부 트랜잭션 수행에 소요된 응답 시간의 표준 편차
최소 응답	해당 애플리케이션에서 호출한 외부 트랜잭션 수행에 소요된 최소 응답 시간
최대 응답	해당 애플리케이션에서 호출한 외부 트랜잭션 수행에 소요된 최대 응답 시간
애플리케이션	애플리케이션 이름

13.2.5.예외 현황 통계

다음은 예외 목록에 대한 설명이다.

표 13-9: 예외 목록

항목	설명
발생 건수	예외가 발생한 건수
발생 비율	예외 목록에 표시된 모든 예외의 발생 건수의 합에 대한 해당 예외의 발생 건수의 비율로 단위는 퍼센트이다.
예외	예외의 이름

예외 목록에서 특정 예외를 선택하면 하단에 해당 예외가 발생한 애플리케이션 목록이 표시된다. 다음은 이에 대한 설명이다.

표 13-10: 예외와 관련한 상세 목록

항목	설명
발생 건수	애플리케이션에서 특정 예외가 발생한 건수
발생 비율	예외와 관련한 상세 목록에 표시된 모든 애플리케이션의 예외 발생 건수의 합에 대한 해당 애플리케이션의 예외 발생 건수의 비율로 단위는 퍼센트이다.

표 13-10: 예외와 관련한 상세 목록

항목	설명
애플리케이션	예외가 발생한 애플리케이션 이름
내용	예외와 관련한 내용으로 주로 예외 메시지와 스택트레이스 등이 표시된다.

13.2.6.DB 현황 통계

다음은 DB 목록에 대한 설명이다.

표 13-11: DB 목록

항목	설명
해시 코드	JDBC 커넥션 객체의 해시 코드
DB SID	오라클 데이터베이스 SID로 오라클 데이터베이스를 사용하는 경우에만 표시된다. 오라클 데이터베이스를 사용하는 경우에도 SID를 표시하려면 별도의 설정이 필요하다.
클래스	java.sql.Connection 인터페이스에 대한 구현 클래스로 사용하는 JDBC 드라이버에 따라서 달라진다.
이름	JNDI에 등록된 이름
IP	해당 DB 커넥션을 사용하고 있는 자바 쓰레드가 처리하고 있는 서비스를 요청한 사용자 IP 주소
호출 시간	해당 DB 커넥션을 사용하고 있는 자바 쓰레드가 처리하고 있는 서비스의 요청이 도착한 시간
응답 시간	해당 DB 커넥션을 사용하고 있는 자바 쓰레드가 처리하고 있는 서비스의 경과 시간
쓰레드	해당 DB 커넥션을 사용하고 있는 자바 쓰레드 아이디
상태	액티브 서비스 상태 코드와 동일
애플리케이션	해당 DB 커넥션을 사용하고 있는 자바 쓰레드가 처리하고 있는 서비스 요청의 애플리케이션 이름으로, 액티브 서비스 목록과 동일하게 해당 칼럼을 클릭하면 액티브 서비스 상세 내역을 확인할 수 있다.

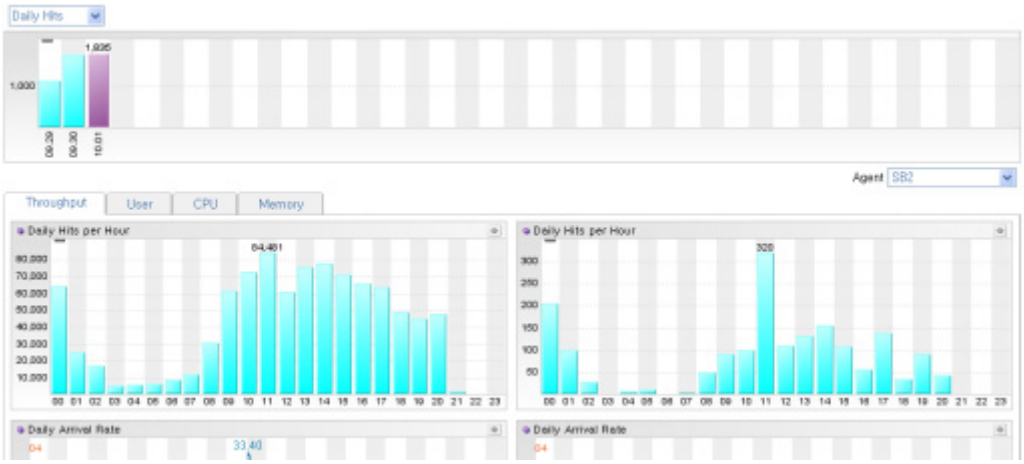
IP에서 애플리케이션까지의 항목은 해당 DB 커넥션을 사용하고 있는 자바 쓰레드가 처리하고 있는 액티브 서비스에 대한 설명으로 DB 커넥션이 대기중인 상태이면 아무런 내용도 표시되지 않는다.

Notice: 프로그래밍의 부주의로 동일한 DB 커넥션 객체가 동일 시점에 2개 이상의 자바 쓰레드에 할당될 수도 있다. 이런 경우에는 애플리케이션에 [중복 할당]이라는 표시가 나타난다.

13.3. 통계 분석 - 통계 현황

[실시간 모니터링 | 통계 현황] 메뉴에서 선택한 날짜 일반 성능 데이터를 다양한 차트로 모니터링할 수 있다.

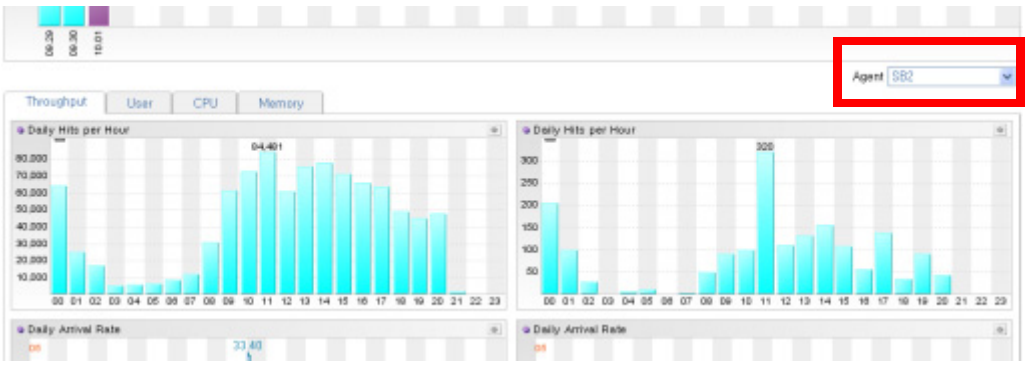
그림 13-11:통계 현황(1)



우선 화면 상단에 있는 막대 차트를 통해서 일자별 호출 건수와 일자별 방문자 수를 확인할 수 있다. 기본으로 일자별 호출 건수가 보인다. 막대 차트에서 일반 성능 데이터를 확인할 날짜에 해당하는 막대를 클릭하면 화면 중간에 있는 여러 탭에 해당 일반 성능 데이터가 나타난다.

그리고 화면 중간에 있는 여러 탭을 통해서 업무 처리량, 사용자, CPU, 메모리 등으로 분류된 선택한 날짜 일반 성능 데이터를 확인할 수 있다.

그림 13-12:통계 현황(2)

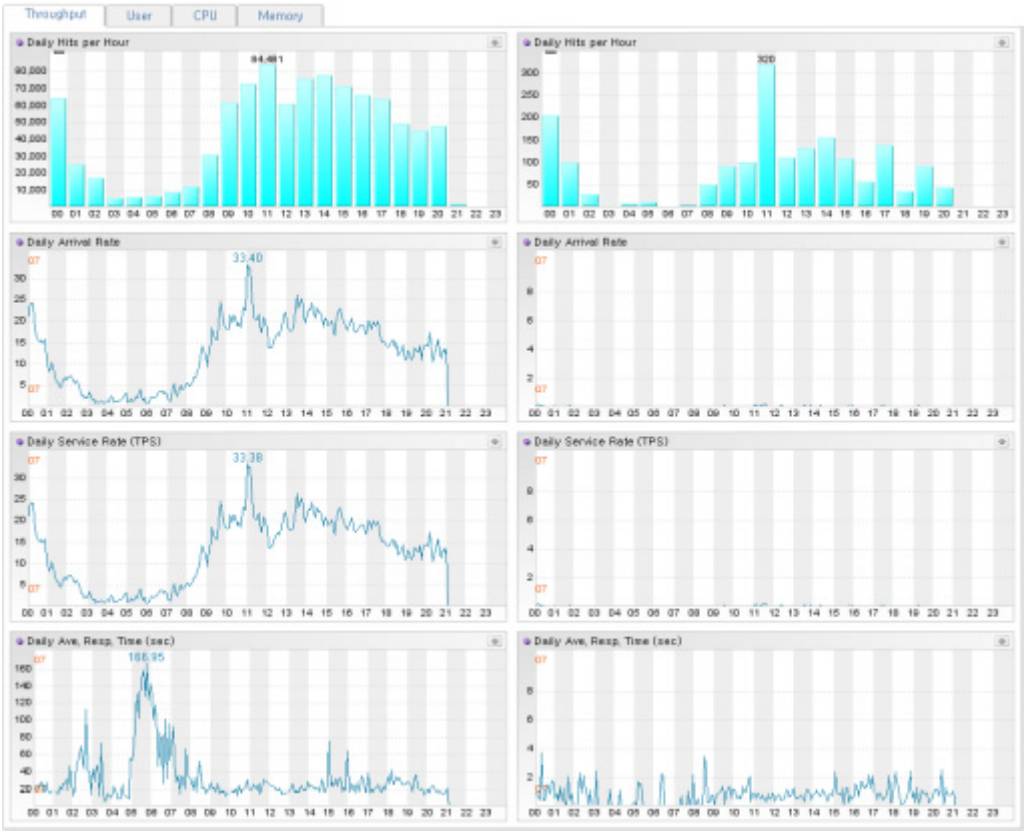


통계 현황에서는 라인 차트를 주로 사용한다.

13.3.1.업무 처리량

업무 처리량 탭에서는 선택한 날짜 서비스 요청률, 서비스 처리율, 평균 응답 시간, 시간당 호출 건수 등을 라인 차트와 막대 차트 등을 통해서 확인할 수 있다.

그림 13-13:통계 현황 - 업무 처리량 탭

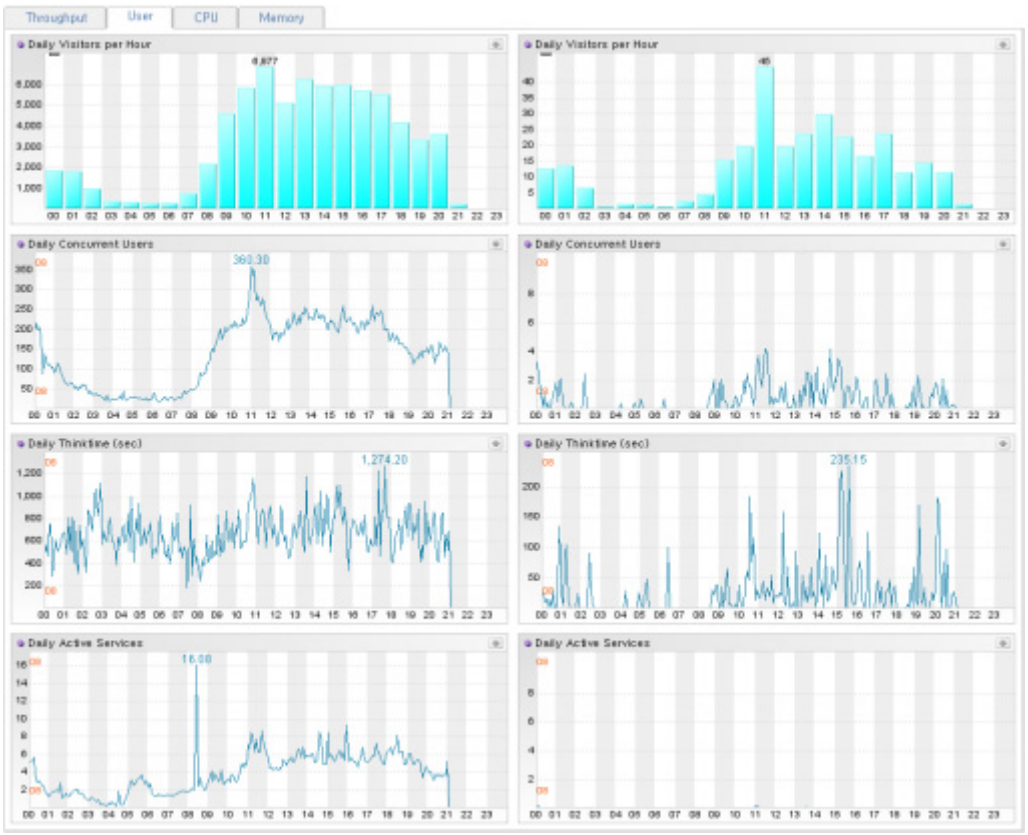


업무 처리량 탭의 왼쪽에 있는 차트는 특정 제니퍼 에이전트가 아닌 모니터링하는 전체 제니퍼 에이전트에서 수집한 값에 대한 누적 값 혹은 평균 값을 표시한다. 그리고 오른쪽에 있는 차트가 막대 차트이면 아무런 내용도 표시되지 않고, 라인 차트이면 제니퍼 에이전트 별로 수집한 성능 데이터를 개별적인 선으로 표시한다. 업무 처리량 탭 오른쪽 상단에 있는 리스트에서 특정 제니퍼 에이전트를 선택하면 오른쪽에 있는 막대 차트에는 해당 제니퍼 에이전트에서 수집한 성능 데이터가 나타나고 라인 차트에도 해당 제니퍼 에이전트에서 수집한 성능 데이터만이 선으로 표시된다.

13.3.2.사용자

사용자 탭에서는 선택한 날짜 동시단말 사용자 수, 액티브 서비스 개수, 대기 시간, 시간당 방문자 수 등을 라인 차트와 막대 차트 등을 통해서 확인할 수 있다.

그림 13-14:통계 현황 - 사용자 탭

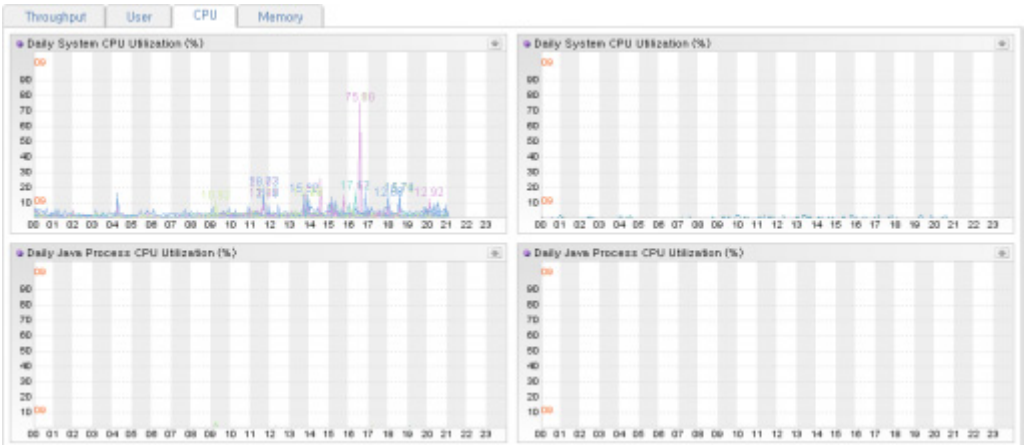


사용자 탭의 왼쪽에 있는 차트는 특정 제니퍼 에이전트가 아닌 모니터링하는 전체 제니퍼 에이전트에서 수집한 값에 대한 누적 값 혹은 평균 값을 표시한다. 그리고 오른쪽에 있는 차트가 막대 차트이면 아무런 내용도 표시되지 않고, 라인 차트이면 제니퍼 에이전트 별로 수집한 성능 데이터를 개별적인 선으로 표시한다. 사용자 탭 오른쪽 상단에 있는 리스트에서 특정 제니퍼 에이전트를 선택하면 오른쪽에 있는 막대 차트에는 해당 제니퍼 에이전트에서 수집한 성능 데이터가 나타나고 라인 차트에도 해당 제니퍼 에이전트에서 수집한 성능 데이터만이 선으로 표시된다.

13.3.3.CPU

CPU 탭에서는 선택한 날짜 시스템 CPU 사용률, 자바 프로세스 CPU 사용률 등을 라인 차트 등을 통해서 확인할 수 있다.

그림 13-15:통계 현황 - CPU 탭



CPU와 관련한 성능 데이터는 업무 처리량이나 사용자와 관련한 성능 데이터와는 다르게 모든 제니퍼 에이전트에 대한 누적 값 혹은 평균 값이 의미가 없다. 따라서 CPU 탭의 왼쪽에 있는 라인 차트는 제니퍼 에이전트 별로 수집한 성능 데이터를 개별적인 선으로 표시한다. 그리고 오른쪽에는 아무런 내용도 표시되지 않는다. CPU 탭 오른쪽 상단에 있는 리스트에서 특정 제니퍼 에이전트를 선택하면 오른쪽에 있는 라인 차트에 해당 제니퍼 에이전트에서 수집한 성능 데이터만이 선으로 표시된다.

13.3.4.메모리

메모리 탭에서는 선택한 날짜 자바 힙 메모리 사용률, 자바 힙 메모리 사용량, 시스템 메모리 사용량, 자바 프로세스 메모리 사용량 등을 라인 차트 등을 통해서 확인할 수 있다.

그림 13-16:통계 현황 - 메모리 탭

메모리와 관련한 성능 데이터는 업무 처리량이나 사용자와 관련한 성능 데이터와는 다르게 모든 제니퍼 에이전트에 대한 누적 값 혹은 평균 값이 의미가 없다. 따라서 메모리 탭의 왼쪽에 있는 라인 차트는 제니퍼 에이전트 별로 수집한 성능 데이터를 개별적인 선으로 표시한다. 그리고 오른쪽에는 아무런 내용도 표시되지 않는다. 메모리 탭 오른쪽 상단에 있는 리스트에서 특정 제니퍼 에이전트를 선택하면 오른쪽에 있는 라인 차트에 해당 제니퍼 에이전트에서 수집한 성능 데이터만이 선으로 표시된다.

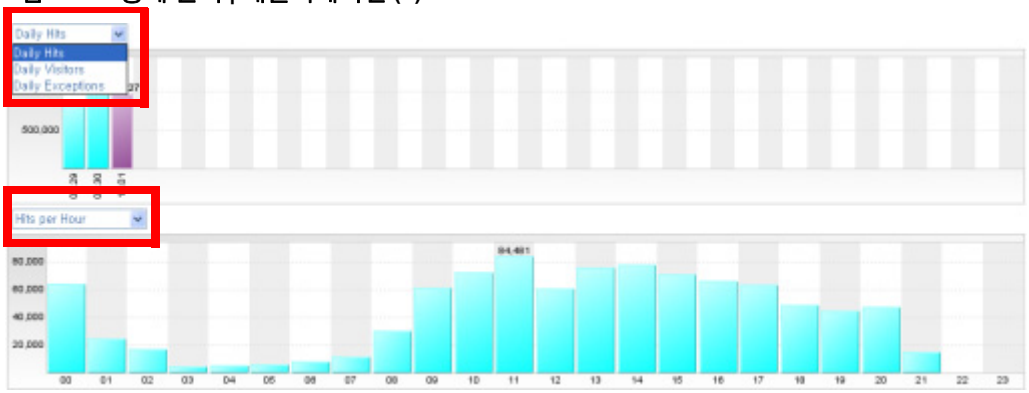
13.4. 통계 분석 - 애플리케이션

제니퍼 서버는 제니퍼 에이전트로부터 10분 단위로 수집한 애플리케이션, SQL, 외부 트랜잭션, 예외 처리 현황 통계 데이터를 데이터베이스에 저장한다. 그리고 사용자는 **[통계 분석 | 애플리케이션]** 메뉴를 통해서 이 데이터를 열람할 수 있다.

여기서 제공하는 데이터는 10분 단위로 요약한 통계 값이다. 예를 들어, 10분 동안의 어떤(ex login.jsp/login.aspx) 애플리케이션에 대한 전체 호출 건수와 평균 응답 시간 등은 제공하지만 어떤(ex login.jsp/login.aspx) 애플리케이션의 개별 트랜잭션에 대한 요청 시간과

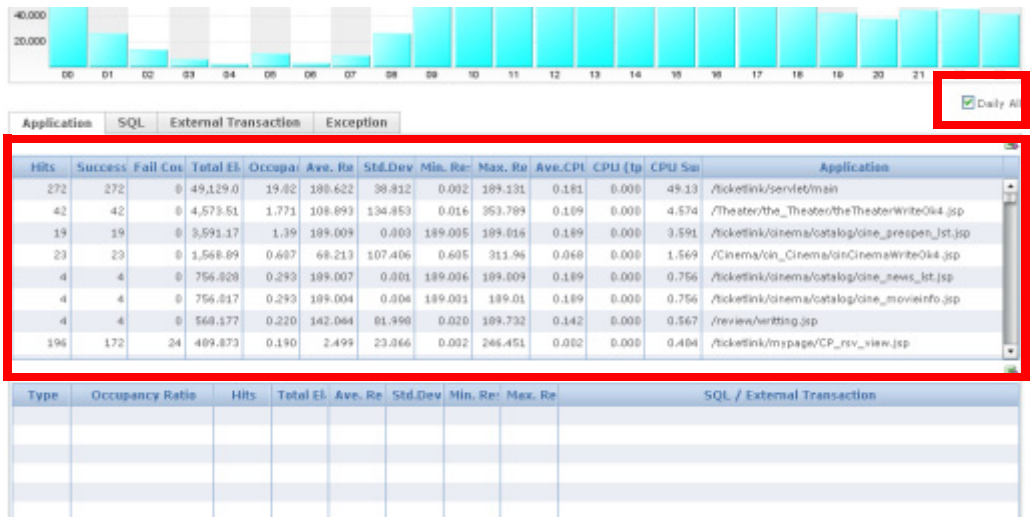
응답 시간 등은 제공하지 않는다. 개별 트랜잭션을 분석하려면 [X-View와 프로파일링]을 참조한다.

그림 13-17:통계 분석 | 애플리케이션 (1)



첫번째 막대 차트에는 일자별 호출 건수, 일자별 방문자 수, 일자별 예외 건수 등이 표시된다. 기본은 일자별 호출 건수이다. 첫번째 막대 차트에서 특정 일을 클릭하면 두번째 차트에 해당 일에 대한 시간당 호출 건수, 시간당 방문자 수, 동시단말 사용자 수, 대기 시간, 액티브 서비스 개수, 서비스 요청률, 서비스 처리율, 평균 응답 시간, 시간당 예외 건수 등이 표시된다. 기본은 시간당 호출 건수이고 시간당 호출 건수, 시간당 방문자 수, 시간당 예외 건수는 1시간 단위의 막대 차트로 표시되고 다른 데이터는 5분 단위의 라인 차트로 표시된다.

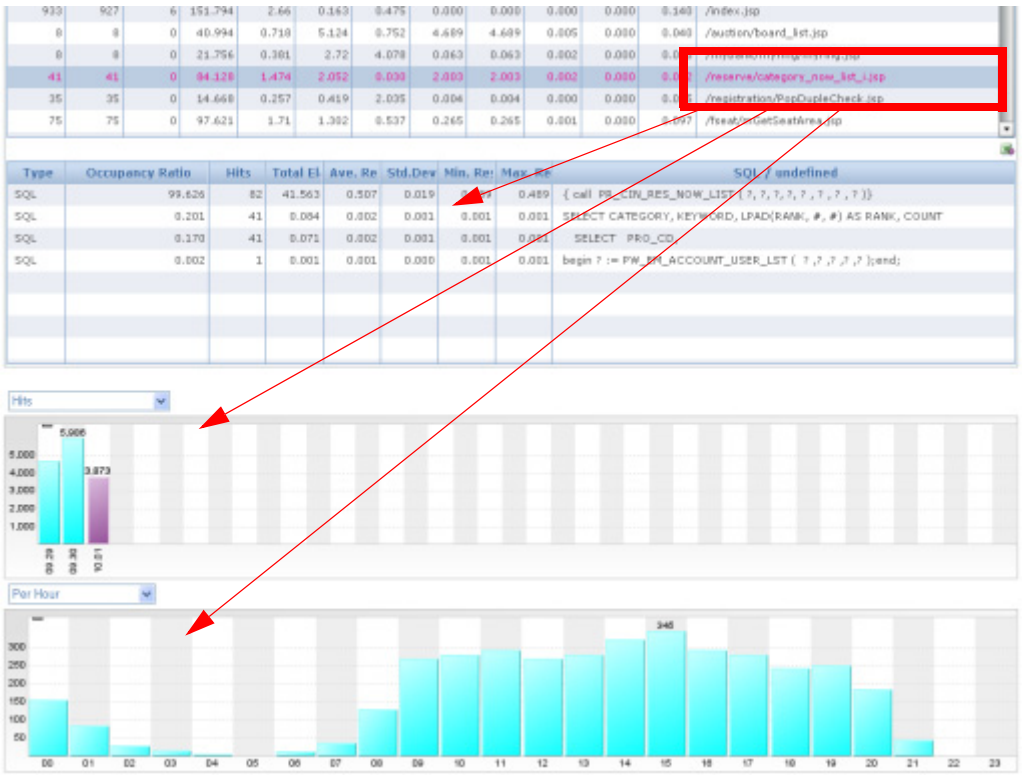
그림 13-18:통계 분석 | 애플리케이션 (2)



두번째 차트의 유형으로 시간당 호출 건수, 시간당 방문자 수, 시간당 예외 건수 중에서 하나를 선택한 후에 특정 시간을 클릭하면 하단 그리드에는 해당 시간대의 애플리케이션, SQL, 외부 트랜잭션, 예외 등의 처리 현황 통계 데이터가 표시된다.

Notice: 오른쪽에 있는 **[일자별 전체]** 체크 박스를 선택하면, 시간대와 상관없이 해당 날짜의 애플리케이션, SQL, 외부 트랜잭션, 예외 등의 전체 처리 현황 통계 데이터가 표시된다.

그림 13-19:통계 분석 | 애플리케이션 (3)



첫번째 그리드에서 특정 애플리케이션, 특정 SQL, 특정 외부 트랜잭션, 특정 예외를 선택하면 두번째 그리드와 세번째와 네번째 차트에 그와 관련한 정보가 표시된다.

예를 들어, 첫번째 그리드에서 특정 애플리케이션을 선택하면 두번째 그리드에서는 동일 시간대에 해당 애플리케이션에서 수행된 SQL과 외부 트랜잭션 목록이 표시되고, 세번째 차트에는 해당 애플리케이션에 대한 호출 건수, 실패 건수, 응답 시간의 합, 평균 응답 시간, 최소 응답, 최대 응답, CPU 시간, CPU(tpmC) 등을 일자별로 보여준다. 세번째 차트에서 특정 일을 클릭하면 네번째 차트에 선택한 날짜에 대한 데이터가 막대 차트 혹은 라인 차트로 표시된다.

13.4.1.애플리케이션 처리 현황 통계

다음은 애플리케이션 목록에 대한 설명이다.

표 13-12: 애플리케이션 목록

항목	설명
호출 건수	애플리케이션 수행 건수
성공 건수	애플리케이션 수행이 성공한 건수
실패 건수	애플리케이션 수행이 실패한 건수
응답 시간의 합	애플리케이션 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	애플리케이션 수행에 소요된 응답 시간의 표준 편차
최소 응답	애플리케이션 수행에 소요된 최소 응답 시간
최대 응답	애플리케이션 수행에 소요된 최대 응답 시간
평균 CPU	애플리케이션 수행에 사용된 평균 CPU 사용 시간
CPU(tpmC)	애플리케이션 수행에 사용된 CPU(tpmC) 값
CPU 합	애플리케이션 수행에 사용된 CPU 사용 시간의 합
애플리케이션	애플리케이션 이름으로 일반적으로 URI를 의미하다.

애플리케이션 목록에서 특정 애플리케이션을 선택하면 하단에 해당 애플리케이션에서 수행된 SQL과 외부 트랜잭션 목록이 표시된다. 다음은 이에 대한 설명이다.

표 13-13: 애플리케이션과 관련한 상세 목록

항목	설명
유형	SQL은 SQL을 의미하고 TX는 외부 트랜잭션을 의미한다.
애플리케이션 수행 점유 비율	SQL 혹은 외부 트랜잭션의 응답 시간이 해당 애플리케이션 응답 시간에서 차지하는 점유 비율로 단위는 퍼센트이다.
호출 건수	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행 건수
응답 시간의 합	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행에 소요된 응답 시간의 표준 편차
최소 응답	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행에 소요된 최소 응답 시간

표 13-13: 애플리케이션과 관련한 상세 목록

항목	설명
최대 응답	해당 애플리케이션에서 호출한 SQL 혹은 외부 트랜잭션 수행에 소요된 최대 응답 시간
SQL/외부 트랜잭션	SQL 혹은 외부 트랜잭션 이름

13.4.2.SQL 처리 현황 통계

다음은 SQL 목록에 대한 설명이다.

표 13-14: SQL 목록

항목	설명
호출 건수	SQL 수행 건수
응답 시간의 합	SQL 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	SQL 수행에 소요된 응답 시간의 표준 편차
최소 응답	SQL 수행에 소요된 최소 응답 시간
최대 응답	SQL 수행에 소요된 최대 응답 시간
SQL	SQL

SQL 목록에서 특정 SQL을 선택하면 하단에 해당 SQL을 수행한 애플리케이션 목록이 표시된다. 다음은 이에 대한 설명이다.

표 13-15: SQL 과 관련한 상세 목록

항목	설명
애플리케이션 수행 점유 비율	SQL의 응답 시간이 해당 애플리케이션 응답 시간에서 차지하는 점유 비율로 단위는 퍼센트이다.
호출 건수	해당 애플리케이션에서 호출한 SQL 수행 건수
응답 시간의 합	해당 애플리케이션에서 호출한 SQL 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	해당 애플리케이션에서 호출한 SQL 수행에 소요된 응답 시간의 표준 편차
최소 응답	해당 애플리케이션에서 호출한 SQL 수행에 소요된 최소 응답 시간
최대 응답	해당 애플리케이션에서 호출한 SQL 수행에 소요된 최대 응답 시간
애플리케이션	애플리케이션 이름

13.4.3.외부 트랜잭션 처리 현황 통계

다음은 외부 트랜잭션 목록에 대한 설명이다.

표 13-16: 외부 트랜잭션

항목	설명
호출 건수	외부 트랜잭션 수행 건수
응답 시간의 합	외부 트랜잭션 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	외부 트랜잭션 수행에 소요된 응답 시간의 표준 편차
최소 응답	외부 트랜잭션 수행에 소요된 최소 응답 시간
최대 응답	외부 트랜잭션 수행에 소요된 최대 응답 시간
트랜잭션	외부 트랜잭션 이름

외부 트랜잭션 목록에서 특정 외부 트랜잭션을 선택하면 하단에 해당 외부 트랜잭션을 수행한 애플리케이션 목록이 표시된다. 다음은 이에 대한 설명이다.

표 13-17: 외부 트랜잭션과 관련한 상세 목록

항목	설명
애플리케이션 수행 점유 비율	외부 트랜잭션의 응답 시간이 해당 애플리케이션 응답 시간에서 차지하는 점유 비율로 단위는 퍼센트이다.
호출 건수	해당 애플리케이션에서 호출한 외부 트랜잭션 수행 건수
응답 시간의 합	해당 애플리케이션에서 호출한 외부 트랜잭션 수행에 소요된 응답 시간의 합
평균 응답 시간	응답 시간의 합을 호출 건수로 나눈 값
표준 편차	해당 애플리케이션에서 호출한 외부 트랜잭션 수행에 소요된 응답 시간의 표준 편차
최소 응답	해당 애플리케이션에서 호출한 외부 트랜잭션 수행에 소요된 최소 응답 시간
최대 응답	해당 애플리케이션에서 호출한 외부 트랜잭션 수행에 소요된 최대 응답 시간
애플리케이션	애플리케이션 이름

13.4.4.예외 현황 통계

다음은 예외 목록에 대한 설명이다.

표 13-18: 예외 목록

항목	설명
발생 건수	예외가 발생한 건수
발생 비율	예외 목록에 표시된 모든 예외의 발생 건수의 합에 대한 해당 예외의 발생 건수의 비율로 단위는 퍼센트이다.
예외	예외의 이름

예외 목록에서 특정 예외를 선택하면 하단에 해당 예외가 발생한 애플리케이션 목록이 표시된다. 다음은 이에 대한 설명이다.

표 13-19: 예외와 관련한 상세 목록

항목	설명
발생 건수	애플리케이션에서 특정 예외가 발생한 건수
발생 비율	예외와 관련한 상세 목록에 표시된 모든 애플리케이션의 예외 발생 건수의 합에 대한 해당 애플리케이션의 예외 발생 건수의 비율로 단위는 퍼센트이다.
애플리케이션	예외가 발생한 애플리케이션 이름
내용	예외와 관련한 내용으로 주로 예외 메시지와 스택트레이스 등이 표시된다.

예외와 관련한 상세 목록에서 애플리케이션 필드를 클릭하면 팝업 창에서 상세한 예외 내용을 확인할 수 있다.

13.5. 보고서

제니퍼가 제공하는 보고서에 대해서 설명한다.

13.5.1. 일일 보고서

[통계 분석 | 보고서 | 일일 보고서] 메뉴는 선택한 날짜에 대한 성능 데이터를 요약해서 보고서 형태로 제공해준다.

그림 13-20:일일 보고서

상단에는 검색 조건이 있고 하단에는 요약 정보 탭과 상세 정보 탭이 있다. 요약 정보 탭은 선택한 날짜에 대한 성능 데이터를 요약한 보고서를 제공하고, 상세 정보 탭은 선택한 날짜에 대해 PERF_X_01~31 테이블에 저장되어 있는 데이터를 제공한다. 기본적으로 요약 정보 탭이 선택되어 있다.

13.5.1.1. 요약 정보

일일 보고서를 열람하려면 검색 조건에 적절한 값을 입력한 후에 오른쪽 하단에 있는 **[검색]** 버튼을 클릭한다. 다음은 검색 조건에 대한 설명이다.

- 노드 - 노드를 구성한 경우에는 제니퍼 에이전트 선택 영역에서 특정 노드를 선택하여, 해당 노드에 속하는 제니퍼 에이전트들에 대해서만 보고서를 열람할 수 있다.
- 에이전트 - 제니퍼 에이전트 선택 영역에서 보고서를 열람할 제니퍼 에이전트를 선택한다. 모든 제니퍼 에이전트들에 대해서 보고서를 열람하려면 [모두]를 선택한다.
- 날짜 - 보고서를 열람할 날짜를 선택한다.
- 시간 - 보고서를 열람할 때 사용할 성능 데이터의 시간 범위를 지정한다. 예를 들어, 22시에서 04시까지 사이에 사용자가 거의 없는 자바 애플리케이션을 모니터링하고 있다면, 시간을 04시에서 22시까지 설정하여 조회하면 정확한 성능 데이터에 대한 평균 값을 얻을 수 있다. 기본 시간 범위는 09시에서 18시이다.

일일 보고서가 제공하는 항목은 다음과 같다.

표 13-20: 일일 보고서가 제공하는 항목

보고서 항목	설명
요약 정보	선택한 날짜에 대한 호출 건수, 평균 응답 시간, 서비스 요청률, 방문자 수, 동시 단말 사용자 수, 대기 시간 등의 성능 데이터를 제공한다.
업무 현황	특정 애플리케이션 그룹에 대한 호출 건수, 평균 응답 시간, 최소 응답 시간, 최대 응답 시간, 평균 CPU 사용률 등의 성능 데이터를 제공한다.
프로그램 현황	애플리케이션, SQL, 외부 트랜잭션에 대해서 호출 건수가 많은 10개와 평균 응답 시간이 긴 10건을 제공한다.

표 13-20: 일일 보고서가 제공하는 항목

보고서 항목	설명
예외 현황	선택한 날짜에 발생한 예외 정보를 보여준다.
에이전트 별 현황	선택한 날짜에 대한 제니퍼 에이전트 별 호출 건수, 평균 응답 시간, 서비스 요청률, 동시단말 사용자 수, 액티브 서비스 개수, 시스템 CPU 사용률, 자바 힙 메모리 사용량, 액티브 JDBC 커넥션 개수 등에 대한 일일 전체 누적 값, 평균 값, 최대 값 등을 제공한다.

각 항목의 내용을 엑셀로 다운로드할 수 있다.

13.5.1.2. 상세 정보

상세 정보 탭을 선택하면 요약 정보 탭에 있던 검색 조건과 함께, 열람할 일반 성능 데이터 목록 및 정렬 방법을 선택할 수 있다.

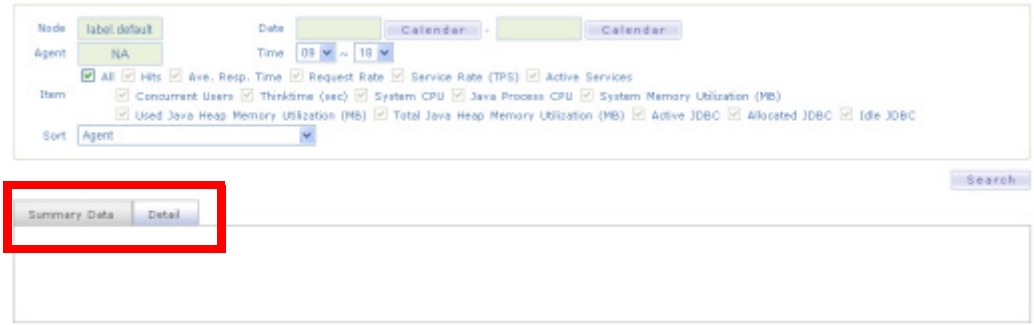
Notice: 상세 정보 탭에서는 단순하게 PERF_X_01~31 테이블에 있는 모든 데이터를 보여준다.

상세 정보 내역을 엑셀로 다운로드할 수 있다.

13.5.2.주간 보고서

[통계 분석 | 보고서 | 주간 보고서] 메뉴는 선택한 기간에 대한 성능 데이터를 요약해서 보고서 형식으로 제공한다.

그림 13-21:주간 보고서



상단에는 검색 조건이 있고 하단에는 요약 정보 탭과 상세 정보 탭이 있다. 요약 정보 탭은 선택한 기간에 대한 성능 데이터를 요약한 보고서를 제공하고, 상세 정보 탭은 선택한 기간에 대해 S_PERF_X 테이블에 저장되어 있는 데이터를 제공한다. 기본적으로 요약 정보 탭이 선택되어 있다.

13.5.2.1. 요약 정보

주간 보고서를 열람하려면 검색 조건에 적절한 값을 입력한 후에 오른쪽 하단에 있는 **[검색]** 버튼을 클릭한다. 다음은 검색 조건에 대한 설명이다.

- 노드 - 노드를 구성한 경우에는 제니퍼 에이전트 선택 영역에서 특정 노드를 선택하여, 해당 노드에 속하는 제니퍼 에이전트들에 대해서만 보고서를 열람할 수 있다.
- 에이전트 - 제니퍼 에이전트 선택 영역에서 보고서를 열람할 제니퍼 에이전트를 선택한다. 모든 제니퍼 에이전트들에 대해서 보고서를 열람하려면 [모두]를 선택한다.
- 날짜 - 보고서를 열람할 기간을 선택한다.
- 시간 - 보고서를 열람할 때 사용할 성능 데이터의 시간 범위를 지정한다. 예를 들어, 22시에서 04시까지 사이에 사용자가 거의 없는 자바 애플리케이션을 모니터링하고 있다면, 시간을 04시에서 22시까지 설정하여 조회하면 정확한 성능 데이터에 대한 평균 값을 얻을 수 있다. 기본 시간 범위는 09시에서 18시이다.

일일 보고서가 제공하는 항목은 다음과 같다.

표 13-21: 주간 보고서가 제공하는 항목

보고서 항목	설명
요약 정보	선택한 기간에 대한 호출 건수, 평균 응답 시간, 서비스 요청률, 방문자 수, 동시 단말 사용자 수, 대기 시간 등의 성능 데이터를 제공한다.
예외 현황	선택한 기간에 발생한 예외 정보를 보여준다.
에이전트 별 현황	선택한 기간에 대한 제니퍼 에이전트 별 호출 건수, 평균 응답 시간, 서비스 요청률, 동시단말 사용자 수, 액티브 서비스 개수, 시스템 CPU 사용률, 자바 힙 메모리 사용량, 액티브 JDBC 커넥션 개수 등에 대한 일일 전체 누적 값, 평균 값, 최대 값 등을 제공한다.

각 항목의 내용을 엑셀로 다운로드할 수 있다.

13.5.2.2. 상세 정보

상세 정보 탭을 선택하면 요약 정보 탭에 있던 검색 조건과 함께, 열람할 일반 성능 데이터 목록 및 정렬 방법을 선택할 수 있다.

Notice: 상세 정보탭에서는 단순하게 S_PERF_X 테이블에 있는 모든 데이터를 보여준다.

상세 정보 내역을 엑셀로 다운로드할 수 있다.

13.5.3. 월간 보고서

[통계 분석 | 월간 보고서] 메뉴는 선택한 월에 대한 성능 데이터를 요약해서 보고서 형식으로 제공한다.

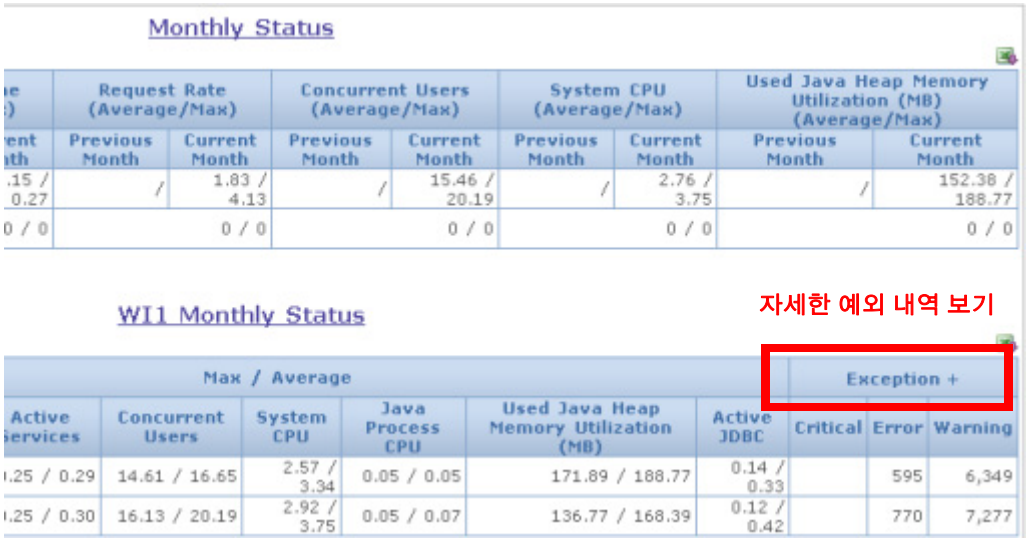
월간 보고서를 열람하려면 검색 조건에 적절한 값을 입력한 후에 오른쪽 하단에 있는 **[검색]** 버튼을 클릭한다. 다음은 검색 조건에 대한 설명이다.

- 노드 - 노드를 구성한 경우에는 제니퍼 에이전트 선택 영역에서 특정 노드를 선택하여, 해당 노드에 속하는 제니퍼 에이전트들에 대해서만 보고서를 열람할 수 있다.
- 에이전트 - 제니퍼 에이전트 선택 영역에서 보고서를 열람할 제니퍼 에이전트를 선택한다. 모든 제니퍼 에이전트들에 대해서 보고서를 열람하려면 [모두]를 선택한다.
- 년/월 - 보고서를 열람할 년과 월을 선택한다.
- 시간 - 보고서를 열람할 때 사용할 성능 데이터의 시간 범위를 지정한다. 예를 들어, 22시에서 04시까지 사이에는 사용자가 거의 없는 자바 애플리케이션을 모니터링하고 있다면, 시간을 04시에서 22시까지 설정하여 조회하면 정확한 성능 데이터에 대한 평균 값을 얻을 수 있다. 기본 시간 범위는 09시에서 18시이다.

이 보고서는 제니퍼 에이전트 별 호출 건수, 평균 응답 시간, 서비스 요청률, 동시단말 사용자 수, 시스템 CPU 사용률, 자바 힙 메모리 사용량 등에 대한 월간 전체 누적 값, 평균 값, 최대 값 등을 전월 대비로 제공한다. 또한 전월 대비 증감율도 함께 표시를 해준다.

그리고 제니퍼 에이전트 별로 선택한 월에 대한 일자별 호출 건수, 평균 응답 시간, 서비스 요청률, 액티브 서비스 개수, 동시단말 사용자 수, 시스템 CPU 사용률, 자바 프로세스 CPU 사용률, 자바 힙 메모리 사용량, 액티브 JDBC 커넥션 개수 등에 대한 일일 전체 누적 값, 평균 값, 최대 값 등을 경보 내역과 함께 제공한다.

그림 13-22:월간 보고서



Monthly Status

	Request Rate (Average/Max)		Concurrent Users (Average/Max)		System CPU (Average/Max)		Used Java Heap Memory Utilization (MB) (Average/Max)	
	Previous Month	Current Month	Previous Month	Current Month	Previous Month	Current Month	Previous Month	Current Month
.15 / 0.27	/	1.83 / 4.13	/	15.46 / 20.19	/	2.76 / 3.75	/	152.38 / 188.77
0 / 0		0 / 0		0 / 0		0 / 0		0 / 0

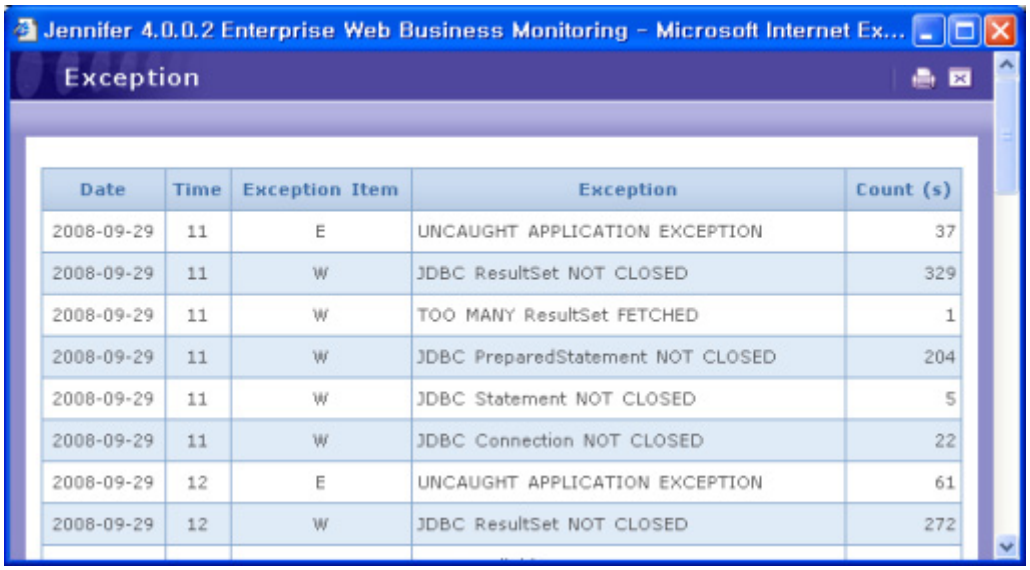
WI1 Monthly Status

자세한 예외 내역 보기

Max / Average						Exception +		
Active Services	Concurrent Users	System CPU	Java Process CPU	Used Java Heap Memory Utilization (MB)	Active JDBC	Critical	Error	Warning
.25 / 0.29	14.61 / 16.65	2.57 / 3.34	0.05 / 0.05	171.89 / 188.77	0.14 / 0.33		595	6,349
.25 / 0.30	16.13 / 20.19	2.92 / 3.75	0.05 / 0.07	136.77 / 168.39	0.12 / 0.42		770	7,277

- 자세한 예외 내역 보기 - 해당 링크를 클릭하면 자세한 예외 내역을 확인할 수 있다.

그림 13-23:경보 내역 팝업 창



Jennifer 4.0.0.2 Enterprise Web Business Monitoring - Microsoft Internet Ex...

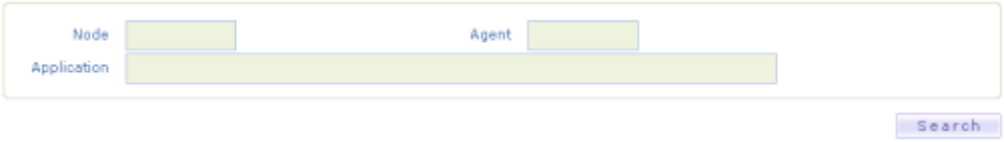
Exception

Date	Time	Exception Item	Exception	Count (s)
2008-09-29	11	E	UNCAUGHT APPLICATION EXCEPTION	37
2008-09-29	11	W	JDBC ResultSet NOT CLOSED	329
2008-09-29	11	W	TOO MANY ResultSet FETCHED	1
2008-09-29	11	W	JDBC PreparedStatement NOT CLOSED	204
2008-09-29	11	W	JDBC Statement NOT CLOSED	5
2008-09-29	11	W	JDBC Connection NOT CLOSED	22
2008-09-29	12	E	UNCAUGHT APPLICATION EXCEPTION	61
2008-09-29	12	W	JDBC ResultSet NOT CLOSED	272

13.5.4.애플리케이션 보고서

[통계 분석 | 보고서 | 애플리케이션] 메뉴는 특정 애플리케이션에 대한 성능 데이터를 차트로 확인할 수 있도록 제공한다.

그림 13-24:애플리케이션 보고서 검색 조건



애플리케이션 성능 데이터를 조회하려면 검색 조건에 적절한 값을 입력한 후에 오른쪽 하단에 있는 **[검색]** 버튼을 클릭한다. 다음은 검색 조건에 대한 설명이다.

- 노드 - 노드를 구성한 경우에는 제니퍼 에이전트 선택 영역에서 특정 노드를 선택하여, 해당 노드에 속하는 제니퍼 에이전트들에 대해서만 애플리케이션에 대한 성능 데이터를 조회할 수 있다.
- 에이전트 - 제니퍼 에이전트 선택 영역에서 애플리케이션에 대한 성능 데이터를 조회할 제니퍼 에이전트를 선택한다. 모든 제니퍼 에이전트들에 대해서 애플리케이션에 대한 성능 데이터를 조회하려면 [모두]를 선택한다.
- 애플리케이션 - 애플리케이션 입력 필드를 클릭하면 팝업 창이 나타나고 이 팝업 창을 통해서 성능 데이터를 검색할 애플리케이션을 지정한다.

첫번째 차트는 해당 애플리케이션에 대한 호출 건수, 실패 건수, 응답 시간의 합, 평균 응답 시간, 최소 응답, 최대 응답, CPU 시간, CPU(tpmC) 등을 일자별로 보여준다. 첫번째 차트에서 특정 일을 클릭하면 두번째 차트에 선택한 날짜에 대한 데이터가 막대 차트 혹은 라인 차트로 표시된다.

13.6. 보고서 템플릿

성능 데이터를 관계형 데이터베이스에 저장하기 때문에 성능 데이터에 대한 다양한 분석이 가능하고, 이를 활용하여 보고서 템플릿을 작성할 수 있다.

보고서 템플릿은 사용자 정의 파라미터와 SQL을 기반으로, RTF 형식과 HTML 형식의 보고서를 제공한다. RTF는 출력이나 보관용으로 사용하고 HTML은 주로 조회용으로 사용한다.

13.6.1.보고서 템플릿에 대한 이해

보고서 템플릿은 보고서에 대한 기본 정보와 여러 개의 아이템으로 구성된 실행할 수 있는 보고서이다. 그리고 아이템은 특정 성능 데이터를 SQL로 추출하여, 그 결과를 차트나 테이블 등의 양식으로 표시하는 방법을 정의한다. 예를 들어, 특정 날짜의 서비스 처리율

을 라인 차트로 표시하려면 서비스 처리율을 추출하는 SQL과 유형이 TIME-LINE인 아이템을 작성해야 한다.

보고서 템플릿을 별도의 파일로 Import/Export하여 재활용할 수 있다. 그래서 우선 제니퍼가 제공하는 예제 보고서 템플릿 파일을 Import하여 보고서 템플릿을 이해하도록 한다.

- 먼저 **[통계 | 보고서 | 보고서 템플릿]** 메뉴로 이동한다.
- 보고서 템플릿 목록 하단에 있는 버튼을 통해서 JENNIFER_HOME/server/doc/report/SampleDailyReport.dat 파일을 import한다. 이 파일은 특정 날짜에 대한 성능 데이터를 보여주는 일일 보고서의 하나이다. 확장자가 dat으로 끝나는 파일만을 업로드할 수 있다.

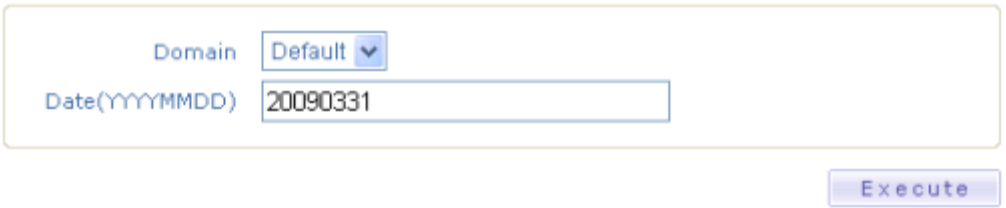
그림 13-25:보고서 템플릿 Import 폼

- 보고서 템플릿을 Import하면 보고서 템플릿 목록에 Import한 보고서 템플릿이 나타난다. 이 보고서 템플릿의 제목은 Daily Report이다.

그림 13-26:보고서 템플릿 목록

- 보고서 템플릿 목록에서 **[보기]** 링크를 클릭하면, 보고서 템플릿을 실행할 수 있는 화면으로 이동한다.

그림 13-27:보고서 검색(사용자 정의 파라미터) 조건

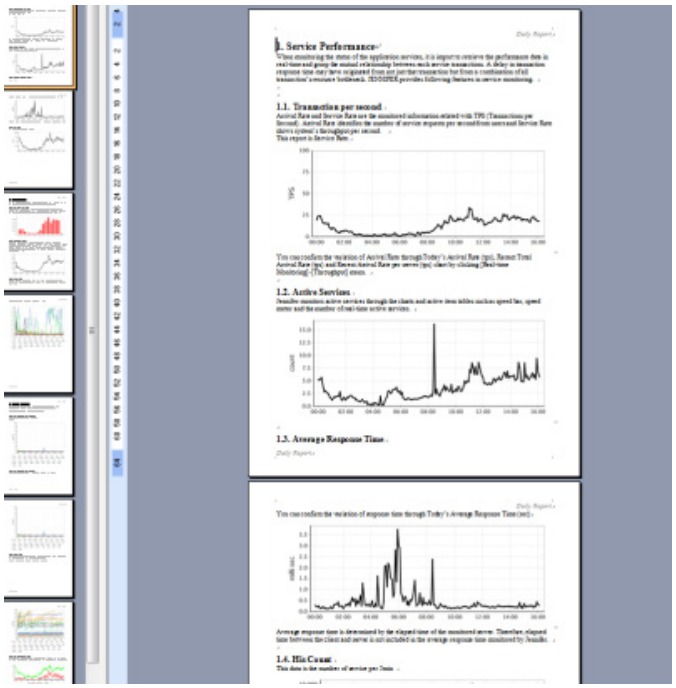


- 이 화면에서 검색 조건을 입력한 후에 **[실행]** 버튼을 클릭하면 하단에 HTML 형식의 보고서가 나타난다.

그림 13-28:HTML 양식의 보고서

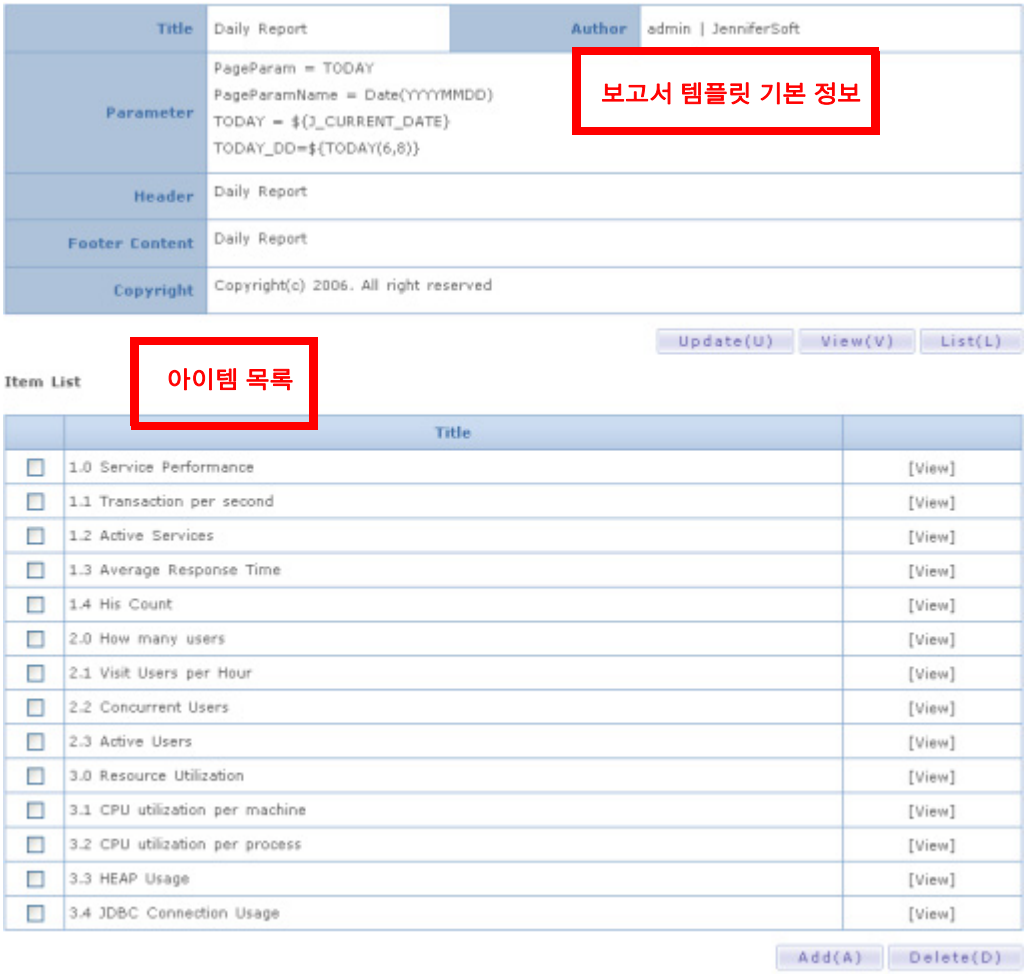
- 보고서 템플릿 목록에서 **[리포트 다운로드]** 링크를 클릭하면, 보고서 템플릿이 실행되면서 RTF 양식의 보고서가 다운로드된다.

그림 13-29:RTF 양식의 보고서



- 보고서 템플릿 목록에서 템플릿 제목을 클릭하면, 보고서 템플릿을 구성하는 기본 정보와 아이템 목록을 확인할 수 있다.

그림 13-30:보고서 템플릿 구성



Title	Daily Report
Author	admin | JenniferSoft
Parameter	PageParam = TODAY PageParamName = Date(YYYYMMDD) TODAY = ${J_CURRENT_DATE} TODAY_DD=${TODAY(6,8)}
Header	Daily Report
Footer Content	Daily Report
Copyright	Copyright(c) 2006. All right reserved

Update(U) View(V) List(L)

Item List

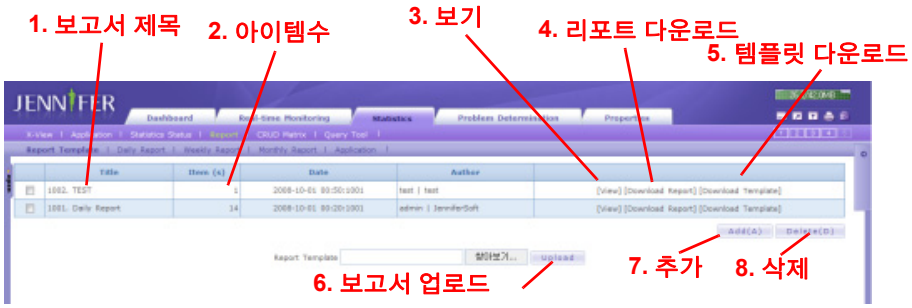
	Title	
☐	1.0 Service Performance	[View]
☐	1.1 Transaction per second	[View]
☐	1.2 Active Services	[View]
☐	1.3 Average Response Time	[View]
☐	1.4 His Count	[View]
☐	2.0 How many users	[View]
☐	2.1 Visit Users per Hour	[View]
☐	2.2 Concurrent Users	[View]
☐	2.3 Active Users	[View]
☐	3.0 Resource Utilization	[View]
☐	3.1 CPU utilization per machine	[View]
☐	3.2 CPU utilization per process	[View]
☐	3.3 HEAP Usage	[View]
☐	3.4 JDBC Connection Usage	[View]

Add(A) Delete(D)

13.6.2.보고서 템플릿 관리

다음으로는 보고서 템플릿을 관리하고 작성하는 방법을 설명한다. 우선 보고서 템플릿 목록 화면에서 보고서 템플릿을 관리하는 방법을 설명한다.

그림 13-31:템플릿 리스트



1. 보고서 제목 - 보고서 템플릿의 제목으로 해당 칼럼을 클릭하면 해당 보고서 템플릿을 편집할 수 있는 상세 화면으로 이동한다. 이 이름을 보고서 자체의 이름으로도 사용한다.
2. 아이템 수 - 하나의 보고서 템플릿은 1개 이상의 아이템으로 구성된다. 이 칼럼에서 보고서 템플릿의 총 아이템 수를 보여준다.
3. 보기 - 보고서 템플릿이 실행되면서 HTML 양식의 보고서가 화면에 나타난다.
4. 리포트 다운로드 - 보고서 템플릿이 실행되면서 RTF 양식의 보고서가 다운로드된다.
5. 템플릿 다운로드 - 공유 및 재사용을 위해서 보고서 템플릿 파일을 다운로드한다. 그러면 이 보고서 템플릿 파일을 다른 제니퍼 서버에서 사용할 수 있다.
6. 보고서 업로드 - 보고서 템플릿 파일을 업로드할 때 사용한다.
7. 추가 버튼 - 해당 버튼을 클릭하면 새로운 보고서 템플릿을 구성할 수 있는 화면으로 이동한다.
8. 삭제 버튼 - 보고서 템플릿 목록에서 삭제할 보고서 템플릿의 첫번째 칼럼의 체크 박스를 선택한 후에 해당 버튼을 클릭하면 보고서 템플릿이 삭제된다.

13.6.3.보고서 템플릿 작성과 사용자 정의 파라미터

새로운 보고서 템플릿을 작성하려면 보고서 템플릿 목록 화면 하단에 있는 **[추가]** 버튼을 클릭하여 보고서 템플릿 입력 화면으로 이동한다. 보고서 템플릿 입력 화면에서 보고서에 대한 기본 정보와 보고서 템플릿을 실행하는데 필요한 공통 파라미터를 설정한다.

다음은 보고서 템플릿 입력 필드에 대한 설명이다.

- 제목 - 보고서 템플릿의 제목으로 필수 입력 항목이다.
- 사용자 아이디 - 보고서 템플릿 작성자 아이디로 선택 입력 항목이다.
- 사용자 이름 - 보고서 템플릿 작성자 이름으로 선택 입력 항목이다.
- 파라미터 - 보고서 템플릿을 실행하는데 필요한 공통 파라미터로 선택 입력 항목이다.
- 머리글 - 보고서 전체의 머리글로 선택 입력 항목이다.

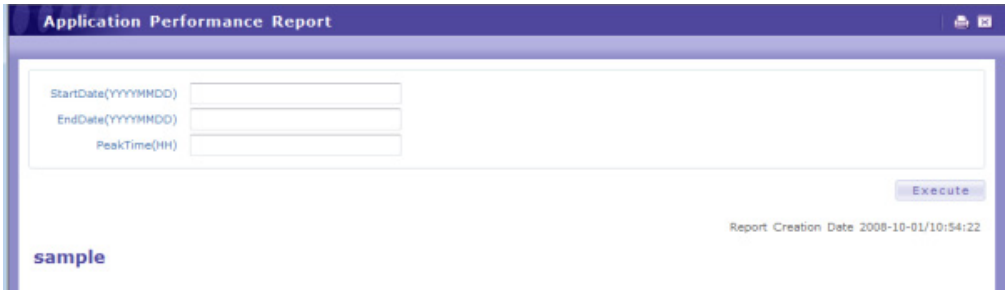
- 바닥글 - 보고서 전체의 바닥글로 선택 입력 항목이다.
- 저작권 - 저작권으로 선택 입력 항목이다.

보고서 템플릿을 사용하려면 파라미터를 명확하게 이해해야 한다. 보고서 템플릿에서 사용할 사용자 정의 파라미터를 설정할 수 있다. 우선 PageParam 옵션으로 사용자 정의 파라미터 키를 콤마[,]를 구분자로 입력한다. 그리고 PageParamName 옵션으로 사용자 정의 파라미터 키에 대한 이름을 설정한다. 이 이름은 사용자 정의 파라미터 입력 창에서 사용된다.

예를 들어, startday, endday, peaktime 등의 사용자 정의 파라미터를 StartDate(YYYYMMDD), EndDate(YYYYMMDD), PeakTime(HH) 등의 이름으로 사용자로부터 입력받으려면 보고서 템플릿의 파라미터 입력 필드에 다음을 입력한다.

```
PageParam = startday,endday,peaktime
PageParamName = StartDate(YYYYMMDD),EndDate(YYYYMMDD),PeakTime(HH)
```

그림 13-32:입력 파라미터



PageParam 옵션으로 설정한 사용자 정의 파라미터의 기본 값을 설정할 수도 있다. 예들들어서 사용자 정의 파라미터 startday, endday, peaktime 등의 기본 값을 20080923, 20080929, 10으로 설정하려면 다음과 같이 한다.

```
PageParam = startday,endday,peaktime
PageParamName = StartDate(YYYYMMDD),EndDate(YYYYMMDD),PeakTime(HH)

startday = 20080923
endday = 20080929
peaktime = 10
```

사용자 정의 파라미터에 대한 초기 값으로 상수가 아닌 예약 파라미터를 사용할 수 있다. 다음은 사용 가능한 예약 파라미터 목록이다.

표 13-22: 예약 파라미터

변수명	설명
J_CURRENT_DATE	yyyyMMdd 포멧의 오늘 날짜
J_CURRENT_TIME	HHmmss 포멧의 현재 시간
J_BEFORE_WEEK	yyyyMMdd 포멧의 일주일 전 날짜
J_BEFORE_MONTH	yyyyMMdd 포멧의 한달 전 날짜
J_BEFORE_DATE	yyyyMMdd 포멧의 어제 날짜

예약 파라미터는 다음과 같이 사용할 수 있다.

```
PageParam = TODAY
PageParamName = Date(YYYYMMDD)

TODAY = ${J_CURRENT_DATE}
```

또한 PageParam 옵션으로 정의한 파라미터를 기반으로 파생 사용자 정의 파라미터를 설정할 수 있다. 예를 들어, 사용자로부터는 TODAY를 입력받고 이를 기반으로 dd 포멧의 일자 사용자 정의 파생 파라미터를 설정하려면 다음과 같이 한다.

```
PageParam = TODAY
PageParamName = Date(YYYYMMDD)

TODAY = ${J_CURRENT_DATE}
TODAY_DD = ${TODAY(6,2)}
```

TODAY_DD는 ${parameter(start, length)} 형식의 연산자를 통해서 TODAY 파라미터 값의 일부분을 그 값으로 한다.

