



프로그램 최적화

목 차

1. 프로그램 최적화 이해
 2. 아키텍처 레벨 최적화 I
 3. 아키텍처 레벨 최적화 II
 4. 모델 레벨 최적화
 5. 프로그램 레벨 최적화
 6. 도구 I - MAT
 7. 도구II - Code Pro
 8. 도구 III - SonarQube
- 별도 교재 → 플랫폼 레벨 최적화 (JVM/GC)**



1. 프로그램 최적화 이해 (wikipedia.org)

- 1.1 최적화 정의
- 1.2 최적화 수준
- 1.3 최적화와 플랫폼
- 1.4 길이 축소 최적화
- 1.5 무의미한 최적화
- 1.6 트레이드-오프
- 1.7 병목 지점 찾기
- 1.8 최적화가 필요한 곳
- 1.9 요약

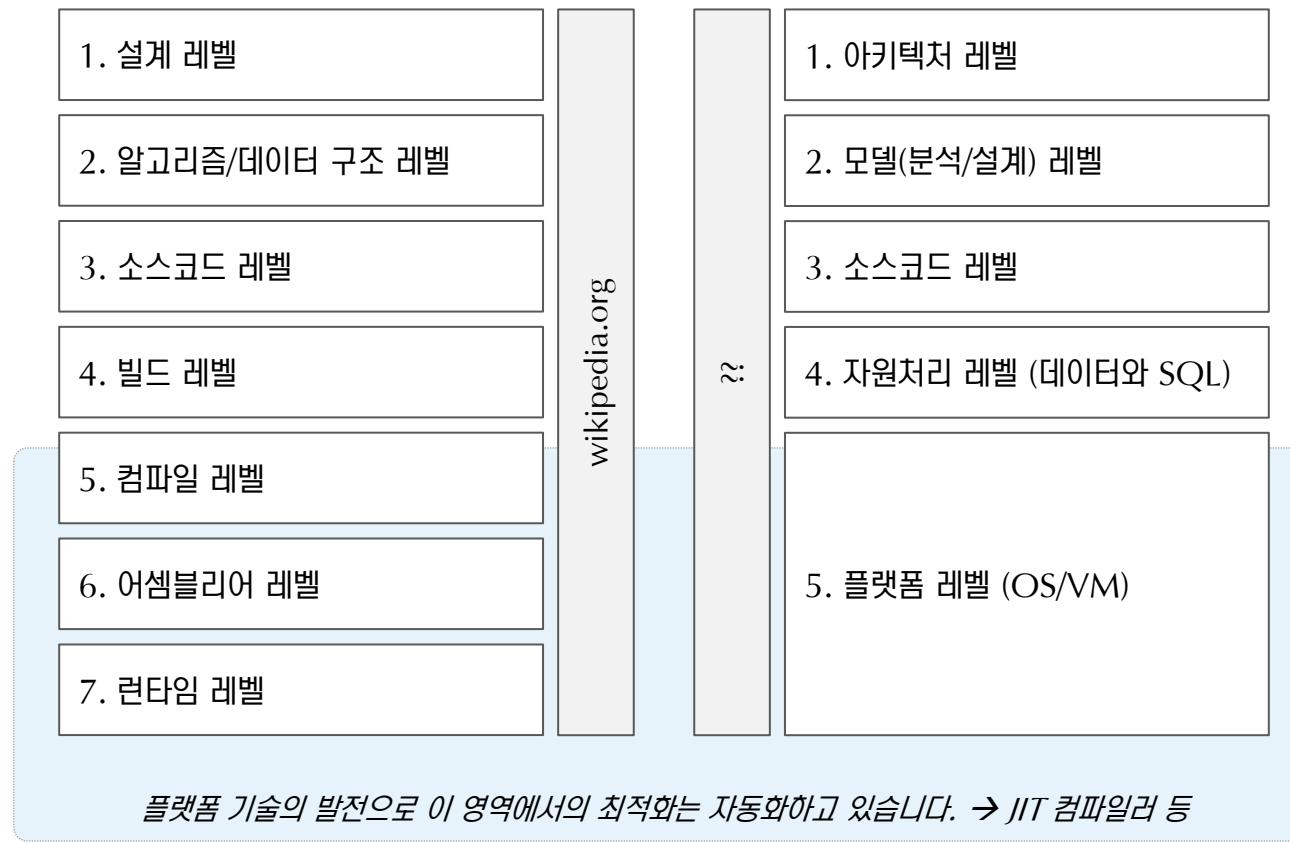
1.1 최적화(optimization) 정의

- ✓ 컴퓨터 과학에서 프로그램 최적화 또는 소프트웨어 최적화란 작업 효율을 높이거나 보다 적은 자원을 사용하도록 소프트웨어 시스템을 변경하는 작업 절차를 의미합니다.
- ✓ 일반적으로 컴퓨터 프로그램은 최적화 되어서 보다 빨리 실행되거나, 보다 적은 메모리 저장소나 보다 적은 다른 자원을 요구하거나, 보다 적은 전원을 사용합니다.
- ✓ 최적화를 잘 수행하더라도 모두 상황에서 최적(optimal)인 시스템은 있을 수 없습니다.
- ✓ 수행 시간, 메모리, 저장소, CPU 등 어떤 자원에 우선순위를 두는가에 따라 최적화 방향을 달라질 수 있습니다.
- ✓ 따라서, 시스템의 환경과 목표를 이해하고, 다양한 트레이드-오프를 고려하여 최적화 작업을 수행해야 합니다.
- ✓ 효율적인 자원 구성과 사용으로 수행 성능(performance)이나 범위성(scalability) 등과 같은 품질 목표를 달성하는 것이 최적화의 궁극적인 목표입니다. 이것은 아키텍처 설계의 목표, 방향과 일치합니다.



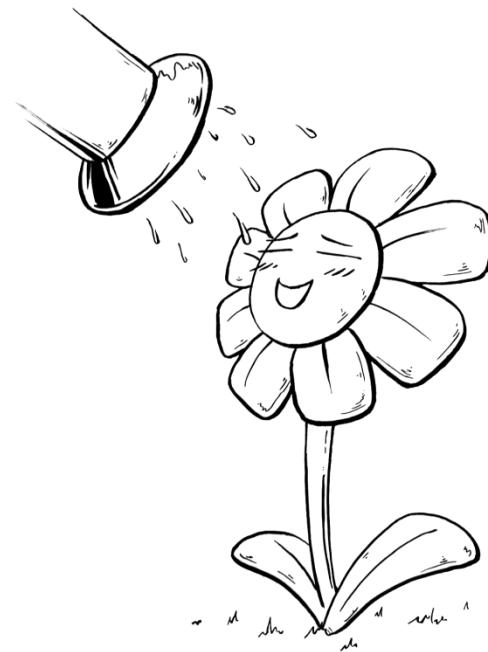
1.2 최적화(optimization) 수준

- ✓ 최적화는 다양한 수준(level)에서 이루어지며, 높은 수준일수록 영향도가 크고 후에 변경이 어렵습니다.
- ✓ 따라서 최적화는 위에서부터 아래로 진행을 하는 것이 보다 효율적입니다.
- ✓ 하지만, 전체적인 수행성능(overall performance)은 아래 수준 최적화에 영향을 받을 수도 있습니다.
- ✓ 최적화는 시스템 운영 상황에 영향을 많이 받기 때문에, 낮은 수준의 최적화는 운영 직전에 수행합니다.



1.3 최적화[optimization]와 플랫폼

- ✓ 코드 최적화는 플랫폼 의존적인 것과 플랫폼 독립적인 것으로 분류할 수 있습니다.
- ✓ 플랫폼 독립적인 방식은 효율적일 수 있지만 최적화에 한계가 있습니다. ← 컴파일 레벨
- ✓ 특정 속성이나 매개 변수를 상황에 맞추어 조정할 수 있는 플랫폼 의존적 방식이 고도의 최적화가 가능합니다.
- ✓ 이 경우, 런타임 플랫폼인 운영체제, 가상 머신, HW 아키텍처 등에 맞추어 최적화를 합니다. ← 플랫폼 레벨



1.4 길이 축소 최적화

- ✓ 계산 작업은 서로 다른 효율성 수준을 목표로 여러 가지 방식으로 구현할 수 있습니다.
- ✓ 같은 기능을 보다 효율적으로 수행하는 방법 중에 하나가 길이 축소(strength reduction) 같은 것입니다.
- ✓ 1에서 10까지 더하는 연산을 루프를 이용하는 방법과 공식을 이용하는 방식으로 계산해 봅니다.
- ✓ 필요할 경우 컴파일러가 이러한 최적화를 수행하지만, 후자가 전자보다 낫다고 말할 수는 없습니다.

```
public class WaysOfSum {  
    public static void main(String[] args) {  
        int limit = 10;  
        sumWithLoop(limit);  
        sumWithFomula(limit);  
    }  
  
    public static void sumWithLoop(int limit) {  
        int sum = 0;  
        for(int i=0; i<=limit; i++) {  
            sum += i;  
        }  
  
        System.out.println("sum: " + sum);  
    }  
  
    public static void sumWithFomula(int limit) {  
        int sum = limit * (1+limit) / 2;  
        System.out.println("sum: " + sum);  
    }  
}
```

- ✓ limit이 작을 경우, 루프가 빠르다.
- ✓ 공식은 가독성이 떨어진다.
- ✓ 루프는 오버플로우 가능성 있다.
- ✓ 대체로 더하기 연산이 곱하기나 나누기 보다 훨씬 빠르다. 하지만, HW 아키텍처 특성에 따라 다르다.

결과:
sum: 55
sum: 55

1.5 무의미한 최적화

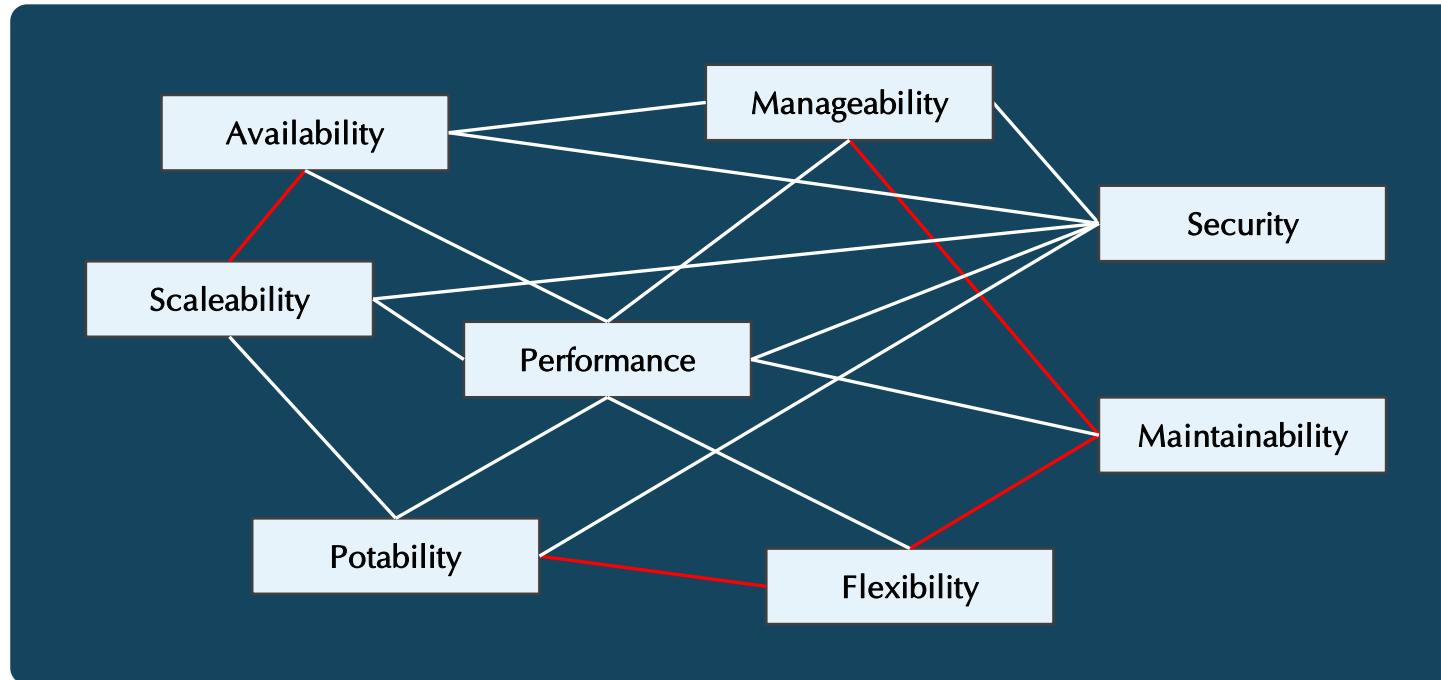
- ✓ 현장에서 일부 개발자들이 가지고 있는 믿음 중에 하나가 상수와 변수 비교에서 상수를 앞에 두는 것이 성능 면에서 우월하다는 것입니다. 따라서, 대부분 상수를 앞에 두고 방식의 코딩을 합니다.
- ✓ 물론, 이런 방식의 코딩이 주는 다른 효과는 NullPointerException 회피로 보입니다.
- ✓ 최적화 측면에서 효과가 없을 뿐만 아니라, 결정적으로 가독성에 문제가 있어 보입니다.

```
public static void compareLiteralFirst(String answer) {  
    //  
    if ("YES".equals(answer)) {  
        // do something  
    }  
  
    public static void compareVariableFirst(String answer) {  
        //  
        if (answer.equals("YES")) {  
            // do something  
    }  
}
```

- ✓ 가독성에 문제가 있습니다. "Yes"와 대답을 비교하는 것이 아니라, 대답이 "Yes"인지 확인을 하는 것이 바람직합니다.
- ✓ 물론 상수를 이렇게 사용하면 안됩니다.

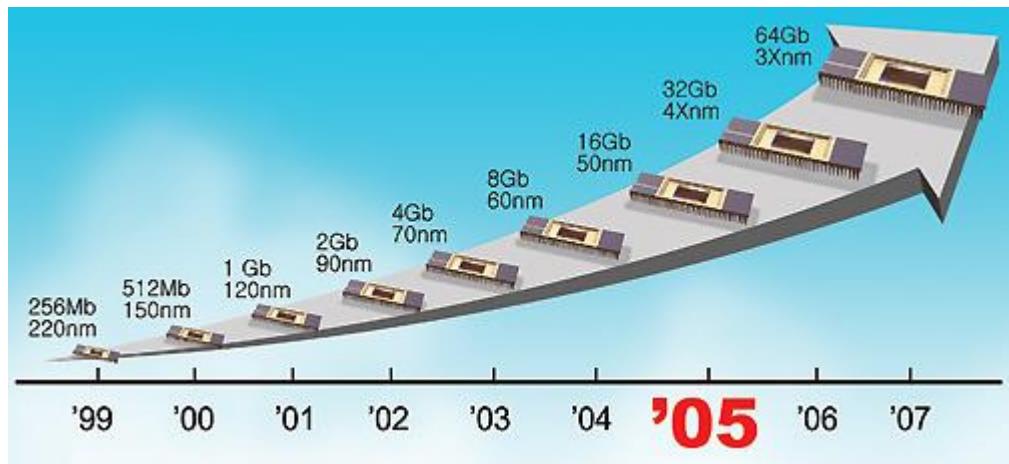
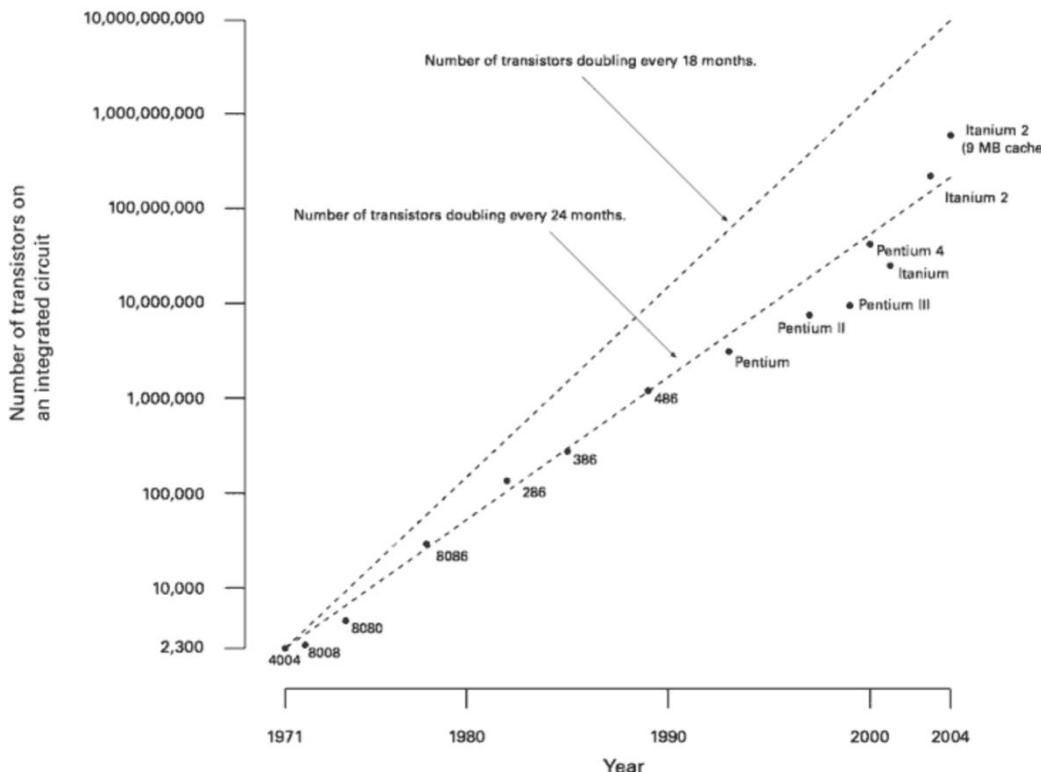
1.6 트레이드-오프(1/2)

- ✓ 어떤 경우 최적화는 정교한 알고리즘, 특별한 처리나 트릭을 사용하여야 합니다.
- ✓ 최적화 프로그램은 그렇지 않은 프로그램 보다 이해하기 어려울 수 있으며, 유지보수가 더 힘들 수 있습니다.
- ✓ 최적화는 수행 성능 관련 한 두 가지를 개선하는데 초점을 두고 있습니다. ← 실행 시간, 메모리나 디스크 사용, 소비 전력, 대역폭 등과 같은 자원



1.6 트레이드-오프(2/2) – 자원의 가치 변화

- ✓ 무어의 법칙: 반도체 집적회로의 성능은 18개월마다 두 배로 증가한다.
- ✓ 황의 법칙: 반도체 집적회로의 성능은 12개월마다 두 배로 증가한다.
- ✓ 최적화의 대상이 자원, 그리고 그 자원의 비용을 다른 관점에서 바라 볼 필요가 있습니다.



출처: <http://www.idomin.com/news/articleView.html?idxno=164819>

1.7 병목 지점 찾기

- ✓ 수행 성능에 원인을 제공하는 컴포넌트인 병목 지점 찾기는 최적화 작업 중에 한 가지입니다.
- ✓ 리소스를 과도하게 소비하거나 I/O 지연을 일으키거나, 네트워크 대역폭을 소비하는 지점을 핫스팟이라고 합니다.
- ✓ 이런 작업에서도 파레토 법칙이 통합니다. → 20% 오퍼레이션이 80%의 자원을 소비합니다.
- ✓ SW 엔지니ering에서는 보다 구체적으로, 10% 프로그램이 실행 시간의 90%를 점유합니다.
- ✓ 복잡한 알고리즘이나 데이터 구조가 최적화의 수단이지만, 단순한 작업에는 단순함을 유지할 필요가 있습니다.
- ✓ 메모리 추가로 쉽게 문제를 해결할 수 있으며, 네트워크 대역폭 확장으로 문제를 해결할 수도 있습니다.

전체 결과의 80%가 원인의 20%에서 일어난다.

이탈리아 경제학자, 빌프레도 파레토

대한민국 국토의 74%를 5.5%의 땅부자들이 소유하고 있다. 2(2:8):8(2:8)

1.8 최적화가 필요한 곳

- ✓ 최적화가 가독성을 낮출 수 있으며, 성능 향상을 위한(기능 추가가 아닌) 코드가 추가할 수 있습니다.
- ✓ 그 결과 시스템은 더 복잡해지고, 유지보수하기 어려우며, 버그를 찾기 어려울 수도 있습니다.
- ✓ 따라서, 최적화는 개발의 마지막 단계에서 수행하는 경우가 많습니다.
- ✓ 설부른 최적화는 모든 악의 근원이다. ← Donald Knuth

전체적인 수행 성능 분석을 통해서, 구체적으로 어떤 부분에서 어떤 비효율이 있는지 확인한 후에야, 최적화 작업 여부를 결정해야 한다. 설계 → 코딩 → 프로파일링 → 최적화 흐름을 따라 가기 위해서 단순한 설계(loose coupling, high cohesion)를 유지해야 한다.

1.9 요약

- ✓ 운영 체제, 가상 머신, 컴파일러, 빌드 도구 등이 발전하면서 낮은 수준의 최적화 니즈는 줄어들고 있습니다.
- ✓ 자원 비용의 비약적인 감소도 최적화를 바라보는 관점을 바꿔 놓고 있습니다.
- ✓ 인적 자원의 비용은 지속적으로 상승하고 있습니다. 따라서, 인적 자원 비용을 줄이되 물적 자원을 투입하는 방식으로 최적화(문제를 해결)하는 경향이 있습니다.

프로그램 최적화의 첫 번째 규칙은 “하지 말라”이다. 두 번째 규칙은 전문가만 최적화 작업을 하되, “아직은 이르니 하지 말라”이다.

Michael Jackson



2. 아키텍처 레벨 최적화 I

-
- 2.1 상황 이해
 - 2.2 설계 구조
 - 2.3 테스트 결과
 - 2.4 요약

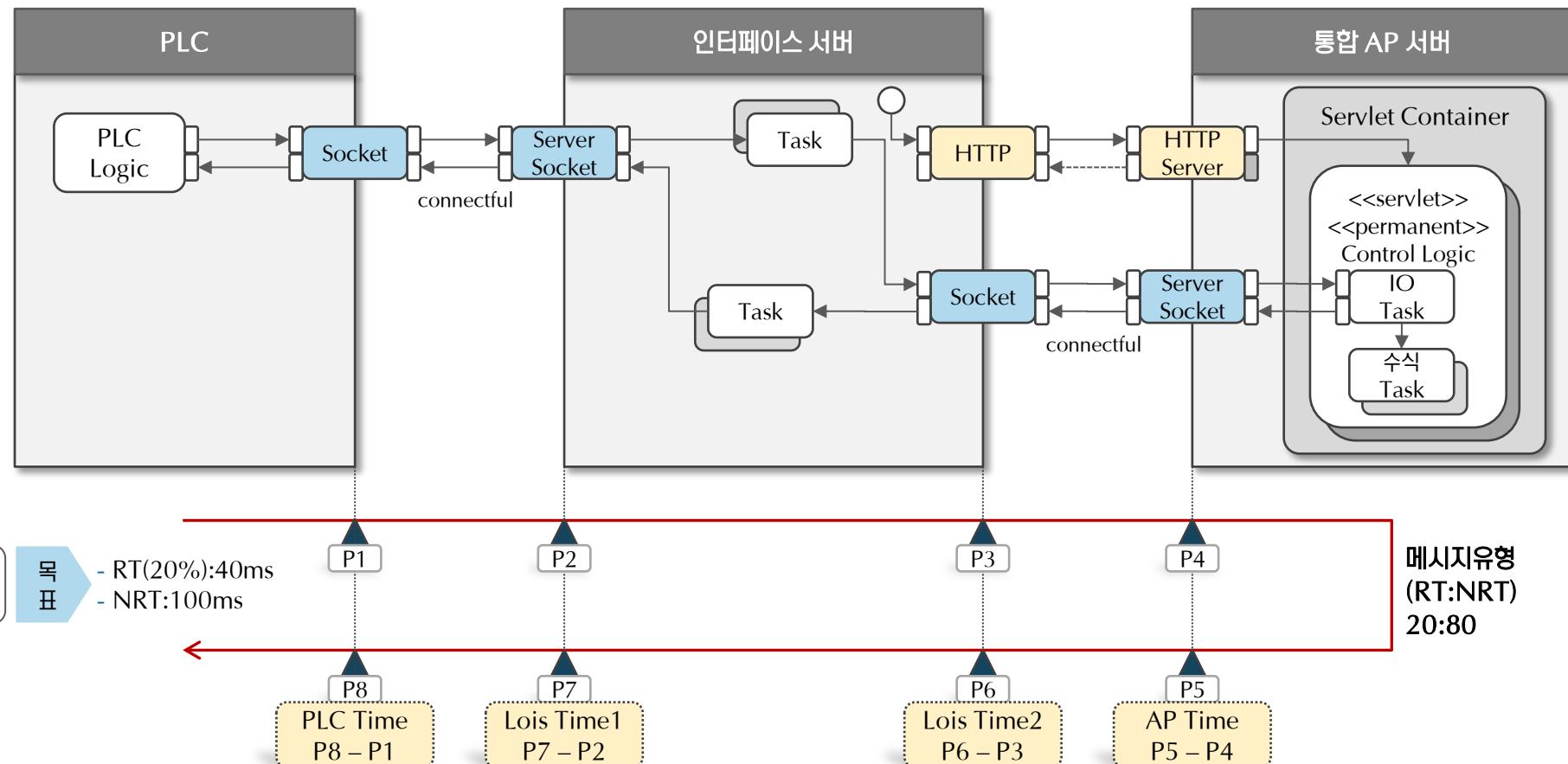
2.1 상황 이해

- ✓ 철강 분야의 생산 관리 시스템(Manufacturing Execution System)에서의 사례입니다.
- ✓ 기존 C로 개발하여 운영하던 MES를 Java 기반으로 재구축하려는 기획 단계에서 수행한 파일럿입니다.
- ✓ 자바로 개발했을 때, PLC장비로부터 프로세스 컴퓨터로 이르는 제어가 40ms 안에 이루어질 수 있는가?
- ✓ 통합 App 서버는 Java/WAS 기반 시스템입니다. 간단한 테스트에서 불규칙한 결과를 얻은 상황이었습니다.



2.2 설계 구조

- ✓ Socket 기반으로 인터페이스 서버의 중개 기능을 개발하였습니다.
- ✓ 인터페이스 서버와 통합 AP 서버간의 통신은 비동기 방식으로 처리하였습니다.
- ✓ Servlet 컨테이너 제약조건 극복을 위해 HTTP 커넥션 풀을 준비하였습니다.
- ✓ Peer-Peer 간의 반복적인 메시지 전송이므로, connectful 소켓 방식을 선택하였습니다.