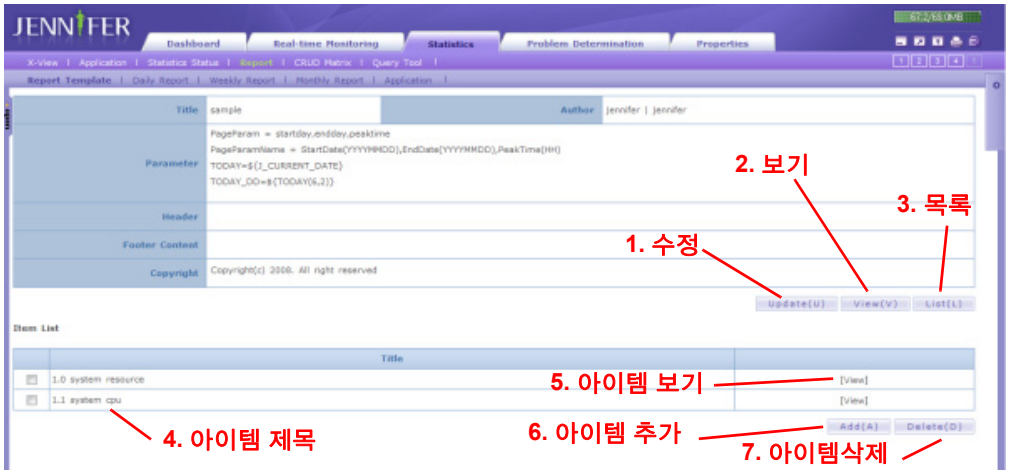
이렇게 정의한 사용자 정의 파라미터는 아이템을 구성하는 SQL에서 다음과 같이 사용할 수 있다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       SERVICE_RATE
  FROM PERF_X_${TODAY_DD}
 WHERE LOG_DT = '${TODAY}'
   AND AGENT_ID = 'TOT'
```

13.6.4.아이템 작성

보고서 템플릿의 실질적인 내용은 아이템을 통해서 구성한다. 일반적으로 보고서 템플릿은 여러 개의 아이템으로 구성된다. 아이템 관리 방법은 다음과 같다.

그림 13-33:보고서 아이템 관리



1. 수정 - 해당 버튼을 클릭하면 보고서 템플릿의 내용을 수정할 수 있는 보고서 템플릿 입력 화면으로 이동한다.
2. 보기 - 해당 버튼을 클릭하면 보고서 템플릿이 실행되면서 HTML 양식의 보고서가 팝업 창에 나타난다.
3. 목록 - 해당 버튼을 클릭하면 보고서 템플릿 목록 화면으로 이동한다.
4. 아이템 제목 - 아이템의 제목으로, 클릭하면 해당 아이템의 내용을 수정할 수 있는 입력 화면이 화면 오른 쪽에 나타난다.
5. 아이템 보기 - 링크를 클릭하면 전체 보고서가 아닌 해당 아이템만을 실행시킬 수 있는 새로운 팝업 창이 나타난다.
6. 아이템 추가 - 해당 버튼을 클릭하면 새로운 아이템을 추가할 수 있는 입력 화면이 화면 오른 쪽에 나타난다.
7. 아이템 삭제 - 아이템 목록에서 삭제할 아이템의 첫번째 칼럼의 체크 박스를 선택한 후에 해당 버튼을 클릭하면 아이템이 삭제된다.

다음은 아이템 입력 필드에 대한 설명이다.

- Chapter/Section/제목 - Chapter와 Section에는 숫자를 입력한다. 아이템 목차는 2 단계만 가능하고 1.0은 새로운 CHAPTER의 시작이고 1.1은 1.0의 하위 SECTION이다. 이를 통해서 아이템의 순서가 정렬된다. 그리고 제목에는 아이템의 제목을 입력한다.
- 유형 - SQL로 추출한 데이터를 표시할 방법을 선택한다.

- 머리글 - 아이템의 머리글로 선택 입력 항목이다.
- 파라미터 - 유형에 따른 결과를 제어할 때 사용하는 옵션을 입력한다.
- SQL 쿼리 - 데이터를 추출할 SQL을 입력한다.
- 반복 파라미터 - SQL을 반복적으로 수행할 반복 파라미터를 입력한다.
- 바닥글 - 아이템의 바닥글로 선택 입력 항목이다.

다음은 아이템 유형에 대한 설명이다.

표 13-23: 보고서 아이템 유형

아이템 명	설명
NONE	아이템을 사용하지 않는다. 주로 머리글 혹은 바닥글을 이용해서 텍스트로 보고서에 대한 설명을 제공하는 아이템의 경우에 이 유형을 사용한다.
LINE	SQL 결과를 라인 차트로 보여준다.
TIME-LINE	SQL 결과를 시계열 차트로 보여준다.
3D LINE	SQL 결과를 3D 효과를 준 라인 차트로 보여준다.
PIE	SQL 결과를 파이 차트로 보여준다.
BAR	SQL 결과를 막대 차트로 보여준다.
3D BAR	SQL 결과를 3D 효과를 준 막대 차트로 보여준다.
STACKED BAR	SQL 결과를 계층적 막대 차트로 보여준다.
3D STACKED BAR	SQL 결과를 3D 효과를 준 계층적 막대 차트로 보여준다.
LIST(EXCEL TYPE)	SQL 결과를 테이블 형식으로 보여준다.
SQL Query(Query Only)	보고서 작성에 임시적으로 필요한 쿼리를 수행한다. 보고서에는 아무런 내용도 나타나지 않는다.

파라미터를 통해서 아이템의 옵션을 조정할 수 있다. 다음은 주요 파라미터에 대한 설명이다.

표 13-24: 아이템 파라미터

아이템 파라미터	설명
GraphTitle	차트의 제목
GraphWidth	차트의 넓이
GraphHeight	차트의 높이
GraphMaxY	차트의 Y 축 최대 값

표 13-24: 아이템 파라미터

아이템 파라미터	설명
GraphLabelX	차트 X 축 범례로, 2개 이상인 경우에는 콤마를 구분자로 구분한다. 그리고 **#SQL**로 설정하면 SELECT 절의 두번째 칼럼을 범례로 사용한다.
GraphLabelY	차트의 Y 축 이름
GraphPieExplode	파이 차트에서 특정 항목을 강조하려면, 해당 항목의 이름을 설정한다.
GraphShowValue	막대 차트에 값을 표시하려면 true를, 표시하지 않으려면 false를 입력한다.
GraphShowLegend	차트 범례의 표시 여부
GraphShapeVisible	차트 각 포인트 별 강조점 표시 여부로 TIME-LINE 유형에서만 사용 가능하다.
GraphItemLabelRotation	X축 라벨의 회전도(기본값은0.523)
MaxRows	차트를 표시하기 위한 레코드의 최대 크기로 설정하지 않으면 제니퍼 서버의 default_max_rows_for_report_item 옵션으로 설정한 값을 따라간다. 기본 값은 3600이다.
GraphLineWidth	차트 라인 두께로 기본값은 2.0이고 LINE, TIME-LINE 유형에서만 사용 가능하다.
datadb_url	외부 데이터베이스 연결을 위한 JDBC URL
datadb_driver	외부 데이터베이스 연결을 위한 Driver 클래스 이름
datadb_user	외부 데이터베이스 연결을 위한 사용자 계정
datadb_password	외부 데이터베이스 연결을 위한 패스워드
TableColumnAlign	아이템 유형으로 List(Excel Type)를 선택한 경우에, 각 칼럼 좌우 정렬 기준을 지정한다. left, center, right 등을 콤마를 구분자로 지정한다
TableColumnValign	아이템 유형으로 List(Excel Type)를 선택한 경우에, 각 칼럼 상하 정렬 기준을 지정한다. top, middle, bottom 등을 콤마를 구분자로 지정한다.
TableColumnWidth	아이템 유형으로 List(Excel Type)를 선택한 경우에, 각 칼럼 넓이를 지정한다.
TableLabel	아이템 유형으로 List(Excel Type)를 선택한 경우에, 기본적으로 테이블 칼럼 이름은 SQL 칼럼 이름이 된다. 테이블 칼럼 이름을 별도의 이름으로 지정할 때 TableLabel 파라미터를 사용한다. 콤마를 구분자로 구분한다

Notice: 아이템 파라미터를 보고서 템플릿 파라미터 입력 필드에도 지정할 수 있다. 그 경우에는 모든 아이템에 해당 파라미터가 적용된다.

아이템 입력 화면의 SQL 필드에 아이템을 구성하는데 필요한 데이터를 가져오는 쿼리를 입력한다. SQL 필드 오른쪽 아래에 있는 **[실행]** 링크를 클릭하면 입력한 쿼리를 바로 테스트할 수 있다.

그림 13-34:아이템 SQL

13.6.5.SQL 작성 요령

각 아이템은 SQL을 실행하여 차트 혹은 테이블에 표현할 데이터를 조회한다. 따라서 정해진 형식을 준수하는 SQL을 사용해야 한다.

13.6.5.1. 주요 테이블에 대한 이해

아이템을 작성하려면 성능 데이터베이스 테이블 구조를 이해해야 한다. 이에 대한 자세한 사항은 테이블 스키마를 참조한다. 여기서는 몇 가지 주요 작성 요령을 설명한다.

자주 사용하는 테이블은 PERF_X_01~31, APPL_10M_01~31, SQLS_10M_01~31, TX_10M_01~31 등의 일자별 테이블이다. 이 테이블들은 성능 향상을 위해서 일자별로 분리되어 있다. 즉, PERF_X_01~31 테이블은 존재하지 않고, 일자별로 PERF_X_01, PERF_X_02 등의 테이블이 존재한다. 따라서 SQL에서 일자별 테이블을 사용하는 경우에는 기본적으로 보고서 템플릿 파라미터를 다음과 같이 설정한다.

```
PageParam = TODAY
PageParamName = Date(YYYYMMDD)
TODAY = ${J_CURRENT_DATE}
TODAY_DD=${TODAY(6,8)}
```

그리고 SQL에서 일자별 테이블은 사용자 정의 파생 파라미터 TODAY_DD를 이용해서 다음과 같이 작성한다

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       SERVICE_RATE
  FROM PERF_X_${TODAY_DD}
 WHERE AGENT_ID = 'W11'
```

일자별 테이블로는 특정 일자 데이터만을 조회할 수 있다. 그런데 주간 보고서와 같이 여러 날짜에 걸친 데이터를 조회하려면 S_PERF_X 등과 같이 S_로 시작하는 통계 요약 테이블을 사용한다.

Notice: 일반 일자별 테이블에는 5분 혹은 10분 단위로 데이터가 저장되어 있는 반면에 S_로 시작하는 통계 요약 테이블에는 1시간 단위로 데이터가 저장되어 있다. 그리고 SummaryActor 스케줄러가 일자별 테이블에 있는 데이터를 하루에 한번 통계 요약 테이블에 저장하기 때문에 오늘 날짜 데이터는 통계 요약 테이블에서 조회할 수 없다.

성능 데이터 보관 주기가 1달을 넘는 경우에는 일자별 테이블에 월이 다른 데이터가 저장되어 있을 수 있다. 따라서 SQL을 작성할 때 다음 조건을 추가하도록 한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       SERVICE_RATE
  FROM PERF_X_${TODAY_DD}
 WHERE AGENT_ID = 'W11'
   AND LOG_DT = '${TODAY}'
```

AGENT_ID 칼럼을 올바르게 사용해야 한다. W11과 같이 특정 제니퍼 에이전트에 대한 성능 데이터를 조회하려면 다음과 같이 검색 조건을 설정하면 된다.

```
 WHERE AGENT_ID = 'W11'
```

In additon그리고 전체 성능 데이터를 조회하려면 TOT라는 가상 제니퍼 에이전트 아이디를 사용한다.

```
 WHERE AGENT_ID = 'TOT'
```

따라서 실제 제니퍼 에이전트 아이디로 TOT를 사용해서는 안된다.

그리고 모든 개별 제니퍼 에이전트에 대해서 성능 데이터를 조회하려면 TOT와 @로 시작하는 에이전트 그룹 아이디를 제외해야 한다. 따라서 SQL을 다음과 같이 작성한다.

```
 WHERE AGENT_ID <> 'TOT'
   ADN AGENT_ID NOT LIKE '@%'
```

일자별 테이블에서 시간당 데이터를 조회할 때는 데이터 칼럼에 따라서 다른 그룹 함수를 사용해야 한다. 예를 들어, 시간당 방문자 수는 동일 시간대에 대해서는 5분 마다 그 시간대 누적 값을 저장하기 때문에 MAX 함수를 사용한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       MAX(VISIT_HOUR)
  FROM PERF_X_${TODAY_DD}
 WHERE AGENT_ID = 'TOT'
   AND LOG_DT = '${TODAY}'
GROUP BY LOG_HH
```

그리고 호출 건수는 SUM 함수를 사용한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       AVG(HIT)
  FROM PERF_X_${TODAY_DD}
 WHERE AGENT_ID = 'TOT'
   AND LOG_DT = '${TODAY}'
GROUP BY LOG_HH
```

그리고 나머지는 AVG 함수를 사용한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       AVG(SERVICE_RATE)
  FROM PERF_X_${TODAY_DD}
 WHERE AGENT_ID = 'TOT'
   AND LOG_DT = '${TODAY}'
GROUP BY LOG_HH
```

애플리케이션별 성능 데이터는 APP_10M_01~31, SQL_10M_01~31, TX_10M_01~31 등의 일자별 테이블에서 조회한다. 이 테이블들은 구조가 단순함으로 APPL_10M_01-31, SQLS_10M_01-31, TX_10M_1-31 등을 참조하도록 한다.

Notice: 애플리케이션 관련 일자별 테이블에는 10분 단위로 데이터가 저장된다.

그런데 데이터 양의 축소와 성능 향상을 위해서 애플리케이션 성능 데이터 테이블은 메타 테이블을 사용한다. 예를 들어, SQL 문장을 SQLS_10M_01~31 테이블에 그대로 저장하면 데이터 양이 크게 증가할 수 있다. 그래서 SQLS_10M_01~31 테이블에는 해쉬 값만을 저장하고 실제 SQL 쿼리는 SQLS 테이블에 저장한다. 따라서 조회시에는 해당 애플리케이션 성능 데이터 테이블과 메타 테이블을 조인해야 한다. 애플리케이션 메타 테이블은 다음과 같다.

표 13-25: 애플리케이션 메타 테이블

테이블	설명
APPLS	애플리케이션 이름이 저장된다.
SQLS	SQL 문장이 저장된다.
TXNAMES	외부 트랜잭션 이름이 저장된다.
ERRORS	예외 메시지가 저장된다.
METHODS	자바 메소드 이름이 저장된다.

13.6.5.2. 시계열 차트를 위한 SQL 작성

시계열 차트는 TIME-LINE 유형의 아이템을 의미한다. 시계열 차트에서 사용하는 SQL은 시간 값을 가진 순차 데이터를 조회해야 한다. 예를 들어, 특정 날짜의 서비스 요청률을 시계열 차트로 표현하려면 다음과 같은 SQL을 사용한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       SERVICE_RATE
  FROM PERF_X_${TODAY_DD}
 WHERE LOG_DT = '${TODAY}'
   AND AGENT_ID = 'TOT'
```

SELECT 절의 첫번째 칼럼에서는 시계열 차트의 X 축에 표현할 날짜/시간을 조회한다. 가능한 포멧은 다음과 같다

```
yyyyMM
yyyyMMdd
yyyyMMddHH
yyyyMMddHHmm
yyyyMMddHHmmss
```

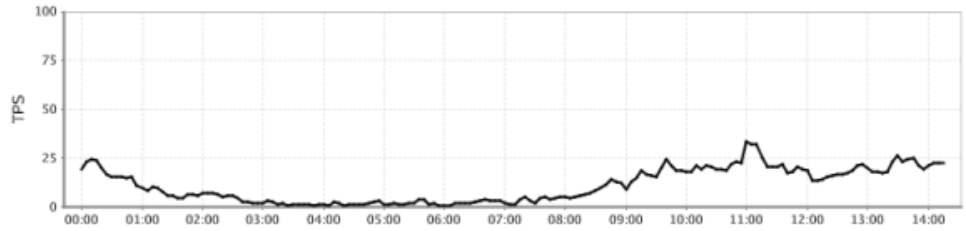
그리고 두번째 칼럼부터는 시계열 차트의 Y 축에 표현할 값을 조회한다. 2개 이상의 칼럼을 사용하면 선이 2개 이상 나타난다.

범례는 아이템 파라미터의 GraphLabelX 옵션과 GraphShowLegend 옵션으로 설정한다.

```
GraphLabelX = TPS
GraphShowLegend = true
```

이 SQL을 수행한 결과는 다음과 같다.

그림 13-35:시계열 차트(1)



시계열 차트에 서비스 요청률과 함께 동시단말 사용자 수를 표시하려면 다음 SQL을 사용한다.

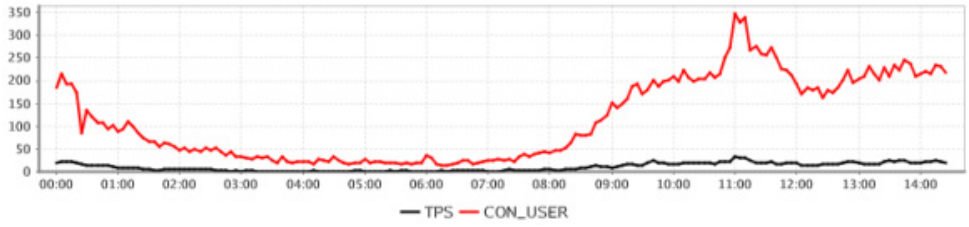
```
SELECT LOG_DT || LOG_HH || LOG_MM,
       SERVICE_RATE,
       CONCURRENT_USER
  FROM PERF_X_${TODAY_DD}
 WHERE LOG_DT = '${TODAY}'
   AND AGENT_ID = 'TOT'
```

범례는 아이템 파라미터의 GraphLabelX 옵션과 GraphShowLegend 옵션으로 설정한다.

```
GraphLabelX = TPS,CON_USER
GraphShowLegend = true
```

이 SQL을 수행한 결과는 다음과 같다.

그림 13-36:시계열 차트(2)



그런데 범례를 설정으로 고정하지 않고 데이터에 의해서 결정하려면 다음과 같이 아이템 파라미터 옵션을 설정한다.

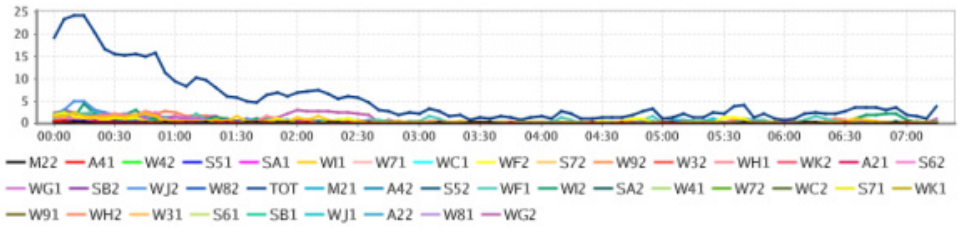
```
GraphLabelX = #SQL
GraphShowLegend = true
```

예를 들어, 시계열 차트에 제니퍼 에이전트 별 서비스 요청률을 표현하려면 다음 SQL을 사용한다. 여기서 두번째 칼럼이 범례가 된다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       AGENT_ID,
       SERVICE_RATE
  FROM PERF_X_${TODAY_DD}
 WHERE LOG_DT = '${TODAY}'
   AND AGENT_ID <> 'TOT'
   AND AGENT_ID NOT LIKE '@%'
```

이 SQL을 수행한 결과는 다음과 같다.

그림 13-37:시계열 차트(3)



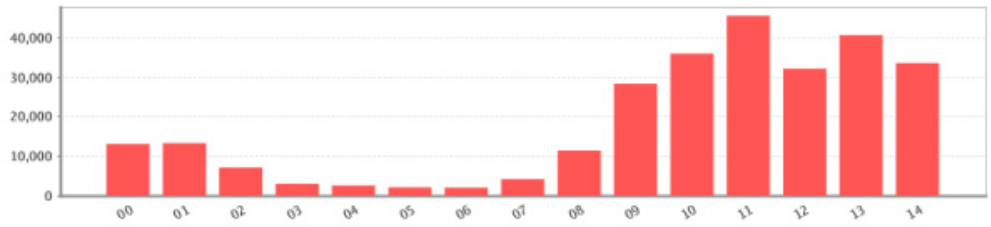
13.6.5.3. 시리즈 차트를 위한 SQL 작성

BAR 유형의 막대 차트와 LINE 유형의 라인 차트 등을 사용하는 경우의 SQL 작성 요령을 설명한다. 시리즈 차트에는 범례가 존재하지 않는다. 따라서 SQL에서 첫번째 칼럼은 X축이 되고 나머지 칼럼들이 데이터를 의미하게 된다. 예를 들어, 임의의 날짜의 시간당 방문자 수를 막대 차트로 표현하려면 다음 SQL을 사용한다.

```
SELECT X.A, SUM(X.B)
  FROM
       (
        SELECT LOG_HH A,
               VISIT_HOUR B
          FROM PERF_X_${TODAY_DD}
         WHERE AGENT_ID = 'TOT'
           AND LOG_DT = '${TODAY}'
       ) X
GROUP BY X.A
ORDER BY 1
```

이 SQL을 수행한 결과는 다음과 같다.

그림 13-38:막대 차트 (1)

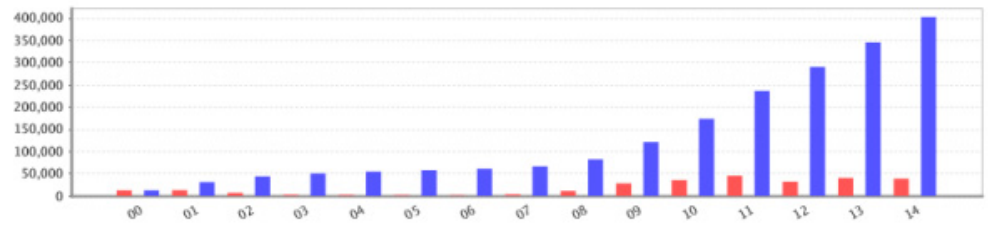


만약 SQL에 칼럼을 추가하면 막대가 추가된다. 예를 들어, 시간당 방문자 수와 함께 시간당 호출 건수를 표현하려면 다음 SQL을 사용한다.

```
SELECT X.A, SUM(X.B), SUM(X.C)
  FROM
       (
        SELECT LOG_HH A,
               VISIT_HOUR B,
               HIT C
          FROM PERF_X_${TODAY_DD}
         WHERE AGENT_ID = 'TOT'
           AND LOG_DT = '${TODAY}'
       ) X
GROUP BY X.A
ORDER BY 1
```

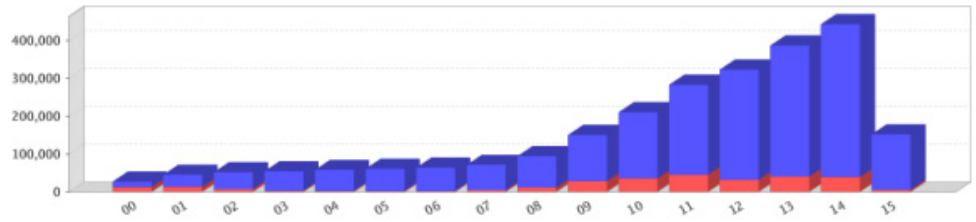
이 SQL을 수행한 결과는 다음과 같다.

그림 13-39:막대 차트(2)



유형으로 3D STACKED BAR를 선택하면 다음 그림처럼 표시된다.

그림 13-40:3D 누적 막대 차트



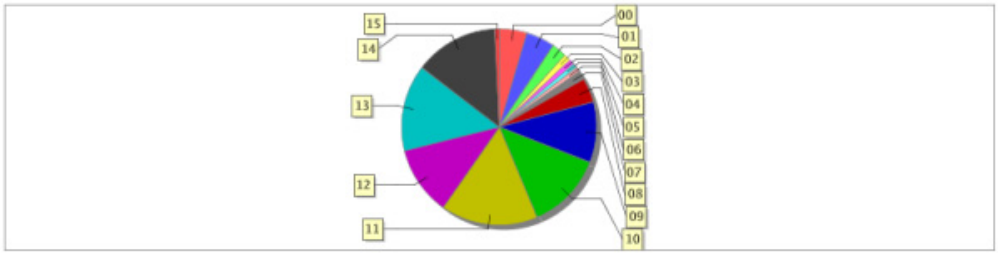
13.6.5.4. 1차원 차트를 위한 SQL 작성

1차원 차트는 1개 칼럼의 데이터만을 표시하는 차트이다. PIE 유형이 여기에 속한다. 파이 차트를 위한 SQL에는 2개의 칼럼이 필요하다. 첫번째 칼럼은 항목을, 두번째 칼럼은 데이터를 의미한다. 예를 들어, 임의의 날짜의 시간당 방문자 수를 막대 차트로 표현하려면 다음 SQL을 사용한다.

```
SELECT X.A, SUM(X.B)
  FROM
       (
        SELECT LOG_HH A,
               VISIT_HOUR B
          FROM PERF_X_${TODAY_DD}
         WHERE AGENT_ID = 'TOT'
           AND LOG_DT = '${TODAY}'
       ) X
GROUP BY X.A
ORDER BY 1
```

이 SQL을 수행한 결과는 다음과 같다.

그림 13-41:파이 차트



13.6.5.5. 테이블 아이템 유형

테이블 아이템 유형을 위한 SQL에는 제약이 없다. 단, SQL에서 조회하는 칼럼의 숫자와 아이템 파라미터의 TableLabel 옵션으로 설정한 라벨의 숫자가 동일해야 한다.

```
SELECT HASH, NAME FROM APPLS
```

예를 들어, 앞의 SQL을 사용하면 아이템 파라미터의 TableLabel 옵션을 다음과 같이 설정한다.

```
TableLabel = 애플리케이션 해시 코드,이름
```

13.6.6.보고서 템플릿 실행

보고서 템플릿을 실행하는 방법을 설명한다.

13.6.6.1. HTML 형식으로 실행

보고서 템플릿 목록 화면에서 **[보기]** 링크를 클릭하여 보고서 템플릿을 HTML 형식으로 실행할 수 있다. 처음에는 사용자 정의 파라미터에 대한 기본 값을 사용하지만 상단에 있는 사용자 정의 파라미터 입력 폼을 통해서 임의의 파라미터를 사용자로부터 입력받아서 보고서를 실행할 수 있다.

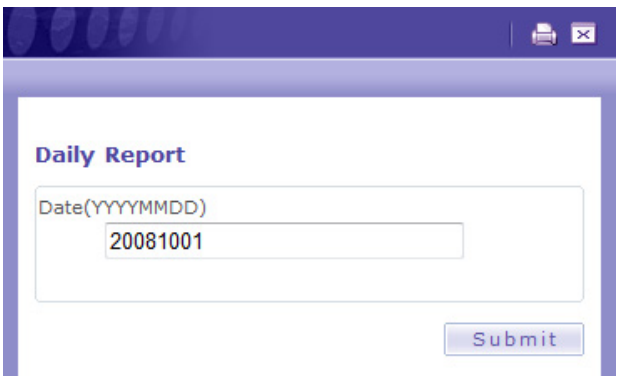
그림 13-42:HTML 형식으로 실행

1. 사용자 정의 파라미터 입력 폼 - 사용자가 입력할 수 있는 사용자 정의 파라미터 목록이 나타난다.
2. 실행 버튼 - 해당 버튼을 클릭하면 보고서를 HTML 형식으로 생성한다.

13.6.6.2. RTF 형식으로 실행

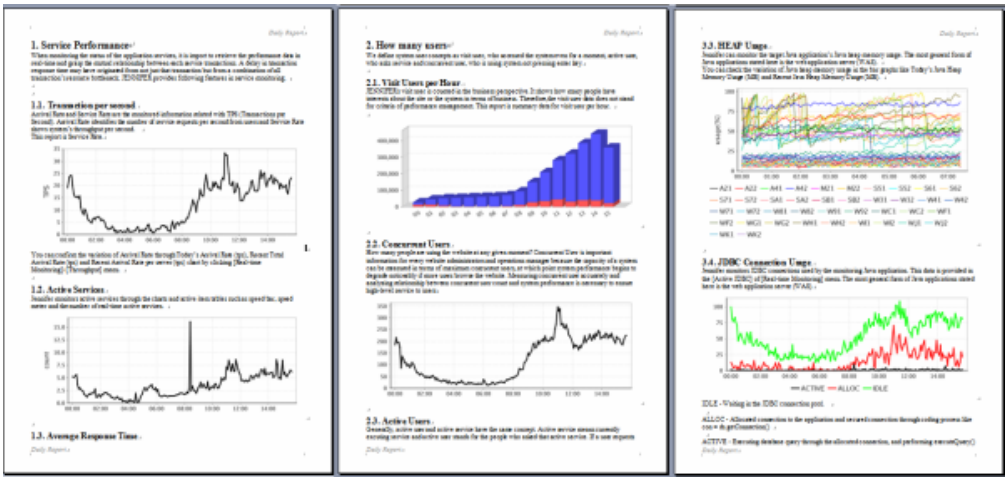
보고서 템플릿 목록 화면에서 **[리포트 다운로드]** 링크를 클릭하면 파라미터 입력 폼으로 구성된 파업 창이 나타난다. 파라미터를 입력한 후에 **[서브밋]** 버튼을 클릭하면 보고서 템플릿을 실행시켜서 그 결과를 RTF 형식으로 다운로드할 수 있다.

그림 13-43:파라미터 입력 팝업 창



다음은 RTF로 다운로드한 파일의 예제이다.

그림 13-44:RTF 파일



13.6.7.외부 데이터베이스 사용하기

보고서 템플릿에서 제니퍼 데이터베이스가 아닌 외부 데이터베이스를 사용할 수도 있다. 외부 데이터베이스를 사용하려면 보고서 템플릿 파라미터 입력 필드에 외부 데이터베이스 연결 정보를 설정한다.

```
datadb_driver = org.apache.derby.jdbc.EmbeddedDriver
datadb_url= jdbc:derby:c:/backup/db/jennifer
datadb_user = jennifer
datadb_password = jennifer
```

특정 아이템에 대해서만 외부 데이터베이스를 사용하려면 이를 아이템 파라미터 입력 필드에 지정한다.

아파치 더비가 아닌 다른 데이터베이스를 사용하는 경우에는, 해당 JDBC 드라이버 JAR 파일을 JENNIFER_HOME/server/common/lib 디렉토리에 복사한 후에 제니퍼 서버를 재시작한다.

13.6.8.반복 파라미터를 이용한 아이템 작성

반복 파라미터로 특정 데이터 그룹에 대해서 아이템을 반복적으로 실행할 수 있다.

예를 들어, 한 차트에 모든 제니퍼 에이전트에 대한 서비스 처리율을 표시하는 것이 아니라, 별도 차트로 제니퍼 에이전트 별 서비스 처리율을 표시하려면 반복 파라미터를 사용해야 한다.

이를 위해서 아이템 반복 파라미터를 다음과 같이 설정한다.

```
AGENT=SQL:SELECT DISTINCT AGENT_ID FROM PERF_X_${TODAY_DD}
```

반복 파라미터는 =을 구분자로 구분된다. 왼쪽은 반복 파라미터 이름으로 일반 사용자 정의 파라미터와 같이 이 이름을 SQL 작성에 사용할 수 있다. 그리고 오른쪽은 SQL:로 시작해야 하고, 반복 파라미터 값을 조회하기 위한 SQL을 설정한다.

다음은 AGENT로 정의한 반복 파라미터를 사용하는 SQL이다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       SERVICE_RATE
  FROM PERF_X_${TODAY_DD}
 WHERE LOG_DT = '${TODAY}'
   AND AGENT_ID = '${AGENT}'
```

이 아이템을 실행하면 제니퍼 에이전트별 서비스 처리율이 별도 차트에 반복적으로 표시된다.

13.6.9.2개의 Y축을 갖는 아이템 작성

유형이 TIME-LINE인 아이템은 왼쪽과 오른쪽 Y축을 다르게 설정해서 데이터를 표시할 수 있다. 이를 위해서는 SQL을 다음과 같이 작성한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM,
       A,
       B,
       C
  FROM SOME_TABLE
```

SQL은 선이 여러 개인 경우와 동일하며, 첫번째 칼럼은 시간이고, 나머지 A, B, C 칼럼은 데이터 칼럼으로 그 숫자에는 제한이 없다.

아이템 파라미터 GraphLabelX를 콤마[,]를 구분자로 데이터 칼럼을 설정한다.

```
GraphLabelX = A,B,C
```

아이템 파라미터 GraphLabelY를 콤마[,]를 구분자로 2개를 지정한다. 첫번째는 왼쪽 Y축을 지칭하는 이름이 되고, 두번째는 오른쪽 Y 축을 지칭하는 이름이 된다.

```
GraphLabelY = Y1,Y2
```

그리고 아이템 파라미터 GraphX2YMap를 이용해서 GraphLabelX과 GraphLabelY 파라미터로 지정한 값들을 매핑한다. GraphX2YMap에는 콤마[,]를 구분자로 2개의 값을 설정하는데 각각의 값에는 GraphLabelX로 지정한 데이터 칼럼 인덱스를 &를 구분자로 설정한다. 인덱스는 1부터 시작한다.

```
GraphX2YMap = 1&2,3
```

Notice: 다음은 A와 B 칼럼은 Y1(왼쪽 Y 축)을 기준으로 표시하고, C 칼럼은 Y2(오른쪽 Y 축)를 기준으로 표시함을 의미한다.

예를 들어, 일자별 방문자 수와 호출 건수를 Y축을 달리해서 표현하려면 SQL을 다음과 같이 작성한다.

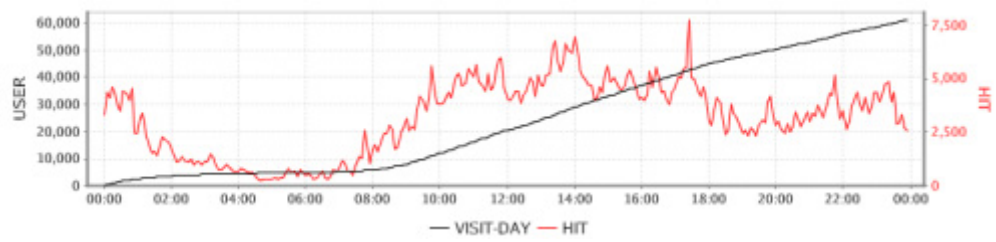
```
SELECT LOG_DT || LOG_HH || LOG_MM,
       VISIT_DAY,
       HIT
  FROM PERF_X_${TODAY_DD}
 WHERE LOG_DT = '${TODAY}'
   AND AGENT_ID = 'TOT'
   AND HIT IS NOT NULL
ORDER BY 1
```

그리고 아이템 파라미터를 다음과 같이 설정한다.

```
GraphLabelX = VISIT_DAY,HIT
GraphX2YMap = 1,2
GraphLabelY = USER,HIT
GraphShowLegend = true
GraphLineWidth = 1.0
```

해당 아이템 실행 결과는 다음과 같다.

그림 13-45:결과 그래프



13.6.10.스케줄러를 이용한 보고서 자동 생성

ReportActor 스케줄러는 보고서 템플릿을 자동으로 실행해서 RTF 형식 보고서를 게시판에 저장한다. ReportActor 스케줄러를 설정하는 방법은 다음과 같다.

```
time_actor_13 = com.javaservice.jennifer.server.timeactor.ReportActor
2350  1001
```

- 첫번째 파라미터는 보고서 템플릿을 실행하는 시간으로 HHmm 포멧으로 설정한다. 예를 들어, 2350은 23시 50분에 지정한 보고서 템플릿을 실행하라는 의미이다.
- 두번째 파라미터는 자동으로 실행할 보고서 템플릿 아이디이다.

RTF 형식 보고서 결과물은 **[구성 관리 | 게시판]** 메뉴의 [Report] 유형에서 확인할 수 있다.

13.7. 보고서 템플릿 2.0

기존 보고서 템플릿을 개선한 보고서 템플릿 2.0은 사용자 정의 파라미터와 데이터셋을 기반으로, HTML, PDF, RFT 형식의 보고서를 사용자가 직접 정의하여 생성할 수 있는 기능을 제공한다.

13.7.1.보고서 템플릿 작성

보고서 템플릿을 추가하는 방법은 다음과 같다.

- 통계 분석 | 보고서 | 보고서 템플릿 2] 메뉴로 이동한다.
- 보고서 템플릿 목록 오른쪽 하단의 [추가] 버튼을 클릭한다.
- 보고서 템플릿 입력 폼에 내용을 입력한 후에 하단의 [저장] 버튼을 클릭한다.

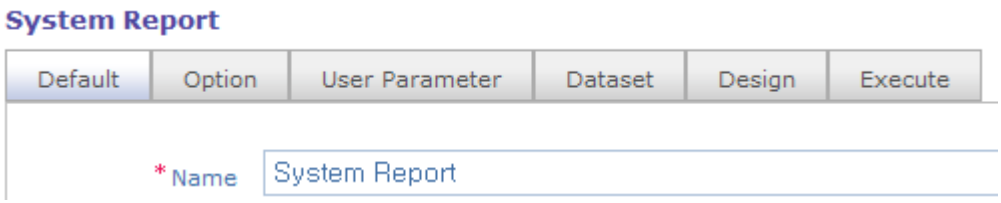
다음은 보고서 템플릿 입력 폼의 필드에 대한 설명이다.

표 13-26: 보고서 템플릿 입력 폼

필드	설명
이름	보고서 템플릿 이름
설명	보고서 템플릿에 대한 설명
머리글	보고서 템플릿 실행 유형이 PDF나 RTF인 경우에만 의미가 있다. PDF와 RTF 파일 상단에 표시되는 머리글
바닥글	보고서 템플릿 실행 유형이 PDF나 RTF인 경우에만 의미가 있다. PDF와 RTF 파일 상단에 표시되는 바닥글
서브보고서	보고서 템플릿이 다른 보고서 템플릿을 포함할 수 있다. 이 때 포함되는 보고서 템플릿을 서브보고서라고 한다. 그러나 모든 보고서 템플릿을 서브보고서로 사용할 수 있는 것은 아니고, 서브보고서 체크박스를 체크한 보고서 템플릿만을 서브보고서로 사용할 수 있다.

보고서 템플릿을 저장하면 화면 상단에 여러 개의 탭이 나타난다. 당 아이템 실행 결과는 다음과 같다.

그림 13-46:



각 탭은 다음과 같은 기능을 제공한다.

- 기본 - 이름, 머리글, 바닥글, 서브보고서 여부 등을 설정할 수 있다.
- 옵션 - 보고서 템플릿 옵션을 설정할 수 있다.
- 사용자 파라미터 - 보고서 템플릿 실행을 위해서 사용자로부터 입력 받아야 하는 데이터를 정의할 수 있다.
- 데이터셋 - 보고서 작성에 필요한 데이터 추출 방식을 설정할 수 있다.
- 디자인 - 텍스트, 테이블, 차트, 서브보고서 등의 디자인 요소를 설정할 수 있다.
- 실행 - 보고서 템플릿을 실행할 수 있다. 보고서 템플릿 목록에서 [실행] 링크를 클릭하면 다른 탭은 표시되지 않고 실행 탭만이 표시된다.

보고서 템플릿을 수정하는 방법은 다음과 같다.

- [통계 분석 | 보고서 | 보고서 템플릿 2] 메뉴로 이동한다.

- 보고서 템플릿 목록 중에서 수정할 보고서 템플릿의 이름이나 [보기] 링크를 클릭한다.
- 보고서 템플릿 입력 폼의 내용을 수정한 후에 하단의 [저장] 버튼을 클릭한다.

보고서 템플릿을 삭제하는 방법은 다음과 같다.

- [통계 분석 | 보고서 | 보고서 템플릿 2] 메뉴로 이동한다.
- 보고서 템플릿 목록 중에서 삭제할 보고서 템플릿의 체크 박스를 선택한다.
- 보고서 템플릿 목록 하단의 [삭제] 버튼을 클릭한다.

13.7.2.옵션

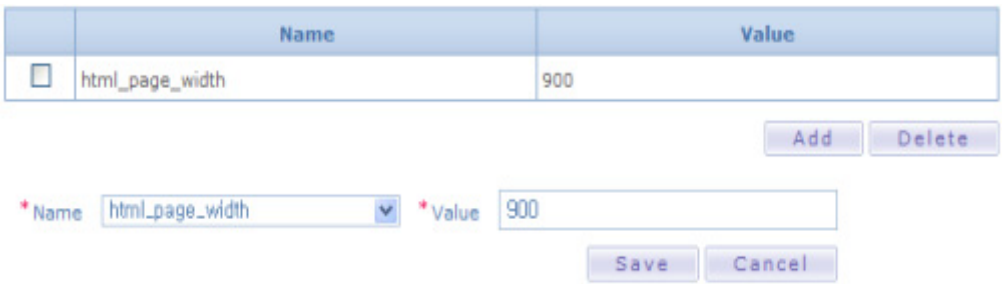
모든 차트에 대해서 넓이나 높이를 개별적으로 설정하는 것은 불편할 수 있다. 이를 해결하기 위해서 옵션 탭에서 차트 넓이나 높이에 대한 기본 값을 설정을 할 수 있다.

Notice: 옵션 탭에서 설정한 것보다 개별 디자인 요소에 대해서 설정한 것이 우선권을 갖는다.

옵션을 추가하는 방법은 다음과 같다.

- 보고서 템플릿 옵션 탭을 클릭한다.
- 옵션 목록 오른쪽 하단의 [추가] 버튼을 클릭한다.
- 옵션 입력 폼에 내용을 입력한 후에 하단의 [저장] 버튼을 클릭한다.

그림 13-47:



다음은 옵션에 대한 설명이다.

표 13-27:

이름	설명
html_page_width	보고서 템플릿 실행 유형이 HTML인 경우에만 적용되는 옵션으로, 화면 넓이를 지정하는데 사용한다.

표 13-27:

이름	설명
chart_width	차트 넓이. 차트 성격상 파이 차트는 라인 차트와 막대 차트와는 다른 비율로 넓이와 높이를 설정해야 한다. 따라서 이 옵션으로 라인 차트와 막대 차트 넓이를 지정하고, 파이 차트 넓이는 개별적으로 지정하는 것을 권장한다.
chart_height	차트 높이. 차트 성격상 파이 차트는 라인 차트와 막대 차트와는 다른 비율로 넓이와 높이를 설정해야 한다. 따라서 이 옵션으로 라인 차트와 막대 차트 높이를 지정하고, 파이 차트 높이는 개별적으로 지정하는 것을 권장한다.
chart_legend	차트 범례 표시 여부로 true를 입력하면 범례를 표시한다.
line_thickness	라인 차트 라인 두께로 숫자만을 입력해야 한다.
line_color_y	라인 차트에서 왼쪽 축에 기반한 라인 색상으로 RGB 형식으로 입력한다.
line_color_z	라인 차트에서 오른쪽 축에 기반한 라인 색상으로 RGB 형식으로 입력한다.
bar_color	막대 차트에서 막대 색상으로 RGB 형식으로 입력한다.
pie_color	파이 차트에서 파이 색상으로 RGB 형식으로 입력한다.
start_angle	파이 차트에서 시작 각도로 0에서 360 사이에서 입력한다.

색상과 관련한 옵션은 RGB 형식으로 입력한다. 예를 들어, 라인 색상을 빨간 색으로 하려면 다음과 같이 입력한다.

```
255,0,0
```

여러 색을 입력해야 하는 경우에는 슬러시[/]를 구분자로 구분하여 입력한다.

```
255,0,0/0,255,0/0,0,255
```

옵션을 수정하는 방법은 다음과 같다.

- 보고서 템플릿 옵션 탭을 클릭한다.
- 옵션 목록 중에서 수정할 옵션의 이름을 클릭한다.
- 옵션 입력 폼의 내용을 수정한 후에 하단의 [저장] 버튼을 클릭한다.

옵션을 삭제하는 방법은 다음과 같다.

- 보고서 템플릿 옵션 탭을 클릭한다.
- 옵션 목록 중에서 삭제할 옵션의 체크 박스를 선택한다.
- 옵션 목록 하단의 [삭제] 버튼을 클릭한다.

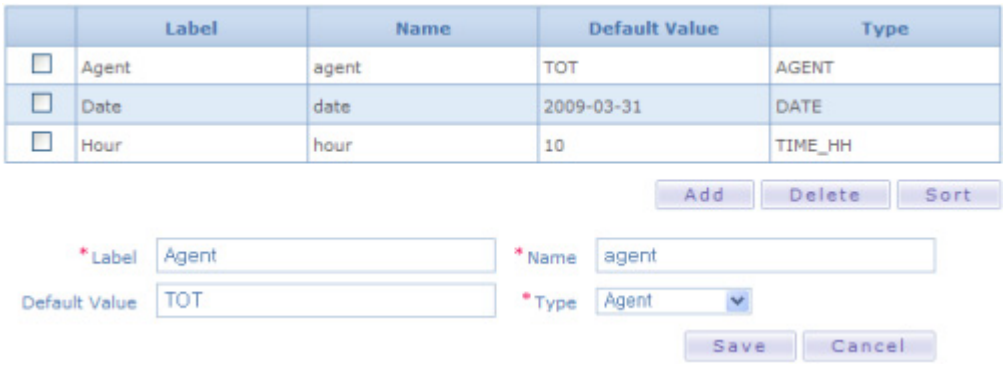
13.7.3.사용자 파라미터

사용자 파라미터 탭에서 보고서 템플릿 실행을 위해서 사용자로부터 입력받아야 하는 데이터를 정의할 수 있다.

사용자 파라미터를 추가하는 방법은 다음과 같다.

- 보고서 템플릿 사용자 파라미터 탭을 클릭한다.
- 사용자 파라미터 목록 오른쪽 하단의 [추가] 버튼을 클릭한다.
- 사용자 파라미터 입력 폼에 내용을 입력한 후에 하단의 [저장] 버튼을 클릭한다.
- 옵션 입력 폼에 내용을 입력한 후에 하단의 [저장] 버튼을 클릭한다.

그림 13-48:



다음은 사용자 파라미터 입력 폼의 필드에 대한 설명이다.

표 13-28:

필드	설명
표시	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 나타나는 라벨이다. 보고서 템플릿 사용자가 이해할 수 있는 이름을 사용하도록 한다.
이름	데이터셋 SQL에서 사용하는 키워드로 영어 소문자와 숫자만을 사용하는 것을 권장한다.
기본 값	기본 값을 설정한다.
유형	유형에 따라서 보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 다르게 표시된다.

다음은 사용자 파라미터 유형에 대한 설명이다.

표 13-29:

유형	설명
도메인	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 도메인이 드랍다운 박스로 나타난다. 도메인을 선택한 경우에는 도메인 목록 유형 사용자 파라미터는 추가할 수 없다.
도메인 목록	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 여러 개의 도메인을 선택할 수 있는 다중선택 박스로 나타난다. 도메인 목록을 선택한 경우에는 도메인 유형 사용자 파라미터는 추가할 수 없다.
에이전트	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 제니퍼 에이전트가 드랍다운 박스로 나타난다. 에이전트를 선택한 경우에는 에이전트 목록 유형 사용자 파라미터는 추가할 수 없다.
에이전트 목록	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 여러 개의 제니퍼 에이전트를 선택할 수 있는 다중선택 박스로 나타난다. 에이전트 목록을 선택한 경우에는 에이전트 유형 사용자 파라미터는 추가할 수 없다.
년	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 년을 선택할 수 있는 필드가 나타난다.
월	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 월을 선택할 수 있는 드랍다운 박스가 나타난다.
날짜	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 팝업 화면으로 날짜를 선택할 수 있는 필드가 나타난다.
시간(HH)	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 시간을 선택할 수 있는 드랍다운 박스가 나타난다.
시간(HHMM)	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 시간과 5분 단위 분을 선택할 수 있는 드랍다운 박스가 나타난다.
경보 유형	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 경보 유형을 선택할 수 있는 드랍다운 박스가 나타난다.
경보 이름	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 경보 이름을 선택할 수 있는 드랍다운 박스가 나타난다.
애플리케이션	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 팝업 화면으로 애플리케이션을 선택할 수 있는 필드가 나타난다.
텍스트	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 임의의 문자를 입력할 수 있는 필드가 나타난다.
숫자	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에 임의의 숫자를 입력할 수 있는 필드가 나타난다.
줄 바꿈	보고서 템플릿 실행 화면 사용자 파라미터 입력 폼에서 줄 바꿈을 하고 싶을 때 사용한다. 이 유형은 사용자로부터 파라미터를 입력 받는 것과는 관련성이 없다 .

사용자 파라미터를 수정하는 방법은 다음과 같다.

- 보고서 템플릿 사용자 파라미터 탭을 클릭한다.
- 사용자 파라미터 목록 중에서 수정할 사용자 파라미터의 표시를 클릭한다.
- 사용자 파라미터 입력 폼의 내용을 수정한 후에 하단의 [저장] 버튼을 클릭한다.

사용자 파라미터를 삭제하는 방법은 다음과 같다.

- 보고서 템플릿 사용자 파라미터 탭을 클릭한다.
- 사용자 파라미터 목록 중에서 삭제할 사용자 파라미터의 체크 박스를 선택한다.
- 사용자 파라미터 목록 하단의 [삭제] 버튼을 클릭한다.

사용자 파라미터의 순서를 변경하는 방법은 다음과 같다.

- 보고서 템플릿 사용자 파라미터 탭을 클릭한다.
- 사용자 파라미터 목록 오른쪽 하단의 [정렬] 버튼을 클릭한다.
- 정렬 팝업 창에서 사용자 파라미터의 순서를 변경한다.

13.7.4. 데이터셋

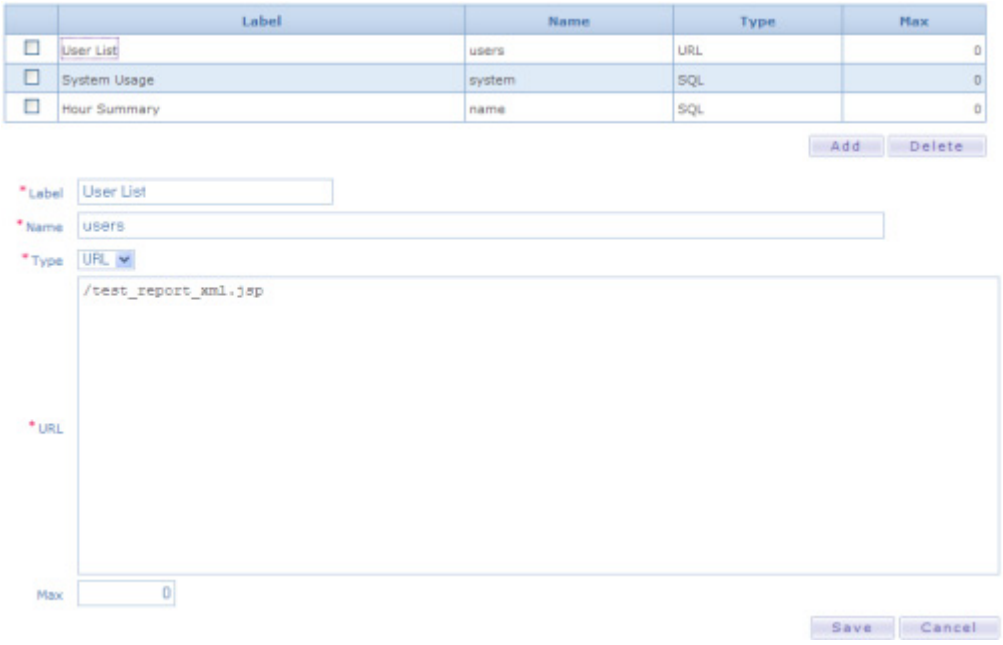
데이터셋 탭에서 보고서 작성에 필요한 데이터 추출 방식을 설정할 수 있다.

13.7.4.1. 기본 관리

데이터셋을 추가하는 방법은 다음과 같다.

- 보고서 템플릿 데이터셋 탭을 클릭한다.
- 데이터셋 목록 오른쪽 하단의 [추가] 버튼을 클릭한다.
- 데이터셋 입력 폼에 내용을 입력한 후에 하단의 [저장] 버튼을 클릭한다.

그림 13-49:



다음은 데이터셋 입력 폼의 필드에 대한 설명이다.

표 13-30:

필드	설명
표시	데이터셋에 대한 표시
이름	데이터셋 이름으로 영어 소문자와 숫자만을 사용하는 것을 권장한다.
유형	데이터를 추출하는 방식으로 SQL 혹은 URL을 선택한다.
SQL/URL	유형으로 SQL을 선택한 경우에는 데이터 추출에 필요한 SQL을 입력하고, 유형으로 URL을 선택한 경우에는 URL을 입력한다.
최대	성능 저하를 방지하기 위해서 데이터셋 최대 크기를 설정한다.

데이터셋을 수정하는 방법은 다음과 같다.

- 보고서 템플릿 데이터셋 탭을 클릭한다.
- 데이터셋 목록 중에서 수정할 데이터셋의 표시를 클릭한다.
- 데이터셋 입력 폼의 내용을 수정한 후에 하단의 [저장] 버튼을 클릭한다.

데이터셋을 삭제하는 방법은 다음과 같다.

- 보고서 템플릿 데이터셋 탭을 클릭한다.
- 데이터셋 목록 중에서 삭제할 데이터셋의 체크 박스를 선택한다.

- 데이터셋 목록 하단의 [삭제] 버튼을 클릭한다.

SQL 작성 사용자 파라미터를 사용하여 SQL을 작성하는 방법은 다음과 같다.

우선 사용자 파라미터에 date를 이름으로 하는 DATE 유형 사용자 파라미터와 agent를 이름으로 하는 AGENT 유형 사용자 파라미터가 있다가 가정한다.

다음은 사용자 관련 데이터를 가져오는 쿼리이다.

```
SELECT LOG_DT || LOG_HH || LOG_MM AS TIME,
        AGENT_ID,
        CONCURRENT_USER,
        ACTIVE_USER,
        THINKTIME,
        HIT,
        VISIT_DAY,
        VISIT_HOUR
   FROM PERF_X_${date.day}
  WHERE LOG_DT = '${date}'
    AND AGENT_ID = '${agent}'
```

SQL을 작성할 때 다음 사항을 참고하도록 한다.

- 사용자 파라미터를 ${}를 이용해서 SQL안에서 사용할 수 있다.
- 사용자 파라미터 유형이 DATE인 경우에는 ${date.day}는 일자만을 반환하고 ${date.format}은 yyyy-MM-dd 포멧으로 변환된 날짜를 반환한다.
- 사용자 파라미터를 VARCHAR 유형 칼럼과 비교하는 경우에는 '${agent}'와 같이 작은 따옴표를 사용해야 한다.
- TIME LINE 유형 라인 차트 처리가 필요한 경우에는 SELECT 절에서 LOG_DT, LOG_HH, LOG_MM 칼럼을 붙여서 사용한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM AS TIME,
       ...
```

PERF_X_01~31 테이블에 있는 주요 성능 정보를 TIME LINE 유형 라인 차트로 보여줄 때는 다음 SQL로 데이터셋을 구성한다.

```
SELECT LOG_DT || LOG_HH || LOG_MM AS TIME,
       CONCURRENT_USER,
       ACTIVE_USER,
       ACTIVE_SERVICE,
       ARRIVAL_RATE,
       SERVICE_RATE,
       RESPONSE_TIME,
       THINKTIME,
       HEAP_TOTAL,
       HEAP_USED,
       SYS_CPU,
       JVM_CPU,
       SYS_MEM_USED,
       JVM_NAT_MEM,
       JDBC_ACTIVE,
       JDBC_IDLE
  FROM PERF_X_${date.day}
 WHERE LOG_DT = '${date}'
   AND AGENT_ID = '${agent}'
```

PERF_X_01~31 테이블에 있는 주요 성능 정보를 막대 차트나 SIMPLE LINE 유형 라인 차트로 보여줄 때는 다음 SQL로 데이터셋을 구성한다.

```
SELECT LOG_HH,
       SUM(HIT) AS HIT,
       SUM(VISIT_HOUR) AS VISIT
  FROM PERF_X_${date.day}
 WHERE LOG_DT = '${date}'
   AND AGENT_ID = '${agent}'
 GROUP BY LOG_HH
```

13.7.4.2. URL 작성

약속된 XML 형식을 반환하는 URL을 호출하여 이를 데이터셋으로 사용할 수도 있다. 데이터베이스가 아닌 파일에 저장되는 X-View 트랜잭션 데이터와 REMON 데이터 등을 이용해서 보고서를 작성할 때 유용하게 사용할 수 있다.

해당 URL이 제니퍼 서버에 있는 경우에는 /로 시작하고 외부 URL을 호출하는 경우에는 http://로 시작해야 한다.

해당 URL은 약속된 XML을 반환해야 한다. 아래는 JSP를 이용한 예제이다.

```
<%@ page contentType="text/xml; charset=UTF-8" %>
<%@ page import="java.sql.Types" %>
<%
out.clear();
%><?xml version="1.0" encoding="UTF-8" ?>

<dataset>
    <fields>
        <field name="name" type="<%= Types.VARCHAR %>" />
        <field name="age" type="<%= Types.INTEGER %>" />
        <field name="level" type="<%= Types.INTEGER %>" />
    </fields>
    <% if (!"true".equals(request.getParameter("get_fields"))) { %>
    <rows>
        <row>
            <cell>Kim</cell>
            <cell>33</cell>
            <cell>5</cell>
        </row>
        <row>
            <cell>Lee</cell>
            <cell>31</cell>
            <cell>2</cell>
        </row>
    </rows>
    <% } %>
```

XML을 작성할 때 다음 사항을 참고하도록 한다.

- XML 인코딩으로는 UTF-8을 사용해야 한다.
- 최상위 태그는 dataset이고 이 태그는 fields와 rows 태그로 구성된다.
- fields 태그는 필드 메타 정보를 기술하는 여러 개의 field 태그로 구성된다. field 태그 name 속성으로 필드 이름을 지정하고, type 속성으로 필드 속성을 지정한다. type 속성을 지정할 때는 java.sql.Types 클래스의 상수를 이용해서 설정한다.

- rows 태그는 row와 cell 태그를 통해서 데이터를 정의하는데 HTTP 요청 파라미터 get_fields가 true가 아닌 경우에만 처리하도록 한다.
- 사용자 파라미터 이름을 이름으로 요청 파라미터가 전달된다. 예를 들어, age라는 사용자 파라미터가 있다면 다음과 같이 사용자가 입력한 값을 추출할 수 있다.

```
String age = request.getParameter("age");
```

13.7.4.3. 데이터셋 필드

저장 버튼을 클릭하면 데이터셋을 실행하여 필드 정보를 생성한다. SQL이나 URL이 올바르지 않으면 예외가 발생하고 필드 정보가 표시되지 않는다.

필드 정보는 다음과 같다.

표 13-31:

이름	설명
이름	필드 이름으로 주로 데이터베이스 칼럼 이름을 사용한다.
표시	화면에 출력될 때 사용하는 표시
숫자	데이터의 숫자 여부
유형	데이터 유형
텍스트 정렬	데이터 좌우 정렬 방법

- 표시 칼럼 왼쪽에 있는 [수정] 버튼을 통해서 필드 표시와 텍스트 정렬 방법을 수정할 수 있다.

그림 13-50:

Field

Name	Label	Number	Type	Text Alignment
name	Update name	false	VARCHAR	center
age	Update age	true	INTEGER	left
salary	Update salary	true	INTEGER	left

13.7.5.디자인

디자인 탭에서 보고서 템플릿 디자인 요소를 정의한다. 디자인 요소에는 텍스트, 테이블, 라인 차트, 막대 차트, 파이 차트, 서브보고서 등이 있다.

디자인 요소를 추가하는 방법은 다음과 같다.

- 보고서 템플릿 디자인 탭을 클릭한다.
- 디자인 요소 목록 오른쪽 하단의 여러 [추가] 버튼 중에 하나를 클릭한다.
- 디자인 요소 입력 폼에 내용을 입력한 후에 하단의 [저장] 버튼을 클릭한다.

디자인 요소를 수정하는 방법은 다음과 같다.

- 보고서 템플릿 디자인 탭을 클릭한다.
- 디자인 요소 목록 중에서 수정할 디자인 요소의 내용을 클릭한다.
- 디자인 요소 입력 폼의 내용을 수정한 후에 하단의 [저장] 버튼을 클릭한다.

디자인 요소를 삭제하는 방법은 다음과 같다.

- 보고서 템플릿 디자인 탭을 클릭한다.
- 디자인 요소 목록 중에서 삭제할 디자인 요소의 체크 박스를 선택한다.
- 디자인 요소 목록 하단의 [삭제] 버튼을 클릭한다.

디자인 요소의 순서를 변경하는 방법은 다음과 같다.

- 보고서 템플릿 디자인 탭을 클릭한다.
- 디자인 요소 목록 오른쪽 하단의 [정렬] 버튼을 클릭한다.
- 정렬 팝업 창에서 디자인 요소의 순서를 변경한다.

Notice: 디자인 요소 목록에 있는 [보기] 링크를 통해서 해당 디자인 요소만을 실행할 수도 있다.

13.7.5.1. 텍스트

디자인 요소의 하나로 텍스트를 추가할 때 사용한다. 텍스트 디자인 요소 필드 정보는 다음과 같다.

표 13-32:

필드	설명
내용	화면에 표시할 텍스트 내용을 입력한다. ${}를 이용해서 사용자 파라미터를 함께 표시할 수 있다.
유형	유형으로 제목 1, 제목 2, 제목 3, 본문 등을 선택할 수 있으며, 유형에 따라서 화면에 표시되는 방식이 달라진다.

13.7.5.2. 테이블

디자인 요소의 하나로 테이블을 추가할 때 사용한다. 테이블 디자인 요소 필드 정보는 다음과 같다.

표 13-33:

필드	설명
제목	테이블 제목
데이터셋	테이블을 표시하는데 사용할 데이터셋을 선택한다.
필드	선택한 데이터셋 중에서 화면에 표시할 필드를 선택한다. 하단에 있는 위 아래 버튼을 통해서 테이블 칼럼 순서를 변경할 수 있다.
그룹	특정 필드를 기준으로 그룹으로 묶어서 반복적으로 표시할 수도 있다.

13.7.5.3. 라인 차트

디자인 요소의 하나로 라인 차트를 추가할 때 사용한다. 라인 차트 디자인 요소 필드 정보는 다음과 같다.

표 13-34:

필드	설명
제목	라인 차트 제목
데이터셋	라인 차트를 표시하는데 사용할 데이터셋을 선택한다.
X	라인 차트 X 축으로 사용할 필드를 선택한다. 문자 유형 필드만을 선택할 수 있다.
Y	왼쪽 축을 기준으로 라인을 표시할 데이터 필드들을 선택한다. 숫자 유형 필드만을 선택할 수 있다.
Z	오른쪽 축을 기준으로 라인을 표시할 데이터 필드들을 선택한다. 숫자 유형 필드만을 선택할 수 있다.
넓이	차트 넓이를 설정한다.
높이	차트 높이를 설정한다.
유형	라인 차트 유형을 선택한다. X 축으로 선택한 필드에 따라서 TIME LINE 혹은 SIMPLE LINE을 선택하면 된다. 일반적으로 막대 차트로 표시한 데이터를 라인 차트로 보여주고 싶을 때는 SIMPLE LINE을 선택한다.
범례 표시	범례 표시 여부를 설정한다.
선 두께 선	두께를 설정한다.
선 색(Y)	왼쪽 축을 기준으로하는 선 색을 RGB 형식으로 설정한다.

표 13-34:

필드	설명
최대(Y)	왼쪽 축 최대 값을 설정한다. 실제 값이 최대 값보다 큰 경우에는 실제 값을 기준으로 왼쪽 축이 표시된다.
선 색(Z)	오른쪽 축을 기준으로하는 선 색을 RGB 형식으로 설정한다.
최대(Z)	오른쪽 축 최대 값을 설정한다. 실제 값이 최대 값보다 큰 경우에는 실제 값을 기준으로 오른쪽 축이 표시된다.

13.7.5.4. 막대 차트

디자인 요소의 하나로 막대 차트를 추가할 때 사용한다. 막대 차트 디자인 요소 필드 정보는 다음과 같다.

표 13-35:

필드	설명
제목	막대 차트 제목
데이터셋	막대 차트를 표시하는데 사용할 데이터셋을 선택한다.
X	막대 차트 X 축으로 사용할 필드를 선택한다. 문자 유형 필드만을 선택할 수 있다.
Y	막대 차트로 표시할 데이터 필드를 선택한다. 숫자 유형 필드만을 선택할 수 있다.
넓이	차트 넓이를 설정한다.
높이	차트 높이를 설정한다.
막대 색상	막대 색상을 RGB 형식으로 설정한다.
3D 표시	막대를 3D로 표시할지 여부를 선택한다.
값 표시	값 표시 여부를 설정한다. None을 선택하면 값을 표시하지 않고, All을 선택하면 모든 값을 표시하며, Only Max를 선택하면 최대 값만을 표시한다.
최대	최대 값을 설정한다. 실제 값이 최대 값보다 큰 경우에는 실제 값을 기준으로 표시된다.

13.7.5.5. 파이 차트

디자인 요소의 하나로 파이 차트를 추가할 때 사용한다. 파이 차트 디자인 요소 필드 정보는 다음과 같다.

표 13-36:

필드	설명
제목	파이 차트 제목
데이터셋	파이 차트를 표시하는데 사용할 데이터셋을 선택한다.
X	파이 차트 X 축으로 사용할 필드를 선택한다. 문자 유형 필드만을 선택할 수 있다.
Y	파이 차트로 표시할 데이터 필드를 선택한다. 숫자 유형 필드만을 선택할 수 있다.
넓이	차트 넓이를 설정한다.
높이	차트 높이를 설정한다.
범례 표시	범례 표시 여부를 설정한다.
3D 표시	파이를 3D로 표시할지 여부를 선택한다.
값 표시	값 표시 여부를 선택한다.
시작 각도	파이 시작 각도를 설정한다. 0, 90, 180, 270 중에서 선택할 수 있다.
표시	각 파이 영역별로 의미 있는 값을 보여줄 때 사용한다. 예를 들어, X로 선택한 필드의 값이 C, E, W라면 이를 다음과 같이 슬러시[/]를 구분자로 설정한다. C=심각/E=예외/W=경고
파이 색상	파이 색상을 RGB 형식으로 설정한다.

13.7.5.6. 서브보고서

디자인 요소의 하나로 서브보고서를 추가할 때 사용한다. 서브보고서 디자인 요소 필드 정보는 다음과 같다.

표 13-37:

필드	설명
제목	서브보고서 제목
서브보고서	서브보고서로 사용할 보고서 템플릿을 선택한다.
파라미터	서브보고서로 전달할 사용자 파라미터를 선택한다.
바인딩 파라미터	위에서 선택한 파라미터를 바인딩할 서브보고서 사용자 파라미터를 선택한다.

서브보고서를 사용하려면 다음 조건을 충족해야 한다.

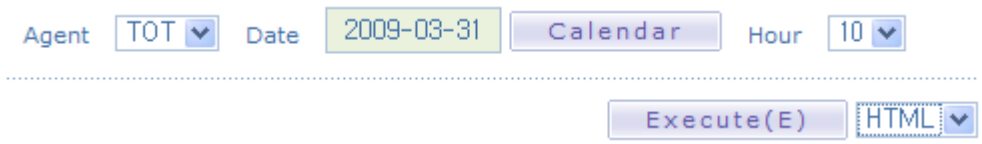
- 사용자 파라미터에 도메인 목록 혹은 에이전트 목록 유형이 있어야 한다. 따라서 서브보고서 디자인 요소 입력 폼 파라미터 필드에서는 도메인 목록 혹은 에이전트 목록 유형만을 선택할 수 있다.

- 서브보고서로 사용되는 보고서 템플릿의 사용자 파라미터에 도메인 혹은 에이전트 유형이 있어야 한다. 따라서 서브보고서 디자인 요소 입력 폼 바인딩 파라미터 필드에서는 도메인 혹은 에이전트 유형만을 선택할 수 있다.
- 바인딩 파라미터를 제외한 서브보고서로 사용되는 보고서 템플릿의 사용자 파라미터와 동일한 사용자 파라미터가 서브보고서를 포함하는 보고서 템플릿에 존재해야 한다.

13.7.6. 실행

- 실행 탭을 클릭하면 사용자 파라미터 탭에서 정의한 사용자 파라미터 입력 폼이 상단에 표시된다.

그림 13-51:



보고서 템플릿을 실행하려면 사용자 파라미터 입력 폼 하단 오른쪽에 있는 [실행] 버튼을 클릭한다. 이 때 [실행] 버튼 오른쪽에 있는 드랍다운 박스를 통해서 보고서 결과 유형을 선택할 수 있다.

보고서 결과 유형은 다음과 같다.

- HTML - HTML 형식으로 보고서가 화면에 표시된다.
- PDF - PDF 파일 보고서를 다운로드 할 수 있다.
- RTF - RTF 파일 보고서를 다운로드 할 수 있다.
- CSV - CSV 파일 보고서를 다운로드 할 수 있다. 단, 이 경우에는 테이블 디자인 요소만이 표시된다.

보고서 결과 유형으로 PDF를 선택한 경우에 영어가 아닌 다른 언어는 PDF 파일에 올바르게 표시되지 않는다. 이 문제를 해결하려면 제니퍼 서버의 ui_report_pdf_font 옵션으로 해당 언어를 처리할 수 있는 폰트 파일 위치를 명시적으로 설정해주어야 한다. 예를 들어, 윈도우에서 한글을 지원하는 굴림 폰트를 사용하려면 다음과 같이 설정한다.

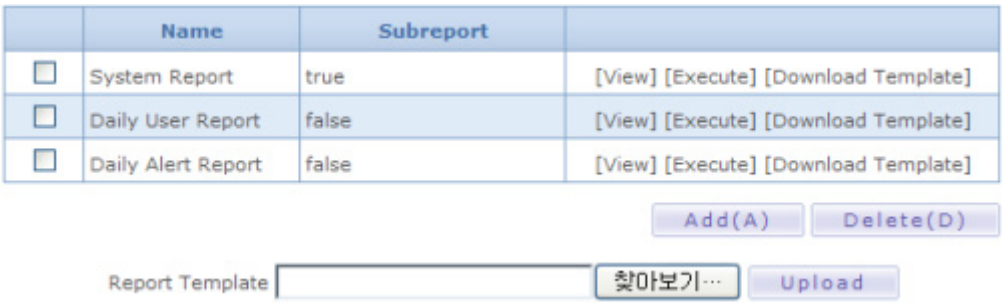
```
ui_report_pdf_font = c:/windows/fonts/gulim.ttc,0
```

리눅스나 유닉스에서는 사용할 폰트를 제니퍼 서버를 설치한 서버에 복사한 후에 해당 위치를 설정해주면 된다.

13.7.7.보고서 템플릿 Export와 Import

- 보고서 템플릿을 XML 파일로 Export하고 Import할 수 있다.

그림 13-52:



보고서 템플릿을 Export하는 방법은 다음과 같다.

- [통계 분석 | 보고서 | 보고서 템플릿 2] 메뉴로 이동한다.
- 보고서 템플릿 목록 중에서 Export할 보고서 템플릿의 [템플릿 다운로드] 버튼을 클릭한다.

보고서 템플릿을 Import하는 방법은 다음과 같다.

- [통계 분석 | 보고서 | 보고서 템플릿 2] 메뉴로 이동한다.
- Import할 보고서 템플릿을 지정한 후 [업로드] 버튼을 클릭한다.

Notice: 서브보고서가 포함된 보고서 템플릿은 서브보고서로 사용하는 보고서 템플릿도 개별적으로 Export하고, Import 해야 한다. 그리고 Import를 한 후에 서브보고서 디자인 요소에서 이를 명시적으로 다시 설정해주어야 한다.

13.8. 성능데이터 추이 분석(PTA)

성능데이터 추이 분석이란 제니퍼4.1에서 새롭게 추가된 기능으로 여러가지 서로 다른 성능 데이터를 비교 분석하는 기능을 말한다.

임의의 날짜에 대한 다양한 성능 통계 데이터를 상호 비교함으로써 현 시스템의 성능 데이터들의 상호 연관성이나 변화들을 쉽게 파악하는 수단이다. 이것을 통해서 최선의 성능 관리 방안을 도출하기 위한 도움을 얻을 수 있다.

애플리케이션 성능 모니터링을 통해 수집되는 데이터는 크게 자원, 사용자, 그리고 서비스 성능데이터로 구분될 수 있다. 많은 사용자가 사용할 수록 서비스의 요청건수는 늘어나고 그에 따른 시스템의 리소스를 필요로 한다.

하지만 이들간의 상관관계는 단순하거나 수학적으로 명확하게 정의되기 어렵다.혹자들은 계산식에 의해 요청에 따른 자원 사용량을 계산하려고 하지만 거의 불가능에 가깝다. 따라서 산업계에서는 기본 부하량 계산식에 과거의 경험을 접목 시켜 시스템의 필요용량을 산정하고 관리하고 있다.

여기서 경험이라는 것이 성능 관리에 있어 부인할 수 없는 한축이라면 이 성능 관리 경험을 강화하고 보다 명확히 이해할 수 있는 수단이 필요하다.

제니퍼의 PTA 기능은 수집되는 성능데이터의 변화 추이를 일자단위로 상호 비교분석할 수 있는 기능이다. 어제 오늘의 처리량 변화 혹은 처리량과 CPU사용량의 상관관계들을 비교함으로써 각 성능데이터의 연관도를 인지하고 개선 포인트를 잡아가는데 도움을 된다.

그림 13-53:

PTA 데이터는 다른 성능데이타와 별도의 관리와 저장주기를 갖는다.

13.8.1.통계적 PTA 활용

성능데이터 추이분석 기능은 제니퍼 4.1.0 정식버전을 설치했다면 자동으로 enable된다.

- 메뉴 : [통계분석|PTA]

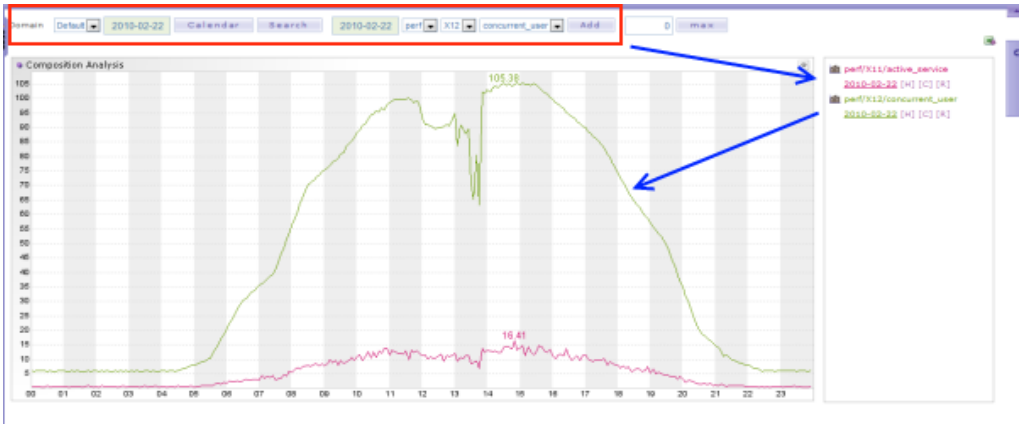
그러나 만약 이전 버전을 업그레이드 했다면 명시적으로 메뉴를 등록해 주어야 한다.

```
stat_pta.jsp
```

사용법은 직관적이며 아래와 같다. 먼저 [복합분석] 메뉴 화면에서 상단의 입력내용을 선택한다. 추가 버튼을 클릭하면 해당 그래프가 화면에 표시된다.

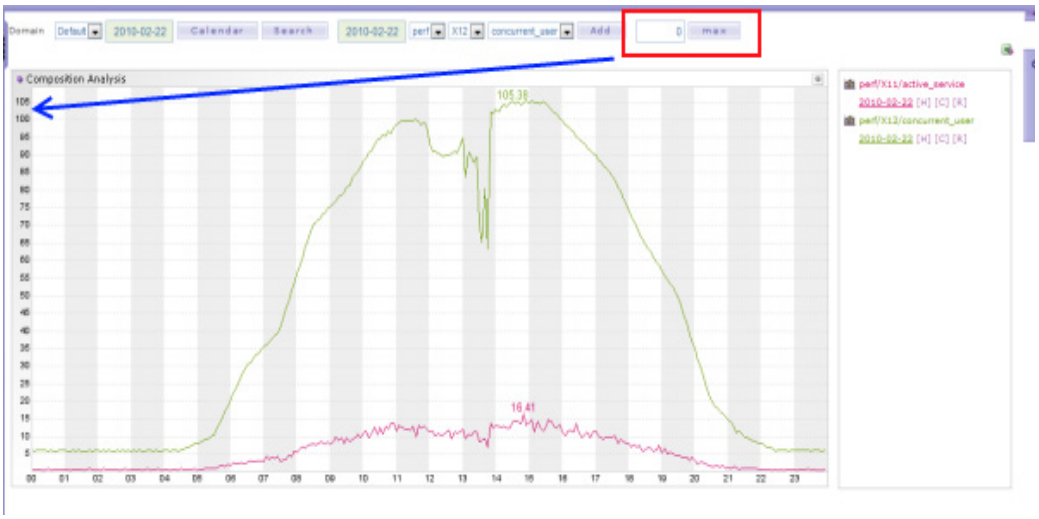
날짜선택 --> 검색 --> 선택박스 --> 추가

그림 13-54:



그래프의 전체 스케일은 데이터에 따라 자동으로 변경된다. 하지만 고정 시키고자하면 “MAX” 값을 설정할 수 있다.

그림 13-55:

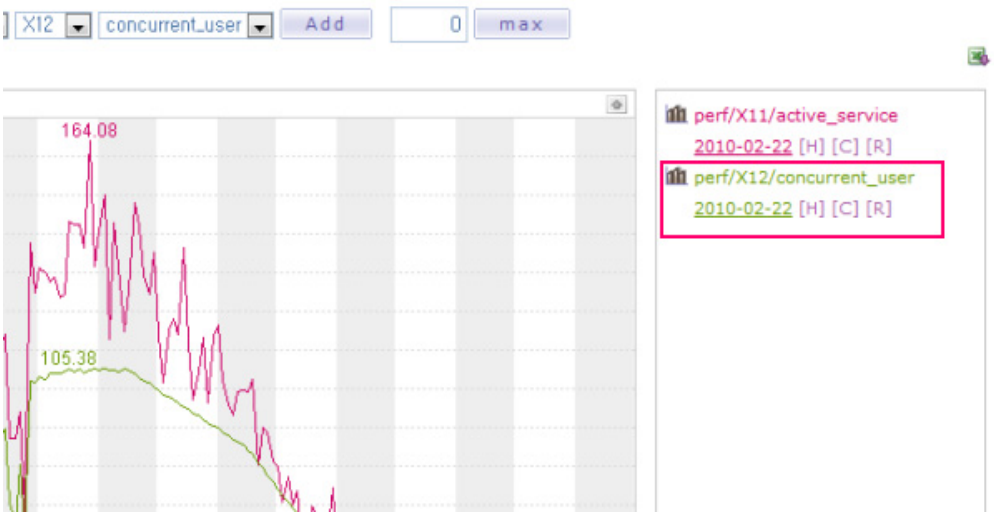


추이분석 기능의 기본 목적은 서로 다른 성능데이터에 대한 자유로운 비교분석이다. 따라서 데이터의 성격상 전혀 다른 값의 범위를 가질 수 있다. 이런 경우 개별 그래프의 스케일을 변경할 수 있다.

그림 13-56:

개별 그래프를 임시로 감추거나 불필요한 그래프를 화면에서 제거할 수 있다.

그림 13-57:

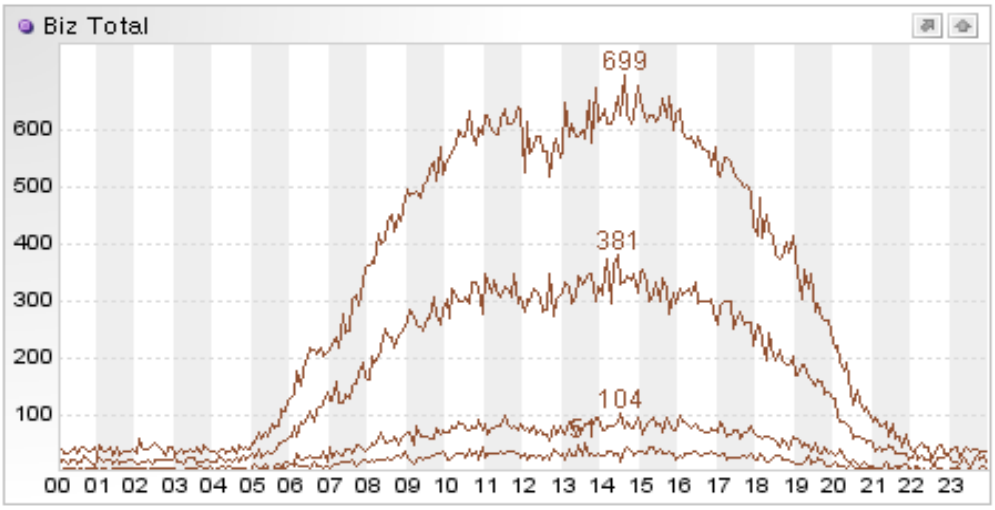


마지막으로 우측상단의 버튼을 클릭하면 그래프 좌표상의 데이터들을 엑셀로 받을 수 있다.

13.8.2. 실시간 PTA 활용

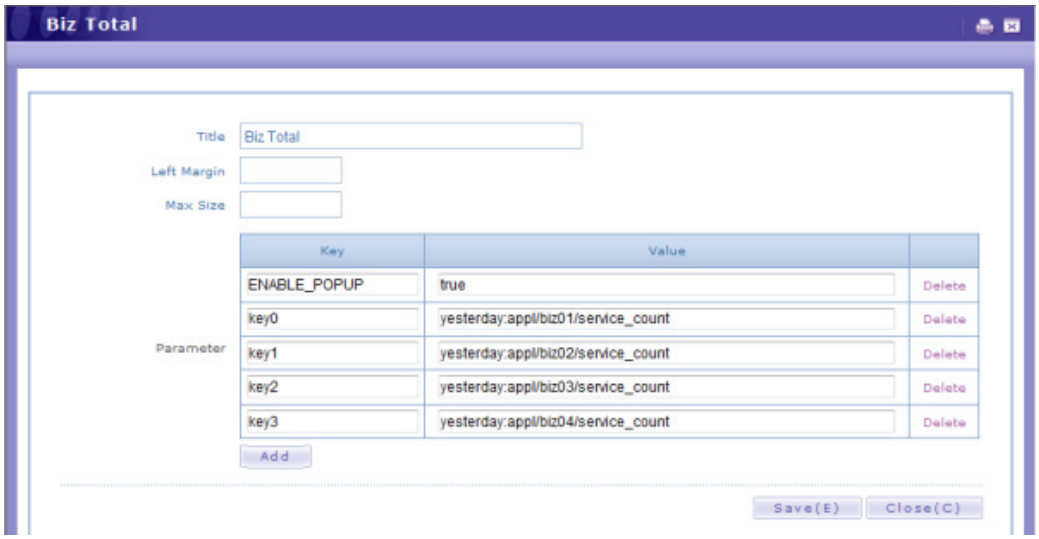
어떤 성능데이터의 일자단위 성능 추이를 실시간으로 모니터링 할 때 사용한다.

그림 13-58:



여러가지 서로 다른 성능데이터를 한 화면에서 비교할 수 있다.

그림 13-59:



그래프에 표현하고자 하는 데이터는 key0~key99까지 연속된 이름으로 설정한다. 즉 key0, key1,,, 이런식으로 설정하며 만약 중간에 결번에 발생하면 거기까지만 화면에 표현된다.

```
key0 = yesterday:appl/biz01/service_count
key1 = today:appl/biz01/service_count
```

그림 13-60:

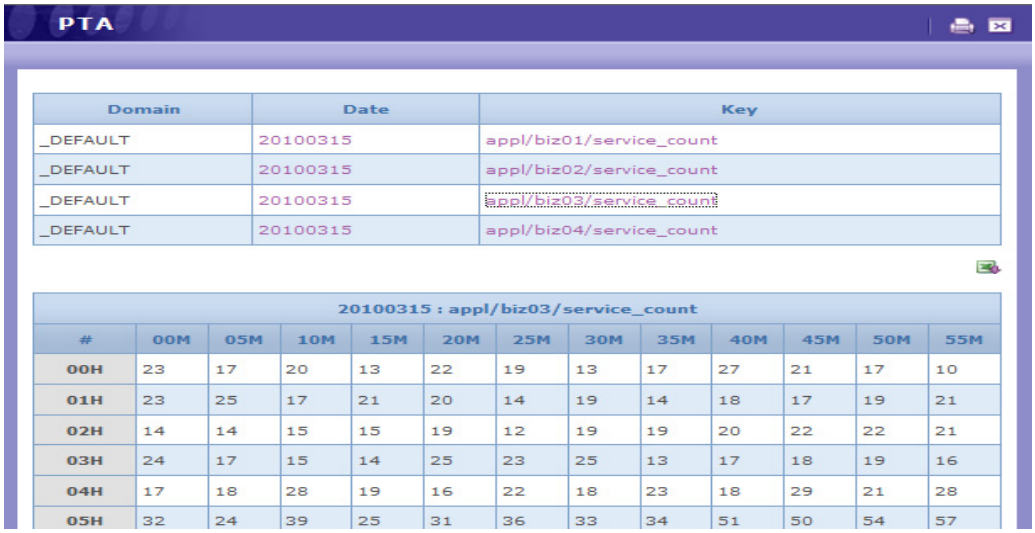
Key	Value	
ENABLE_POPUP	true	Delete
key0	today:ext/comp/biz_h	Delete
key1	today:ext/comp/biz_l	Delete

아래 옵션을 설정하고 그래프의 타이틀을 입력하면,

```
ENABLE_POPUP = true
```

실시간 PTA 그래프의 오른쪽 상단에는 상세 보기 버튼이 활성화된다. 이 버튼을 클릭하면 팝업창을 통해 실데이터를 확인하고 엑셀로 다운로드 할 수 있다.

그림 13-61:



13.8.2.1. 복합데이터 생성하기

실시간 모니터링 시 다른 여러개를 연산하여 하나의 성능 지표를 만들어 낼 수 있다. 서버 옵션에 다음과 같이 설정한다.

```
pta_items.a1=today:appl/biz01/service_count
pta_items.a2=today:appl/biz02/service_count
pta_items.a3=today:appl/biz03/service_count

pta_items.a4=today:appl/biz04/service_count
pta_items.a5=today:appl/biz05/service_count
pta_items.a6=today:appl/biz06/service_count

pta_items.a7=today:appl/biz07/service_count
pta_items.a8=today:appl/biz08/service_count
pta_items.a9=today:appl/biz09/service_count

pta_compose.biz_h= (a7+a8+a9)/(a1+a2+a3)
pta_compose.biz_l = (a4+a5+a6)/(a1+a2+a3)
```

Composed PTA 데이터를 생성하려면 pta_items.xxx 옵션으로 연산에 참여할 아이템을 설정한다.

그리고 새로 생성할 key이름을 pta_compose.xxx를 이용하여 연산식을 등록한다.

이값을 모니터링 할 때는 사용자 정의 화면의 차트 옵션에 다음 값을 입력한다.

```
key0 = today:ext/comp/biz_h
key1 = today:ext/comp/biz_l
```

13.8.3.PTA 데이터 관리

다양한 성능 추이를 신속하게 비교 분석하기 위해서 제니퍼는 별도의 저장 방식을 구현하고 있다.

13.8.3.1. 저장 경로

데이터 저장 위치는 제니퍼 서버 옵션으로 설정할 수 있다.

```
pta_dir=../../data/pta/
```

각 데이터는 “pta_dir”에 설정된 경로에 아래에 파일 단위로 관리된다.

13.8.3.2. 백업과 복구

성능 추이 데이터에 대한 백업및 복구는 제니퍼의 기본 백업기능을 이용한다. 하지만 디렉토리 복사를 통해서도 백업되거나 복구될 수 있다.

13.8.3.3. 저장 주기

배포되는 제니퍼 서버 설정 파일에는 아래와 같이 저장 주기가 1년을 기본값으로 셋팅되어 있다. 하지만 원칙적으로 디스크량이나 백업 정책에 따라 재설정 해야 한다.

```
time_actor_16 = com.javaservice.jennifer.server.timeactor.CleanerActor
02    PTA MONTH 12
```

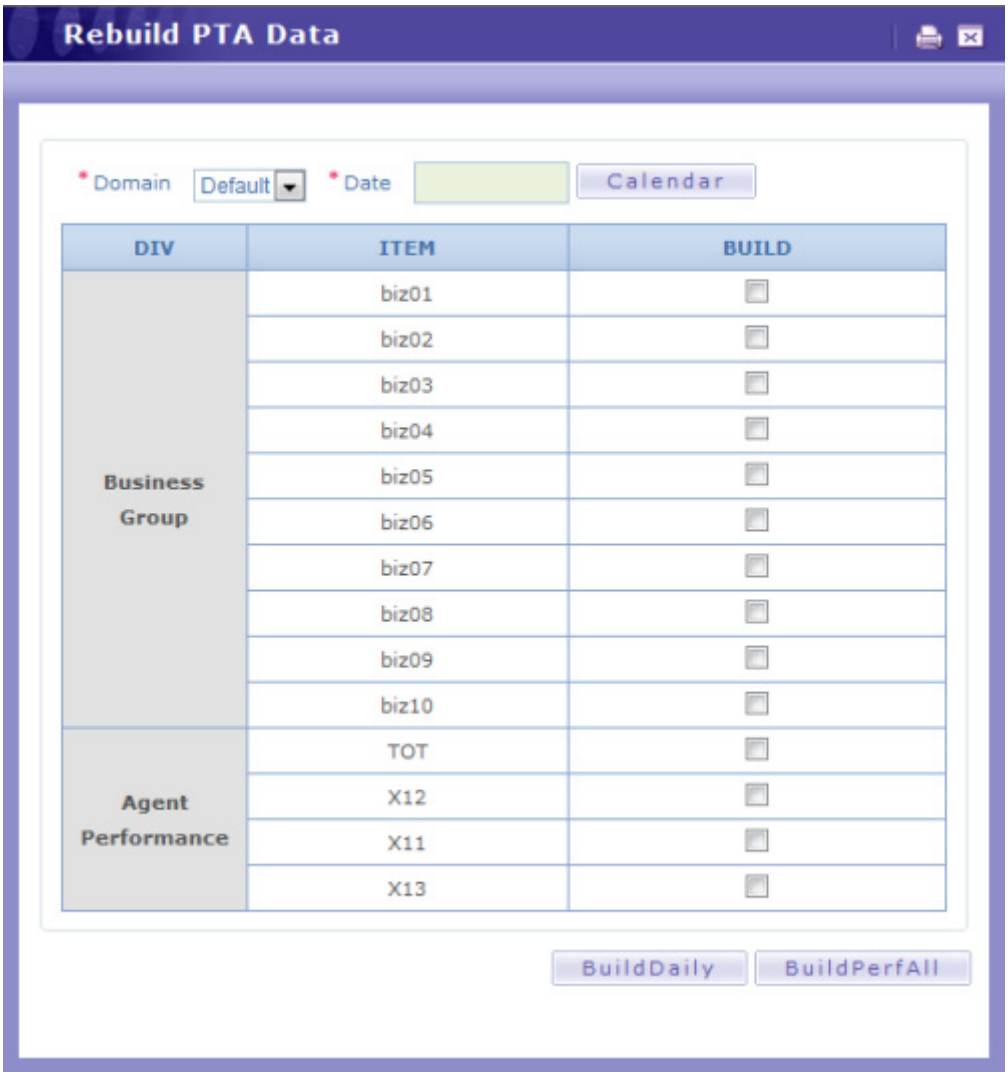
직접 디렉토리에서 데이터를 삭제할 수도 있다.

13.9. PTA데이타 빌드

이전버전의 제니퍼에서 v4.5버전업을 하거나 다른 이유로 PTA데이터를 다시 생성해야 하는 경우를 위해 리빌드 기능 추가되었다

PTA 리빌드를 위해서는 [Tools] 의 [Rebuild PTA Data]를 사용한다.

그림 13-62:PTA 데이터 빌드



13.9.1. 일자별 빌드

날짜를 선택하고 [BuildDaily] 를 클릭하면 해당 일자에 대한 PTA데이터를 생성한다. 에이전트별 데이터는 Database의 PERF_X테이블에서 복사하고 업무별 데이터는 X-View 파일 데이터를 복사한다.

13.9.2.PerfX 모두 빌드

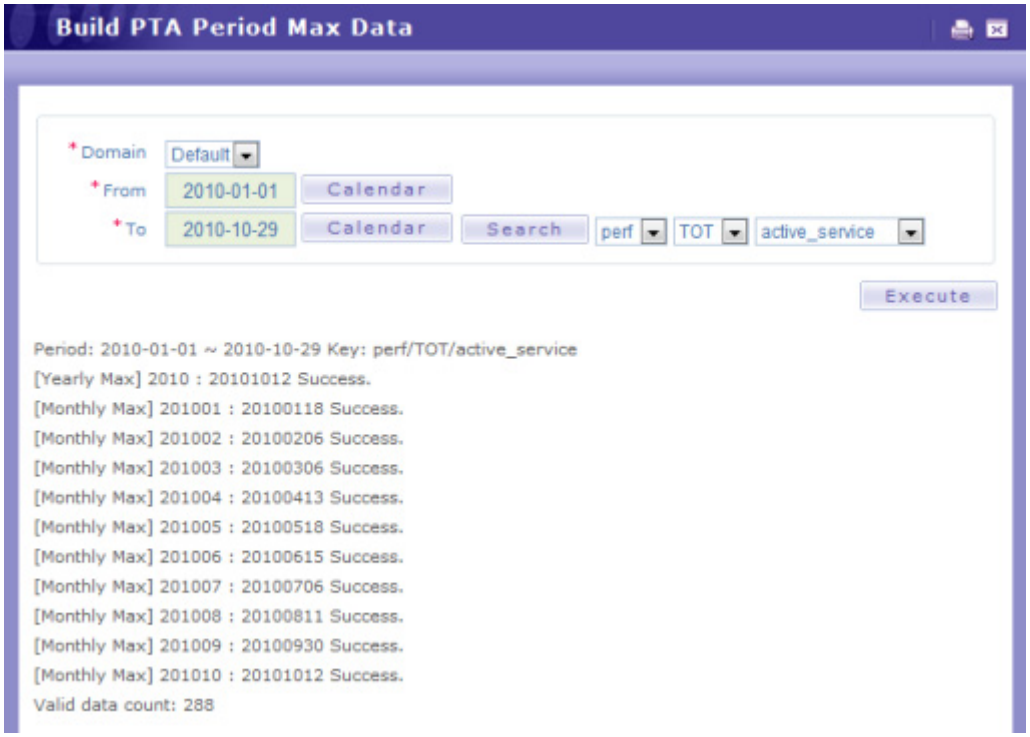
이제니퍼4.5로 업그레이드되면 PTA데이타거 없는데 하루 단위로 생성하면 매우 귀찮다 그런 경우에 [BuildPerfAll]을 실행한다.

이버튼이 클릭되면 화면에서 지정한 에이전트에 대해 모든 PerfX(에이전트별 성능 테이블)에 저장된 성능 정보를 복사하여 PTA데이터를 만든다

13.9.3.기간별 MAX데이터 생성하기

성능 데이터를 비교분석할때는 지정한 일자 뿐만 아니라 그해의 최대나 그달의 최대 값들과 비교하고자할때가 있다. 이경우에 사용하는 것이 [Build PTA Period MAX Data] 메뉴이다.

그림 13-63:기간별 MAX데이터 생성하기 1



최대 값을 구하기 위한 기간을 선택하고 구하고자하는 데이터를 선택하고 실행한다.

```
Period: 2010-01-01 ~ 2010-10-29 Key: perf/TOT/active_service
[Yearly Max] 2010 : 20101012 Success.
[Monthly Max] 201001 : 20100118 Success.
[Monthly Max] 201002 : 20100206 Success.
[Monthly Max] 201003 : 20100306 Success.
[Monthly Max] 201004 : 20100413 Success.
[Monthly Max] 201005 : 20100518 Success.
[Monthly Max] 201006 : 20100615 Success.
[Monthly Max] 201007 : 20100706 Success.
[Monthly Max] 201008 : 20100811 Success.
[Monthly Max] 201009 : 20100930 Success.
[Monthly Max] 201010 : 20101012 Success.
Valid data count: 288
```

위 예에서는 perf/TOT/active_service에 대한 최대 일자를 찾아서 “2010”이라는 대표 일자에 입력하고 마찬가지로 각 달의 최대 데이터를 찾아서 “2010XX”에 복사한다.

이값은 PTA화면에서 조회할때 날짜 부분에 “2010”이라고 입력하면 조회가 된다.

그림 13-64:기간별 MAX데이터 생성하기 1

13.9.4.기간별 MAX데이터를 위한 스케줄러

TimeActor를 이용하여 매일 월간 최대 값을 업데이트할 수 있다.

```
time_actor_22 = com.javaservice.jennifer.server.timeactor.PtaFindMax
02    perf/TOT/active_service
```

제니퍼 서버 설정에 위와 같이 기술하면 매일 새벽2시에 perf/TOT/active_service에 대한 당월 MAX와 당해MAX인 일자 데이터를 복사한다.

13.10.J-SCORE

J-SCORE는 여러가지 서로 다른 성능 관련 데이터를 조합하며 시스템을 위한 하나의 통합된 수치로 환산하여 표준 성능 지표를 만든다.

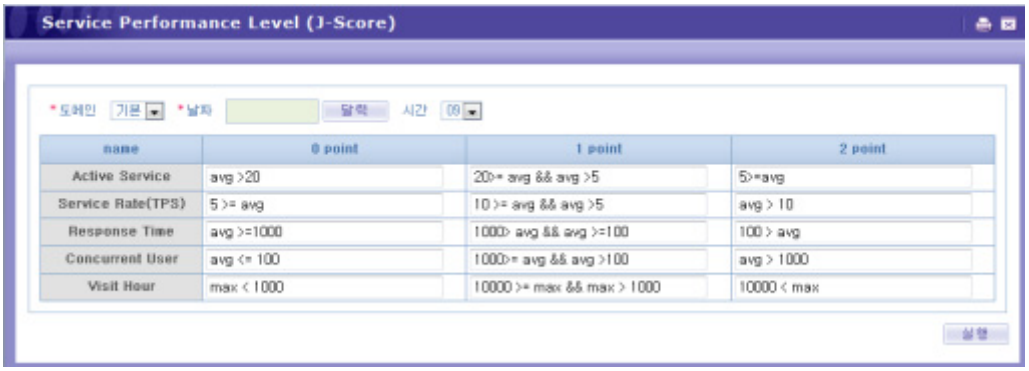
J-SCORE는 서로 다른 이질적인 수치를 하나의 표준 성능 수치로 변환하는데 의미가 있다. 여러가지 서로 다른 이질적인 비즈니스를 위한 시스템이 존재하는 상황에서 두 시스템간의 차이를 분석하고 그것을 적절하게 보정하여 J-SCORE를 산출하면 모든 관련 당사자들이 일관된 관점에서 성능 현황을 이해할 수 있다.

13.10.1.실행하기

J-SORE는 기능적으로는 단순하다. 화면을 열고 날짜와 시간을 선택하고 실행하면 결과를 확인할 수 있다.

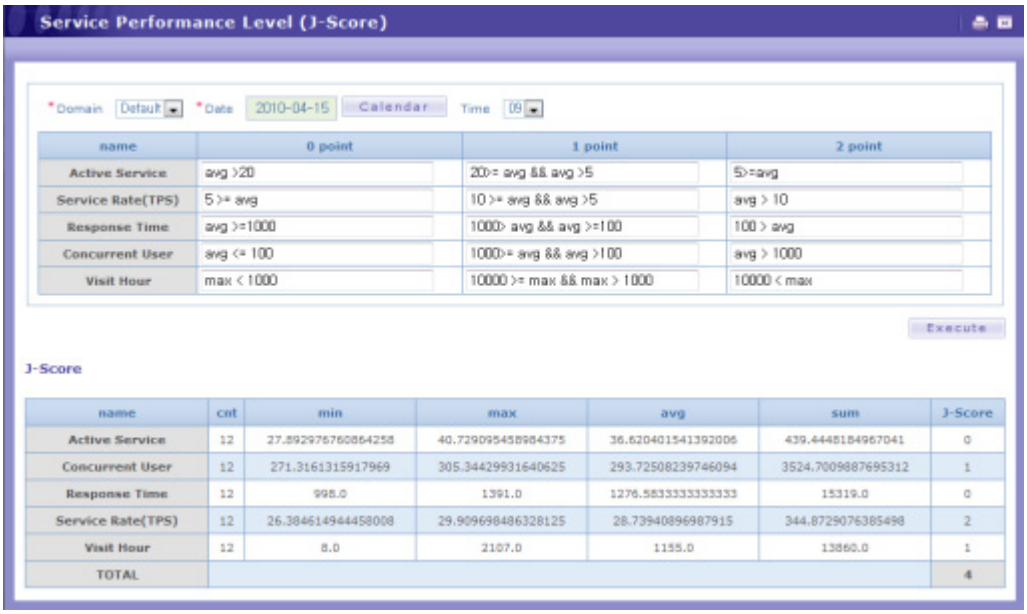
- 제니퍼 화면 좌측의 [Tools | J-Score] 메뉴를 클릭하면 아래와 같은 화면이 열린다.

그림 13-65:



- 날짜를 입력하고 오른쪽 아래의 “Execute” 버튼을 클릭하면 결과를 확인할 수 있다.

그림 13-66:



13.10.2. 설정파일 작성 및 등록

J-SCORE에 사용될 성능 지표들과 계산 식을 등록한다. 아래 파일에 샘플 설정이 있다.

```
JENNIFER_SERVER/bin/score.xml
```

샘플 설정내용은 다음과 같다.

```
<?xml version="1.0" encoding="utf-8" ?>
<jscore>
<rule>
  <id>perf/TOT/active_service</id>
  <name>Active Service</name>
  <score  point="0"><![CDATA[   avg >20            ]]></score>
  <score  point="1"><![CDATA[   20>= avg && avg >5 ]]></score>
  <score  point="2"><![CDATA[   5>=avg             ]]></score>
</rule>

...

<rule>
  <id>perf/TOT/visit_hour</id>
  <name>Visit Hour</name>
  <score  point="0"><![CDATA[   max < 1000            ]]></score>
  <score  point="1"><![CDATA[   10000 >= max && max > 1000 ]]></score>
  <score  point="2"><![CDATA[   10000 < max             ]]></score>
</rule>

</jscore>
```

Notice: PTA에 등록된 모든 데이터는 J-SCORE에 사용할 수 있다.

설정파일 작성법은 다음과 같다.

- <jscore>에는 여러개의 <rule>이 정의 될 수 있다.
- 하나의 <rule>에는 <id>와 <name> 테그가 필수로 정의되어 있어야 한다.
- <score>는 하나 이상 존재해야 한다.
- <score>에는 속성으로 point가 설정되어야 하는데 하나의 룰에는 동일한 point를 가진 <score>를 설정할 수 없다.
- 각 <score>를 위해 하나의 수식이 설정되는데 이때 설정되는 수직은 계산결과가 Boolean 값이어야 한다.
- 수식을 작성할 때는 파라미터를 사용할 수 있는데 사용 가능한 파라미터는 다음과 같다 .

표 13-38: 설정파일 작성 파라미터

이름	설명
cnt	선택한 시간의 데이터 중에서 0 이상인 값의 개수, 한시간의 데이터 수는 12개이다.(고정)
max	선택한 시간대 데이터 중에서 최대값
min	선택한 시간대 데이터 중에서 최소값
avg	선택한 시간대 데이터의 평균값
sum	선택한 시간대 데이터의 합

설정파일을 JENNIFER_SERVER/bin/score.xml이 아닌 다른 곳에 두려면 JENNIFER SERVER설정에 다음 옵션을 설정한다.

```
score_filename=/jennifer/server/myscore.xml
```

13.11. 보고서(웹리포트)

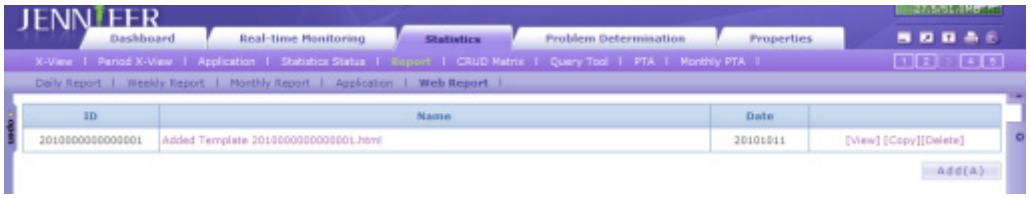
제니퍼에 추가된 웹리포트는 순수한 웹 기반 리포팅 모듈이다. 제니퍼에서 수집되는 각종 성능데이터를 손쉽게 출력할 수 있도록 해준다.

- 웹 기반 WYSIWYG 을 지원 하며, 별도의 솔루션 없이 보고서를 디자인하고 출력할 수 있어 편리하다.
- 보고서를 디자인 하기 위해 Component 개념을 체용하고 있다. 즉 보고서 Item은 별도로 수정되고 배포될 수 있으며, 사용자에 의해서도 생성 및 수정 될 수 있다.
- html/css/javascript 만으로 보고서를 생성하고 수정 한다. 차트나 내용은 서버 모듈이 담당하지만 출력될 출력물 자체는 모두 웹기반으로 처리된다. 그 만큼 유연하고 편리하며 웹이 가질 수 있는 장점을 모두 수용 한다. 브라우저만으로 출력이 가능하다.

13.11.1. 빠른 시작

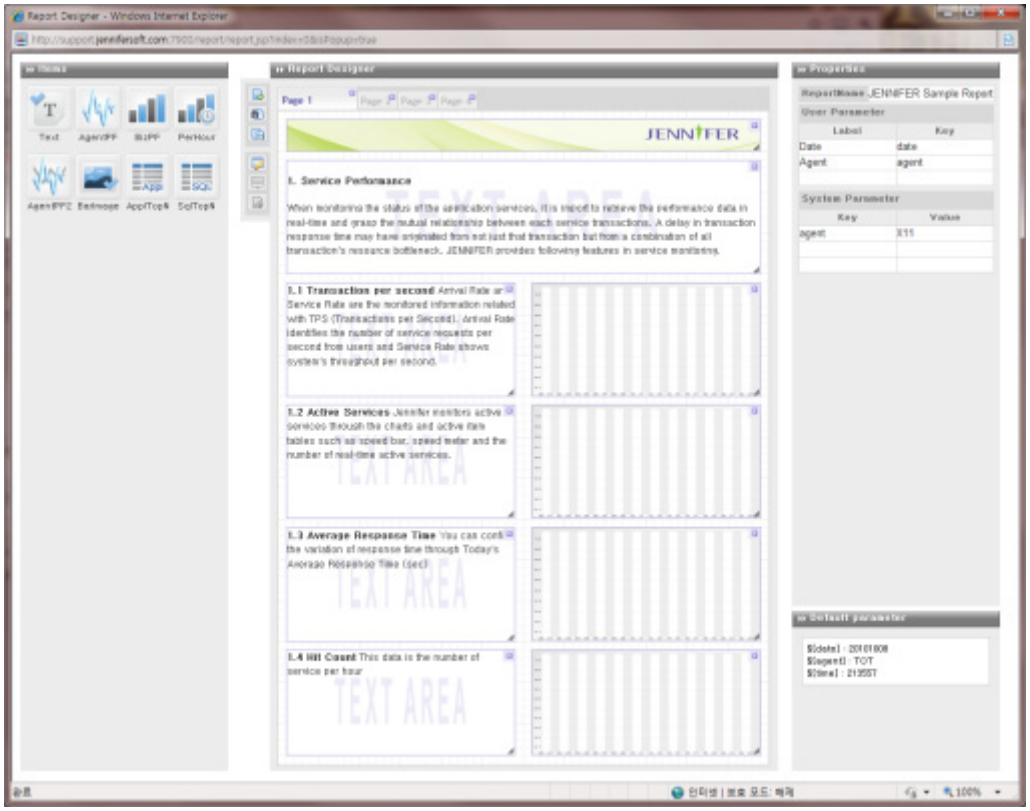
제니퍼4.5 버전을 설치하고 Statistics/Report/WebReport 메뉴를 선택하면 다음과 같은 리스트화면을 볼 수 있다.

그림 13-67:리스트 화면



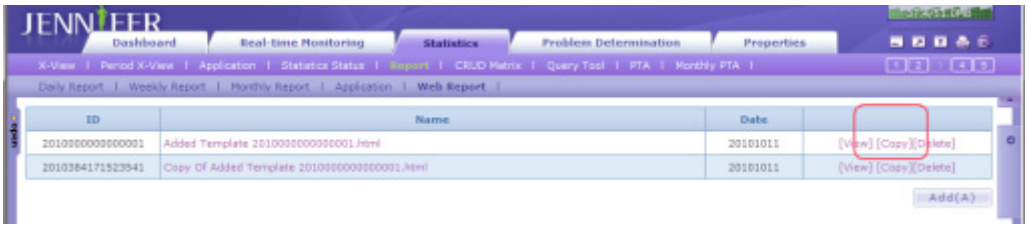
제니퍼4.5 배포본에는 샘플 보고서가 포함되어있다. 리스트에서 샘플을 오픈하면 아래와 같이 보고서 탬플릿을 확인할 수 있다.

그림 13-68:보고서 템플릿



원본 템플릿을 수정하면 복구할 수 없으니 템플릿을 복사하여 작업을 한다. 리스트에서 [copy]를 클릭하면 템플릿이 복사된다.

그림 13-69:템플릿 복사

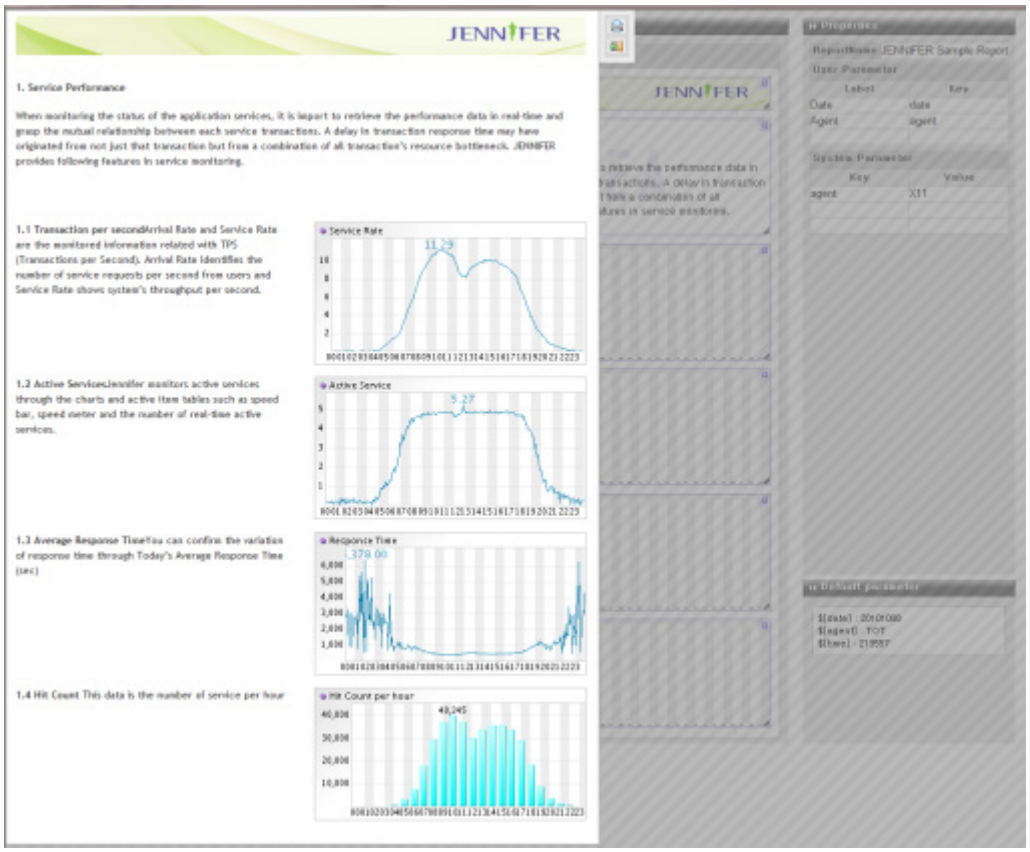


복사된 템플릿을 클릭하여 오픈한다.

그림 13-70:템플릿 오픈

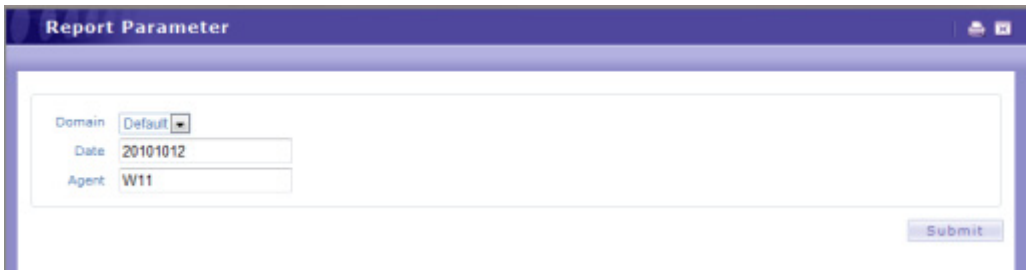
Preview 버튼을 클릭하면 현재 페이지에 대한 출력화면을 확인할 수 있다.

그림 13-71:보고서 출력화면



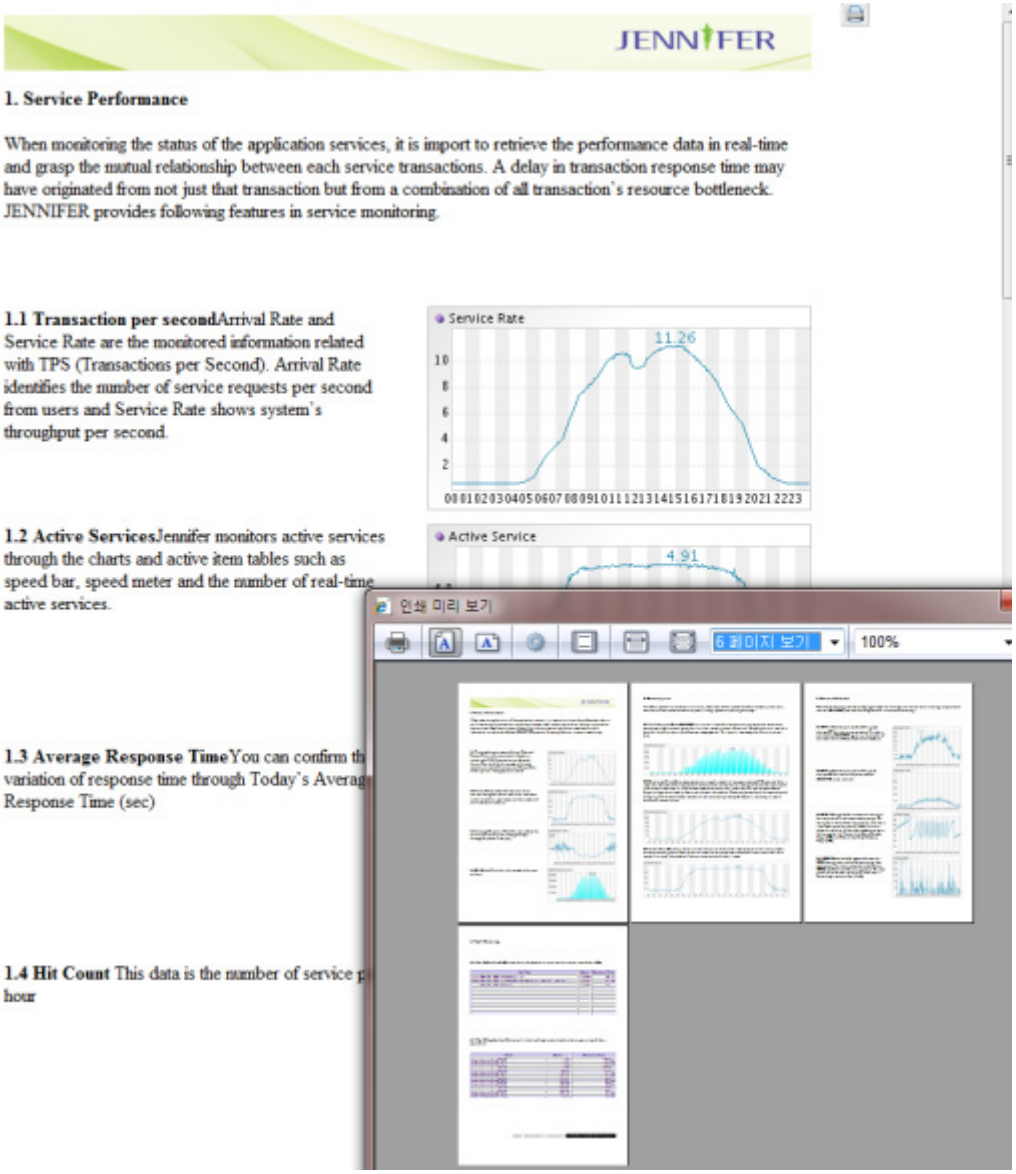
다시 이전화면으로 돌아가서 [view]버튼을 클릭하면 아래와 같이 입력화면을 볼 수 있다.

그림 13-72:입력화면



파라미터를 입력하고 Submit을 클릭하면 아래와 같은 출력할 수 있는 인쇄 화면이 나타난다.

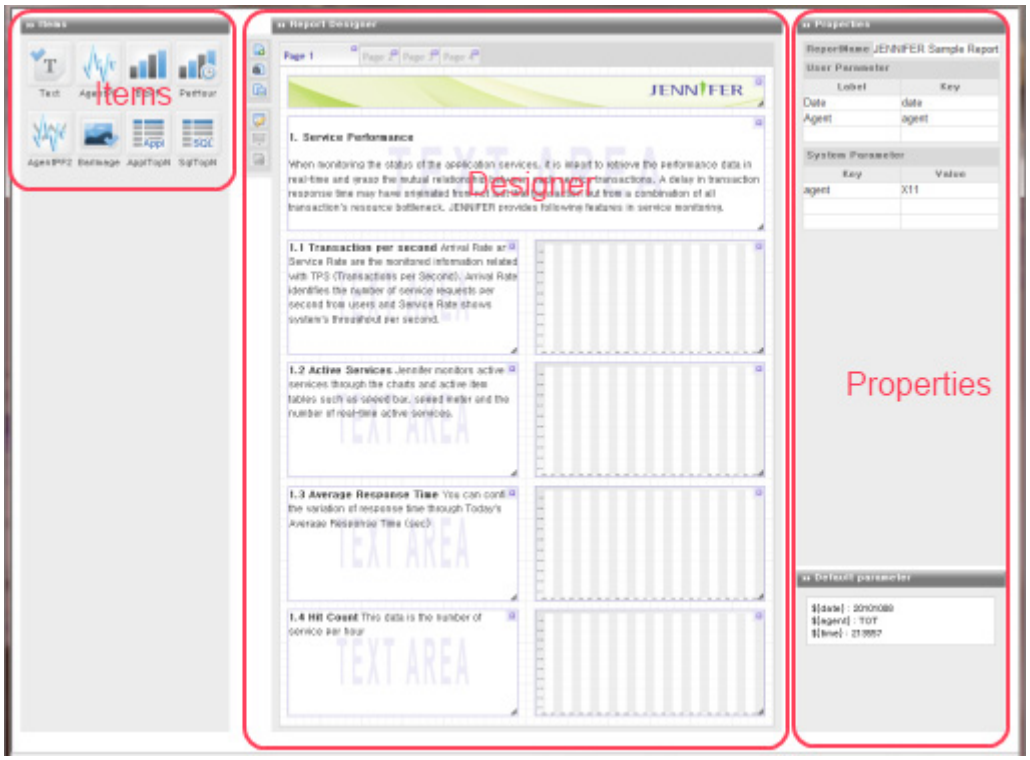
그림 13-73:인쇄화면



13.11.2.화면 설명

제니퍼의 웹리포트 템플릿 디자인화면은 크게 3개의 영역으로 구분할 수 있다.

그림 13-74:디자인 화면



13.11.2.1.Items

차트나 테이블을 생성하는 보고서 컴포넌트이다. 리포트 아이템은 개별로 작성되고 배포 될 수 있다. 버전업 될 때 임의로 추가 될 수 있다. 또한 사용자도 자신이 원하는 아이템을 만들어 추가 할 수 있다.

13.11.2.2.Report Designer

보고서 템플릿을 작성하는 곳이다. 페이지 단위로 디자인한다. 빈 페이지에는 Items에서 아이템 하나를 드레그하여 편집한다. 하나의 아이템을 클릭하면 오른쪽 Properties 영역에서 해당 요소에 대한 속성 값들을 수정 할 수 있다.

13.11.2.3.Properties

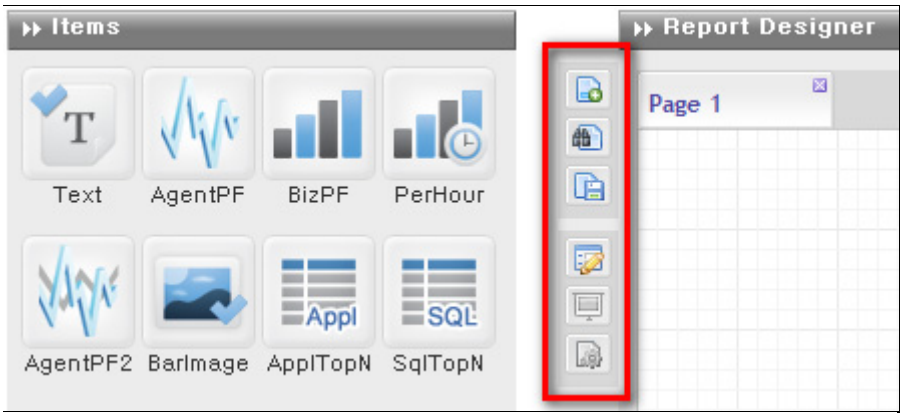
각 요소에 대한 속성 값을 설정하는 곳이다.

- 리포트 속성 : 리포트 전체에서 사용할 수 있는 속성으로 파라미터를 추가 설정 할 수 있으며, 페이지를 나타내는 탭이 있는 부분을 클릭하여 설정 할 수 있다.
- 페이지 속성 : 한페이지에서 사용할 속성 값으로 탭의 페이지 이름(글자 부분)을 클릭하여 설정 한다.

- 아이템 속성 : 아이템 하나가 사용할 속성 값들을 입력/수정 한다. 각 아이템을 클릭하면 설정 할 수 있다.

13.11.2.4.웹리포트 메뉴

그림 13-75:웹리포트 메뉴

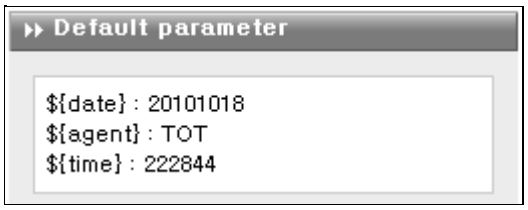


- 페이지 추가 : 새로운 페이지를 추가 한다.
- 미리보기 : 현재 페이지에 입력 결과를 적용 하여 출력 결과를 미리 보기 한다.
- 저장 : 작성한 보고서 템플릿을 저장 한다.
- 출력보기 : 현재 작성된 보고서 템플릿을 출력 전 단계로 실행 한다. 만약 사용자 입력 파라미터를 설정 하였을 경우 입력 파라미터에 대한 입력 창이 실행 된다.
- 발표(PT) : 보고서 템플릿을 PT(presentation) 모드로 실행 한다.(구현 예정)
- 보정 : 보고서 디자이너를 이용한 템플릿과 실제 출력 될 출력물 간의 비율을 보정 한다 .(구현 예정)

13.11.3.입력 파라미터 설정

웹 리포트는 속성 값에 대한 파라미터 처리가 가능 하다. 파라미터는 일반적으로 자주 변경되거나 반복되는 속성 값을 지정 할 때 사용 하는데 보고서 디자이너 부분을 클릭하여 정의 할 수 있다.

그림 13-76:입력 파리미터 설정



웹 리포트에서 사용하는 파라미터는 다음 3가지가 있다.

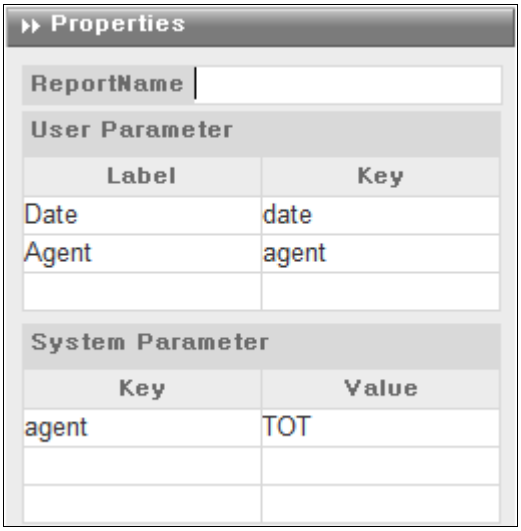
- 디폴트 파라미터 : 기본적으로 적용 되는 파라미터로 ${agent}, ${date}, ${time} 가 정의 되어 있다.
- 시스템 파라미터 : 시스템 파라미터에 정의 된 파라미터는 “출력보기” 할 경우 기본으로 적용되는 파라미터 이다.
- 사용자 파라미터 : 사용자 파라미터는 “출력보기” 할 경우 사용자가 그 값을 입력하거나 수정 할 수 있는 파라미터 이다.

동일한 이름의 파라미터를 지정 하였을 경우 적용되는 우선 순위는 사용자 파라미터, 시스템 파라미터, 디폴트 파라미터 이다.

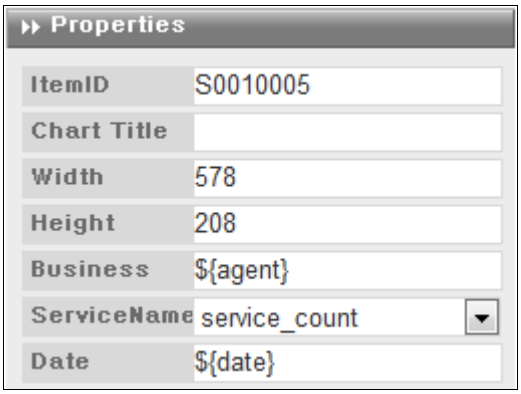
파라미터를 정의 하는 방법은 사용자 파라미터를 적용 할 경우 Label 부분에 화면상에 표시될 라벨명을 입력하고 Key 부분에는 아이템에서 사용할 key(영어와 숫자를 허용)를 정의 하면 된다. (key에 이미 정의되어 있는 디폴트 파라미터나 시스템 파라미터와 동일하게 key를 정의 하면 디폴트 값이 해당 값으로 입력 된다.)

사용자 파라미터에 value 대신 Label이 있는 이유는 사용자 파라미터는 사용자로 부터 입력 받아 처리되는 파라미터 이기 때문이다.

시스템 파라미터를 정의 하는 방법은 key를 정의 하고, value 값에 해당 key가 가질 값을 입력 하면 된다.(보통 상수 값으로 보면 된다.)

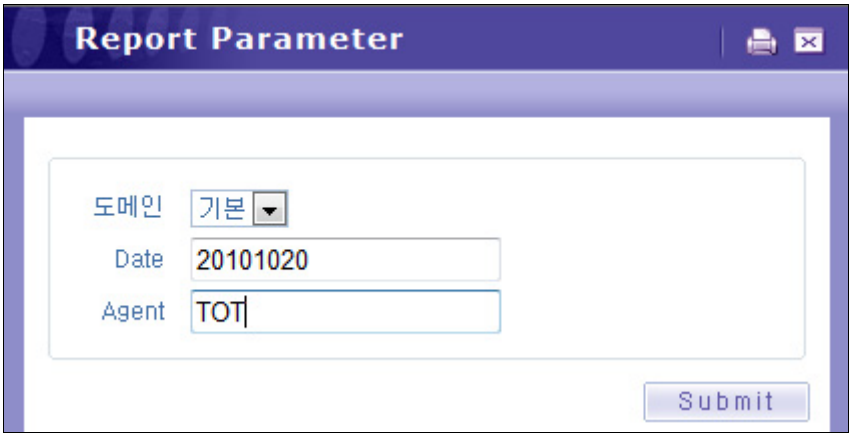


key가 정의된 파라미터는 아이템의 속성에서 “${key}” 형식으로 사용할 수 있다.



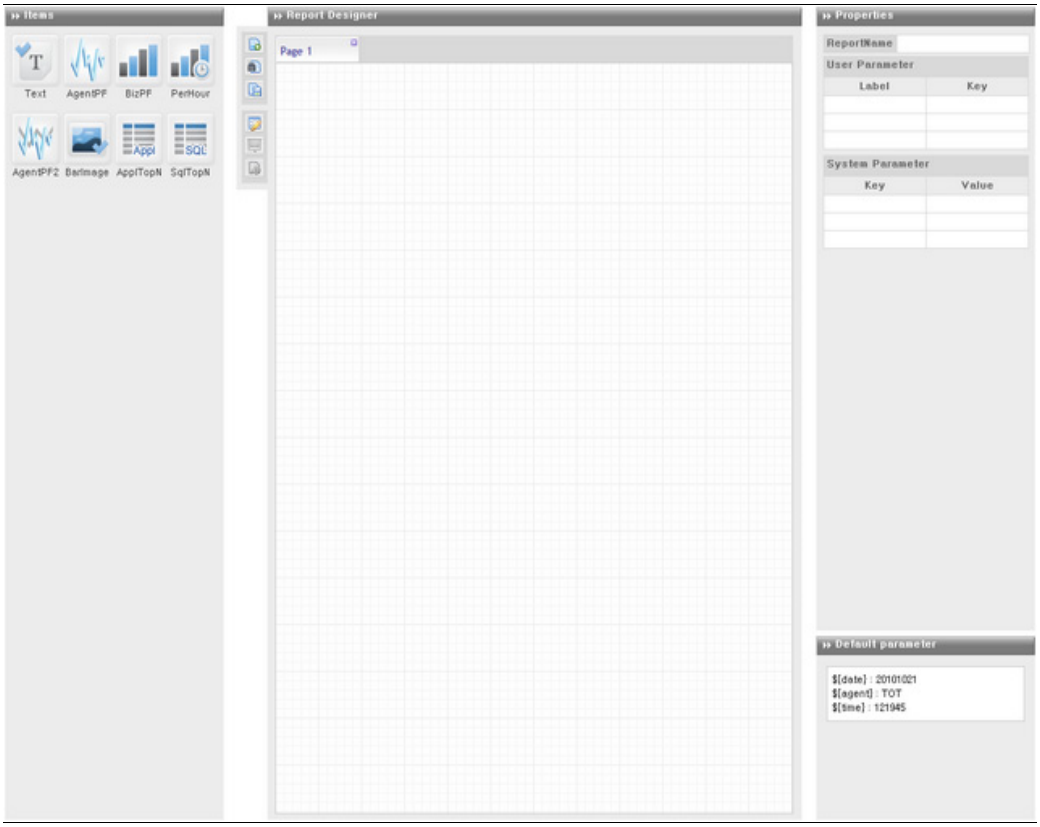
설정된 파리미터는 - “출력보기” 버튼을 클릭하여 아래와 같이 확인 할 수 있다.(보고서를 저장 후 출력보기 버튼을 클릭 하여야 한다.)

사용자 파라미터만 “출력보기”에서 입력 받을 수 있는 폼으로 변환 되어 사용자로 부터 값을 입력 받아 적용 할 수 있다.

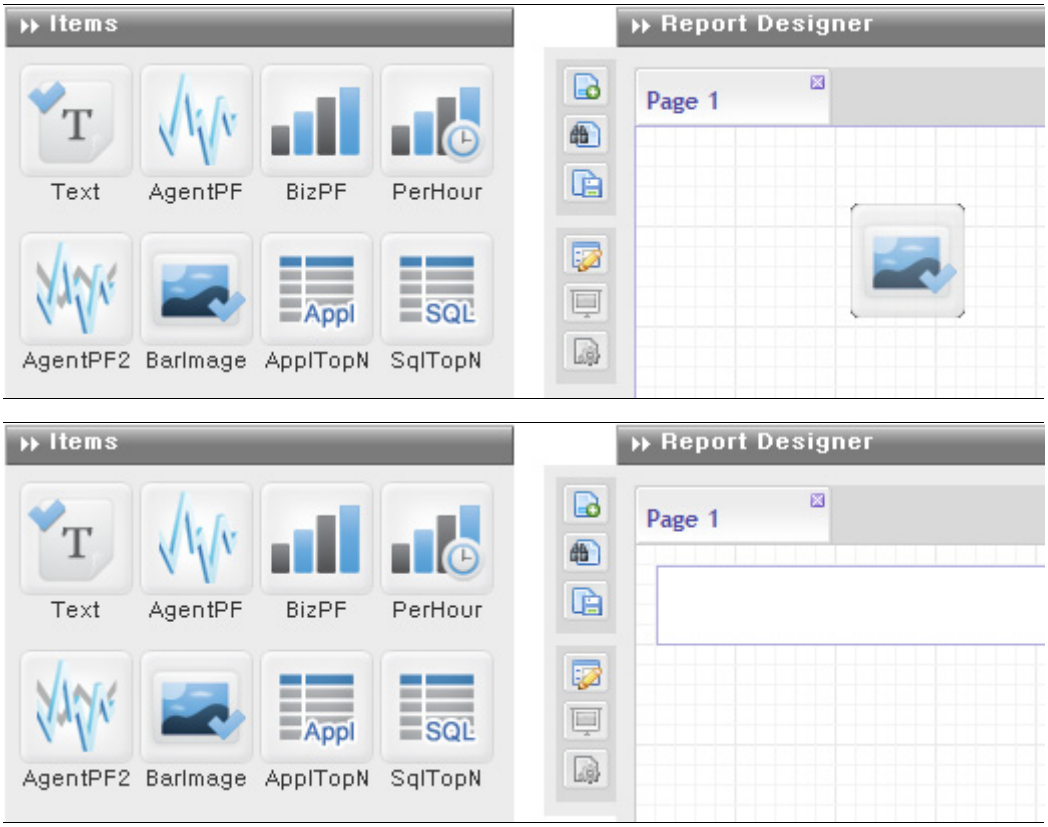


13.11.4.리포트 작성하기

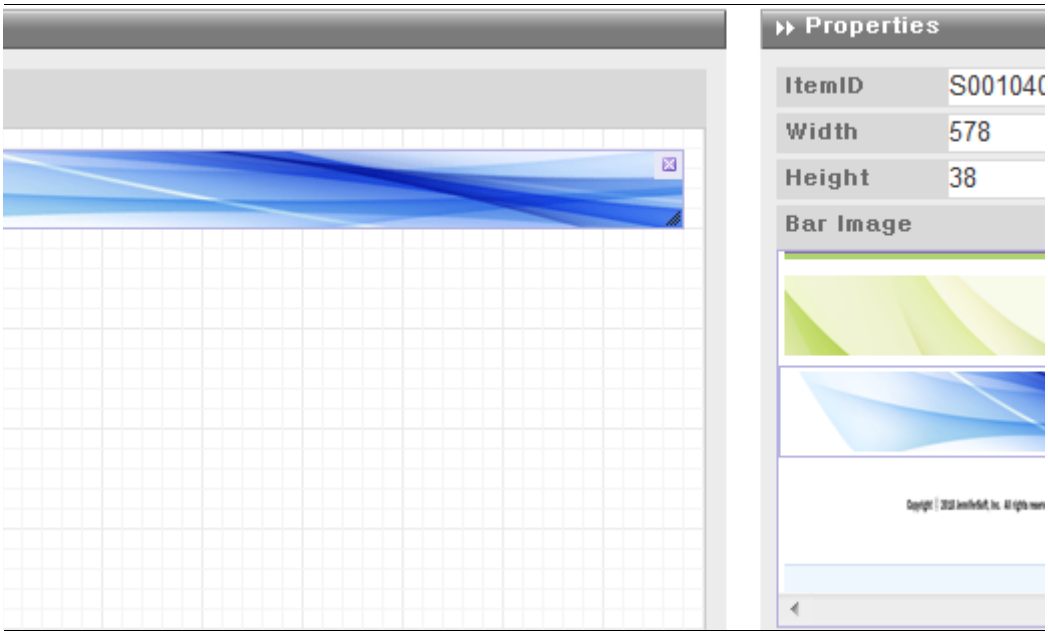
보고서 템플릿 목록에서 추가(Add) 버튼을 선택 하여 보고서 디자이너를 통해 템플릿 작업을 시작 한다.



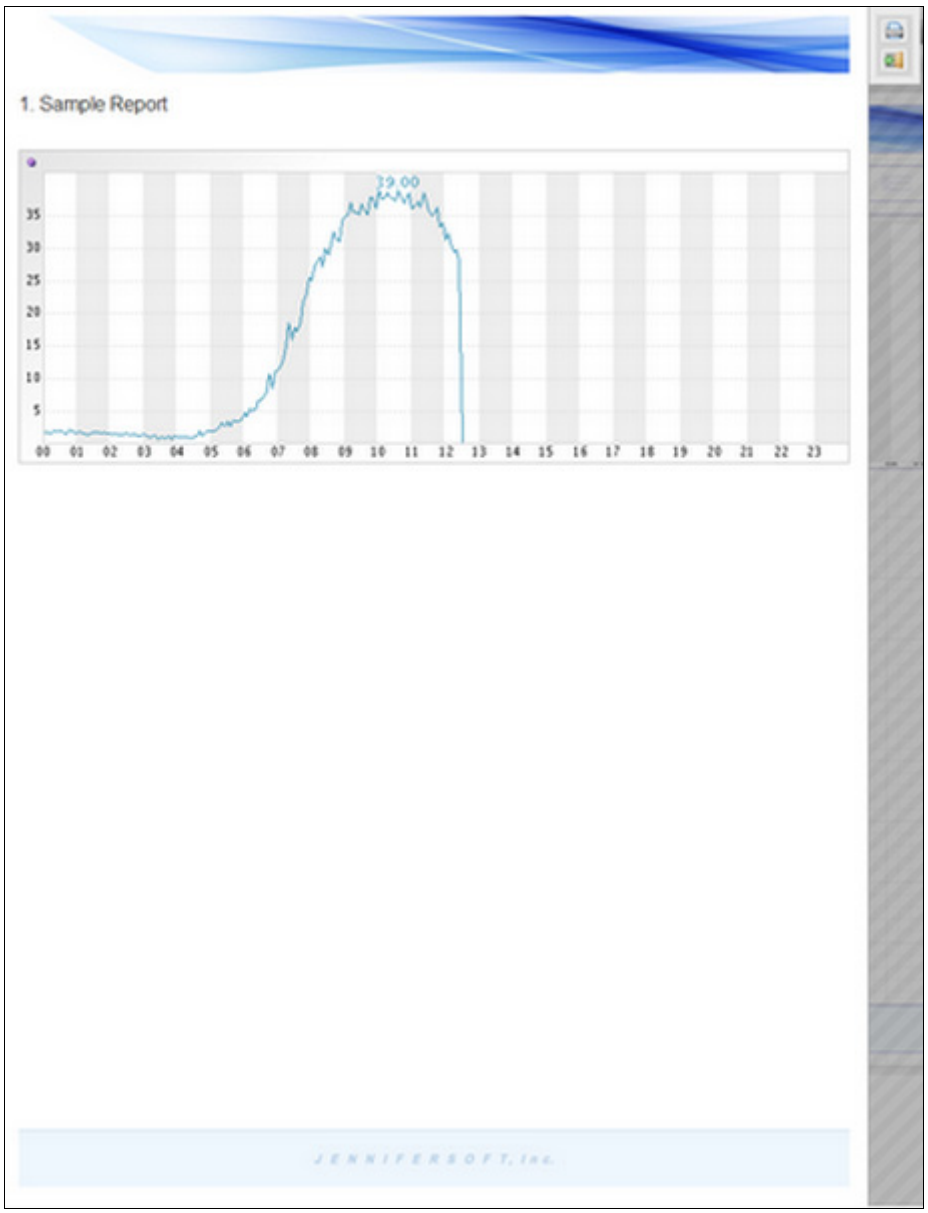
아이템을 디자인 영역(페이지)으로 드래그 하여 배치 한다. 화면에서는 BarImage 아이템을 배치 하였다.



페이지에 배치된 아이템을 클릭하여 속성 설정을 한다.

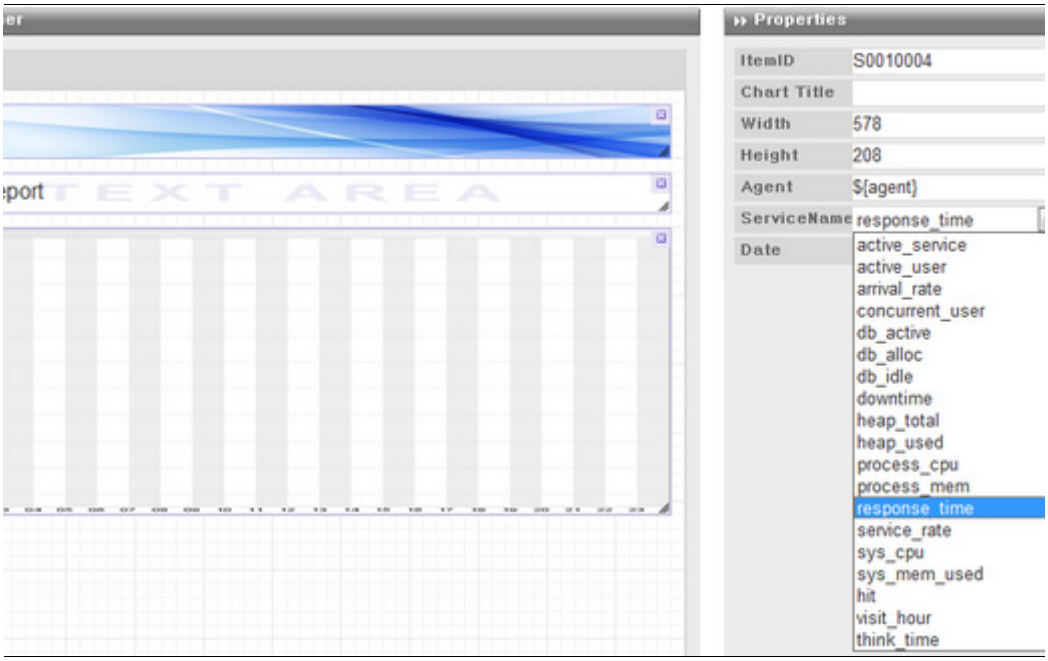


추가 하고 싶은 몇 개의 아이템을 더 추가 한 다음 (미리보기) 버튼을 클릭하여 배치된 아이템들이 원하는 대로 배치가 되었는지 확인 한다.



아이템을 추가 한 후 바로 미리보기 하면 아이템 속성의 기본 값들이 적용된다. 추가한 아이템(화면에서는 AgentPF)을 선택하여 속성 값을 변경 한다.

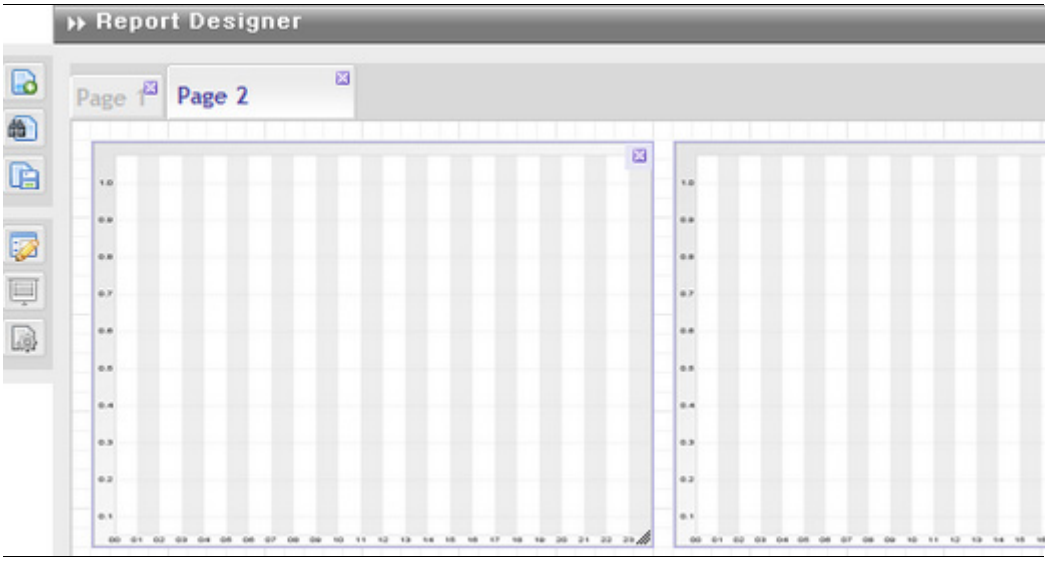
(미리보기에서는 페이지 단위로 출력이 가능 하다. 현재 선택되어 있는 페이지를 출력 할 수 있다.)



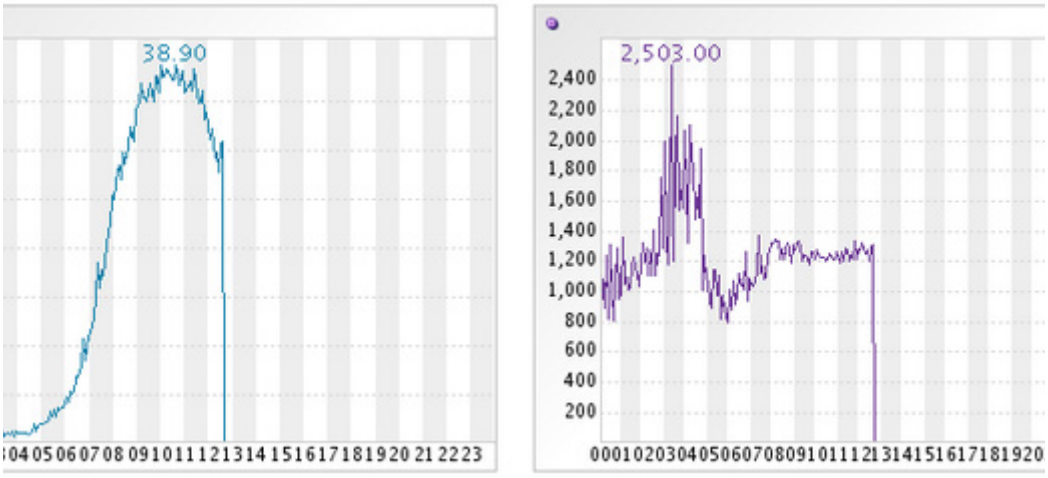
변경된 값이 잘 적용되었는지 확인 한다.

추가 버튼을 클릭 페이지를 추가 해 보자.

(페이지는 최대 10페이지 까지 추가 가능 하다.)

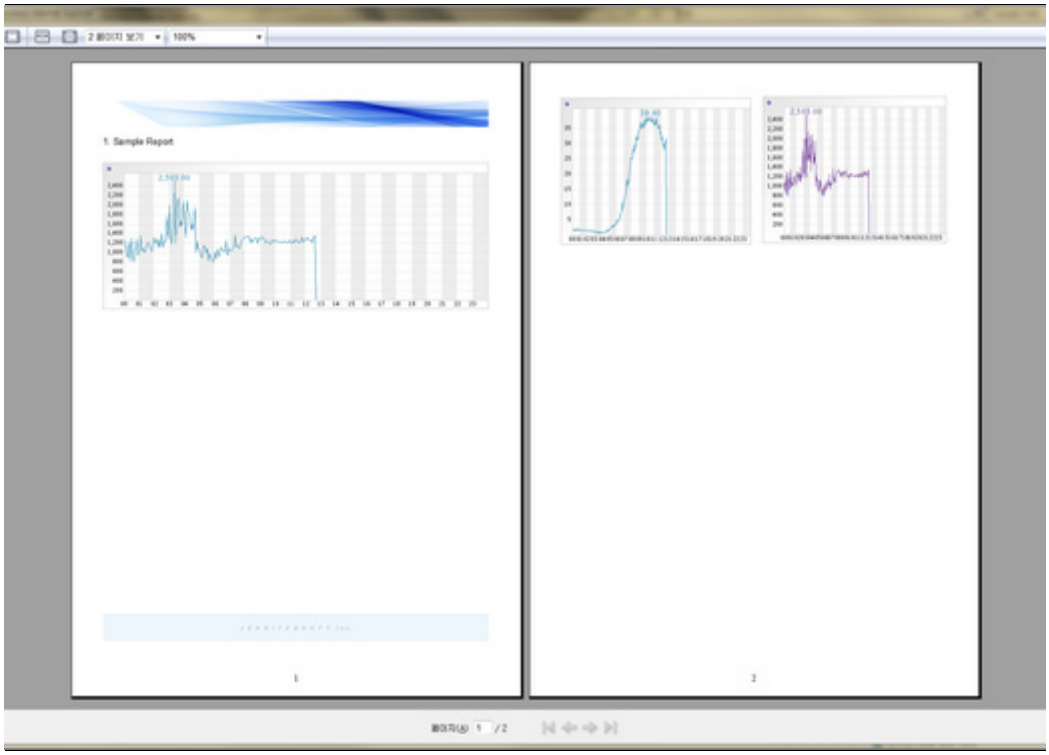


이전 페이지와 동일하게 원하는 아이템들을 배치하고 속성 들을 설정하여 페이지를 작성 한다.



보고서 템플릿을 다 작성 한 후 저장 버튼을 클릭하여 작성한 템플릿을 저장 한다.

출력보기 버튼을 클릭하여 작성된 템플릿을 실행한다. 실행된 템플릿의 결과를 확인 한 후 프린트 한다.



13.11.5.아이템 설명

기본적으로 추가되는 아이템은 8개 이다.(아이템 들은 우선순위에 따라 지속적으로 추가 될 예정이다.)

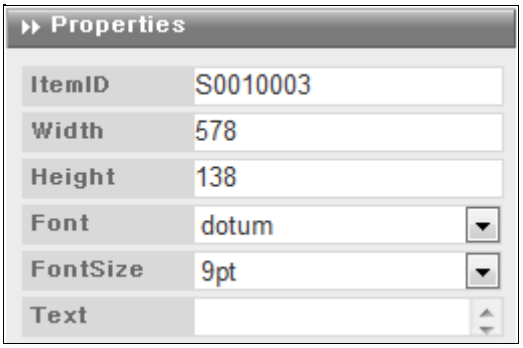
아이템들은 공통적인 속성으로 ItemID, Width, Height를 가진다.

ItemID는 현재 보고 있는 속성이 어떤 아이템인지 구분하기 위해 사용하고, Width, Height는 화면에 출력될 아이템의 크기를 조절 하기 위해 사용한다.

13.11.5.1.Text

Text 아이템으로 일반적인 텍스트를 입력 하기 위한 아이템 이다. 기본적인 Text와 Html이 입력 가능하나 Html은 잘못 사용할 경우 원하는 화면이 나오지 않을 수도 있으니 간단한 Html 만 사용하기를 권한다.

Text 아이템은 다음과 같은 속성을 가진다.

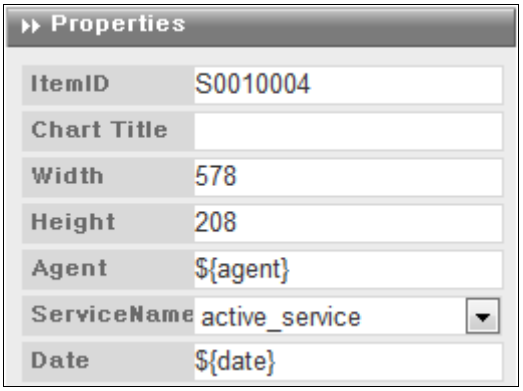


- Font : Text의 폰트를 설정한다.
- FontSize : Text의 크기를 설정한다.
- Text : 페이지에 표시될 텍스트를 입력한다.

13.11.5.2.AgentPF

에이전트별 성능 정보를 차트로 표현한다.

AgentPF 아이템은 다음과 같은 속성을 가진다.



- ChartTitle : 챠트에 표시될 제목을 의미한다.
- Agent : 조회할 데이터의 에이전트를 의미한다.
- ServiceName : 조회할 데이터를 선택한다.
- Date : 조회할 데이터의 날짜를 입력한다.

선택할 수 있는 ServiceName

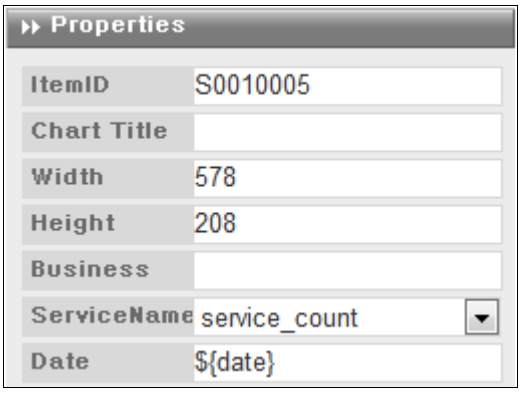
- active_service : 에이전트별 액티브서비스
- active_user : 에이전트별 액티브 유저

- arrival_rate : 에이전트별 Arrival Rate
- concurrent_user : 에이전트별 동시 사용자
- db_active : 에이전트별 액티브 DB연결(TOT 제외)
- db_alloc : 에이전트별 사용중인 DB연결
- db_idle : 에이전트별 비사용중인 DB연결
- downtime : 5분당 에이전트별 다운타임(최대 300초)
- heap_total : 에이전트별 최대 HEAP 메모리
- heap_used : 에이전트별 HEAP 메모리 사용량
- process_cpu : 에이전트별 프로세스 CPU사용량(%)
- process_mem : 에이전트별 프로세스 메모리 사용량(MB)
- response_time : 에이전트별 평균 응답시간(ms)
- service_rate : 에이전트별 초당 처리율(TPS)
- sys_cpu : 에이전트가 수행된 시스템 전체 CPU사용량
- sys_mem_used : 에이전트가 수행되는 시스템 전체 메모리 사용량
- hit : 에이전트별 5분당 처리건수
- visit_hour : 에이전트별 시간당 방문자 수
- think_time :에이전트별 Think Time

13.11.5.3.BizPF

BizPF 아이템은 업무별 성능 정보를 시간당 막대 차트로 출력한다.

BizPF 아이템은 다음과 같은 속성을 가진다.



- ChartTitle : 챠트에 표시될 제목을 의미한다.

- Business : 조회할 데이터의 비지니스를 설정한다.
- ServiceName : 조회할 데이터를 선택한다.
- Date : 조회할 데이터의 날자를 의미한다.

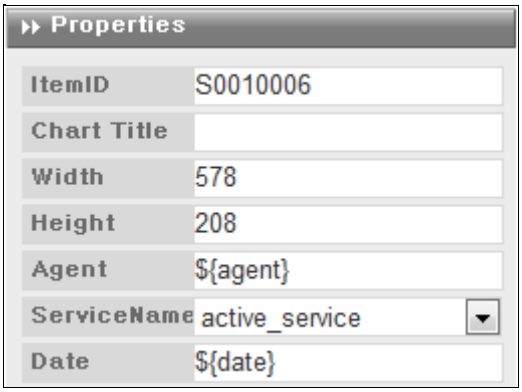
선택할 수 있는 ServiceName

- service_count : (SUM)업무별 5분당 호출 건수
- service_time : (Average)업무별 평균 응답시간(ms)
- sql_time : (Average)업무별 SQL 평균 수행시간(ms)
- error_count : (SUM)업무별 5분당 에러 건수
- etx_time : (Average)업무별 외부 트랜잭션 평균 수행시간(ms)
- sla_fail : (SUM)업무별 5분당 SLA 기준값(업무 그룹에서 설정)을 초과한 서비스 수
- client_time : (Average)최종 사용자 응답시간(브라우저 응답시간 설정시만 사용가능)
- turnaround_count ; (SUM)최종사용자 응답 건수(브라우저 응답시간 설정시만 사용가능)

13.11.5.4.PerHour

PerHour 아이템은 에이전트별 성능 정보를 시간당 막대 그래프로 출력한다.

PerHour 아이템은 다음과 같은 속성을 가진다.



- ChartTitle : 챠트에 표시될 제목을 의미한다.
- Agent : 조회할 데이터의 에이젼트를 의미한다.
- ServiceName : 조회할 데이터의 서비스명을 의미한다.

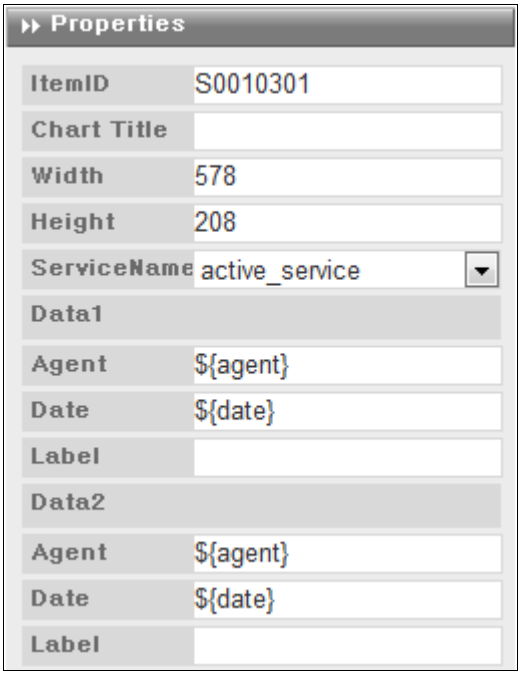
- Date : 조회할 데이터의 날자를 의미한다.

Notice: 선택할 수 있는 ServiceName 항목은 AgentPF과 동일하다 단 시간당 데이터로 변경할때 “hit” 와 “down_time” 은 시간당 SUM을 visit_hour는 해당 시간의 최대값을 사용한다.

13.11.5.5.AgentPF2

AgentPF2 아이템은 두개의 에이전트별 성능데이터를 비교하는 차트를 출력한다.

AgentPF2 아이템은 다음과 같은 속성을 가진다.



- ChartTitle : 챠트에 표시될 제목을 의미한다.
- ServiceName : 조회할 데이터를 선택한다.
- Agent : 조회할 데이터의 에이젼트를 의미한다.
- Label : 챠트에 표시될 데이터의 라벨을 의미한다.

Notice: 선택할 수 있는 ServiceName 은 AgentPF과 동일하다 “Data1” 과 “Data2” 를 위한 데이터를 입력한다.

13.11.5.6.BarImage

BarImage 아이템은 주로 보고서의 제목이나 머리말, 꼬릿말 등에 사용할 수 있는 이미지 아이템이다.

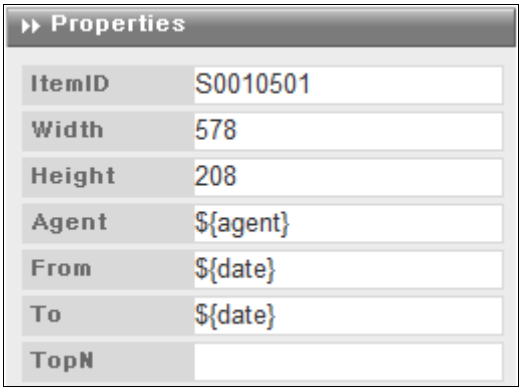
BarImage 아이템은 다음과 같은 속성을 가지고 있다.

- Bar Image : 이미지를 선택 하면 페이지에 등록한 아이템에 적용 된다.

13.11.5.7.ApplTopN

ApplTopN 아이템은 지정한 기간에 수행된 서비스 중에서 느린 것부터 N개를 출력한다.

ApplTopN 아이템은 다음과 같은 속성을 가지고 있다.

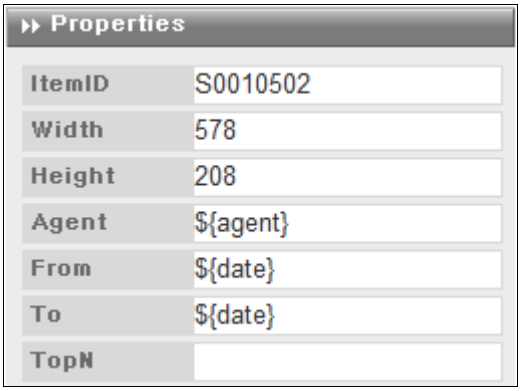


- Agent : 조회할 데이터의 에이젼트를 입력한다.(TOT는 전체)
- From : 조회할 데이터의 시작일를 입력한다.
- To : 조회할 데이터의 종료일를 입력한다.
- TopN : 상위 몇 개 까지 조회 할 지를 입력한다.

13.11.5.8.SqlTopN

SqlTopN 아이템은 지정한 기간에 지정한 에이전트에서 수행된 SQL중에서 느린 순서로 N개를 출력한다.

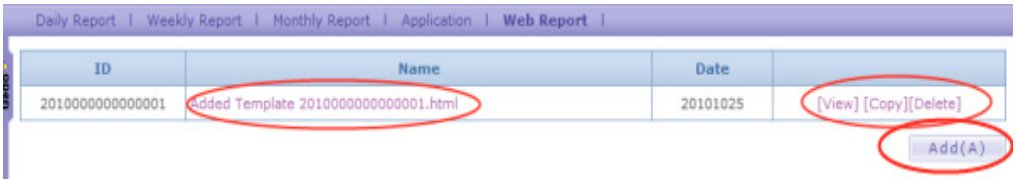
SqlTopN 아이템은 다음과 같은 속성을 가지고 있다.



- Agent : 조회할 데이터의 에이젼트를 입력한다.(TOT는 전체)
- From : 조회할 데이터의 시작일를 입력한다.
- To : 조회할 데이터의 종료일를 입력한다.
- TopN : 상위 몇 개 까지 조회 할 지를 입력한다.

13.11.6.보고서 템플릿 관리하기

보고서는 템플릿단위로 복사되고 수정될 수 있다.



- 보고서 이름 : 보고서를 편집하는 화면이 열린다.
- [View] : 해당 보고서 출력을 위한 파라미터 화면이 열리고 파라미터를 입력하고 클릭하면 보고서가 출력된다.
- [Copy] : 해당 보고서를 복사한다.
- [Delete] : 보고서를 삭제한다.
- Add버튼 : 새로운 보고서를 작성한다.

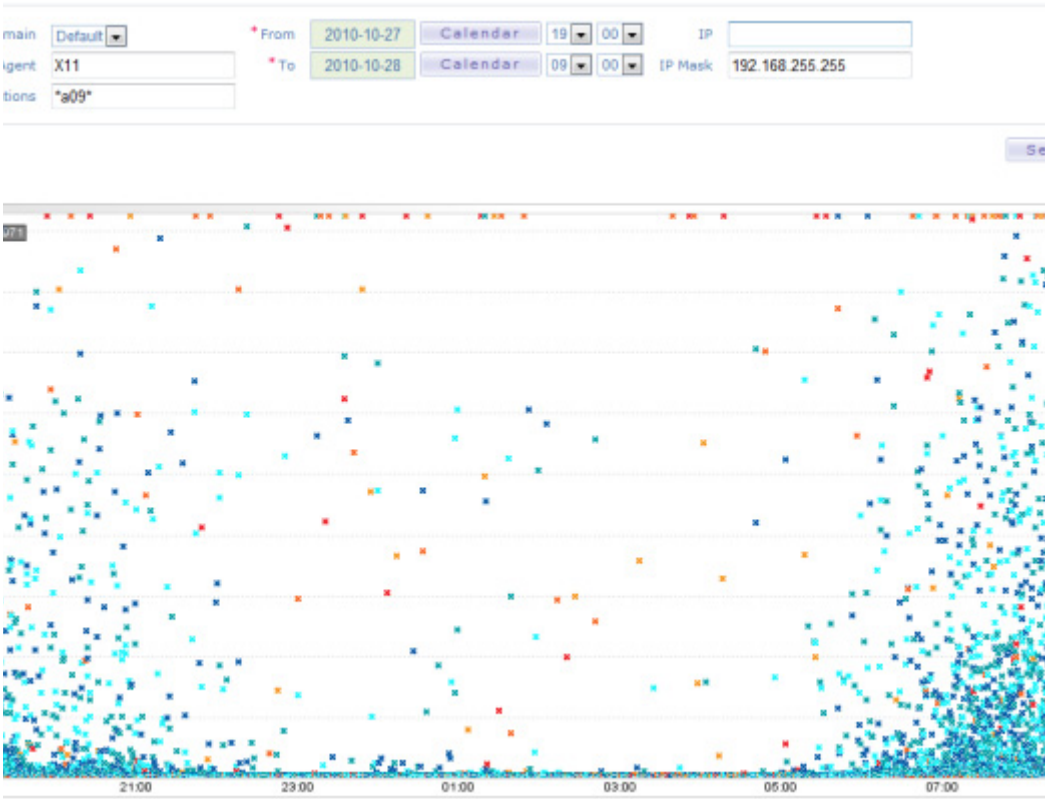
13.12. 고급 통계

제니퍼4.5이후에 새롭게 추가된 통계 기능에 대해서 사용법을 설명한다.

13.12.1. 기간 X-View

일정 기간동안의 XView를 조회하는 기능이다. 장기간에 걸쳐 일정조건의 서비스의 성능을 분석할때 사용한다.

그림 13-77: 기간 X-View



Notice: 단 너무 많은 데이터를 조회하는 경우 메모리 에러가 발생할 수 있음으로 적절하게 조회 조건을 설정해야 한다.

13.12.1.1. 조회조건 설정

- From/To: 검색을 원하는 시간을 설정한다.

- Agent : 에이전트를 지정한다.
- Applications : 필터링할 애플리케이션 서비스명을 설정한다. 설정 방법은 아래와 같다