2.3 테스트 결과 (1/3)

- ✓ 테스트 결과 80만 건 호출 중 20~30여 건이 100ms를 초과하였습니다. → WAS와 네트워크 상태
- ✓ 인터페이스 서버와 통합 AP 서버간 통신 방식이 Connectful이므로 안정적인 메시지 전송이 이루어졌습니다.
- ✓ Maximum 80개의 Connectful 소켓을 생성하여 연결한 후 테스트하였습니다.
- ✓ 시나리오 별 80만 건, 총 480만 건 호출에서 모두 42~86ms 내로 처리하였습니다. → 평균 4ms

4k 전문			1.5k 전문		
CASE1	CASE2	CASE3	CASE1	CASE2	CASE3
Max 62ms	Max 68ms	Max 63ms	Max 60ms	Max 86ms	Max 42ms
평균 4ms					

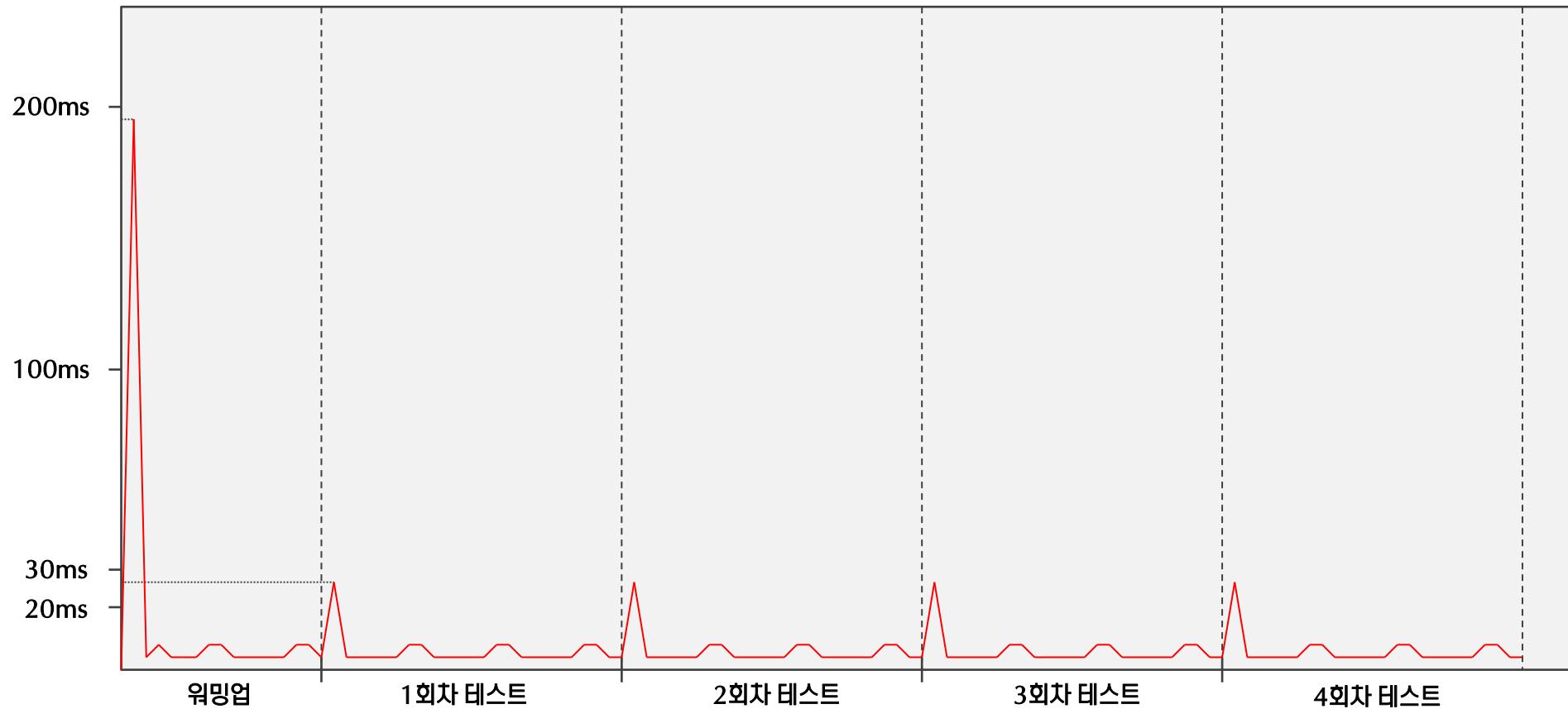
2.3 테스트 결과 (2/3)

- ✓ 초기 워밍업 기간 동안 20~30ms의 속도를 보여주다가, 일정 시간 후 4~6ms 패턴을 유지하였습니다.
- ✓ 데이터를 확인한 결과 추가로 최적화 작업을 할 이유가 없었습니다.
- ✓ 만약 추가로 최적화가 필요하다면 http 구간이 그 대상이 될 수 있습니다.

PLC	Thread	:	80 request:10	28 P06	CC	6001BILL1651]	[N]PLC26	test1	CC	1297390858953	1297390858957	4 P26	CC	26001BILL1851]	
[N]PLC6	test1	CC	1297390858905	1297390858933	35 P05	CC	5001BILL1641]	[N]PLC27	test1	CC	1297390858957	1297390858961	4 P27	CC	27001BILL1861]
[S]PLC5	test1	CC	1297390858898	1297390858933	35 P02	CC	2001BILL1611]	[N]PLC28	test1	CC	1297390858960	1297390858964	4 P28	CC	28001BILL1871]
[N]PLC2	test1	CC	1297390858900	1297390858935	34 P04	CC	4001BILL1631]	[N]PLC29	test1	CC	1297390858963	1297390858967	5 P29	CC	29001BILL1881]
[N]PLC4	test1	CC	1297390858900	1297390858934	30 P01	CC	1001BILL1601]	[S]PLC30	test1	CC	1297390858967	1297390858971	4 P30	CC	30001BILL1891]
[N]PLC1	test1	CC	1297390858905	1297390858935	14 P16	CC	16001BILL1751]	[N]PLC31	test1	CC	1297390858971	1297390858976	5 P31	CC	31001BILL1901]
[N]PLC16	test1	CC	1297390858921	1297390858935	32 P09	CC	9001BILL1681]	[N]PLC32	test1	CC	1297390858975	1297390858980	5 P32	CC	32001BILL1911]
[N]PLC9	test1	CC	1297390858903	1297390858935	10 P17	CC	17001BILL1761]	[N]PLC33	test1	CC	1297390858978	1297390858982	4 P33	CC	33001BILL1921]
[N]PLC17	test1	CC	1297390858924	1297390858934	21 P15	CC	15001BILL1741]	[N]PLC34	test1	CC	1297390858982	1297390858986	4 P34	CC	34001BILL1931]
[S]PLC15	test1	CC	1297390858917	1297390858938	24 P14	CC	14001BILL1731]	[S]PLC35	test1	CC	1297390858986	1297390858990	4 P35	CC	35001BILL1941]
[N]PLC14	test1	CC	1297390858914	1297390858938	30 P11	CC	11001BILL1701]	[N]PLC36	test1	CC	1297390858989	1297390858993	4 P36	CC	36001BILL1951]
[N]PLC11	test1	CC	1297390858905	1297390858935	44 P12	CC	12001BILL1711]	[N]PLC37	test1	CC	1297390858992	1297390858996	4 P37	CC	37001BILL1961]
[N]PLC12	test1	CC	1297390858908	1297390858952	36 P08	CC	8001BILL1671]	[N]PLC38	test1	CC	1297390858996	1297390859000	4 P38	CC	38001BILL1971]
[N]PLC8	test1	CC	1297390858900	1297390858936	4 P23	CC	23001BILL1821]	[N]PLC39	test1	CC	1297390858999	1297390859003	4 P39	CC	39001BILL1981]
[N]PLC23	test1	CC	1297390858948	1297390858952	28 P13	CC	13001BILL1721]	[S]PLC40	test1	CC	1297390859004	1297390859009	5 P40	CC	40001BILL1991]
[N]PLC13	test1	CC	1297390858911	1297390858939	38 P03	CC	3001BILL1621]	[N]PLC41	test1	CC	1297390859007	1297390859011	4 P41	CC	41001BILL2001]
[N]PLC3	test1	CC	1297390858999	1297390858937	5 P24	CC	24001BILL1831]	[N]PLC42	test1	CC	1297390859011	1297390859015	4 P42	CC	42001BILL2011]
[N]PLC24	test1	CC	1297390858948	1297390858953	37 P10	CC	10001BILL1691]	[N]PLC43	test1	CC	1297390859015	1297390859019	4 P43	CC	43001BILL2021]
[S]PLC10	test1	CC	1297390858900	1297390858937	10 P18	CC	18001BILL1771]	[N]PLC44	test1	CC	1297390859020	1297390859024	4 P44	CC	44001BILL2031]
[N]PLC18	test1	CC	1297390858928	1297390858938	5 P21	CC	21001BILL1801]	[S]PLC45	test1	CC	1297390859023	1297390859027	4 P45	CC	45001BILL2041]
[N]PLC21	test1	CC	1297390858938	1297390858943	36 P07	CC	7001BILL1661]	[N]PLC46	test1	CC	1297390859025	1297390859030	5 P46	CC	46001BILL2051]
[N]PLC7	test1	CC	1297390858903	1297390858939	5 P20	CC	20001BILL1791]	[N]PLC47	test1	CC	1297390859027	1297390859031	4 P47	CC	47001BILL2061]
[S]PLC20	test1	CC	1297390858934	1297390858939	4 P19	CC	19001BILL1781]	[N]PLC49	test1	CC	1297390859037	1297390859041	4 P49	CC	49001BILL2081]
[N]PLC19	test1	CC	1297390858939	1297390858943	5 P22	CC	22001BILL1811]	[N]PLC48	test1	CC	1297390859037	1297390859042	5 P48	CC	48001BILL2071]
[N]PLC22	test1	CC	1297390858942	1297390858947	5 P25	CC	25001BILL1841]	[N]PLC6	test1	CC	1297390859039	1297390859043	4 P06	CC	6002BILL1652]

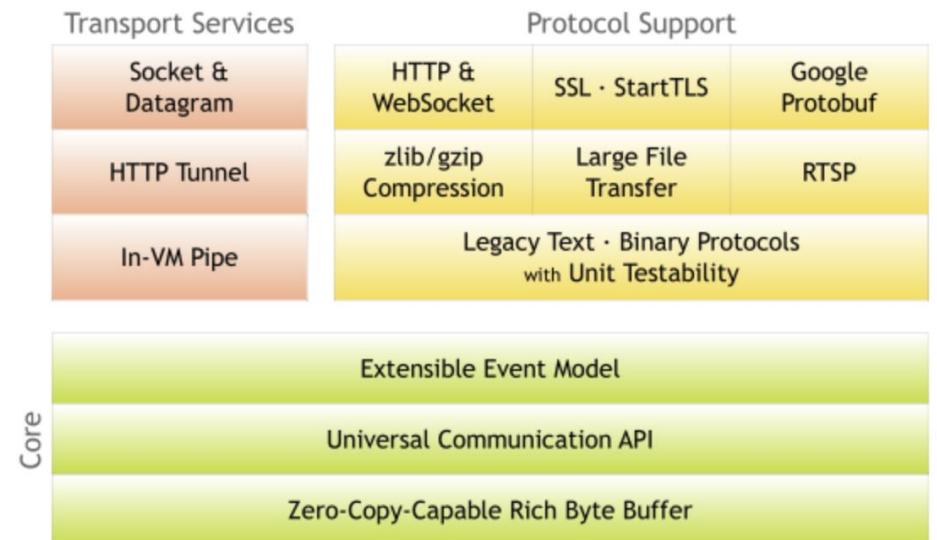
2.3 테스트 결과 (3/3)

- ✓ 서버 기동 후 워밍업 단계에서 200ms 정도 오르는 구간이 있습니다. → 객체 생성, 커넥션 준비 등
- ✓ 각 회차 테스트 초반에 20ms~30ms 오르는 구간이 있으며, 패턴은 규칙적으로 반복되었습니다.
- ✓ 그 외 규칙적으로 10ms 내외로 오르는 구간이 있으며, GC 등과 관련이 있는 것으로 추정하였습니다.



2.4 요약

- ✓ 파일럿을 통해서 C와 Java에 대한 성능 논쟁을 마무리하고 프로젝트를 시작하였습니다.
- ✓ GC에 대한 조정 없이 기본 JVM 설정으로 목표 수치에 도달하였습니다.
- ✓ JVM 1.6, WebLogic WAS 환경 등 추가로 최적화 가능한 요소들이 많았습니다.
- ✓ 인터페이스의 커넥션 최적 갯수는 80개로 확인되었습니다.
- ✓ **netty를 사용하면 개발, 운영, 품질 측면에서 더 좋을 결과를 얻을 수 있습니다.**



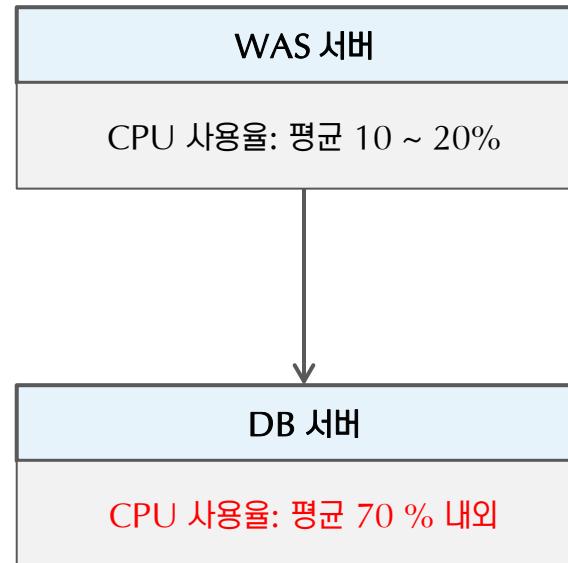


3. 아키텍처 레벨 최적화 II

-
- 3.1 상황 이해
 - 3.2 최적화 포인트
 - 3.3 기본 틀
 - 3.4 상태 분석
 - 3.5 최적화 이미지
 - 3.6 요약

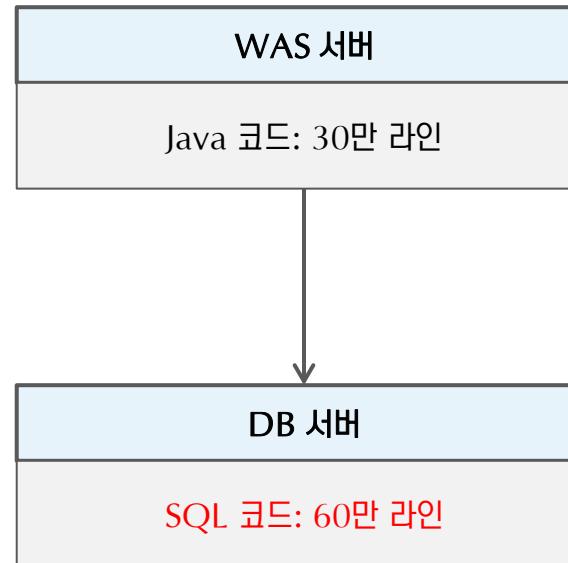
3.1 상황 이해[1/2]

- ✓ 고객 사의 시스템은 CPU 사용량을 보면 여러 종류의 서버 간에 부하 불균형을 보여 주고 있었습니다.
- ✓ DB 서버의 부하를 줄이기 위해 다양한 SQL 튜닝과 파티션 작업 등을 통해 부하를 줄이려 노력하고 있습니다.
- ✓ 그러나 결과는 나아 지지 않습니다. 최적화 분야 전문가들이 다수 다녀갔지만 크게 달라지지 않았습니다.
- ✓ 최적화 레벨에 대한 초점에 문제가 있는 것 같습니다.



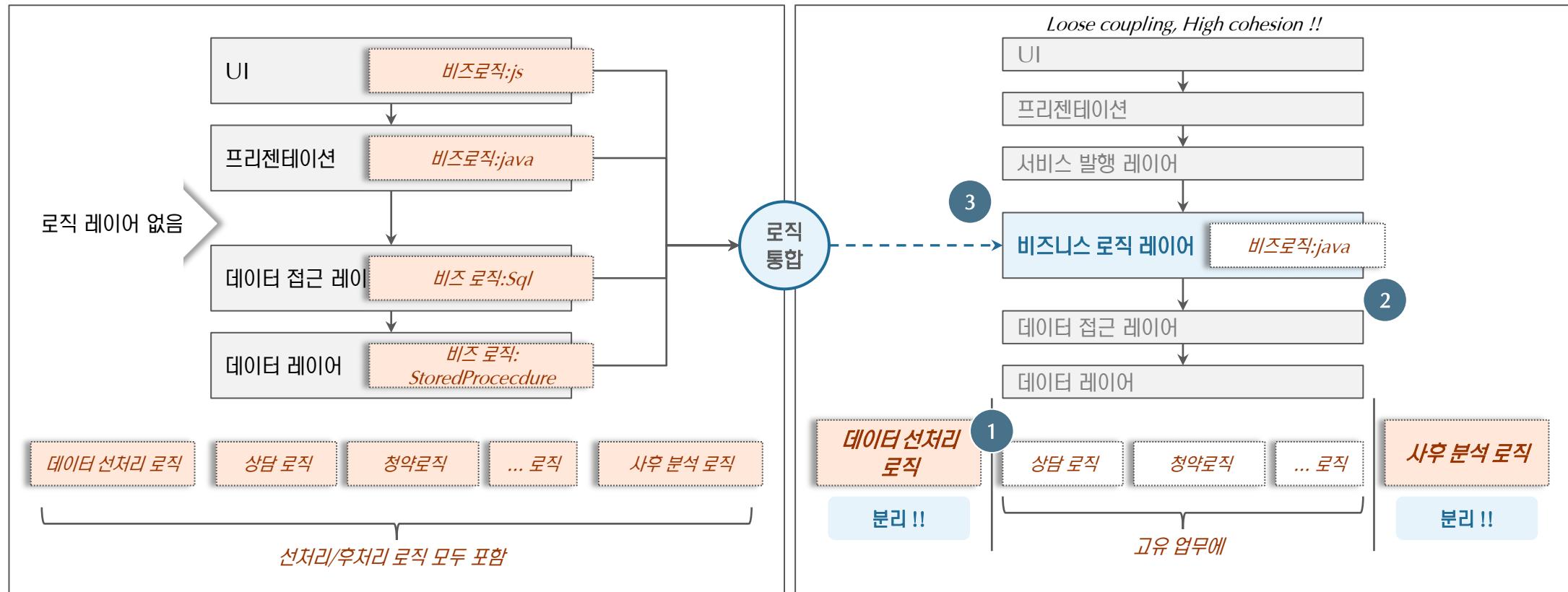
3.1 상황 이해[2/2]

- ✓ DB 서버에 부하가 많이 걸리는 것은 많은 일을 하기 때문입니다. 일을 줄여 주는 설계를 했어야 했습니다.
- ✓ 형상관리 서버로부터 소스코드와 SQL 코드를 확인한 결과 대략 SQL 코드가 두 배 정도로 많습니다.
- ✓ 이런 경우 SQL을 프로그래밍 언어로 사용하였다고 판단할 수 있습니다. 질의 언어를 다른 용도로 사용했습니다.
- ✓ 최적화 레벨은 아키텍처 수준이어야 하는 상황입니다.



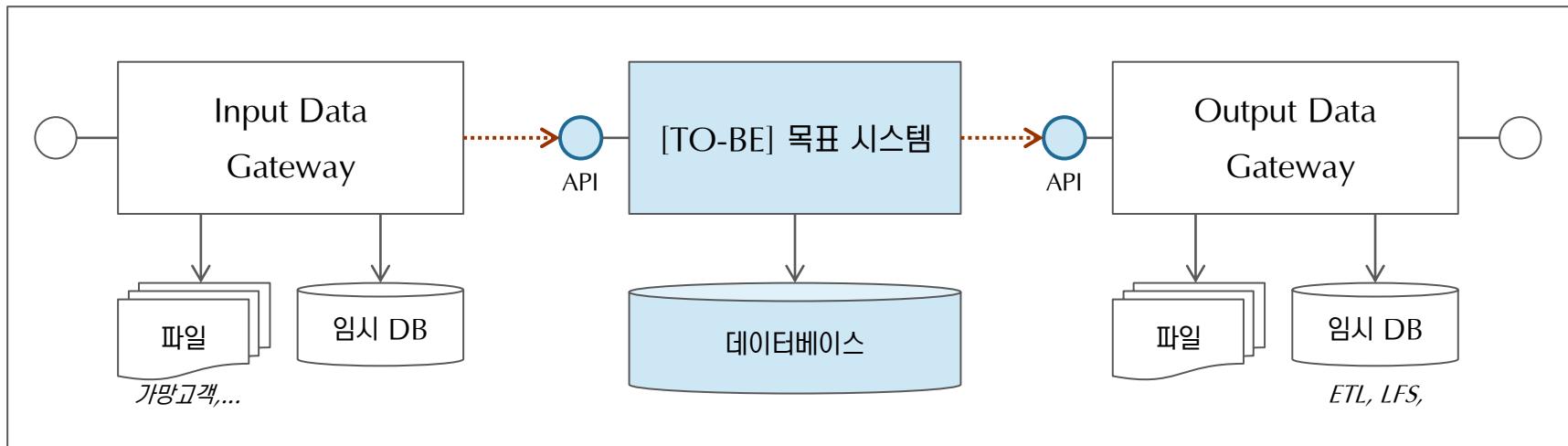
3.2 최적화 포인트

- ✓ 목표 시스템의 성능과 복잡성 이슈는 소프트웨어 설계의 기본 규칙이 무너졌을 때 발생합니다.
- ✓ 소프트웨어의 설계 원칙인 SoC 관점으로 돌아가서 해결 방안을 마련해야 하겠습니다.
- ✓ 비즈니스 로직은 high cohesion하고, 처리 대상 업무는 loose coupling하는 방식으로 개선해야 합니다.



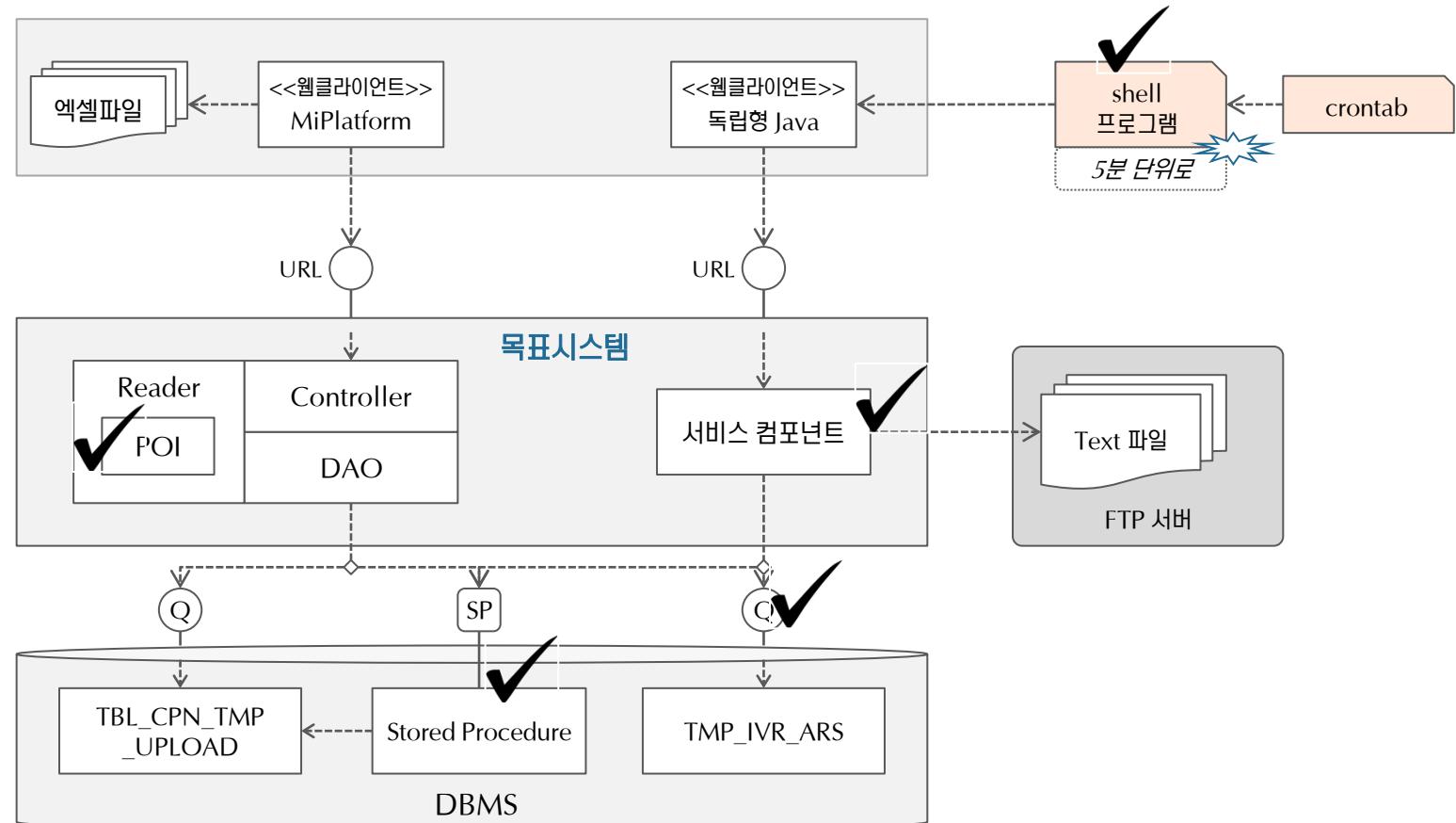
3.3 기본 틀

- ✓ 모든 데이터는 목표 시스템이 제공하는 API를 통해서만 DB로 들어올 수 있습니다. 가공은 Gateway에서.
- ✓ 모든 데이터는 Output Data Gateway를 통해서 나갑니다. 클라이언트의 요구에 따른 가공은 Gateway에서 하도록 합니다. 목표 시스템은 외부 시스템과의 데이터 교환을 서비스 API로 제한함으로써 외부 요청 변화에 따른 영향을 최소화 합니다.



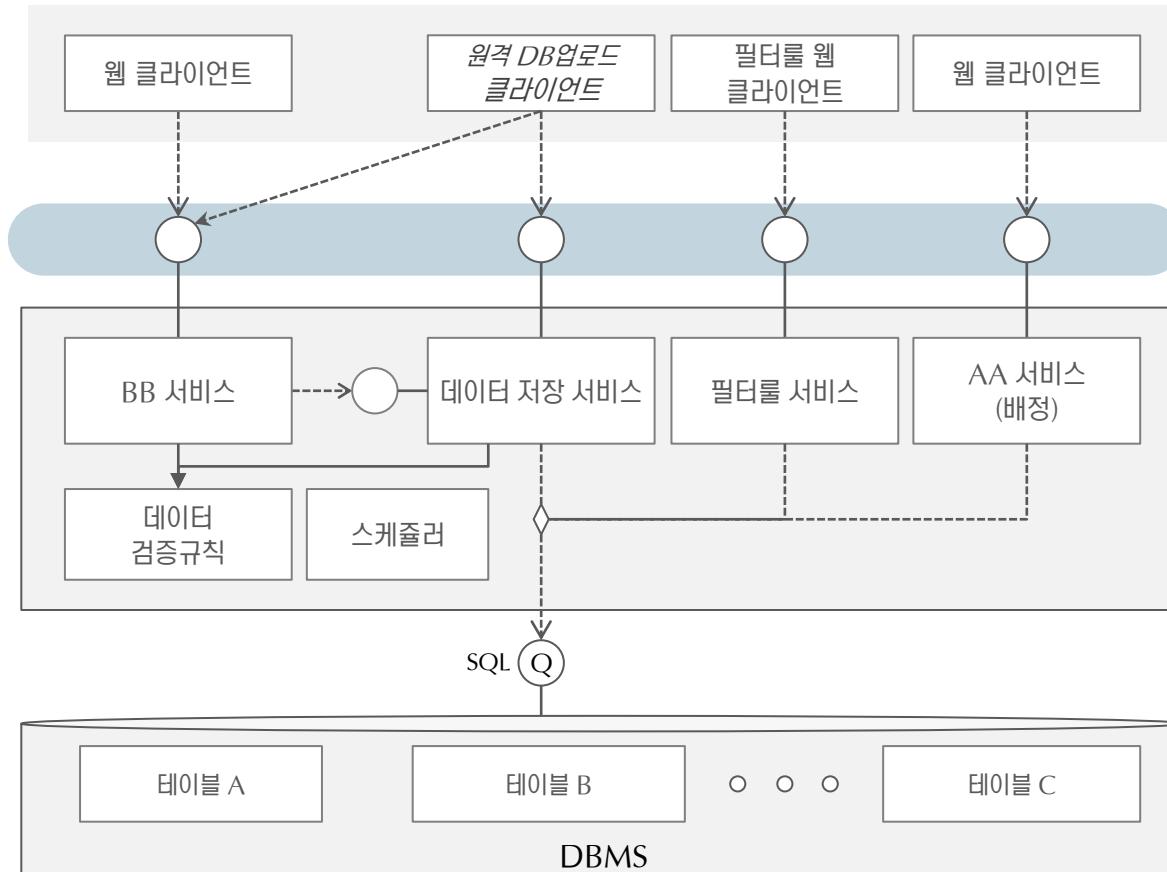
3.4 상태 분석

- ✓ 목표 시스템의 목표 기능(DB 업로드) 관련 아키텍처 분석을 하여 문제를 찾았습니다.
- ✓ UI, 서비스 인터페이스, DB 등 여러 부분에서 일관성, 단순성, 등에서 아키텍처 결함을 갖고 있습니다.
- ✓ 상호 작용하는 시스템의 종류가 많아서 관리성이 떨어지며, 처리가 격리(isolation) 되지 않아 유지보수가 어렵습니다.