```
*AAA* : AAA를 포함한 서비스 명
   BBB* : BBB로 시작하는 서비스 명
   *CCC : CCC로 끝나는 서비스 명
  AA*DD : AA로 시작해서 DD로 끝나는 서비스 명
```

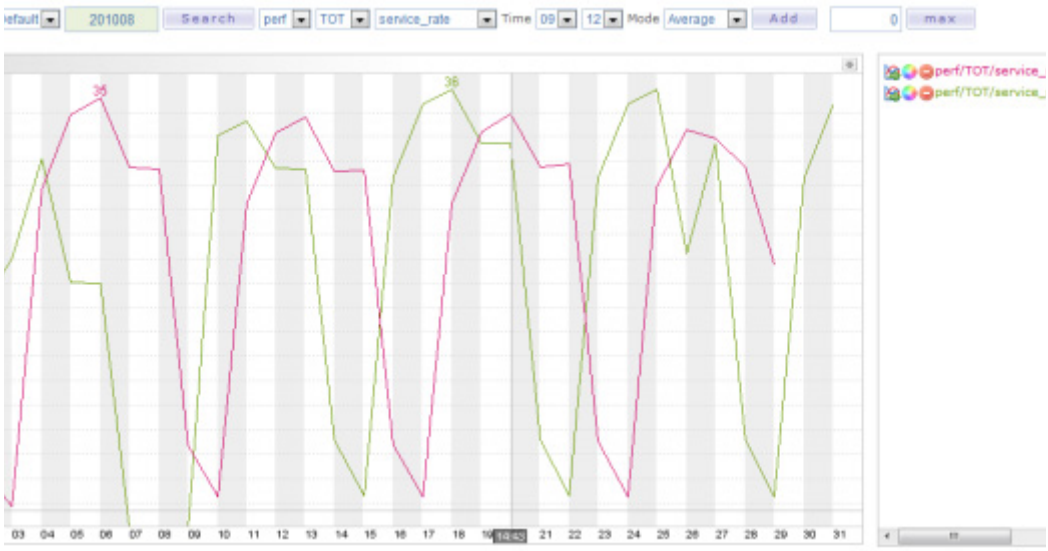
- IP : 필터링하고자 하는 클라이언트 IP
- IP Mask : 필터링 하고자 하는 클라이언트 IP 대역을 설정 설정 방법은 아래와 같다

```
192.168.0.255 : 192.168.0으로 시작하는 모든 IP
 192.168.255.255 : 192.168로 시작하는 모든 IP
 192.255.255.255 : 192로 시작하는 모든 IP
```

13.12.2. 월별 성능 추이 분석(Monthly PTA)

시스템의 성능을 월 단위로 분석한다. 한달동안 특정시간대에 어떤 값을 갖는지를 표현한다. X축에는 1부터 31까지 일자로 표현된다.

그림 13-78:월별 성능 추이 분석(Monthly PTA)



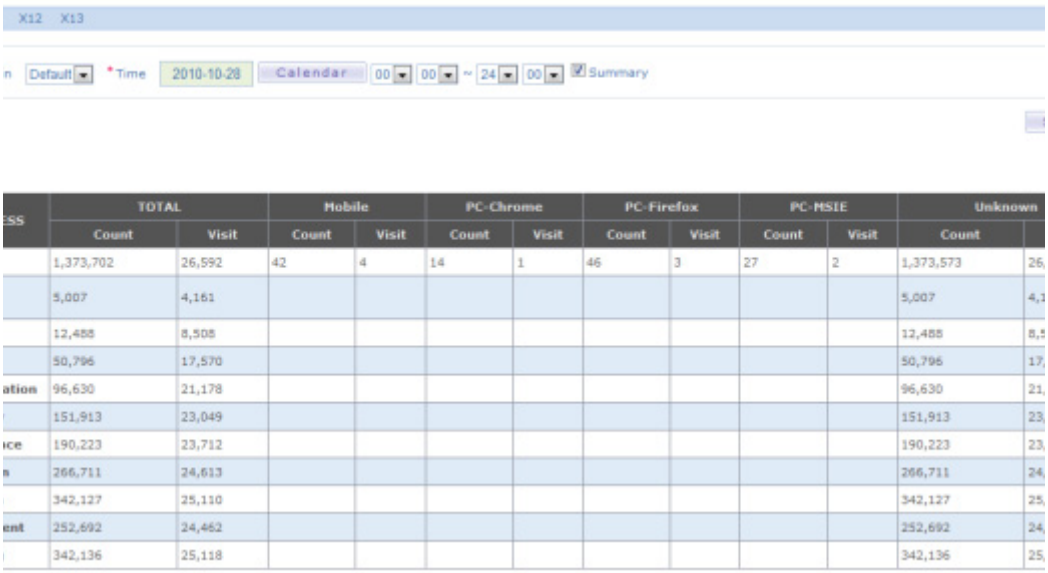
월을 입력하고 [Search]버튼을 클릭하면 오른쪽에 사용가능한 값에 대한 선택박스가 나타난다. perf/TOT/service_rate 와 같이 값을 차례로 선택한 후에 시간 범위를 선택한다.

예를 들어 초당 처리량에 대한 9시부터 12시 사이의 평균값에 대한 한달동안 추의를 보기 위해서는 9시부터 10시를 선택하고 Mode에 Average 선택하고 [Add] 를 클릭하면 화면에 그래프가 추가된다.

13.12.3.브라우저 접속 통계(Browsers)

사용자가 어떤 브라우저를 사용하여 접속하는지 혹은 모바일에서 접속하는지 아니면 PC에서 접속하는지에 대한 통계를 산출한다.

그림 13-79:브라우저 접속 통계(Browsers)



ESS	TOTAL		Mobile		PC-Chrome		PC-Firefox		PC-MSIE		Unknown	
	Count	Visit	Count	Visit	Count	Visit	Count	Visit	Count	Visit	Count	
	1,373,702	26,592	42	4	14	1	46	3	27	2	1,373,573	26
	5,007	4,161									5,007	4,1
	12,488	8,508									12,488	8,5
	50,796	17,570									50,796	17,
ation	96,630	21,178									96,630	21,
	151,913	23,049									151,913	23,
ice	190,223	23,712									190,223	23,
n	266,711	24,613									266,711	24,
	342,127	25,110									342,127	25,
ent	252,692	24,462									252,692	24,
	342,136	25,118									342,136	25,

날짜를 선택하고 시간을 선택하면 해당 시간대에서 수행된 서비스를 기반으로 업무별 호출건수와 방문 사용자의 수를 브라우저 별로 계산하여 출력한다.

Notice: 만약 [Summary] Check를 해제하고 수행하면 Http User-Agent 정보를 파싱하지 않고 통계를 생성한다

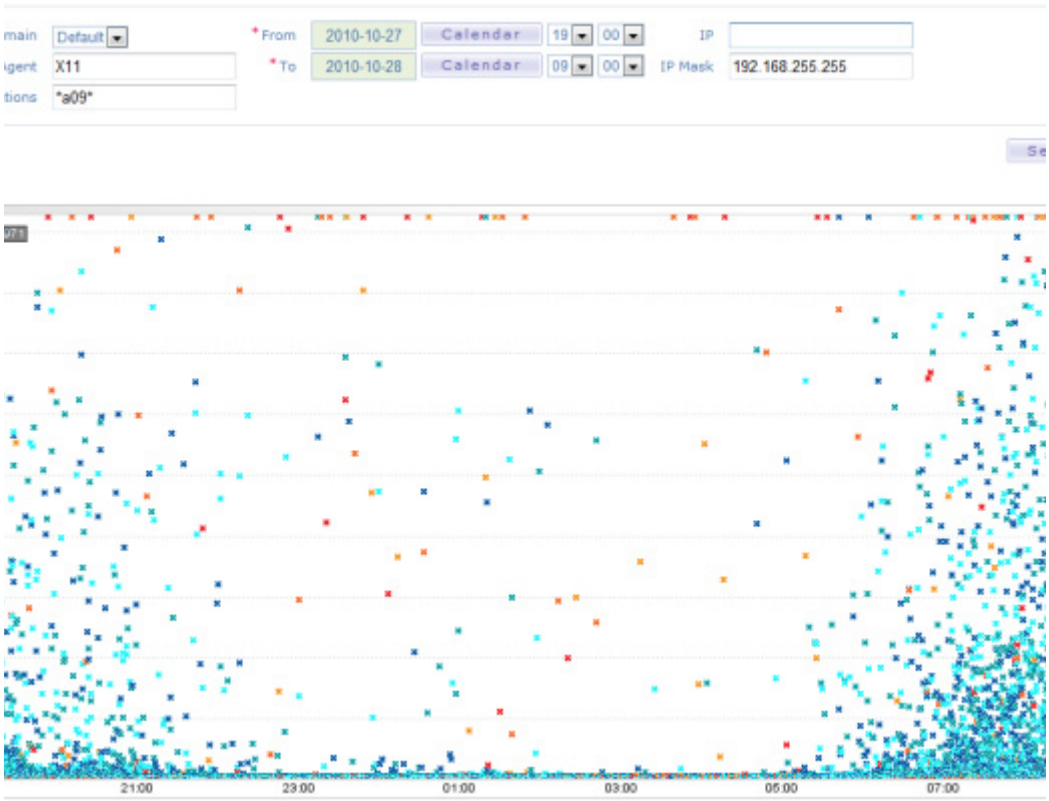
13.13.고급 통계

제니퍼4.5이후에 새롭게 추가된 통계 기능에 대해서 사용법을 설명한다.

13.13.1.기간 X-View

일정 기간동안의 XView를 조회하는 기능이다. 장기간에 걸쳐 일정조건의 서비스의 성능을 분석할때 사용한다.

그림 13-80:기간 X-View



Notice: 단 너무 많은 데이터를 조회하는 경우 메모리 에러가 발생할 수 있음으로 적절하게 조회 조건을 설정해야 한다.

13.13.2.조회조건 설정

- From/To: 검색을 원하는 시간을 설정한다.
- Agent : 에이전트를 지정한다.

- Applications : 필터링할 애플리케이션 서비스명을 설정한다. 설정 방법은 아래와 같다.

```
*AAA* : AAA를 포함한 서비스 명
   BBB* : BBB로 시작하는 서비스 명
   *CCC : CCC로 끝나는 서비스 명
  AA*DD : AA로 시작해서 DD로 끝나는 서비스 명
```

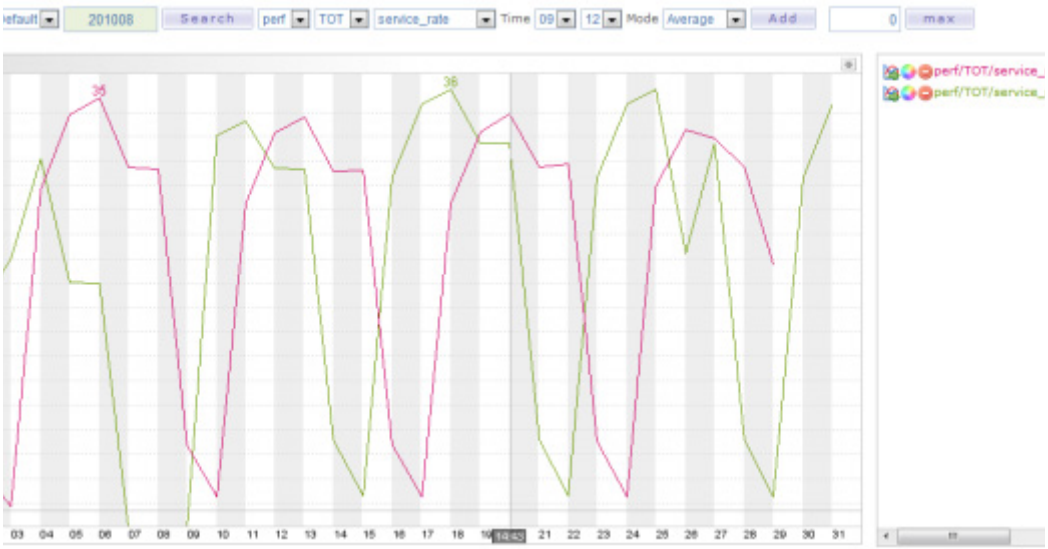
- IP : 필터링하고자 하는 클라이언트 IP
- IP Mask : 필터링 하고자 하는 클라이언트 IP 대역을 설정 설정 방법은 아래와 같다.

```
192.168.0.255 : 192.168.0으로 시작하는 모든 IP
 192.168.255.255 : 192.168로 시작하는 모든 IP
 192.255.255.255 : 192로 시작하는 모든 IP
```

13.13.3.월별 성능 추이 분석(Monthly PTA)

시스템의 성능을 월 단위로 분석한다. 한달동안 특정시간대에 어떤 값을 갖는지를 표현한다. X축에는 1부터 31까지 일자로 표현된다.

그림 13-81:월별 성능 추이 분석(Monthly PTA)



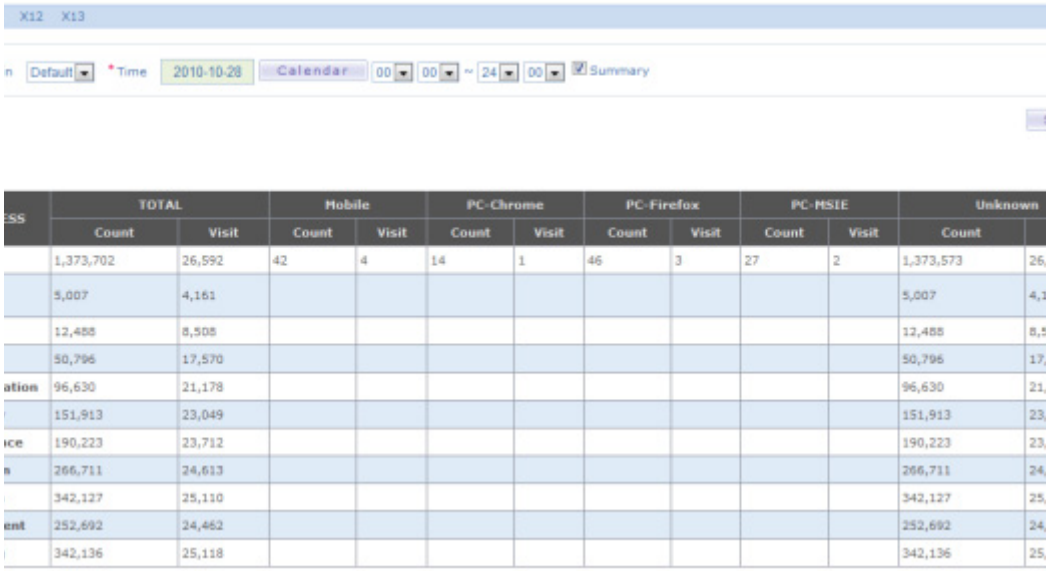
월을 입력하고 [Search]버튼을 클릭하면 오른쪽에 사용가능한 값에 대한 선택박스가 나타난다. perf/TOT/service_rate 와 같이 값을 차례로 선택한 후에 시간 범위를 선택한다.

예를 들어 초당 처리량에 대한 9시부터 12시 사이의 평균값에 대한 한달동안 추의를 보기 위해서는 9시부터 10시를 선택하고 Mode에 Average 선택하고 [Add] 를 클릭하면 화면에 그래프가 추가된다.

13.13.4.브라우저 접속 통계(Browsers)

사용자가 어떤 브라우저를 사용하여 접속하는지 혹은 모바일에서 접속하는지 아니면 PC에서 접속하는지에 대한 통계를 산출한다.

그림 13-82:브라우저 접속 통계(Browsers)



ESS	TOTAL		Mobile		PC-Chrome		PC-Firefox		PC-MSIE		Unknown	
	Count	Visit	Count	Visit	Count	Visit	Count	Visit	Count	Visit	Count	
	1,373,702	26,592	42	4	14	1	46	3	27	2	1,373,573	26
	5,007	4,161									5,007	4,1
	12,488	8,508									12,488	8,5
	50,796	17,570									50,796	17
ation	96,630	21,178									96,630	21
	151,913	23,049									151,913	23
ice	190,223	23,712									190,223	23
n	266,711	24,613									266,711	24
	342,127	25,110									342,127	25
ent	252,692	24,462									252,692	24
	342,136	25,118									342,136	25

날짜를 선택하고 시간을 선택하면 해당 시간대에서 수행된 서비스를 기반으로 업무별 호출건수와 방문 사용자의 수를 브라우저 별로 계산하여 출력한다.

Notice: 만약 [Summary] Check를 해제하고 수행하면 Http User-Agent 정보를 파싱하지 않고 통계를 생성한다

14

고급 모니터링

프로그래밍으로 제니퍼 에이전트를 확장하는 방법과 제니퍼 서버가 수집한 성능 데이터를 다른 애플리케이션과 연동하는 방법을 설명한다. 그리고 SOA(Service Oriented Architecture) 구현을 위한 솔루션을 모니터링하는 방법을 설명한다.

14.1. 제니퍼 에이전트 확장

제니퍼는 자바 애플리케이션 소스 코드의 수정없이 제니퍼 에이전트를 통해서 다양한 성능 데이터를 수집한다. 그런데 제니퍼 에이전트가 제공하는 API와 어뎁터로 모니터링 기능을 확장할 수 있다.

이를 위해서 간단한 프로그래밍이 필요하고 소스 코드를 컴파일하려면 JENNIFER_HOME/agent 디렉토리의 jennifer.jar 파일을 클래스 패스에 등록해야 한다.

14.1.1.ActiveTraceUtil

ActiveTraceUtil 클래스를 이용해 트랜잭션 데이터를 임의로 수정할 수 있다. ExtraAgentUtil 클래스와 함께 제니퍼 에이전트 확장 방법 중에서 애플리케이션 소스 코드의 수정을

필요로 한다. 따라서 제니퍼 에이전트를 제거하기 위해서는 ActiveTraceUtil 클래스를 사용하는 모든 소스 코드를 수정해야 한다. ActiveTraceUtil 클래스의 API는 다음과 같다.

```
package com.javaservice.jennifer.agent;

public class ActiveTraceUtil {

    public ActiveTraceUtil();
    public ActiveTraceUtil(ActiveObject activeObject);

    public String getGUID();
    public void setGUID(String guid);

    public long getTUID();

    public void setAppException(String message);
    public void setAppException(int exceptionCode, String message);

    public void setApplicationName(String name);
    public void addActiveServiceName(String name);

    public void setExternalTransactionName(String name);

    public static void addProfile(String message);

}
```

ActiveTraceUtil 객체는 트랜잭션을 처리하는 특정 자바 쓰레드에 배타적인 방식으로 사용되어야 한다. 즉, 전역 변수나 javax.servlet.Servlet 객체의 멤버 필드로 사용해서는 안된다. 메소드 안에서 로컬 변수로 이 객체를 생성해서 사용하는 것을 권장한다.

14.1.1.1. 애플리케이션 이름 수정

제니퍼 에이전트의 옵션을 통해서 설정하는 방법(애플리케이션 서비스 네이밍을 참조) 이외에 ActiveTraceUtil 클래스의 setApplicationName 메소드로 애플리케이션 이름을 설정할 수 있다..

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.setApplicationName("APP_NAME_001");
```

트랜잭션에서 앞의 코드가 수행되면 해당 트랜잭션의 애플리케이션 이름은 APP_NAME_001이 된다.

그리고 애플리케이션 이름은 변경하지 않고, **[실시간 모니터링 | 애플리케이션]** 메뉴의 액티브 서비스 탭에 나타나는 애플리케이션 이름에 임의의 내용을 추가하려면 ActiveTraceUtil 클래스의 addActiveServiceName 메소드를 사용한다.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.addActiveServiceName("ACTIVE_NAME_001");
```

애플리케이션 이름이 /index.jsp인 트랜잭션에서 앞의 코드가 수행되면 액티브 서비스 탭에는 다음과 같이 나타난다.

```
/index.jsp?ACTIVE_NAME_001
```

그림 14-1: 액티브 서비스 탭

ActiveTraceUtil 클래스의 addActiveServiceName 메소드를 한번 이상 호출할 수 있다.

```
activeTraceUtil.addActiveServiceName("ACTIVE_NAME_001");
...
activeTraceUtil.addActiveServiceName("ACTIVE_NAME_002");
```

애플리케이션 이름이 /index.jsp인 트랜잭션에서 앞의 코드가 수행되면 액티브 서비스 탭에는 다음과 같이 나타난다.

```
/index.jsp?ACTIVE_NAME_001+ACTIVE_NAME_002
```

14.1.1.2. 외부 트랜잭션 이름 수정

제니퍼 에이전트의 옵션을 통해서 설정하는 방법(외부 트랜젝션 네이밍을 참조) 이외에 ActiveTraceUtil 클래스의 setExternalTransactionName 메소드로 외부 트랜잭션 이름을 설정할 수 있다.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.setExternalTransactionName("TX_NAME_001");
```

트랜잭션에서 앞의 코드가 수행되면 외부 트랜잭션 이름은 TX_NAME_001이 된다.

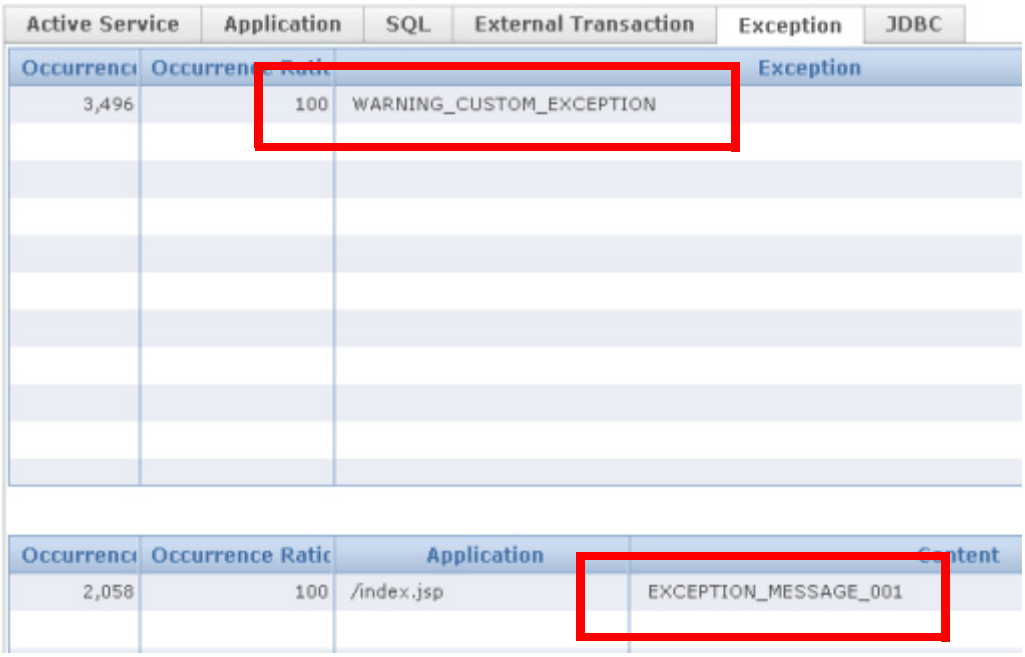
14.1.1.3. 임의의 예외를 발생시키기

ActiveTraceUtil 클래스의 setAppException 메소드를 통해서 트랜잭션 수행 중에 임의의 예외가 발생한 것으로 설정할 수 있다.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.setAppException("EXCEPTION_MESSAGE_001");
```

트랜잭션에서 앞의 코드가 수행되면 해당 트랜잭션에서 WARNING_CUSTOM_EXCEPTION 유형의 예외가 발생한 것으로 간주된다. setAppException 메소드의 파라미터는 예외 메시지가 된다.

그림 14-2: 예외 탭



예외에 대한 자세한 사항은 경보와 예외의 이해를 참조한다.

14.1.1.4. 프로파일에 메세지 추가

ActiveTraceUtil 클래스의 addProfile 메소드로 X-View 프로파일 데이터에 임의의 메시지를 추가할 수 있다. 이 메소드는 다른 메소드와는 다르게 ActiveTraceUtil 클래스의 정적 메소드이다.

```
ActiveTraceUtil.addProfile("PROFILE_MESSAGE_001");
```

트랜잭션 수행 중에 앞의 코드가 수행되면 X-View 프로파일 데이터에 PROFILE_MESSAGE_001 메시지가 나타난다.

그림 14-3: X-View 프로파일

이 메소드로 JSP 구간별 응답 시간 등을 손쉽게 확인할 수 있다.

```
<%
    ...
    ActiveTraceUtil.addProfile("JSP_POINT_001");
    new OneAction().execute();
    new TwoAction().execute();
    ActiveTraceUtil.addProfile("JSP_POINT_002");
    ...
%>
```

앞의 JSP가 수행되면 X-View 프로파일 데이터에서 JSP 구간별 분석을 할 수 있다.

그림 14-4: JSP 구간별 프로파일

X-View 프로파일에 대한 자세한 사항은 X-View와 프로파일링을 참조한다.

14.1.1.5. 트랜잭션 아이디 제어

ActiveTraceUtil 클래스의 getTUID 메소드로 트랜잭션의 UUID를 획득할 수 있다. 트랜잭션의 UUID는 제니퍼 에이전트가 자동으로 설정한다.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
long uuid = activeTraceUtil.getTUID();
```

이를 통해서 제니퍼가 수집하는 트랜잭션 데이터와 비지니스 이벤트 혹은 데이터와의 연관 관계를 설정할 수 있다.

ActiveTraceUtil 클래스의 getGUID와 setGUID 메소드로 글로벌 트랜잭션 연계 추적을 위한 GUID를 설정할 수 있다.

```
ActiveTraceUtil activeTraceUtil = new ActiveTraceUtil();
activeTraceUtil.setGUID("UNIQUE_GUID_001");
...
String guid = activeTraceUtil.getGUID();
```

UUID는 long 유형이고, GUID는 java.lang.String 유형이다. 글로벌 트랜잭션 연계 추적과 관련한 자세한 사항은 글로벌 트랜잭션 연계추적을 참조한다.

14.1.2.ExtraAgent 어뎁터

제니퍼 에이전트가 기본적으로 수집하지 않는 데이터를 모니터링해야 하는 경우 ExtraAgent 어뎁터로 수집할 수 있다.

ExtraAgent 어뎁터를 사용하려면 우선 com.javaservice.jennifer.agent.ExtraAgent 인터페이스를 구현한 클래스를 작성해야 한다. ExtraAgent 인터페이스는 다음과 같다.

```
package com.javaservice.jennifer.agent;

import com.javaservice.jennifer.protocol.RmPacket;

public interface ExtraAgent {

    public RmPacket process(String agent);

}
```

ExtraAgent 어뎁터의 process 메소드에서 임의의 데이터를 수집한다. 수집한 데이터로 RmPacket 객체를 만들어서 이를 반환하면 제니퍼 에이전트가 제니퍼 서버 혹은 REMON에 이를 REMON 데이터 형태로 전송한다. 따라서 제니퍼 서버는 ExtraAgent 어뎁터에서 전송받은 데이터를 REMON이 전송한 REMON 데이터와 동일하게 취급한다.

다음은 자바 쓰레드의 상태별 개수를 수집하는 예제이다. JMX(Java Management Extension)를 이용하기 때문에 자바 1.5 이상에서 사용할 수 있다.

```
package example;

import java.lang.management.ManagementFactory;
import java.lang.management.ThreadMXBean;

import com.javaservice.jennifer.agent.ExtraAgent;
import com.javaservice.jennifer.protocol.RmPacket;
import com.javaservice.jennifer.type.INT;

public class ThreadCountExtraAgent implements ExtraAgent {

    public RmPacket process(String agent) {
        ThreadMXBean threads = ManagementFactory.getThreadMXBean();
        RmPacket result = new RmPacket("THREAD_COUNT", "EA1");
        result.addField("TOTAL", new INT(threads.getThreadCount()));
        result.addField("DAEMON", new
INT(threads.getDaemonThreadCount()));
        return result;
    }

}
```

THREAD_COUNT는 REMON 데이터의 스크립트 아이디이고, EA1는 REMON 데이터의 에이전트 아이디이다. 스크립트 아이디와 에이전트 아이디에 대한 자세한 사항은 REMON 스크립트를 참조하고, RmPacket 클래스를 사용하는 자세한 방법은 RmPacket 클래스의 사용을 참조한다.

소스 코드를 컴파일한 후에 임의의 JAR 파일로 패키지한다.

다음은 ExtraAgent 어뎁터 사용에 필요한 설정이다. 제니퍼 에이전트의 extra_enable 옵션을 true로 설정한다.

```
extra_enable = true
```

제니퍼 에이전트의 extra_agent_class 옵션으로 ExtraAgent 인터페이스를 구현한 클래스를 설정한다. 2개 이상의 ExtraAgent 어뎁터를 사용하는 경우에는 세미 콜론[;]을 구분자로 구분한다.

```
extra_agent_class = example.ThreadCountExtraAgent
```

ExtraAgent 인터페이스를 구현한 클래스를 담고 있는 JAR 파일을 제니퍼 에이전트의 extra_agent_classpath 옵션으로 설정한다. 2개 이상의 JAR 파일을 사용하는 경우에는 세미 콜론[;]을 구분자로 구분한다.

```
extra_agent_classpath = /jennifer/extension.jar
```

ExtraAgent 어뎁터의 수행 주기를 제니퍼 에이전트의 extra_data_interval 옵션으로 설정한다. 단위는 밀리 세컨드이다.

```
extra_data_interval = 5000
```

ExtraAgent 어뎁터가 수집하는 데이터를 제니퍼 서버가 아닌 REMON에 전송하기 위해서는 추가적으로 제니퍼 에이전트의 extra_data_send_target 옵션으로 REMON의 IP 주소와 포트 번호를 설정해야 한다.

```
extra_data_send_target=127.0.0.1:7701
```

Warning: 제니퍼 서버로 전송하는 경우와 REMON으로 전송하는 하는 경우의 프로토콜이 다르기 때문에 제니퍼 서버로 전송하는 경우에는 이 옵션을 설정해서는 안된다.

REMON으로 전송하면 필터를 이용해서 데이터를 다양한 방법으로 가공할 수 있다. 자세한 사항은 데이터 제어를 참조한다.

이 옵션들은 자바 애플리케이션을 정지하지 않고 수정할 수 있다. 단, ExtraAgent 구현 클래스를 수정한 경우에는 다음과 같은 방법으로 이를 반영한다.

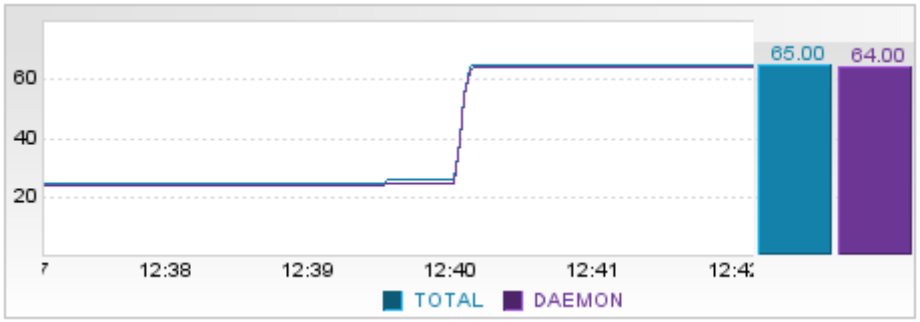
- 제니퍼 에이전트의 extra_enable 옵션을 false로 설정한다.
- 수정된 클래스로 다시 JAR 파일을 패키지한 후 기존 JAR 파일을 대체한다.
- 제니퍼 에이전트의 extra_enable 옵션을 true로 설정한다.

단, 제니퍼 에이전트의 extra_agent_hotswap 옵션이 true로 설정되어 있어야 한다. 기본값은 true이다.

```
extra_agent_hotswap = true
```

제니퍼 서버는 ExtraAgent 어뎁터가 전송한 데이터를 REMON 데이터와 동일하게 처리한다. 따라서 ExtraAgent 어뎁터로 수집한 데이터를 모니터링하는 방법은 사용자 정의 차트를 참조한다. 다음은 자바 쓰레드 개수를 나타내는 라인 차트이다.

그림 14-5: 자바 쓰레드 개수 모니터링



14.1.3.CustomTrace 어뎁터

CustomTrace 어뎁터로 트랜잭션에서 호출되는 메소드의 전후로 임의의 작업을 수행할 수 있다.

CustomTrace 어뎁터를 사용하려면 우선 com.javaservice.lwst.CustomTraceAdapter 인터페이스를 구현한 클래스를 작성해야 한다. CustomTraceAdapter 인터페이스는 다음과 같다.

```
package com.javaservice.lwst;

public interface CustomTraceAdapter {

    public CustomStat start(String className, String methodName,
Object traceParam);

    public void end(CustomStat stat, Throwable e);

}
```

트랜잭션에서 특정 클래스의 특정 메소드가 호출되면, 해당 메소드가 수행되기 전에 CustomTrace 어뎁터의 start 메소드가 호출된다. start 메소드의 첫번째 파라미터는 호출되는 클래스 이름이고, 두번째 파라미터는 호출되는 메소드 이름이다. 그리고 세번째 파라미터는 int와 long 등의 기본 유형이 아닌 객체 유형의 첫번째 메소드 파라미터이다.

그리고 해당 메소드의 수행이 완료되면 CustomTrace 어뎁터의 end 메소드가 호출된다. 그런데 start 메소드에서 null을 반환한 경우에는 end 메소드가 호출되지 않는다. end 메소드의 첫번째 파라미터는 start 메소드에서 반환한 CustomStat 객체이고, 두번째 파라미터는 해당 메소드에서 발생한 예외 객체이다. 예외가 발생하지 않았으면 null이 된다.

start 메소드에서 반환하는 com.javaservice.lwst.CustomStat 클래스는 다음과 같다. start 메소드에서 null을 반환하면 end 메소드가 호출되지 않기 때문에 임의의 처리가 필요하지 않는 클래스의 메소드 호출에 대해서는 null을 반환하는 것을 권장한다.

```
package com.javaservice.lwst;

public class CustomStat {

    public String className;

    public String methodName;

    public long startTime;

    public Object traceValue;

}
```

다음은 CustomTrace 어뎁터로 예외를 처리하는 예제이다. 일반적으로 트랜잭션에서 발생한 예외가 트랜잭션이 종료하기 전에 처리되면 해당 트랜잭션은 성공한 것으로 간주된다.

```
public void execute() {
    try {
        new OrderManager().order("0-322-59443-1", 3);
    catch (IllegalStateException ignore) {
    }
}
```

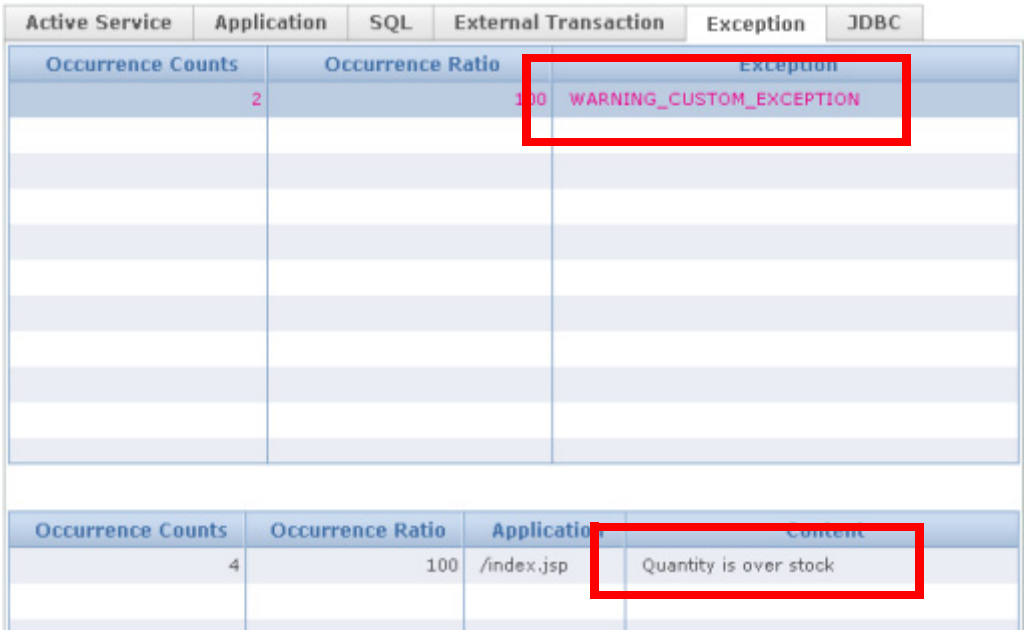
OrderManager 클래스의 order 메소드는 재고가 부족하면 java.lang.IllegalStateException 예외를 던진다. 그런데 트랜잭션에서 앞의 코드가 수행되어도 catch 문에서 이 예외를 처리했기 때문에 트랜잭션은 성공한 것으로 간주된다. 그런데 CustomTrace 어뎁터를 이용하면 기존 소스 코드의 수정없이 이 트랜잭션을 실패한 것으로 처리할 수 있다.