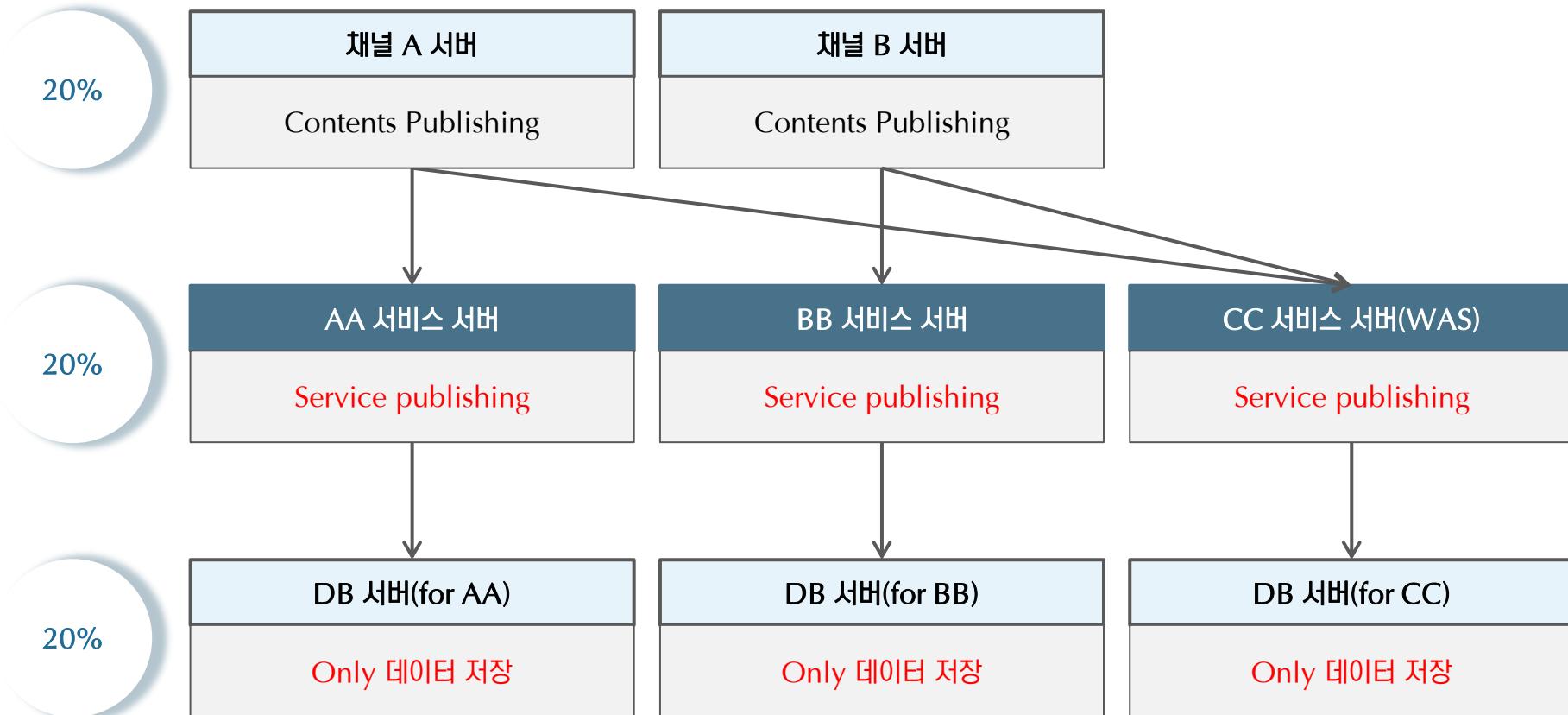
3.5 최적화 이미지(1/3)

- ✓ SoC 원칙을 지켜서 입력/출력 간결화하는 것을 목표로 구조를 개선합니다.
- ✓ UI는 클라이언트라는 관점과 상호작용은 단순화 하는 방향으로 개선합니다.
- ✓ 서비스 인터페이스는, 단순함과 명확함을 유지하면서, RESTful로 진화할 수 있는 기반을 마련합니다.
- ✓ SQL은 Query(질의) 언어로 사용하며, 연산이나 로직 처리할 기회를 봉쇄합니다.



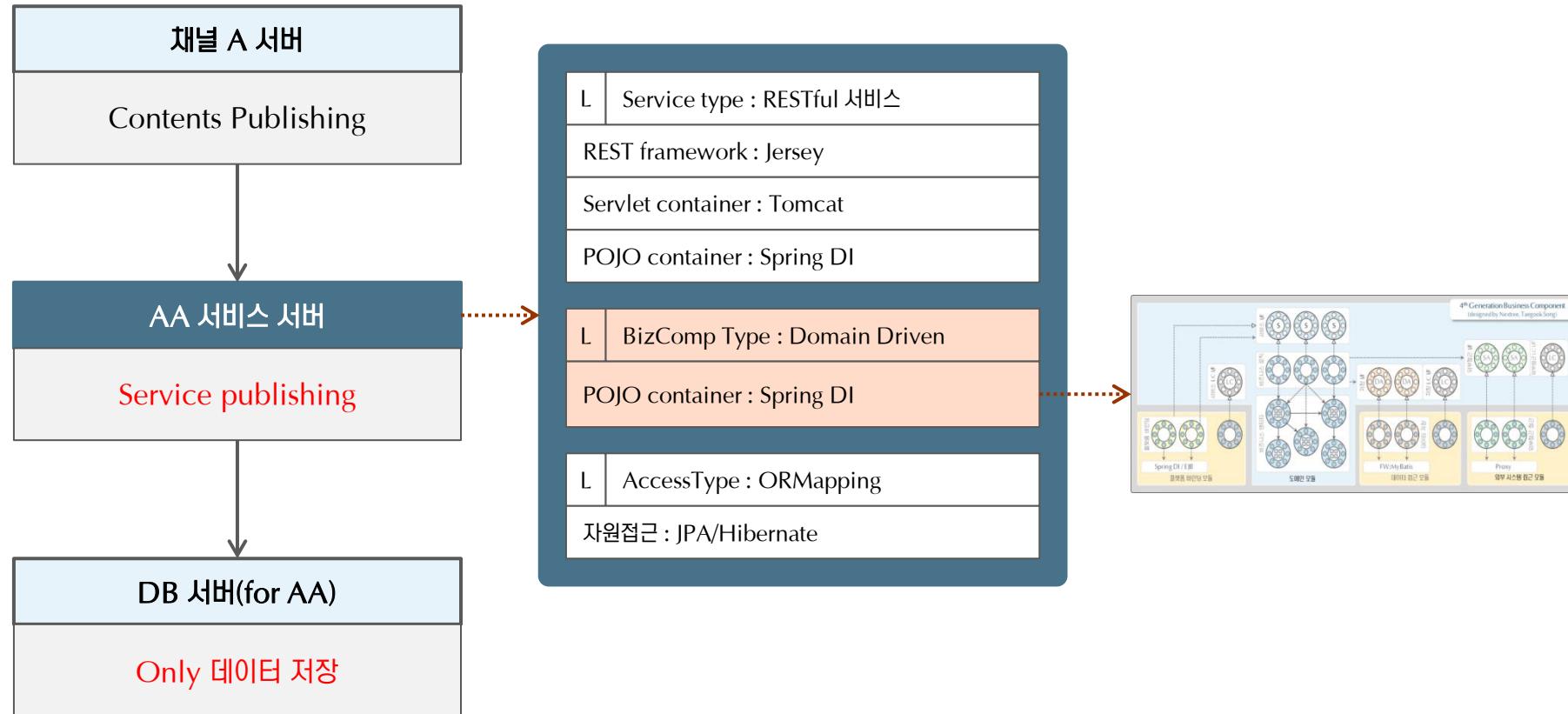
3.5 최적화 이미지(2/3) – Simple

- ✓ 업무를 단위로 횡으로 나누었습니다. → 업무에 따른 분리
- ✓ 서비스 서버를 새로 구성하고 DB의 SP를 이동했습니다. → 관심사 분리와 레이어드 아키텍처
- ✓ 업무 별로 DB 서버를 분리했습니다. → DB 서버에는 데이터를 두고 질의만 수행



3.5 최적화 이미지(3/3) – 서비스 서버

- ✓ 서비스 서버는 비즈니스 로직이라는 관심사를 한 곳에 모아서 처리하는 서버입니다.
- ✓ 외부에서 보면 (SOA의) 서비스로 보이고, 내부로 보면 비즈니스 컴포넌트입니다.
- ✓ 자원 접근 또는 데이터 맵핑은 SQL을 최소화 하는 방향으로 개선을 합니다. ← 과도한 SQL 코드량과 사용



3.6 요약

- ✓ 아키텍처 레벨의 최적화가 필요한 상황에서 특정 대상을 최적화하는 것은 의미가 없을 수 있습니다.
- ✓ 아키텍처 레벨은 최적화 관점 보다는 올바른 설계 관점에서 접근하는 것이 바람직합니다.
- ✓ 아키텍처 레벨의 최적화 원칙은 간단합니다. → 관심사의 분리(Separation of Concern)
- ✓ 최적화에 필요한 기술들을 나열해 보면 다음과 같습니다.

아키텍팅: Application	아키텍팅: Integration	아키텍팅: Ent. system	아키텍팅: 마이크로서비스
<ul style="list-style-type: none">✓ 품질 속성✓ 아키텍처 뷰와 뷰 포인트✓ 아키텍처 스타일과 패턴✓ 레이어드 아키텍처✓ 필터-파이프 아키텍처✓ 아키텍처 문서화	<ul style="list-style-type: none">✓ 컴포넌트와 커넥터✓ 커넥터✓ Integration 패턴✓ RPC와 강한 결합✓ 메시징과 느슨한 결합✓ Hexagonal 패턴과 RESTful	<ul style="list-style-type: none">✓ 아키텍팅 프로세스✓ 레이어 설계✓ Integration 설계✓ 자원 접근 설계✓ Special element 설계✓ 프레임워크와 플랫폼 설계와 활용	<ul style="list-style-type: none">✓ 연결 기술의 진화✓ SOA와 ROA✓ 서비스를 위한 컴포넌트✓ Microservices✓ 서비스 로깅과 모니터링✓ 서비스 보안: OAuth 2.0



4. 모델 레벨 최적화

-
- 4.1 문제 이해
 - 4.2 매트릭
 - 4.3 현재 모델
 - 4.4 문제의 핵심
 - 4.5 개선을 위한 아이디어
 - 4.6 모델 개선
 - 4.7 요약

4.1 문제 이해

- ✓ VehicleLight.java의 어떤 메소드가 Complexity 기준치를 넘었으며, 금방 더 복잡해 질 수 있습니다.

VehicleLight.java

```
private void exclusiveSignalOn() {
    //
    switch(signalColor) {
        case Green:
            if (redSignal.isSignalOn()) redSignal.turnOff();
            if (orangeSignal.isSignalOn()) orangeSignal.turnOff();
            if (!greenSignal.isSignalOn()) {
                greenSignal.turnOn();
                sayState();
            }
            break;
        case Red:
            if (greenSignal.isSignalOn()) greenSignal.turnOff();
            if (orangeSignal.isSignalOn()) orangeSignal.turnOff();
            if (!redSignal.isSignalOn()) {
                redSignal.turnOn();
                sayState();
            }
            break;
        case Orange:
            if (greenSignal.isSignalOn()) greenSignal.turnOff();
            if (redSignal.isSignalOn()) redSignal.turnOff();
            if (!orangeSignal.isSignalOn()) {
                orangeSignal.turnOn();
                sayState();
            }
            break;
    }
}
```

4.2 메트릭

- ✓ 해당 메소드의 복잡도(Cyclomatic Complexity)가 13으로 기준치를 초과했습니다.
- ✓ CodePro 기준 값은 6으로 지나치게 낮습니다. 개발에서 12 정도를 기준으로 하여 사용합니다.
- ✓ 문제는 새로운 색상의 신호등이 등장할 때 이 코드의 복잡도는 급증할 수 있습니다.
- ✓ 그리고 현재와 같은 방식의 처리는 중복 체크가 과도하게 이루어집니다.

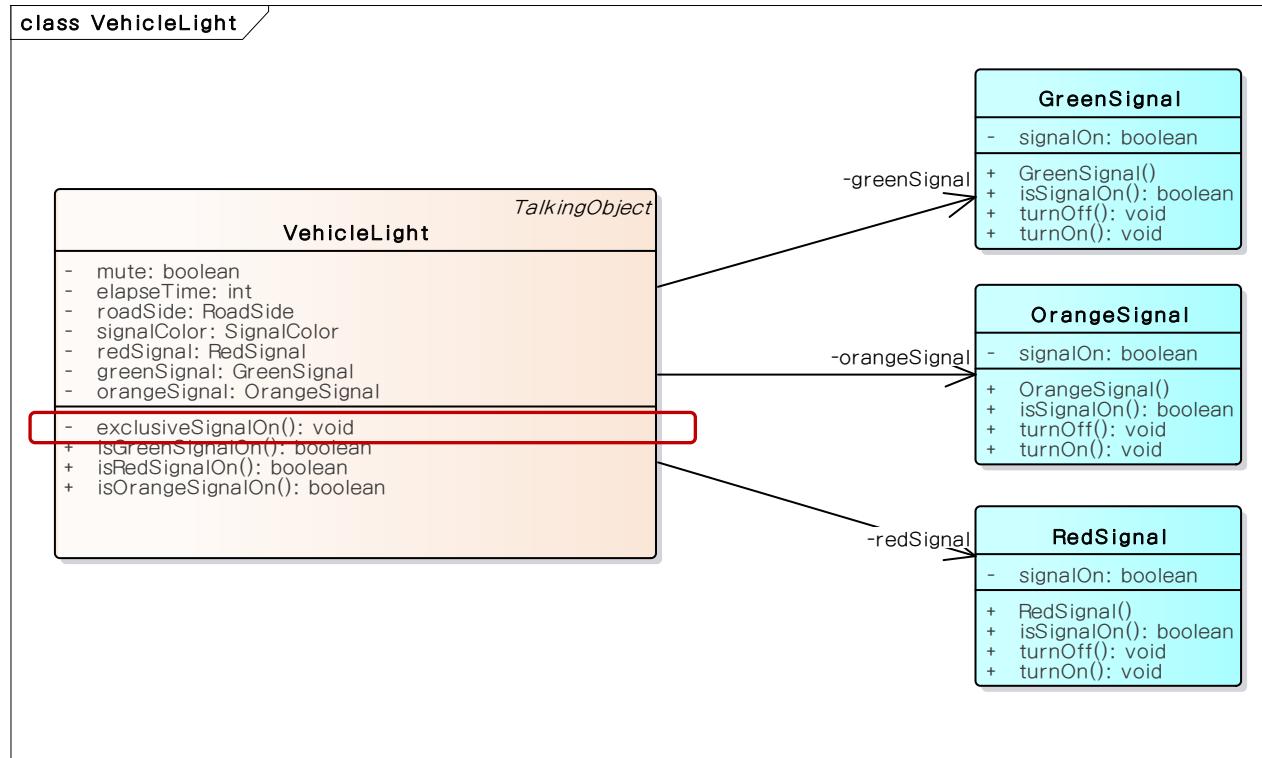
The screenshot shows the Eclipse IDE's Metrics view. The title bar includes tabs for Problems, Javadoc, Declaration, Console, Metrics, and Heap Dump Details. The Metrics tab is selected. The main area displays a table of metrics for the file 'namoo.pedxing.java.step08' at 16. 3. 9 오후 6:51. The table has two sections: 'Metric' and 'Minimum and Maximum'. The 'Metric' section lists various code statistics, and the 'Minimum and Maximum' section shows the minimum and maximum values for the 'Cyclomatic Complexity' metric, which is highlighted in red. A red circle highlights the value '13' under 'maximum'.

Metric	Value
Abstractness	29.1%
Average Block Depth	0.91
Average Cyclomatic Complexity	1.52
Average Lines Of Code Per Method	5.54
Average Number of Constructors Per Type	0.66
Average Number of Fields Per Type	2.04
Average Number of Methods Per Type	4.54
Average Number of Parameters	0.60
Comments Ratio	17.2%
Efferent Couplings	15
Lines of Code	875
Number of Characters	32,367
Number of Comments	151
Number of Constructors	16
Number of Fields	49
Number of Lines	1,400
Number of Methods	109
Number of Packages	12
Number of Semicolons	432
Number of Types	24
Weighted Methods	191

Name	Value
minimum	1
maximum	13

4.3 현재 모델

- ✓ 현재 모델에서는 세 가지 색상의 신호등이 있습니다. 화살표 모양의 지시등이 더 들어올 수 있습니다.
- ✓ 그러면 현재 코드에서 case 가 하나 더 추가되면 상황은 매우 복잡하게 전개됩니다.
- ✓ 개념을 변경하지 않으면(최적화 하지 않으면) 보다 더 복잡한 코드를 마주하게 됩니다.
- ✓ 어떤 고민이 필요할까요?



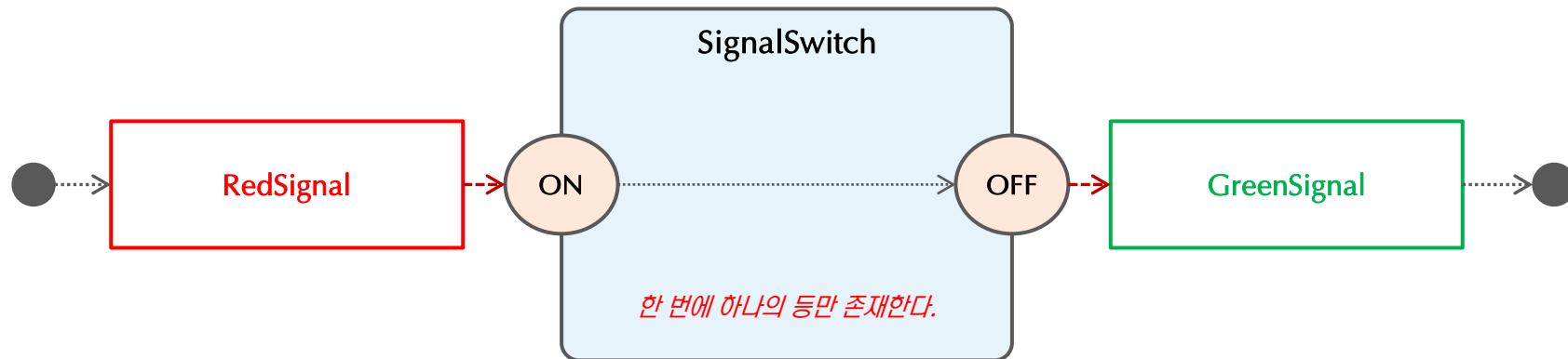
4.4 문제의 핵심

- ✓ 현재 모델에서는 세 개의 등이 있고, 그 중에 한 개만 켜져야하므로 매번 다른 등을 체크하고 있습니다.
- ✓ 뭔가 좀 더 근본적인 대책이 필요합니다.
- ✓ 그 대책은 코드 수준이 아닌, 모델 수준에서 이루어져야 합니다.
- ✓ “하나만 켜지는 것을 보장하는 것”이 이 문제의 핵심입니다.

```
private void exclusiveSignalOn() {
    //
    switch(signalColor) {
        case Green:
            if (redSignal.isSignalOn()) redSignal.turnOff();
            if (orangeSignal.isSignalOn()) orangeSignal.turnOff();
            if (!greenSignal.isSignalOn()) {
                greenSignal.turnOn();
                sayState();
            }
            break;
        case Red:
            if (greenSignal.isSignalOn()) greenSignal.turnOff();
            if (orangeSignal.isSignalOn()) orangeSignal.turnOff();
            if (!redSignal.isSignalOn()) {
                redSignal.turnOn();
                sayState();
            }
            break;
        case Orange:
            if (greenSignal.isSignalOn()) greenSignal.turnOff();
            if (redSignal.isSignalOn()) redSignal.turnOff();
            if (!orangeSignal.isSignalOn()) {
                orangeSignal.turnOn();
                sayState();
            }
            break;
    }
}
```

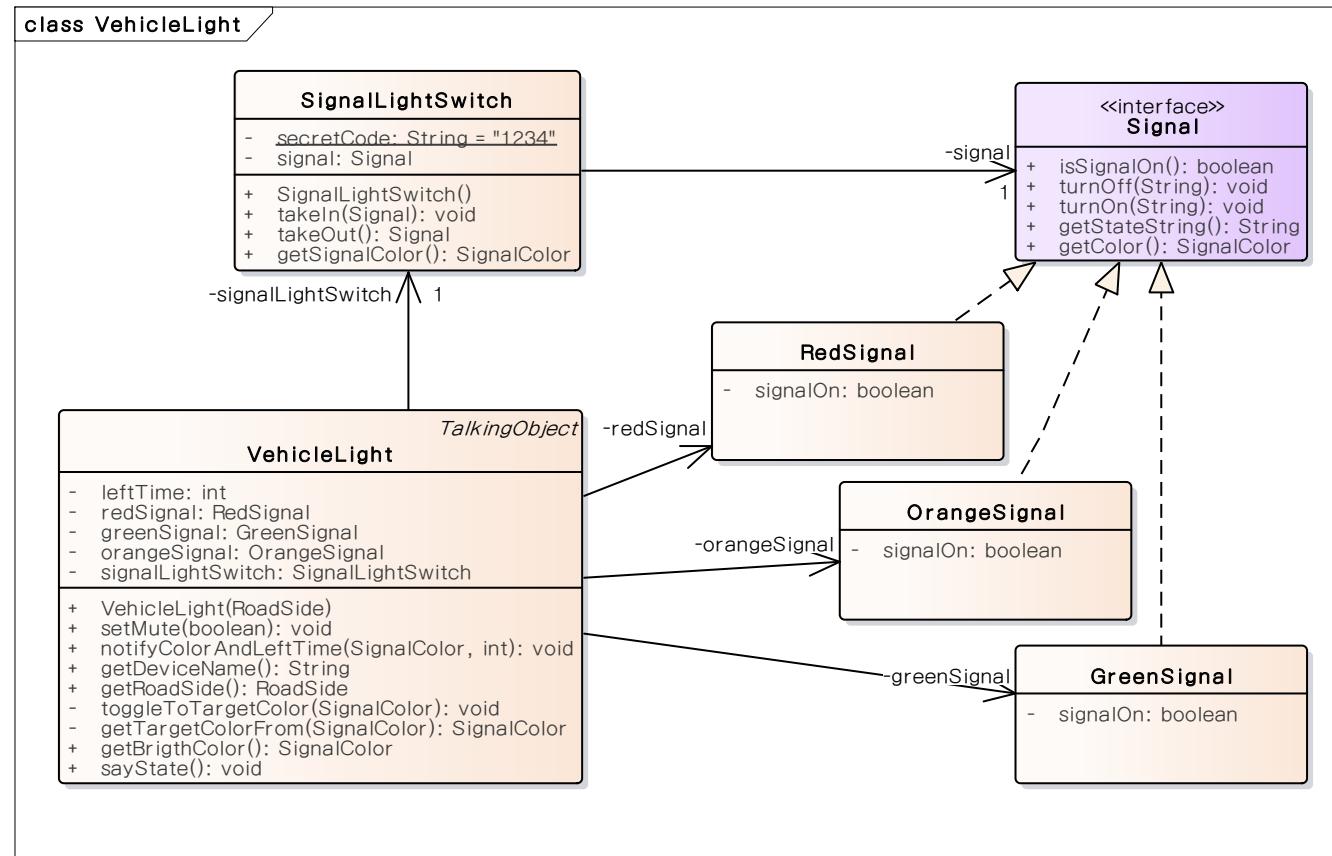
4.5 개선을 위한 아이디어(1/2)

- ✓ 문제를 해결하기 위해서는 아이디어나 새로운 개념이 필요합니다. 코드 최적화 수준을 벗어날 필요가 있습니다.
- ✓ 물론 현장에서는 새로운 아이디어나 개념이 필요없을 정도로 모든 것이 정교하게 정의되어 있습니다.
- ✓ 객체 모델 수준에서 문제를 풀어가기 위해 새로운 역할을 식별하고 역할 담당 객체를 등장 시킵니다.
- ✓ 신호등은 여러 개지만, 스위치는 하나라고 가정하면, 반드시 하나만 켜져야 하는 문제를 해결할 수 있습니다.



4.5 개선을 위한 아이디어(2/2)

- ✓ 새로운 역할 담당자의 등록, 그리고 세 개의 신호등(signal)을 한꺼번에 다루기 위한 인터페이스 정의 등으로 모델이 변경되었습니다.
- ✓ 객체는 늘었지만, 각 객체의 처리 작업은 단순하게 변경될 것입니다.



4.6 모델 개선(1/2)

- ✓ 이제 해당 메소드의 Cyclomatic Complexity는 4로 낮아졌습니다.
- ✓ 뿐만 아니라 이제는 어떤 유형의 신호등이 추가되더라도 “하나만 켜져야”하는 규칙을 지킬 수 있습니다.
- ✓ 복잡도가 낮아지면서 코드도 한결 간단해 졌으며, 가독성과 유지보수성이 높아졌습니다.

VehicleLight.java

```
synchronized private void toggleToTargetColor(SignalColor signalColor) {  
    //  
    talkingAtRight("차량신호등이 바뀝니다.");  
    signalLightSwitch.takeOut();  
  
    switch(signalColor) {  
        case Red:  
            signalLightSwitch.takeIn(redSignal);  
            break;  
  
        case Green:  
            signalLightSwitch.takeIn(greenSignal);  
            break;  
  
        case Orange:  
            signalLightSwitch.takeIn(orangeSignal);  
            break;  
    }  
}
```

4.6 모델 개선[2/2]

- ✓ 새로운 책임에 따른 새로운 역할 담당자인 SignalLightSwitch이 등장했습니다.
- ✓ 이름 그대로 들어오는 신호등을 켜주고, 나가는 신호등을 꺼주는 일을 합니다.
- ✓ 코드 내용도 간단 명료합니다. 이제 모델 수준의 최적화 작업이 끝났습니다.

SignalLightSwitch.java

```
public class SignalLightSwitch {  
    //  
    private static String secretCode = "1234";  
    private Signal signal;  
  
    ...  
  
    public void takeIn(Signal signal) {  
        this.signal = signal;  
        signal.turnOn(secretCode);  
    }  
  
    public Signal takeOut() {  
        signal.turnOff(secretCode);  
        return signal;  
    }  
  
    public SignalColor getSignalColor() {  
        return signal.getColor();  
    }  
}
```

4.7 요약

- ✓ 새로운 책임에 따른 새로운 역할 담당자인 SignalLightSwitch이 등장했습니다.
- ✓ 이름 그대로 들어오는 신호등을 켜주고, 나가는 신호등을 꺼주는 일을 합니다.
- ✓ 코드 내용도 간단 명료합니다. 이제 모델 수준의 최적화 작업이 끝났습니다.
- ✓ 모델 레벨의 최적화를 위해 어느 정도의 객체 모델링 역량을 갖추어야 합니다.