ExceptionTraceAdapter 클래스는 CustomTraceAdapter 인터페이스를 구현한 클래스이다. end 메소드로 IllegalStateException 예외가 전달되면 ActiveTraceUtil 클래스의 setAppException 메소드로 트랜잭션에서 예외가 발생한 것으로 처리한다.

```
package example;

import com.javaservice.jennifer.agent.ActiveTraceUtil;
import com.javaservice.lwst.CustomStat;
import com.javaservice.lwst.CustomTraceAdapter;

public class ExceptionTraceAdapter implements CustomTraceAdapter {

    public CustomStat start(String className, String methodName,
Object traceValue) {
        CustomStat result = new CustomStat();
        result.className = className;
        result.methodName = methodName;
        result.traceValue = traceValue;
        result.startTime = System.currentTimeMillis();
        return result;
    }

    public void end(CustomStat stat, Throwable e) {
        if (e != null && e instanceof IllegalStateException) {
            new ActiveTraceUtil().setAppException(e.getMessage());
        }
    }

}
```

다음은 **[실시간 모니터링 | 애플리케이션]** 메뉴의 예외 탭에서 이를 확인한 그림이다.

그림 14-6: CustomTrace 어뎁터를 이용한 예외 처리



다음은 CustomTrace 어뎁터 사용에 필요한 설정이다. 우선 제니퍼 에이전트의 custom_trace_adapter_class_name 옵션으로 CustomTraceAdapter 인터페이스를 구현한 클래스를 설정한다. 하나의 CustomTrace 어뎁터만을 사용할 수 있다.

```
custom_trace_adapter_class_name = example.ExceptionTraceAdapter
```

CustomTraceAdapter 인터페이스를 구현한 클래스를 담고 있는 JAR 파일을 제니퍼 에이전트의 custom_trace_adapter_class_path 옵션으로 설정한다.

```
custom_trace_adapter_class_path = /jennifer/extension.jar
```

CustomTraceAdapter 구현 클래스를 수정한 경우에는 다음과 같은 방법으로 이를 반영한다.

- 수정된 클래스로 다시 JAR 파일을 패키지한 후 기존 JAR 파일을 대체한다.
- **[구성 관리 | 구성 설정]** 메뉴에서 해당 제니퍼 에이전트 설정을 변경한다. 내용 자체를 변경할 필요는 없고 **[수정]** 버튼을 클릭만 하면 된다.

단, 제니퍼 에이전트의 custom_trace_hotswap 옵션이 true로 설정되어 있어야 한다. 기본값은 true이다.

```
custom_trace_hotswap = true
```

모든 메소드의 호출에 CustomTrace 어뎁터가 적용되는 것은 아니다. 제니퍼 에이전트의 custom_trace로 시작하는 옵션을 통해서 CustomTrace 어뎁터가 적용될 메소드를 설정한다.

Warning: CustomTraceAdapter 인터페이스를 구현한 클래스에도 CustomTrace 어뎁터를 적용할 수 있다. 이 경우에 무한 반복 호출이 일어날 수 있다. 따라서 CustomTraceAdapter 인터페이스를 구현한 클래스에는 CustomTrace 어뎁터를 적용해서는 안된다.

예를 들어, 앞의 business.OrderManager 클래스에 CustomTrace 어뎁터를 적용하려면 제니퍼 에이전트의 custom_trace_class 옵션으로 설정한다.

```
custom_trace_class = business.OrderManager
```

그리고 business.ProductManager 클래스에도 CustomTrace 어뎁터를 적용하려면 [;]을 구분자로 custom_trace_class 옵션에 설정한다. custom_trace_class 옵션뿐만 아니라 CustomTrace 어뎁터 적용과 관련한 모든 옵션에 2개 이상을 설정하려면 [;]을 구분자로 사용한다.

```
custom_trace_class = business.OrderManager;business.ProductManager
```

CustomTrace 어뎁터는 특정 메소드에 적용된다. 즉, 앞의 예와 같이 설정하면 business.OrderManager 클래스의 모든 메소드에 CustomTrace 어뎁터가 적용된다. 따라서 특정 메소드만에만 CustomTrace 어뎁터를 적용하려면 제니퍼 에이전트의 custom_trace_target_method 옵션으로 설정한다.

```
custom_trace_target_method = order
```

그런데 CustomTrace 어뎁터를 적용할 클래스들 중에서 이름이 동일한 메소드가 있고, 이 중 특정 클래스의 특정 메소드에만 CustomTrace 어뎁터를 적용하려면 다음과 같이 구체적으로 설정한다.

```
custom_trace_target_method = business.OrderManager.order
```

반대로 특정 메소드에만 CustomTrace 어뎁터를 적용하지 않으려면 제니퍼 에이전트의 custom_trace_ignore_method 옵션으로 설정한다.

```
custom_trace_ignore_method = business.OrderManager.checkStock
```

메소드의 접근자를 통해서도 CustomTrace 어뎁터를 적용할 수 있다. 예를 들어, public 접근자와 접근자가 없는 메소드에만 CustomTrace 어뎁터를 적용하려면 다음과 같이 설정한다.

```
custom_trace_access_method = public;none
```

custom_trace_access_method 옵션에 설정이 가능한 값은 다음과 같다.

- public - public 접근자
- protected - protected 접근자
- private - private 접근자
- none - 접근자가 없는 경우

그리고 특정 클래스를 상속한 모든 클래스에 CustomTrace 어뎁터를 적용하려면 제니퍼 에이전트의 custom_trace_super 옵션을 사용한다. 예를 들어, business.ejb.BaseSessionBean을 상속한 모든 클래스에 CustomTrace 어뎁터를 적용하려면 다음과 같이 설정한다.

```
custom_trace_super = business.ejb.BaseSessionBean
```

그러나 이 경우에 직접적으로 상속받은 클래스에만 CustomTrace 어뎁터를 적용한다. 예를 들어, A 클래스가 business.ejb.BaseSessionBean 클래스를 상속하고 B 클래스가 A 클래스를 상속했다면, A 클래스에는 CustomTrace 어뎁터를 적용하지만 B 클래스에는 CustomTrace 어뎁터를 적용하지 않는다.

특정 인터페이스를 구현한 모든 클래스에 CustomTrace 어뎁터를 적용하려면 제니퍼 에이전트의 custom_trace_interface 옵션을 사용한다. 예를 들어, business.IManager 인터페이스를 구현한 모든 클래스에 CustomTrace 어뎁터를 적용하려면 다음과 같이 설정한다.

```
custom_trace_interface = business.IManager
```

그러나 이 경우에 직접적으로 구현한 클래스에만 CustomTrace 어뎁터를 적용된다. 예를 들어, A 클래스가 business.IManager 인터페이스를 구현하고 B 클래스가 A 클래스를 상속했다면, A 클래스에는 CustomTrace 어뎁터를 적용하지만 B 클래스에는 CustomTrace 어뎁터를 적용하지 않는다.

클래스의 이름을 이용해서도 CustomTrace 어뎁터를 적용할 수 있다. 이 경우에는 제니퍼 에이전트의 custom_trace_prefix, custom_trace_postfix, custom_trace_ignore_prefix 옵션을 사용한다. 예를 들어, 이름이 business로 시작하는 모든 클래스에 CustomTrace 어뎁터를 적용하려면 다음과 같이 한다.

```
custom_trace_prefix = business
```

이름이 Manager로 끝나는 모든 클래스에 CustomTrace 어뎁터를 적용하려면 다음과 같이 한다.

```
custom_trace_postfix = Manager
```

특정 이름으로 시작하는 클래스에는 CustomTrace 어뎁터를 적용하지 않으려면 제니퍼 에이전트의 custom_trace_ignore_prefix 옵션을 사용한다. 이 옵션은 다른 모든 옵션에 우선한다.

```
custom_trace_ignore_prefix =
```

제니퍼 에이전트의 custom_trace로 시작하는 옵션을 통해서 동일한 내용을 다양한 방법으로 설정할 수 있다. CustomTrace 어뎁터를 적용하는데 있어서 custom_trace_target_method 옵션을 통해서 실제로 필요한 메소드에만 CustomTrace 어뎁터를 적용하도록 한다.

예를 들어, pkg.ClassA와 pkg.ClassB 클래스가 있다. 여기서 ClassA 클래스의 run 메소드와 ClassB 클래스의 process 메소드에 CustomTrace 어뎁터를 적용한다고 가정한다. 그런데 ClassB 클래스에 run 메소드가 존재하고 이 메소드에는 CustomTrace 어뎁터를 적용하지 않는다면 다음과 같이 설정한다.

```
custom_trace_class = pkg.ClassA;pkg.ClassB
custom_trace_target_method = run;process
custom_trace_ignore_method = pkg.ClassB.run
```

또는 다음과 같이 설정할 수도 있다.

```
custom_trace_class = pkg.ClassA;pkg.ClassB
custom_trace_target_method = pkg.ClassA.run;pkg.ClassB.process
```

14.1.4.제니퍼 에이전트에서 서버로 경보 전송

ExtraAgentUtil 클래스로 임의의 경보를 발령할 수 있다. ActiveTraceUtil 클래스와 함께 제니퍼 에이전트 확장 방법 중에서 애플리케이션 소스 코드의 수정을 필요로 한다. 따라서 제니퍼 에이전트를 제거하면 ExtraAgentUtil 클래스를 사용하는 모든 소스 코드를 수정해야 한다. ExtraAgentUtil 클래스의 전체 이름은 다음과 같다.

```
com.javaservice.jennifer.agent.ExtraAgentUtil
```

경보 발령을 위해서 ExtraAgentUtil 클래스가 제공하는 메소드는 다음과 같다.

- sendAlertFATAL(String) - USER_DEFINED_FATAL 경보 발령
- sendAlertERROR(String) - USER_DEFINED_ERROR 경보 발령
- sendAlertWARNING(String) - USER_DEFINED_WARNING 경보 발령
- sendAlertMESSAGE(String) - USER_DEFINED_MESSAGE 경보 발령

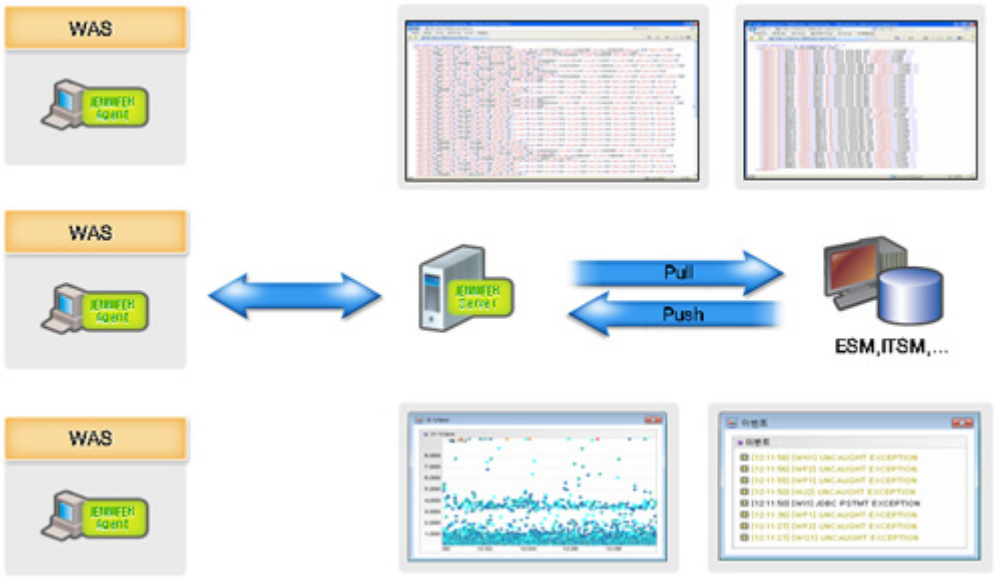
- sendAlert(String) - sendAlertWARNING 메소드와 동일
- sendAlert(int, String) - 첫번째 파라미터로 지정한 임의의 경보를 발령한다. 첫번째 파라미터에 com.javaservice.jennifer.util.Def 클래스에 존재하는 경보 유형과 동일한 이름의 상수 필드를 사용한다.

경보에 대한 자세한 사항은 경보와 예외 유형을 참조한다.

14.2. 성능 데이터 연동

제니퍼 서버가 수집한 성능 데이터를 다른 애플리케이션과 공유하는 방법을 설명한다. 이는 풀(Pull) 방법과 푸시(Push) 방법으로 나누어진다.

그림 14-1: 성능 데이터 연동 개념도



14.2.1.풀 방식의 성능 데이터 연동

외부 애플리케이션이 제니퍼 서버로 HTTP 요청을 하고, 이 요청을 받은 제니퍼 서버가 XML 형식의 문서를 반환하는 것이 풀 방식의 성능 데이터 연동이다.

14.2.1.1. 일반 성능 데이터 조회

다음 요청을 통해서 요청 시점의 일반 성능 데이터를 XML 파일로 확인할 수 있다.

```
http://jennifer_server_ip:7900/get_perf_agent.jsp
```

XML 파일의 기본 골격은 다음과 같다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<perf version="4.0.1.1" jennifer="SYS1"
      time="2008-11-10 14:16:45.716">


</perf>
```

최상위 태그는 perf 태그이고, perf 태그의 속성은 다음과 같다.

- version - 제니퍼 서버 버전
- jennifer - 제니퍼 서버 도메인 아이디
- time - XML 파일 생성 시간

그리고 perf 태그는 제니퍼 에이전트를 의미하는 instance 하위 태그로 구성된다.

```
<instance agent="TOT"
    ac0="1" ac1="1" ac2="0" ac3="5"
    act_serv="7"
    act_user="7"
    tps="25.633333"
    res_time="0.16205852"
    con_user="271.74252"
    error_rate="0.2"
    reject_rate="0.0"
    hit_hour="61679" visit_hour="4208"
    proc_cpu="0.0" proc_mem="0.0"
    hit_day="600767" visit_day="28807"
    alert_fatal="7" alert_error="3844" alert_warn="9874" />

<instance agent="W11" ...
```

다음은 각 속성에 대한 설명이다.

표 14-39: instance 태그 속성

필드명	설명
ac0 ~ ac3	경과 시간 구간별 액티브 서비스 개수를 의미한다.
act_serv	전체 액티브 서비스 개수로 각 구간별 액티브 서비스 개수의 합을 의미한다.
act_user	액티브 사용자 수
tps	서비스 처리율
res_time	평균 응답 시간
con_user	동시단말 사용자 수
error_rate	예외 발생률
reject_rate	PLC에 의한 서비스 거부율
hit_hour	현재 시간대 호출 건수
visit_hour	현재 시간대 방문자 수
proc_cpu	시스템 CPU 사용률
proc_mem	시스템 메모리 사용량
hit_day	금일 호출 건수

표 14-39: instance 태그 속성

필드명	설명
visit_day	금일 방문자 수
alert_fatal	금일 심각 경보 건수
alert_error	금일 에러 경보 건수
alert_warn	금일 경고 경보 건수

14.2.1.2. 비즈니스 그룹 성능 데이터 조회

다음 요청을 통해서 요청 시점의 비즈니스 그룹별 성능 데이터를 XML 파일로 확인할 수 있다.

```
http://jennifer_server_ip:7900/get_perf_biz.jsp
```

XML 파일의 기본 골격은 다음과 같다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<perf version="4.0.1.1" jennifer="SYS1"
      time="2008-11-10 14:16:45.716">

</perf>
```

최상위 태그는 perf 태그이고, perf 태그의 속성은 다음과 같다.

- version - 제니퍼 서버 버전
- jennifer - 제니퍼 서버 도메인 아이디
- time - XML 파일 생성 시간

그리고 perf 태그는 비즈니스 그룹을 의미하는 biz 하위 태그로 구성된다.

```
<biz id="BIZ/A" div="DIV" name="G01"
     res_time="0.11274193548387097" tps="2.066666666666667"
     accu_error="0" accu_hit="62" />

<biz id="BIZ/B" div="DIV" name="G02"
     res_time="3.1610000000000005" tps="0.43333333333333335"
     accu_error="0" accu_hit="13" />
```

다음은 각 속성에 대한 설명이다.

표 14-40: biz 태그 속성

필드명	설명
id	고유한 아이디
div	비즈니스 그룹 구분명
name	비즈니스 그룹 이름
res_time	비즈니스 그룹의 평균 응답 시간
tps	비즈니스 그룹의 서비스 처리율
accu_error	비즈니스 그룹의 누적 예외 건수
accu_hit	비즈니스 그룹의 누적 호출 건수

비즈니스 그룹을 통해서 특정 애플리케이션 그룹에 대한 서비스 처리율, 평균 응답 시간 등의 데이터를 수집할 수 있다. 예를 들어, 주문이나 결제와 같은 중요한 업무에 대한 성능 데이터를 별도로 수집할 수 있다.

[구성 관리 | 비즈니스 그룹] 메뉴에서 비즈니스 그룹을 설정한다. 비즈니스 그룹을 추가하는 방법은 다음과 같다.

- 비즈니스 그룹 목록 하단에 있는 **[추가]** 버튼을 클릭하면 오른쪽에 비즈니스 그룹 입력 폼이 나타난다.
- 비즈니스 그룹 입력 폼에 내용을 입력한 후에, 하단에 있는 **[저장]** 버튼을 클릭한다.

Notice: 비즈니스 그룹에 대한 수정 사항을 반영하려면 비즈니스 그룹 목록 하단에 있는 **[적용]** 버튼을 클릭한다.

비즈니스 그룹과 관련한 정보는 다음과 같다.

표 14-41: 비즈니스 그룹 정보

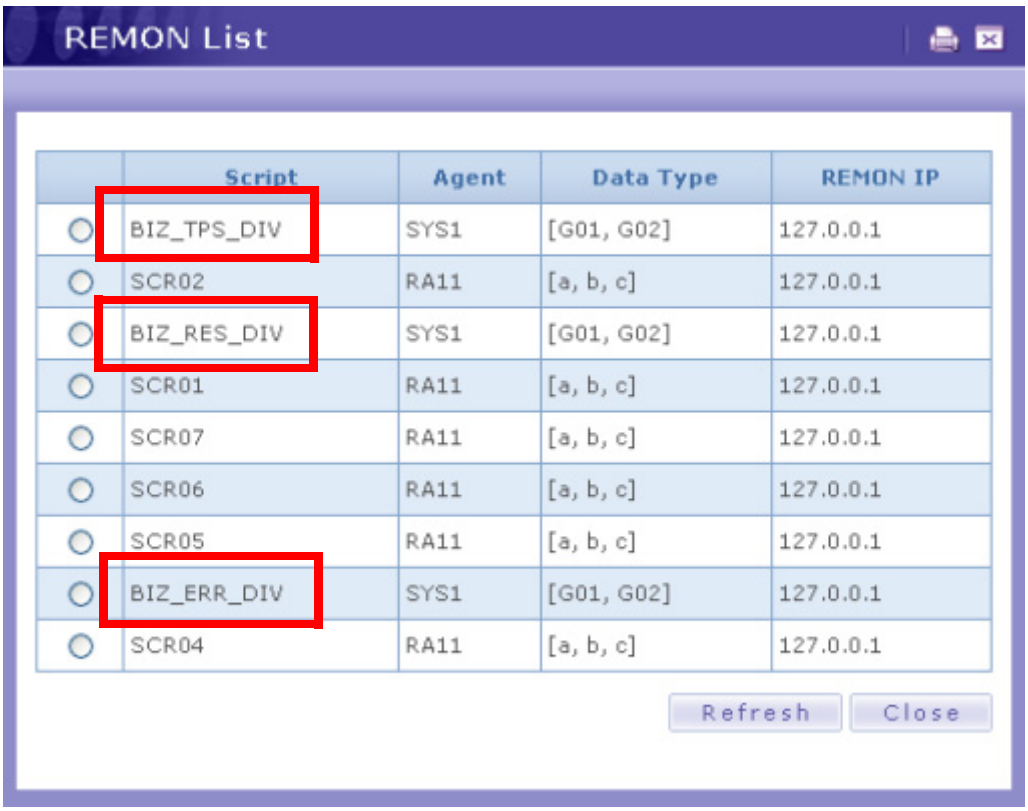
항목	설명
아이디	고유한 아이디로, 임의의 값을 입력한다. 경우에 따라서는 [/]를 구분자로 해서 트리 형태의 아이디 체계를 사용할 수도 있다.
구분명	REMON 데이터의 접미사가 된다. 구분명이 동일한 비즈니스 그룹은 동일한 REMON 데이터가 된다.
이름	비즈니스 그룹의 이름으로, REMON 데이터의 필드 이름이 된다.
애플리케이션	해당 비즈니스 그룹에 속하는 애플리케이션 이름을 줄바꿈을 구분자로 입력한다. 와일드 카드[*]를 사용할 수 있다.

표 14-41: 비즈니스 그룹 정보

항목	설명
외부 트랜잭션	해당 비즈니스 그룹에 속하는 외부 트랜잭션 이름을 줄바꿈을 구분자로 입력한다. 와일드 카드[*]를 사용할 수 있다. 외부 트랜잭션은 REMON 데이터로 수집되지 않는다. 단, **[통계 분석 | 보고서 | 일일 보고서]** 메뉴에서 해당 비즈니스 그룹에 대한 외부 트랜잭션 처리 현황을 확인할 수 있다.
도메인	제니퍼 서버를 선택한다.
에이전트	특정 제니퍼 에이전트에 대해서만 비즈니스 그룹 데이터를 도출하려면 콤마[,]를 구분자로 제니퍼 에이전트 아이디를 입력한다.
상태	비즈니스 그룹의 사용 여부를 나타낸다.

비즈니스 그룹을 설정하면 기본적으로 3개의 REMON 데이터를 수집한다.

그림 14-2: 비즈니스 그룹 REMON 데이터 목록



다음은 REMON 데이터에 대한 설명이다.

BIZ_TPS_[구분명] - 비즈니스 그룹에 대한 서비스 처리율을 의미한다.

BIZ_RES_[구분명] - 비즈니스 그룹에 대한 평균 응답 시간을 의미한다.

BIZ_ERR_[구분명] - 비즈니스 그룹에 대한 3초 동안 발생한 예외 건수를 의미한다.

구분명이 같은 비즈니스 그룹은 동일한 REMON 데이터가 되고, 비즈니스 그룹의 이름은 REMON 데이터의 필드가 된다.

14.2.2.푸시 방식의 성능 데이터 연동

XVLog나 SMS 어뎁터로 제니퍼 서버가 수집한 성능 데이터를 다른 애플리케이션으로 전송하는 것이 푸시 방식의 성능 데이터 연동이다.

14.2.2.1. XVLog를 통한 X-View 트랜잭션 데이터 처리

XVLog 어뎁터로 실시간으로 수집한 X-View 트랜잭션 데이터에 대해 임의의 작업을 수행할 수 있다.

Notice: XVLog 어뎁터를 컴파일하려면 JENNIFER_HOME/server/lib/jenniferserver.jar 파일을 클래스 패스에 등록해야 한다.

XVLog 어뎁터를 사용하려면 우선 com.javaservice.server.XVLog 인터페이스를 구현한 클래스를 작성해야 한다. XVLog 인터페이스는 다음과 같다.

```
package com.javaservice.jennifer.server;

import com.javaservice.jennifer.util.CircularObjectQueue;

public interface XVLog {

    public void execute(CircularObjectQueue queue);

}
```

반복적으로 XVLog 인터페이스를 구현한 클래스의 execute 메소드가 호출된다. execute 메소드의 파라미터는 CircularObjectQueue 객체인데, 이 객체에는 X-View 트랜잭션 데이터를 나타내는 TxPerfPacket 객체가 담겨 있다.

CircularObjectQueue 객체에 담겨 있는 TxPerfPacket 객체는 다음과 같은 구조를 갖는다. public 접근자인 멤버 필드로 필요한 데이터를 확인할 수 있다.

```
package com.javaservice.jennifer.protocol;

public class TxPerfPacket {

    public long store_time;
    public long end_time;
    public int elapsed;
    public int cpu_time;
    public int sql_time;
    public int fetch_time;
    public int etx_time;
    public int service_name_hash;
    public long tx_uuid;
    public int client_ip;
    public long wmonid;
    public byte err_type;
    public String agent;

    //using for global transaction trace
    public String guid;

    public int agent_profile_key;
    public String user_id;
    public byte turnaround;
}
```

일반적으로 사용되는 멤버 필드는 store_time 필드부터 agent 필드까지이고, 나머지는 특별한 상황을 위해 준비된 멤버 필드이다.

- store_time - long 유형으로, 제니퍼 서버가 X-View 트랜잭션 데이터를 저장한 시간을 의미한다. 제니퍼 서버가 설치된 하드웨어 시간을 기준으로 한다.
- end_time - long 유형으로, 트랜잭션의 종료 시간을 의미한다. 제니퍼 에이전트가 설치된 하드웨어 시간을 기준으로 한다.
- elapsed - int 유형으로, 트랜잭션의 응답 시간을 의미한다. 단위는 밀리 세컨드이다.
- cpu_time - int 유형으로, 트랜잭션이 사용한 CPU 시간을 의미한다. 단위는 밀리 세컨드이다.

- sql_time - int 유형으로, 트랜잭션에서 수행된 SQL 응답 시간을 의미한다. 단위는 밀리 세컨드이다.
- fetch_time - int 유형으로, 트랜잭션에서 수행된 SQL Fetch 시간을 의미한다. 단위는 밀리 세컨드이다.
- etx_time - int 유형으로, 트랜잭션에서 수행된 외부 트랜잭션 응답 시간을 의미한다. 단위는 밀리 세컨드이다.
- service_name_hash - int 유형으로, 애플리케이션 이름 해시 값을 의미한다.
- tx_uuid - long 유형으로, 트랜잭션 UUID를 의미한다.
- client_ip - int 유형으로, 클라이언트 IP 주소를 의미한다.
- wmonid - long 유형으로, 방문자 수와 동시단말 사용자 수를 계산하기 위한 wmonid 쿠키 값을 의미한다.
- err_type - byte 유형으로, 트랜잭션에서 발생한 예외 유형을 의미한다.
- agent - java.lang.String 유형으로, 제니퍼 에이전트 아이디를 의미한다.
- guid - java.lang.String 유형으로, 글로벌 트랜잭션 연계 추적으로 위한 GUID를 의미한다.

다음은 XVLog 어뎁터로 X-View 트랜잭션 데이터를 콘솔에 기록하는 예제이다.

```
package example;

import java.text.DateFormat;
import java.text.SimpleDateFormat;

import com.javaservice.jennifer.protocol.TxPerfPacket;
import com.javaservice.jennifer.server.XVLog;
import com.javaservice.jennifer.server.db.HashedString_DB;
import com.javaservice.jennifer.util.CircularObjectQueue;
import com.javaservice.jennifer.util.IpUtil;

public class ConsoleXVLog implements XVLog {

    private static DateFormat df =
                        new SimpleDateFormat("yyyyMMddHHmmssS");

    public void execute(CircularObjectQueue queue) {
        while (queue.size() > 0) {
            TxPerfPacket packet = (TxPerfPacket) queue.dequeue();
            String time =
                   df.format(new java.util.Date(packet.end_time));
            String appName =
                   HashedString_DB.getAPPL(packet.service_name_hash);
            System.out.println("Time=" + time + ", UUID=" +
packet.tx_uuid + ", Agent=" + packet.agent + ", App="
+ appName + ", Client IP=" + IpUtil.intToAddress(packet.client_ip) + ",
Elapsed Time="
+ packet.elapsed + ", SQL Time=" + packet.sql_time + ", TX Time=" +
packet.etx_time);
        }
    }

}
```

다음은 XVLog 어댑터 사용에 필요한 설정이다. 우선 제니퍼 서버의 enable_xvlog 옵션을 true로 설정한다.

```
enable_xvlog = true
```

그리고 제니퍼 서버의 xvlog_class 옵션으로 XVLog 인터페이스를 구현한 클래스를 설정한다.

```
xvlog_class = example.ConsoleXVLog
```

XVLog 어댑터를 컴파일한 후에 JAR 파일로 패키지하여 JENNIFER_HOME/server/common/lib 디렉토리에 복사한 후 제니퍼 서버를 재시작한다.

14.2.2.2. 경보 데이터 연동

SMS 어댑터로 경보 데이터를 다른 애플리케이션에 전송할 수 있다.

Notice: SMS 어댑터를 컴파일하려면 JENNIFER_HOME/server/common/lib/jenniferserver.jar 파일을 클래스 패스에 등록해야 한다.

SMS 어댑터를 사용하려면 우선 com.javaservice.jennifer.server.Sendsms 인터페이스를 구현한 클래스를 작성해야 한다. Sendsms 인터페이스는 다음과 같다.

```
package com.javaservice.jennifer.server;

public interface Sendsms {

    public void sendsms(ErrorObject[] alerts);

}
```

경보가 발령되면 Sendsms 인터페이스를 구현한 클래스의 sendsms 메소드가 호출된다. sendsms 메소드의 파라미터는 ErrorObject 객체 배열인데, 이 객체는 발령된 경보 데이터를 나타낸다.

com.javaservice.jennifer.server.ErrorObject 클래스는 다음과 같다.

```
package com.javaservice.jennifer.server;

import com.javaservice.jennifer.server.db.HashedString_DB;

public class ErrorObject {

    public long time = 0;

    public String agentname = null;

    public int type = 0;

    public int group_type = 0;

    public int reqkey_hash = 0;

    public String getReqKey() {
        return reqkey_hash == 0 ? null :
                HashedString_DB.getAPPL(reqkey_hash);
    }

    public float value = -1f;

    public String ipaddr = null;

    public String msg = null;

    public long tx_uuid;

}
```

public 접근자인 멤버 필드로 발령된 경보 내용을 확인할 수 있다. 다음은 각 필드에 대한 설명이다.

- time - long 유형으로, 경보가 발령된 시간을 의미한다.
- agentname - java.lang.String 유형으로, 경보와 관련한 제니퍼 에이전트 아이디를 의미한다. 경보가 특정 제니퍼 에이전트와 관련이 없는 경우에는 null일 수 있다.

- group_type - int 유형으로, 경보 분류 항목을 의미한다. 1은 CRITICAL을, 2는 ERROR를, 3은 WARNING을, 4는 MESSAGE를 의미한다. 그리고 0은 알려지지 않은 경보 유형(UNKNOWN)을 의미한다.
- type - int 유형으로, ERROR_JVM_DOWN, WARNING_JVM_CPU_HIGH 등과 같은 경보 유형을 의미한다. com.javaservice.jennifer.util.Def 클래스의 경보 유형과 동일한 이름의 상수 필드로 비교한다.
- reqkey_hash - int 유형으로, 애플리케이션 이름 해시 값을 의미한다. 애플리케이션 이름은 getReqKey 메소드로 획득한다.
- float value - float 유형으로, WARNING_APP_BAD_RESPONSE 유형의 경보와 같이 특정 임계치를 초과해서 경보가 발령된 경우에 임계치를 초과한 값을 의미한다. 따라서 임계치와는 상관이 없는 경보 유형에서는 의미가 없다.
- ipaddr - java.lang.String 유형으로, 경보와 관련한 제니퍼 에이전트 혹은 WMOND가 설치된 하드웨어의 IP 주소를 의미한다. 경보 유형이 ERROR_SYSTEM_DOWN인 경우에는 WMOND를 설치한 하드웨어의 IP 주소를 나타내고, 경보가 특정 제니퍼 에이전트와 관련이 없는 경우에는 null이 된다.
- msg - java.lang.String 유형으로, 경보와 관련한 메시지를 의미한다.
- tx_uuid - long 유형으로, 트랜잭션 UUID를 의미한다.

다음은 SMS 어뎁터로 ERROR_SERVICE_QUEUING 유형의 경보만을 데이터베이스에 저장하는 예제이다.

```
package example;

import com.javaservice.jennifer.server.ErrorObject;
import com.javaservice.jennifer.server.Sendsms;
import com.javaservice.jennifer.util.Def;

public class SaveAlertSendsms implements Sendsms {

    public void sendsms(ErrorObject[] alerts) {
        for (int i = 0; i < alerts.length; i++) {
            ErrorObject alert= alerts[i];
            if (alert.type == Def.ERROR_SERVICE_QUEUING) {
                // 데이터베이스 저장
            }
        }
    }

}
```

경보 데이터 연동 작업을 하는데 도움이 되는 클래스와 메소드에 대해서 설명한다. 첫번째는 ErrStr 클래스로, int 유형의 경보 유형을 java.lang.String 유형의 경보 이름으로 전환하는데 사용한다.

```
com.javaservice.jennifer.util.ErrStr
```

- getShortMsg(int id) - 정적 메소드로, 경보 유형을 파라미터로 이 메소드를 호출하면 축약 경보 이름이 반환된다.
- getFullMsg(int id) - 정적 메소드로, 경보 유형을 파라미터로 이 메소드를 호출하면 전체 경보 이름이 반환된다.
- getTypeChar(int id) - 정적 메소드로, 경보 유형을 파라미터로 이 메소드를 호출하면 경보 분류 항목이 반환된다. ‘C’는 CRITICAL을, ‘E’는 ERROR를, ‘W’은 WARNING을, ‘I’는 MESSAGE를 의미한다.

두번째는 AlertMsgCtr 클래스로, 경보 차트에 나타나는 메시지를 획득하는데 사용한다.

```
com.javaservice.jennifer.util.AlertMsgCtr
```

- getMessage(ErrorObject e) - 정적 메소드로, 이 메소드를 호출하면 경보에 대한 자세한 메시지가 반환된다. 이 메시지는 경보 차트에 출력되는 메시지와 동일하다.

세번째는 TrapSender 클래스로, SNMP TRAP 메시지를 전송하는데 사용한다.

```
com.javaservice.jennifer.server.snmp.TrapSender
```

- addTrapMsg(String s) - 이 메소드를 호출하면 SNMP TRAP으로 전송할 메시지가 추가된다.

다음은 SMS 어뎁터 사용에 필요한 설정이다. 제니퍼 에이전트의 sms_adapter_class_name 옵션으로 Sendsms 인터페이스를 구현한 클래스를 설정한다.

```
sms_adapter_class_name = example.SaveAlertSendsms
```

Notice: 두 개 이상의 SMS 어뎁터를 등록할 수 있는데 두개 이상의 SMS 전송 클래스는 세미 콜론[;]을 구분자로 구분한다.

SMS 어뎁터를 컴파일한 후에 JAR 파일로 패키지하여 JENNIFER_HOME/server/common/lib 디렉토리에 복사한다. 그리고 제니퍼 서버를 재시작한다.

14.3. Specialized 모니터링

SOA 구현을 위한 솔루션을 모니터링하는 방법을 설명한다.

Notice: 해당 솔루션이 사용되는 방법에 따라서 다음에서 설명하는 내용이 적합하지 않을 수 있다. 따라서 해당 솔루션을 모니터링하는데 이 내용을 참고 자료로 사용하는 것을 권장한다.

14.3.1.오라클 WLI 모니터링

WLI(WebLogic Integration)를 모니터링하려면 다음 설정이 필요하다.

우선 애플리케이션 서비스 시작점이 javax.servlet.http.HttpServlet 클래스가 아닌 경우가 많기 때문에 제니퍼 에이전트의 tx_server_class 옵션과 tx_server_target_method 옵션에 다음 내용을 설정한다.

```
tx_server_class =
com.bea.wli.knex.runtime.core.dispatcher.Dispatcher;weblogic.jms.clien
t.JMSSession
tx_server_target_method = remoteDispatch;onMessage
```

그리고 애플리케이션 이름을 간단하게 하기 위해서 제니퍼 에이전트의 tx_server_ntype 옵션을 SIMPLE로 설정한다.

```
tx_server_ntype = SIMPLE
```

그리고 WLI는 JDBC 연결을 위해서 아파치 BeeHive를 사용한다. JDBC 모니터링을 위해서 제니퍼 에이전트의 jdbc_connection_justget 옵션을 다음과 같이 설정한다.

```
jdbc_connection_justget =
org.apache.beehive.controls.system.jdbc.JdbcControlImpl.getConnectionF
romDataSource(String,Class)
```

앞의 설정은 애플리케이션 서비스 시작점과 JDBC 모니터링을 위한 것이다. 그런데 상세한 모니터링을 위해서는 CustomTrace 어뎁터를 사용해서 애플리케이션의 주요 정보를 추출해야 한다.

Notice: 애플리케이션 소스 코드의 수정이 필요하다.

단, WLI를 위한 CustomTrace 어뎁터 구현 클래스를 제니퍼가 제공하므로 CustomTrace 어뎁터를 별도로 구현할 필요는 없다. 제니퍼 에이전트의

custom_trace_adapter_class_name 옵션과 custom_trace_adapter_class_path 옵션을 다음과 같이 설정한다.

```
custom_trace_adapter_class_name = jennifer.custom.AdapterWLI
custom_trace_adapter_class_path = /jennifer/agent/lwst40.custom.jar
```

Notice: `lwst40.custom.jar` 파일은 `JENNIFER_HOME/agent` 디렉토리에 존재한다.

CustomTrace 어뎁터가 적용되는 클래스는 제니퍼 에이전트의 custom_trace_class 옵션으로 설정한다.

```
custom_trace_class = jennifer.JenniferGate
```

JenniferGate 클래스는 별도의 JAR 파일로 제공되는 것이 아니다. WLI를 사용하는 애플리케이션에서 다음 코드를 작성하여 클래스 패스에 등록한다.

```
package jennifer;

public class JenniferGate {
    public static void TRAN_NAME(String s){}
    public static void TRAN_STEP(String s){}
    public static void TRAN_ID(String s){}
    public static void WARN(String s){}
    public static void PROFILE(String s){}
}
```

그리고 애플리케이션에서 JenniferGate 클래스의 메소드를 호출하여 주요 정보를 설정하면, 앞에서 설정한 CustomTrace 어뎁터가 적용되면서 해당 정보를 추출한다. 다음은 JenniferGate 클래스의 메소드에 대한 설명이다.

- TRAN_NAME - 애플리케이션 서비스 이름을 변경하려면 이 메소드를 호출한다.
- TRAN_STEP - **[실시간 모니터링 | 애플리케이션]** 메뉴의 액티브 서비스 탭에 나타나는 액티브 서비스의 상태를 설정하려면 이 메소드를 호출한다.
- TRAN_ID - 글로벌 트랜잭션 연계 추적을 위한 GUID를 설정하려면 이 메소드를 호출한다.
- WARN - 현재 트랜잭션에서 임의의 예외가 발생한 것으로 설정하려면 이 메소드를 호출한다. 이 메소드를 호출하면 WARNING_CUSTOM_EXCEPTION 유형의 예외가 발생된다.
- PROFILE - X-View 프로파일 데이터에 임의의 프로파일 항목을 추가하려면 이 메소드를 호출한다.

14.3.2.오라클 ALSB 모니터링

제니퍼는 ESB(Enterprise Service Bus)와 같은 SOA 솔루션을 모니터링하는 것에 초점을 맞추고 있지는 않다. 하지만 ESB 기반 애플리케이션도 WAS에서 동작하기 때문에 제니퍼로 일정 부분의 모니터링이 가능하다. 여기서는 ESB 솔루션의 하나인 오라클 ALSB(Oracle Service Bus 혹은 Aqualogic Service Bus)를 제니퍼로 모니터링하는 방법을 설명한다.

우선, 제니퍼 에이전트의 alsb_enabled 옵션을 true로 설정한다. 기본 값은 true이다.

```
alsb_enabled = true
```

ALSB는 하나의 요청을 2개 이상의 자바 쓰레드, 즉 2개 이상의 트랜잭션으로 처리한다. 따라서 글로벌 트랜잭션 연계 추적이 필요하다. 글로벌 트랜잭션 연계 추적을 하려면 제니퍼 에이전트의 trace_related_transaction 옵션을 true로 설정한다.

```
trace_related_transaction = true
```

글로벌 트랜잭션 연계 추적을 하는데 있어서 제니퍼 에이전트가 트랜잭션에 부여하는 UUID를 GUID로 사용하기 위해서 제니퍼 에이전트의 enable_guid_from_tuid 옵션을 true로 설정한다.

```
enable_guid_from_tuid = true
```

그리고 ALSB가 요청을 처리하다가 발생한 예외 데이터를 추적해야 하다. ALSB는 예외 데이터를 SOA 메시지에 포함하여 전달한다. 따라서 예외 데이터를 확인하기 위해서는 SOA 메시지를 저장해야 한다. SOA 메시지를 저장하려면 제니퍼 에이전트의 dump_soa_msg 옵션을 true로 설정한다.

```
dump_soa_msg = true
```

dump_soa_msg 옵션을 true로 설정하면 제니퍼 에이전트는 SOA 메시지를 제니퍼 에이전트의 로컬 데이터베이스에 저장한다. 제니퍼 에이전트의 로컬 데이터베이스를 사용하려면 제니퍼 에이전트의 다음 옵션을 설정한다.

```
agent_db_enabled = true
agent_db_rootpath = .
agent_db_keep_days = 7
```

agent_db_enabled 옵션으로 제니퍼 에이전트의 로컬 데이터베이스 사용 여부를 설정하고, agent_db_rootpath 옵션으로 제니퍼 에이전트의 로컬 데이터베이스 디렉토리 위치를 설정한다. 그리고 agent_db_keep_days 옵션으로 데이터 보관 주기를 설정한다. 기본 값은 7일이다.

Notice: SOA 메시지는 X-View 메시지 텝에서 오른쪽 마우스를 클릭하면 나타나는 컨텍스트 메뉴의 **[상세 보기]** 메뉴를 통해서 확인할 수 있다.

ASLB는 하나의 요청을 여러 개의 자바 쓰레드, 즉 여러 개의 트랜잭션으로 처리한다. 첫 번째 트랜잭션의 애플리케이션 서비스 시작점은 javax.servlet.http.HttpServlet 클래스이지만 다른 연계 트랜잭션의 애플리케이션 서비스 시작점은 별도로 설정해야 한다. 이를

위해서 제니퍼 에이전트의 tx_server_class, tx_server_target_method, tx_server_ntype 옵션을 다음과 같이 설정한다.

```
tx_server_class = com.bea.wli.sb.pipeline.MessageProcessor
tx_server_target_method = processResponse
tx_server_ntype = CLASS
```

그리고 ALSB는 외부 애플리케이션과의 연계를 주요 목적으로 한다. 따라서 외부 트랜잭션 시작점을 설정해야 한다. 기본적으로 오라클 턱시도나 시벨 등과 연동하는 경우에는 제니퍼 에이전트의 tx_client_class, tx_client_target_method, lwst_txclient_method_using_return 옵션을 다음과 같이 설정한다.

```
tx_client_class =
com.bea.wli.sb.transports.tuxedo.TuxedoTransportProvider;com.bea.alsb.
transports.siebel.SiebelTransportProvider


tx_client_target_method = sendMessageAsync;processMessage


lwst_txclient_method_using_return =
weblogic.wtc.gwt.TDMImport.getRemoteName();
com.bea.alsb.transports.siebel.impl.SiebelEndpointConfigurationImpl.ge
tBusinessName()
```

다른 외부 애플리케이션과 연동을 하면 이를 외부 트랜잭션 시작점에 추가해야 한다.

사용자는 ALSB에 의한 워크 플로우 단계들을 X-View 프로파일로 확인할 수 있고, X-View 차트의 GUID 뷰를 통해서 글로벌 트랜잭션 연계를 모니터링할 수 있다.

Notice: ALSB를 모니터링하면 제니퍼가 수집하는 호출 건수와 실제 호출 건수의 차이가 클 수 있다. 하나의 요청을 여러 개의 자바 쓰레드로 처리하면 제니퍼는 각각을 별도의 트랜잭션으로 간주하여 호출 건수도 증가시킨다. 따라서 호출 건수를 분석할 때는 상대적 추세 관점에서만 사용하는 것을 권장한다.

14.4. 제니퍼 모바일

스마트 폰에서 사용하기 위한 제니퍼 클라이언트이다. 이것을 제니퍼모바일이라 부른다.

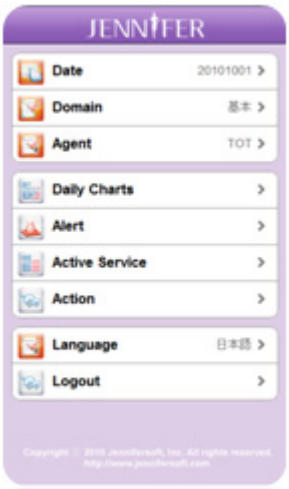
- 제니퍼 서버에 포함되어있다 별도의 설치가 필요없다.
- 웹 포트만을 사용한다.

- 표준 브라우저를 지원하는 폰은 모두 사용할 수 있다.
- 로그인은 제니퍼사용자 아이디와 패스워드를 같이 사용한다.

14.4.1.기본파라미터

제니퍼모바일화면에서는 DAte, Domain, Agent를 기본 파라미터로 선택해야 한다.

그림 14-3: 기본 파라미터

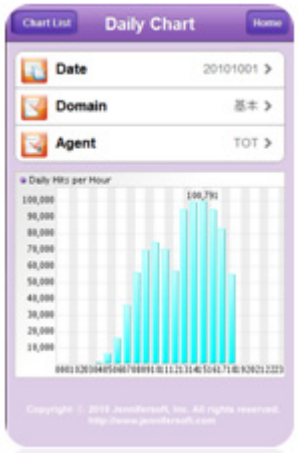


- Date : 날짜 선택
- Domain : 도메인 선택
- Agent : 에이전트 선택

14.4.2.Daily Charts

조회를 원하는 차트 선택 단 일자별 데이터를 조회된다.

그림 14-4: Daily Charts



- Hit Per Hour: 시간당 호출건수
- Service Rate : 초당 서비스율
- Average Response Time: 평균 응답시간
- Visitors per Hour : 시간당 방문자수
- Concurrent Users : 동시 사용자
- Active Service : 액티브 서비스
- System CPU Utilization :시스템 CPU 사용량(단 TOT는 제외)
- Process CPU Utilization : 프로세스 CPU 사용량(단 TOT는 제외)
- Heap Memory Rate: 힙 메모리 사용량 (단 TOT는 제외)
- Process Memory Usage: 프로세스 메모리 사용량(단 TOT는 제외)
- System Memory Usage : 시스템 메모리 사용량(단 TOT는 제외)

14.4.3.Alert

발생한 경보 내역을 조회할 수 있다. 아래의 조건을 입력하고 검색하면 시간의 역순으로 출력된다.

- Type : 경보 타입을 선택
- Time : 경보가 발생한 시간
- Condition : 특별히 필터링하고자 하는 경보 이름 혹은 상세 문자열
- Max Count : 최대 출력 갯수

14.4.4.Active Service

현재의 액티브 서비스 수와 상세 내역을 조회할 수 있다. 단 TOT가 선택된 경우에는 Active Service만 조회되고 일반 에이전트가 선택된 경우에는 해당에이전트의 시스템 CPU사용율 까지 보여진다.

그림 14-5: Active Service

14.4.5.기타

- Language : 사용자 언어를 선택한다.
- Logout : 로그 아웃

14.4.6.Action 등록 및 사용

제니퍼모바일에서는 폰에서 서버 관리를 위한 주요 쉘을 호출할 수 있다. 제니퍼를 설치한 경우 예제는 서버 실행과 종료 프로세스 리스트 로그뷰등이 포함되었다.

Notice: 주의: 실제 시스템을 제어하는 백도어가 될 수 있다 따라서 보안문제를 검증한 다음에 사용해야 한다.

제니퍼 에이전트 모듈을 시스템에 설치하고 $JENNIFEr_AGENT/action디렉토리에 있는 쉘파일들의 실행권한과 경로를 맞추어 주어야 한다.

main.sh를 실행하면 제니퍼 모바일의 요청을 처리하는 RmAgent가 실행된다.

```
JAVA_HOME=/usr/java
JENNIFER_SERVER=192.168.0.101

$JAVA_HOME/bin/java -cp ../rmagent/RmAgent.jar RmAgent.MobileAction \
             -a X11,X12,X13 \
             -t 127.0.0.1:6902 \
             -l 127.0.0.1:7715 \
             -cmd:start "./start.sh"  \
             -cmd:stop ./stop.sh \
             -cmd:log ./log.sh \
             -cmd:ps ./ps.sh \
             -cmd:dummy2 ./dummy2.sh
```

Notice: main.sh 내부의 정보는 사용자가 직접 시스템 환경에 맞게 수정해야 한다.

- start.sh : WAS 실행
- stop.sh : WAS 중단
- log.sh : LOG 조회
- ps.sh : 프로세스 보기

각 Action은 모바일 화면에도 등록되어야 하는데 제니퍼서버의 설정에서 아래와 같이 등록한다

```
mobileActions=start:Start;stop:Stop;log:Log View;ps:Process
```

하나의 액션은 ” ;”으로 구분하여 등록한다. main.sh에 등록된 cmd:<action>에서 <action>의 이름을 기술하고 그것이 화면에서 보여질 이름을 적어준다.

15

REMON

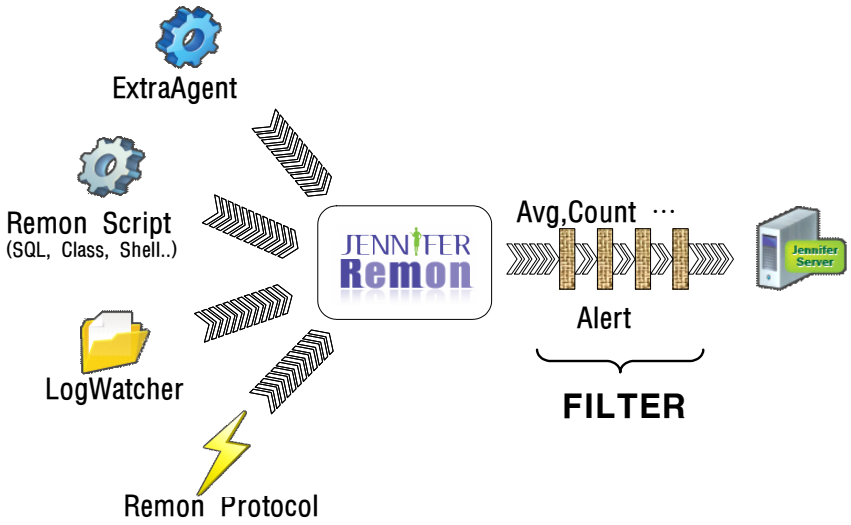
REMON은 제니퍼 독립 에이전트의 하나로, 비정형 데이터를 수집하고 가공하여 이를 제니퍼 서버에 전송한다. 일반적으로 모니터링 솔루션은 데이터 수집, 가공, 전송, 저장, 뷰 등의 기능으로 구성되는데 REMON은 데이터 수집, 가공, 전송을 일반화한 프레임워크이다.

REMON을 사용하면, 임의의 운영 체계에서 임의의 스크립트를 실행시키거나, 임의의 데이터베이스에 대해서 임의의 SQL을 수행시키거나, 특정 자바 인터페이스를 구현한 자바 클래스를 실행시키는 방법등으로 비정형 데이터를 수집하여 제니퍼 서버에 전송할 수 있다.

이를 통해서 제니퍼 에이전트만으로는 수집이 불가능한 비즈니스 데이터와 시스템 운영 체계에 대한 모니터링이 가능하다.

15.1. REMON 아키텍처

그림 15-1: REMON 구성도



REMON은 데이터를 직접 수집하거나, 외부 데이터소스로부터 데이터를 받아들인다.

첫번째로 데이터를 직접 수집하는 모듈을 REMON 스크립트라고 한다. REMON 스크립트는 단순하게 유닉스 쉘 스크립트를 의미하는 것이 아니다. 유닉스 쉘 스크립트뿐만 아니라 SQL 스크립트, 자바 클래스 스크립트 등이 REMON 스크립트에 포함된다. 그리고 REMON 스크립트는 데이터 수집 위치에 따라서 로컬 스크립트와 원격 스크립트로 구분된다. 데이터 수집이 REMON 자체에서 이루어지면 로컬 스크립트라고 하고, 원격의 다른 운영 체계에서 이루어지면 원격 스크립트라고 한다.

로컬 스크립트의 종류는 다음과 같다.

- 운영 체계 쉘 스크립트
- 클래스 스크립트

원격 스크립트의 종류는 다음과 같다.

- 텔넷 스크립트
- SSH2 스크립트
- SQL 스크립트

두번째로 REMON이 데이터를 직접 수집하지 않고 외부 데이터소스로부터 데이터를 받기도 한다. 외부 데이터소스는 다음과 같다.

- REMONX
- LogWatcher

이렇게 REMON 스크립트와 외부 데이터소스로부터 수집한 데이터를 REMON 데이터라고 한다. REMON 데이터는 데이터의 출처와 상관없이 동일한 형태를 갖는다.

REMON 데이터는 에이전트 아이디와 스크립트 아이디의 조합으로 고유하게 구분된다.

- 에이전트 아이디 - REMON 데이터의 출처를 의미한다. 대문자와 숫자, 밑줄[_] 만으로 구성되어야 하며 크기는 128자까지 가능하다. 단, REMON 데이터를 X-ViewC 차트로 모니터링하는 경우에는 3자이어야 한다.
- 스크립트 아이디 - REMON 데이터의 성격을 의미한다. CPU 사용률, 네트워크 사용량, 제품 제고량 등과 같이 데이터의 성격을 규정하는 역할을 한다. 대문자와 숫자, 밑줄[_] 만으로 구성되어야 하며 크기는 256자까지 가능하다.

그리고 REMON 데이터는 1개 이상의 필드로 구성된다. 단, 모든 필드의 데이터 유형은 동일해야 한다. 필드 유형으로는 int, float, long, double, string 등이 가능하다.

Notice: 기본적으로 필드의 개수에는 제한이 없다. 단, REMON 데이터를 제니퍼 서버에서 저장하는 경우에는 테이블 최대 칼럼 수에 영향을 받는다.

마지막으로 REMON은 REMON 데이터를 UDP 방식으로 제니퍼 서버에 전송한다. REMON 데이터에 대한 가공이 필요한 경우, 필터를 사용하여 REMON 데이터를 가공한 후에 제니퍼 서버에 전송할 수 있다. 하나의 REMON 데이터에 대해서 여러 개의 필터를 중첩해서 적용할 수 있다.

Notice: REMON 데이터를 2개 이상의 제니퍼 서버에 중복해서 혹은 선택적으로 전송할 수 있다.

그리고 REMON에 대한 전반적인 관리는 콘솔 기반의 제어 콘솔로 이루어진다.

15.2. 설치 및 설정

제니퍼 설치 패키지에 있는 remon 디렉토리가 REMON 모듈이다. REMON의 디렉토리 구조는 다음과 같다.

- conf - REMON 설정 파일들이 위치한다.
- lib - REMON이 사용하는 자바 라이브러리 파일들이 위치한다.
- log - 로그 파일이 위치한다.
- logwatcher - 로그 감시기의 역할을 수행하는 LogWatcher이다. REMON에 기반하기 때문에 REMON 디렉토리에 위치한다.
- remonx - REMON을 설치하지 않고 임의의 스크립트 수행 결과를 REMON에 전송하는 REMONX 모듈이 위치한다.
- script - REMON 스크립트 파일들이 위치한다.

REMON을 설치하려면 우선 이 디렉토리를 REMON을 설치할 운영 체계에 복사한다.

Notice: 유닉스와 리눅스에 설치하는 경우에는 복사 후에 실행 파일에 대해서 실행 권한을 주어야 한다.

REMON은 자바로 구현되어 있기 때문에 자바 실행파일의 패스를 JAVA_HOME 환경 변수로 설정한다. 그리고 REMON을 설치한 디렉토리도 REMON_HOME 환경 변수로 설정한다.

Notice: REMON을 실행하려면 자바 1.4.2 이상을 사용해야 한다.

JAVA_HOME과 REMON_HOME 환경 변수는 REMON_HOME 디렉토리의 env.sh(env.bat) 파일에서 설정한다. 유닉스 혹은 리눅스의 경우에는 다음과 같이 수정한다.

```
REMON_HOME=/usr/jennifer/remon
JAVA_HOME=/usr/java
```

윈도우의 경우에는 다음과 같이 수정한다.

```
set REMON_HOME=C:/jennifer/remon
set JAVA_HOME=C:/Program Files/Java/jdk1.6.0_07
```

REMON이 사용하는 네트워크 정보와 REMON 데이터를 전송할 제니퍼 서버 등을 REMON_HOME/conf/remon.conf 파일에 설정한다.

우선 에이전트 아이디를 다음과 같이 설정한다. 데이터를 수집할 때 명시적으로 에이전트 아이디를 부여하지 않은 모든 REMON 데이터의 에이전트 아이디는 이 설정을 따른다.

```
remon.agent = R01
```

REMON이 외부 데이터소스로부터 데이터를 받아들이는 UDP 포트를 local.udp.port 옵션으로 설정한다. 기본 포트 번호는 7701이다.

```
local.udp.port = 7701
```

또한 REMON은 제어 콘솔과 제니퍼 서버로부터 TCP 요청을 받아들인다. 이를 위한 TCP 포트를 local.tcp.port 옵션으로 설정한다. 기본 포트 번호는 7701이다.

```
local.tcp.port = 7701
```

REMON에서는 데이터 혹은 요청을 받아들이는 UDP 포트와 TCP 포트로 같은 번호를 사용할 수 있다. 기본적으로 7701 포트 번호를 함께 사용한다.

그리고 REMON 자체의 IP 주소를 local.ip 옵션으로 설정한다. 제니퍼 서버가 REMON 서버로 역방향 TCP 호출을 할때 이 IP 주소를 사용한다.

```
local.ip = 127.0.0.1
```

마지막으로 REMON 데이터를 전송할 제니퍼 서버 정보를 설정한다.

```
jennifer.sys1.ip = 127.0.0.1
jennifer.sys1.port = 6902
jennifer.sys1.agent = *
```

sys1은 임의의 값으로 일반적으로 제니퍼 서버의 도메인 아이디를 의미한다. 여러 개의 제니퍼 서버에 REMON 데이터를 전송해야 하는 경우에는 sys1 대신에 다른 값을 사용해서 옵션을 추가한다.

```
jennifer.sys1.ip = 192.168.0.1
jennifer.sys1.port = 6902
jennifer.sys1.agent = *

jennifer.sys2.ip = 192.168.0.2
jennifer.sys2.port = 6902
jennifer.sys2.agent = *
```

jennifer.sys1.ip 옵션에는 제니퍼 서버의 IP 주소를 설정하고, jennifer.sys1.port 옵션에는 제니퍼 서버의 server_udp_listen_port 옵션으로 설정한 포트 번호를 설정한다. 기본 포트 번호는 6902이다.

일반적으로 모든 REMON 데이터는 모든 제니퍼 서버에 전송된다. 이는 jennifer.sys1.agent 옵션을 *로 설정했기 때문이다. 만약 특정 REMON 데이터만을 송신하려면 [,]를 구분자로 에이전트 아이디를 설정한다.

```
jennifer.sys1.agent = R01,R03,R05
```

Notice: 스크립트 아이디가 아닌 에이전트 아이디만으로 REMON 데이터를 전송할 제니퍼 서버를 선택할 수 있다.

15.3. 실행과 제어

15.3.1. 실행하기

REMON을 실행하려면 REMON_HOME 디렉토리의 remon.sh를 실행한다.

```
./remon.sh
```

이 때 SeedKey를 입력해야 한다. 기본적으로 jennifer를 입력한다. SeedKey는 REMON 스크립트와 설정 파일에서 사용하는 패스워드를 암호화할 때 사용하는 키이다. 따라서 이 SeedKey 값을 잘못 입력하면 REMON이 정상적으로 동작하지 않는다.

SeedKey를 변경하려면 REMON을 실행할 때 변경할 값을 입력하면 된다. 그런데 SeedKey를 변경한 경우에는 REMON 설정 파일이나 스크립트에 존재하는 모든 패스워드를 다시 수정해야 한다.

15.3.2. 정지하기

REMON을 정지하려면 REMON_HOME 디렉토리의 shutdown.sh를 실행한다.

```
./shutdown.sh
```

15.3.3. 제어 콘솔

REMON은 콘솔 기반의 제어 콘솔로 관리한다.

Notice: 제니퍼 3.x의 REMON은 웹 기반 사용자 인터페이스를 제공했지만 제니퍼 4.0의 REMON은 콘솔 기반 사용자 인터페이스만을 제공한다.

제어 콘솔을 실행하려면 REMON_HOME 디렉토리의 console.sh를 실행한다.

```
~/jennifer/remon$ ./console.sh


Jennifer REMON: Release 4.0.1.1(2008-10-31)
Copyright (c) JenniferSoft 1998,2008. All rights reserved.


REMON>
```

제어 콘솔을 종료하려면 exit 명령어를 수행한다.

```
REMON> exit
```

제어 콘솔에서 여러 명령어로 REMON 스크립트 제어, 필터 제어 등의 작업을 수행한다. 각 명령어에 대한 상세한 내용을 확인하려면 help 명령어를 수행한다.

```
REMON> help
```

다음은 주요 명령어에 대한 설명이다.

제어 콘솔에서 shutdown 명령어로 REMON을 종료할 수 있다. 이 경우에는 제어 콘솔도 함께 종료된다.

```
REMON> shutdown
```

REMON을 실행하는 자바의 환경 변수를 확인하려면 env 명령어를 수행한다.

```
REMON> env
```

특정 문자열로 시작하는 환경 변수만을 확인하려면 다음과 같이 명령어를 수행한다.

```
REMON> env [임의의 문자열]


예) env java
```

REMON의 환경 설정을 확인하려면 conf 명령어를 수행한다.

```
REMON> conf
```

특정 문자열로 시작하는 REMON 환경 설정만을 확인하려면 다음과 같이 명령어를 수행한다.

```
REMON> conf [임의의 문자열]


예) conf local
```

REMON의 자바 힙 메모리 사용량 상태와 데이터 큐 현황을 확인하려면 perf 명령어를 수행한다.

```
REMON> perf
```

REMON 운영 중에 발생한 에러 정보를 확인하려면 err 명령어를 수행한다.

```
REMON> err
```

REMON 운영 중에 발생한 에러 정보를 삭제하려면 errc 명령어를 수행한다.

```
REMON> errc
```

REMON 스크립트를 수행하려면 원격 서버 혹은 데이터베이스의 패스워드를 REMON 스크립트 파일에 설정하는 경우가 있다. 보안을 위해서 SeedKey를 기반으로 암호화한 값을 설정해야 하는데 이 때 enc 명령어를 사용한다.

```
REMON> enc <암호화할  문자열>
```

기타 다양한 명령어는 관련 부분에서 설명하도록 한다.

15.4. 데이터 수집

REMON은 REMON 스크립트를 이용하여 데이터를 직접 수집하거나 REMONX, ExtraAgent, LogWatcher 등의 외부 데이터소스로부터 데이터를 받아들인다.

15.4.1.REMON 스크립트

모든 REMON 스크립트는 텍스트 파일로 확장자를 통해서 스트립트 유형이 구분된다. 그리고 REMON 스크립트는 REMON_HOME/script 디렉토리에 존재해야 한다. 하위 디렉토리와 다른 디렉토리는 인식할 수 없다.

Notice: REMON 스크립트 샘플은 REMON_HOME/script/sample 디렉토리에서 확인할 수 있다.

REMON 스크립트의 종류는 다음과 같다.

- 운영 체계 쉘 스크립트
- 텔넷 스크립트
- SSH2 스크립트
- SQL 스크립트
- 클래스 스크립트

모든 REMON 스크립트는 실행을 제어하기 위한 속성을 갖는다. 다음은 REMON 스크립트 종류에 상관없이 공통적으로 사용 가능한 속성에 대한 설명이다.

표 15-1: REMON 스크립트 공통 속성

속성	설명	비고
script	데이터의 성격을 구분하는 역할을 하는 스크립트 아이디를 설정한다. 모든 스크립트에 대해서 이를 설정해야 한다. 에이전트 아이디와 함께 REMON 데이터를 지칭한다.	대문자, 숫자, _
agent	데이터의 출처를 구분하는 에이전트 아이디를 지정한다. 지정하지 않으면 REMON 설정 파일의 remon.agent 옵션 값을 기본 값으로 한다. 스크립트 아이디와 함께 REMON 데이터를 지칭한다. X-ViewC 차트로 모니터링하려면 에이전트 아이디는 3자이어야 한다.	대문자, 숫자, _
interval	REMON 스크립트가 수행되는 주기를 설정한다. 단위는 초이고 기본 값은 1초이다.	양의 정수
fieldname	REMON 스크립트가 수집하는 REMON 데이터는 1개 이상의 필드로 구성된다. 이 필드에 대해서 이름을 지정할 때 사용한다. 콤마[,]를 구분자로 설정한다. 필드 이름을 설정하지 않으면 f0, f1, f1 등의 이름이 순차적으로 부여된다.	소문자,숫자
fieldtype	REMON 데이터를 구성하는 필드의 유형을 지정한다. 모든 필드의 유형은 동일해야 한다. 따라서 fieldname 속성과는 달리 하나의 값만을 설정한다. 기본 값은 string이다.	[int | float | long | double | string]
request	REMON 스크립트가 제니퍼 서버의 요청을 받을때만 실행한다는 것을 의미한다. 기본값은 false이다.	[true | false]
autostart	REMON이 실행될 때 REMON 스크립트를 자동으로 실행할지에 대한 여부를 설정한다. false로 설정하면 REMON이 시작해도 REMON 스크립트는 동작하지 않는다. 반면에 true로 설정하면 REMON이 시작할 때 REMON 스크립트도 함께 동작한다. 기본 값은 true이다.	[true | false]

REMON 스크립트 속성은 #$로 시작하는 한 줄의 텍스트로 기술한다.

```
#$ agent = R01
```

단, 윈도우 bat 파일의 경우에는 앞에 REM을 적어준다.

```
REM #$ agent= R01
```

15.4.1.1. 운영 체계 쉘 스크립트 (.sh / .bat)

REMON을 설치한 운영 체계의 쉘 스크립트를 REMON 스크립트로 사용할 수 있다. 이를 운영 체계 쉘 스크립트라고 한다. 운영 체계 쉘 스크립트는 로컬 스크립트로, 확장자가 sh 혹은 bat으로 끝나야 한다.

다음은 REMON을 설치한 유닉스 혹은 리눅스 운영 체계의 네트워크 소켓 연결 상태를 수집하는 예제이다. 다음 내용을 REMON_HOME/script 디렉토리에 net.sh라는 이름으로 저장한다.

```
#!/bin/sh

################################################
#  REMON Script Attribute
#$ script = NET
#$ agent = R01
#$ interval = 5
#$ fieldtype = int
#$ fieldname = est,tim,fin
################################################

net=`netstat -an`
est=`echo $net | grep EST | wc -l`
tim=`echo $net | grep TIME | wc -l`
fin=`echo $net | grep FIN | wc -l`
echo $est,$tim,$fin
```

est, tim, fin 등의 필드로 ESTABLISHED 상태의 TCP 연결 개수, TIME_WAIT 상태의 TCP 연결 개수, FIN_WAIT 상태의 TCP 연결 개수를 int 유형으로 5초마다 수집한다. 그리고 이 REMON 데이터는 NET을 스크립트 아이디, R01을 에이전트 아이디로한다.

Warning: REMON이 운영 체계 쉘을 직접 실행하기 때문에 실행 권한이 있어야 한다.

15.4.1.2. 텔넷 스크립트(.telnet)

텔넷을 이용해서 원격으로 다른 운영 체계에 접속해서 쉘 스크립트를 수행하여 데이터를 수집할 수 있다. 이를 텔넷 스크립트라고 한다. 텔넷 스크립트는 원격 스크립트로, 확장자가 telnet으로 끝나야 한다.

다음은 텔넷 스크립트에서 사용하는 속성에 대한 설명이다

표 15-2: 텔넷 스크립트에서 사용하는 속성

속성	설명	비고
telnet.ip	접속 서버 IP	호스트 이름 또는 IP 주소
telnet.port	접속 서버 포트 번호	정수
telnet.user	접속 서버 사용자 아이디	문자열
telnet.password	접속 서버 사용자 패스워드	제어 콘솔의 enc 명령어로 패스워드를 암호화해서 등록해야 한다. SeedKey를 수정한 경우에는 다시 설정해야 한다.
prompt.user	로그인 prompt 문자열	
prompt.password	패스워드 prompt 문자열	
prompt.char	일반 prompt 문자열	
prompt.beforelogin	로그인 prompt가 나타나기 전에 사용자가 입력해야 하는 문자가 있는 경우 등록	
prompt.preprompt	로그인 패스워드를 입력하고 최초 prompt.char가 화면에 출력되기 전에 사용자가 입력해야 하는 문자가 있는 경우 등록	

로컬 스크립트는 interval 속성으로 수행 간격을 조절하지만, 원격 스크립트는 데이터를 주기적으로 수집하기 위해서 반드시 무한 루프 구조로 작성되어야 한다.

```
while true
do
  netstat -an | wc -l
  sleep 5
done
```

15.4.1.3. SSH2 스크립트(.ssh2)

SSH2를 이용해서 원격으로 다른 운영 체계에 접속해서 쉘 스크립트를 수행하여 데이터를 수집할 수 있다. 이를 SSH2 스크립트라고 한다. SSH2 스크립트는 원격 스크립트로, 확장자가 ssh2로 끝나야 한다.

다음은 SSH2 스크립트에서 사용하는 속성에 대한 설명이다.

표 15-3: SSH2 스크립트에서 사용하는 속성

표 15-4:

속성	설명	비고
ssh2.ip	접속 서버 IP	호스트 이름 또는 IP 주소
ssh2.port	접속 서버 포트 번호	정수
ssh2.user	접속 서버 사용자 아이디	문자열
ssh2.password	접속 서버 사용자 패스워드	제어 콘솔의 enc 명령어로 패스워드를 암호화해서 등록해야 한다. SeedKey를 수정한 경우에는 다시 설정해야 한다.

로컬 스크립트는 interval 속성으로 수행 간격을 조절하지만, 원격 스크립트는 데이터를 주기적으로 수집하기 위해서 반드시 무한 루프 구조로 작성되어야 한다.

```
while true
do
  netstat -an | wc -l
  sleep 5
done
```

15.4.1.4. SQL 스크립트(.sql)

외부 데이터베이스에 대해서 SQL을 수행하여 데이터를 수집할 수 있다. 이를 SQL 스크립트라고 한다. SQL 스크립트는 원격 스크립트로, 확장자가 sql로 끝나야 한다.

다음은 SQL 스크립트에서 사용하는 속성에 대한 설명이다.

표 15-5: SQL 스크립트에서 사용하는 속성

속성	설명	비고
jdbc.driver	JDBC 드라이버 클래스 이름	
jdbc.url	데이터베이스 접속 URL	
jdbc.user	데이터베이스 사용자 아이디	
jdbc.password	데이터베이스 사용자 패스워드	제어 콘솔의 enc 명령어로 패스워드를 암호화해서 등록해야 한다. SeedKey를 수정한 경우에는 다시 설정해야 한다.

SQL로 조회한 칼럼 이름이 데이터의 필드 이름이 되기 때문에 fieldname 속성을 설정할 필요가 없다.

Notice: JDBC 드라이버 파일을 REMON_HOME/lib/script 디렉토리에 복사한 후에 REMON을 재시작해야 한다.

SQL 스크립트는 원격 스크립트이지만 interval 속성으로 실행 주기를 설정할 수 있다.

다음은 SQL 스크립트 예제이다.

```
#########################################################
#  REMON Script Attribute
#$ script = PERF_CNT
#$ agent = R01
#$ interval = 2
#$ fieldtype = int

#$ jdbc.driver = org.apache.derby.jdbc.ClientDriver
#$ jdbc.url =  jdbc:derby://localhost:1527/jennifer
#$ jdbc.user = jennifer
#$ jdbc.password = rf7CtZ/Oy6YGEtFFY/fo2A==

SELECT COUNT(*) CNT FROM PERF_X_29
```

15.4.1.5. 클래스 스크립트(.cls)

remon.IClassScript 인터페이스를 구현한 클래스를 실행시켜서 데이터를 수집할 수 있다. 이를 클래스 스크립트라고 한다. 클래스 스크립트는 로컬 스크립트로, 확장자가 cls로 끝나야 한다.

다음은 클래스 스크립트에서 사용하는 속성에 대한 설명이다.

표 15-6: 클래스 스크립트에서 사용하는 속성

속성	설명	비고
exec	remon.IClassScript 인터페이스를 구현한 클래스 이름	

remon.IClassScript 인터페이스를 구현한 클래스에서 데이터 수집에 필요한 대부분의 작업을 수행하기 때문에 다른 REMON 스크립트보다 단순한다. 다음은 REMON의 자바 힙 메모리 사용량 데이터를 수집하는 예제이다.

```
#############################################
#  REMON Script Attribute
#$ script = REMON_HEAP
#$ agent = R01
#$ interval = 1

#$ exec = example.HeapMemoryClassScript
```

exec 속성으로 설정한 클래스는 remon.IClassScript 인터페이스를 구현해야 한다. 일반적으로 remon.IClassScript 인터페이스를 구현한 remon.AbstractClassScript 클래스를 상속하는 것을 권장한다.

Notice: 컴파일을 하려면 REMON_HOME/lib/sys/remon.jar 파일을 클래스 패스에 등록해야 한다. 그리고 컴파일한 클래스를 JAR 파일로 패키지하여 REMON_HOME/lib/script 디렉토리에 복사한 후에 REMON을 재시작한다. 클래스를 수정하면 앞의 작업을 반복한 후에 REMON을 재시작한다.

remon.AbstractClassScript 클래스는 다음과 같다.

```
package remon;

import java.util.Properties;
import com.javaservice.jennifer.protocol.IRmPacket;

abstract public class AbstractClassScript implements IClassScript {

    final public Properties properties = new Properties();
    abstract public IRmPacket process(String[] param);

}
```

따라서 remon.AbstractClassScript 클래스를 상속한 후에 process 메소드를 구현해야 한다.

```
package example;

import remon.AbstractClassScript;

import com.javaservice.jennifer.protocol.IRmPacket;
import com.javaservice.jennifer.protocol.RmPacket;
import com.javaservice.jennifer.type.INT;

public class HeapMemoryClassScript extends AbstractClassScript {

    public IRmPacket process(String[] param) {
        return null;
    }

}
```

process 메소드에서는 com.javaservice.jennifer.protocol.IRmPacket 인터페이스를 구현한 com.javaservice.jennifer.protocol.RmPacket 객체를 생성하고 전송할 데이터를 이 객체에 담은 후에 반환한다.

```
private static final int MB = 1024 * 1024;

public IRmPacket process(String[] param) {
    long now = System.currentTimeMillis();
    RmPacket packet = new RmPacket(now, "SCR1", "A01");

    Runtime runtime = Runtime.getRuntime();
    int total = (int) (runtime.totalMemory() / MB);
    int used = (int) (runtime.totalMemory() / MB - runtime.freeMemory()
/ MB);

    packet.addField("total", new INT(total));
    packet.addField("used", new INT(used));

    return packet;
}
```

우선 데이터를 수집한 시간, 스크립트 아이디, 에이전트 아이디를 파라미터로 RmPacket 객체를 생성한다. 단, 스크립트 아이디와 에이전트 아이디는 클래스 스크립트에 설정한 내용이 있으면 그 값으로 변경된다. 따라서 예제에서 REMON 데이터의 스크립트 아이디는 HEAP이 아니고 REMON_HEAP이 된다.

그리고 RmPacket 클래스의 addField 메소드를 통해서 데이터를 추가한다. addField 메소드의 첫번째 파라미터로 필드명을 지정하고, 두번째 파라미터로 데이터를 지정한다. 두번째 파라미터의 유형은 com.javaservice.jennifer.type.TYPE 클래스의 하위 클래스로 이를 통해서 값을 지정한다.

Notice: 다른 REMON 스크립트와는 다르게 하나의 remon.IClassScript 구현 클래스에서 다양한 유형의 TYPE 클래스를 사용할 수 있다.

RmPacket 클래스의 자세한 사용 방법은 [RmPacket 클래스의 사용(623 페이지)]을 참조한다.

remon.AbstractClassScript 클래스의 properties 멤버 필드에 클래스 스크립트에 설정한 모든 속성이 전달된다. 따라서 클래스가 실행될 때 필요한 속성들을 클래스 스크립트에 등록하여 전달할 수 있다.

```
################################################
#  REMON Script Attribute
...
#$ max = 1000
...


public IRmPacket process(String param[]) {
    ...
    String max = properties.get("max");
    ....
}
```

15.4.2.REMON 스크립트 등록 및 제어

모든 REMON 스크립트는 텍스트 파일로 REMON_HOME/script 디렉토리에 존재해야 한다. 그런데 이 디렉토리에 REMON 스크립트를 추가했다고 해당 REMON 스크립트가 동작하는 것은 아니다. 제어 콘솔을 통해서 이를 등록해야 한다.

Notice: 단, REMON을 재시작하면 REMON_HOME/script 디렉토리에 있는 모든 REMON 스크립트가 자동으로 등록된다.

실행중인 REMON 스크립트 목록을 확인하려면 제어 콘솔에서 ls 명령어를 수행한다.

```
REMON> ls
```

그리고 REMON_HOME/script 디렉토리에는 있지만 아직 REMON에 등록되지 않은 REMON 스크립트 목록을 확인하려면 제어 콘솔에서 lsn 명령어를 수행한다.

```
REMON> lsn
perf.sql
net.sh
```

이중에서 특정 REMON 스크립트를 등록하려면 REMON 스크립트 파일 이름을 파라미터로 해서 load 명령어를 수행한다.

```
REMON> load [REMON 스크립트 파일 이름 Prefix]


예) load net.sh
```

모든 REMON 스크립트를 등록하려면 파라미터 없이 load 명령어를 수행한다.

```
REMON> load
```

load 명령어를 통해서 새롭게 추가한 REMON 스크립트는 정지 상태에 있다. 이를 시작하려면 스크립트 아이디와 에이전트 아이디 조합을 파라미터로 start 명령어를 수행한다.

```
REMON> start [스크립트 아이디.에어전트 아이디 Prefix]
```

```
start NET
```

- 스크립트 아이디가 NET으로 시작하는 모든 REMON 스크립트를 시작한다.

```
start NET.R
```

- 스크립트 아이디가 NET이고 에이전트 아이디가 R로 시작하는 모든 REMON 스크립트를 시작한다.

```
start NET.R01
```

- 스크립트 아이디가 NET이고 에이전트 아이디가 R01로 시작하는 모든 REMON 스크립트를 시작한다.

모든 REMON 스크립트를 시작하려면 파라미터 없이 start 명령어를 수행한다. .

```
REMON> start
```

그리고 제어 콘솔에서 REMON 스크립트가 수집한 마지막 데이터를 확인하려면 스크립트 아이디와 에이전트 아이디 조합을 파라미터로 data 명령어를 수행한다.

```
REMON> data [스크립트 아이디.에어전트 아이디 Prefix]


예) data NET
2008-07-29 19:50:48.811:NET:R01(est=4,tim=3,fin=1)
```

모든 데이터를 확인하려면 파라미터 없이 data 명령어를 수행한다.

```
REMON> data
```

REMON 데이터를 삭제하려면 dataclear 명령어를 수행한다. 데이터 수집과 전송에 영향을 미치지 않는다.

```
REMON> dataclear
```

REMON 스크립트 내용을 수정한 후에 이를 반영하려면 스크립트 아이디와 에이전트 아이디 조합을 파라미터로 reload 명령어를 수행한다.

```
REMON> reload [스크립트 아이디.에어전트 아이디 Prefix]
```

모든 REMON 스크립트를 재시작하려면 파라미터 없이 reload 명령어를 수행한다.

```
REMON> reload
```

REMON 스크립트를 제거하려면 스크립트 아이디와 에이전트 아이디 조합을 파라미터로 해서 unload 명령어를 수행한다. 단, 정지 상태에 있는 REMON 스크립트만을 제거할 수 있다.

```
REMON> unload [스크립트 아이디.에어전트 아이디 Prefix]
```

모든 REMON 스크립트를 제거하려면 파라미터 없이 unload 명령어를 수행한다.

```
REMON> unload
```

REMON 스크립트 파일 내용을 cat 명령어로 확인할 수 있다. 단, 이 경우에는 정확한 스크립트 아이디와 에이전트 아이디를 파라미터로 입력해야 한다.

```
REMON> cat NET.R01
```

15.4.3.외부 데이터소스

REMON은 데이터를 직접 수집하기도 하지만 외부 데이터소스로부터 데이터를 받아들이기도 한다.

15.4.3.1. REMONX

REMONX는 자바를 설치하기 어려운 환경에서 데이터를 REMON에 전송하기 위한 C 모듈이다. 레거시 쉘에 삽입하여 사용하거나 기존 프로그램에 REMONX를 추가해서 사용한다.

REMONX는 REMON_HOME/remonx 디렉토리에 운영 체계 별로 존재한다. 운영 체계에 맞는 실행 파일이 없는 경우에는 직접 컴파일을 해야한다. 컴파일 방법은 makefile을 참고한다.

REMONX의 실행 방법은 다음과 같다.

```
remonx [-multi] [-tcp] -h host_ip -p port -s script_id -a agent_id -N/
S/D -d data -f fieldname
```

다음은 각 옵션에 대한 설명이다.

표 15-7: REMONX 실행 옵션

옵션명	설명
-multi	multi-row 데이터를 stdin으로 받아 전송할때 사용한다. 따라서 -multi가 지정되면 -d 옵션은 사용하지 않는다. ex) tail -f access.log | remonx -multi ...
-tcp	REMONX가 데이터를 TCP 방식으로 전송할 때 사용한다. 기본적으로 UDP 방식으로 전송한다.
-h host_ip	데이터를 전송할 REMON IP 주소를 설정한다.
-p port	데이터를 전송할 REMON 포트 번호를 설정한다.
-s script_id	REMONX가 전송하는 REMON 데이터의 스크립트 아이디를 설정한다.
-a agent_id	REMONX가 전송하는 REMON 데이터의 에이전트 아이디를 설정한다. X-ViewC 차트로 모니터링하려면 에이전트 아이디는 3자이어야 한다.

표 15-7: REMONX 실행 옵션

옵션명	설명
-N | -S | -D	REMONX가 전송하는 데이터 타입으로 -N은 숫자를, -S는 문자열을, -D는 증감량을 의미한다. 기본 값은 -S이다.
-d data	REMONX가 전송하는 데이터이다.
-f fieldname	필드 이름을 설정한다. -N 혹은 -D 유형의 데이터에만 의미가 있다.

다음은 REMONX로 유닉스 혹은 리눅스 운영 체계의 TCP 연결 상태를 수집하는 예제이다.

```
#!/bin/sh

while true
do
    est=`netstat -an | grep EST | wc -l`
    tim=`netstat -an | grep TIME | wc -l`
    fin=`netstat -an | grep FIN | wc -l`
    remonx -h 127.0.0.1 -p 7701 -a RX1 -s NET -N
           -d "$est,$tim,$fin" -f  “est,tim,fin”
    sleep 1
done
```

est, tim, fin 등의 필드로 ESTABLISHED 상태의 TCP 연결 개수, TIME_WAIT 상태의 TCP 연결 개수, FIN_WAIT 상태의 TCP 연결 개수를 int 유형으로 1초마다 수집한다. 그리고 REMONX는 이 데이터를 NET을 스크립트 아이디로, RX1을 에이전트 아이디로해서 REMON에 전송한다.

REMONX를 이용해서 경보를 발령할 수도 있다.

```
remonx -h 127.0.0.1 -p 7701 -s ALERT -a R01 -S -d  “FATAL: error
message”
```

데이터는 FATAL, ERROR, WARN 중의 하나로 시작해야 한다. 이 구분에 따라서 경보 유형이 다음과 같이 결정된다.

- FATAL - 제니퍼 서버에서 USER_DEFINED_FATAL 경보가 발령된다.
- ERROR - 제니퍼 서버에서 USER_DEFINED_ERROR 경보가 발령된다.
- WARN - 제니퍼 서버에서 USER_DEFINED_WARNING 경보가 발령된다.

C가 아닌 자바로 되어 있는 REMONX도 존재한다. 일반적으로 제니퍼 에이전트를 설치해서 모니터링하기에는 적합하지 않은 배치 형태로 동작하는 자바 애플리케이션을 모니터

링할 때 사용한다. 이를 사용하려면 REMON_HOME/remonx/jar/remonx.jar 파일을 클래스 패스에 등록해서 사용한다.

15.4.3.2. ExtraAgent

ExtraAgent는 제니퍼 에이전트의 모니터링 기능을 확장하기 위한 기능이다. ExtraAgent가 수집한 데이터는 REMON 혹은 제니퍼 서버로 전송된다.

경보 발령이나 평균 처리와 같은 2차 데이터 가공이 필요한 경우에는 REMON으로 데이터를 전송한다.

15.4.3.3. LogWatcher

LogWatcher는 로그 감시기의 역할을 수행한다. 다양한 애플리케이션이나 운영 체계가 기록하는 텍스트 로그 파일에서 임의로 지정한 패턴을 검출하고 이벤트를 발생시켜서 관련 데이터를 REMON에 전달한다. REMON은 LogWatcher로부터 전송받은 데이터를 2차 가공하여 제니퍼 서버에 전달한다.

15.5. 데이터 제어

REMON 스크립트 혹은 외부 데이터소스로부터 수집한 데이터를 REMON을 통해서 가공할 수 있다. 데이터 가공은 remon.IFilter 인터페이스를 구현한 클래스로 이루어진다. 이를 중첩해서 사용할 수 있기 때문에 필터라고 지칭한다.

기본적으로 자주 사용하는 필터를 제공하며, 사용자가 임의의 필터를 작성하여 추가할 수 있다.

Notice: REMON 스크립트 단위로 필터를 설정한다.

15.5.1. 기본으로 제공하는 필터

사용할 수 있는 필터는 확인하려면 제어 콘솔에서 filter 명령어를 수행한다.

```
REMON> filter
```

특정 이름으로 시작하는 필터에 대한 정보를 확인하려면 이름을 파라미터로 filter 명령어를 수행한다.

```
REMON> filter [필터 이름 Prefix]
예) filter ALERT
```

필터는 에이전트 아이디와 상관없이 스크립트 아이디만을 기준으로 REMON 스크립트에 적용된다. REMON 스크립트 별 필터 적용 현황을 확인하려면 lsf 명령어를 수행한다.

```
REMON> lsf
```

특정 스크립트에 대한 필터 적용 현황을 확인하려면 스크립트 아이디를 파라미터로 lsf 명령어를 수행한다.

```
REMON> lsf [스크립트 아이디 Prefix]
예) lsf NET
```

특정 REMON 스크립트에 특정 필터를 추가하려면 addf 명령어를 수행한다.

```
REMON> addf <스크립트 아이디> <필터 이름>
```

필터 설정을 변경한 후에 이를 반영하려면 스크립트 아이디를 파라미터로 appf 명령어를 수행한다. 그 전에는 필터 변경 사항이 반영되지 않는다.

```
REMON> appf NET
```

REMON 스크립트에서 필터를 제거하려면 delf 명령어를 수행한다. 필터 인덱스는 lsf 명령어로 확인할 수 있다.

```
REMON> delf <스크립트 아이디> <필터 인덱스>
예) delf NET 1
```

REMON 스크립트에 설정된 모든 필터를 제거하려면 스크립트 아이디를 파라미터로 rmf 명령어를 수행한다.

```
REMON> rmf <스크립트 아이디 Prefix>
예) rmf NET
```

REMON 스크립트에 설정된 필터들 중에서 특정 필터를 다른 필터로 변경하려면 스크립트 아이디와 필터 인덱스와 필터 이름을 파라미터로 setf 명령어를 수행한다.

```
REMON> setf <스크립트 아이디> <필터 인덱스> <변경할 필터 이름>
예) setf NET 1 ALERT
```

REMON 스크립트에 설정된 필터들 사이에 새로운 필터를 추가하려면 스크립트 아이디와 필터 인덱스와 필터 이름을 파라미터로 insf 명령어를 수행한다.

```
REMON> insf <스크립트 아이디> <필터 인덱스> <추가할 필터 이름>
예) insf NET 1 ALERT
```

다음은 주요 필터에 대한 설명이다. 기타 다른 필터들에 대한 설명은 제어 콘솔에서 filter 명령어로 확인한다.

15.5.1.1. DB_SAVE

기본적으로 REMON이 제니퍼 서버에 전송하는 REMON 데이터는 제니퍼 서버 성능 데이터베이스에 저장되지 않는다. 만약 특정 REMON 데이터를 제니퍼 서버 성능 데이터베이스에 저장하려면 SERVER_DB_SAVE 필터를 사용한다. 예를 들어, 스크립트 아이디가 NET인 REMON 스크립트가 전송한 REMON 데이터를 제니퍼 서버 성능 데이터베이스에 저장하려면 다음 명령어를 수행한다.

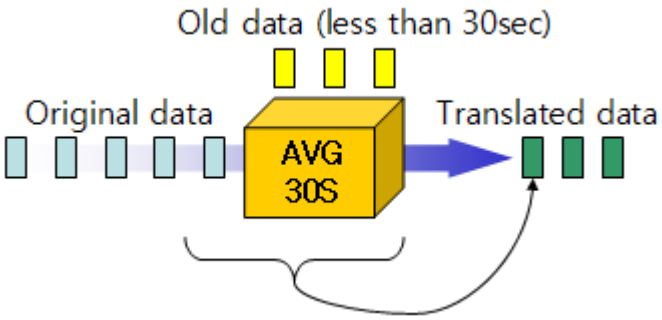
```
REMON> addf NET SERVER_SAVE 30
...
REMON> appf NET
...
```

addf 명령어의 마지막 파라미터는 저장 주기를 의미한다. 단위는 초이다. 앞의 예제에서는 30으로 지정했기 때문에, 제니퍼 서버는 30초마다 해당 REMON 데이터를 데이터베이스에 저장한다.

15.5.1.2. AVERAGE

기본적으로 마지막에 수집한 REMON 데이터를 제니퍼 서버에 전송한다. 그런데 특이치를 제거하기 위해서 특정 기간 동안의 평균 값을 제니퍼 서버에 전송할 필요도 있다. 이를 위해 사용하는 필터를 AVERAGE 필터라고 하고, AVG_30S, AVG_5M, AVG_60S 등의 필터가 존재한다.

그림 15-2: AVG_30S 실행 모습



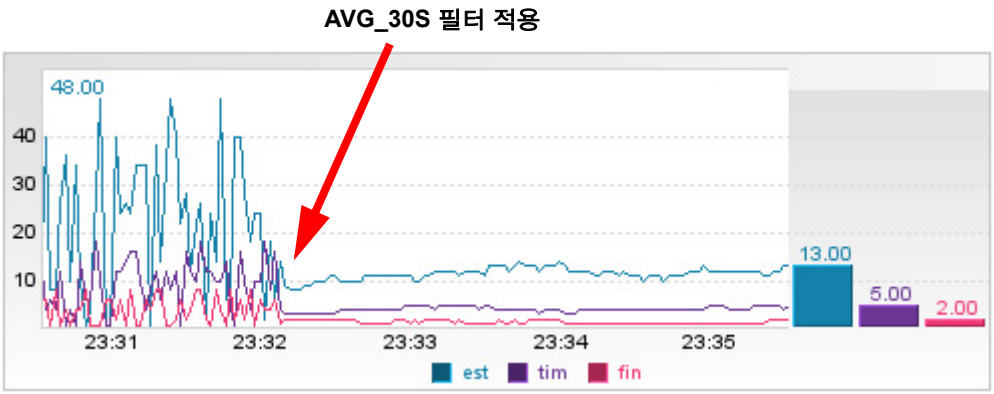
AVG_30S는 30초 평균 값을 생성한다. 필터에 전달되는 데이터 중에서 30초 이내의 데이터들을 메모리에 보관하고 있다가 새로운 데이터가 전달되면 이것들을 평균 데이터로 변형하여 반환한다.

NET을 스크립트 아이디로 하는 REMON 스크립트에 AVG_30S 필터를 적용하려면 다음 명령어를 수행한다.

```
REMON> addf NET AVG_30S
...
REMON> appf NET
...
```

다음 차트는 AVG_30S 필터를 적용하기 전과 적용한 이후의 데이터의 변화를 보여준다.

그림 15-3: AVG_30S 필터 적용 예



AVG_5M 필터는 데이터를 5분 평균 값으로 가공하고, AVG_60S 필터는 데이터를 1분 평균 값으로 가공한다.

15.5.1.3. SEND

SEND 필터는 데이터 전송 주기를 수정할 때 사용한다.

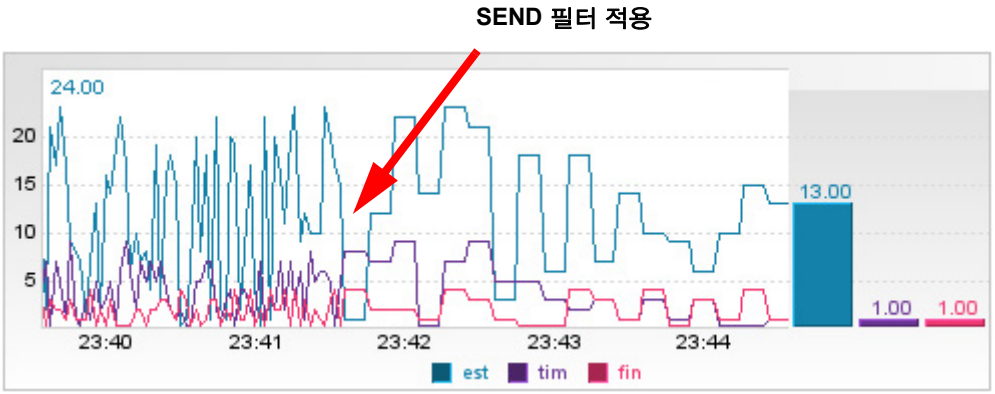
NET을 스크립트 아이디로 하는 REMON 스크립트의 REMON 데이터 전송 주기를 10초로 변경하려면 SEND 필터를 다음과 같이 적용한다.

```
REMON> addf NET SEND 10
...
REMON> appf NET
...
```

addf의 마지막 파라미터는 전송 주기로 단위는 초이다.

다음 차트는 SEND 필터를 적용하기 전과 적용한 이후의 데이터의 변화를 보여준다.

그림 15-4: SEND 필터 적용 예



REMON 스크립트가 1초 단위로 데이터를 수집하더라도 SEND 필터는 마지막 데이터 전송 시간에서 10초가 지난 경우에만 데이터를 전송하고 나머지는 전송하지 않는다. 따라서 SEND 필터가 적용되면 차트에서 관찰되는 모습은 앞의 그림과 같다.

15.5.1.4. STOP

STOP 필터는 REMON 스크립트가 수집한 데이터를 제니퍼 서버에 전송하지 않을 때 사용한다. 이 필터를 적용하면 REMON 스크립트가 데이터를 수집하기는 하지만 이를 제니퍼 서버에 전송하지는 않는다.

```
REMON> addf NET STOP
...
REMON> appf NET
...
```

15.5.1.5. ALERT

ALERT 필터를 이용하여 REMON 스크립트가 수집한 데이터에 대한 경보 처리를 할 수 있다. ALERT 필터를 추가할 때는 경보 조건과 경보 메시지 등의 파라미터를 설정해야 한다.

예를 들어, NET을 스크립트 아이디로 하는 REMON 스크립트는 est, tim, fin 등의 필드로 int 형의 데이터를 수집하고 있다.

```
REMON> data NET
2008-07-31 21:20:11.809:NET:R01(est=5,tim=2,fin=1)
```

이 때 est 필드의 값이 20 보다 큰 경우에 USER_DEFINED_FATAL 경보를 발령하려면 다음과 같이 ALERT 필터를 적용한다.

```
REMON> addf NET ALERT "est > 20"  "FATAL: The est is over 20(${est})"
...
REMON> appf NET
...
```

파라미터에 공백이 포함되면 파라미터의 시작과 끝을 ["]로 묶어야 한다.

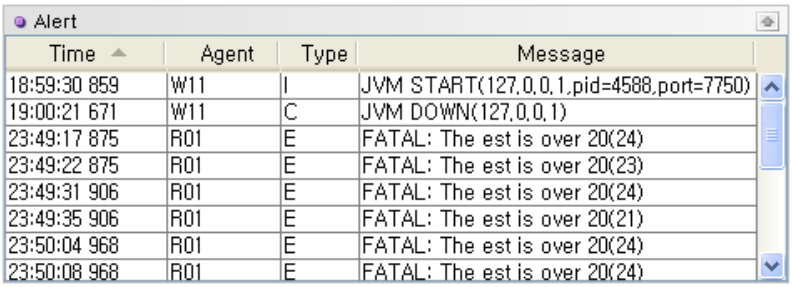
첫번째 파라미터는 조건식으로 변수와 연산자를 사용할 수 있다. 변수로는 데이터 필드 이름을 사용하고, 연산자에는 사칙 연산자(+, -, *, /), 부울 연산자(&&, ||), 괄호, 삼항 연산자(? :) 등을 사용할 수 있다.

두번째 파라미터는 경보 메시지로 FATAL, ERROR, WARN 중의 하나로 시작해야 한다. 이 구분에 따라서 경보 유형이 다음과 같이 결정된다.

- FATAL - 제니퍼 서버에서 USER_DEFINED_FATAL 경보가 발령된다.
- ERROR - 제니퍼 서버에서 USER_DEFINED_ERROR 경보가 발령된다.
- WARN - 제니퍼 서버에서 USER_DEFINED_WARNING 경보가 발령된다.

메시지에는 데이타 필드의 특정 값을 포함시킬 수 있다. ${필드 이름} 형식으로 기술하면 메시지에서 해당 영역은 데이터 값으로 치환되서 서버로 전달된다.

그림 15-5: 경보 차트



REMON에서 발령한 경보도 다른 경보와 같이 제니퍼 서버의 ALERT_01~31 테이블에 저장된다. 그래서 필요에 따라서 보고서 템플릿과 쿼리 수행기로 과거 데이터를 조회할 수 있다.

동일한 경보가 동일한 REMON 데이터에서 연속으로 발생하면 첫번째 경보만이 제니퍼 서버에 전달되고 나머지는 모두 무시된다. 만약 모든 경보를 매번 전달하려면 ALERT_ALWAYS 필터를 사용한다.

```
REMON> addf NET ALERT_ALWAYS
...
REMON> appf NET
...
```

15.5.2.새로운 필터 추가

사용자가 새로운 필터를 작성하여 사용할 수 있다.

15.5.2.1. 필터 클래스 구현

우선 remon.IFilter 인터페이스를 구현한 클래스를 작성해야 한다. 일반적으로 remon.IFilter 인터페이스를 구현한 remon.AbstractFilter 클래스를 상속해서 새로운 필터를 구현한다.

Notice: 컴파일을 하려면 REMON_HOME/lib/sys/remon.jar 파일을 클래스 패스에 등록해야 한다. 그리고 컴파일한 클래스를 JAR 파일로 패키지하여 REMON_HOME/lib/script 디렉토리에 복사한 후에 REMON을 재시작한다. 클래스를 수정하면 앞의 작업을 반복한 후에 REMON을 재시작한다.

remon.IFilter 인터페이스는 다음과 같다.

```
package remon;


public interface IFilter {
    public void open(String script, String[] args);
    public RmPacket process(RmPacket pack);
    public void close();
}
```

remon.AbstractFilter 클래스는 다음과 같다

```
package remon;

import com.javaservice.jennifer.protocol.RmPacket;

abstract public class AbstractFilter implements IFilter {

    public void open(String script, String[] args) {
    }

    public void close() {
    }

    public RmPacket process(RmPacket rmdata) {
        return null;
    }

}
```

기본적으로 open, close, process 메소드를 구현해야 한다. 각 메소드는 다음과 같은 특징을 갖는다.

- open - 필터가 로드될 때 호출된다.
- close - 필터가 제거될 때 호출된다.
- process - REMON 데이터가 수집될 때마다 호출된다.

필터는 각 REMON 스크립트별로 적용된다. 따라서 REMON 스크립트에 필터가 처음 적용될 때 open 메소드가 호출된다. 이 메소드에서 필터에 대한 초기화 작업을 수행한다. open 메소드의 첫번째 파라미터는 java.lang.String 유형으로 스크립트 아이디이다. 그리고 두번째 파라미터는 java.lang.String[] 유형으로 제어 콘솔에서 addf 명령어로 필터를 추가할 때 설정하는 파라미터 목록을 나타낸다. 예를 들어, 다음과 같이 필터를 추가한다고 가정하자.

```
REMON> addf NET DOUBLE 10 true
```

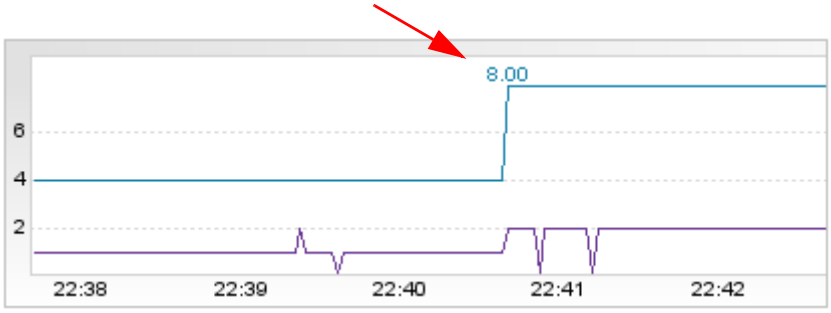
```
// script는 NET
public void open(String script, String[] args) {
    String second = args[0]; // 10
    String isOrder = args[1]; // true
}
```

그리고 close 메소드는 필터가 제거될 때 호출된다. 따라서 close 메소드에서는 open 메소드나 process 메소드에서 할당한 자원을 해제하는 로직을 구현한다. 그리고 실제 데이터의 가공은 process 메소드에서 구현한다. 이 메소드는 REMON 데이터가 수집될 때마다 호출된다. 만약 이 메소드가 null을 반환하면 다음 필터나 제니퍼 서버로 REMON 데이터가 전달되지 않는다. 원본 REMON 데이터는 process 메소드의 RmPacket 클래스 유형의 파라미터로 전달된다. 예를 들어, 데이터의 값을 2배로 증가시키는 필터는 다음과 같이 구현한다.

```
package example;

import remon.AbstractFilter;

import com.javaservice.jennifer.protocol.RmPacket;
import com.javaservice.jennifer.type.FIELD;
import com.javaservice.jennifer.type.INT;
import com.javaservice.jennifer.type.TYPE;

public class DoubleFilter extends AbstractFilter {

    public RmPacket process(RmPacket pack) {
        int count = pack.count();
        for (int i = 0; i < count; i++) {
            FIELD field = pack.getField(i);
            TYPE type = field.getValue();
            if (type.getType() == TYPE.INT) {
                INT t = (INT) type;
                t.value *= 2;
            }
        }
        return pack;
    }

}
```

다음 차트는 DOUBLE 필터를 적용하기 전과 적용한 이후의 데이터의 변화를 보여준다.

그림 15-6: DOUBLE 필터 적용 예



RmPacket 클래스의 자세한 사용 방법은 [RmPacket 클래스의 사용(623 페이지)]을 참조한다.

15.5.2.2. 필터 등록

새로운 필터를 등록하는 방법은 다음과 같다.

1. 우선 remon.IFilter 인터페이스를 구현한 클래스를 JAR 파일로 패키지하여 REMON_HOME/lib/script 디렉토리에 복사한다.
2. 그리고 REMON_HOME/conf/filter.conf 파일에 필터 정보를 설정한다.

```
[DOUBLE]
    class = example.DoubleFilter
    option =
    desc = Make the value into a double.
```

[] 사이에 있는 DOUBLE이 필터 이름이 된다.

그리고 class 속성으로 remon.IFilter 인터페이스를 구현한 클래스를 설정한다. option 속성에는 필터가 사용하는 파라미터에 대한 설명을 기술하고, desc 속성에는 필터 자체에 대한 설명을 기술한다.

1. REMON을 재시작한다.

2. 제어 콘솔에서 filter 명령어로 필터가 올바르게 등록되었는지를 확인한다.

```
REMON> filter DOUBLE
Available Filters
------------------
[DOUBLE]
    class = example.DoubleFilter
    option =
    desc = Make the value into a double.
```

15.6. 데이터 관리 및 뷰

REMON은 수집한 REMON 데이터를 제니퍼 서버에 전송한다. 제니퍼 서버에서 REMON 데이터를 관리하고 확인하는 방법을 설명한다.

15.6.1.제니퍼 서버에서 REMON 데이터 디버깅

제니퍼 서버 시작 콘솔에서 REMON이 전송하는 REMON 데이터를 확인하려면 제니퍼 서버의 remon_debug 옵션을 true로 설정한다. 기본 값은 false이다.

```
debug_remon = true
```

15.6.2.데이터 저장

REMON데이터는 제니퍼서버에서 파일과 DB에 저장될 수 있다.

15.6.2.1. 데이터베이스 저장

DB_SAVE 필터가 적용된 REMON 스크립트가 전송하는 데이터는 파일과 데이터베이스에 동시에 저장한다. 단, 제니퍼 서버의 enable_remon_db_save 옵션을 false로 설정하면 데이터를 저장하지 않는다. 기본 값은 true이다.

REMON 데이터가 저장되는 테이블의 이름은 다음과 같다.

```
RM_<스크립트_아이디>_01~31
```

예를 들어, 스크립트 아이디가 CPU인 REMON 스크립트가 전송하는 데이터는 RM_CPU_01~31 테이블에 일자별로 저장된다.

REMON 데이터를 저장하는 모든 테이블에는 다음 칼럼이 존재한다.

표 15-8: REMON 테이블

칼럼	유형	제약 조건	설명
LOG_TIME	BIGINT		기록 시간
SCRIPT	VARCHAR(256)		스크립트 아이디
AGENT	VARCHAR(128)		에이전트 아이디
LOG_DT	CHAR(8)		날짜(YYYYMMDD)
LOG_HH	CHAR(2)		시간(HH)
LOG_MM	CHAR(2)		분(MM)
LOG_SS	CHAR(2)		초(SS)
MESSAGE	VARCHAR(32672)		REMON 메시지

그리고 REMON 데이터의 각 필드 이름으로 칼럼이 추가된다. 필드 유형에 따라서 칼럼 유형도 결정된다.

Notice: 필드 유형이 변경되면 REMON 데이터가 정상적으로 저장되지 않는다. 따라서 이 경우에는 REMON 스크립트 아이디를 수정하는 것을 권장한다.

15.6.2.2. 파일 저장

FILE_SAVE 필터가 적용된 REMON 스크립트가 전송하는 데이터는 파일과 데이터베이스에 동시에 저장한다. 단, 제니퍼 서버의 enable_remon_file_save 옵션을 false로 설정하면 데이터를 저장하지 않는다. 기본 값은 true이다.

Notice: 저장된 데이터에 대한 조회는 [TOOLS | REMON Data Serarch]를 참조한다..

15.6.3.데이터 삭제

제니퍼 서버는 데이터베이스에 저장된 REMON 데이터를 CleanerActor 스케줄러로 자동으로 삭제한다. 기본적으로 제니퍼 서버에 다음과 같이 설정되어 있다.

```
time_actor_11 = com.javaservice.jennifer.server.timeactor.CleanerActor
02 REMON DAY 7
```

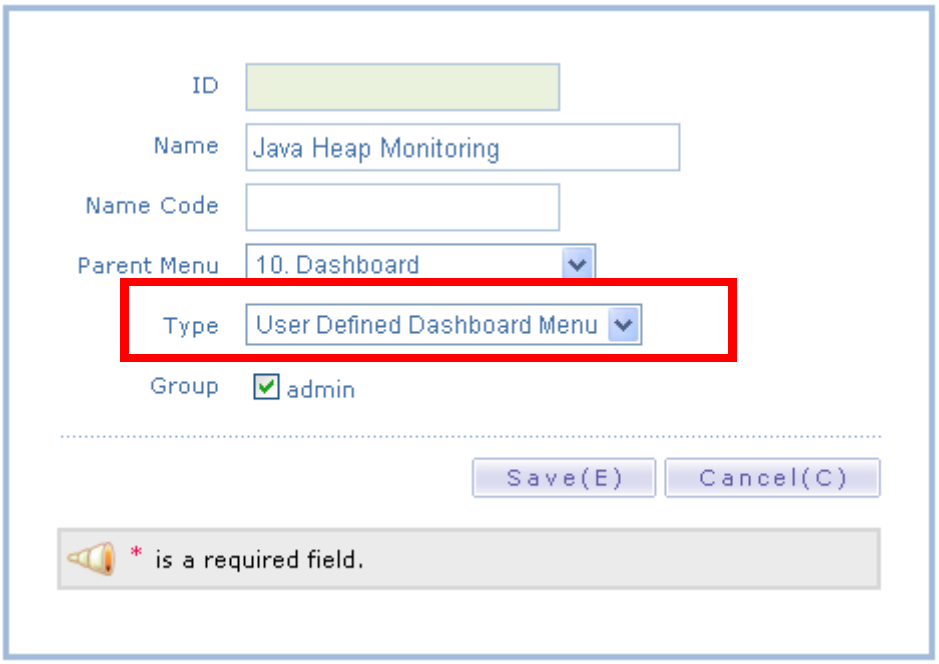
기본 설정에 따르면 7일이 지난 REMON 데이터는 매일 새벽 2시에 자동으로 삭제된다.

15.6.4.사용자 정의 대시보드를 통한 REMON 데이터 모니터링

사용자 정의 대시보드를 통해서 REMON 데이터를 차트로 모니터링할 수 있다. 사용자 정의 대시보드에 대한 자세한 사항은 [사용자 정의 대시보드(62 페이지)]를 참조한다.

우선 [구성 관리 | 메뉴 관리] 메뉴에서 사용자 정의 대시보드 유형의 메뉴를 추가한다.

그림 15-7: 사용자 정의 대시보드 유형의 메뉴 추가



메뉴를 추가한 후에 해당 메뉴로 이동한다. 그리고 화면 오른쪽에 있는 **[편집]** 버튼을 클릭하여 사용자 정의 대시보드 편집 화면으로 이동한다.

그림 15-8: 사용자 정의 대시보드 화면

차트 선택 영역의 [사용자 정의 차트] 그룹에서 LINE 차트를 선택한다. 그러면 화면에 차트가 나타난다. 이 차트를 드래그 앤 드랍으로 차트 배치 영역에 놓는다.

그림 15-9: LINE 차트 선택

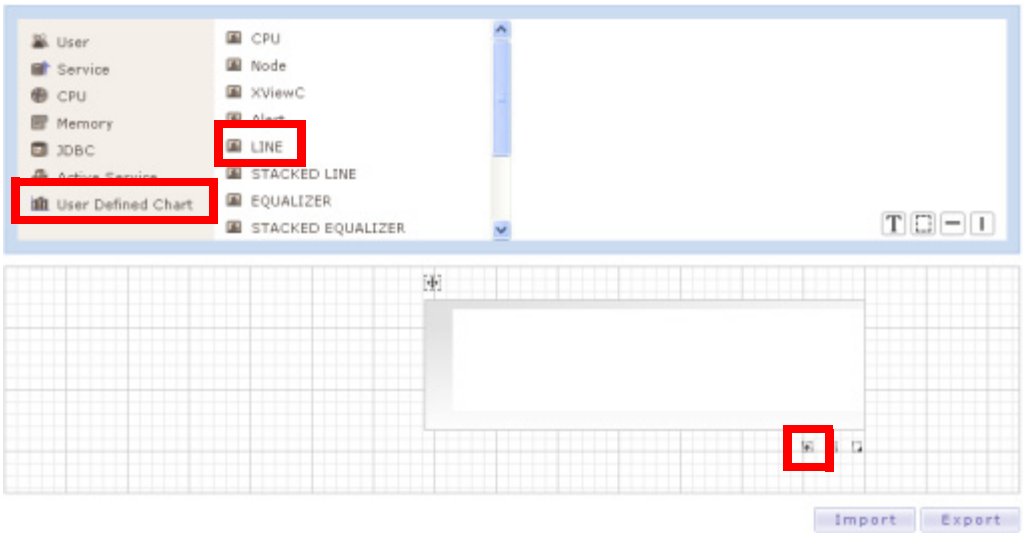
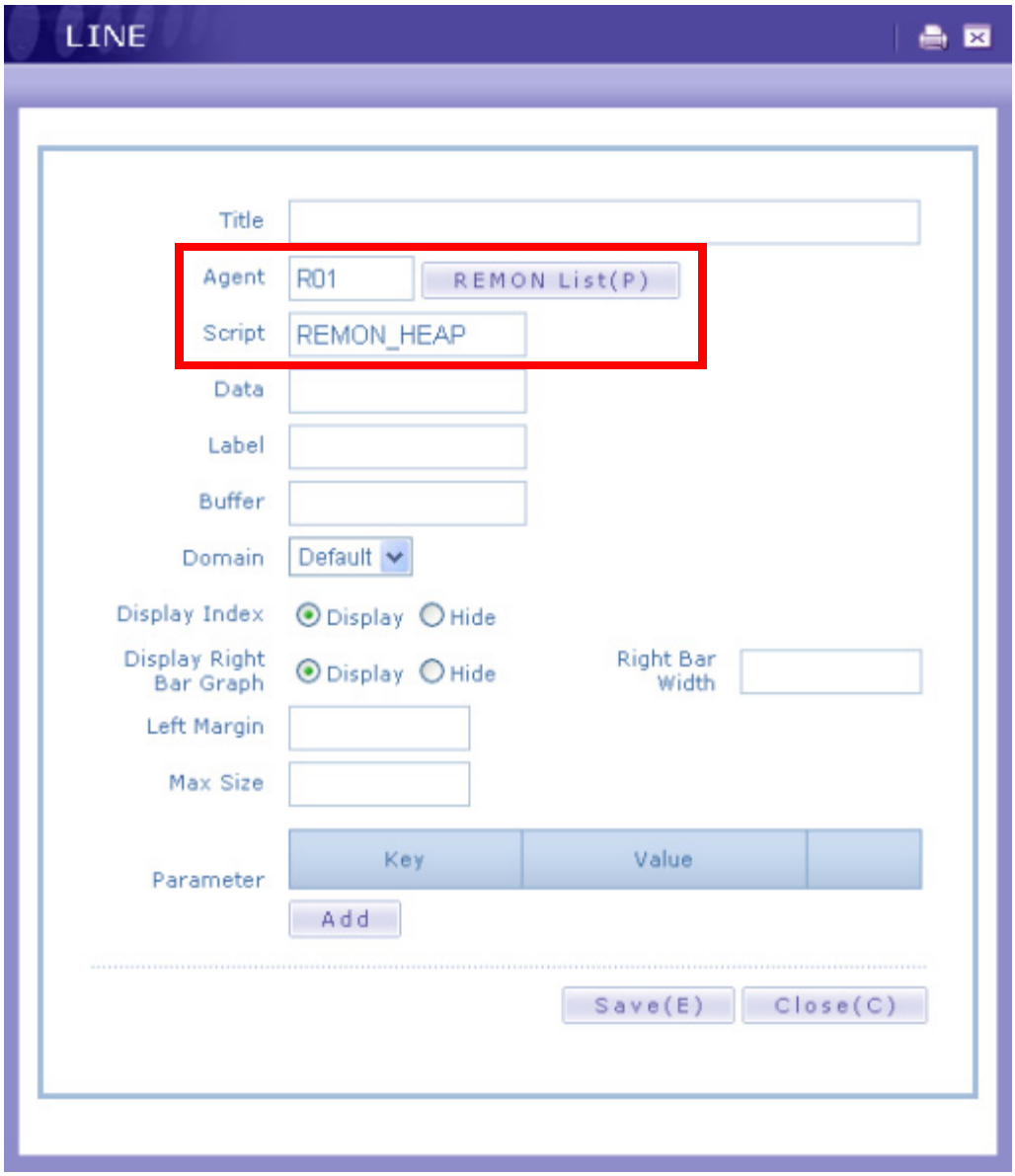


차트 배치 영역에 놓은 차트의 오른쪽 하단에 있는 **[옵션 설정]** 아이콘을 클릭하면 옵션 설정 팝업 창이 나타난다.

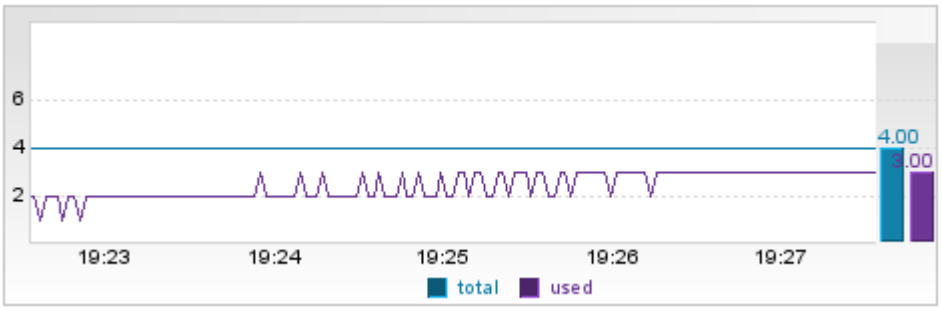
그림 15-10:옵션 설정 팝업 창



옵션 설정 팝업 창에서 **[REMON 목록]** 버튼을 클릭한다. 그리고 모니터링할 REMON 데이터를 선택한 후에 하단에 있는 **[선택]** 버튼을 클릭한다. 그러면 에이전트와 스크립트 필드에 REMON 데이터의 에이전트 아이디와 스크립트 아이디가 입력된다.

그리고 옵션 설정 팝업 창 하단에 있는 **[저장]** 버튼을 클릭하면 LINE 차트에 REMON 데이터가 표시된다.

그림 15-11:LINE 차트로 REMON 데이터 모니터링



15.7. LogWatcher

LogWatcher는 로그 감시기의 역할을 수행한다. 다양한 애플리케이션이나 운영 체계가 기록하는 텍스트 로그 파일에서 임의로 지정한 패턴을 검출하고 이벤트를 발생시켜서 관련 데이터를 REMON에 전달한다.

15.7.1.설치와 구성

LogWatcher는 REMON_HOME/logwatcher 디렉토리에 존재하며, log.sh, logwatcher.jar, logw.conf 등으로 구성되어 있다. 설치를 하려면 우선 logwatcher 디렉토리를 모니터링이 필요한 운영 체계에 복사한다.

그리고 logw.conf 파일에 주요 설정을 한다. 우선 logw.conf 파일 앞단에서 공통 옵션을 설정한다. 이 옵션들은 [XXX]로 시작하는 로그 파일 설정보다 앞에 놓여야 한다.

주요 옵션은 다음과 같다.

```
agent = L01
control_port = 6001
target_host = 127.0.0.1
target_port = 7701
```

다음은 공통 옵션에 대한 설명이다.

- agent - LogWatcher로 수집하는 REMON 데이터의 에이전트 아이디를 설정한다. REMON의 에이전트 아이디와 동일한 제약 조건을 갖는다.
- control_port - LogWatcher도 제어 콘솔로 관리한다. 이 옵션으로 제어 콘솔의 요청을 받아들이는 TCP 포트 번호를 설정한다.

- target_host - LogWatcher로 수집한 REMON 데이터를 전송할 REMON의 IP 주소를 설정한다.
- target_port - LogWatcher로 수집한 REMON 데이터를 전송할 REMON의 포트 번호를 설정한다.

그리고 LogWatcher로 모니터링할 로그 파일들에 대한 설정도 logw.conf 파일에 한다. 각각의 로그 파일 설정은 [XXX]로 시작하는데 XXX에는 대문자와 숫자로 구성된 임의의 값을 입력한다. 이는 LogWatcher로 수집하는 REMON 데이터의 스크립트 아이디가 된다. REMON의 스크립트 아이디와 동일한 제약 조건을 갖는다. 다음은 ALOG, BLOG라는 이름으로 2개의 로그 파일 모니터링을 설정한 예제이다.

```
[ALOG]
...

[BLOG]
...
```

다음은 ALOG에 대한 상세 설정이다.

```
[ALOG]
file = /tmp/${today}.log
rule_01 = [*]ABC, ok
rule_02 = [4]###, fatal
```

- file - 모니터링할 파일 경로를 설정한다. 날짜에 따라서 파일 이름이 달라지는 경우에는 ${today} 변수를 사용한다.
- rule_nn - rule_01, rule_02 등의 옵션으로 룰을 설정한다. 각각의 룰은 [,]를 구분자로 구분되는데 첫번째 값은 텍스트 검사 조건이고, 두번째 값은 이 룰에 대한 이름으로 REMON 데이터의 필드 이름으로 사용된다. 따라서 앞의 예제로 수집한 REMON 데이터에는 2개의 필드(rule_01에 대한 ok 필드, rule_02에 대한 fatal 필드)가 존재한다.

룰을 적용하지 않고 로그 라인을 그대로 REMON에 전송하려면 send_raw_data 옵션을 true로 설정한다.

```
[BLOG]
file = /tmp/access.log
send_raw_data = true
```

logw.conf 파일을 변경하면 반드시 LogWatcher를 재시작해야 한다.

LogWatcher는 자바로 구현되어 있기 때문에 자바를 JAVA_HOME 환경 변수로 설정한다.

Notice: LogWatcher를 실행하려면 자바 1.4.2 이상을 사용해야 한다.

JAVA_HOME 환경 변수는 REMON_HOME/logwatcher 디렉토리의 env.sh(env.bat) 파일에서 설정한다. 유닉스 혹은 리눅스의 경우에는 다음과 같이 수정한다.

```
JAVA_HOME=/usr/java
```

윈도우의 경우에는 다음과 같이 수정한다.

```
set JAVA_HOME=C:/Program Files/Java/jdk1.6.0_07
```

LogWatcher를 실행하려면 REMON_HOME/logwatcher 디렉토리의 logw.sh를 실행한다.

```
./logw.sh
```

LogWatcher를 정지하려면 REMON_HOME/logwatcher 디렉토리의 shutdown.sh를 실행한다.

```
./shutdown.sh
```

제어 콘솔로 LogWatcher를 관리할 수 있다. 제어 콘솔을 시작하려면 REMON_HOME/logwatcher 디렉토리의 console.sh를 실행한다.

```
./console.sh
```

모니터링 현황을 확인하려면 제어 콘솔에서 ls 명령어를 입력한다.

```
logw> ls
```

15.7.2.LogWatcher와 REMON 데이터

LogWatcher는 줄 단위로 로그 파일을 읽어서 모든 룰에 대한 매치 여부를 확인한다. 그리고 각 룰에 대한 매치 횟수를 카운팅하여 REMON 데이터를 구성한다. 따라서 각 룰은 필드가 되고, 룰의 매치 회수는 필드에 대한 값이 된다.

그리고 5초 주기로 이 REMON 데이터를 REMON에 전송한다.

Notice: send_raw_data 속성을 true로 지정한 경우에는 1초마다 데이터를 전송한다.

예를 들어, ALOG에 대해서 5초 동안 쌓인 로그가 100 줄이 있다고 가정하자. 그리고 10개의 줄이 rule_01에 부합하고, 5개의 줄이 rule_02에 부합하면 다음 데이터가 REMON에 전송된다.

```
에이전트 아이디 - L01
스크립트 아이디 - ALOG
ok 필드 값 - 10
fatal 필드 값 - 5
```

15.7.3.룰 설정

룰은 [검사 조건] 과 [필드 이름]으로 구성된다. 검사 조건은 [단어 위치] 를 지정하는 키워드와 검출할 단어로 구성된다. 그리고 여러 개의 조건을 조합하기 위해서 && 혹은 || 연산자를 검사 조건에 사용할 수 있다. 그러나 하나의 룰에 &&과 ||를 동시에 사용할 수는 없다.

```
[*] - any position
[n] - n's position (n is any positive number like 0, 1... 999)
[$] - end of the line
```

다음은 설정 가능한 검사 조건 예제이다. 모든 룰은 줄 단위로 적용된다.

[*]ABC - 임의의 위치에 ABC 문자열이 존재하면 룰에 부합한다.

[4]BBB - 5번째 글자에서 시작하는 BBB 문자열이 존재하면 룰에 부합한다.

[$]XXX - 줄이 XXX 문자열로 끝나면 룰에 부합한다.

[*]AAA || [$]XXX - 임의의 위치에 AAA 문자열이 존재하거나 줄이 XXX 문자열로 끝나면 룰에 부합한다.

Notice: [*]AAA || [$]XXX && [*]BB와 같이 &&와 ||를 혼용해서 사용할 수 없다.

15.7.4.아파치 웹 서버 액세스 로그를 X-ViewC 차트로 모니터링하기

아파치 웹 서버 액세스 로그를 LogWatcher와 REMON과 X-ViewC 차트로 모니터링하는 방법을 설명한다.

Notice: X-ViewC 차트는 포멧이 일정한 로그 파일을 분포도 형태로 보여주는 차트이다. 로그의 각 라인에는 시간, 값(응답 시간 혹은 사용량 등), 그리고 이름(URL 등)이 포함되어 있어야 한다. 자바 GC 로그, 아파치 웹 서버 액세스 로그 등을 X-ViewC 차트로 모니터링할 수 있다.

우선 LogWatcher로 아파치 웹 서버 액세스 로그 파일을 읽는다. 아파치 웹 서버가 설치된 서버에 LogWatcher를 설치하고 다음과 같이 logw.conf 파일을 설정한다.

```
agent = L01
control_port = 6001
target_host = 127.0.0.1
target_port = 7701

[ACCESS]
send_raw_data = true
file = /usr/local/apache2/logs/www_access_log
```

X-ViewC 차트로 모니터링하려면 에이전트 아이디는 3자로 설정해야 한다.

LogWatcher와 REMON을 실행한다. 그리고 REMON 제어 콘솔에서 LogWatcher가 보내는 REMON 데이터에 XVIEW_APACHE 필터를 적용한다

```
REMON> addf ACCESS XVIEW_APACHE
...
REMON> appf ACCESS
...
```

기본적으로 X-ViewC 차트의 Y축은 액세스 바이트이다. Y축을 응답 시간으로 하려면 다음과 같이 필터를 적용한다. .

```
REMON> addf ACCESS XVIEW_APACHE 2
...
REMON> appf ACCESS
...
```

Notice: 단, 아파치 웹 서버 액세스 로그 파일에 응답 시간도 기록되어야 한다. 설정 방법은 아파치 웹 서버 매뉴얼을 참고한다.

마지막으로 제니퍼 서버의 사용자 정의 대시보드 편집 화면으로 이동한다. 차트 선택 영역의 [사용자 정의 차트] 그룹에서 X-ViewC 차트를 선택한다. 그러면 화면에 차트가 나타난다. 이 차트를 드래그 앤 드랍으로 차트 배치 영역에 놓는다.

그림 15-12:X-ViewC 차트

15.8. RmPacket 클래스의 사용

클래스 스크립트, 필터, ExtraAgent 등을 구현하려면 RmPacket 클래스를 이해해야 한다. 이를 위해서 RmPacket 클래스를 사용하는 방법을 설명한다.

패키지 이름을 포함한 RmPacket 클래스의 이름은 다음과 같다.

```
com.javaservice.jennifer.protocol.RmPacket
```

RmPacket은 com.javaservice.jennifer.protocol.IRmPacket 인터페이스를 구현한 클래스로 REMON 데이터를 나타낸다. IRmPacket 인터페이스가 제공하는 메소드를 통해서 REMON 데이터의 기본 정보를 확인할 수 있다.

```
package com.javaservice.jennifer.protocol;


public interface IRmPacket {
    public String getAgent();
    public String getScript();
    public long getTime();
    public byte[] toBytes() throws Exception;
}
```

getAgent 메소드는 REMON 데이터의 에이전트 아이디를 반환하고, getScript 메소드는 REMON 데이터의 스크립트 아이디를 반환한다. 그리고 getTime 메소드는 REMON 데이터가 생성된 시간을 반환한다. 마지막으로 toBytes 메소드는 REMON 데이터를 바이트 형식의 배열로 반환한다. 클래스 스크립트, 필터, ExtraAgent 등을 구현할 때 toBytes 메소드를 사용해야 하는 경우는 없다.

REMON 데이터를 구성하고 읽는 작업은 RmPacket 클래스가 제공하는 메소드로 수행한다. 먼저 클래스 스크립트와 ExtraAgent 등에서 새로운 RmPacket 객체를 만들어서 REMON 데이터를 수집하는 방법을 설명한다.

RmPacket 객체를 생성하는 첫번째 방법은 다음과 같다. 스크립트 아이디와 에이전트 아이디를 파라미터로 생성자를 수행한다.

```
RmPacket packet = new RmPacket("HEAP", "R01");
```

이 경우에는 REMON 데이터를 수집한 시간은 현재 시간이 된다. 다음과 같은 방식으로 REMON 데이터를 수집한 시간을 지정할 수 있다.

```
long time = ...
RmPacket packet = new RmPacket(time, "HEAP", "R01");
```

REMON 데이터는 필드로 구성된다. 따라서 각 필드는 RmPacket 클래스의 addField 메소드로 추가한다. addField 메소드의 첫번째 파라미터는 java.lang.String 유형으로 필드 이름을 의미한다. 그리고 두번째 파라미터는 com.javaservice.jennifer.type.TYPE 유형으로 필드 데이터 유형에 맞는 하위 클래스를 사용한다.

```
RmPacket packet = ...
packet.addField("max", new INT(4));
```

INT 클래스는 int 유형의 데이터를 추가할 때 사용한다. 다음은 TYPE 클래스의 하위 클래스에 대한 설명이다.

표 15-9: TYPE 클래스

클래스	설명	데이터베이스 칼럼 유형
BTYE	1개의 바이트 데이터	이 유형의 필드는 데이터베이스에 저장되지 않는다.
BYTES	32,767개의 바이트 데이터(Short 최대값)	이 유형의 필드는 데이터베이스에 저장되지 않는다.
TINY_BYTES	256개의 바이트 데이터	이 유형의 필드는 데이터베이스에 저장되지 않는다.
LONG_BYTES	2,147,483,647개의 바이트 데이터(Integer 최대 값)	이 유형의 필드는 데이터베이스에 저장되지 않는다.
SHORT	short 형의 데이터	INT
INT	int 형의 데이터	INT
LONG	long 형의 데이터	INT
FLOAT	float 형의 데이터	REAL
DOUBLE	double 형의 데이터	REAL
STRING	java.lang.String 형의 데이터로, 가능한 최대 글자수는 32,672이다.	VARCHAR(32672)
TINY_STRING	java.lang.String 형의 데이터로, 가능한 최대 글자수는 256이다.	VARCHAR(257)
LONG_STRING	java.lang.String 형의 데이터로, 가능한 최대 글자수는 2,147,483,647이다.	이 유형의 필드는 데이터베이스에 저장되지 않는다.

각 TYPE 클래스는 생성자를 통해서 값을 할당한다.

```
new INT(4);
new FLOAT(3.2f);
new STRING( “WARNING” );
```

그리고 필터를 구현하는 경우와 같이 전달받은 RmPacket 객체로 REMON 데이터의 내용을 확인하고 값을 수정하는 방법은 다음과 같다. 우선 RmPacket 클래스의 count 메소드로 필드의 개수를 파악할 수 있다.

```
int count = packet.count();
```

그리고 인덱스를 파라미터로하는 getField 메소드로 필드를 나타내는 FIELD 객체를 획득한다.

```
int count = packet.count();
for (int i = 0; i < count; i++) {
    FIELD field = packet.getField(i);
}
```

FIELD 객체의 getName 메소드로 필드 이름을 확인할 수 있고, getValue 메소드로 값을 나타내는 TYPE 객체를 획득할 수 있다.

```
for (int i = 0; i < count; i++) {
    FIELD field = packet.getField(i);
    String name = field.getName();
    TYPE type = field.getValue():
}
```

TYPE 객체의 실제 유형은 TYPE 객체의 getType 메소드가 반환하는 값과 TYPE 클래스의 상수를 비교해서 파악한다. 상수의 이름은 TYPE 하위 클래스의 이름과 동일하다.

```
TYPE type = field.getValue();
switch (type.getType()) {
case TYPE.INT:
    INT t = (INT) type;
    ...
    break.
case TYPE.STRING:
    STRING t = (STRING) type;
    ...
    break;
}
```

각 TYPE 객체의 값을 확인하려면 value 멤버 필드를 사용한다. 이 필드의 유형은 TYPE 하위 클래스에 따라서 상이하다.

```
INT t = (INT) type;
int value = t.value;
```

해당 값을 변경하는 경우에도 TYPE 유형의 value 멤버 필드를 직접 수정한다.

```
TYPE type = field.getValue();
if (type.getType() == TYPE.INT) {
    INT t = (INT) type;
    t.value *= 2;
}
```

16

APPENDIX

16.1. 일러두기

- 각 옵션에서 사용되는 유효한 VALUE를 설명하는 도메인 키워드와 기술방법은 다음과 같다

도메인	표기 설명
CLASS	패키지명을 포함한 full class name을 기술한다.
METHOD	메소드 명만 기술한다.
CLASS_METHOD	Full class name와 메소드명을 연결하여 기술한다. ex) java.lang.Thread.sleep
CLASS_METHOD_PARAM	Full class name과 파라미터 를 포함한 메소드명을 기술한다. ex) java.lang.Thread.sleep(long)
CLASS_FIELD	Full class name와 필드명을 연결하여 기술한다. ex) java.lang.Thread.MAX_PRIORITY
DIRPATH	디렉토리 경로를 기술한다.
JARPATH	JAR파일 경로를 기술한다.
SQL	SQL문장을 기술한다.

도메인	표기 설명
BOOL	true or false만을 기술한다.
MILLISECOND	밀리세컨드단위의 시간값을 기술한다.
SECOND	초단위 시간값을 기술한다.
DATE	일단위 시간값을 기술한다.
FILEPATH	파일 경로를 기술한다.
STRING	문자열을 기술한다.
INT	정수형 숫자를 기술한다
HOST	IP주소나 hostname을 기술한다.
PORT	유효한 네트웍 포트값을 기술한다.
CHAR	한 문자를 기술한다.
AGENT	제니퍼 에이전트명(3자, 영문,숫자)을 기술한다.

- “default:”의 의미는 해당 옵션이 설정파일에 기술되지 않았을 경우 제니퍼가 내부에서 사용하는 기본값을 나타낸다.
- 각 옵션 설명의 마지막에 나오는 샘플 설정은 기본값이거나 권장값 혹은 단순 예제이다.
- 옵션에 대한 충분한 지식이 없는 경우에 임의로 변경하는 것을 삼가해야 한다.

16.2. 제니퍼 에이전트 옵션

16.2.1.agent_fileroot

제니퍼 에이전트가 사용하는 임시 디렉토리를 설정한다. 설정된 값은 존재하고 WRITE가 가능해야 한다.

- default: .
- value: DIRPATH
- agent_fileroot=/jennifer/agent/data

16.2.2.active_graph_interval

제니퍼는 경과 시간에 따라서 액티브 서비스를 4 개의 그룹으로 분리한다. 이를 기초로 액티브 서비스 관련 차트의 각 구간 별 색상을 다르게 줄 수 있다. active_graph_interval에 대해서 콤마를 구분자로 밀리세컨드 단위의 값을 입력한다. 만약 active_graph_interval에 대해서 0,1000,3000,8000을 입력하면 액티브 서비스를 0 ~ 1 초, 1 ~ 3 초, 3 ~ 8 초, 그리고 8 초 이상으로 구분한다는 뜻이다.

- default: 0,1000,3000,8000
- value: MILLISECOND,MILLISECOND,MILLISECOND,MILLISECOND(오름차순 정렬,4개 숫자)
- active_graph_interval = 0,1000,3000,8000

Notice: 본 옵션은 제니퍼 서버에도 존재하며 양쪽의 값이 일치해야 한다.

16.2.3.active_param_access_method

ACTIVE PARAMETER TRACING에 포함된 클래스에서 대상 메소드의 접근 권한을 지정한다.

- default: public,protected,private,none
- value:(public|protected|private|none){,(public|protected|private|none)}*
- active_param_access_method=public,protected,private,none

16.2.4.active_param_class

ACTIVE PARAMETER TRACING를 위한 대상 클래스를 지정한다.