5. 프로그램 레벨 최적화

[참조] <https://www.youtube.com/watch?v=f2aNWtt0QRo>

Chris Bailey – IBM Java Service Architect
26th September 2013

Memory-Efficient Java Code
Write low overhead application code

How to Write Memory-Efficient Java Code

JavaOne

구독중 7,100

3,165

+ 추가 공유 *** 더보기

5.1 Java 객체의 크기

5.2 상황 이해

5.3 라이브러리 분석

5.4 인터페이스 개선

5.5 요약

[공유 #1] LBS 튜닝

5.1 Java 객체의 크기(1/3)

✓ int value 값은 32bits입니다. Integer object의 크기는 얼마일까요? (32bit에서)

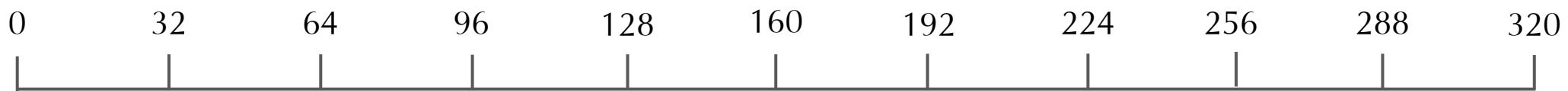
1. 32 bits (1:1)
2. 48 bits (1.5:1)
3. 64 bits (2:1)
4. 96 bits (3:1)
5. 128 bits(4:1)

```
public static void main(String[] args) {  
    Integer myInteger = new Integer(10);  
}
```

5.1 Java 객체의 크기(2/3)

✓ Java 객체의 메타 데이터

- Class : 클래스 정보에 대한 포인터
- Flags : shape, 해시 코드, 등
- Lock : 객체 잠금 정보
- Size : 배열의 길이



32 비트	Class	Flags	Locks	int: 10	Java Object	
	Class	Flags	Locks	Size	int: 10	
(크기 비율) Integer 객체: int = 4:1						
64 비트	Class	Flags	Locks	int, eg. 10	Java Object	
	Class	Flags	Locks	Size	int, eg. 10	Array Object
(크기 비율) Integer 객체: int = 9:1						

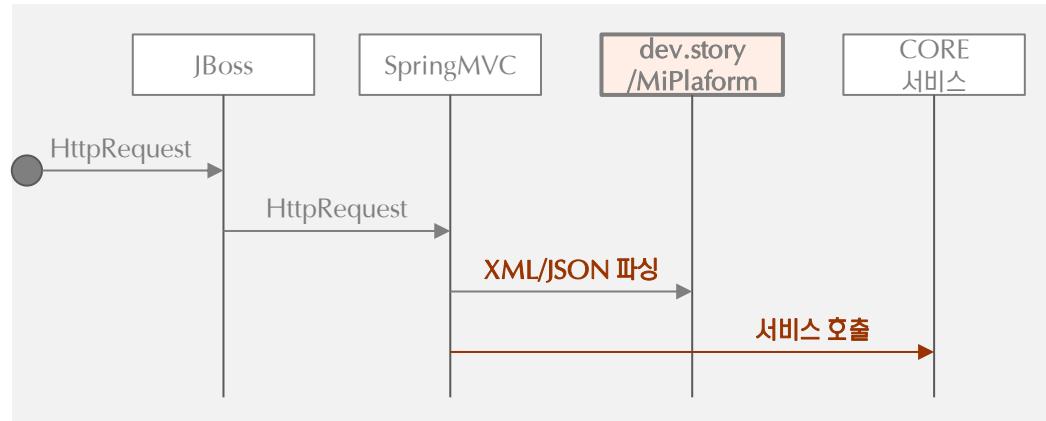
5.1 Java 객체의 크기(3/3)

- ✓ 객체의 크기를 염두에 두고 프로그래밍에서 사용하여야 합니다.
- ✓ Map 객체는 빠른 찾기 성능을 제공해 주는 반면 객체 크기가 매우 크며, 오버헤드 역시 큽니다.
- ✓ 적은 수의 객체는 ArrayList와 같은 작은 컬렉션 객체를 활용하여 처리하여야 합니다.

Collection	Default Capacity	Empty Size	10K Overhead
HashSet	16	144	360K
HashMap	16	128	360K
Hashtable	11	104	360K
LinkedList	1	48	240K
ArrayList	10	88	40K
StringBuffer	16	72	24

5.2 상황 이해

- ✓ 서비스 요청을 위해 HTTP 프로토콜을 타고 들어온 HttpRequest를 받아서 처리하는 과정을 분석합니다.
- ✓ 첫 번째 분석은 CORE 서비스 자체의 인터페이스 설계 방식의 적절성 여부입니다.
- ✓ 두 번째 분석은 매개 변수(parameter)로 사용되는 클래스인 RSPDataSet과 RSPResultSet입니다.
- ✓ 서비스 인터페이스 설계, 메시지 변환, 매개 변수 구성을 잘못하고 있는지 확인해야 합니다.



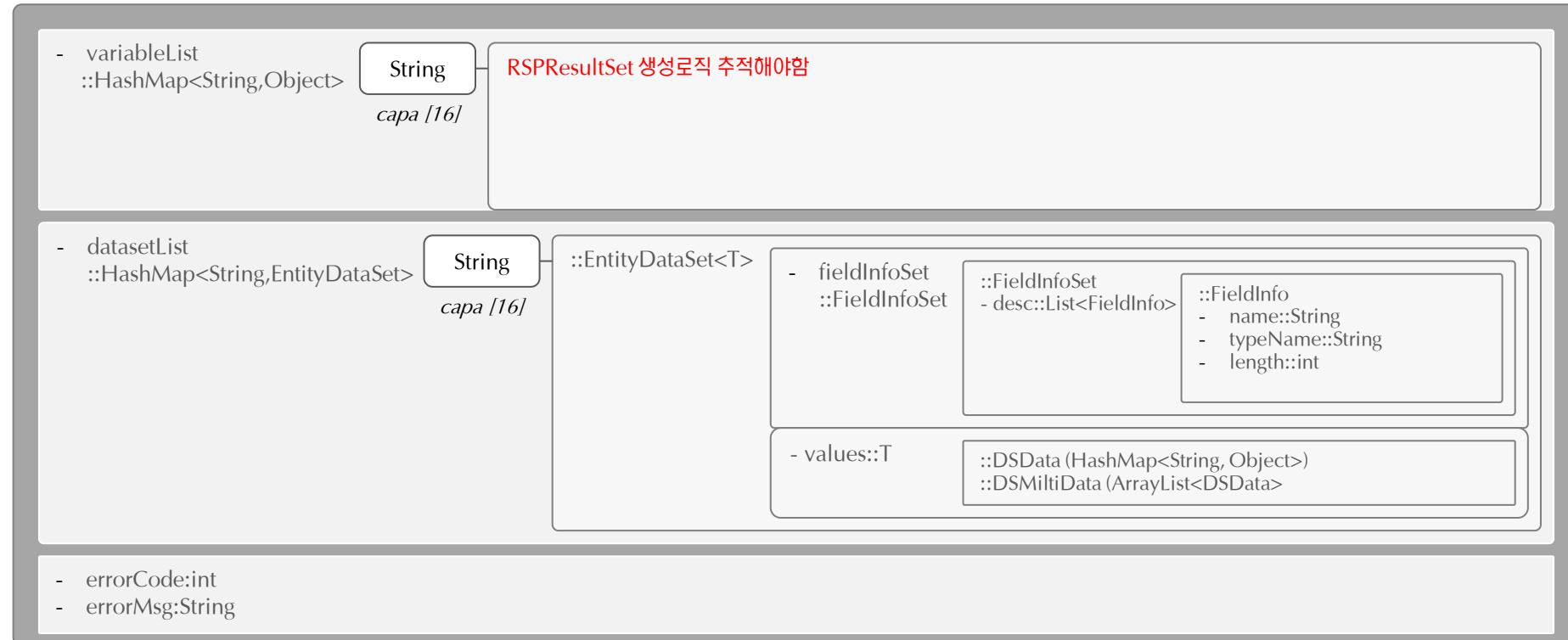
5.3 라이브러리 분석 1

- ✓ RSPDataSet 객체는 모든 서비스 오퍼레이션의 매개변수(parameter)로 사용합니다. RSPDataSet 클래스는 여러 Map을 사용하고, Map 안에 Map을 품고 있는 “거대한(2000바이트 +)” 객체입니다.



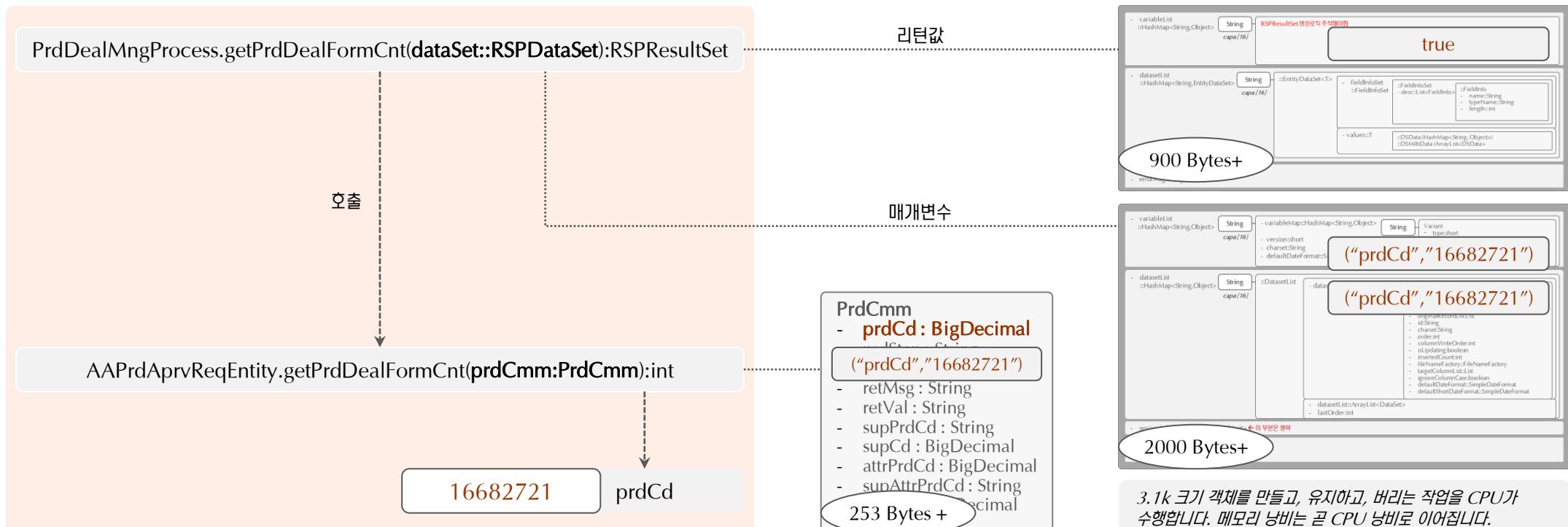
5.3 라이브러리 분석(1/2)

- ✓ RSPResultSet 객체는 모든 서비스 오퍼레이션의 리턴(=반환) 타입으로 사용합니다. 반환하는 데이터가 어떤 형태이든 모두 RSPResultSet 객체로 감싸서 반환하기 때문에 데이터 추적이 어렵습니다.
- ✓ RSPResultSet 객체는 MiPlatform, JSP 등 UI에서 요구하는 다양한 형태의 데이터를 모두 담고 있습니다.



5.3 라이브러리 분석(2/2)

- ✓ 8바이트 상품 코드를 저장한 후 메모리 사용량을 측정해 보면 거의 2000바이트 정도의 메모리를 사용하고, 리턴 객체에 900 바이트 정도를 사용합니다.
 - ✓ 8바이트 상품코드를 가진 PrdCmm 객체는 253바이트의 메모리를 사용, 모두 합쳐 3.1k 정도 사용합니다.



8바이트 상품코드를 보내서 boolean 값을 리턴하는데 사용하는 메모리: **3.1k Bytes +**

5.4 인터페이스 개선(2/2)

- ✓ 완전한 단어를 사용하여 간결하게 이름을 부여하되, 업무 관점에 의미가 명확한 이름을 사용합니다.
- ✓ 매개변수는 컬렉션 타입, 일반화 수준이 높은 타입 등을 피하고 구체적인 타입과 이름을 사용하며, 컬렉션 타입을 사용할 때는 List와 같이 단순한 것을 사용해야 합니다. 그리고, 리턴타입 역시 구체적인 타입을 사용합니다.
- ✓ 대개는 primitive 타입, 사용자 정의 타입, 두 타입의 List, OffsetList, NameValueList 정도로 제한합니다.

상품 코드 “1234567”인 상품이 딜 상품인지 여부를 확인하는 좋지 않은 서비스 이름

PrdDealMngProcess.getPrdDealFormCnt(dataSet::RSPDataSet) :Map<String, EntityDataSet>

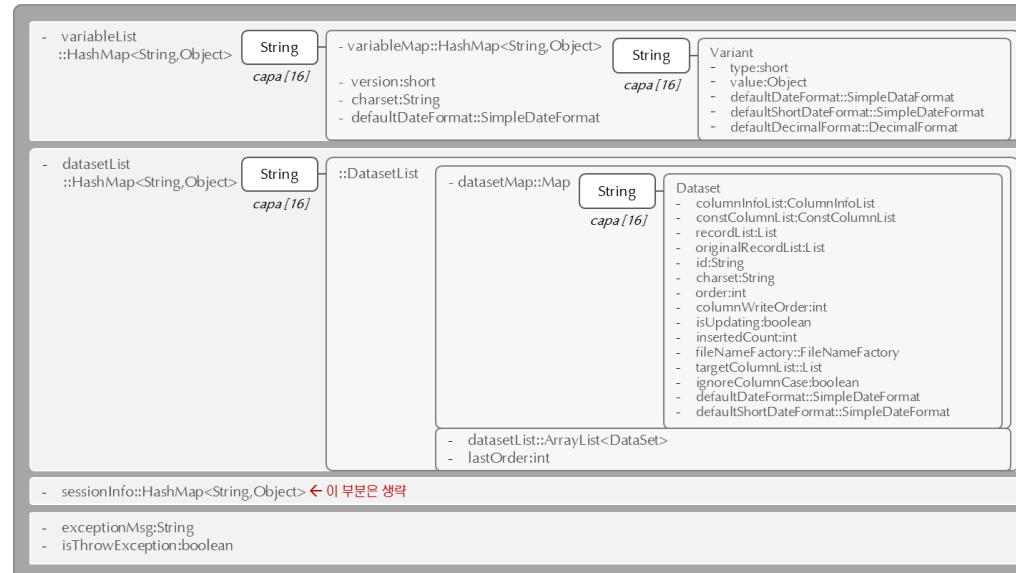
개선 예제

상품 코드 “1234567”인 상품이 딜 상품인지 여부를 확인하는 좋은 서비스 이름

DealProductService.isDealProduct(String productId) :boolean

5.5 요약

- ✓ 프로그래밍 레벨의 최적화는 대상은 자료 구조, 알고리즘, 객체 관계 등으로 매우 다양합니다.
- ✓ 프로그램을 단순하게 유지함으로써 유지보수를 용이하게 하고, 성능 최적화는 자원 활용 방법을 이용하여 풀어 가는 것이 비즈니스 지원 애플리케이션 설계의 방향입니다.
- ✓ 사례를 통해서 객체의 크기를 전혀 고려하지 않은 라이브러리 설계의 위험성에 대해 알아 보았습니다.



[공유 #1] LBS 시스템 튜닝

✓ 자바 리플렉션을 사용한 로직

- 성능진단시 CPU 사용율이 90~ 95%까지 사용하는 로직을 프로파일링 을 한 결과 객체간 Value 복제 및 전환시 자바 리플렉션을 사용한 로직이 문제가 되어 해당 로직을 각 항목별로 Getter 와 Setter를 사용한 로직으로 변경하니 CPU 사용율이 20% 이하로 줄어들었습니다.

✓ 과도한 파일 I/O로 인한 서버간 접속 장애

- 채널서버와 도메인 서버간 임계치 테스트 진행시 채널서버에서 도메인 서버 접속이 안됨을 확인하였습니다.
- 리눅스 서버에서 톰캣의 스레드 오픈 파일개수로 카운트되어 대량의 트래픽이 발생하여 오픈파일 개수가 서버의 설정치를 넘어가게 되어 접속 장애가 발생하였습니다.
- 리눅스 파일 오픈개수를 조정(/etc/security/limits.conf 수정)하고, GC 옵션을 조정하였습니다.
- 참고 <http://lunatine.net/limits-conf-nofile-big-value-effect/>

✓ 가상서버에서의 CPU 사용율 증가

- 물리서버와 가상서버가 혼재된 00서버의 상황에서 동일 로직 / 동일 사양의 서버에서 서로 다른 결과가 확인되어 원인 파악한 결과 가상서버에서는 CPU사용율이 물리 서버보다 증가하여 성능에 영향을 주었습니다.
- "빅데이터: 플럼(Flume) 토플로지 설계" 의 4.2 항목 참조(<http://www.nextree.co.kr/p2704>)

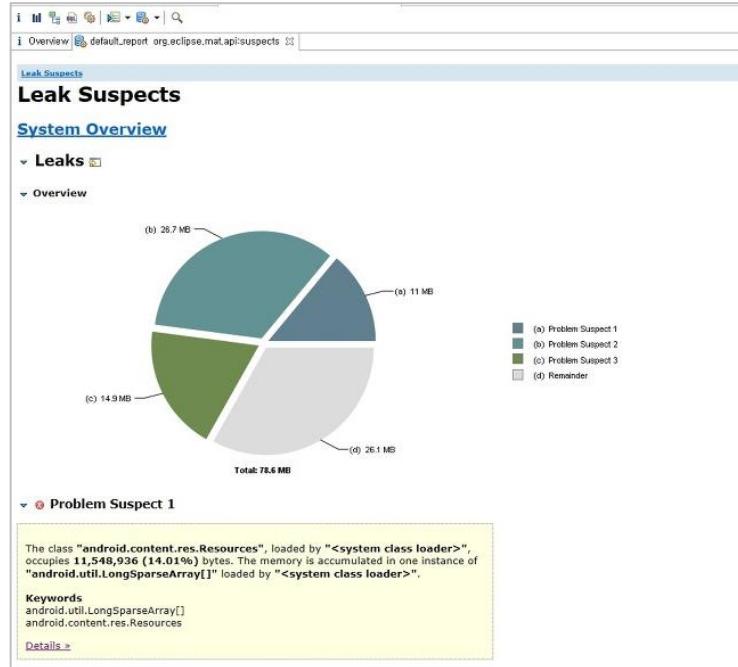


6. 도구 I - MAT

-
- 6.1 Memory Analyzer Tool
 - 6.2 Heap Dump 개요
 - 6.3 Heap Dump 생성
 - 6.4 Dump 파일 열기
 - 6.5 Heap Dump 열기
 - 6.6 히스토그램
 - 6.7 Dominator Tree
 - 6.8 Inspector
 - 6.9 보고서

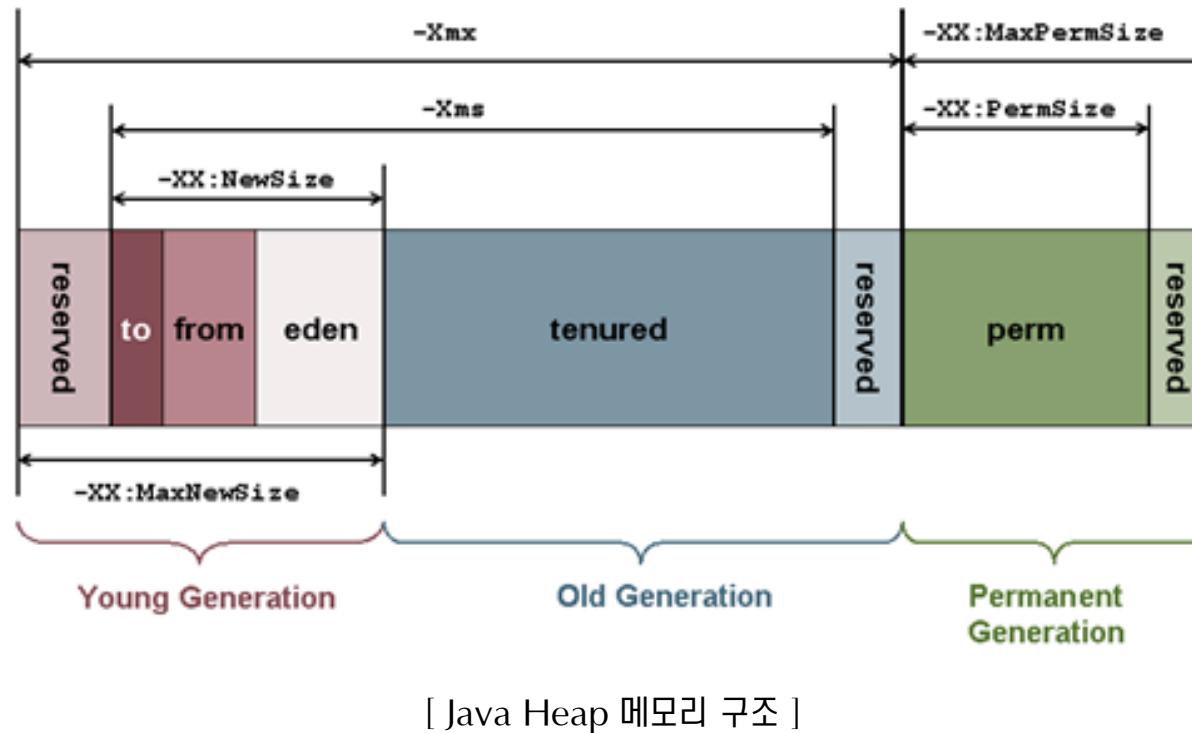
6.1 Memory Analyzer Tool

- ✓ MAT(Memory Analyzer Tool)는 빠르고 다양한 기능을 갖춘 JAVA Heap 분석 도구입니다.
- ✓ Memory Leak을 발견하고, 메모리 사용량을 줄이거나 GC 대상에서 제외된 요소를 찾을 수 있습니다.
- ✓ MAT를 사용하여 수천만에 이르는 Object의 Heap dump를 추출할 수 있습니다.
- ✓ Report 기능을 통해 Memory Leak의 원인을 파악할 수 있습니다.



6.2 Heap Dump 개요

- ✓ 아래 그림은 특정 시점에서 JAVA 응용프로그램에 대한 Heap 정보의 스냅 샷입니다.
- ✓ HPROF라는 바이너리 형식으로 저장됩니다.
- ✓ 모든 Object와 Field, Primitive Type, Object Reference 정보를 가지고 있습니다.



6.3 Heap Dump 생성(1/3)

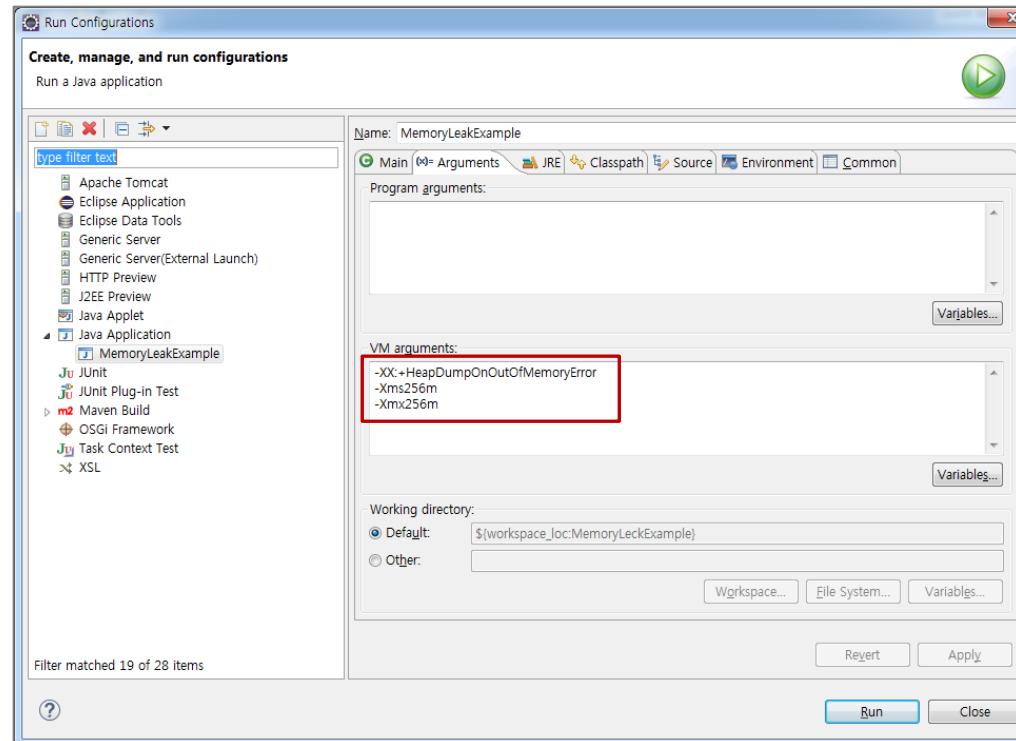
- ✓ Out Of Memory Error 발생하는 시점의 Heap Dump를 생성하겠습니다.
- ✓ 다음과 같이 Out Of Memory Error를 발생하는 코드를 작성합니다.

```
public static void main(String[] args) {  
    List<String> list = new ArrayList<String>();  
    while(true) {  
        list.add("Out Of Memory Error soon");  
    }  
    return null;  
}
```

MemoryLeakExample.java

6.3 Heap Dump 생성[2/3]

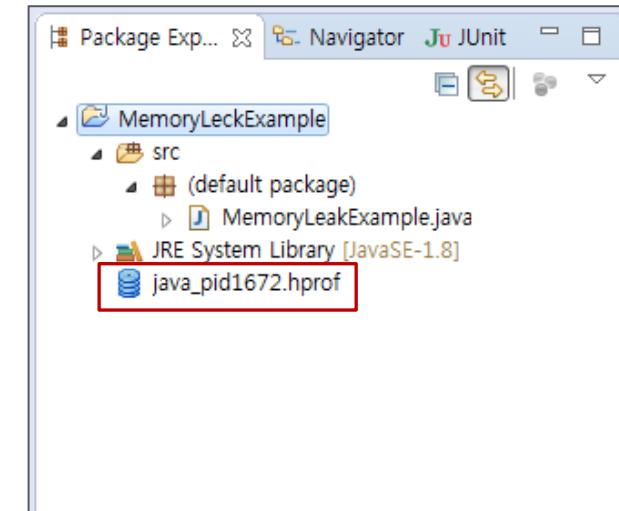
- ✓ Run Configurations 창을 불러옵니다.
- ✓ -XX:+HeapDumpOnOutOfMemoryError 옵션을 추가하여 Out of Memory 발생 시점에 Heap Dump를 생성하도록 설정합니다.
- ✓ -Xms256m, -Xmx256m 옵션을 추가하여 Heap Size를 256MB로 설정합니다.



6.3 Heap Dump 생성(3/3)

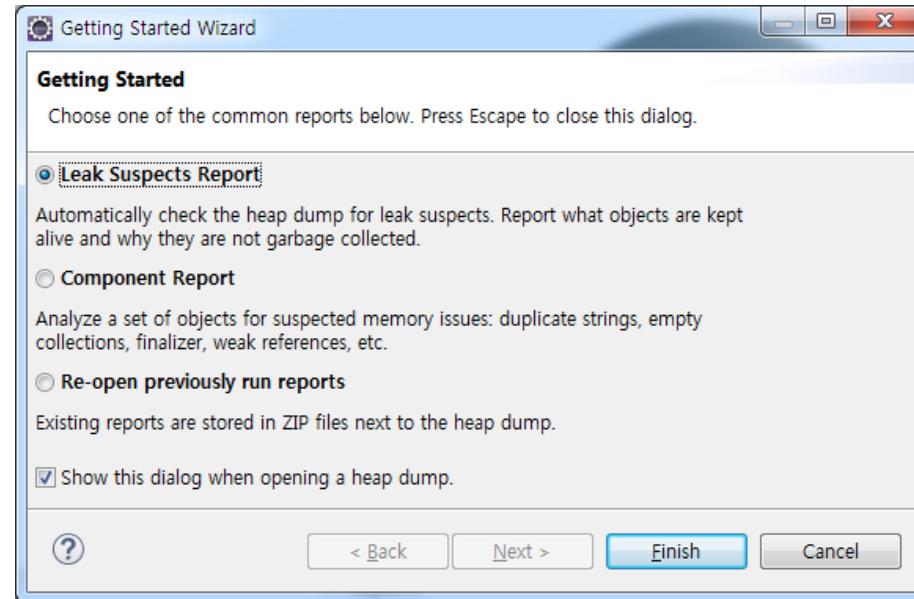
- ✓ 실행 후 Out Of Memory Error 발생을 콘솔창에서 확인할 수 있습니다.
- ✓ Heap Dump 파일명, 크기, 생성 소요시간 등의 정보가 콘솔창에 보여집니다.
- ✓ 프로젝트를 Refresh 한 후 Heap Dump파일이 생성된 것을 확인할 수 있습니다. (java_pid****.hprof)
- ✓ Heap Dump 파일 생성 경로를 명시하지 않은 경우 기본설정에 의해 해당 프로젝트 경로에 생성됩니다.