- default: NONE
- value: CLASS{;CLASS}*
- active_param_class=com.TextA

Notice: ACTIVE PARAMETER TRACING: 실시간 액티브 서비스 모니터링 시 중요 로직의 파라미터를 보다 적은 부하로 모니터링 하기 위해서 사용한다. 예를 들어, 검색 키워드 같은 정보를 서비스 PROFILING에 설정하지 않고 서비스 큐잉이 발생할 때만 모니터링하고자 한다면 이 기능을 사용한다.

16.2.5.active_param_ignore_method

ACTIVE PARAMETER TRACING에 포함된 클래스에서 제외할 메소드만을 지정한다.메소드명 혹은 클래스+메소드를 지정할 수 있다.

- default: NONE
- value:(CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
- active_param_ignore_method=access;com.Test.save

16.2.6.active_param_ignore_prefix

ACTIVE PARAMETER TRACING에서 제외할 클래스들의 공통 PREFIX를 지정한다.

- default: NONE
- value: STRING{;STRING}*
- active_param_ignore_prefix=com.util;com.jennifersoft.util

16.2.7.active_param_interface

ACTIVE PARAMETER TRACING를 위한 대상 클래스들의 공통 인터페이스 클래스를 지정한다.

- default: NONE
- value:CLASS{;CLASS}*
- active_param_interface=com.ITestClassA;org.ITestClassB

16.2.8.active_param_postfix

ACTIVE PARAMETER TRACING를 위한 대상 클래스들의 공통 POSTFIX를 지정한다.

- default: NONE
- value: STRING{;STRING}*
- active_param_postfix=Adapeter;VO;Action

16.2.9.active_param_prefix

ACTIVE PARAMETER TRACING를 위한 대상 클래스들의 공통 PREFIX를 지정한다.

- default: NONE
- value: STRING{;STRING}*
- active_param_prefix=com;org.apache.common

16.2.10.active_param_super

ACTIVE PARAMETER TRACING를 위한 대상클래스들의 공통 SUPER클래스를 지정한다.

- default: NONE
- value:CLASS{;CLASS}*
- active_param_super=com.TestSuperA;com.TestSuperB

16.2.11.active_param_target_method

ACTIVE PARAMETER TRACING에 포함된 클래스에서 대상 메소드만을 지정한다. 메소드명 혹은 클래스+메소드를 지정할 수 있다.

- default: NONE
- value:(CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
- active_param_target_method=access;com.Test.save

16.2.12.active_param_type

ACTIVE PARAMETER TRACING의 대상 메소드 파라미터 타입을 지정한다.

Notice: java.lang.String과 byte[]만 가능하다.byte[]은 String으로 변환 가능해야 한다.

- default: java.lang.String
- value: java.lang.Sring|byte[]
- active_param_type=java.lang.String

16.2.13.agent_boot_class

제니퍼 에이전트 모듈이 초기화 되기 위한 클래스를 지정한다. 이 클래스가 지정되지 않으면 HttpServlet과 같은 서비스 시작 클래스가 호출될 때 초기화 된다.

```
default: NONE
value: CLASS
agent_boot_class=javax.servlet.GenericServlet
```

16.2.14.agent_db_enabled

제니퍼 에이전트에서 사용하는 Simple ISAM DB모듈이 포함되어 있다. 이것을 활성화한다.

- default: false
- value: BOOL
- agent_db_enabled = true

16.2.15.agent_db_keep_hours

제니퍼 에이전트 DB(Simple ISAM DB)는 시계열 데이터만을 저장한다. 데이터의 보관 주기를 지정한다.

- default: 24
- value:DATE
- agent_db_keep_hours = 2

16.2.16.agent_db_rootpath

제니퍼 에이전트 DB(Simple ISAM DB)가 사용할 저장 디렉토리를 지정한다.

> **Notice:** full path로 지정하며, 별도의 디렉토리를 지정할 것을 권장한다. 반드시 Writable해야 한다.

```
default: .
value: DIRPATH
agent_db_rootpath =/jennifer/agent/db
```

16.2.17.agent_fileroot

제니퍼 에이전트는 제니퍼 에이전트를 설치한 운영 체계의 특정 디렉토리에 로딩된 클래스나 성능 데이터를 임시로 저장한다. 이 때 사용하는 디렉토리는 제니퍼 에이전트를 설치한 자바 애플리케이션을 수행한 WORKING_DIRECTORY이다. 이 디렉토리에는 .data와 .class라는 서브디렉토리가 만들어 진다. 제니퍼 에이전트 설정에서 디렉토리를 변경할 수 있다. 설정된 디렉토리는 존재하고 액세스 가능해야 한다. 자바(WAS) 프로세스 런타임에 수정할 수는 있으나 일부 데이터가 (변경순간)일시적으로 손실될 수 있다. 자바(WAS) 프로세스가 실행되기 전에 설정할 것을 권고한다.

- default: .
- value: DIRPATH
- agent_fileroot=/jennifer/agent/data

16.2.18.agent_name

WAS에 설치된 제니퍼 에이전트의 고유한 이름이다. 제니퍼 서버는 해당 이름으로 WAS의 인스턴스를 구분한다. 영어+숫자로 반드시 세자로 등록해야 한다.

```
default: NUL
value: AGENT
agent_name = W11
```

16.2.19.agent_tcp_port

제니퍼 서버로부터의 요청을 받아주는 제니퍼 에이전트의 TCP 포트다. 해당 포트를 통하여 SQL, Method, Application, Error 등의 문자열 정보 및 10분 주기의 통계 데이터를 넘겨준다. TCP Socket을 바인딩하므로 물리적인 서버 별로 포트는 고유해야 한다. 해당 포트를 변경하려면 제니퍼 에이전트가 설치된 WAS를 재시작해야 한다.

- default: 7750
- value: PORT
- agent_tcp_port = 7750

16.2.20.alsb_enabled

WebLogic ALSB 추적을 위한 byte code instrumentation을 활성화 한다

- default: true
- value: BOOL
- alsb_enabled=true

16.2.21.app_bad_responsetime

애플리케이션의 응답시간이 해당 옵션 값 이상을 초과한 경우, 로그 파일에 기록되고 제니퍼 에러에 등록된다. 매우 작은 시간을 지정하면 로그 파일의 크기가 커지고 , 에러 건수가 증가하기 때문에 사이트의 성능 특성을 고려하여 적절하게 지정하도록 한다. 단위는 밀리세컨드이다.

- default: 30000
- value: MILLISECOND
- app_bad_responsetime = 30000

16.2.22.approximate_tpmc_on_this_system

모니터링 대상 서버의 하드웨어의 한개 CPU에 대한(machine TPMC / number of CPU) 알려진 tpmC 값을 지정한다. 이를 통해서 애플리케이션이 사용한 CPU 시간을 tpmC 단위로 환산하여 통계 분석의 애플리케이션 메뉴에서 보여준다. 단 tpmC가 비단 CPU 관점에서만의 단위가 아니기 때문에 그 값이 100% 정확하지는 않다.

- default: 0
- value: MILLISECOND
- approximate_tpmc_on_this_system = 30000

```
cpu_sum_tpmc = cpu_sum * approximate_tpmc_on_this_system / 60;
```

16.2.23.class_dump_after_hooking

제니퍼 에이전트는 모든 로딩되는 클래스의 정보를 관리하기 위해 별도의 디렉토리에 보관한다. 이때 제니퍼 로직이 Instrument된 이전의 상태의 클래스로 보관되는데 이 옵션이 true로 설정되면 제니퍼가 추적로직을 삽입한 이후 상태로 보관된다

- default: false
- value: BOOL
- class_dump_after_hooking = false

16.2.24.connection_trace_class

제니퍼가 인식하지 못하는 JDBC드라이버의 Key커넥션을 등록한다. 제니퍼가 JDBC드라이버를 인식하지 못하면 커넥션 상태(IDLE,ALLOCATED,ACTIVE)가 모니터링 되지 않는다.값이 변경되면 WAS가 재기동 되어야 한다.

- default: NONE
- value: CLASS{;CLASS}*
- connection_trace_class=com.ibm.db2.jcc.b.bb

16.2.25.config_refresh_check_interval

해당 옵션 값 간격으로 제니퍼 에이전트 구성파일의 변경사항을 체크한다. 비동기 처리를 통해서 구성파일을 읽고 변경하므로 성능저하는 없다. 단위는 (ms)이다.

- default: 30000
- value: MILLISECOND
- config_refresh_check_interval = 3000

16.2.26.custom_trace_access_method

METHOD CUSTOM TRACING에 포함된 클래스에서 대상 메소드의 접근 권한을 지정한다.

- default: public,protected,private,none
- value:(public|protected|private|none){,(public|protected|private|none)}*
- custom_trace_access_method=public,protected,private,none

16.2.27.custom_trace_adapter_class_name

METHOD CUSTOM TRACING을 위한 어뎁터 클래스를 지정한다.

- default: NONE
- value: CLASS
- custom_trace_adapter_class_name=jennifer.custom.AdapterCustomError

16.2.28.custom_trace_adapter_class_path

METHOD CUSTOM TRACING을 위한 어뎁터 클래스가 포함된 jar파일 패스를 지정한다

- default: NONE
- value: JARPATH{;JARPATH}*
- custom_trace_adapter_class_path=/jennifer/lwst40.custom.jar

16.2.29.custom_trace_class

METHOD CUSTOM TRACING를 위한 클래스를 지정한다.

- default: NONE
- value: CLASS{;CLASS}*
- custom_trace_class=com.TestClassA;com.TestClassB

Notice: METHOD CUSTOM TRACING : 제니퍼는 임의의 메소드에 대한 커스터마이즈된 추적을 위해 어뎁터를 사용할 수있는 구조를 제공한다. 사용자는 추적을 위한 어뎁터를 프로그램하고 이것을 제니퍼에 등록할 수있다.

16.2.30.custom_trace_hotswap

Custom Trace를 위한 Adapter클래스가 변경 시 리로딩 될지를 선택한다.

- default: true
- value: BOOL
- custom_trace_hotswap = false

16.2.31.custom_trace_ignore_method

METHOD CUSTOM TRACING에 포함된 클래스에서 제외할 메소드만을 지정한다.

- default: NONE
- value:(CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
- custom_trace_ignore_method=access;com.Test.save

16.2.32.custom_trace_ignore_prefix

METHOD CUSTOM TRACING에서 제외할 클래스들의 공통 PREFIX를 지정한다.

- default: NONE
- value: STRING{;STRING}*
- custom_trace_ignore_prefix=com.util;com.jennifersoft.util

16.2.33.custom_trace_interface

METHOD CUSTOM TRACING를 위한 대상 클래스들의 공통 인터페이스 클래스를 지정한다.

- default: NONE
- value: CLASS{;CLASS}*
- custom_trace_interface=com.ITestClassA;org.ITestClassB

16.2.34.custom_trace_param_type

METHOD CUSTOM TRACING를 위한 메소드에서 추적하고자 하는 파라미터 타입을 설정한다.객체 타입만 가능하며 기본값은 java.lang.String이다.

- default: java.lang.String
- value: CLASS
- custom_trace_param_type=java.lang.String

단, 모든 파라미터를 추적하고자 하면 “all”로 설정한다.

```
custom_trace_param_type=all
```

16.2.35.custom_trace_postfix

METHOD CUSTOM TRACING를 위한 대상 클래스들의 공통 POSTFIX를 지정한다.

- default: NONE
- value: STRING{,STRING}*

- custom_trace_postfix=Adapeter;VO;Action

16.2.36.custom_trace_prefix

METHOD CUSTOM TRACING를 위한 대상 클래스들의 공통 PREFIX를 지정한다.

- default: NONE
- value: STRING{;STRING}*
- custom_trace_prefix=com;org.apache.common

16.2.37.custom_trace_super

METHOD CUSTOM TRACING를 위한 대상클래스들의 공통 SUPER 클래스를 지정한다.

- default: NONE
- value: CLASS{;CLASS}*
- custom_trace_super=com.TestSuperA;com.TestSuperB

16.2.38.custom_trace_target_method

METHOD CUSTOM TRACING에 포함된 클래스에서 대상 메소드만을 지정한다.

```
default: NONE
value:(CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
custom_trace_target_method=access;com.Test.save
```

16.2.39.dbsession_query

JDBC Connection 연결 상황을 모니터링 할 때, 데이터베이스의 DB 세션 아이디를 추출한다.

1. oracle.jdbc.driver.OracleConnection: Oracle 9i 이하
2. oracle.jdbc.driver.PhysicalConnection: Oracle 10g

- default: NONE
- value: CLASS:SQL{,CLASS:SQL}*

- dbsession_query = oracle.jdbc.driver.PhysicalConnection: select sid from v$session where audsid = (select userenv ('SESSIONID') from dual)

16.2.40.debug_connection_open

DB Connection이 오픈될 때 그 위치를 로깅하도록 설정하는 옵션

Notice: 오버헤드가 큼으로 디버깅 목적으로 잠시만 사용해야 한다.

- default: false
- value: BOOL
- debug_connection_open = true

16.2.41.debug_vendor_wrap

jdbc_vendor_wrap=true인 경우 Vendor Wrap 기능이 잘 동작하는지를 확인하기 위한 옵션이다.

Notice: 단, vendor_wrap_test_sql 값이 설정되어 있어야 한다.

- default: false
- value: BOOL
- debug_vendor_wrap = true

16.2.42.default_cookie_domain

해당옵션의 설정으로 여러 시스템에 동시에 접속한 고유한 사용자를 같은 사용자로 구분할 수 있다. 예를들어, 아래와 같은 몇개의 사내 시스템을 동시에 접속하여 사용한다고 가정한다.

```
http://www.jennifersoft.com
http://sales.jennifersoft.com
http://mail.jennifersoft.com
```

이 사용자를 한명의 동시단말 사용자로 카운팅하려면 해당 옵션에 .jennifersoft.com 으로 설정한다. 해당 옵션 값 맨 앞에 (“.”)를 반드시 넣어야 한다. 만약 이 옵션을 설정하고

http://192.168.0.1의 IP로 접속하면 해당 사용자는 동시단말 사용자 및 방문자 수로 카운팅되지 않는다..

- default: NONE
- value: STRING
- default_cookie_domain = .jennifersoft.com

16.2.43.dump_http_header_url_prefix

Http Header를 로그 파일에 로깅할 URL의 prefix를 지정한다. 운영부하를 감안해서 사용해야 한다.

- default: NONE
- value: STRING{,STRING}*
- dump_http_header_url_prefix = /biz1/board, /biz2/save

16.2.44.dump_http_hide_all

Http Header나 Parameter를 로깅할 때 값을 모두 감추도록 설정한 것이다.("***"로 표시)

- default: true
- value: BOOL
- dump_http_hide_all = false

16.2.45.dump_http_hide_key

Http Header나 Parameter를 로깅할 때 값을 감출 키를 설정한다. 설정된 키에 대한 값은 "***"로 출력된다.

- default: NONE
- value: STRING{,STRING}*
- dump_http_hide_key = sso, password

16.2.46.dump_http_parameter_url_prefix

Http Request 파라미터를 덤프할 URL의 prefix를 지정한다.

- default: NONE
- value: STRING{,STRING}*
- dump_http_header_parameter_prefix = /biz1/board, /biz2/save

16.2.47.dump_trigger_sleep_interval

액티브서비스 목록이 덤프 최소 간격을 설정한다. 빈번한 액티브서비스 덤프는 시스템에 과부하를 유발한다. 단위는 밀리세컨드이다.

`default: 180000`

`value: MILLISECOND`

dump_trigger_sleep_interval = 180000

16.2.48.enable

해당 옵션이 true 인 경우에만 제니퍼가 동작한다. 해당 옵션이 false로 설정된 경우, 애플리케이션에 trace를 하지 않아 제니퍼 서버로 성능 정보를 보내지 않는다. 제니퍼를 설치하지 않는 것과 동일한 효과를 볼 수 있으며, 실시간으로 반영된다.

- default: false
- value: BOOL
- enable = true

16.2.49.enable_active_profile

액티브 서비스 모니터링시 현재 진행중인 프로파일을 보여줄지의 여부를 지정한다.

- default: true
- value: BOOL
- enable_active_profile=true

16.2.50.enable_active_thread_kill

제니퍼 서버의 요청에 의해서 모니터링 대상 자바 애플리케이션의 액티브 쓰레드를 중지할 수 있다. 기본값이 true다. 그러나 일부 WAS에서 액티브 쓰레드를 중지하면 웹 애플리

케이션 서버가 재시작하는 현상이 발생하기도 하는데 해당 옵션을 false로 하여 이런 현상을 미연에 방지할 수 있다.

- default: true
- value: BOOL
- enable_active_thread_kill = true

16.2.51.enable_all_request_logging

해당 옵션 값이 true이면 응답 시간과 무관하게 모든 요청을 로그파일에 기록한다. 그러나 모든 요청을 로그 파일에 기록하는 것은 성능 관리 차원에서 권장하지 않는다.

- default: false
- value: BOOL
- enable_all_request_logging = false

16.2.52.enable_auto_callablestatement_close

java.sql.CallableStatement 객체의 close 메소드가 호출되지 않을 경우, 제니퍼 에이전트가 해당 객체를 가비지 컬렉션 시점에 자동으로 close 시킬지의 여부를 지정한다.

- default: false
- value: BOOL
- enable_auto_callablestatement_close=false

16.2.53.enable_auto_connection_close

java.sql.Connection 객체의 close 메소드가 명시적으로 호출되지 않을 경우, 제니퍼 에이전트가 해당 객체를 가비지 컬렉션 시점에 자동으로 close 시킬지의 여부를 지정한다.

- default: false
- value: BOOL
- enable_auto_connection_close = false

16.2.54.enable_auto_preparedstatement_close

java.sql.PreparedStatement 객체의 close 메소드가 호출되지 않을 경우, 제니퍼 에이전트가 해당 객체를 가비지 컬렉션 시점에 자동으로 close 시킬지의 여부를 지정한다.

- default: false
- value: BOOL
- enable_auto_preparedstatement_close = false

16.2.55.enable_auto_resultset_close

java.sql.ResultSet 객체의 close 메소드가 호출되지 않을 경우, 제니퍼 에이전트가 해당 객체를 가비지 컬렉션 시점에 자동으로 close 시킬지의 여부를 지정한다.

- default: false
- value: BOOL
- enable_auto_resultset_close = false

16.2.56.enable_auto_statement_close

java.sql.Statement 객체의 close 메소드가 명시적으로 호출되지 않을 경우, 제니퍼 에이전트가 해당 객체를 가비지 컬렉션 시점에 자동으로 close 시킬지의 여부를 지정한다.

- default: false
- value: BOOL
- enable_auto_statement_close = false

16.2.57.enable_class

제니퍼 에이전트는 모든 로딩되는 클래스의 정보를 관리하기 위해 별도의 디렉토리에 보관한다. 이 기능을 On/Off하기 위해 사용된다.

- default: true
- value: BOOL
- enable_classdump = false

16.2.58.enable_dbstat

WAS쪽 DB세션이 사용한 DB리소스를 추적하고 X-View에 출력한다. 단, DB에 overhead가 유발될 수 있기 때문에 테스트 목적으로만 사용해야 한다.

- default: false
- value: BOOL
- enable_dbstat = false

16.2.59.enable_dummy_httpsession_trace

모니터링 대상 애플리케이션의 JSP코드에서 명시적으로 session=false를 지정하지 않으면 자동으로 javax.servlet.HttpSession객체가 만들어지는데 경우에 따라서는 불필요한 HttpSession객체의 생성으로 인하여 성능이 저하될 수 있다. 해당 옵션을 true로 설정하면 불필요하게 HttpSession을 만드는 애플리케이션을 추적하여 이를 에러/예외 사항으로 취급한다.

- default: false
- value: BOOL
- enable_dummy_httpsession_trace = false

16.2.60.enable_dump_triggering

자동 서비스덤프 기능을 enable/disable 한다.

- default: false
- value: BOOL
- enable_dump_triggering = true

16.2.61.enable_guid_from_tuid

제니퍼 에이전트는 각 쓰레드마다 TUID를 부여하여 사용하는데, 이를 하나의 트랜잭션 GUID로 설정할지의 여부를 지정한다.

- default: false
- value: BOOL
- enable_guid_from_tuid = true

16.2.62.enable_hooking_boot

Boot Class Path에 설정된 클래스를 추적하기 위해서는 true로 설정한다.

- default: false
- value: BOOL
- enable_hooking_boot = true

16.2.63.enable_jdbc_callablestatement_fullstack_trace

java.sql.CallableStatement 객체에 대한 전체 스택트레이스를 남길지의 여부를 지정한다. enable_jdbc_stack_trace 가 true 인 경우에만 유효하며, LWST Stack Trace 에 비하여 성능저하가 있을 수 있으니 선별적으로 사용할 것을 권장한다.

- default: false
- value: BOOL
- enable_jdbc_callablestatement_fullstack_trace = false

16.2.64.enable_jdbc_callablestatement_trace

java.sql.CallableStatement 객체에 대한 모니터링 여부를 지정한다.

```
default: true
value: BOOL
enable_jdbc_callablestatement_trace = true
```

16.2.65.enable_jdbc_connection_fullstack_trace

java.sql.Connection 객체에 대한 전체 스택트레이스를 남길지의 여부를 지정한다. enable_jdbc_stack_trace 가 true 인 경우에만 유효하며, LWST Stack Trace 에 비하여 성능 저하가 있을 수 있으니 선별적으로 사용할 것을 권장한다.

- default: false
- value: BOOL
- enable_jdbc_connection_fullstack_trace = false

16.2.66.enable_jdbc_databasemetadata_trace

java.sql.DatabaseMetaData 객체에 대한 모니터링 여부를 지정한다.

- default: true
- value: BOOL
- enable_jdbc_databasemetadata_trace = true

16.2.67.enable_jdbc_datasource_trace

DataSource를 추적을 on/off한다. 디폴트는 true이다.

- default: true
- value: BOOL
- enable_jdbc_datasource_trace = true

16.2.68.enable_jdbc_oracle_dependency_used

오라클 9i JDBC 드라이버를 사용하면서 JDBC2.0 표준 API 가 아닌 오라클에 종속적인 CLOB/BLOB 형을 애플리케이션에서 사용하면서 아래와 같이 캐스팅을 하면 java.lang.ClassCastException이 발생할 수 있다.

```
CLOB clob = ((OracleResultSet) rs).getClob(i)
```

이런 경우, 오라클 10g JDBC 드라이버를 사용하고 애플리케이션 코드를 JDBC 2.0 API에 맞추어 수정할 것을 권장하나 이런 수정이 불가능 하면 해당 옵션을 true 로 지정하여 사용한다.

- default: false
- value: BOOL
- enable_jdbc_oracle_dependency_used = false

16.2.69.enable_jdbc_preparedstatement_fullstack_trace

java.sql.Connection 객체에 대한 전체 스택트레이스를 남길지의 여부를 지정한다. enable_jdbc_stack_trace 가 true 인 경우에만 유효하며, LWST Stack Trace 에 비하여 성능 저하가 있을 수 있으니 선별적으로 사용할 것을 권장한다.

- default: false
- value: BOOL
- enable_jdbc_preparedstatement_fullstack_trace = false

16.2.70.enable_jdbc_preparedstatement_trace

java.sql.PreparedStatement 객체에 대한 모니터링 여부를 지정한다.

```
default: true
value: BOOL
enable_jdbc_preparedstatement_trace = true
```

16.2.71.enable_jdbc_resultset_fullstack_trace

jjava.sql.ResultSet 객체에 대한 전체 스택트레이스를 남길 지의 여부를 지정한다. enable_jdbc_stack_trace 가 true 인 경우에만 유효하며, LWST Stack Trace 에 비하여 성능저하가 있을 수 있으니 선별적으로 사용할 것을 권장한다.

- default: false
- value: BOOL
- enable_jdbc_resultset_fullstack_trace = false

16.2.72.enable_jdbc_resultset_trace

java.sql.ResultSet 객체에 대한 모니터링 여부를 지정한다.

- default: true
- value: BOOL
- enable_jdbc_resultset_trace = true

16.2.73.enable_jdbc_sql_trace

JDBC및 SQL에 대한 모니터링 여부를 지정한다.

- default: false
- value: BOOL

- enable_jdbc_sql_trace = true

16.2.74.enable_jdbc_stack_trace

JDBC Connection/Statement/ResultSet등의 JDBC객체의 close 메소드가 명시적으로 호출되지 않았을 때, 해당 JDBC객체가 애플리케이션의 어느 위치에서 생성되었는지를 알려면 스택트레이스가 필요한데 해당 스택트레이스를 남길지의 여부를 지정한다.

- default: true
- value: BOOL
- enable_jdbc_stack_trace = true

16.2.75.enable_jdbc_statement_fullstack_trace

java.sql.Statement 객체에 대한 전체 스택트레이스를 남길지의 여부를 지정한다. enable_jdbc_stack_trace 가 true 인 경우에만 유효하며, LWST Stack Trace 에 비하여 성능저하가 있을 수 있으니 선별적으로 사용할 것을 권장한다.

- default: false
- value: BOOL
- enable_jdbc_statement_fullstack_trace = false

16.2.76.enable_jdbc_statement_trace

java.sql.Statement 객체에 대한 모니터링 여부를 지정한다.

- default: true
- value: BOOL
- enable_jdbc_statement_trace = true

16.2.77.enable_jdbc_trace_parent

본 옵션이 false이면 아래와 같이 프로그램 되어있는 경우 Statement Not Closed 에러가 발생한다.

```
Statement stmt=...
  ResultSet rs = stmt.executeQuery(...);
  ...
  Statment stmt2= rs.getStatement();
  rs.close();
  stmt2.close();
```

- default: true
- value: BOOL
- enable_jdbc_trace_parent = true

16.2.78.enable_jdbc_vendor_wrap

제니퍼가 SQL을 포함한 JDBC수행 정보를 수집할 때 기 정의된 Wrapper Class를 사용한다. 그런데 Vendor JDBC Driver 의존적인 Wrapper를 사용하고자 할 때 true로 설정한다.

Notice: 이 값이 true로 설정되면 enable_jdbc_oracle_dependency_used 옵션은 무시된다.

- default: false
- value: BOOL
- enable_jdbc_vendor_wrap = false

16.2.79.enable_logfile_daily_rotation

해당 옵션이 true 인 경우, 제니퍼 에이전트의 로그파일을 날짜 별로 남기고 형식은 로그파일명.yyyymmdd 형식으로 남긴다.

- default: false
- value: BOOL
- enable_logfile_daily_rotation = false

16.2.80.enable_long_running_thread_auto_kill

서비스 요청에 따른 쓰레드 경과시간이 지정한 일정 시간을 초과할 경우, 해당 서비스를 자동으로 강제중단 시킬지를 지정한다.

- default: false
- value: BOOL
- enable_long_running_thread_auto_kill = false

16.2.81.enable_non_servlet_thread_jdbc_trace

기본적으로 서블릿이나 JSP 쓰레드에서의 JDBC를 이용한 데이터베이스 작업에서만 JDBC 및 SQL에 대해서 모니터링 한다. 서블릿이나 JSP와 상관이 없는 쓰레드에서 발생하는 JDBC/SQL정보를 추출하려면 해당 옵션을 true로 지정하여 사용한다. EJB CMP 빈의 경우 아직 지원하지 않으므로 이 속성의 값을 false로 지정해야 한다.

- default: false
- value: BOOL
- enable_non_servlet_thread_jdbc_trace = false

16.2.82.enable_oracle_xdb_xmltype_trace

Oracle XDB oracle.xdb.XMLType 사용시 명시적으로 close()를 하지 않을 경우 native memory leak을 야기한다. 명시적으로 close하지 않은 어플리케이션을 추적할 지 결정한다.

- default: true
- value: BOOL
- enable_oracle_xdb_xmltype_trace = true

16.2.83.enable_rollback_uncommited_close

Non-XA JDBC 드라이버를 사용하여 java.sql.Connection 객체의 commit 혹은 rollback 메소드를 호출하여 데이터베이스 트랜잭션을 관리하는 애플리케이션에서 Connection 객체의 setAutoCommit(false)를 호출하고 나서 commit 혹은 rollback 을 호출하지 않을 때 Connection 객체를 커넥션 풀에 반환하면 제니퍼 에이전트가 자동으로 rollback 메소드를

호출해 주는 기능이다. 애플리케이션의 코드에 대단히 종속적인 기능이므로 반드시 검증 후에 사용해야 한다.

- default: false
- value: BOOL
- enable_rollback_uncommited_close = false

16.2.84.enable_sql_error_trace

JDBC API를 이용한 데이터베이스 처리중에 java.sql.SQLException이 발생한 경우에 그 예외를 로그파일에 기록할지의 여부를 지정한다.

- default: false
- value: BOOL
- enable_sql_error_trace = false

16.2.85.enable_web_service_logging

WAS의 애플리케이션 서비스의 성능을 로컬 파일로 로깅한다.일반적으로는 사용되지 않는다.

- default: false
- value: BOOL
- enable_web_service_logging=false

16.2.86.enable_wrap_context_jdbc_trace

제니퍼는 InitialContext로부터 DataSource가 lookup되면 SQL을 추적하지만, 그 외 다른 javax.naming.Context를 구현한 객체에 의한 DataSource lookup을 사용하는 코드에서는 기본 옵션 상태에서 SQL추적이 안된다. 이런 경우에 true로 설정해서 사용한다. EJB 가 초기화될 때 ClassCastException이 발생할 수 있다.

- default: false
- value: BOOL
- enable_wrap_context_jdbc_trace = false

16.2.87.extra_agent_class

제니퍼 에이전트의 모니터링 기능을 확장한 ExtraAgent를 구현한 클래스를 지정한다.

- default: NONE
- value: CLASS{;CLASS}*
- extra_agent_class = jennifer40.TestExtraAgent

16.2.88.extra_agent_classpath

제니퍼 에이전트의 모니터링 기능을 확장한 ExtraAgent를 구현한 클래스를 지정한다.

- default: NONE
- value: JARPATH{;JARPATH}*
- extra_agent_classpath = /jennifer/agent/lwst40.custom.jar

16.2.89.extra_agent_enabled

제니퍼 에이전트의 모니터링 기능을 확장한 ExtraAgent를 활성화한다.

- ■efault: false
- value: BOOL
- extra_agent_enabled = true

16.2.90.extra_agent_hotswap

제니퍼 에이전트에서 ExtraAgent Adapter클래스가 변경 되었을 때 리로딩 될지를 설정한다.

- default: true
- value: BOOL
- extra_agent_hotswap = true

16.2.91.extra_data_interval

ExtraAgent가 호출되는 주기를 지정한다.(Unit: ms)

- default: 1000
- value: MILLISECOND
- extra_data_interval = 1000

Notice: 1000미만 값을 설정할 수 없다.

16.2.92.extra_data_send_target

ExtraAgent로부터 수집되는 데이터를 보내기 위한 REMON 프로세스의 주소를 지정한다.

이 값이 설정되지 않으면 제니퍼 서버로 바로 전송된다.

- default: NONE
- value: HOST:PORT
- extra_data_send_target = 127.0.0.1:7701

16.2.93.extra_enable

제니퍼 에이전트의 모니터링 기능을 확장한 ExtraAgent컴포넌트를 설정할지의 여부를 지정한다.

- default: false
- value: BOOL
- extra_enable=true

16.2.94.guid_param

로직에서 GUID를 추출하기 위한 클래스메소드를 설정한다.JDBC CONNECTION설정 처럼 파라미터까지 설정해야 한다.

- default: NONE
- value: CLASS_METHOD_PARAM
- guid_param=com.TestAction.process(String)

16.2.95.guid_param_index

guid_param에서 String혹은 byte[]타입이 여러개이면서 첫번째가 아닌 경우 인덱스를 설정한다. ‘0’ 은 첫번째를 의미한다. 기본값은 -1이다.

- default: -1
- value: INT
- guid_param_index=1

16.2.96.guid_return

로직에서 GUID를 추출하기 위한 클래스메소드를 설정한다. 설정된 메소드의 리턴값이 GUID로 사용된다.

- default: NONE
- value: CLASS_METHOD_PARAM

guid_return = com.TestAction.process(String)

16.2.97.hook_class_max_size

LWST모듈에서 byte code를 instrument할 수있는 최대 크기를 설정한다. 기본값은 1048576(1mb)이다

- default: 1048576
- value: INT
- hook_class_max_size=1048576

16.2.98.hotfix_disable_thread_active_count

IBM JDK1.3 에서 Thread수가 많은 경우, waiting 이 발생할 수 있다. 해당 옵션을 true로 설정하면 이러한 현상이 발생하지 않는다.

- default: false
- value: BOOL
- hotfix_disable_thread_active_count = false

16.2.99.hotfix_remote_address_for_wmonid

제니퍼는 쿠키를 이용하여 동시단말 사용자 수와 방문자 수를 측정한다. 쿠키를 과도하게 사용하는 시스템에 제니퍼를 설치할 경우 쿠키 Overflow가 발생할 수 있다. 쿠키 Overflow가 발생할 경우 해당 옵션을 true로 설정한다. 동시단말 사용자와 방문자 수를 측정할 수 없으므로 반드시 필요한 경우가 아니면 사용하지 않기를 권장한다.

- default: false
- value: BOOL
- hotfix_remote_address_for_wmonid = false

16.2.100.http_agent_classpath

Http Agent 구현 모듈을 포함한 클래스 패스를 설정한다.

- defult: NONE
- value: JARPATH{;JARPATH}*
- http_agent_classpath = jenniferhttp.jar

16.2.101.http_post_request_tracking_list

HTTP Post방식으로 들어오는 요청은 URL에서 그 값을 볼 수 없다. 해당 옵션에 HTTP request의 key값을 나열해주면 액티브 서비스 목록에 함께 표시된다.

- default: NONE
- value: STRING{,STRING}*
- http_post_request_tracking_list = txid,txid2,txid3

16.2.102.http_service_class

HttpServlet.service(ServletRequest, ServletResponse) 를 재정의한 서블릿 클래스를 등록한다.

- default: NONE
- value: CLASS{;CLASS}*
- http_service_class = mysys.TestServlet

16.2.103.http_service_method

웹 애플리케이션 서비스 시작 메소드를 설정할 때 사용한다. 이값은 http_service_class에 설정된 클래스의 서비스 메소드가 service가 아닌 경우 사용된다.

- default: service
- value: STRING{,STRING}*
- http_service_method=service

16.2.104.ignore_jdbc_trace_url

배치(batch) 업무등과 같이 JDBC추적이 필요하지 않은 애플리케이션을 등록하여 JDBC 추적을 하지 않도록 설정한다. URL이 하나 이상인 경우에는 콤마를 구분자로 구분한다.

- default: NONE
- value: STRING{,STRING}*
- ignore_jdbc_trace_url = /batch.jsp

16.2.105.ignore_rollback_uncommited_error

제니퍼는 Statement.execute()에 대해서 Uncommit/Rollback을 추적하는데, 특정 애플리케이션이 Statement.execute()를 통해서 SELECT를 할 경우, 트랜잭션이 commit/rollback 없이 Connection 을 close 했다는 에러로 등록되는데, 해당 에러를 감지하지 않게하는 설정이다.

- default: false
- value: BOOL
- ignore_rollback_uncommited_error = false

16.2.106.ignore_url

등록된 URL을 추적에서 완전히 제외시킨다. ignore_url_postfix 와 동일한 효과다.

- default: NONE
- value: STRING{,STRING}*
- ignore_url = /board/check.jsp

16.2.107.ignore_url_post_request_parsing_prefixes

HTTP Post 방식의 요청에서 javax.servlet.http.HttpServletRequest 객체의 getInput-Stream 메소드를 호출하여 로직을 수행하는 경우에, http_post_request_tracking_list 혹은 url_additional_request_keys 옵션을 사용하면 getInputStream 메소드가 null 을 반환하여 애플리케이션이 정상적으로 동작하지 않을 수 있다. 이를 방지하기 위해서 getInput-Stream 메소드를 호출하는 URL의 prefix 를 등록한다.

- default: NONE
- value: STRING{,STRING}*
- ignore_url_post_request_parsing_prefixes = /kesa.web,/insa/x_servlet

16.2.108.ignore_url_postfix

이미지와 같은 정적 컨텐츠를 웹 애플리케이션 서버가 직접 서비스하는 경우에 컨텐츠에 대한 모니터링을 하지 않으려면 해당 옵션에 정적 컨텐츠의 (“.”) 와 확장자를 포함하여 등록한다. 두 개 이상인 경우 공백(space)나 콤마(,)를 구분자로 입력한다.

```
default: NONE
value: STRING{[ ‘ ‘|,]STRING}*
ignore_url_postfix = .gif .GIF .jpg .JPG .zip .html .HTML .txt .css .js
.swf .htc .HTC .png .PNG
```

16.2.109.ignore_url_prefix

모니터링에서 무시해야하는 URL이 여러개인 경우 그것을 prefix로 설정할 수있다.

- default: NONE
- value: STRING{,STRING}*
- ignore_url_prefix=/admin, /console

16.2.110.is_all_request_logging

서비스 트랜잭션 성능 데이터를 제니퍼 로그 파일로 로깅하도록 설정한다.

- default: false
- value: BOOL
- is_all_request_logging=false

16.2.111.jdbc_connection_close

JDBC Connection Pool을 자체 제작하였거나, Connection Pool을 Wrapping 하는 경우에 JDBC Connection을 반환하는 부분을 설정한다.

1. 패키지.클래스명.메소드명은 반드시 full package명이 기술되어야 한다.
2. 메소드의 파라미터중에 java.sql.Connection 이 반드시 포함되어 있어야 한다..
3. jdbc_connection_get 과 반드시 쌍으로 존재할 필요는 없다. conn.close() 형태로 pool로 반환한다면 해당 옵션이 필요없다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
jdbc_connection_close =
oracle.apps.fnd.common.Context.releaseJDBCConnection(Connection,String)
```

• jdbc_connection_get

JDBC Connection Pool을 자체 제작하였거나, Connection Pool을 Wrapping 하는 경우에 JDBC Connection을 얻어오는 부분을 설정한다.

1. 패키지.클래스명.메소드명은 반드시 full package명이 기술되어야 한다.
2. 메소드의 return type이 반드시 java.sql.Connection 이어야 한다.
3. 메소드의 파라미터도 기술되어야 한다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
jdbc_connection_get = db.DBConnectionManager.getConnection(String)
```

• jdbc_connection_justget

Connection반환을 추적할 수 없는 경우에 Connection획득부분만 이 옵션에 설정한다.여기에 설정된 Connection은 LEAK여부를 검사하지 않는다,

Warning: 단 만약 명시적인 반환메소드가 존재하는 Connection Pool을 여기에 설정하면 서비스 에러가 발생할 수있다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
jdbc_connection_justget=db.DBConnectionManager.getConnection(String)
```

• jdbc_resultset_warning_fetch_count

특정 애플리케이션이 수행되는 과정에서 java.sql.ResultSet 객체의 next 메소드가 지정한 값 이상으로 호출되면 WARNING_JDBC_TOOMANY_RS_NEXT 유형의 장애가 발생한다.

```
default: 10000
value: INT
jdbc_resultset_warning_fetch_count = 10000
```

• jdump_dir

서비스덤프 파일이 생성될 디렉토리 위치를 지정한다.

```
default: ./
value: DIRPATH
jdump_dir = ./
```

• leakcheck_${xxx}_close

사용자 정의형 리소스릭을 추적하기 위한 설정이다. 애플리케이션이 리소스를 반환하는 부분을 지정한다.

Notice: ${xxx}에는 임의의 문자열을 사용할 수있지만, leakcheck_${xxx}_close의 ${xxx}와 일치해야 한다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
leakcheck_mypool_close=com.MyPool.release(ResItem)
```

• leakcheck_${xxx}_get

사용자 정의형 리소스릭을 추적하기 위한 설정이다. 애플리케이션이 리소스를 가져가는 부분을 지정한다..

Notice: ${xxx}에는 임의의 문자열을 사용할 수있지만, leakcheck_${xxx}_close의 ${xxx}와 일치해야 한다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
leakcheck_mypool_get=com.MyPool.get()
```

• license_filename

제니퍼 에이전트의 라이센스 키 파일의 위치를 지정한다. 해당 옵션에 파일명만 기입하면 JVM의 현재 디렉토리에 파일이 생성된다. 파일의 전체경로로 지정하는 것을 권장한다.

```
default: license.txt
value: FILEPATH
license_filename = license.txt
```

• liveobject_class

LIVE OBJECT COUNTING을 위한 대상 클래스를 지정한다.

LIVE OBJECT COUNTING

메모리에 생성된(created) 객체 인스턴스의 수를 카운팅 한다.

```
default: NONE
value: CLASS{;CLASS}*
liveobject_class=com.TextA
```

• liveobject_interface

LIVE OBJECT COUNTING을 위한 대상 클래스들의 공통 인터페이스클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
liveobject_interface=com.ITestClassA;org.ITestClassB
```

• liveobject_postfix

LIVE OBJECT COUNTING을 위한 대상 클래스들의 공통 POSTFIX를 지정한다.

```
default: NONE
value: STRING{,STRING}*
liveobject_postfix=Adapeter;VO;Action
```

• liveobject_prefix

LIVE OBJECT COUNTING을 위한 대상 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
value: STRING{,STRING}*
liveobject_prefix=com;org.apache.common
```

• liveobject_super

LIVE OBJECT COUNTING을 위한 대상 클래스들의 공통 SUPER클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
```

```
liveobject_super=com.TestSuperA;com.TestSuperB
```

• logfile

제니퍼 에이전트가 남기는 로그를 기록하는 파일의 위치를 지정한다. 해당 로그 파일에는 애플리케이션에서 발생하는 에러 로그, 서비스 덤프 내용, 디버그 내용들이 기록된다. 해당 옵션에 파일 명만 기입하면 JVM의 현재 디렉토리에 파일이 생성된다. 파일의 전체경로로 지정하는것을 권장한다. 제니퍼 에이전트의 로그를 날짜별로 기록하기 원하면 enable_logfile_daily_rotaion 옵션을 활성화 시킨다.

```
default: jennifer.log
value: FILEPATH
logfile = jennifer.log
```

• logfile_encoding_characterset

제니퍼 에이전트가 로그를 남길 때, 로그파일의 인코딩 형식을 지정한다. 대부분의 경우 WAS의 기본 인코딩을 사용하므로 해당 옵션을 사용하지 않지만, z/OS인 경우 Cp933 으로 설정해준다.

```
default: NONE
value: STRING
logfile_encoding_characterset = Cp933
```

• long_running_thread_auto_kill_timeout

사용자의 요청을 처리하는 쓰레드의 경과시간이 해당 옵션값을 초과한 경우에 중단시킨다. 단 해당 옵션은 enable_long_running_thread_auto_kill 이 true일 경우에만 의미를 갖는다.

```
default: -1
value: INT
long_running_thread_auto_kill_timeout = 300000
```

• lwst_call_stack_transfering

제니퍼 에이전트에서 메소드 레벨에서의 Call Stack(중요 메소드, SQL, EJB 등)을 제니퍼 서버로 전송할지 결정한다.

```
default: true
value: BOOL
lwst_call_stack_transfering = true
```

• lwst_collection_auto_stacktrace_size

콜렉션을 모니터링 할때 엘리먼트 개수가 해당 옵션 값 이상이되면 자동으로 stack trace를 남기도록 하는 설정이다.

```
default: 100000
value: INT
lwst_collection_auto_stacktrace_size = 100000
```

• lwst_collection_minimum_monitoring_size

콜렉션을 모니터링 할때 엘리먼트 개수가 해당 옵션 값 이상일 경우에만 추적하도록 설정한다.

```
default: 3000
value: INT
lwst_collection_minimum_monitoring_size = 3000
```

• lwst_debug

LWST모듈의 디버그 정보를 로깅한다. 특히 로딩되는 클래스명과 로더등을 로그파일에 남긴다.

```
default: false
value: BOOL
lwst_debug=false
```

• lwst_enable

제니퍼 LWST의 동작 여부를 지정한다. 이 옵션이 false이면 제니퍼 LWST가 동작하지 않는다.

LWST 모니터링 대상클래스는 build시(lwst.bat실행) 추적 로직이 삽입되기도 하고 실행시에 클래스 로드 시점에 추적 로직이 삽입되기도 한다. lwst_enable옵션은 런타임에 로딩되는 클래스에 대한 추적여부를 결정한다.

```
default: true
value: BOOL
lwst_enable=false
```

• lwst_logfile

제니퍼 에이전트의 lwst와 관련된 로그를 기록하는 파일의 위치를 지정한다. 해당 로그 파일에는 tx-server, tx-naming, tx-client, profile 대상의 클래스/메소드 정보, lwst 디버그 내용 등이 기록된다. 해당 옵션에 파일 명만 기입하면 JVM의 현재 디렉토리 내에 파일이 생성된다. 파일의 전체 경로로 지정하는 것을 권장한다.

```
default: lwst.log
```

```
value: FILEPATH
lwst_logfile = lwst.log
```

• lwst_posthooking_dump

LWST는 class byte code를 핸들링한다. 원하는 옵션이 정확히 적용된 내용을 코드상에서 확인하기 위해서 byte code가 instrument된 상태를 덤프한다.

```
default: NONE
value: CLASS
lwst_posthooking_dump=com.Test
```

• lwst_profile_method_param_limit

메소드의 파라미터나 리턴 값을 프로파일링할때 최대 문자열을 지정한다.

```
default: 100
value: INT
lwst_profile_method_param_limit = 100
```

• lwst_profile_method_using_param

메소드의 파라미터를 프로파일링 한다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
lwst_profile_method_using_param=foo.XBoardVO.setOid(String);foo.XBoardVO
.get(String)
```

• lwst_profile_method_using_return

메소드의 리턴값을 프로파일링 한다.(String리턴만 가능)

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
lwst_profile_method_using_return=foo.ATomServlet.getActName(String,Strin
g)
```

• lwst_trace_local_port

lwst_trace_local_port 옵션으로 지정한 로컬 포트로 모니터링 대상 애플리케이션이 TCP/IP 소켓을 사용하면 제니퍼 에이전트는 자동으로 해당 애플리케이션의 스택 트레이스를 저장한다. 사용자는 이 정보를 제니퍼 사용자 인터페이스의 **[장애 진단 | 파일/소켓]** 메뉴에서 확인할 수 있다.

하나의 포트만 지정할 수 있으며, Socket이 열릴 때마다 Stack Trace를 남기기 때문에, 풀링이 되지 않는 연결의 포트를 지정하면 성능저하가 발생할 수있다.

```
default: 0
value: INT
lwst_trace_local_port=7700
```

• lwst_trace_remote_port

lwst_trace_local_port 옵션으로 지정한 원격 포트로 모니터링 대상 애플리케이션이 TCP/IP 소켓을 사용하면 제니퍼 에이전트는 자동으로 해당 애플리케이션의 스택 트레이스를 저장한다. 사용자는 이 정보를 제니퍼 사용자 인터페이스의 **[장애 진단 | 파일/소켓]** 메뉴에서 확인할 수 있다.

하나의 포트만 지정할 수있으며, Socket이 열릴때 마다 Stack Trace를 남김으로 풀링이되지 않는 연결의 포트를 지정하면 성능저하가 발생할 수있다.

```
default: 0
value: INT
lwst_trace_local_port=1521
```

• lwst_txclient_method_using_param

TX-CLIENT(외부 트랜잭션) 이름을 특정 메소드의 파라미터 값으로 사용하도록 지정한다. 두개 이상의 메소드는 세미 콜론(;)을 구분자로 구분한다.

Notice: 지정된 메소드의 첫번째 String 파라미터가 서비스 명으로 사용된다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
lwst_txserver_method_using_param=foo.Foo.getFoo(String)
```

• lwst_txclient_method_using_return

TX-CLIENT(외부 트랜잭션) 이름을 특정 메소드의 리턴 값을 사용하도록 지정한다.두개 이상의 메소드는 세미 콜론(;)을 구분자로 구분한다.

Notice: 이 옵션은 메소드 리턴 타입이 String인 경우에만 사용할 수있다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
lwst_profile_method_using_return=foo.Foo.getFoo(String,String)
```

• lwst_txserver_method_using_param

서비스 트랜잭션명을 특정 메소드의 파라미터 값으로 사용하도록 지정한다. 두개 이상의 메소드는 세미 콜론(;)으로 구분한다.

Notice: 지정된 메소드의 첫번째 String 파라미터가 서비스 명으로 사용된다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
lwst_txserver_method_using_param=foo.Foo.getFoo(String)
```

• lwst_txserver_method_using_return

서비스 트랜잭션명을 특정 메소드의 리턴 값을 사용하도록 지정한다. 두개 이상의 메소드는 세미 콜론(;)을 구분자로 구분한다.

Notice: 이 옵션은 메소드 리턴 타입이 String인 경우에만 사용할 수있다.

```
default: NONE
value: CLASS_METHOD_PARAM{;CLASS_METHOD_PARAM}*
lwst_profile_method_using_return=foo.Foo.getFoo(String,String)
```

• max_num_of_active_service

set_limit_active_service가 true인 경우에 한하여, 액티브 서비스가 해당 옵션 값보다 커지면 서비스 큐잉 현상을 막기 위해서 추가적인 요청을 배척(Reject)한다.

```
default: 1024
value: INT
max_num_of_active_service = 100
```

• max_num_of_direct_alert

제니퍼 에이전트에서 초당으로 직접 발령가능한 최대 경보 수를 설정한다. 본 설정은 에이전트가 프로그래매틱하게 발령하는 USER_DEFINED_XXX 경보들만 해당된다.

```
default: 10
value: INT
max_num_of_direct_alert=10
```

• max_num_of_parsed_sql

매번 SQL을 파싱하는 부하를 줄이기 위해, 전혀 파싱할 필요가 없는 SQL들과 파싱된 SQL들을 메모리에 캐싱을 한다. 해당 옵션으로 그 메모리에 캐싱할 수 있는 SQL수를 지정한다.

Notice: 해당 옵션값을 너무 크게 설정하면 대량의 메모리를 점유할 수 있으니 애플리케이션과 사용되는 SQL특성을 고려하여 적절하게 설정한다. 기본값 사용을 권장한다.

```
default: 513
value: INT
max_num_of_parsed_sql = 513
```

• max_num_of_text

제니퍼는 단위 시간동안 제니퍼 서버가 데이터를 수집하기 전까지 TEXT(APP이름, SQL 문, TX-CLIENT 이름, 에러)을 메모리에 보관한다. 이때 보관할 최대 크기이다. 이 값이 커지면 비정상 상황일 때 제니퍼가 사용하는 메모리가 증가할 수 있다.

특히 SQL의 경우 제니퍼는 모든 SQL을 파싱하여 상수부를 제거함으로 정상 상황에서 사용되는 SQL 종류가 과도하게 증가할 경우는 드물다. 그러나 TPS가 높고, 일정시간단위 동안 SQL 의 수가 많은 경우 해당 옵션값을 늘려줘야한다.

```
default: 1003
value: INT
max_num_of_text = 1003
```

• non_public_class

public이 아닌 클래스의 접근 권한을 public으로 변환한다.

```
default: NONE
value: CLASS
non_public_class=com.TestClass
```

• non_public_member

public이 아닌 필드/메소드의 접근자를 public으로 변환한다.

```
default: NONE
value: CLASS_FIELD|CLASS_METHOD
non_public_member=com.TescClass.field
```

• number_of_dump_trigger

자동으로 서비스덤프가 생성될 액티브 서비스 개수를 지정한다.

```
default: 100
value: INT
number_of_dump_trigger = 90
```

• number_of_tcp_workers

제니퍼 서버의 요청을 처리하는 제니퍼 에이전트의 쓰레드 수를 지정한다. 해당 옵션 값을 반영하려면 제니퍼 에이전트가 설치된 WAS를 재시작해야 한다.

```
default: 3
value: INT
number_of_tcp_workers = 5
```

• patch@

테 스트 목적으로 백엔드 연동 모듈 등의 특정 클래스/메소드의 반환 값을 상수처리 하거나 혹은 운영 환경에 적용된 특정 클래스를 임시로 변경하고 싶을 때도 patch@ 옵션을 사용할 수 있다. 다음과 같이 변경하고자 하는 원본 클래스명을 patch@ 다음에 지정하고 패치할 클래스의 물리적 파일 위치를 지정한다.

```
default: NONE
format: patch@CLASS=FILEPATH
patch@foo.bar.BizLogic=C:/foo/bar/PatchedBizLogic.clz
```

• profile_access_method

서비스 PROFILING에 포함된 클래스에서 대상 메소드의 접근 권한을 지정한다.

```
default: public,protected,private,none
value:(public|protected|private|none){,(public|protected|private|none)}*
profile_access_method=public,protected,private,none
```

• profile_buffer_size

트랜잭션당 최대 프로파일 엔트리 개수이다. X-View에서 최대 프로파일 선과 같다.

```
default: 1000
value: INT
profile_buffer_size=1000
```

• profile_call

프로파일링 대상 메소드의 코드를 검토하여 해당 코드가 이 옵션으로 등록한 메소드를 호출하면 이를 X-View 그래프에 표시한다. 두개 이상의 메소드는 세미 콜론(;)을 구분자로 구분한다.

```
default: NONE
value: (CLASS_METHOD|\*\.METHOD){;(CLASS_METHOD|\*\.METHOD)}*
profile_call=java.lang.Thread.sleep;*.wait
```

앞의 옵션처럼 설정하면 java.lang.Thread 클래스의 sleep 메소드나 어떤 클래스나 객체의 wait 메소드를 호출하는 메소드를 X-View 그래프에 표시한다.

• profile_class

서비스 PROFILING을 위한 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
profile_class=com.TestClassA;com.TestClassB
```

• profile_class_on

이 설정이 true이면 profile_class에 설정된 클래스가 최초 로딩시 부터 호출 정보를 추적한다.

```
default: false
value: BOOL
profile_class_on=true
```

• profile_default_on

이 설정이 true이면 프로파일 대상 클래스가 최초 로딩시 부터 호출 정보를 추적한다.

```
default: false
value: BOOL
profile_default_on=false
```

• profile_ignore_method

서비스 PROFILING에 포함된 클래스에서 제외할 메소드만을 지정한다.

> **Notice:** 메소드명혹은 클래스+메소드를 지정할 수있다.

```
default: NONE
value: (CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
profile_ignore_method=access;com.Test.save
```

• profile_ignore_postfix

서비스 PROFILING에서 제외할 클래스들의 공통 POSTFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
profile_ignore_postfix=Util;VO
```

• profile_ignore_prefix

서비스 PROFILING에서 제외할 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
```

```
value: STRING{;STRING}*
profile_ignore_prefix=com.util;com.jennifersoft.util
```

• profile_interface

서비스 PROFILING을 위한 대상 클래스들의 공통 인터페이스 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
profile_interface=com.ITestClassA;org.ITestClassB
```

• profile_interface_on

이 설정이 true이면 profile_interface에 설정된 클래스가 최초 로딩시 부터 호출 정보를 추적한다.

```
default: false
value: BOOL
profile_interface_on=true
```

• profile_method_max_byte

PROFILING이 적용될 메소드의 최대 코드 사이즈를 지정한다. 코드 사이즈는 메소드 바이너리 기준이다.

Notice: 메소드 코드는 64k를 넘을 수 없다. 프로파일 적용 후 코드 오버플로어를 막기위해 사용된다.

```
default: 48000
value: INT
profile_method_max_byte=48000
```

• profile_method_min_byte

PROFILING이 적용될 메소드의 최소 코드 사이즈를 지정한다. 코드 사이즈는 메소드 바이너리 기준이다.

Notice: getter/setter와 같은 단순 메소드를 프로파일에서 제거하기 위해 사용된다.

```
default: 0
value: INT
profile_method_min_byte = 0
```

• profile_postfix

서비스 PROFILING을 위한 대상 클래스들의 공통 POSTFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
profile_postfix=Adapeter;VO;Action
```

• profile_prefix

서비스 PROFILING을 위한 대상 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
profile_prefix=com;org.apache.common
```

• profile_super

서비스 PROFILING을 위한 대상 클래스들의 공통 SUPER 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
profile_super=com.TestSuperA;com.TestSuperB
```

• profile_super_on

이 설정이 true이면 profile_super에 의해 설정된 클래스가 최초 로딩시 부터 호출 정보를 추적한다.

```
default: false
value: BOOL
profile_super_on=true
```

• profile_target_method

서비스 PROFILING에 포함된 클래스에서 대상 메소드만을 지정한다.

> **Notice:** 메소드명혹은 클래스+메소드를 지정할 수있다.

```
default: NONE
value: (CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
profile_target_method=access;com.Test.save
```

• recurrsive_call_max_count

RequestDispatcher의 forward()를 잘못 사용할 경우, IOException발생할 때, 통상 error 페이지로 분기하는데, 이때 error페이지가 없으면 또다시 IOException이 발생하고, 자칫 무

한 반복 재귀 호출이 일어날 수 있으며, 이는 JVM의 StackOverFlow를 유발하여 JVM이 다운될 수 있다. 이처럼 동일 Thread에서 HttpServlet.service()가 반복적으로 호출될 때, 그 호출 횟수를 제한하는 기능이다.

```
default: 50000
value: INT
recurrsive_call_max_count = 50000
```

• remote_address_header_key

X-View를 통해서 사용자 IP를 확인할 수 있다. 해당 IP는 request.getRemoteAdd()의 값이다. 사용자 IP가 아닌 L4나 웹서버의 IP가 보일 경우 request.getHeader(key)에서 실제 사용자 IP를 얻을 수 있다. 해당 키값을 옵션에 설정하면 실제 사용자의 IP가 나온다.

```
default: NONE
value: STRING
remote_address_header_key=key1
```

• request_reject_message

request_reject_type의 속성을 message로 지정한 경우 사용자에게 보여줄 메시지를 지정한다.

```
default: Workload so high. Please, try again later!
value: STRING
request_reject_message = Workload so high. Please, try again later!
```

• request_reject_redirect_url

request_reject_type의 속성을 redirect로 지정한 경우, 사용자에게 보여줄 페이지를 지정한다.

Notice: 반드시 static 페이지로 지정해야 한다. PLC기능에 의해서 해당 페이지도 reject 될 수 있다.

```
default: NONE
value: STRING
request_reject_redirect_url = /error.html
```

• request_reject_type

set_limit_active_service가 true인 경우에 한하여, PLC에 의해 사용자의 요청이 거부될 경우 어떠한 형식으로 관련 메세지를 보여줄것인가를 지정한다. message 와 redirect 형식이 있다.

```
default: message
```

```
value: message|redirect
request_reject_type = message
```

• request_set_character_encoding

아파치 톰켓과 같이 웹 애플리케이션 서버 차원에서 다국어 문자 지원이 미비할 때, HttpServletRequest의 getParameter 메소드가 다국어(예: 한국어)를 지원하지 않을 수 있다. 그런데 서블릿 2.3부터는 HttpServletRequest의 setCharacterEncoding 메소드를 통해서 다국어에 대한 처리를 쉽게 할 수 있다.

그런데 http_post_request_tracking_list이나 url_additional_request_keys 속성을 사용하면 모니터링 대상 애플리케이션에서 HttpServletRequest의 setCharacterEncoding 메소드를 호출하여도 제니퍼 에이전트가 먼저 HttpServletRequest의 getParameter 메소드를 호출하기 때문에 다국어에 대한 처리가 올바르게 되지 않을 수 있다. 이 문제를 해결하기 위해서 이 속성을 지정하면 제니퍼 에이전트가 맨 앞단에서 setCharacterEncoding 메소드를 호출한다.

```
default: NONE
value: STRING
request_set_character_encoding = KSC5601
```

• server_udp_listen_port

제니퍼 에이전트는 주기적으로 (기본 1초) 통계 데이터를 해당 포트를 통해서 제니퍼 서버로 전송하며 ,대략 230바이트 정도의 데이터를 전송한다. 해당 데이터를 받아 줄 제니퍼 서버의 UDP포트를 지정한다.

Notice: 해당 포트로 Wmond가 보내주는 모니터링 대상 서버의 CPU정보도 취합한다.

```
default: 6902
value: PORT
server_udp_listen_port = 6902
```

• server_udp_lwst_call_stack_port

제니퍼 에이전트는 사용자의 요청이 수행되고 종료될 때마다 프로파일링 데이터를 해당 포트를 통해서 제니퍼 서버로 전송한다. 해당 데이터를 받아 줄 제니퍼 서버의 UDP 포트를 지정한다.

```
default: 6703
value: PORT
server_udp_lwst_call_stack_port = 6703
```

• server_udp_runtime_port

제니퍼 에이전트는 사용자의 요청이 발생할 때마다 실시간 데이터를 해당 포트를 통해서 제니퍼 서버로 전송하며, 매번 약 12~33바이트 정도의 데이터를 전송한다. 해당 데이터를 받아 줄 제니퍼 서버의 UDP포트를 지정한다.

```
default: 6901
value: PORT
server_udp_runtime_port = 6901
```

• service_log_class

서비스 트랜잭션 추적에서 사용할 ServiceLogger 어뎁터 클래스를 지정한다. 여기에 지정된 클래스는 ServiceLogger.Adapter를 Implemdnt해야 한다.

```
default: NONE
value: CLASS
service_log_class=jennifer.service.MemAllocTrace
```

• service_log_end

서비스 트랜잭션 추적 종료부에 ServiceLogger.Adapter의 동작을 허용한다.

```
default: false
value: BOOL
service_log_end = true
```

• service_log_classpath

ServiceLogger 어뎁터를 포함한 jar파일의 경로를 지정한다.

```
default: NONE
value: JARPATH{;JARPATH}*
service_log_classpath=/jennifer/agent/lwst40.custom.jar
```

• service_log_start

서비스 트랜잭션 추적 시작부에 ServiceLogger.Adapter의 동작을 허용한다.

```
default: false
value: BOOL
service_log_start = true
```

• session_class

WAS의 HTTP SESSION 객체를 지정한다. 아래의 값중에 하나만 지정가능하다

```
default: NONE
```

```
value: CLASS{;CLASS}*
session_class=org.apache.tomcat.session.StandardManager;org.apache.tomcat.modules.session.SimpleSessionStore$SimpleSessionManager;org.apache.catalina.session.StandardManager

session_class=com.evermind.server.http.HttpApplication;org.apache.jserv.JServServletManager;com.evermind._ay;jrun.servlet.session.SessionService

session_class=weblogic.servlet.internal.session.MemorySessionContext;weblogic.servlet.internal.session.JDBCSessionContext;weblogic.servlet.internal.session.ReplicatedSessionContext;weblogic.servlet.internal.session.FileSessionContext;weblogic.servlet.internal.session.CookieSessionContext

session_class=jeus.servlet.engine.ContextGroup

session_class=com.fujitsu.interstage.jservlet.tomcat.session.StandardManager
```

• set_limit_active_service

성능 제어기능인 PCL(Peak Load Control)의 사용여부를 지정한다.

```
default: false
value: BOOL
set_limit_active_service = false
```

• socket_simple_trace

소켓에 의한 전송량과 상태 등은 추적하지 않고 특정 포트가 열렸다는 정보만을 X-View에 표시한다. lwst_trace_remote_port나 lwst_trace_local_port 옵션으로 지정한 포트는 그 옵션 지정에 따른 방식으로 처리된다.

```
default: false
value: BOOL
socket_simple_trace=true
```

• sql_bad_responsetime

트랜잭션 중에 SQL의 응답시간이 해당 옵션 값 이상을 초과한 경우, 로그파일에 기록되고 제니퍼 에러에 등록된다. 매우 작은 시간을 지정하면 로그 파일의 크기가 커지고 , 에러 건 수가 증가하기 때문에 사이트의 성능 특성을 고려하여 적절하게 지정한다. 단위는 밀리세컨드(ms)이다.

```
default: 20000
```

```
value: MILLISECOND
sql_bad_responsetime = 20000
```

• stacktrace_class

LWST STACKTRACING를 위한 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
stacktrace_class=com.TestClassA;com.TestClassB
```

LWST STACKTRACING

Java VM은 내부에서 쓰레드 별로 콜스택을 관리하고 자바코어 덤프시에 이것을 보여준다 하지만 Java VM의 스택은 오버헤드가 크기 때문에 제니퍼에서는 필요한 클래스들만 설정하여 별도로 Light한 스택을 관리한다 제니퍼가 관리하는 이 스택을 LWST STACK이라 한다.

• stacktrace_interface

LWST STACKTRACING를 위한 대상 클래스들의 공통 인터페이스 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
stacktrace_interface=com.ITestClassA;org.ITestClassB
```

• stacktrace_postfix

LWST STACKTRACING를 위한 대상 클래스들의 공통 POSTFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
stacktrace_postfix=Adapeter;Action
```

• stacktrace_prefix

LWST STACKTRACING를 위한 대상 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
stacktrace_prefix=com.mybiz
```

• stacktrace_stacksize

LWST STACKTRACING를 위한 스택의 크기를 지정한다.

Notice: profile_buffer_size 옵션은 커야 하지만 이 옵션은 그에 비해서는 작아야 한다.

```
default: 200
value:INT
stacktrace_stacksize=200
```

• stacktrace_super

LWST STACKTRACING를 위한 대상클래스들의 공통 SUPER 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
stacktrace_super=com.TestSuperA;com.TestSuperB
```

• trace_related_transaction

멀티 쓰레드에 의해 동작하는 트랜잭션을 추적할지의 여부를 지정한다. ESB 환경에서 적용되는 옵션이다.

```
default: true
value: BOOL
trace_related_transaction = true
```

• tx_bad_responsetime

애플리케이션의 외부 트랜잭션 (TX-tuxedo, tmax, cics, etc...)의 응답 시간이 지정한 시간 이상을 초과한 경우, 로그파일에 기록되고 제니퍼 에러로 등록된다. 매우 작은 시간을 지정하면 로그 파일의 크기가 커지고 에러건수가 증가하기 때문에 사이트의 성능 특성을 고려하여 적절하게 지정한다. 단위는 밀리세컨드(ms)이다.

```
default: 10000
value: MILLISECOND
tx_bad_responsetime  = 10000
```

• tx_client_access_method

TX-CLIENT(외부 트랜잭션) 추적에 포함된 클래스에서 대상 메소드의 접근권한을 지정한다.

```
default: public,protected,private,none
value:(public|protected|private|none){,(public|protected|private|none)}*
tx_client_access_method=public,protected,private,none
```

• tx_client_class

TX-CLIENT(외부 트랜잭션) 추적을 위한 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
tx_client_class=com.TestClassA;com.TestClassB
```

TX CLIENT(외부 트랜잭션) 추적

제니퍼는 외부 트랜잭션 호출과 현 서비스의 상관관계를 모니터링한다. 외부 트랜잭션은 텍시도나 CICS처럼 이미 알려진 경우도 있지만 단순 소켓통신처럼 그 시스템에서만 존재하는 경우도 있다. 제니퍼에서 외부 연계는 tx_client로 지정하여 모니터링 한다.

• tx_client_ignore_method

TX-CLIENT(외부 트랜잭션) 추적에 포함된 클래스에서 제외할 메소드만을 지정한다.

Notice: 메소드명혹은 클래스+메소드를 지정할 수있다.

```
default: NONE
value: (CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
tx_client_ignore_method=access;com.Test.save
```

• tx_client_ignore_prefix

TX-CLIENT(외부 트랜잭션) 추적에서 제외할 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
tx_client_ignore_prefix=com.util;com.jennifersoft.util
```

• tx_client_interface

TX-CLIENT(외부 트랜잭션) 추적을 위한 대상 클래스들의 공통 인터페이스 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
tx_client_interface=com.ITestClassA;org.ITestClassB
```

• tx_client_ntype

TX-CLIENT(외부 트랜잭션) 추적에서 Naming타입을 지정한다.

```
default: FULL
value: FULL|CLASS|METHOD|SIMPLE
```

```
tx_client_ntype=CLASS
```

• tx_client_postfix

TX-CLIENT(외부 트랜잭션) 추적을 위한 대상 클래스들의 공통 POSTFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
tx_client_postfix=Adapeter;VO;Action
```

• tx_client_prefix

TX-CLIENT(외부 트랜잭션) 추적을 위한 대상 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
tx_client_prefix=com;org.apache.common
```

• tx_client_super

TX-CLIENT(외부 트랜잭션) 추적을 위한 대상클래스들의 공통 SUPER클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
tx_client_super=com.TestSuperA;com.TestSuperB
```

• tx_client_target_method

TX-CLIENT(외부 트랜잭션) 추적에 포함된 클래스에서 대상 메소드만을 지정한다.

Notice: 메소드명혹은 클래스+메소드를 지정할 수있다.