The screenshot shows the Eclipse IDE's Console view. It displays the following text:
<terminated> MemoryLeakExample [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (2016. 3. 8. 오후 2:33)
java.lang.OutOfMemoryError: Java heap space
Dumping heap to java_pid3204.hprof ...
Heap dump file created [1262865234 bytes in 2.331 secs]
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at java.util.Arrays.copyOf(Unknown Source)
at java.util.Arrays.copyOf(Unknown Source)
at java.util.ArrayList.grow(Unknown Source)
at java.util.ArrayList.ensureExplicitCapacity(Unknown Source)
at java.util.ArrayList.ensureCapacityInternal(Unknown Source)
at java.util.ArrayList.add(Unknown Source)
at MemoryLeakExample.main(MemoryLeakExample.java:11)



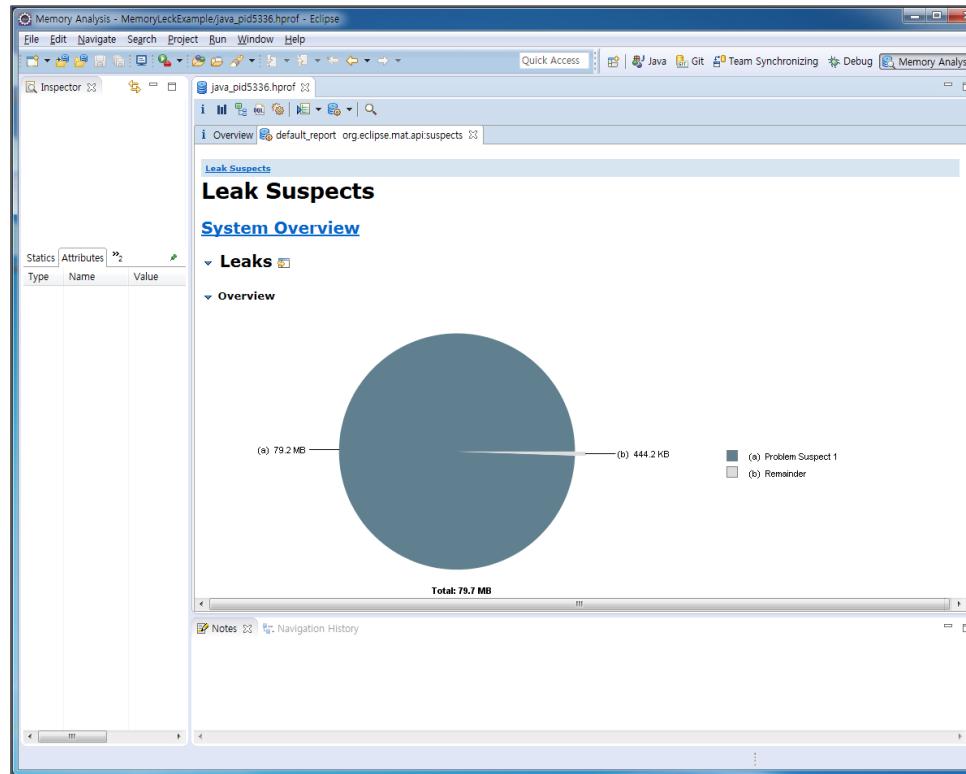
6.4 Dump 파일 열기(1/3)

- ✓ Memory Analysis 탭으로 이동합니다.
- ✓ Open Heap Dump 아이콘을 클릭합니다.
- ✓ 생성된 Heap Dump파일을 선택합니다. (java_pid****.hpof)
- ✓ Leak Suspects Report 선택후 Finish를 클릭합니다.



6.4 Dump 파일 열기(2/3)

- ✓ Overview 탭에서 Heap Dump 분석 요약정보를 보여줍니다.
- ✓ Reports 중 Leack Suspects 항목에서 Memory Leak의 원인으로 추정되는 정보를 확인할 수 있습니다.



6.4 Dump 파일 열기(3/3)

- ✓ Memory Leak의 원인으로 추정되는 Suspect 1의 상세 정보를 보여줍니다.
 - 참고: MAT는 Leak 원인을 추정할 뿐 판단하지 않습니다. Leak 원인 판단은 사용자의 몫입니다.
- ✓ Problem Suspect 1 상세 내용을 보면 이 쓰레드가 점유하는 메모리 용량과 전체 Heap에서 차지하는 비율이 얼마인지 알 수 있습니다.

▼ **Problem Suspect 1**

The thread **java.lang.Thread @ 0xf4f3aaa8 main** keeps local variables with total size **83,071,632 (99.46%) bytes**.

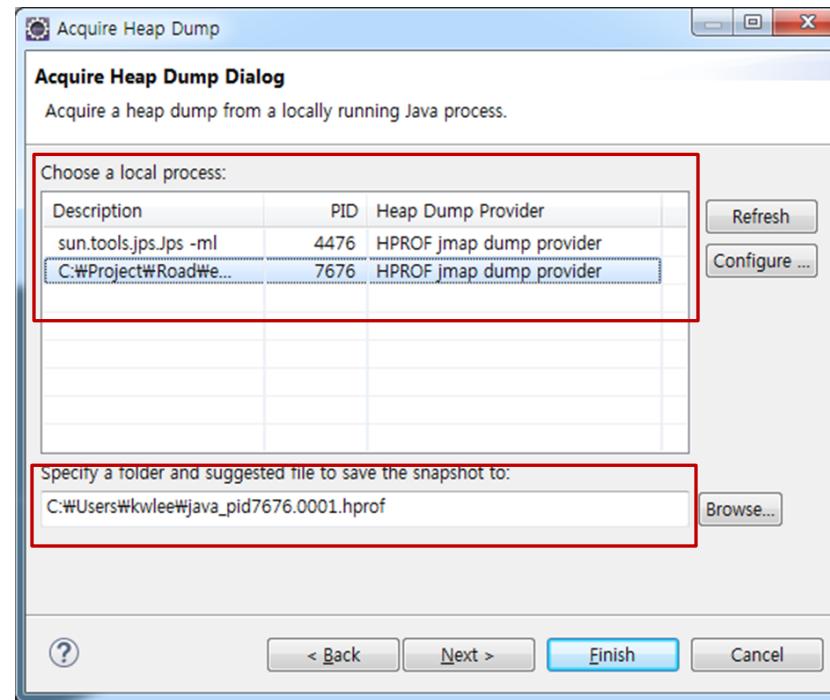
The memory is accumulated in one instance of "**java.lang.Object[]**" loaded by "**<system class loader>**".

The stacktrace of this Thread is available. [See stacktrace](#).

Keywords
java.lang.Object[]
[Details »](#)

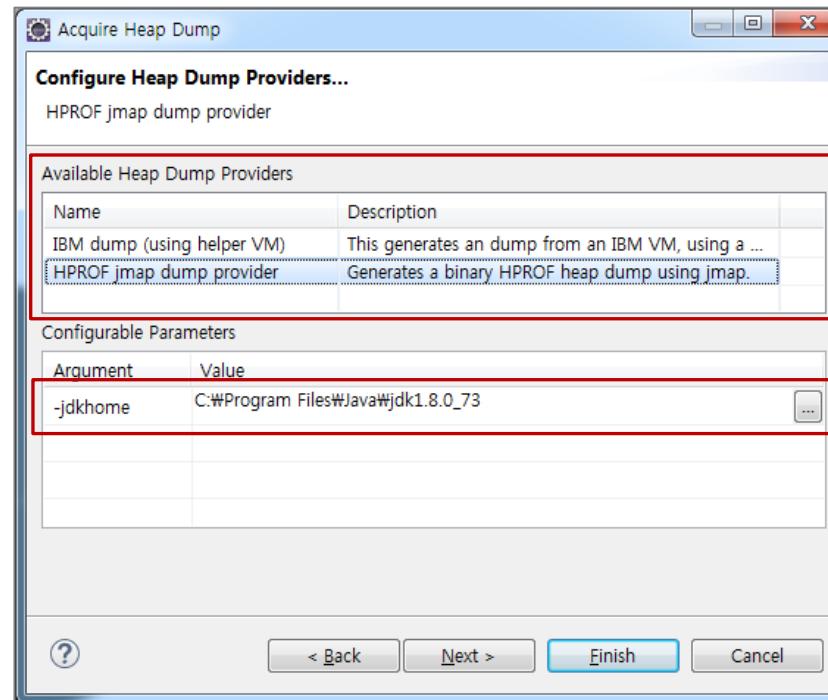
6.5 Heap Dump 얻기(1/2)

- ✓ 로컬에 실행중인 JVM 프로세스 중 하나를 선택해 Heap Dump를 추출할 수 있습니다.
- ✓ Acquire Heap Dump 버튼을 클릭하고, 로컬 프로세스 중 원하는 프로세스를 선택합니다.
- ✓ Heap Dump 파일을 저장할 경로를 설정합니다.



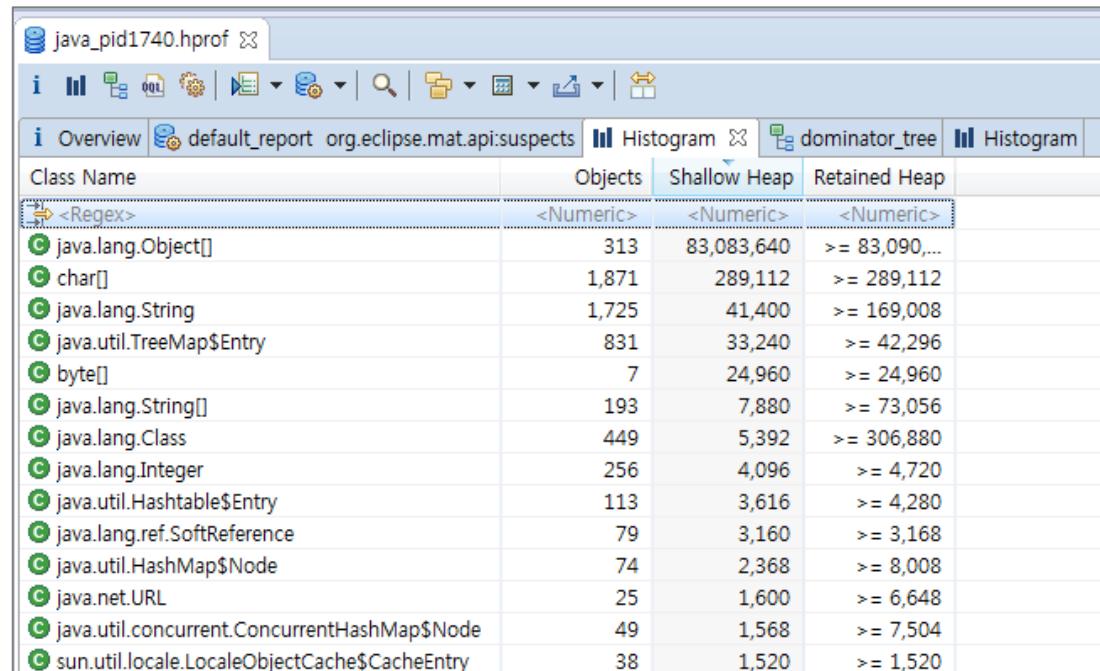
6.5 Heap Dump 얻기[2/2]

- ✓ Configure 선택 후, Heap Dump Provider 목록에서 HPROF jmap dump provider를 선택합니다.
- ✓ -jdkhome 항목에 설치된 JDK 경로를 설정하고, Finish를 선택합니다.
- ✓ Acquire Heap Dump 기능으로 Heap Dump 파일 생성시 자동으로 해당 파일이 로딩됩니다.



6.6 히스토그램(1/2)

- ✓ Leak Suspects에서 제공하는 정보만으로 Memory Leak 원인을 판단하기 어렵습니다.
- ✓ Histogram은 특정 Class별 메모리 사용량을 보여줍니다.
- ✓ 각 Class의 Instance 수와 Shallow Heap, Retained Heap 정보를 보여줍니다.

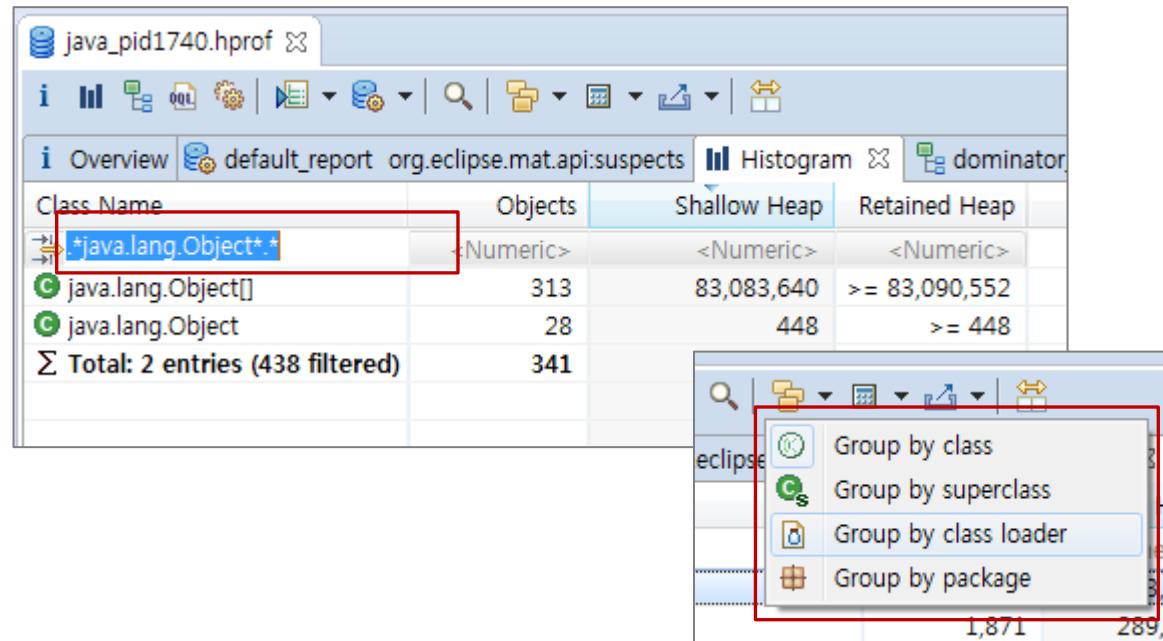


The screenshot shows the Eclipse MAT (Memory Analysis Tool) interface with the 'Histogram' tab selected. The main window displays a table of memory usage statistics for different Java classes. The columns represent the number of objects, shallow heap size, and retained heap size for each class. The table includes entries for common Java types like Object[], String, and Integer, along with some internal classes from the java.util package.

Class Name	Objects	Shallow Heap	Retained Heap
<Regex>	<Numeric>	<Numeric>	<Numeric>
java.lang.Object[]	313	83,083,640	>= 83,090,...
char[]	1,871	289,112	>= 289,112
java.lang.String	1,725	41,400	>= 169,008
java.util.TreeMap\$Entry	831	33,240	>= 42,296
byte[]	7	24,960	>= 24,960
java.lang.String[]	193	7,880	>= 73,056
java.lang.Class	449	5,392	>= 306,880
java.lang.Integer	256	4,096	>= 4,720
java.util.Hashtable\$Entry	113	3,616	>= 4,280
java.lang.ref.SoftReference	79	3,160	>= 3,168
java.util.HashMap\$Node	74	2,368	>= 8,008
java.net.URL	25	1,600	>= 6,648
java.util.concurrent.ConcurrentHashMap\$Node	49	1,568	>= 7,504
sun.util.locale.LocaleObjectCache\$CacheEntry	38	1,520	>= 1,520

6.6 히스토그램(2/2)

- ✓ 필요에 따라 Class loader, Package 단위로 Grouping 할 수 있습니다.
- ✓ Package로 Grouping하면 자신이 개발한 Object에 중점을 두고 분석하기에 적합합니다.
- ✓ 정규 표현식으로 원하는 패턴과 일치하는 항목으로 필터링 할 수 있습니다.



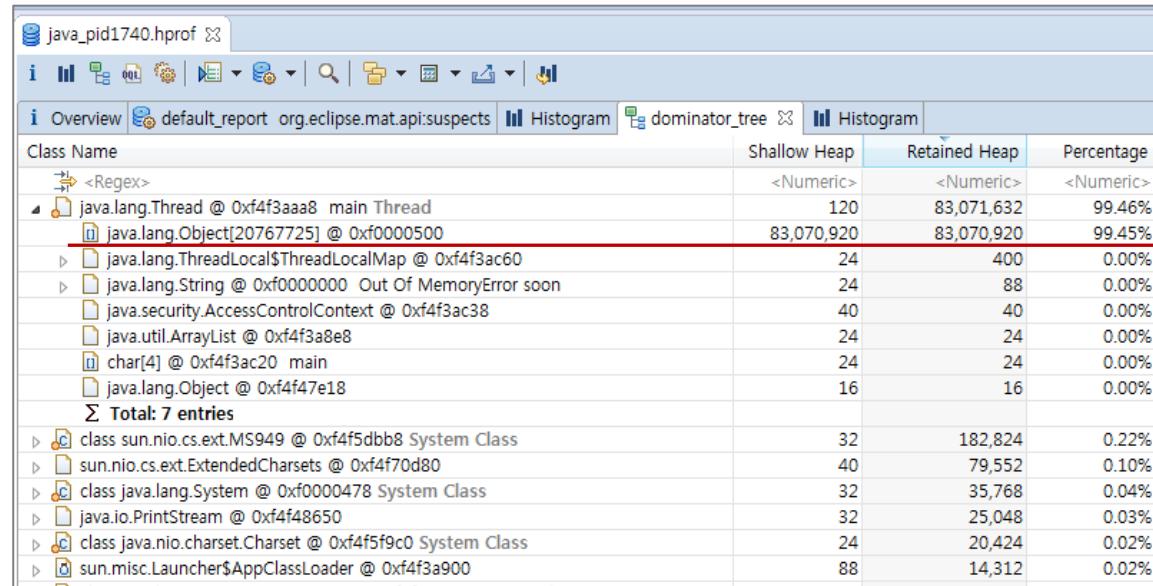
6.7 Dominator Tree (1/2)

- ✓ Dominator Tree 화면은 가장 큰 메모리를 차지하는 순으로 Object 및 Class를 나열합니다.
- ✓ Leak Suspects 와 달리 각 Object 및 Class의 상세정보를 확인할 수 있습니다.
- ✓ Leak Suspects에서 얻은 정보를 보다 정밀하게 검토하기에 적절합니다.

Class Name	Shallow Heap	Retained Heap	Percentage
<Regex>	<Numeric>	<Numeric>	<Numeric>
java.lang.Thread @ 0xf4f3aaa8 main Thread	120	83,071,632	99.46%
sun.nio.cs.ext.MS949 @ 0xf4f5dbb8 System Class	32	182,824	0.22%
sun.nio.cs.ext.ExtendedCharsets @ 0xf4f70d80	40	79,552	0.10%
java.lang.System @ 0xf0000478 System Class	32	35,768	0.04%
java.io.PrintStream @ 0xf4f48650	32	25,048	0.03%
java.nio.charset.Charset @ 0xf4f5f9c0 System Class	24	20,424	0.02%
sun.misc.Launcher\$AppClassLoader @ 0xf4f3a900	88	14,312	0.02%
java.lang.ProcessEnvironment @ 0xf4f58420 System Class	32	10,688	0.01%
java.io.File @ 0xf4f5a9b8 System Class	48	6,760	0.01%
sun.nio.cs.StandardCharsets @ 0xf4f5f7e8 System Class	160	5,912	0.01%
sun.util.locale.BaseLocale @ 0xf4f59bf0 System Class	8	5,504	0.01%
java.lang.Integer\$IntegerCache @ 0xf4f5b3c0 System Class	16	5,120	0.01%
java.util.Locale @ 0xf4f5a470 System Class	152	3,488	0.00%
java.lang.ClassLoader @ 0xf4f63778 System Class	32	3,072	0.00%
sun.misc.MetaIndex @ 0xf4f5a878 System Class	8	2,728	0.00%
sun.misc.VM @ 0xf4f5fc80 System Class	72	2,704	0.00%
java.lang.CharacterDataLatin1 @ 0xf4f598f0 System Class	24	2,696	0.00%
sun.misc.MetaIndex @ 0xf4f50cb0	24	2,328	0.00%
sun.usagetracker.UsageTrackerClient @ 0xf4f58668 System Class	136	1,984	0.00%

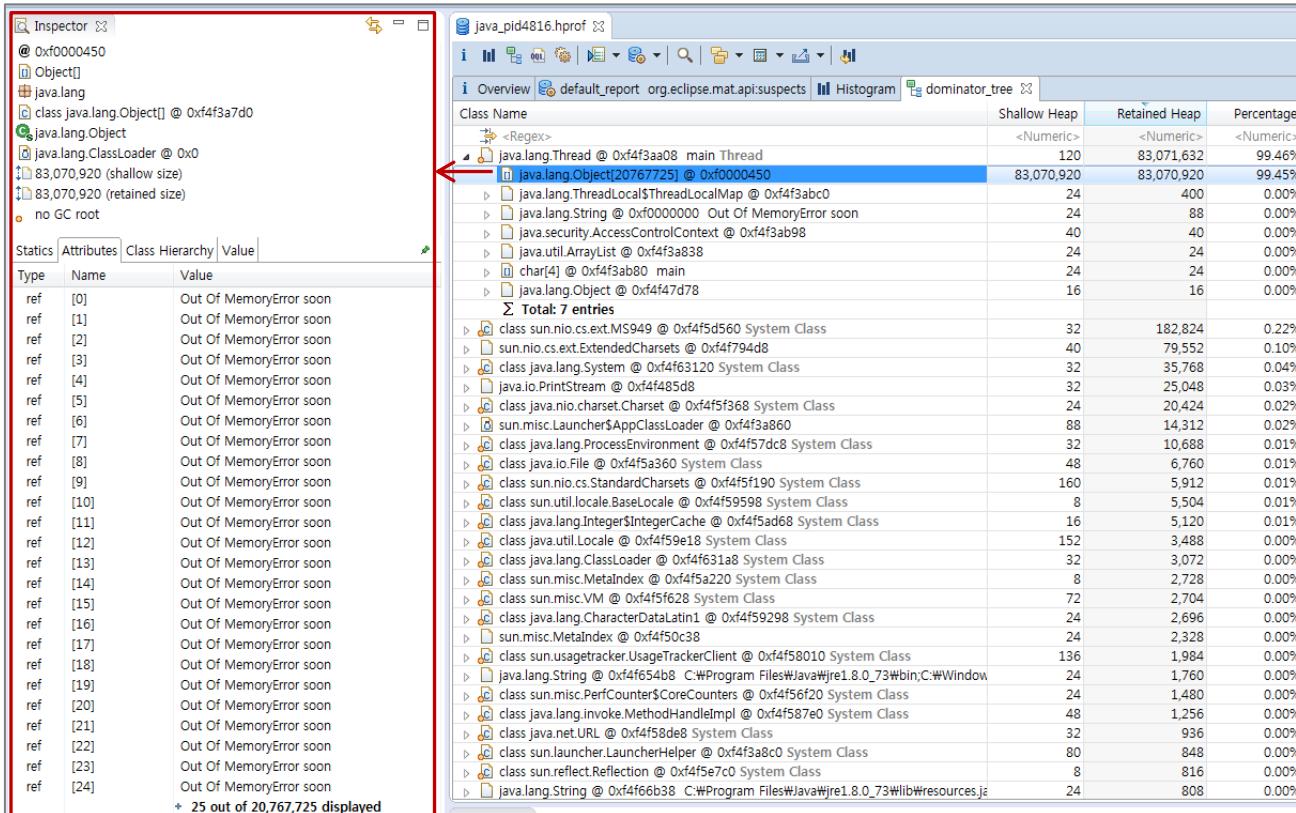
6.7 Dominator Tree [2/2]

- ✓ `java.lang.Thread` 항목이 99% 이상의 메모리를 점유하는 것을 확인할 수 있습니다.
- ✓ `java.lang.Thread`의 상세내용을 확인하면 구체적으로 어떤 Class가 메모리를 점유하는지 알 수 있습니다.
- ✓ 아래와 같은 경우 20767725만큼 배열 크기가 증가하는 시점에 Out Of Memory Error가 발생한 것으로 판단할 수 있습니다.



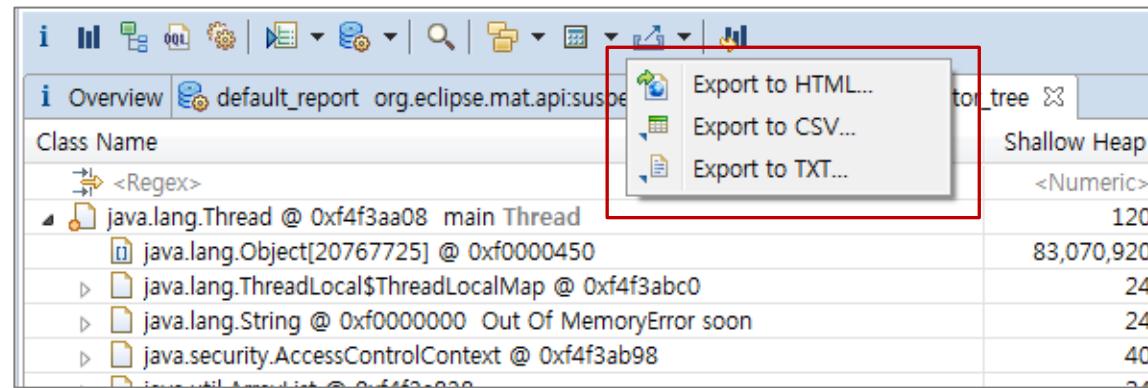
6.8 Inspector

- ✓ Inspector는 선택한 Class 및 Object에 대한 상세정보를 제공합니다.
- ✓ Inspector창에서 메모리주소, 패키지정보, 클래스정보, Attribute 정보 등을 보여줍니다.
- ✓ 아래는 java.lang.Object[20767725]의 상세내용을 조회한 Inspector 화면입니다. Attributes 항목에 Out Of Memory Error soon 문자열이 무수히 배열에 추가되어 있는 것을 확인할 수 있습니다.



6.9 보고서

- ✓ MAT는 응용프로그램의 메모리 상태에 대한 보고서를 작성하기 유용합니다.
- ✓ Histogram, DominatorTree 등의 정보를 여러가지 형식(HTML, CSV, TEXT)으로 결과를 내보낼 수 있습니다.
- ✓ 추출한 파일은 자주 사용하는 스프레드 시트 프로그램 또는 기타 도구 등으로 다룰 수 있습니다.



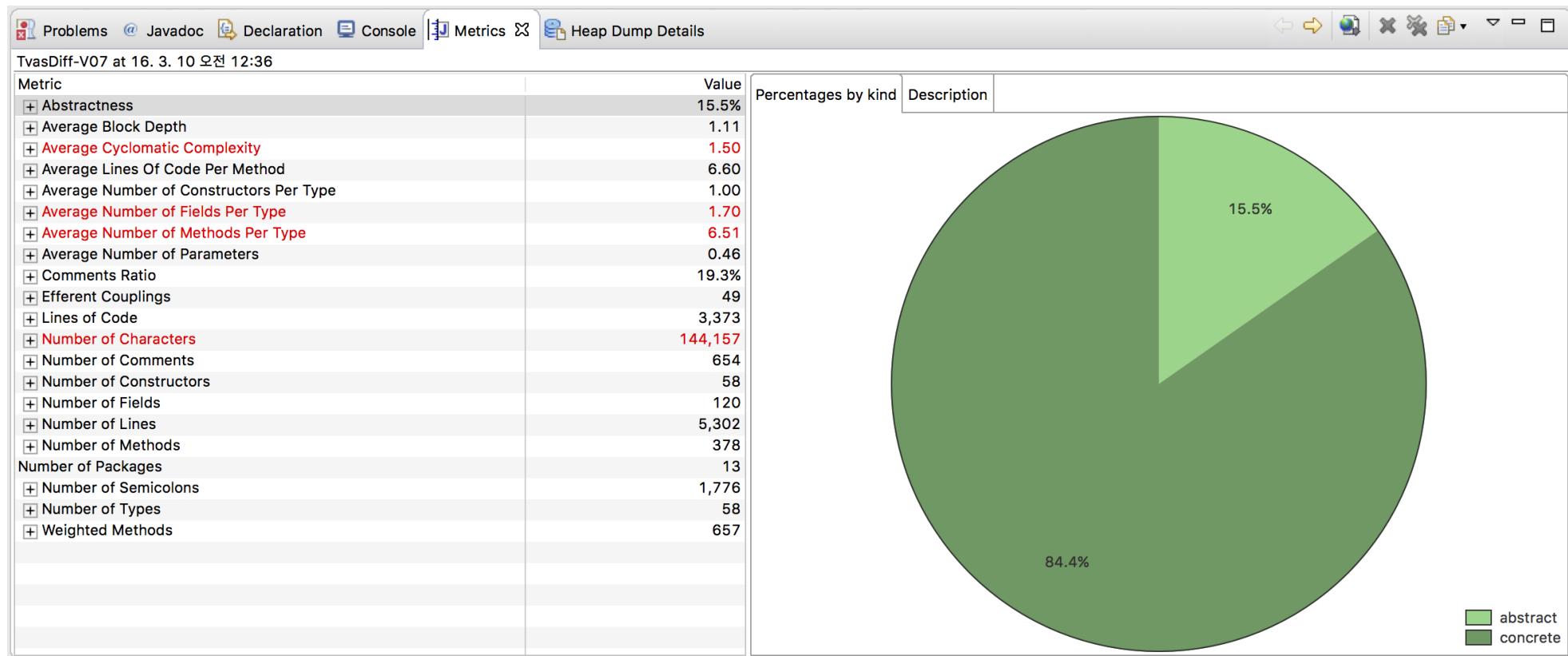


7. 도구 II - CodePro

-
- 7.1 자가 진단
 - 7.2 코드 중복
 - 7.3 블록 깊이
 - 7.4 복잡성
 - 7.5 메소드 평균 LoC
 - 7.6 평균 매개변수 개수
 - 7.7 의존관계
 - 7.8 LoC

7.1 자가 진단

- ✓ CodePro는 구글에서 개발한 오픈소스 정적분석도구입니다.
- ✓ Audit, Metrics, Dependency 분석 등의 기능을 제공합니다.
- ✓ 개발자는 CodePro와 같은 도구를 이용하여 자신이 개발한 코드를 점검하는 습관을 길러야 합니다.



7.2 코드 중복

- ✓ 서로 다른 메소드의 소스코드가 일부 몇 라인을 제외하고는 똑같습니다.
- ✓ 비슷한 로직 구현 시 기존 소스를 그대로 복사-붙여 넣기 후 일부 다른 부분만 변경한 결과입니다.
- ✓ 기존 로직을 그대로 복사한다는 것은 기존 로직이 가지고 있는 오류 가능성까지도 모두 복사한다는 의미입니다.
- ✓ 아래는 서로 다른 기능이지만 약 87%의 중복된 소스코드가 존재합니다.

The screenshot shows two identical Java code snippets from the `DlvcCalcInfoQryCond` class. Both snippets are enclosed in a dashed box, indicating they are the same code. The code is as follows:

```
DlvcCalcInfoQryCond.class;
String pYmStr = "";
String pYmStrMsg = "";

if(pDlvcCalcInfoQryCond.getCalcYm().length() > 5){
    pYmStr = pDlvcCalcInfoQryCond.getCalcYm().substring(0, 6);
    pYmStrMsg = "" + pYmStr.substring(2,4) + " " + pYmStr.substring(4,6) + "월 ";
} else{
    throw new DevPrcException(Message.getMessage("cmm.msg.011"
        , new String[] { "정산월" })); // (0)월을 확인해주세요
}

// 1. 배송비 정산마감상태 조회
pDlvcCalcInfoQryCond.setCalcYm(pYmStr);
String pDlvcCalcDedInStCd = StringUtil.NVL(dlvcCalcInfoEntity.getSupStfCalcDedInSt(pDlvcCalcInfoQryCo
    // 험력사상계정상마감상태코드 != "30"
    if(!DtrConstants.DTR_DLVC_CALC_DEDLN_ST_CD_30.equals(pDlvcCalcDedInStCd)) {
        //dtr.msg.022=(0)할 수 없습니다. 진행상태를 확인해 주세요
        throw new DevPrcException(Message.getMessage("dtr.msg.023"
            , new String[] { "반려" }));
    }
    DlvcCalcInfo pDlvcCalcInfo = new DlvcCalcInfo();
}
```

7.3 블록 깊이(Block Depth)

- ✓ 평균 0.7, 최대 16 Depth까지 존재합니다. (일반적으로 3 Depth 까지를 허용합니다.)
- ✓ Depth가 깊어질수록 가독성이 떨어지고, 소스코드 추적이 어려워집니다.
- ✓ 가장 흔한 Depth 증가의 원인은 곳곳에 있는 NULL 체크입니다. 작은 고민으로 충분히 낫출 수 있습니다.

```
public List<FreqGftTgtRegExtrctFileRecInfo> read0370Excel(File file,
    List<FreqGftTgtRegExtrctFileRecInfo> freqGftTgtRegExtrctFileRecList) {
    // int errCount = 0;
    POIFSFileSystem fs;
    FreqGftTgtRegExtrctFileRecInfo freqGftTgtRegExtrctFileRecInfo;
    int rows;
    try {
        fs = new POIFSFileSystem(new FileInputStream(file));

        HSSFWorkbook wb = new HSSFWorkbook(fs);
        int sheetNum = wb.getNumberOfSheets();

        for (int k = 0; k < sheetNum; k++) {
            HSSFSheet sheet = wb.getSheetAt(k);
            rows = sheet.getPhysicalNumberOfRows();

            for (int r = 1; r < rows; r++) {
                HSSFRow row = sheet.getRow(r);
                freqGftTgtRegExtrctFileRecInfo = new FreqGftTgtRegExtrctFileRecInfo();
                if (row != null) {
                    int cells = row.getLastCellNum();

                    try {
                        for (int inx = 0; inx < cells; inx++) {
                            HSSFCell cell = row.getCell(inx);

                            if (cell != null) {
                                String value = null;

                                switch (cell.getCellType()) {
                                    case HSSFCell.CELL_TYPE_FORMULA:
                                        value = cell.getCellFormula();
                                        break;
                                }
                            }
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

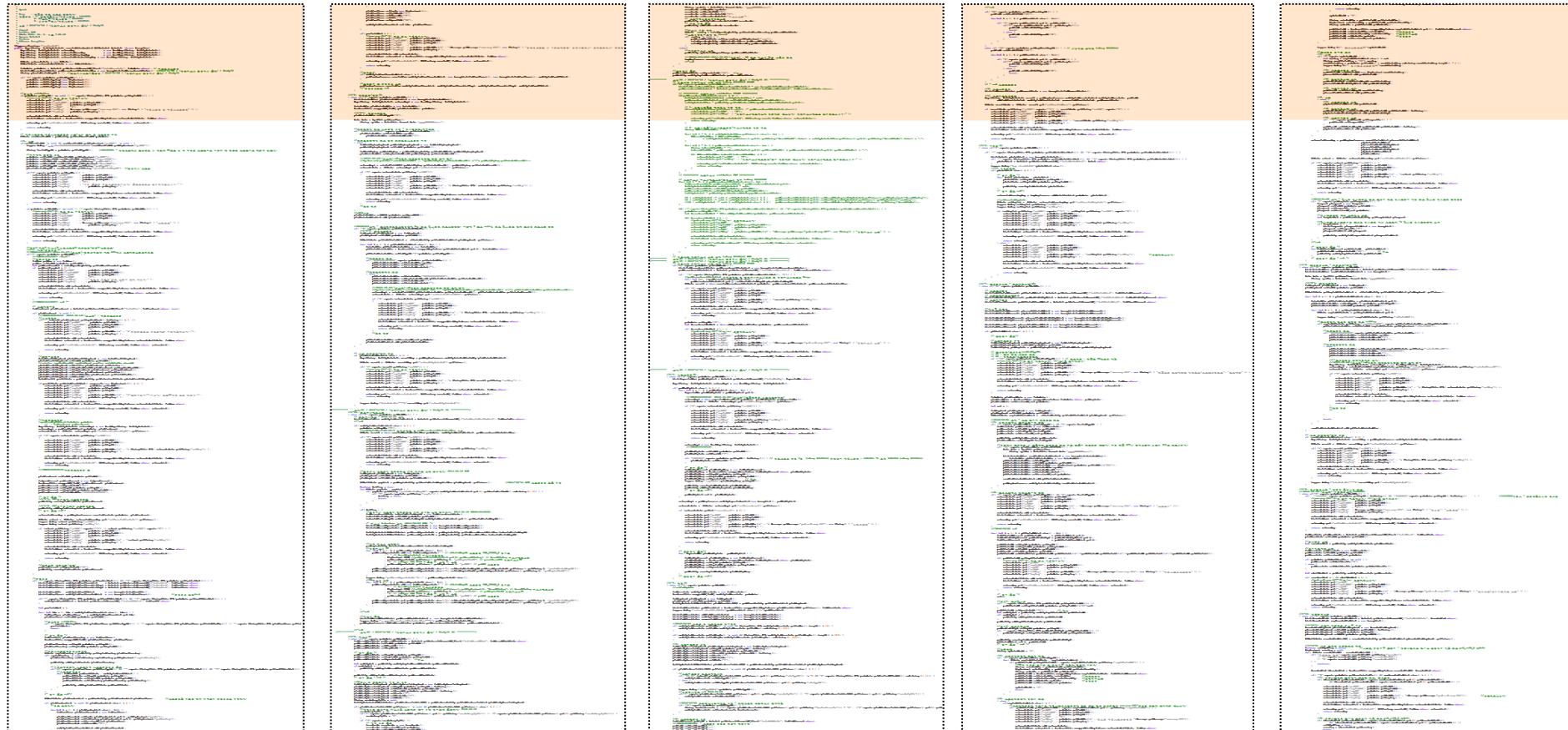
7.4 복잡성 (Cyclomatic Complexity)

- ✓ 조건문 분기에 따라 소스코드가 실행될 수 있는 경우의 수를 나타내냅니다.
- ✓ 중첩된 조건문과 조건문이 포함한 많은 AND, OR조건으로 인해 경우의 수가 점차적으로 증가하게 됩니다.
- ✓ 경우의 수가 증가할수록 로직 변경 시 각 경로를 모두 고려 해야 하는 어려움이 있습니다.
- ✓ 오류가 발생할 경우 수많은 조건을 모두 고려 해야 하기 때문에 원인을 추적하기 어렵습니다.

```
if("Y".equals(pickTypByClaimInfo.getDtctCd())){  
  
    // 클레임상태가 확정(30)이고 클레임구분이 파손(90)이면 반품완료처리(ec전용센터의경우)  
    if ("30".equals(pickTypByClaimInfo.getClaimPrgStCd())  
        && "1500".equals(pickTypByClaimInfo.getShipBrnCd())  
        && "90".equals(pickTypByClaimInfo.getClaimGbnCd()) ) {  
        // 클레임구분이 전체클레임(10), 손망설(50), 파손(90), 협력사입고완료(배송비제외)(41)이면서  
        // 반품완료(03)인 경우 택배사를 기타(ZE)로 변경  
        try {  
            pickTypByClaimInfo.setStCd("03");  
            sDlvsCoCd = getDlvsCoCdCheck(pickTypByClaimInfo);  
        } catch (DevEntException ex) {  
            throw new DevEntException(ex.toString().replace("dev.story.framework.exception.DevEntException: ", ""));  
        }  
        //logger.debug("savePickTypByClaimList b.5. 택배사코드 확인 ======> [" + sDlvsCoCd + "]");  
        pickTypByClaimInfo.setDlvsCoCd(sDlvsCoCd);  
  
        /* echo 상태 -> 반품완료일자,반품완료상태 UPDATE */  
        try {  
            handleRtpFsh(pickTypByClaimInfo);  
            //logger.debug("savePickTypByClaimList e.5. 반품처리완료 ======> [Y]");  
        } catch (DevEntException ex) {  
            throw new DevEntException(ex.toString().replace("dev.story.framework.exception.DevEntException: ", ""));  
        }  
    }  
}  
else{  
  
    // 클레임상태가 확정(30)이면 반품완료 처리  
    // 단, 출하지점코드가 택배수거(1500)이고 클레임구분이 부분클레임(20), 파손(90), CS비용(60)이면 제외  
    if ("30".equals(pickTypByClaimInfo.getClaimPrgStCd())  
        && !( "1500".equals(pickTypByClaimInfo.getShipBrnCd())  
            && ( "20".equals(pickTypByClaimInfo.getClaimGbnCd())  
                || "90".equals(pickTypByClaimInfo.getClaimGbnCd())  
                || "60".equals(pickTypByClaimInfo.getClaimGbnCd()) ) ) {
```

7.5 메소드 평균 LOC

- ✓ LOC는 평균 5.45, 최대 1855 라인 까지 존재합니다. 50 라인을 초과하면 한 눈에 확인하기 어렵습니다.
- ✓ 라인수가 많으면 그만큼 하는 일도 많다는 의미가 됩니다. 객체지향 원칙 중 단일 책임의 원칙에 위배됩니다.
- ✓ 한 메소드가 다양한 기능을 수행하기 때문에 메소드 이름만으로는 어떠한 기능을 수행하는지 알 수 없습니다.
- ✓ 메소드 Block이 길어질 수록 메소드, 조건문, 반복문의 시작과 끝, 변수 유효범위 등을 파악하기 어렵습니다.



7.6 평균 매개변수 개수

- ✓ 매개변수는 평균 0.7, 최대 22개까지 존재합니다. 아래와 같이 매개변수가 많아지면 쉽게 파악되지 않습니다.
- ✓ 매개변수가 많다는 것은 한 메소드가 많은 데이터를 제어한다는 의미입니다. 즉 메소드가 여러 가지 일을 한다는 의미입니다. 도메인 모델링을 통해서 알맞은 모델을 설계하고, 식별된 객체를 매개변수로 전달해야 합니다.
- ✓ 일반적으로 매개변수가 5개 이상이면 더 이상 매개변수를 추가하지 말고 객체화 해서 전달해야 합니다.

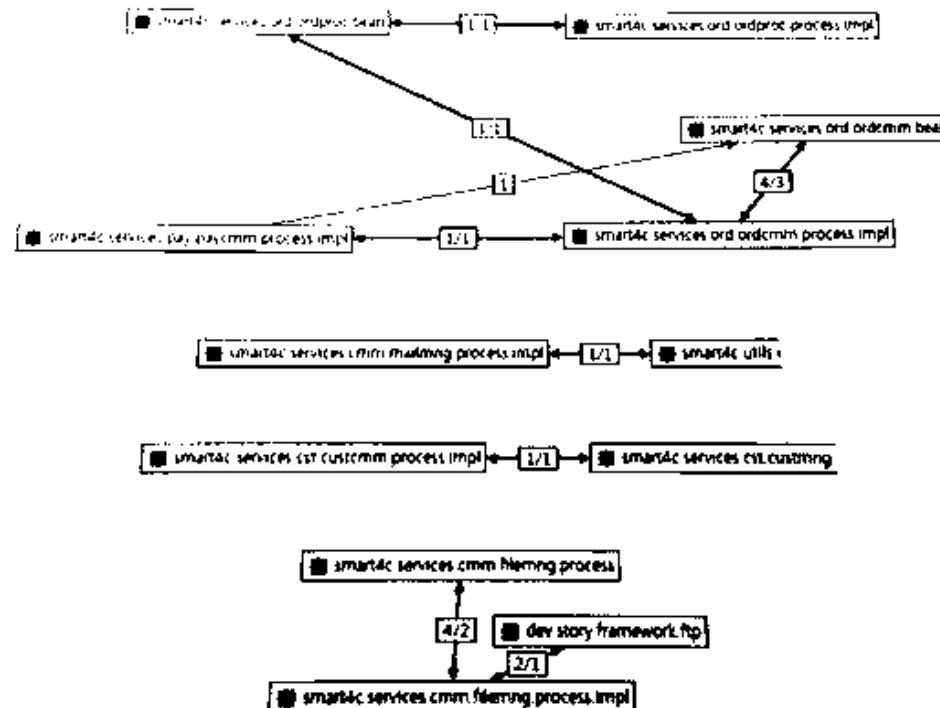
```
// 상품물류 확장 정보가 추가된 상품정보 저장 메소드 : sap 재구축 (2013/01/18 안승훈)
@SuppressWarnings("unchecked")
public Map<String, EntityDataSet> addPrdBsBaseInfoList(PrdmMain prdmMain, List<PrdAttrPrdMinsert> pattrPrdM,
    List<PrdStockInsert> prdAttrPrdm, PrdprcHinsert prdprcHinsert, List<PrdChanlInfo> prdChanlDinsert,
    List<PrdOrdPsbQtyInsert> prdOrdPsbQtyD, List<PrdChanlMappnDinsert> prdChanlMappnD,
    PrdPrdDinsert prdPrdD, List<PrdNmChgHinsert> prdNmChg, List<EcdSectPrdlstInfo> ecdSectPrdlstInfo,
    SafeCertPrd psafeCertPrd, List<PrdSchdDInfo> prdSchdDInfo, List<PrdNumvalDinfo> prdNumvalDinfo,
    List<PrdSpecVal> prdSpecInfo, List<PrdUdaDtl> prdUdaDtl, PrdNtcInfo prdNtcInfo,
    List<PrdMetaDInfo> prdMetaDInfo, List<PrdDesceGenrlDInfo> prdDesceGenrlDInfoList,
    List<PrdDescdHtmlDInfo> prdDescdHtmlDInfoList, ForgnTmPrdMng forgnTmPrdMng,
    List<BundlPrdCmposInfo> bundlPrdCmposInfoList, PrdDtrD prdDtrD) throws Exception;

// 해외상품 세트가 추가된 상품정보 저장 메소드 (2011/11/04 OSM)
@SuppressWarnings("unchecked")
public Map<String, EntityDataSet> addPrdBsBaseInfoList(PrdmMain prdmMain, List<PrdAttrPrdMinsert> pattrPrdM,
    List<PrdStockInsert> prdAttrPrdm, PrdprcHinsert prdprcHinsert, List<PrdChanlInfo> prdChanlDinsert,
    List<PrdOrdPsbQtyInsert> prdOrdPsbQtyD, List<PrdChanlMappnDinsert> prdChanlMappnD,
    PrdPrdDinsert prdPrdD, List<PrdNmChgHinsert> prdNmChg, List<EcdSectPrdlstInfo> ecdSectPrdlstInfo,
    SafeCertPrd psafeCertPrd, List<PrdSchdDInfo> prdSchdDInfo, List<PrdNumvalDinfo> prdNumvalDinfo,
    List<PrdSpecVal> prdSpecInfo, List<PrdUdaDtl> prdUdaDtl, PrdNtcInfo prdNtcInfo,
    List<PrdMetaDInfo> prdMetaDInfo, List<PrdDesceGenrlDInfo> prdDesceGenrlDInfoList,
    List<PrdDescdHtmlDInfo> prdDescdHtmlDInfoList, ForgnTmPrdMng forgnTmPrdMng,
    List<BundlPrdCmposInfo> bundlPrdCmposInfoList) throws Exception;