```
default: NONE
value: (CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
tx_client_target_method=access;com.Test.save
```

• tx_client_using_param

TX-CLIENT(외부 트랜잭션) 이름을 메소드의 스트링 파라미터로 부여한다.

Notice: 이 옵션이 true로 설정되고 메소드에 String파라미터가 존재하면 tx_client_ntype은 무시된다.

```
default: false
value: BOOL
tx_client_using_param=true
```

• tx_naming_access_method

TX-NAMING(서비스 추가 이름)에 포함된 클래스에서 대상 메소드의 접근 권한을 지정한다.

```
default: public,protected,private,none
value:(public|protected|private|none){,(public|protected|private|none)}*
tx_naming_access_method=public,protected,private,none
```

• tx_naming_class

TX-NAMING(서비스 추가 이름)을 위한 클래스를 지정한다.

```
efault: NONE
value: CLASS{;CLASS}*
tx_naming_class=com.TestClassA;com.TestClassB
```

• tx_naming_ignore_method

TX-NAMING(서비스 추가 이름)에 포함된 클래스에서 제외할 메소드만을 지정한다. .

Notice: 메소드 명 혹은 클래스+메소드를 지정할 수 있다.

```
default: NONE
value: (CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
tx_naming_ignore_method=access;com.Test.save
```

• tx_naming_ignore_prefix

TX-NAMING(서비스 추가 이름)에서 제외할 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
tx_naming_ignore_prefix=com.util;com.jennifersoft.util
```

• tx_naming_interface

TX-NAMING(서비스 추가 이름)을 위한 대상 클래스들의 공통 인터페이스 클래스를 지정한다.

```
default: NONE
value: CLASS{;CLASS}*
tx_naming_interface=com.ITestClassA;org.ITestClassB
```

• tx_naming_ntype

TX-NAMING(서비스 추가 이름)에 적용된 클래스가 애플리케이션 서비스 명에 추가될 때는 클래스명+메소드명이 사용된다(단 `tx_naming_using_param=false`). 이때 추가되는 서비스 명을 축약하여 사용하기 위한 옵션이다.

```
default: FULL
value: FULL|CLASS|METHOD|SIMPLE
tx_naming_ntype=CLASS
```

• tx_naming_postfix

TX-NAMING(서비스 추가 이름)을 위한 대상 클래스들의 공통 POSTFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
tx_naming_postfix=Adapeter;VO;Action
```

• tx_naming_prefix

TX-NAMING(서비스 추가 이름)을 위한 대상 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
value: STRING{;STRING}*
tx_naming_prefix=com;org.apache.common
```

• tx_naming_super

TX-NAMING(서비스 추가 이름)을 위한 대상 클래스들의 공통 SUPER 클래스를 지정한다.

```
default: NONE
value:CLASS{;CLASS}*
tx_naming_super=com.TestSuperA;com.TestSuperB
```

• tx_naming_target_method

TX-NAMING(서비스 추가 이름)에 포함된 클래스에서 대상 메소드만을 지정한다.

> **Notice:** 메소드 명 혹은 클래스+메소드를 지정할 수 있다.

```
default: NONE
value:(CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
tx_naming_target_method=access;com.Test.save
```

• tx_naming_using_param

TX-NAMING(서비스 추가 이름)에 설정된 클래스가 호출될 때 첫번째 String 파라미터 값을 Additional 서비스 명으로 사용하도록 한다.

```
default: false
value:BOOL
tx_naming_using_param=true
```

• tx_naming_using_return

TX-NAMING(서비스 추가 이름)에 설정된 클래스가 호출될 때 String 리턴 값을 Additional 서비스 명으로 사용하도록 한다.

```
default: false
value:BOOL
tx_naming_using_return=false
```

• tx_server_access_method

TX-SERVER(서비스 시작)에 포함된 클래스에서 대상 메소드의 접근 권한을 지정한다.

```
default: public,protected,private,none
value:(public|protected|private|none){,(public|protected|private|none)}*
tx_server_access_method=public,protected,private,none
```

• tx_server_class

TX-SERVER(서비스 시작)를 위한 클래스를 지정한다.

```
default: NONE
value:CLASS{;CLASS}*
tx_server_class=com.TestClassA;com.TestClassB
```

• tx_server_ignore_method

TX-SERVER(서비스 시작)에 포함된 클래스에서 제외할 메소드만을 지정한다.

Notice: 메소드명혹은 클래스+메소드를 지정할 수있다.

```
default: NONE
value:(CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
tx_server_ignore_method=access;com.Test.save
```

• tx_server_ignore_prefix

TX-SERVER(서비스 시작)에서 제외할 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
```

```
value:STRING{;STRING}*
tx_server_ignore_prefix=com.util;com.jennifersoft.util
```

• tx_server_interface

TX-SERVER(서비스 시작)를 위한 클래스들의 공통 인터페이스클래스를 지정한다.

```
default: NONE
value:CLASS{;CLASS}*
tx_server_interface=com.ITestClassA;org.ITestClassB
```

• tx_server_ntype

TX-SERVER(서비스 시작)에서 네이밍 타입을 지정한다.

```
default: FULL
value: FULL|CLASS|METHOD|SIMPLE
tx_server_ntype=CLASS
```

• tx_server_postfix

TX-SERVER(서비스 시작)를 위한 클래스들의 공통 POSTFIX를 지정한다.

```
default: NONE
value:STRING{;STRING}*
tx_server_postfix=Adapeter;VO;Action
```

• tx_server_prefix

TX-SERVER(서비스 시작) 를 위한 클래스들의 공통 PREFIX를 지정한다.

```
default: NONE
value:STRING{;STRING}*
tx_server_prefix=com;org.apache.common
```

• tx_server_super

TX-SERVER(서비스 시작)를 위한 클래스들의 공통 SUPER클래스를 지정한다.

```
default: NONE
value:CLASS{;CLASS}*
tx_server_super=com.TestSuperA;com.TestSuperB
```

• tx_server_target_method

TX-SERVER(서비스 시작)에 포함된 클래스에서 대상 메소드만을 지정한다.

Notice: 메소드명혹은 클래스+메소드를 지정할 수있다.

```
default: NONE
value:(CLASS_METHOD|METHOD){;(CLASS_METHOD|METHOD)}*
tx_server_target_method=access;com.Test.save
```

• tx_server_using_param

서비스명을 서비스 시작 메소드의 스트링 파라미터로 부여한다. 이 옵션이 true로 설정되면 tx_server_ntype은 무시된다.

```
default: false
value:BOOL
tx_server_using_param=true
```

• udp_server_host

제니퍼 서버가 기동중인 서버의 IP Address 이다. 해당 IP를 변경하면 실시간으로 반영된다.

```
default: 127.0.0.1
value:IP|HOSTNAME
udp_server_host = localhost
```

• uri_separator

일반 웹시스템에서 URL과 파라미터 사이의 구분은 '?' 가 사용된다.그러나 이동 단말기를 위한 시스템에서는 다른 문자가 사용될 수 있는데 이때 구분 문자를 설정한다.

```
default: NONE
value:CHAR
uri_separator=?
```

• uri_starter

제니퍼에이전트가 HttpRequest에서 URL을 추출할때 앞부분을 잘라내기 위해 설정한다.

```
default: NONE
value: STRING
uri_starter=/http
```

• url_additional_request_keys

MVC(Model-View-Controll) 패턴 등의 설계로 인해서 성격이 다른 사용자의 여러 요청이 하나의 URL로 들어오고 이를 하나의 컨트롤 서블릿 객체가 특정 HTTP파라미터 값을 이용해서 분기하는 구조라면 애플리케이션 트랜잭션을 URL만으로 표현하는 것은 부족하

다. 이 경우 해당 프레임워크에서 공유한 요청을 구분해 내는 HTTP 요청값이 있다면 그 키값을 해당 옵션에 지정한다. 이 경우 애플리케이션명은 URL+키 값 형태가 된다.

```
default: NONE
value:STRING{,STRING}*
url_additional_request_keys = key1,key2
```

• user_defined_jdbc_connectionpool_prefixes

JDBC SQL을 추적할 때, DriverManager.getConnection() 유형을 사용하는 경우에 설정할 수 있다. 여려개라면 (“,”)로 구분하여 기술한다.

```
default: NONE
value:STRING
user_defined_jdbc_connectionpool_prefixes=jdbc:apache:commons:dbcp
```

• user_defined_jdbc_ignore_close

user_defined_jdbc_connectionpool_prefixes사용하여 JDBC를 추적하면 Connection Not Close를 감지하지 않는다. 이것을 감지하려면 false로 설정해야 한다.

```
default: true
value:BOOL
user_defined_jdbc_ignore_close=true
```

• xview_profile_ignore_resp_time

응답시간이 옵션 값 미만의 트랜잭션의 프로파일 정보를 전송하지 않는 옵션이다. 응답시간이 상대적으로 매우 짧은 트랜잭션 보다는 응답시간이 오래 걸린 트랜잭션이 관심의 대상이 될 수 있으므로, 응답시간이 짧은 트랜잭션을 필히 분석할 필요가 없다면 제니퍼 서버로 전송하지 않는다. 따라서 제니퍼 서버의 부하도 줄일 수 있다.

```
default: 0
value:MILLISECOND
xview_profile_ignore_resp_time = 0
```

• xview_profile_udp_packet_size

제니퍼 에이전트는 트랜잭션 프로파일링 데이터를 UDP 패킷으로 제니퍼 서버에 보내는데, UDP패킷에는 64kbyte 의 제약이 있다. 이 경우 패킷을 여러개로 나누어 전송할 수 있는데 한번에 전송되는 UDP 패킷의 크기를 지정한다.

```
default: 32757
value:INT
xview_profile_udp_packet_size = 32757
```

• web_service_logfile

WAS의 애플리케이션 서비스 성능을 로깅하기 위한 파일명을 설정한다.

default: NONE

value:FILEPATH

web_service_logfile=service.log

16.3. 제니퍼 서버 옵션

• active_graph_interval

이 설정은 액티브 서비스 수를 화면에 디스플레이 할때 사용되는 그룹핑 기준이다.

Notice: 이값은 제니퍼 에이전트와 서버가 일치해야 한다.

default: 0,1000,3000,8000

value: MILLISECOND,MILLISECOND,MILLISECOND,MILLISECOND(오름차순 정렬,4개 숫자)

active_graph_interval = 0,1000,3000,8000

위 설정은 0-1,1~3, 3~8, 8+를 의미 한다.

• agent_group

에이전트 그룹을 정의한다. 그룹명은 반드시 '@' 이로 시작되고 2개의 숫자로 구성되어야 한다.따라서 가능한 그룹명은 @01~@99 이다.그룹명 뒤에 ':' 을 붙이고 해당되는 에이전트 명을 ',' 로 분리하여 기술한다. 그룹과 그룹사이는 ';' 로 구분한다

default: NONE

value:@99:AGENT{,AGENT}*{;@99:AGENT{,AGENT}*}*

agent_group = @01:W11,W12; @02:W13,W14

• active_service_alert_limit

액티브 서비스 큐잉 경고를 위한 기준값을 설정한다. 큐잉 카운트가 설정값을 초과하면 경보(ERROR_SERVICE_QUEUING) 가 발령된다.

default: 70

value:INT

active_service_alert_limit = 70

• adf_stream_buffer_size

REMON 에서 “STREAM” 타입의 데이터를 제니퍼 클라이언트에 서비스하기 위한 버퍼 사이즈를 설정한다 . 제니퍼 클라이언트가 데이터를 조회하는 시간간격 (2 초) 동안 버퍼링되는 데이터 크기이다 . 버퍼 사이즈는 REMON SCRIPT ID 별로 적용된다 .

Notice: 제니퍼 클라이언트의 이벤트 창에서 디스플레이되는 메세지 또한 이 버퍼 사이즈에 영향을 받는다.

default: 512

value:INT

adf_stream_buffer_size=512

• agent_death_auto_remove_time

정지된 에이전트를 제니퍼 서버의 관리목록에서 제거하기 위한 기준시간을 설정한다.

default: 3600000

value:MILLISECOND

agent_death_auto_remove_time = 86400000

• agent_death_detection_time

SUMMARY 데이터(6902)가 여기에 설정된 시간동안 전달되지 않으면 에이전트가 다운되었다고 판단한다.

default: 3000

value:MILLISECOND

agent_death_detection_time = 8000

• agent_tcp_connect_timeout

제니퍼 서버에 에이전트에 연결시 사용되는 연결 TIMEOUT 값이다.

default: 3000

value:MILLISECOND

agent_tcp_connect_timeout=3000

• agent_tcp_io_timeout

제니퍼 에이전트와 TCP 통신시 IO Time Out Limit를 설정한다.

default: 5000

value:MILLISECOND

agent_tcp_io_timeout = 5000

• bbs_type_list

제니퍼 서버에서 사용할 수있는 게시판 유형을 정의한다.

```
default: NONE
value:STRING{;STRING}*
bbs_type_list=biz:Business;rpt:Report;chat:Personal
```

• config_refresh_check_interval

해당 옵션 값 간격으로 제니퍼 서버 설정파일의 변경사항을 체크한다.

```
default: 3000
value:MILLISECOND
config_refresh_check_interval = 3000
```

• data_directory

데이터 디렉토리 패스를 설정한다.만약 데이터 디렉토리가 변경되면 이전 데이터를 모두 상실한다.윈도우 환경에서도 '\' 대신에 '/' 를 사용한다.

```
default:../data/
value: DIRPATH
data_directory = ../../data/file/
```

• debug_tcp

제니퍼 클라이언트 TCP연결 요청을 로깅한다.

```
default: false
value:BOOL
debug_tcp=false
```

• default_max_rows_for_report_item

보고서 생성시 하나의 아이템을 위한 최대 데이터 row수이다.이 값이 커지면 제니퍼 서버에 부하를 유발할 수있다.

```
default: 3600
value:INT
default_max_rows_for_report_item=3600
```

• disable_app_mapping_db

제니퍼 서버에서 Application과 SQL/TX간의 맵핑정보 저장하지 않도록 설정한다.기본값은 false이다.맵핑 정보의 활용도가 떨어지는 시스템에 대해서는 true로 설정할 것을 권고한다.

```
default: false
value:BOOL
disable_app_mapping_db=false
```

• domain_name

하나의 제니퍼 서버에서 여러개의 서버 혹은 여러개의 제니퍼 에이전트를 중앙에서 관장하여 모니터링한다. 그러나 경우에 따라서는 이러한 모니터링 대상 서버를 비지니스 시스템 그룹별로 구분하여 별도의 제니퍼 서버로 모니터링해야할 수도 있다. 이때 제니퍼 서버가 관장하는 단위를 도메인(domain)이라 부르며, 각 도메인을 구분할 수 있도록 속성을 지정한다.

```
default: SYS1
value:STRING
domain_name = SYS1
```

• enable_biz_monitor

타 시스템(ex ITSM)과 제니퍼 서버간 데이터 연동시 비즈니스 그룹 성능을 계산한다. 값이 false이면 업무별 성능정보를 타 시스템에 전달할 수없다. 타 시스템과 연계가 있는 경우에만 true로 설정할 것을 권고한다.

```
default: false
value:BOOL
enable_biz_monitor=false
```

• enable_including_standalone_statistics

제니퍼 에이전트는 WAS에서 웹 서비스트 혹은 다른 프로토콜의 서비스를 같이 모니터링할 수 있다. 그런데 웹이 아닌 다른 서비스에 대한 성능 통계를 전체 통계에서 제외하고자 할때 false로 설정한다.

```
default: true
value:BOOL
enable_including_standalone_statistics=true
```

• enable_logfile_daily_rotation

로그파일을 일자별로 생성하도록 한다.

```
default: false
value:BOOL
enable_logfile_daily_rotation=true
```

• enable_minimize_db

데이터베이스 저장을 최소화 한다. 이 옵션이 true로 설정되면 데이터베이스에 성능데이터가 저장되지 않는다. 제니퍼 서버 문제 해결을 위해서만 사용해야 한다.

```
default: false
value:BOOL
```

```
enable_minimize_db=false
```

• enable_remon_data_save

레몬 프로토콜로 전송되는 데이터에 대한 저장을 DISABLE할 수 있다.

```
default: true
value:BOOL
enable_remon_data_save=true
```

• enable_server_trace

제니퍼 서버에서 수행되는 내부 SQL의 성능이나 에러를 로깅한다.

```
default: false
value:BOOL
enable_server_trace=false
```

• enable_visit_user_added_while_reloading

이값이 true로 설정되면 제니퍼서버가 재기동될때 현재의 시간당/일자별 방문자 수에 이전(제니퍼 서버 재기동 전) 시간당/일자별 방문자 수를 합산한다. 쇼핑몰 처럼 불특정 유저가 사용하는 시스템에서만 true를 설정한다.

```
default: false
value:BOOL
enable_visit_user_added_while_reloading=false
```

• enable_xview_data_logging

X-View를 위한 트랜잭션별 성능데이터 저장 여부를 결정한다. 제니퍼 서버의 문제 해결을 위해서만 사용(false로 설정)해야 한다.

```
default: true
value:BOOL
enable_xview_data_logging=true
```

• high_rate_failed_alert_limit

서비스 실패율 경고를 위한 기준값(%)을 설정한다. 실패율이 지정한 퍼센트를 초과하면 경보(ERROR_HIGH_RATE_FAIL)를 발령한다.

Notice: If Service Rate is 3 tps or higher, above rule is ignored.

```
default: 50
value:INT
high_rate_failed_alert_limit = 50
```

• high_rate_reject_alert_limit

PLC에 의해 REJECT되는 서비스 율에 대한 경고를 설정한다. REJECT되는 서비스율이 설정값을 초과하면(ERROR_HIGH_RATE_REJECT) 경보가 발령된다.단 서비스 처리량이 최소 3TPS 이상인 경우에만 이 규칙이 적용된다.

```
default: 50
value:INT
high_rate_reject_alert_limit = 50
```

• ignore_unallowed_alert

ALERT설정에서 true가 아닌 경보는 발령하지 않는다.(무시된다)

```
default: false
value:BOOL
ignore_unallowed_alert=false
```

• jvm_cpu_alert_limit

JVM CPU사용량 경고를 위한 기준값을 설정한다. 사용량이 설정값을 초과하면 경보(ERROR_JVM_CPU_HIGH_LONGTIME)가 발령된다.

```
default: 90
value:INT
jvm_cpu_alert_limit = 90
```

• jvm_cpu_warning_limit

JVM CPU사용량 경고를 위한 기준값을 설정한다. 사용량이 `jvm_cpu_alert_limit` 미만이면서 설정값을 초과하면 경보(WARNING_JVM_CPU_HIGH)가 발령된다.

```
default: 90
value:INT
jvm_cpu_warning_limit = 90
```

• jvm_heap_check_time

jvm_heap_warning_limit에서 지정한 사용량 초과 상태가 유지되는 시간(초)을 지정한다. 조건에 해당되면 WARNING_JVM_HEAP_MEM_HIGH이라는 경보가 발령된다

```
default: 0
value:MILLISECOND
jvm_heap_check_time=300
```

• jvm_heap_warning_interval

경고가 발생하는 시간 간격(초)을 설정한다.

```
default: 0
value:MILLISECOND
jvm_heap_check_warning_interval=3600
```

• jvm_heap_warning_limit

힙 메모리 사용량(%)을 지정한다.

```
default: 0
value:INT
jvm_heap_warning_limit=95
```

• logfile

제니퍼 서버 로그파일(파일명 포함) 위치를 지정한다.

Notice: Windows환경에서도 '\' 대신에 '/'를 사용한다.

```
default: jennifer.log
value:FILEPATH
logfile = ../logs/jennifer.log
```

• logfile_encoding_characterset

제니퍼 서버가 로그를 남길 때, 로그파일의 인코딩 형식을 지정한다. 일반적으로는 제니퍼 서버는 OS 기본 인코딩을 사용하므로 해당 옵션을 사용하지 않는다.

```
default: NONE
value:STRING
logfile_encoding_characterset = Cp933
```

• number_of_tcp_pooled_workers

동시에 처리할 수있는 제니퍼 클라이언트(애플릿)에서의 연결요청 수를 설정한다. 한 클라이언트당 동시에 2개이상의 연결을 요청한다.

Notice: 이 값이 바뀌면 제니서 서버를 재기동해야 한다.

```
default: 80
value:INT
number_of_tcp_pooled_workers = 80
```

• number_of_udp_callstack_workers

제니퍼 에이전트로 부터 전송되는 프로파일 데이터(6703)를 처리하기 위한 쓰레드 수를 설정한다.

Notice: 이 값이 바뀌면 제니서 서버를 재기동해야 한다.

default: 30

value: INT

number_of_udp_callstack_workers = 30

• number_of_udp_listen_workers

제니퍼 에이전트로 부터 전송되는 SUMMARY데이터(6902)를 처리하기 위한 쓰레트 수를 설정한다.

Notice: 이 값이 바뀌면 제니서 서버를 재기동해야 한다.

default: 10

value: INT

number_of_udp_listen_workers = 10

• number_of_udp_runtime_workers

제니퍼 에이전트로 부터 전송되는 RUNTIME데이터(6901)를 처리하기 위한 쓰레스 수를 설정한다.

Notice: 이 값이 바뀌면 제니서 서버를 재기동해야 한다.

default: 10

value: INT

number_of_udp_runtime_workers = 10

• perf_host_update_interval

이 옵션은 PERF_HOST 테이블의 저장 간격이다.

default: 300000

value: MILLISECOND

perf_host_update_interval = 300000

• perf_x_update_interval

이 옵션은 PERF_X 테이블의 저장 간격이다.제니퍼 통계 그래프에서 표시하는 데이터는 5분 기준이다. 따라서 이값이 1분으로 수정되어도 5분 단위 데이터만 그래프에서 사용된다. 1분 데이터는 템플릿 보고서을 사용하거나 쿼리 수행기를 통해서 확인해야 한다.

default: 300000

value: MILLISECOND

perf_x_update_interval = 300000

• remon_addr

제니퍼 화면 컴포넌트 중에는 레몬으로 접속하여 데이터를 조회하는 기능이 있다. 일반적으로 각 레몬 프로세스 IP주소는 제니퍼 서버에 의해 자동으로 감지되지만 네트웍 구성에 따라서는 이것이 불가능할 수도 있다. 이때 레몬 프로세스의 주소를 명시적으로 기술할 수있다. 여러개의 주소를 설정할 수 없다.

```
default: NONE
value: HOST:PORT
remon_addr=192.168.10.1:7701
```

• remon_debug

REMON데이터를 디버깅할때 true로 설정한다.

```
default: false
value: BOOL
remon_debug=false
```

• report_image_height

SQL콘솔이나 웹리포팅에서 그래프의 세로 크기를 설정한다.

```
default: 200
value: INT
report_image_height=200
```

• report_image_width

SQL콘솔이나 웹리포팅에서 그래프의 가로 크기를 설정한다.

```
default: 800
value: INT
report_image_width=800
```

• server_jdbc_trace_overthan

설정된 시간 이상의 SQL(제니퍼 서버내부) 수행 지연이 발생하면 로깅한다.

```
default: 10000
value: MILLISECOND
server_jdbc_trace_overthan=10000
```

• server_tcp_port

UDP 포트와는 별도로, 관리자의 브라우져의 Applet기반의 GUI Viewer와 통신하는 제니퍼 서버의 TCP 포트다. Applet은 주기적으로 제니퍼 서버의 해당 포트를 통해 데이터를 가져가게 된다.

```
default: 6701
value: PORT
server_tcp_port = 6701
```

• server_trace_filename

제니퍼 서버내부에 수행되는 SQL성능을 로깅하기 위한 파일명을 설정한다.

```
default: servertrace.log
value: FILEPATH
server_trace_filename=servertrace.log
```

• server_udp_listen_port

실시간이 아닌 주기적(1초)으로 Agent로 부터 넘어오는 통계 데이터를 취합하는 UDP 포트다. 제니퍼 독립에이전트(wmond, remon)가 보내주는 데이터도 해당 포트로 취합된다.

```
default: 6902
value: PORT
server_udp_listen_port = 6902
```

• server_udp_lwst_call_stack_port

Agent로부터 전송된 실시간 Call Stack 정보를 취합하는 UDP 포트다.

```
default: 6703
value: PORT
server_udp_lwst_call_stack_port = 6703
```

• server_udp_runtime_port

사용자 요청이 발생할때마다 넘어오는 실시간 데이터를 전송하는 UDP 포트다. 이 값은 각 Agent 구성파일의 것과 동일해야 한다.

```
default: 6901
value: PORT
server_udp_runtime_port = 6901
```

• sms_adapter_class_name

Sendsms어뎁터 클래스를 설정한다. 어뎁터 클래스는 제니퍼서버가 기동되기 전에 JENNIFER_HOME/server/common/lib에 jar파일 형태로 배포되어 있어야 한다.

```
default: 6902
value: CLASS{;CLASS}*
sms_adapter_class_name = com.javaservice.jennifer.server.SMSExample
```

• sms_alert_minimal_interval

SMS 메세지 처리 interval을 설정한다. Sendsms어뎁터는 여기에 설정된 간격으로 trigger 된다.

```
default: 60000
value: MILLISECOND
sms_alert_minimal_interval = 1000
```

• specified_agent_ip

에이전트의 실제 IP주소와 서버에서 에이전트로 연결할 때 사용해야 하는 IP 주소가 다를 때 해당 에이전트의 IP주소를 이 설정에 명시한다.설정 방법은 아래 예를 참조한다

```
default: NONE
value: AGENT:IP{,AGENT:IP}*
specified_agent_ip = W11:192.168.10.101, W12:192.168.10.102
```

• supported_language_list

제니퍼 서버가 지원하는 언어의 목록이다. 현재 영어, 한국어, 일본어를 지원하고 있다.

```
default: en,ko,ja,zh,fr
value: STRING{,STRING}*
supported_language_list = en,ko,ja,zh,fr
```

• sys_cpu_alert_limit

시스템 CPU사용량 경고를 위한 기준값을 설정한다. 사용량이 설정값을 초과하면 경보(ERROR_SYSTEM_CPU_HIGH_LONGTIME)가 발령된다.

```
default: 90
value: INT
sys_cpu_alert_limit = 95
```

• sys_cpu_warning_limit

시스템 CPU사용량 경고를 위한 기준값을 설정한다. 사용량이 sys_cpu_alert_limit 미만이면서 설정값을 초과하면 경보(WARNING_SYSTEM_CPU_HIGH)가 발령된다.

```
default: 95
value: INT
sys_cpu_warning_limit = 95
```

• system.derby.system.home

제니퍼 데이터 베이스의 위치를 설정한다

```
system.derby.system.home = ../../data/database
```

• system.derby.drda.startNetworkServer

제니퍼 데이터 베이스에 대한 네트웍 접속을 허용한다. 제니퍼 서버가 기동되기 전에 설정되어야 한다.

Notice: 제니퍼서버 기동 쉘(catalina.sh)에 -Dderby.drda.startNetworkServer=true라고 설정할 수도 있다.

```
default: false
value: BOOL
system.derby.drda.startNetworkServer = true
```

• system.derby.drda.portNumber

system.derby.drda.startNetworkServer = true인 상태에서 제니퍼 데이터 베이스의 서비스 포트를 설정한다.제니퍼 서버가 기동되기 전에 설정되어야 한다.

Notice: 제니퍼서버 기동 쉘(catalina.sh)에 -Dderby.drda.portNumber=1527라고 설정할 수도 있다.

```
default: NONE
value: PORT
system.derby.drda.portNumber = 1527
```

• udp_server_host

제니퍼 서버가 UPD로 바인딩할 서버의 IP Address 다. 서버에 하나 이상의 N/W 카드가 있다면, 해당 N/W카드로 들어오는 요청만 바인딩한다.

```
default: localhost
value:HOST
udp_server_host = 0.0.0.0
```

• unspecified_alert_is_allowed

ALERT설정에서 명시되지 않은 ALERT은 true로 설정된다.

```
default: false
value: BOOL
unspecified_alert_is_allowed=false
```

표 16-10: 제니퍼 서버의 ALERT 설정

- **ALERT Mesage**

```
SYSTEM_MESSAGE=true
USER_DEFINDED_MESSAGE=true
```

- **Alert Critical**

```
ERROR_SYSTEM_DOWN=false
ERROR_JVM_DOWN=true
ERROR_OUTOFMEMORY=true
ERROR_HIGH_RATE_REJECT=true
ERROR_HIGH_RATE_FAIL=false
ERROR_SERVICE_QUEUING=true
ERROR_SYSTEM_CPU_HIGH_LONGTIME=false
ERROR_JVM_CPU_HIGH_LONGTIME=false
USER_DEFINED_FATAL=false
```

- **Alert Error**

```
ERROR_HTTP_IO_EXCEPTION=false
ERROR_UNCAUGHT_EXCEPTION=false
ERROR_RECURRSIVE_CALL=false
ERROR_PLC_REJECTED=false
ERROR_JDBC_CONNECTION_FAIL=false
ERROR_MAYBE_GC_TIME_DELAY=false
ERROR_UNKNOWN_ERROR=false
USER_DEFINED_ERROR=false
ERROR_LOGICAL_PROCESS=false
```

- **Alert Warning**

```
WARNING_JDBC_CONN_UNCLOSED=false
WARNING_JDBC_STMT_UNCLOSED=false
WARNING_JDBC_PSTMT_UNCLOSED=false
WARNING_JDBC_CSTMT_UNCLOSED=false
WARNING_JDBC_RS_UNCLOSED=false
WARNING_JDBC_TOOMANY_RS_NEXT=false
WARNING_JDBC_STMT_EXCEPTION=false
WARNING_JDBC_PSTMT_EXCEPTION=false
WARNING_JDBC_BAD_RESPONSE=false
WARNING_TX_CALL=false
WARNING_TX_BAD_RESPONSE=false
WARNING_APP_BAD_RESPONSE=false
```

```
WARNING_JDBC_UN_COMMIT_ROLLBACK=false
WARNING_JDBC_CONN_ILLEGAL_ACCESS=false
WARNING_ORACLE_XMLTYPE_UNCLOSED=false
USER_DEFINED_WARNING=false
WARNING_CUSTOM_RUNTIME_EXCEPTION=false
WARNING_CUSTOM_EXCEPTION=false
WARNING_CUSTOM_THROWABLE=false
WARNING_RESOURCE_LEAK=false
WARNING_JVM_HEAP_MEM_HIGH=false
WARNING_SYSTEM_CPU_HIGH=false
WARNING_JVM_CPU_HIGH=false
```

• upload_directory

제니퍼서버 게시판에 파일을 업로드할 때 사용할 디렉토리를 지정한다.

```
default: ../../data/upload
value: DIRPATH
upload_directory = ../../data/upload
```

• user_defined_jdbc_ignore_close

JDBC 추적 유형2(DriverManager) 추적시 Connection close감지 여부를 결정한다. 값이 true이면 close를 감지하지 않으며, Connection이 애플리케이션에 할당 되어도 상태가 ALLOCATED로 변경되지 않는다.

- default: true
- value: BOOL
- user_defined_jdbc_ignore_close = true

16.3.1.xvdaily_agents

일자별 X-View프로파일 정보를 저장할 에이전트를 선별 할 수있다.

- default: NONE
- value: AGENT{,AGENT}*
- xvdaily_agents=W11,W12

16.3.2.xvdaily_enable

일자별 X-View 프로파일 데이터 저장 여부를 결정한다. 기본값은 true이다.

- default: true
- value: BOOL
- xvdaily_enable=true

16.3.3.xvdaily_ignore_resp_time

일자별 X-View 프로파일 데이터를 저장하는 기준시간을 설정한다. 여기에 설정된 시간미만의 응답 트랜잭션의 프로파일 정보는 서버에 저장되지 않는다.

- default: 0
- value: MILLISECOND
- xvdaily_ignore_resp_time = 500

16.3.4.xview_point_ignore_resp_time

응답시간이 옵션 값 미만의 트랜잭션에 대해서는 X-View 데이터 파일에 성능 정보를 저장하지 않는다.

- default: 0
- value:MILLISECOND
- xview_point_ignore_resp_time = 0

16.3.5.xview_profile_ignore_resp_time

여기에 설정한 값보다 적은 응답시간을 갖는 트랜잭션의 프로파일 데이터는 무시(저장 않함)된다.

- default: 0
- value: MILLISECOND
- xview_profile_ignore_resp_time = 0

16.3.6.xview_profile_cache_queue_size

프로파일 정보를 저장하기 위해 임시 보관하는 CACHE 크기이다.

- default: 512
- value: INTEGER
- xview_profile_cache_queue_size=512

16.3.7.xview_profile_isam_file_max_size

실시간 X-View 프로파일 데이터를 저장하기 위한 파일 사이즈를 설정한다.

Notice: Support Unit: kb,mb,gb units.

```
default: 0
value: 999kb|999mb|999|gb
xview_profile_isam_file_max_size = 512mb
```

16.3.8.xview_profile_multi_packet_check_interval

X-View 프로파일이 멀티패킷으로 전달될때 모든 패킷이 도착했는지 검사하는 주기를 설정한다.

- default: 2000
- value: MILLISECOND
- xview_profile_multi_packet_check_interval=2000

16.3.9.xview_profile_multi_packet_queue_size

X-View 프로파일이 멀티패킷으로 전달될때 받아들이는 큐사이즈를 설정한다.

- default: 303
- value: INT
- xview_profile_multi_packet_queue_size = 103

16.3.10.xview_profile_multi_packet_time_out

X-View 프로파일이 멀티패킷으로 전달될때 모든 패킷이 도착할 때까지지 기다리는 시간을 설정한다.

- default: 1000
- value: MILLISECOND
- xview_profile_multi_packet_time_out=1000

16.3.11.xview_server_queue_size

서버에 받은 프로파일을 저장할 때 사용하는 큐 사이즈를 설정한다.

- default: 512
- value: INT
- xview_server_queue_size = 512

16.4. 제니퍼 에러 코드

제니퍼 에이전트와 서버에서 발생할 수 있는 에러 리스트이다. 다음 설명은 에러가 발생한 상황이나 위치에 대한 설명이다.

16.4.1.에이전트 에러 코드

제니퍼 에이전트가 실행되면서 발생한 에러코드이다. 제니퍼 에이전트 로그파일에 기록된다.

표 16-11: 에이전트 에러 코드

코드	설명
B003	AgentDataCollector에서 5분 평균 성능 데이터를 생성하면서 발생한 에러
B004	5분 평균 성능 데이터와 사용자, 처리량 데이터를 저장하면서 발생한 시스템 에러
B005	.data/.thrput_xxx파일에 처리량 정보를 저장하면서 발생한 에러
B006	.data/.thrput_xxx파일에 처리량 정보를 로딩하면서 발생한 에러

표 16-11: 에이전트 에러 코드

코드	설명
B007	.data/.user_xxx파일에 사용자 관련 통계 정보를 저장하면서 발생한 에러
B008	.data/.user_xxx파일에 사용자 관련 통계 정보를 로딩하면서 발생한 에러
B009	제니퍼 에이전트 Config파일을 오픈하지 못한 에러
B010	HttpService를 추적하면서 과도한 재귀 호출이 발생하여 남겨지는 로그 단, recurrsive_call_trace=true가 제니퍼 에이전트에 설정되어 있어야 로그가 남는다.
B011	SessionTrace에서 세션 객체를 참조하면서 발생한 에러
B012	System.exit() 호출되었다는 알림 메시지
B013	JSP/Servlet을 수행하던 쓰레드에 의해서 Syste.exit()가 호출되었다는 경고 메시지
B014	TcpAgentWorker가 Active List를 제니퍼 서버에 전송하면서 발생한 에러
B015	TcpAgentWorker가 Active Bad List를 제니퍼서버에 전송하면서 발생한 에러
B016	Agent의 Active Thread에 새로운 우선순위(Priority)를 Set하면서 발생한 에러
B017	Active Thread에 interrupt()를 호출하면서 발생한 에러
B018	Active Thread에 suspend()를 호출하면서 발생한 에러
B019	Suspend된 Thread에 대해 resume()를 호출하면서 발생한 에러
B020	Thread Stop이 disable상태이어서 stop()을 수행할 수 없다는 안내메세지
B021	Active Thread에 stop()를 호출하면서 발생한 에러
B022	Active Service의 상세 수행상태를 제니퍼서버에 전송하면서 발생한 에러
B023	Application 수행 통계(실시간)를 서버로 전송하면서 발생한 에러
B024	SQL및 외부 TX를 포함한 Application 상세 수행 통계(실시간)을 서버로 전송하면서 발생한 에러
B025	SQL수행 통계(실시간) 정보를 제니퍼 서버로 전송하면서 발생한 에러
B026	Application 맵핑 정보를 포함한 SQL 수행 상세 통계(실시간)를 제니퍼 서버로 전송하면서 발생한 에러
B027	외부 TX 수행 통계(실시간) 정보를 제니퍼 서버로 전송하면서 발생한 에러
B028	Application 맵핑 정보를 포함한 외부 TX 수행 상세 통계(실시간)를 제니퍼 서버로 전송하면서 발생한 에러
B029	에러발생 통계(실시간) 정보를 제니퍼 서버로 전송하면서 발생한 에러
B030	Application 맵핑 정보를 포함한 에러 발생 상세 통계(실시간)를 제니퍼 서버로 전송하면서 발생한 에러
B031	매 지정한 시간(기본 10분)마다 수집하는 제니퍼 에이전트 통계 정보를 제니퍼 서버로 전송하면서 발생한 에러. 단, 이곳에서 에러가 발생하면 애플리케이션 관련한 통계 정보가 달라질 수 있다.
B032	Agent가 제니퍼 서버로 부터 요청을 받아 정보를 리턴하는 과정에서 발생하는 에러

표 16-11: 에이전트 에러 코드

코드	설명
B033	제니퍼 에이전트가 기동되는데 -Dconfig.auto=true가 설정되어있음을 알리는 메시지
B034	auto config가 설정된 상태에서 다음 Config파일을 초기화하면서 발생한 에러
B035	제니퍼 서버가 에이전트로 연결하는 TCP 세션이 256개를 초과했다는 알림 메시지로 해당 세션은 바로 close된다.
B036	제니퍼 에이전트가 TCP 포트에서 리스닝하면서 발생한 에러
B037	UDP Destination host를 Set하면서 발생한 에러
B038	Service Arrival 정보를 보내면서 발생한 IO 에러
B039	Service Arrival 정보를 보내면서 발생한 일반 에러
B040	Service End정보를 보내면서 발생한 IO 에러
B041	Service End정보를 보내면서 발생한 일반 에러
B042	Http Service가 아닌 다른 서비스(ex TCP데몬)에 대해서 Service End 정보를 보내면서 발생한 IO 에러
B043	Http Service가 아닌 다른 서비스(ex TCP데몬)에 대해서 Service End 정보를 보내면서 발생한 일반 에러
B044	Agent의 초단위 성능 정보를 제니퍼 서버에 전송하면서 발생하는 IO 에러
B045	Agent의 초단위 성능 정보를 제니퍼 서버에 전송하면서 발생하는 일반 에러
B046	프로파일 정보를 제니퍼 서버로 전송하면서 발생한 IO 에러
B047	프로파일 정보를 제니퍼 서버로 전송하면서 발생한 일반 에러
B048	대용량 프로파일 정보를 제니퍼 서버로 전송하면서 발생한 IO 에러
B049	대용량 프로파일 정보를 제니퍼 서버로 전송하면서 발생한 일반 에러
B050	DB모니터링 툴에게 Tx수행 정보를 전송하면서 발생한 IO 에러
B051	DB모니터링 툴에게 Tx수행 정보를 전송하면서 발생한 일반 에러
B052	에이전트에서 확장된 모니터링 정보를 전송하면서 발생하는 IO 에러
B053	에이전트에서 확장된 모니터링 정보를 전송하면서 발생하는 일반 에러
B054	HTTP Cookie에서 사용자 KEY(WMON ID)를 조회하면서 발생한 에러
B055	사용자 KEY(WMON ID)를 생성해서 HTTP Cookie에 셋팅 하면서 발생한 에러
B056	제니퍼 에이전트를 위한 JNI 라이브러리를 초기화하면서 발생한 에러
B057	JDBC추적 부분에서 enable_jdbc_oracle_dependency_used = true가 SET되어 있는데, 클래스패스에서 Oracle JDBC Driver를 찾을 수 없을 때 발생하는 에러
B058	JDBC추적 부분에서 enable_jdbc_oracle_dependency_used = true가 SET되어 있는데, 초기화 과정에서 발생한 일반 에러

표 16-11: 에이전트 에러 코드

코드	설명
B059	JDBC 추적 부분에서 enable_jdbc_oracle_dependency_used = true가 SET되어 있는데, LWST모듈이 로딩되지 않았다는 에러
B060	JDBC추적 부분에서 enable_jdbc_oracle_dependency_used = true가 SET되어있는데, OracleConnection을 추적하기 위한 클래스를 로딩할 수 없다는 에러
B061	JDBC추적 부분에서 enable_jdbc_oracle_dependency_used = true가 SET되어있는데, OracleConnection을 추적하기 위한 클래스를 로딩화는 과정에 발생한 Securit y 에러
B062	JDBC추적 부분에서 enable_jdbc_oracle_dependency_used = true가 SET되어있는데, OracleConnection을 추적하기 위한 클래스를 로딩화는과정에 발생한 일반 에러
B063	CallableStatement를 close하면서 발생한 에러
B064	OracleCallableStatement를 close하면서 발생한 에러
B065	Oracle Dependency 상태에서 CallableStatement을 추적하면서 발생하는 에러
B066	Connection close()을 호출하면서 발생한 에러
B067	Oracle Dependency 상태에서 Connection을 close하지 않았다는 알림 메시지
B068	Oracle Dependency 상태에서 Oracle Connection을 추적하면서 발생하는 에러
B069	Oracle Dependency 상태에서 Oracle Connection을 추적하면서 발생하는 에러(2)
B070	Oracle Dependency 상태에서 PreparedStatement을 추적하면서 발생하는 에러
B071	ResultSet을 close하지 않았다는 알림 메시지
B072	Oracle Dependency 상태에서 ResultSet을 close하지 않았다는 알림 메시지
B073	Oracle Dependency 상태에서 PreparedStatement을 close하지 않았다는 알림 메시지
B074	Oracle Dependency 상태에서 ResultSet을 추적하면서 발생하는 에러
B075	Oracle Dependency 상태에서 Statement을 close하지 않았다는 알림 메시지
B076	Oracle Dependency 상태에서 Statement을 추적하면서 발생하는 에러
B077	Statement가 closer되지 않았다는 알림 메시지
B078	Start Trace에서 발생한 에러
B079	End Trace에서 발생한 에러
B080	Extra Agent 모듈을 실행하면서 발생한 에러
B081	agent db를 위한 rootpath에 쓰기권한이 없거나 경로가 잘못되서 발생하는 에러
B082	agent db의 header파일에 대한 오픈 에러
B083	agent db의 data/index파일에 대한 오픈 에러
B084	agent db에서 데이터를 검색하면서 발생한 에러
B085	DB STAT모니터링시 초기화 과정에서 발생한 에러

표 16-11: 에이전트 에러 코드

코드	설명
B086	DB STAT모니터링시 초기화 과정에서 발생한 에러
B087	Extra Agent 메세지를 서버에 전송하면서 발생한 에러
B088	DB SID를 조회하면서 발생한 에러

16.4.2.제니퍼에이전트 LWST 에러 코드

제니퍼 Byte Code Instrumentation모듈인 LWST에서 발생할 수 있는 에러 코드이다. 해당 에러가 발생하면 lwst log파일에 기록된다.

표 16-12: LWST 에러 코드

코드	설명
W001	클래스로더에서 클래스 패스 정보를 추출하면서 발생한 에러.
W002	클래스로더에서 클래스 패스 정보를 추출하면서 발생한 에러.
W003	클래스로더에서 클래스 패스 정보를 추출하면서 발생한 에러.
W004	지정한 패스에서 클래스 바이너리를 로딩하면서 발생한 에러
W005	클래스 코드뷰 기능을 위해 Code를 (binary -> text)역으로 생성하면서 발생한 에러
W006	Connection Pool을 추적하기 위한 설정을 해석하면서 발생한 에러
W007	Leak Checker의 설정 정보를 초기화하면서 발생한 에러
W008	Leak Check항목이 10개를 넘었다는 에러 메세지
W009	LWST Class Patch 모듈이 대상 클래스를 로딩하면서 발생한 에러
W010	CUSTOM-TRACE를 위한 대상 구현 클래스를 로딩하면서 발생한 에러
W011	제니퍼의 Class Patch기능을 수행하는데 로딩된 클래스의 이름을 해석할 수 없을 때 발생한 에러
W012	제니퍼의 Class Patch기능을 수행하는데 로딩된 클래스의 이름이 원본 클래스와 이름이 다른 경우에 발생한 에러
W013	제니퍼 LWST모듈이 클래스 Byte Code를 제어하면서 발생한 에러. 이 에러가 발생하면 해당 클래스에 대한 Instrumentation을 취소하고 Byte코드를 원본으로 리턴한다.
W014	클래스의 크기가 너무 커서 instrumentation을 수행하지 않았다는 안내 메시지
W016	해당 클래스의 String 타입의 파라미터가 없어서 파라미터 추적을 하지 못한다는 에러
W017	파라미터 값을 추적하기 위한 Instrumentation 도중에 발생한 에러
W018	String 타입의 리턴이 아니기 때문에 메소드의 리턴값을 추적할 수 없다는 에러메세지

표 16-12: LWST 에러 코드

코드	설명
W024	TX-CLIENT에서String 타입의 파라미터가 없어서 파라미터값을 이용한 외부 트랜잭션 추적을 할 수 없다는 에러
W025	TX_CLIENT에 해당하는 메소드에 대해서 instrumentation을 수행하다가 에러가 발생
W026	TX-NAMING에서String 타입의 파라미터가 없어서 파라미터 값을 이용한 서비스 Naming을 할 수 없다는 에러
W028	TX-NAMING에서String 타입의 리턴이 아니기 때문에 리턴 값을 이용한 서비스 Naming을 할 수 없다는 에러
W032	GUID추출부에서 발생한 에러
W033	GUID추출부에서 발생한 에러
W034	GUID 추출 대상을 초기화 하면서 발생한 에러
W036	TX-CLIENT의 이름을 결정하기 위한 메소드 파라미터를 hooking하면서 발생한 에러
W037	ALSB추적 설정시 발생한 에러
W038	ALSB추적 설정시 발생한 에러
W039	TX-SERVER에서String 타입의 파라미터가 없어서 파라미터값을 이용한 외부 트랜잭션 추적을 할 수 없다는 에러
W040	TX-SERVER에 해당하는 메소드에 대해서 instrumentation을 수행하다가 에러가 발생

16.4.3.제니퍼 서버 에러 코드

제니퍼 서버가 동작하면서 발생하는 비정상적인 상황에 대한 코드이다. 제니퍼 서버 로그 파일에 기록된다..

표 16-13: 제니퍼 서버 에러 코드

코드	설명
S010	AGENT ID를 저장하면서 발생한 에러
S011	AGENT에 대한 그룹ID를 조회하면서 발생한 에러
S012	ALERT 정보를 저장하면서 발생한 에러
S013	ALERT 정보를 저장하면서 발생한 에러
S014	ALERT 정보를 저장하면서 발생한 에러
S015	Application 성능정보를 저장하기 위한 모듈을 초기화하면서 발생한 에러
S016	Application 성능정보를 저장하면서 발생한 에러
S017	제니퍼 데이터 존재하는지 검사하면서 발생한 에러

표 16-13: 제니퍼 서버 에러 코드

코드	설명
S018	PERF_X테이블이 분할되어있는지 확이하면서 발생한 에러
S019	ADF 테이블이생성되어있는지 확인하면서 발생한 에러
S020	ADMIN데이터베이스가 존재하는지 확인하면서 발생한 에러
S021	기존 테이블중에서 ADMIN을 위한 테이블을 이동가능한지 검사하면서 발생한 에러
S022	데이터베이스를 생성하면서 발생한 에러
S023	테이블을 생성하면서 발생한 에러
S024	테이블을 셍성하면서 발생한 일반 에러
S025	PERF_X테이블을 생성하면서 발생한 에러
S026	PERF_X테이블을 생성하면서 발생한 일반 에러
S027	Jennfier32버전의 테이블을 생성하면서 발생한 에러
S028	Jennfier33버전의 테이블을 생성하면서 발생한 일반 에러
S029	도메인명을 등록하면서 발생한 에러
S030	Error 정보를 저장하기 위한 모듈을 초기화 하면서 발생한 에러
S031	Error 정보를 저장하면서 발생한 에러
S032	Application명,SQL,Error,외부 Tx명등의 Text정보를 저장하면서 발생한 에러
S033	WMOND로수집되는 Host IP정보를 저장하면서 발생한 에러
S034	WMOND로수집되는 Host IP정보를 저장하면서 발생한 에러
S035	PERF_HOST에 CPU데이터를 저장하면서 발생한 에러
S036	PERF_TOT에 데이터를 저장하면서 발생한 에러
S037	PERF_X_99에 데이터를 저장하면서 발생한 에러
S038	PERF_X에서 Runtime 데이터를 조회하면서 발생한 에러
S039	SQL_10M_99 테이블에 데이터를 저장하기 위한 모듈을 초기화하면서 발생한 에러
S040	SQL_10M_99 테이블에 데이터를 저장하면서 발생한 에러
S041	SQL파일에서 데이터를 로딩하면서 발생한 에러
S042	DB에서 통계 데이터를 로딩하면서 발생한 에러
S043	TX_10M_99 테이블에 데이터를 저장하기 위한 모듈을 초기화하면서 발생한 에러
S044	TX_10M_99 테이블에 데이터를 저장하면서 발생한 에러
S045	시간당 방문자 정보를 저장하면서 발생한 에러
S046	일자별 방문자 정보를 저장하면서 발생한 에러
S047	시간당 전체 방문자 정보를 저장하면서 발생한 에러

표 16-13: 제니퍼 서버 에러 코드

코드	설명
S048	일자별 전체 방문자 정보를 저장하면서 발생한 에러
S052	SNMP 데이터를 생성하면서 0보다 적은 수가 발생했다는 에러
S053	일자별 X-View프로파일 정보를 저장하면서 발생한 에러
S054	일자별 X-View프로파일 정보를 일자별 파일에 저장하는데 일자가 변경되었는지 검사 하면서 발생한 에러
S055	XVD파일을 읽기모드로 오픈하면서 발생한 에러
S056	XVD파일을 쓰기모드로 오픈하면서 발생한 에러
S057	XVW(X-View 점데이터)파일에서 데이터를 로딩하면서 발생한 에러
S058	XVD에 대한 인덱스 정보를 생성하면서 발생한 에러
S059	XVD를 위해 생성한 인덱스 량에 대한 로그
S060	XVX데이터를 로딩하면서 발생한 에러
S061	XVX에이터를 새로 생성하면서 발생한 에러
S062	XVX(XV일자별 프로파일 인덱스)데이터를 저장하면서 발생한 에러
S063	XVX(XV일자별 프로파일 인덱스)데이터를 로딩하면서 발생한 에러
S064	XVC(XV커스텀)의 점 데이터를 저장하면서 발생한 에러
S065	XVC 프로파일 정보를 저장하기 위한 파일을 읽기모드로 열면서 발생한 에러
S066	XVC 프로파일 정보를 저장하기 위한 파일을 쓰기모드로 열면서 발생한 에러
S067	XVC프로파일 관리모듈을 초기화하면서 발생한 에러
S068	XVC프로파일 관리모듈의 초기화 결과를 알리는 메시지
S069	에이전트가 다운되었거나, 그와 유사한 이유로 데이터가 전송되지 않고 있음
S070	에이전트에 대한 TCP연결이 실패하여 발생한 에러
S071	에이전트에 대한 다운여부를 검사하면서 발생한 에러
S072	SMS 핸들러 클래스를 로딩하면서 발생한 에러
S073	SMS 핸들러의 sendsms(msg)를 호출하면서 발생한 에러
S074	ALERT데이터 저장을 제어하면서 발생한 에러
S075	ADF데이터 저장을 제어하면서 발생한 에러
S076	PERF_HOST 데이터 저장을 제어하면서 발생한 에러
S077	PERF_X 데이터 저장을 제어하면서 발생한 에러
S078	시간당 방문자 데이터 저장을 제어하면서 발생한 에러
S079	일자별 방문자 데이터 저장을 제어하면서 발생한 에러

표 16-13: 제니퍼 서버 에러 코드

코드	설명
S080	전체 PERF_X데이터 저장을 제어하면서 발생한 에러
S081	전체 시간당 방문자 데이터 저장을 제어하면서 발생한 에러
S082	전체 일자별 방문자 데이터 저장을 제어하면서 발생한 에러
S084	제니퍼 Config파일을 오픈하면서 발생한 에러
S085	CRUD 메트릭스 정보를 생성하기 위해 SQL을 파싱하면서 발생하는 에러
S086	DB모니터링 연계 정보을 위한 index파일을 읽기 오픈하면서 발생한 에러
S087	DB모니터링 연계 정보을 위한 index파일을 쓰기 오픈하면서 발생한 에러
S088	DB모니터링 연계 모듈을 초기화하면서 발생한 에러
S090	DB모니터링 연계에서 알수없은 리턴 명령이 보내졌다는 에러 로그
S091	DB모니터링과 연계 도중에 발생한 에러
S092	Agent Grouping 성능 데이터를 저장하면서 발생한 에러
S093	Agent Grouping 성능 데이터를 읽어들이면서 발생한 에러
S094	Agent Grouping 성능 데이터를 읽어들이면서 발생한 에러
S095	Agent Grouping 성능 데이터를 생성하면서 발생한 에러
S096	Mail을 보내기 위한 모듈을 초기화하면서 발생한 에러
S097	제니퍼서버 Message File의 내용을 로딩하면서 발생한 에러
S098	제니퍼서버 Message File의 내용을 로딩하면서 발생한 에러
S099	제니퍼 서버단위 성능데이터를 저장하면서 발생한 에러
S100	제니퍼 서버단위 성능데이터를 로딩하면서 발생한 에러
S101	제니터 서버단위 성능데이터를 파일에서 로딩하여 사용자에게 전송하면서 발생한 에러
S102	제니퍼 서버단위 성능데이터를 생성하면서 발생한 에러
S103	제니퍼 서버에서 처리량 관련 성능 데이터를 저장하면서 발생한 에러
S104	제니퍼 서버에서 처리량 관련 성능 데이터를 로딩하면서 발생한 에러
S105	제니퍼 서버에서 사용자 관련 성능 데이터를 저장하면서 발생한 에러
S106	제니퍼 서버에서 사용자 관련 성능 데이터를 로딩하면서 발생한 에러
S107	어플리케이션 호출 통계를 생성하면서 발생한 에러
S108	제니퍼 서버가 에이전트로부터 어플리케이션 호출 통계 정보를 수집하면서 발생한 에러
S109	어플리케이션 호출통계를 에이전트 별로 저장하면서 발생한 에러
S110	에이전트별 어플리케이션 호출통계를 로딩하면서 발생한 에러
S111	에이전트별 어플리케이션 호출통계를 로딩하면서 발생한 에러

표 16-13: 제니퍼 서버 에러 코드

코드	설명
S112	에이전트별 어플리케이션 호출통계를 로딩하면서 발생한 에러
S113	로딩된 어플리케이션 해쉬를 형변환 하면서 발생한 에러
S114	로딩된 에러의 해쉬키를 형변환 하면서 발생한 에러
S115	요청된 날짜의 에이전트별 성능통계를 서비스하면서 발생한 에러
S116	요청된 날짜의 에이전트 그룹별 성능통계를 서비스하면서 발생한 에러
S117	에러 발생 건수를 서비스하면서 발생한 데이터 오류
S118	제니퍼클라이언트에서 서버로 알수 없는 서비스 요청이 발생했다는 로그메세지
S119	제니퍼클라이언트에서 서버로 알 수 없는 프로토콜의 통신요청이 발생했다는 로그 메시지
S120	제니퍼 클라이언트 요청을 처리하면서 발생한 소켓 에러
S121	제니퍼 클라이언트 요청을 처리하면서 발생한 일반 에러
S122	TCP 서버 소켓 서버에 처리 요청중인 클라이언트가 일정수 이상이 되어 있음을 알리는 경고 메세지
S123	제니퍼 서버가 TCP 리스닝에 실패하여 남긴 로그
S124	제니퍼 서버가 TCP 리스닝 과정에서 발생한 에러
S125	TimeActor가 수행 도중에 발생한 에러
S126	TimeActor를 초기화 하는 과정에서 발생한 에러
S127	UDP Runtime Port 의 리스너를 초기화 하면서 발생한 에러
S128	UDP Runtime Data가 처리되지 못하고 일정 수 이상 대기되고 있다는 경고 메시지
S129	UdpRuntimeListener가 수행도중 비정상적으로 발생한 오류
S130	알 수없는 RuntimeData 전달되었다는 에러 로그
S131	Runtime Data를 처리하는 과정에서 발생한 에러
S132	UDP Summary Port 의 리스너를 초기화 하면서 발생한 에러
S133	UDP Summary Data가 처리되지 못하고 일정 수 이상 대기되고 있다는 경고 메시지
S134	UDP Summary Listener가 수행 도중 비정상적으로 발생한 오류
S135	AGENT에서 전달되는 Summary Data의 길이가 맞지 않는다는 에러 메시지
S136	WMOND에서 전달되는 데이터의 버전이 일치하지 않는다는 에러 메시지
S137	에러 정보를 저장하면서 발생한 SQL에러
S138	에러 정보를 저장하면서 발생한 일반 에러
S139	에러와 어플리케이션 맵핑 정보를 저장하면서 발생한 SQL 에러
S140	에러와 어플리케이션 맵핑 정보를 저장하면서 발생한 일반 에러

표 16-13: 제니퍼 서버 에러 코드

코드	설명
S141	에이전트 프레임워크 확장 데이터의 버전이 일치하지 않는다는 에러 메시지
S142	알 수없는 Summary Data 전달되었다는 에러 로그
S143	Summary Data를 처리하는 과정에서 발생한 에러
S144	프로파일 리스너를 초기화하면서 발생한 에러
S145	프로파일 데이터가 처리되지 못하고 일정 수 이상 대기되고 있다는 경고 메시지
S146	프로파일 Listener가 수행도중 비정상적으로 발생한 오류
S147	멀티 패킷으로 전달된 프로파일 데이터에 대해서 타임아웃이 발생했는지를 검사하면서 발생한 에러
S148	멀티 패킷으로전달된 한건의 프로파일 데이터에 대해서 타임아웃이 발생했는지를 검사하면서 발생한 에러
S149	알수 없는 프로파일 데이터가 전송되었다는 에러 메시지
S150	프로파일 데이터를 처리하면서 발생한 에러
S151	프로파일 데이터를 저장하면서 발생한 에러
S152	프로파일 ISAM파일을 로딩하는 모듈에서 파일을 오픈하며 발생한 에러
S153	프로파일 ISAM파일을 저장하는 모듈에서 파일을 오픈하며 발생한 에러
S154	프로파일 ISAM파일 관리자가 데이터를 관리하면서 발생한 오류
S155	보고서 자동 저장 모듈이 외부 서버에 리포트 파일을 보내면서 발생한 오류
S156	보고서 생성을 위한 DB 연결을 획득하면서 발생한 에러
S157	보고서 생성시 필요한 SQL을 수행하면서 발생한 에러
S158	보고서 데이터가 한번의 SQL문에 의해 3600건 이상이 조회되어 중지되었다는 경고메세지
S159	보고서 생성을 위한 데이터 조회시 발생한 에러
S160	보고서 생성을 위한 (칼럼과 함께)데이터 조회시 발생한 에러
S161	보고서 생성을 위한 제니퍼 데이터 DB를 위한 Connection획득시 발생한 에러
S162	보고서 생성을 위한 제니퍼 관리자 DB를 위한 Connection획득시 발생한 에러
S163	보고서 생성을 위한 데이터 조회시 발생한 에러
S164	보고서 데이터를 수정하면서 발생한 에러
S165	보고서 생성시 PNG파일을 생성하면서 발생한 에러
S166	보고서 그래프를 위한 데이터 셋을 만들면서 발생한 에러
S167	보고서 그래프 생성을 위한 2D 데이터 셋을 만들면서 발생한 에러
S168	보고서 생성시 HTML용 테이블을 생성하면서 발생한 에러

표 16-13: 제니퍼 서버 에러 코드

코드	설명
S169	보고서 생성시 WORD용 테이블을 생성하면서 발생한 에러
S170	보고서 생성시 WORD용 테이블 헤더를 생성하면서 발생한 에러
S171	보고서 생성시 보고서 속성정보를 정리하면서 발생한 에러
S172	보고서 생성시 필요한 SQL을 수행하면서 발생한 에러
S173	보고서 생성시 필요한 SQL을 수행하면서 발생한 에러
S174	HTML용 보고서를 생성하면서 발생한 에러
S175	HTML용 보고서를 생성하면서 SQL타입의 아이템을 처리하면서 발생한 에러
S176	HTML용 보고서를 생성하면서 TABLE타입의 아이템을 처리하면서 발생한 에러
S177	HTML용 보고서를 생성하면서 TABLE타입의 아이템을 처리하면서 발생한 에러
S178	HTML용 보고서를 생성하면서 처리중인 아이템을 위한 SQL이 없다는 에러 메시지
S179	HTML용 보고서를 생성하면서 이미지를 저장하면서 발생한 에러
S180	HTML용 보고서를 생성하면서 처리중인 아이템의 데이터를 조회하면서 발생한 에러
S181	HTML용 보고서를 생성하면서 처리중인 아이템의 데이터를 조회하면서 발생한 에러
S182	WORD형태의 보고서를 생성하면서 발생한 에러
S183	WORD형태의 보고서를 위한 이미지를 생성하면서 해당 아이템을 위한 SQL문이 없다는 에러 메시지
S184	WORD형태의 보고서를 위한 이미지를 생성하면서 발생한 에러
S185	APPLS데이터를 저장하면서 오버플로어가 발생하여 데이터가 Truncated 되었음을 알리는 메시지
S186	ERRORS데이터를 저장하면서 오버플로어가 발생하여 데이터가 생략되었음을 알리는 메시지
S187	텍스트 데이터를 저장하면서 오버플로어가 발생하여 데이터가 생략되었음을 알리는 메시지
S188	SQL 문자열을 저장하면서 오버플로어가 발생하여 데이터가 생략되었음을 알리는 메시지
S189	TX 확장 정보를 저장하면서 발생한 에러
S191	자동으로 생성된 보고서 리모트에 저장하면서 발생한 에러
S192	개별 TX에 대한 수행 요약(Xview 점) 데이터를 조회하면서 발생한 에러
S193	SNMP에서 Trap 메시지를 전송하면서 발생한 에러
S194	REMON 데이터를 저장하기 위한 테이블을 생성하는 과정에서 발생한 에러
S195	REMON데이터 테이블이 존재하는지 검사하는 도중에 발생한 에러
S196	REMON 데이터를 저장하기 위한 테이블의 칼럼이 데이터와 달라서 테이블 컬럼을 추가하는 과정에서 발생한 에러
S197	ADF데이터를 저장하면서 발생한 에러

표 16-13: 제니퍼 서버 에러 코드

코드	설명
S198	Counter정보(에이전트별 호출건수,방문자)를 저장하면서 발생한 에러
S199	Counter정보(에이전트별 호출건수,방문자)를 처음로딩하면서 발생한 에러
S200	에러정보를 저장하면서 발생한 에러
S201	애플리케이션 통계정보를 저장하면서 발생한 에러
S202	임시로 생성한 이미지데이터를 저장하면서 발생한 에러
S203	임시로 생성한 이미지데이터를 로딩하면서 발생한 에러
S204	제니퍼 Data DB의 Connection속성을 SET하면서 발생한 에러
S205	제니퍼 Admin DB의 Connection속성을 SET하면서 발생한 에러
S206	APP와 SQL맵핑 정보를 로딩하면서 발생한 에러

Index

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

ㄱ

ㄴ

ㄷ

ㄹ

ㅁ

ㅂ

ㅅ

ㅇ

ㅈ

더욱 새로워진 제니퍼

애플리케이션성능관리(APM)

애플리케이션성능관리(APM)는 조직과 보유 시스템의 현실에 맞는 적절한 투자와 가용성 측면을 고려하여, 효과적인 성능 모니터링 및 장애 대응 전략을 수립하고 성능관리 체계를 구축하는 것입니다. 애플리케이션성능관리(APM)는 전통적인 시스템관리 솔루션(SMS,NMS)과 달리, 실제 서비스되고 있는 시스템의 서비스 관점에서의 성능적 현황과 내부적 애플리케이션 관점에서의 성능 장애 대응 및 분석 역량을 강화시켜, 보다 지능적인 방법으로 대 고객 서비스의 안정화를 이루어 궁극적으로 총소유비용(TCO)을 효과적으로 낮출 수 있습니다.

제니퍼(JENNIFER)

(주)제니퍼소프트의 제니퍼(Jennifer)는 엔터프라이즈 웹 기반 시스템 하에서의 웹애플리케이션서버(WAS)를 중심으로 한 전문적인 애플리케이션성능관리(APM) 솔루션으로서, 국내 APM시장 1위의 애플리케이션성능관리(APM, Application Performance Management) 제품입니다. 제니퍼는 실시간 서비스 모니터링, 실질적인 장애원인 분석, 직관적인 통합 성능관리 및 운영을 지원하는 제품으로 전통적인 시스템관리SW와는 달리 애플리케이션 내부의 서비스를 모니터링하기 때문에 성능장애진단분야에서도 매우 효율적입니다. 제니퍼는 공공, 금융, 유통, 제조, IT/통신 등 웹을 기반으로 업무가 진행되는 모든 분야에 적용하여 사용될 수 있어 IT 서비스 관리를 필요로 하는 전반적인 실무부서인 시스템 운영팀, 품질관리팀, 성능테스트팀, 애플리케이션 개발팀, 데이타베이스 관리팀, 장애해결 TFT팀에서 반드시 필요로 하는 제품입니다.

제니퍼 대표 기능

- 실시간 부하량 / 서비스 모니터링
- 장애진단 / 성능저하 병목 원인분석
- 전거래 응답시간분포도(X-VIEW)
- 개별 트렌젝션 상세 프로파일링
- 외부자원(CTG,Jolt,WebT,WTC) 연계추적
- 애플리케이션 / SQL 쿼리 튜닝
- 메모리 누수 추적
- 애플리케이션 장애 감지 및 분석
- 서비스 폭주 시 지능적 부하량 제어
- 사용자 정의형 통합 대시보드(Drag & Drop)
- 사용자 권한별 메뉴/화면 구성
- 시스템/WAS 리소스 모니터링
- 템플릿 기반 사용자 정의형 통계 분석 보고서

다이나믹 프로파일링

WAS 서버를 재기동 하지 않은 채 트랜잭션 프로파일링을 변경

다이나믹 스텍트레이스

임의의 클래스/메소드의 풀스텍트레이스를 애플리케이션 소스의 수정 없이 다이나믹하게 추출

도메인별 통합 모니터링

각 업무 시스템 별로 도메인으로 통합하여 대시보드 구성

JENNIFER®
Enterprise Web Business Monitoring

www.jennifersoft.com
153-801 서울 금천구 가산동 60-11 스타밸리 1103~1104호
TEL : 02.2027.0391~9 FAX : 02.2027.0390 E-MAIL : tech@jennifersoft.com