// 해외상품 추가된 상품정보 저장 메소드 (2011/05/02 OSM)
@SuppressWarnings("unchecked")
public Map<String, EntityDataSet> addPrdBsBaseInfoList(PrdmMain prdmMain, List<PrdAttrPrdMinsert> pattrPrdM,
    List<PrdStockInsert> prdAttrPrdm, PrdprcHinsert prdprcHinsert, List<PrdChanlInfo> prdChanlDinsert,
    List<PrdOrdPsbQtyInsert> prdOrdPsbQtyD, List<PrdChanlMappnDinsert> prdChanlMappnD,
    PrdPrdDinsert prdPrdD, List<PrdNmChgHinsert> prdNmChg, List<EcdSectPrdlstInfo> ecdSectPrdlstInfo,
    SafeCertPrd psafeCertPrd, List<PrdSchdDInfo> prdSchdDInfo, List<PrdNumvalDinfo> prdNumvalDinfo,
    List<PrdSpecVal> prdSpecInfo, List<PrdUdaDtl> prdUdaDtl, PrdNtcInfo prdNtcInfo,
    List<PrdMetaDInfo> prdMetaDInfo, List<PrdDesceGenrlDInfo> prdDesceGenrlDInfoList,
    List<PrdDescdHtmlDInfo> prdDescdHtmlDInfoList, ForgnTmPrdMng forgnTmPrdMng,
    List<BundlPrdCmposInfo> bundlPrdCmposInfoList) throws Exception;
```

7.7 의존관계 (dependency)

- ✓ 의존관계 분석결과 상호참조 (cycle)가 존재합니다. 이 경우 컴파일이 되지 않거나, 무한 LOOP가 발생할 수도 있습니다. 레이어를 구성해서 같은 레이어와 상위 레이어는 참조하지 않도록 규칙을 정해야 합니다.
- ✓ util패키지의 경우 공통 영역이지만 이 패키지에서 역으로 업무영역을 참조하고 있습니다.



7.8 LoC

- ✓ 라인 수(Number of Lines): 공백, 주석을 모두 포함한 라인 수를 의미합니다.
- ✓ 코드 라인 수(Line of Code): 공백과 주석을 제외한 순수한 소스코드 라인 수를 의미합니다.
- ✓ 논리적인 LoC (*) 또는 실행문 LoC: 블럭, 조건문 등을 제외한 실행문(;을 포함한 문) 라인 수를 의미합니다.

FIDS_ALL at 9/11/14 9:51 PM	
Metric	Value
+ Abstractness	35.2%
+ Average Block Depth	0.68
+ Average Cyclomatic Complexity	1.11
+ Average Lines Of Code Per Method	4.28
+ Average Number of Constructors Per Type	0.26
+ Average Number of Fields Per Type	3.57
+ Average Number of Methods Per Type	11.00
+ Average Number of Parameters	0.74
+ Comments Ratio	12.2%
+ Efferent Couplings	946
+ Lines of Code	87,750
코드 라인 수	
+ Number of Characters	4,524,957
+ Number of Comments	10,739
+ Number of Constructors	346
+ Number of Fields	6,294
+ Number of Lines	170,068
라인 수	
+ Number of Methods	14,619
Number of Packages	223
+ Number of Semicolons	48,834
실행문 수	
+ Number of Types	1,328
+ Weighted Methods	16,704
* LoC : Line of Code	



8. 도구 III - SonarQube

이충현 상무(chlee@nextree.co.kr)

8.1 SonarQube 소개

8.2 분석

8.3 CI 연계

8.1 SonarQube 소개 (1/10)

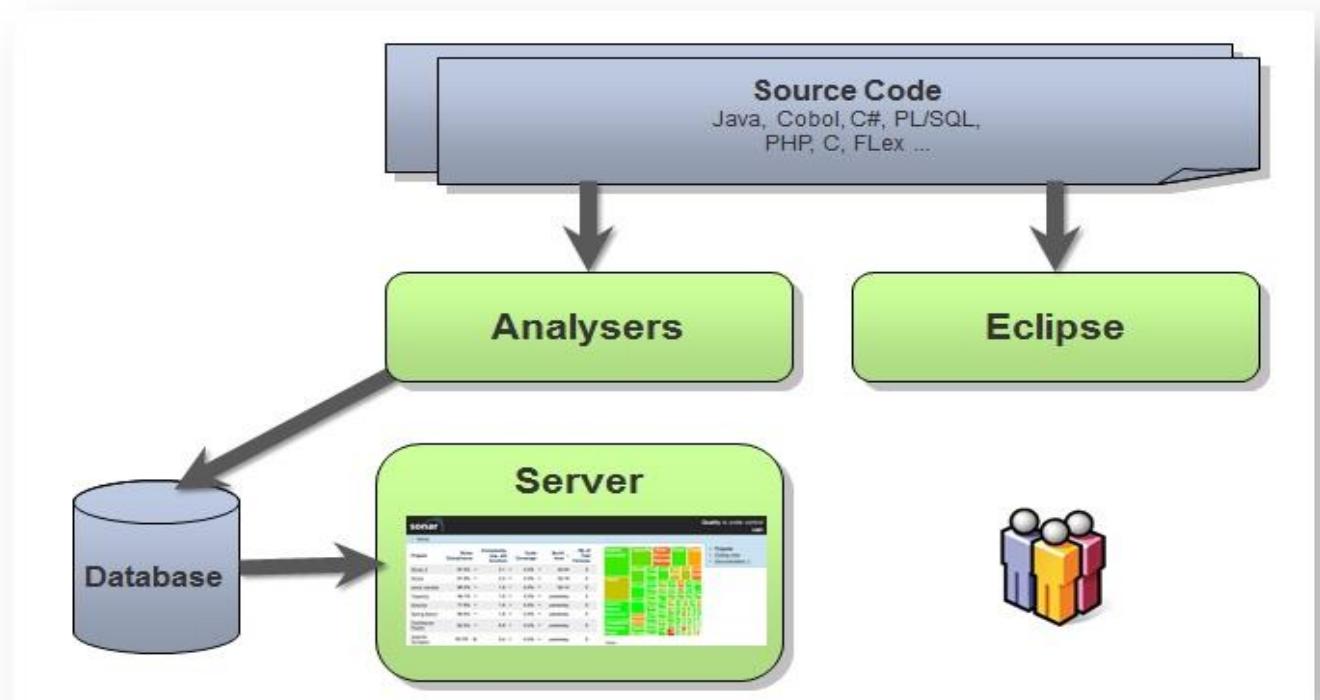
- ✓ www.sonarqube.org
- www.sonarqube.org/downloads

The screenshot shows the official SonarQube website at www.sonarqube.org. The page features a dark header with the SonarQube logo and navigation links: Download, Features, Get Support, Get Involved, Development, Roadmap, Resources, Blog, and Company. Below the header, a large banner with the tagline "Put your technical debt under control" and the subtext "Productivity is falling? Confess your source code to clean it up!" is displayed. To the right of the banner, there's a screenshot of the SonarQube dashboard showing various metrics like Lines of code, Complexity, and Violations. The main content area includes sections for "All in one", "In 3 clicks", "Extend with plugins", "Languages covered", and "Quality is central". On the right side, there are sidebar sections for "Get started" (with steps 1-4) and "SonarQube™ in action" (with a screenshot of the Nemo instance). At the bottom, it says "Powered by SonarSource™" and "SonarSource proposes commercial extensions to cover additional languages and manage portfolios of projects along with Professional".

8.1 SonarQube 소개 (2/10)

✓ 구성 요소

- DB (설정, 분석 대상 프로젝트의 스냅샷 저장)
- Web Server (사용자 화면)
- Analyzer (여러 개로 분석 가능)



8.1 SonarQube 소개 (3/10)

✓ 지원 사양

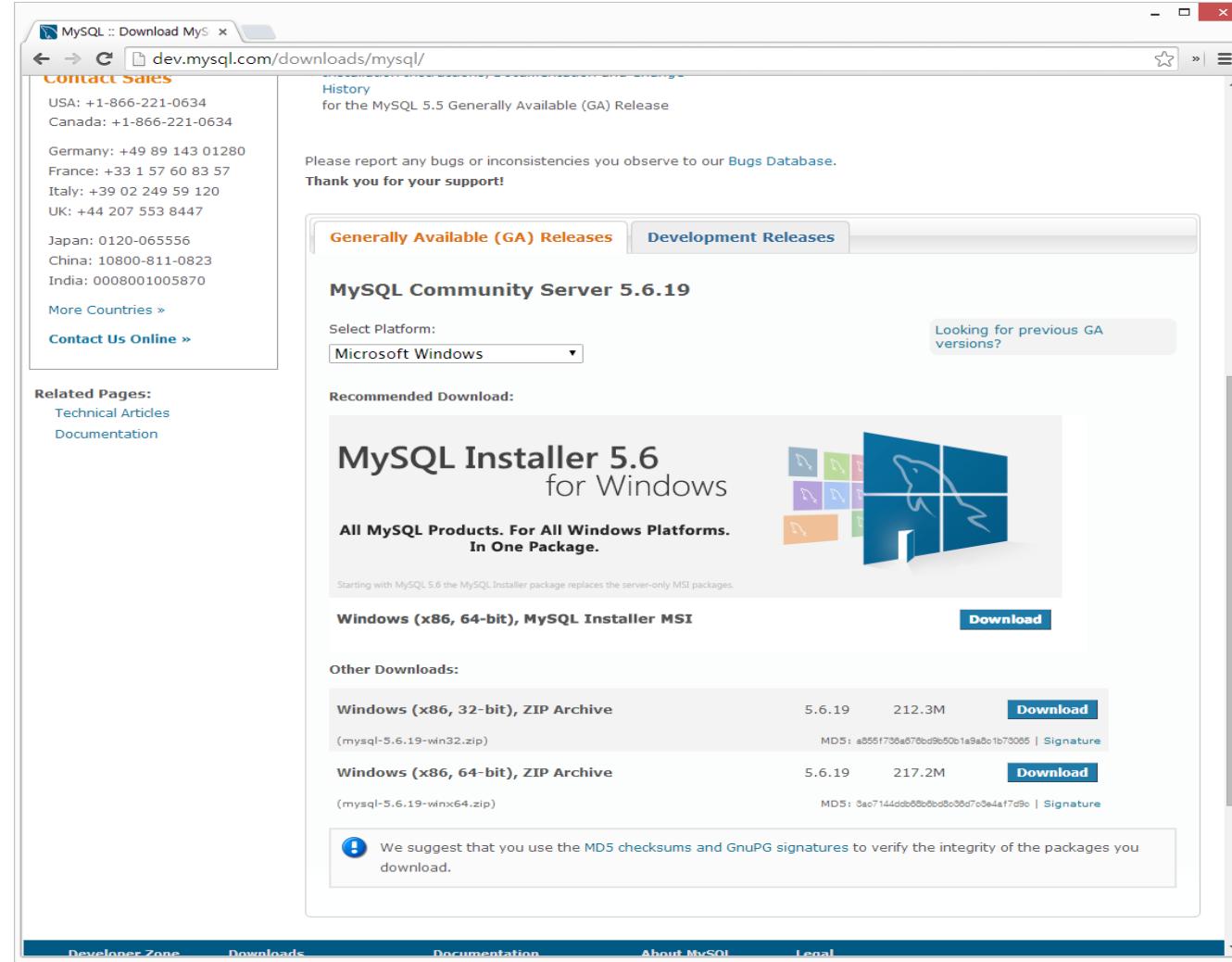
Java	
Oracle JRE	✓ 6 ✓ 7 (faster analysis than with version 6) ✗ 8
IBM JRE	✗
GCJ	✗
Oracle JRockit	✗
Database	
The charset of the database has to be set to "UTF-8" and the language to "English".	
Microsoft SQL Server	✓ 2005 with bundled jTDS driver. Microsoft drivers are not supported . Express Edition is supported. ✓ 2008 with bundled jTDS driver. Microsoft drivers are not supported . Express Edition is supported. ⚠ Collation must be case-sensitive (CS) and accent-sensitive (AS)
MySQL	✓ 5.0, 5.1 and 5.5 with the driver packaged within SonarQube. ✓ 5.6 is supported since SonarQube 3.5
Oracle	✓ 10G with Oracle 11.2.x drivers ✓ 11G with Oracle 11.2.x drivers ✓ XE Editions are supported ⚠ The driver ojdbc14.jar is not supported ⚠ Only the thin mode is supported, not OCI
PostgreSQL	✓ 8.x ✓ 9.x
Web Browser	
To get the full experience SonarQube has to offer, you should enable Javascript in your browser.	
Microsoft Internet Explorer	✗ IE 8 ✓ IE 9 ✓ IE 10 ⚠ IE 11 Not tested (https://jira.codehaus.org/browse/SONAR-5255)
Mozilla Firefox	✓ Supported
Google Chrome	✓ Supported
Opera	⚠ Not tested
Safari	✓ Supported

<http://docs.sonarqube.org/display/SONAR/Requirements>

8.1 SonarQube 소개 (4/10)

✓ MySQL

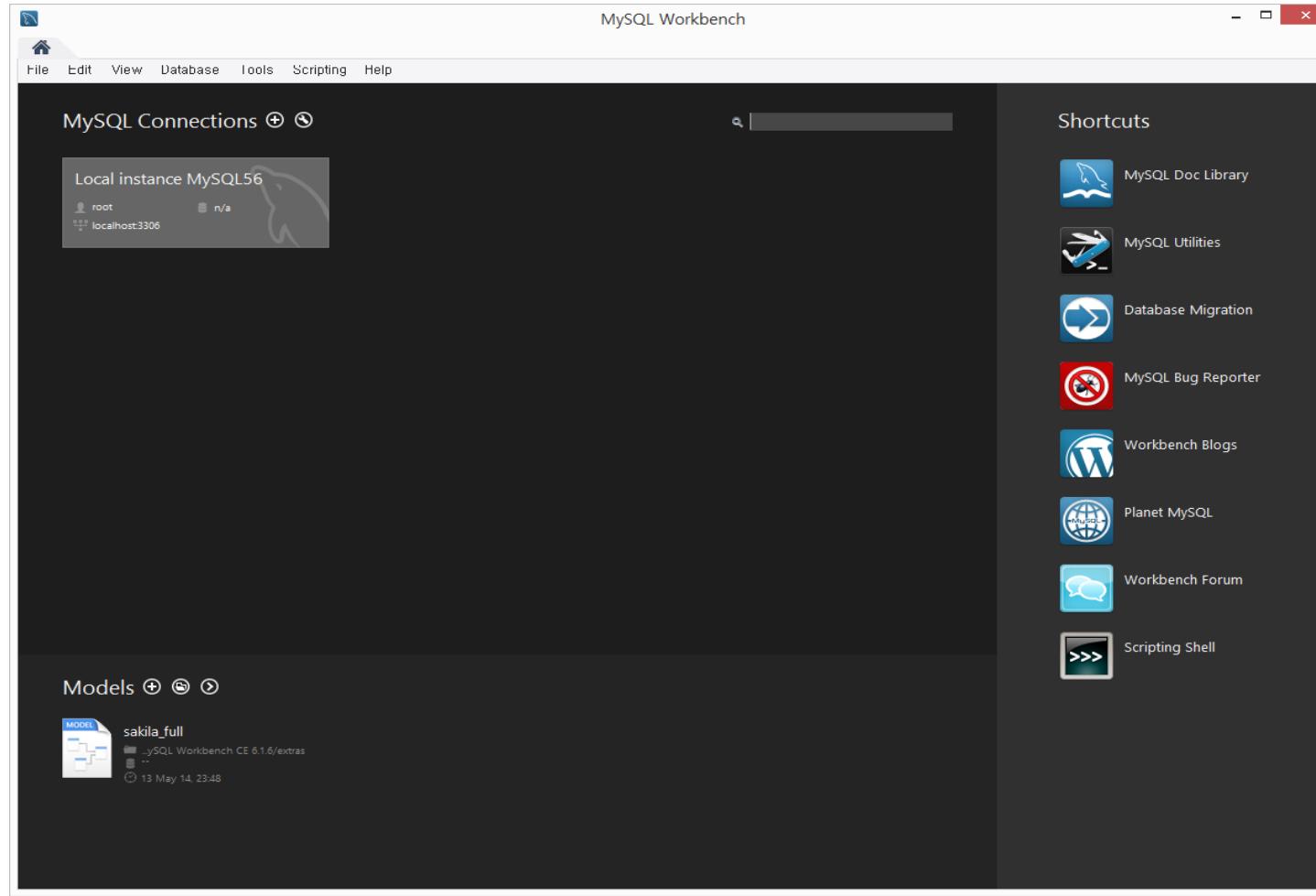
- dev.mysql.com/downloads



8.1 SonarQube 소개 (5/10)

✓ MySQL

- %MySQL_HOME%\MySQL Workbench CE 6.1.6\MySQLWorkbench.exe

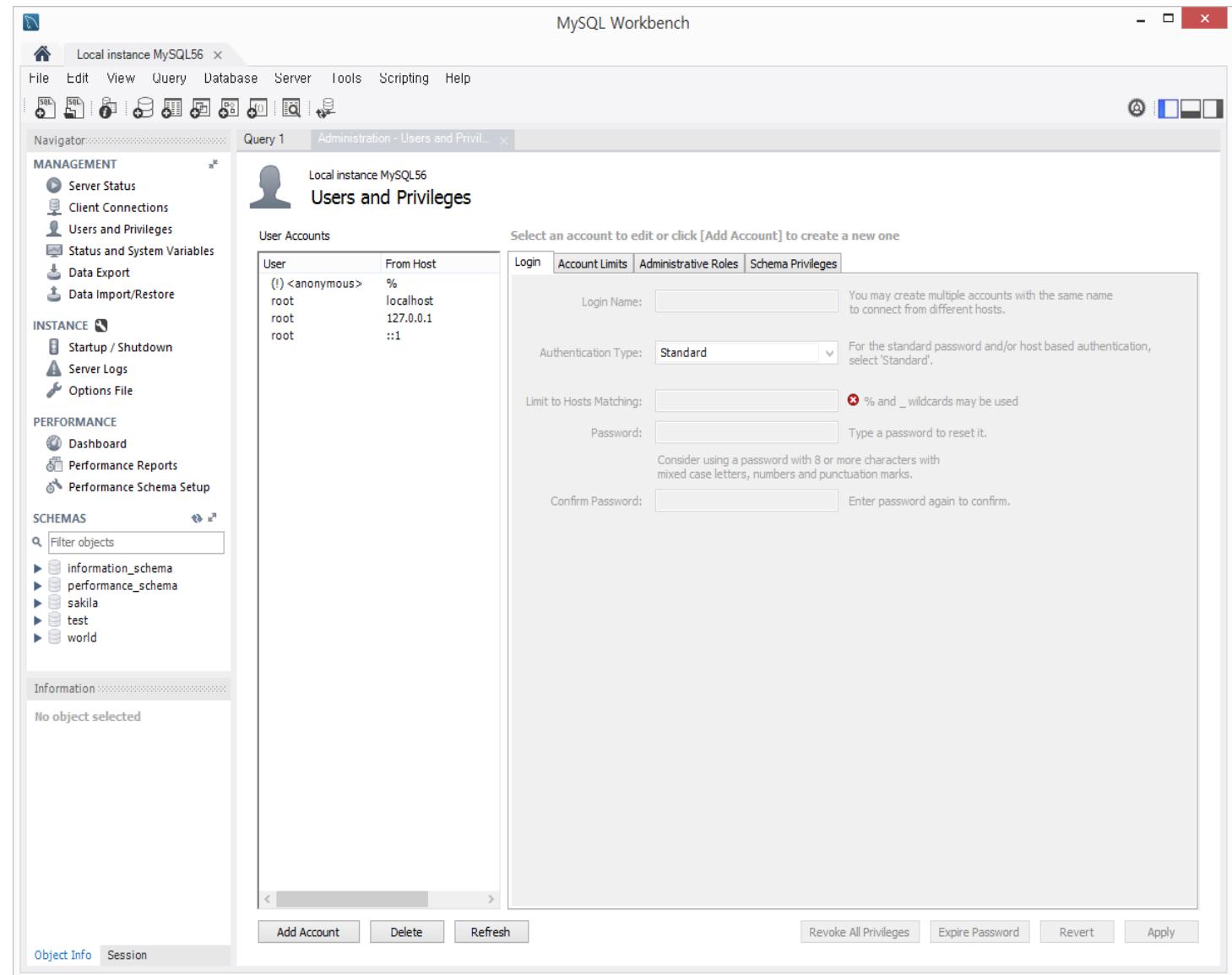


8.1 SonarQube 소개 (6/10)

✓ MySQL

- 사용자/스키마 추가

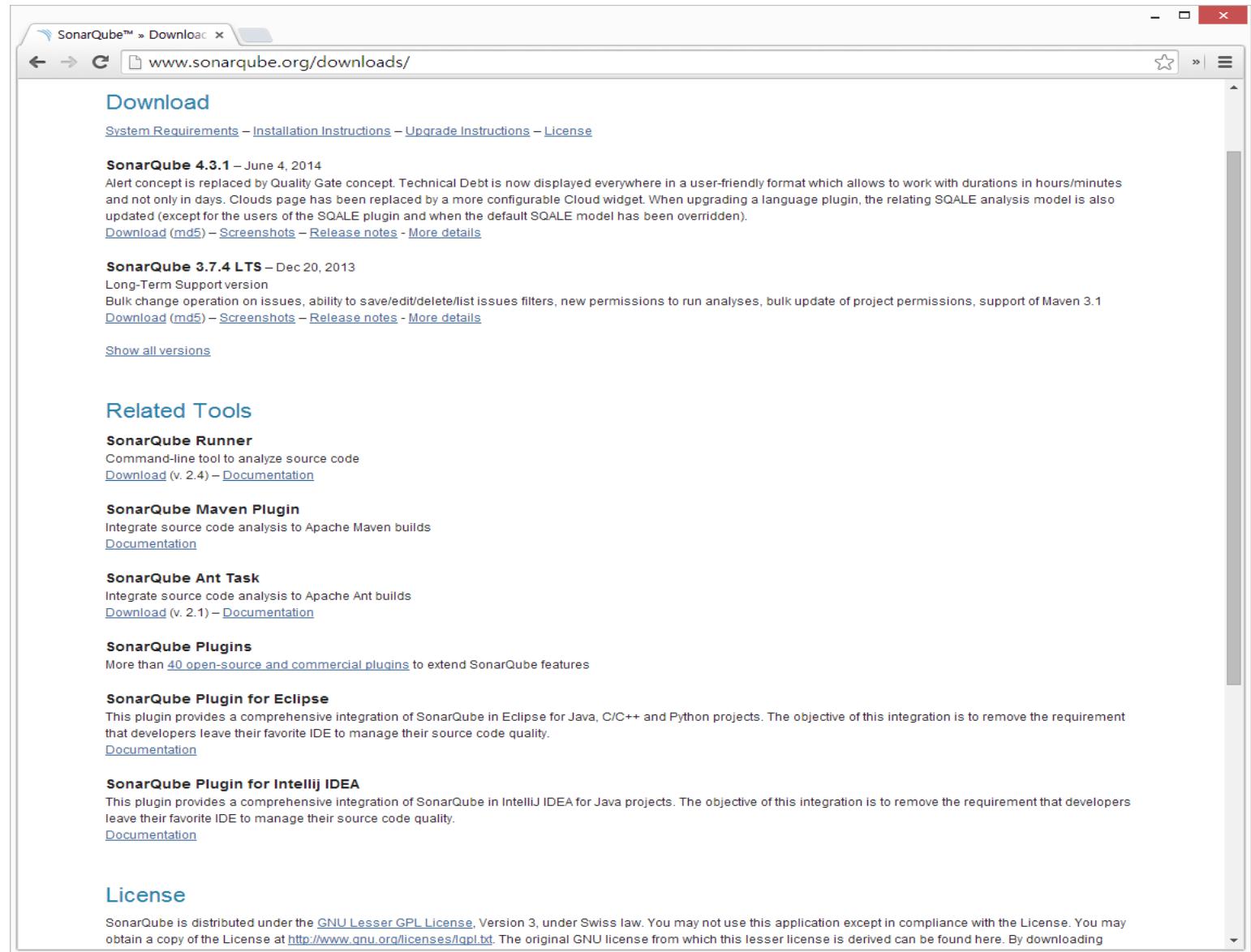
- sonar / sonar_schema



8.1 SonarQube 소개 (7/10)

✓ 다운로드

- SonarQube
- SonarQube Runner

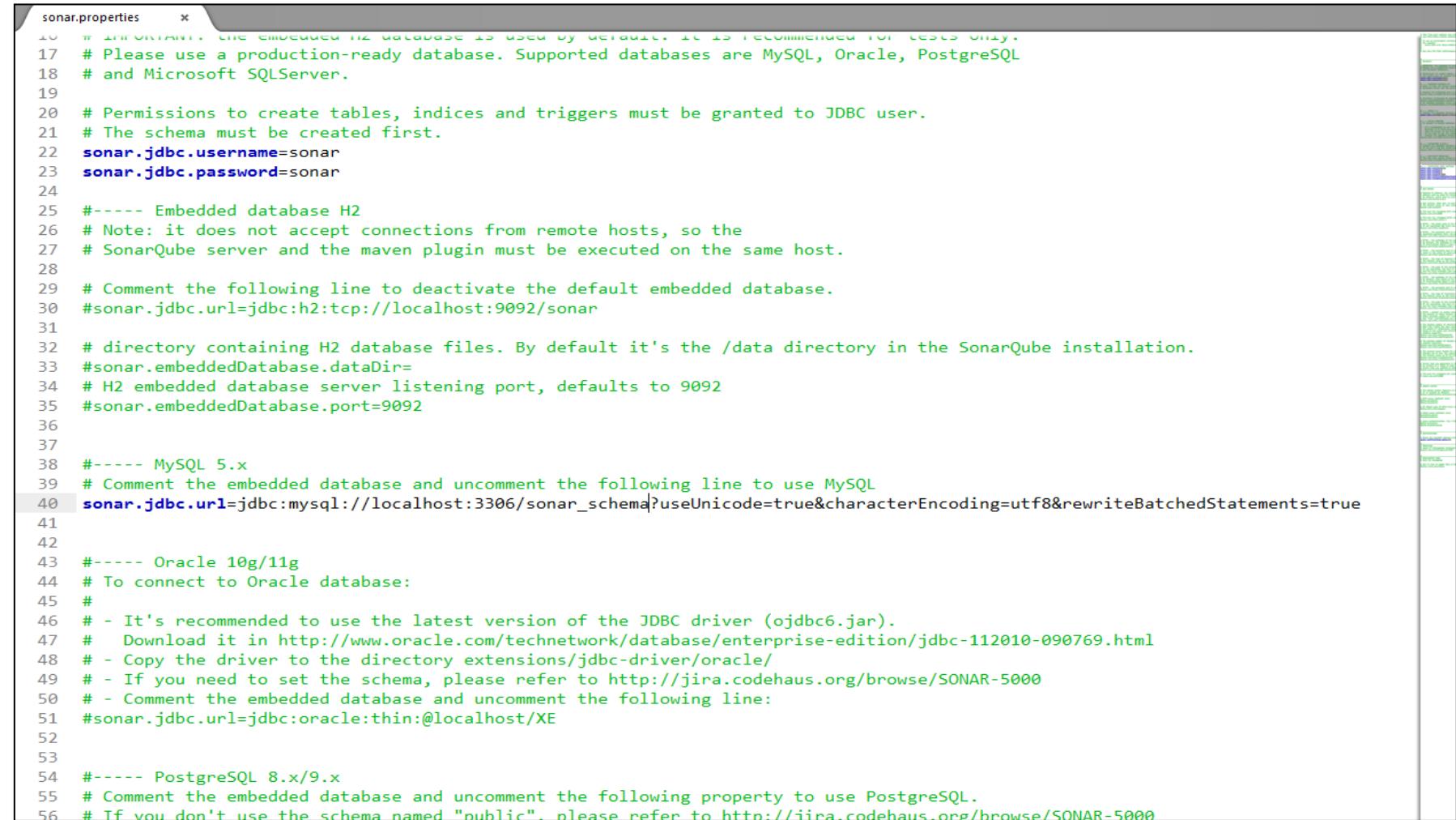


The screenshot shows the 'Download' section of the SonarQube website. At the top, there are links for 'System Requirements', 'Installation Instructions', 'Upgrade Instructions', and 'License'. Below this, the 'SonarQube 4.3.1 – June 4, 2014' section is displayed, noting changes like the replacement of the Alert concept with Quality Gate and the introduction of Cloud widgets. It includes download links for md5, screenshots, release notes, and more details. The 'SonarQube 3.7.4 LTS – Dec 20, 2013' section follows, detailing its status as a Long-Term Support version and listing bulk change operations, new issue filters, and Maven 3.1 support. A link to 'Show all versions' is provided. The 'Related Tools' section lists several integrations: 'SonarQube Runner' (command-line tool), 'SonarQube Maven Plugin' (for Apache Maven), 'SonarQube Ant Task' (for Apache Ant), 'SonarQube Plugins' (over 40 open-source and commercial plugins), 'SonarQube Plugin for Eclipse' (for Java, C/C++, and Python projects), and 'SonarQube Plugin for IntelliJ IDEA' (for Java projects). Each tool has a 'Download' and 'Documentation' link. The 'License' section at the bottom states that SonarQube is distributed under the GNU Lesser GPL License, Version 3, and provides a link to the full license text.

8.1 SonarQube 소개 (8/10)

✓ 설정

- <SonarQube 설치 디렉토리>\conf\sonar.properties



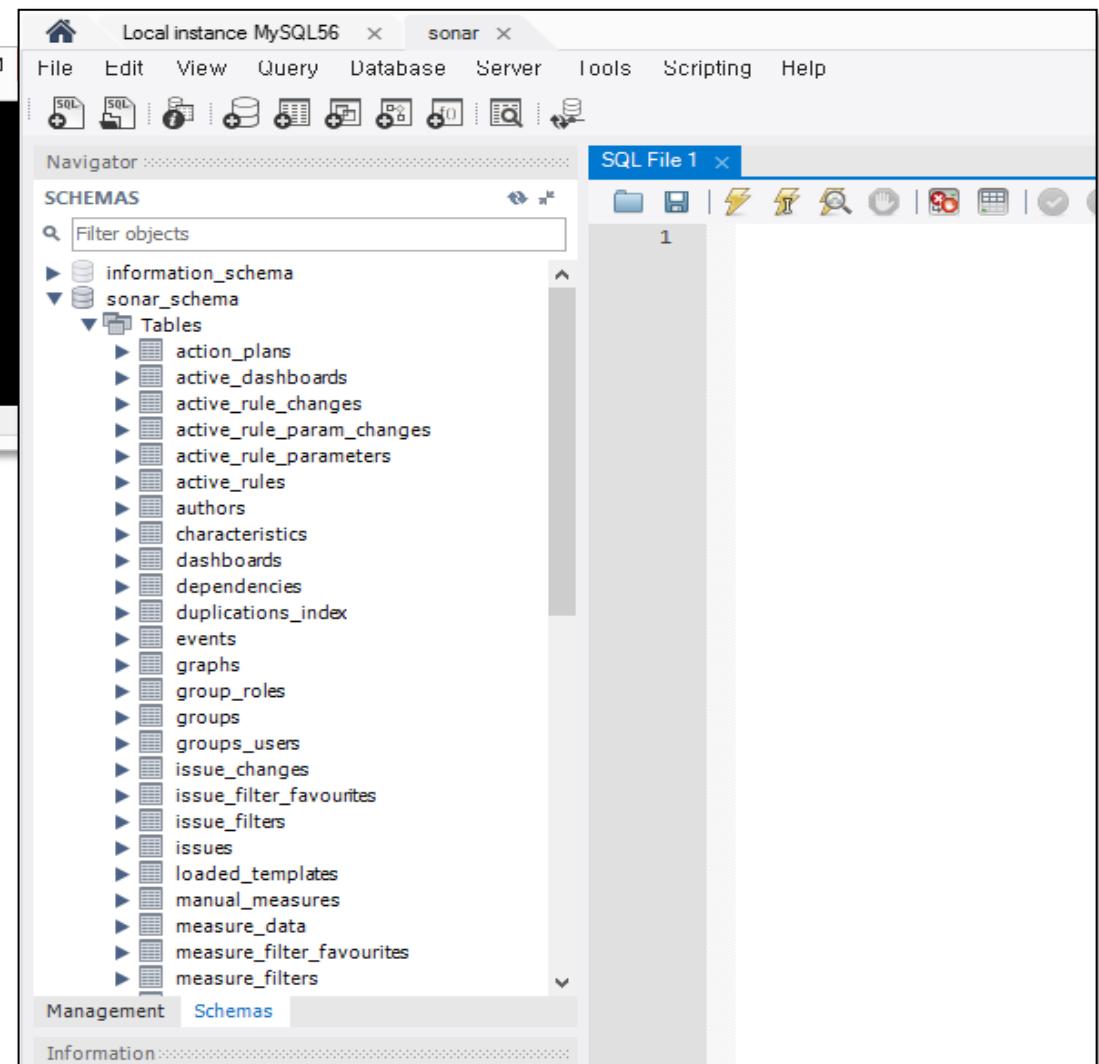
```
sonar.properties *  
10 # INFORMATION: the embedded H2 database is used by default. It is recommended for tests only.  
11 # Please use a production-ready database. Supported databases are MySQL, Oracle, PostgreSQL  
12 # and Microsoft SQLServer.  
13  
14 # Permissions to create tables, indices and triggers must be granted to JDBC user.  
15 # The schema must be created first.  
16 sonar.jdbc.username=sonar  
17 sonar.jdbc.password=sonar  
18  
19 #----- Embedded database H2  
20 # Note: it does not accept connections from remote hosts, so the  
21 # SonarQube server and the maven plugin must be executed on the same host.  
22  
23 # Comment the following line to deactivate the default embedded database.  
24 #sonar.jdbc.url=jdbc:h2:tcp://localhost:9092/sonar  
25  
26 # directory containing H2 database files. By default it's the /data directory in the SonarQube installation.  
27 #sonar.embeddedDatabase.dataDir=  
28 # H2 embedded database server listening port, defaults to 9092  
29 #sonar.embeddedDatabase.port=9092  
30  
31  
32 #----- MySQL 5.x  
33 # Comment the embedded database and uncomment the following line to use MySQL  
34 sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar_schema?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true  
35  
36  
37 #----- Oracle 10g/11g  
38 # To connect to Oracle database:  
39 #  
40 # - It's recommended to use the latest version of the JDBC driver (ojdbc6.jar).  
41 # Download it in http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html  
42 # - Copy the driver to the directory extensions/jdbc-driver/oracle/  
43 # - If you need to set the schema, please refer to http://jira.codehaus.org/browse/SONAR-5000  
44 # - Comment the embedded database and uncomment the following line:  
45 #sonar.jdbc.url=jdbc:oracle:thin:@localhost/XE  
46  
47  
48  
49  
50  
51  
52  
53  
54 #----- PostgreSQL 8.x/9.x  
55 # Comment the embedded database and uncomment the following property to use PostgreSQL.  
56 # If you don't use the schema named "public", please refer to http://jira.codehaus.org/browse/SONAR-5000
```

8.1 SonarQube 소개 (9/10)

✓ 기동

- <SonarQube 설치 디렉토리>\bin\<OS>\

```
c:\#Tools#\sonarqube-4.3.1\bin\windows-x86-32>startsonar
wrapper | --> Wrapper Started as Console
wrapper | Launching a JVM...
jvm 1 | Wrapper (Version 3.2.3) http://wrapper.tanukisoftware.org
jvm 1 | Copyright 1999-2006 Tanuki Software, Inc. All Rights Reserved.
jvm 1 | 2014.06.20 02:42:53 INFO Web server is started
```



8.1 SonarQube 소개 (10/10)

✓ 브라우저

- <http://localhost:9000>
 - admin/admin

Welcome to SonarQube Dashboard

Since you are able to read this, it means that you have successfully started your SonarQube server. Well done!

If you have not removed this text, it also means that you have not yet played much with SonarQube. So here are a few pointers for your next step:

- » Do you now want to [run analysis](#) on a project?
- » Maybe start [customizing dashboards](#)?
- » Or simply browse the [complete documentation](#)?
- » If you have a question or an issue, please visit the [Get Support](#) page.

QG	Name	Version	LOCs	Technical Debt	Last Analysis
	No data				

Projects

Size: Lines of code Color: Coverage 0.0% - 100.0%

No data

SonarQube™ technology is powered by [SonarSource SA](#)
Version 4.3.1 - [Community](#) - [Documentation](#) - [Get Support](#) - [Plugins](#)

8.2 분석 [1/7]

✓ 분석 대상

- Maven 3
 - <http://www.apache.org/dist/maven/maven-3/3.2.1/source/>



8.2 분석 [2/7]

✓ SonarQube Runner

- global setting : sonar-runner.properties

```
sonar-runner.properties x
1 #Configure here general information about the environment, such as SonarQube DB details for example
2 #No information about specific project should appear here
3
4 ##### Default SonarQube server
5 sonar.host.url=http://localhost:9000
6
7 ##### PostgreSQL
8 #sonar.jdbc.url=jdbc:postgresql://localhost/sonar
9
10 ##### MySQL
11 sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar_schema?useUnicode=true&characterEncoding=utf8
12
13 ##### Oracle
14 #sonar.jdbc.url=jdbc:oracle:thin:@localhost/XE
15
16 ##### Microsoft SQLServer
17 #sonar.jdbc.url=jdbc:jtds:sqlserver://localhost/sonar;SelectMethod=Cursor
18
19 ##### Global database settings
20 sonar.jdbc.username=sonar
21 sonar.jdbc.password=sonar
22
23 ##### Default source code encoding
24 sonar.sourceEncoding=UTF-8
25
26 ##### Security (when 'sonar.forceAuthentication' is set to 'true')
27 #sonar.login=admin
28 #sonar.password=admin
```

8.2 분석 [3/7]

✓ SonarQube Runner

- 분석 대상 폴더에 sonar-project.properties 파일 생성

```
sonar-project.properties *
1 # Required metadata
2 sonar.projectKey=apache:maven
3 sonar.projectName=Maven 3
4 sonar.projectVersion=3.2.1
5
6 # Path to the parent source code directory.
7 # Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
8 # Since SonarQube 4.2, this property is optional. If not set, SonarQube starts looking for source code
9 # from the directory containing the sonar-project.properties file.
10 sonar.sources=src
11
12 # Encoding of the source code
13 sonar.sourceEncoding=UTF-8
```

8.2 분석 [4/7]

✓ SonarQube Runner

- 분석 대상 프로젝트 내에서 sonar-runner 실행,
- Sonar 웹에서 확인

The screenshot displays two windows illustrating the SonarQube analysis process.

Top Window (cmd.exe):

```
c:\#analysis_src\apache-maven-3.2.1>c:\#Tools\sonar-runner-2.4\bin\sonar-runner
c:\#Tools\sonar-runner-2.4\bin\...
SonarQube Runner 2.4
Java 1.7.0_45 Oracle Corporation (32-bit)
Windows 8 6.2 x86
INFO: Runner configuration file: c:\#Tools\sonar-runner-2.4\bin\..\conf\sonar-runner.properties
INFO: Project configuration file: c:\#analysis_src\apache-maven-3.2.1\sonar-project.properties
INFO: Default locale: "ko_KR", source code encoding: "UTF-8"
INFO: Work directory: c:\#analysis_src\apache-maven-3.2.1\...
INFO: SonarQube Server 4.3.1
03:08:30.359 INFO - Load batch settings
03:08:30.471 INFO - User cache: C:\Users\출현\sonar\cache
03:08:30.475 INFO - Install plugins
```

Bottom Window (SonarQube - Maven 3):

localhost:9000/dashboard/index/1

Dashboard Projects Measures Issues Quality Profiles Quality Gates Log in Search

Maven 3 >

Version 3.2.1 - Jun 20 2014 03:08

Events All Jun 20 2014 Version 3.2.1

Key: apache:maven

Issues 0 Blocker 0 Critical 0 Major 0 Minor 0 Info 0

Technical Debt 0

Logs (Bottom):

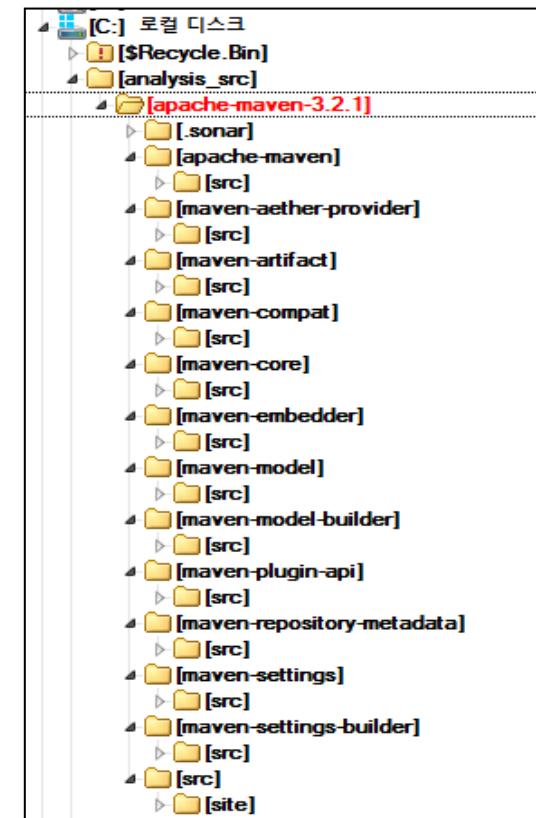
```
Jun 20 2014 03:08:30.359 INFO - Load batch settings
Jun 20 2014 03:08:30.471 INFO - User cache: C:\Users\출현\sonar\cache
Jun 20 2014 03:08:30.475 INFO - Install plugins
Jun 20 2014 03:08:34.902 INFO - Keep one snapshot per month between 2009-06-26 and 2013-06-21
Jun 20 2014 03:08:34.903 INFO - Delete data prior to: 2009-06-26
Jun 20 2014 03:08:34.907 INFO - Clean Maven 3 [id=1]
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
Total time: 5.186s
Final Memory: 11M/109M
INFO: -----
```

8.2 분석 [5/7]

✓ Multi-project

- 분석 대상 프로젝트 내에서 sonar-runner 실행

```
sonar-project.properties x
1 # Required metadata
2 sonar.projectKey=apache:maven
3 sonar.projectName=Maven 3
4 sonar.projectVersion=3.2.1
5
6 # Path to the parent source code directory.
7 # Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
8 # Since SonarQube 4.2, this property is optional. If not set, SonarQube starts looking for source code
9 # from the directory containing the sonar-project.properties file.
10 sonar.sources=src/main/java
11
12 sonar.tests=src/test/java
13
14 sonar.binaries=bin
15
16 sonar.libraries=bin/*.jar
17
18 # Encoding of the source code
19 sonar.sourceEncoding=UTF-8
20
21 sonar.modules=maven-aether-provider,maven-artifact,maven-compat,maven-core,maven-embedder,maven-
   model,maven-model-builder,maven-plugin-api,maven-settings,maven-settings-builder
22
23 maven-aether-provider.sonar.projectName=Maven Aether Provider
24 maven-artifact.sonar.projectName=Maven Artifact
25 maven-compat.sonar.projectName=Maven Compat
26 maven-core.sonar.projectName=Maven Core
27 maven-embedder.sonar.projectName=Maven Embedder
28 maven-model.sonar.projectName=Maven Model
29 maven-model-builder.sonar.projectName=Maven Model Builder
30 maven-plugin-api.sonar.projectName=Maven Plugin API
31 maven-settings.sonar.projectName=Maven Settings
32 maven-settings-builder.sonar.projectName=Maven Settings Builder
```



8.2 분석 [6/7]

✓ Multi-project

▪ Sonar 웹에서 확인

The image shows two screenshots of the SonarQube web interface. The left screenshot displays the main dashboard for a 'Maven 3' project. It includes sections for Lines of code (56,214), Files (645), Duplications (1.0%), Complexity (2.2 /function, 13.9 /class, 15.5 /file), and Events (Jun 20 2014, Version 3.2.1). The right screenshot shows a detailed view of duplicated code in 'SessionScope.java'. It lists 928 duplicated lines across various files, with a focus on 'SessionScope.java' and 'MojoExecutionScope.java'. The code snippets show the implementation of the 'enter()' method.

Duplicated lines (%)
1.0%
Drilldown on 928 Duplicated lines

File	Lines
src/main/java/org/apache/maven/repository/internal	96
src/main/java/org/apache/maven/plugin	88
Maven Core	428
src/main/java/org/apache/maven/project/artifact	72
src/main/java/org/apache/maven/model/merge	66
Maven Model Builder	110
Maven Compat	99
Maven Aether Provider	96
Maven Embedder	80
Maven Model	66

Coverage 57.7% Lines: 137 Duplicated lines: 79 Duplicated blocks: 2

SessionScope.java

```
public void enter()
{
    Linkedlist<ScopeState> stack = values.get();
    if (stack == null)
    {
        stack = new LinkedList<ScopeState>();
    }
}
```

MojoExecutionScope.java

```
final Linkedlist<ScopeState> stack = values.get();
if (stack == null || stack.isEmpty())
{
    throw new IllegalStateException();
}
stack.removeFirst();
```

8.2 분석 [7/7]

✓ 플러그인

The screenshot shows a Confluence page titled "Plugin version matrix" from the SonarQube documentation. The page includes a sidebar with navigation links like Developers' Seven Deadly Sins, User Guide, Setup and Upgrade, etc., and a main content area with a table showing plugin compatibility across different SonarQube versions.

Plugin version matrix

Created by Simon Brandhof, last modified by Olivier Gaudin on Jun 13, 2014

Legend:

- ✗ - incompatible
- ⚠ - not in update center

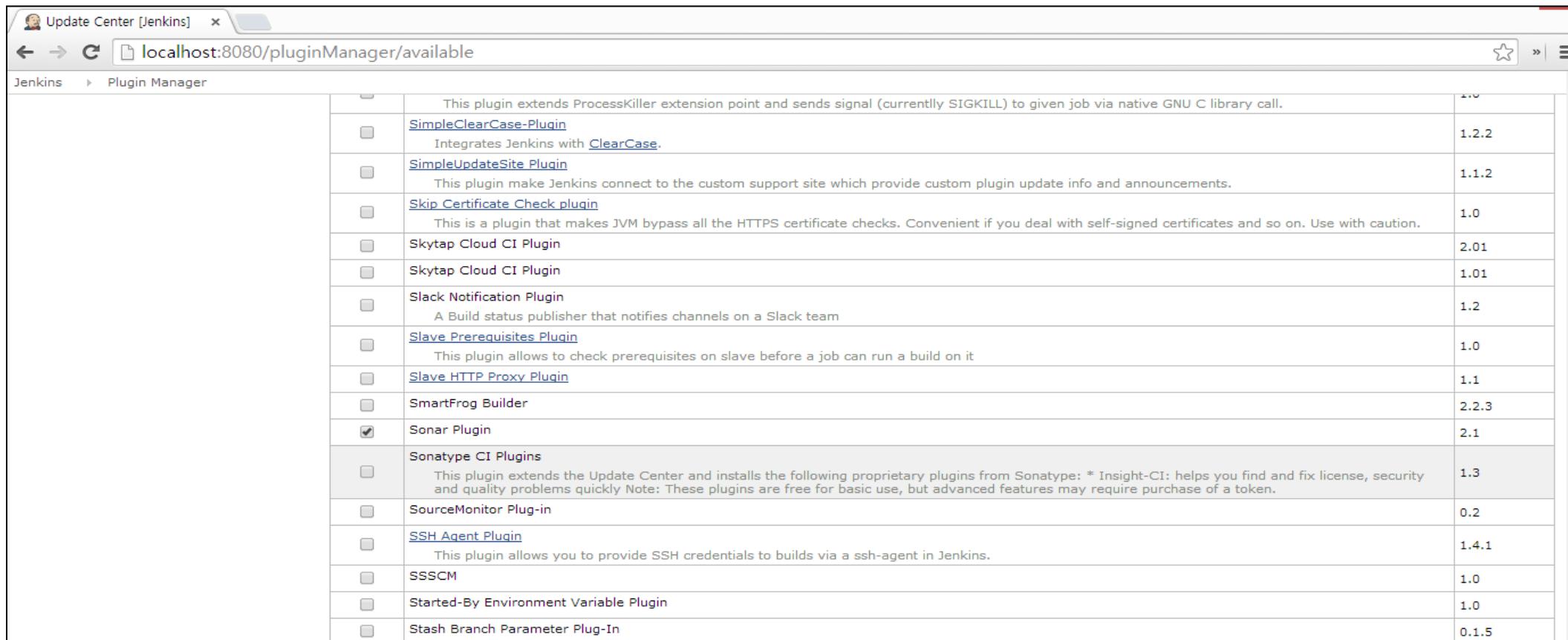
SonarQube Version	3.6	3.7 (LTS)	4.0	4.1	4.2	4.3
Plugin / Release Date	Jun 2013	Jul 2013	Nov 2013	Feb 2014	Mar 2014	Jun 2014
Abacus	0.1	0.1	0.1	0.1	0.1	0.1
ABAP	2.3	2.3	2.3	3.1	3.1	3.1
Analysis Bootstrapper for Visual Studio Projects	✗	1.0	1.0	1.0	1.0	1.0
Android	0.1	0.1	0.1	0.1	1.0	1.0
Artifact Size	0.3	0.3	0.3	0.3	0.3	0.3
Branding	0.4	0.4	0.4	0.4	✗	✗
Build Breaker	1.1	1.1	1.1	1.1	1.1	1.1
Build Stability	1.2	1.2	1.2	1.2	1.2	1.2
C#	✗	3.1	3.1	3.1	3.1	3.1
C/C++	2.3	2.3	2.3	2.3	2.3	2.3
Checkstyle	2.0	2.0	2.0	2.0	2.1	2.1
Chinese Pack	1.5	1.6	1.7	1.8	1.8	1.8
Clirr	1.1	1.1	1.1	1.1	1.1	1.1
Clover	2.9	2.9	2.9	2.9	3.0	3.0
Cobertura	✗	✗	✗	✗	1.6.1	1.6.1
Cobol	✗	2.0	2.0	2.0	2.0	2.0
Crowd	1.0	1.0	1.0	1.0	1.0	1.0
CSS	✗	1.0	1.0	1.0	1.0	1.0
Developer Cockpit	1.3.1	1.3.1	1.3.1	1.3.1	1.4	1.5

EVALUATION LICENSE Are you enjoying Confluence? Please consider purchasing it today.
You have 16 days left in your Gliffy Confluence Plugin free trial

Powered by Atlassian Confluence 5.5.2, Team Collaboration Software · Report a bug · Atlassian News

8.3 CI 연계 (1/7)

✓ Jenkins Sonar 플러그인



The screenshot shows the Jenkins Update Center interface with the URL `localhost:8080/pluginManager/available`. The "Plugin Manager" section is open, displaying a list of available Jenkins plugins. The "Sonar Plugin" is highlighted with a checked checkbox in the left column. The table lists the following plugins:

Plugin Name	Description	Version
SimpleClearCase-Plugin	Integrates Jenkins with ClearCase .	1.2.2
SimpleUpdateSite Plugin	This plugin make Jenkins connect to the custom support site which provide custom plugin update info and announcements.	1.1.2
Skip Certificate Check plugin	This is a plugin that makes JVM bypass all the HTTPS certificate checks. Convenient if you deal with self-signed certificates and so on. Use with caution.	1.0
Skytap Cloud CI Plugin		2.0.1
Skytap Cloud CI Plugin		1.01
Slack Notification Plugin	A Build status publisher that notifies channels on a Slack team	1.2
Slave Prerequisites Plugin	This plugin allows to check prerequisites on slave before a job can run a build on it	1.0
Slave HTTP Proxy Plugin		1.1
SmartFrog Builder		2.2.3
<input checked="" type="checkbox"/> Sonar Plugin		2.1
Sonatype CI Plugins	This plugin extends the Update Center and installs the following proprietary plugins from Sonatype: * Insight-CI: helps you find and fix license, security and quality problems quickly Note: These plugins are free for basic use, but advanced features may require purchase of a token.	1.3
SourceMonitor Plug-in		0.2
SSH Agent Plugin	This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.	1.4.1
SSSCM		1.0
Started-By Environment Variable Plugin		1.0
Stash Branch Parameter Plug-In		0.1.5

8.3 CI 연계 (2/7)

✓ Jenkins Sonar 설정

The screenshot shows the Jenkins configuration interface for Sonar Runner. At the top left, there is a link to 'JDK installations...'. Below it, the 'Sonar Runner' section is displayed. Under 'Sonar Runner installations', there is one entry named 'Local Sonar Runner'. This entry includes fields for 'Name' (set to 'Local Sonar Runner') and 'SONAR_RUNNER_HOME' (set to 'C:\Tools\sonar-runner-2.4'). There is also a checkbox labeled 'Install automatically' which is unchecked. On the right side of the entry, there is a question mark icon and a 'Delete Sonar Runner' button. At the bottom left of the section, there is a 'Add Sonar Runner' button. A link at the bottom of the page reads 'List of Sonar Runner installations on this system'.

8.3 CI 연계 (3/7)

✓ Jenkins Sonar 설정

Sonar

Sonar installations	Name	Local Sonar
	Disable	<input type="checkbox"/> Check to quickly disable Sonar on all jobs.
	Server URL	<code>http://localhost:9000</code>
	Sonar account login	Default is <code>http://localhost:9000</code> admin
	Sonar account password	Sonar account used to perform analysis. Mandatory since Sonar 3.4 when anonymous access is disabled. *****
	Database URL	Sonar account used to perform analysis. Mandatory since Sonar 3.4 when anonymous access is disabled. <code>jdbc:mysql://localhost:3306/sonar_schema?useUnicode=true&characterEncoding=utf8</code>
	Database login	Do not set if default embedded database. sonar
	Database password	Default is sonar. *****
	Database driver	Default is sonar. <code>com.mysql.jdbc.Driver</code>
	Version of sonar-maven-plugin	Do not set if you use the default embedded database on localhost.
	Additional properties	If not specified, then <code>sonar:sonar</code> will be used.
		Additional properties to be passed to the mvn executable (example : -Dsome.property=some.value)
Trigger Exclusions (only for jobs with Sonar as a post build action)		
<input type="checkbox"/> Skip if triggered by SCM Changes		
<input type="checkbox"/> Skip if triggered by the build of a dependency		
Skip if environment variable is defined and set to true <input type="text"/>		

Add Sonar **Delete Sonar**

List of Sonar installations

8.3 CI 연계 (4/7)

✓ Maven Project Job 등록

Maven project name

Description

[Escaped HTML] [Preview](#)

Discard Old Builds [?](#)

This build is parameterized [?](#)

Disable Build (No new builds will be executed until the project is re-enabled.) [?](#)

Execute concurrent builds if necessary [?](#)

Advanced Project Options [Advanced...](#)

Source Code Management [Advanced...](#)

CVS

CVS Projectset

None

Subversion

Modules Repository URL [?](#)

Local module directory (optional) [?](#)

Repository depth [?](#)

Ignore externals [?](#)

Add more locations... [Add more locations...](#)

Check-out Strategy [▼](#)

Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser [▼](#) [?](#)

Advanced... [Advanced...](#)

8.3 CI 연계 (5/7)

✓ Post Step

✓ Sonar 등록

Post Steps

Run only if build succeeds Run only if build succeeds or is unstable Run regardless of build result
Should the post-build steps run only for successful builds, etc.

Invoke Standalone Sonar Analysis

Task to run: []

JDK: (Inherit From Job) []

Path to project properties: []

Project properties:

```
sonar.projectKey=apache:maven
sonar.projectName=Maven 3
sonar.projectVersion=3.2.1

# Path to the parent source code directory.
# Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
# Since SonarQube 4.2, this property is optional. If not set, SonarQube starts looking for source code
# from the directory containing the sonar-project.properties file.
sonar.sources=src/main/java

sonar.tests=src/test/java

sonar.junit.reportsPath=target/surefire-reports

sonar.sourceEncoding=UTF-8

sonar.modules=maven-aether-provider,maven-artifact,maven-compat,maven-core,maven-embedder,maven-model,maven-model-builder,maven-plugin-api,maven-settings,maven-settings-builder

maven-aether-provider.sonar.projectName=Maven Aether Provider
maven-artifact.sonar.projectName=Maven Artifact
maven-compat.sonar.projectName=Maven Compat
maven-core.sonar.projectName=Maven Core
maven-embedder.sonar.projectName=Maven Embedder
maven-model.sonar.projectName=Maven Model
maven-model-builder.sonar.projectName=Maven Model Builder
maven-plugin-api.sonar.projectName=Maven Plugin API
maven-settings.sonar.projectName=Maven Settings
maven-settings-builder.sonar.projectName=Maven Settings Builder
```

JVM Options: [] Delete

Add post-build step ▾

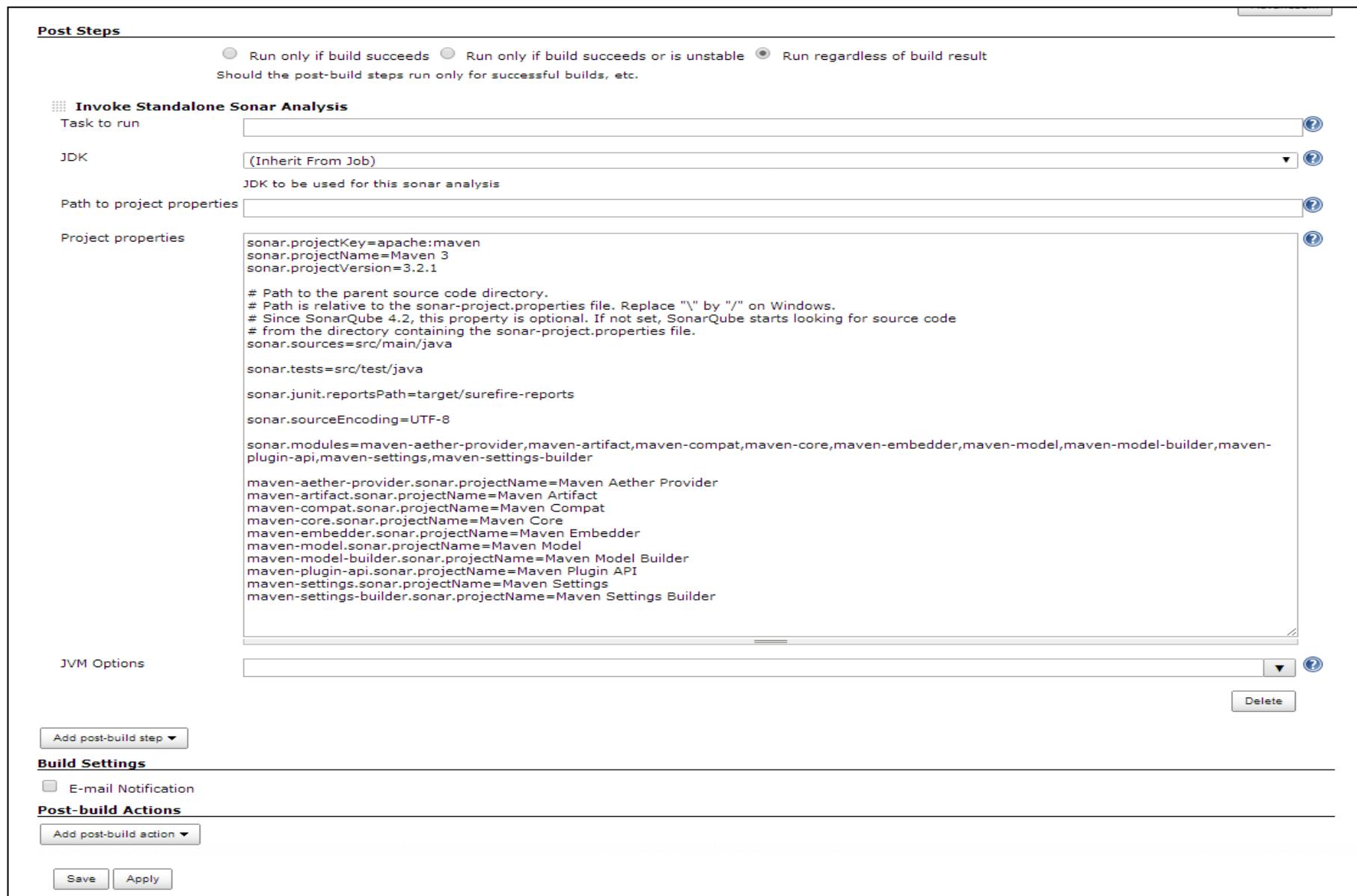
Build Settings

E-mail Notification

Post-build Actions

Add post-build action ▾

Save **Apply**



8.3 CI 연계 (6/7)

✓ Job 실행

Jenkins

Back to Project Status Changes Console Output View as plain text Edit Build Information Previous Build

Console Output

Started by user anonymous
Building in workspace C:\Tools\Jenkins\workspace\Apache Maven3
Updating https://svn.apache.org/repos/asf/maven/maven-3/trunk at revision '2014-06-20T05:36:08,898 +0900'
At revision 1604029
no change for https://svn.apache.org/repos/asf/maven/maven-3/trunk since the previous build
Parsing POMs
[Apache Maven3] \$ java -cp C:\Tools\Jenkins\plugins\maven-plugin\WEB-INF\lib\maven3-agent-1.5.jar;C:\Tools\apache-maven-3.1.1\boot\plexus-classworlds-2.5.1.jar;C:\Tools\apache-maven-3.1.1\coof\logging\jenkins.maven3.agent.Maven3Main C:\Tools\apache-maven-3.1.1\c:\Tools\Jenkins\war\WEB-INF\lib\remoting-2.43.jar;C:\Tools\Jenkins\plugins\maven-plugin\WEB-INF\lib\maven3-interceptor-commons-1.5.jar 52792
====[JENKINS REMOTING CAPACITY]==>channel started
Executing Maven: -B -f C:\Tools\Jenkins\workspace\Apache Maven3\pom.xml install
[INFO] Scanning for projects...
[INFO] -----
[INFO] Reactor Build Order:
[INFO]
[INFO] Apache Maven
[INFO] Maven Model
[INFO] Maven Artifact
[INFO] Maven Plugin API
[INFO] Maven Model Builder
[INFO] Maven Settings
[INFO] Maven Settings Builder
[INFO] Maven Repository Metadata Model
[INFO] Maven Aether Provider
[INFO] Maven Core
[INFO] Maven Compat
[INFO] Maven Embedder
[INFO] Maven Distribution
[INFO]
[INFO] -----
[INFO] Building Apache Maven 3.1-SNAPSHOT
[INFO]
[INFO] --- maven-remote-resources-plugin:1.3:process (default) @ maven ---
[INFO]
[INFO] --- animal-sniffer-maven-plugin:1.6:check (check-java-1.5-compat) @ maven ---
[INFO] Checking unresolved references to org.codehaus.mojo.signature:java15:1.0
[INFO]
[INFO] --- maven-site-plugin:3.2:attach-descriptor (attach-descriptor) @ maven ---
[INFO]
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ maven ---
[INFO] Installing C:\Tools\Jenkins\workspace\Apache Maven3\pom.xml to C:\Windows\System32\config\systemprofile\m2\repository\org\apache\maven\maven-3.1-SNAPSHOT\maven-3.1-SNAPSHOT.pom
[INFO] Installing C:\Tools\Jenkins\workspace\Apache Maven3\target\maven-3.1-SNAPSHOT-site.xml to C:\Windows\System32\config\systemprofile\m2\repository\org\apache\maven\maven-3.1-SNAPSHOT\maven-3.1-SNAPSHOT-site.xml
[INFO]
[INFO] -----
[INFO] Building Maven Model 3.1-SNAPSHOT
[INFO]
[INFO] --- modello-maven-plugin:1.6:java (standard) @ maven-model ---
[INFO] outputDirectory: C:\Tools\Jenkins\workspace\Apache Maven3\maven-model\target\generated-sources\modello
[INFO] Working on model: src/main/ido/maven.mdo
[INFO] Generating current version: 4.0.0
[INFO] --- modello-maven-plugin:1.6:xpp3-reader (standard) @ maven-model ---
[INFO] outputDirectory: C:\Tools\Jenkins\workspace\Apache Maven3\maven-model\target\generated-sources\modello
[INFO] Working on model: src/main/ido/maven.mdo
[INFO] Generating current version: 4.0.0
[INFO] --- modello-maven-plugin:1.6:xpp3-extended-reader (standard) @ maven-model ---
[INFO] outputDirectory: C:\Tools\Jenkins\workspace\Apache Maven3\maven-model\target\generated-sources\modello
[INFO] Working on model: src/main/ido/maven.mdo
[INFO] Generating current version: 4.0.0
[INFO] --- modello-maven-plugin:1.6:xpp3-writer (standard) @ maven-model ---

8.3 CI 연계 (7/7)

✓ Sonar 웹

Sonar dashboard for Maven 3 project.

Version 3.2.1 - Jun 20 2014 05:37

Dashboard

- Hotspots
- Issues
- Time Machine
- TOOLS
 - Components
 - Issues Drilldown
 - Design
 - Libraries
 - Compare

sonarqube

Lines of code: 55,746 (91,134 lines, 18,646 statements)

Files: 623 (177 directories, 700 classes, 4,475 functions, 683 accessors)

Duplications: 0.8% (760 lines, 32 blocks, 22 files)

Issues: 13,055 (12,170 Critical, 549 Major, 315 Minor, 2 Info)

Technical Debt: 103d

Unit Tests Coverage: 100.0% (0 failures, 0 errors, 684 tests, 22 sec)

Complexity: 2.2 /function, 14.2 /class, 16.0 /file (Total: 9,971)

Events: All (Jun 20 2014, Version 3.2.1)

Key: apache:maven

SonarQube™ technology is powered by SonarSource SA
Version 4.3.1 - [Community](#) - [Documentation](#) - [Get Support](#) - [Plugins](#)

✓ 토론

감사합니다...

- ❖ 송태국 (tsong@nextree.co.kr)
- ❖ 넥스트리컨설팅(주) 대표이사
- ❖ 넥스트리소프트(주) 부사장
- ❖ www.nextree.co.kr