하나씩 쉽게 따라 해보는 IBM WebSphere Application Server(WAS) v7 – 26

이정운 (juwlee@kr.ibm.com)

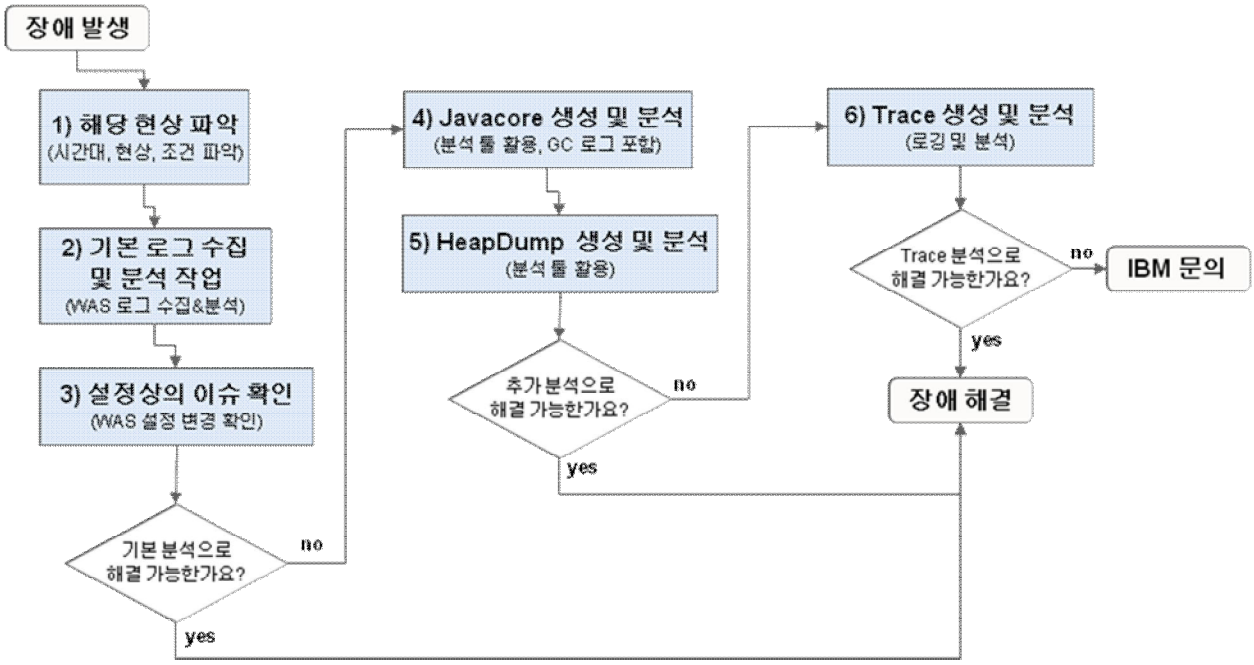
하나씩 쉽게 따라 해보는 IBM WAS v7 스물 여섯 번째 이야기를 오래만에 시작합니다. 어느덧 강의를 시작한지 횟수로 어언 3년이 훌쩍 지나가고 있습니다. 스물 여섯 번째 이야기는 3부 강의의 다섯번째 로서 WebSphere 에서 문제상황과 분석(장애처리)에 대한 기본 개념에 대한 설명을 진행하도록 하겠습니다. 장애 상황과 분석은 사실적으로 워낙 다양하고 방대한 분량을 다루어야 하기 때문에 이미 언급한 것처럼 기본 개념정도만 간단하게 다루도록 하겠습니다.

하지만, 하단부에 진행하는 Javacore 와 Heapdump 분석의 경우에는 WebSphere 가 아닌 다른 WAS, 다른 Java 프로그램의 장애상황 분석에서도 도움이 될 만한 내용이기 때문에, 또는 운영중에 가장 많이 사용할 내용이기 때문에 조금 더 숙지하시고 이해할 수 있는 시간을 할애할 수 있다면 많은 도움이 될 수 있다고 생각합니다.

마지막으로 아마도 이번 강좌가 하나씩 쉽게 따라 해보는 IBM WAS v7 의 마지막이 될 것 같습니다. 그렇다고 완전히 손을 뗀다는 의미는 아니며 다른 강좌거리 다른 할 이야기가 오면 다시 다른 강좌로 돌아오도록 노력하겠습니다. ^^& 그럼 지금부터 하나씩 쉽게 따라 해보는 IBM WAS v7 스물 여섯 번째 이야기 WAS 장애처리에 대해서 강좌를 진행하도록 하겠습니다.

Part 0. 장애처리의 기본 프로세스

실제 강의를 시작하기 전에 장애처리의 기본 프로세스 부분에 대해서 먼저 이야기하고 넘어가도록 하겠습니다. 대부분의 사이트 또는 운영하시는 분들, 또는 공부하시는 분들이 WAS 를 사용하면서 장애를 겪게 되면 가장 많이 하는 행동은 해당 WAS 를 재시작하는 것입니다. 그리고 그것을 통해서 문제를 해결하게 되는 경우 그냥 그런가보다 하고 넘어가는 경우가 많습니다. 하지만, 그럴수록 금방 해결할 수 있는 문제가 점점 커질 수 있는 상황을 유발 할 수 있으므로 좀 더 꼼꼼히 프로세스를 세워서 분석을 통한 장애에 대처하는 자세가 필요합니다.
(그러나, 그냥 테스트나 개발 용도이신 분들은 그렇게 심오하게 고민하실 필요는 없습니다.^^)



위의 그림은 제가 지금 강좌를 진행하면서 생각해본 간단한 장애처리 프로세스 입니다. 당연해 보이지만 실제로 장애가 발생하면 해당 프로세스를 따르는 것은 아주 어렵습니다. 마음속으로 한번 생각해 보시고 본인 환경에 맞게끔 다시 적절히 수정해서 상황에 따라 사용하면 도움이 될 것입니다. 이러한 장애처리 프로세스가 필요한 이유 중 가장 중요한 것은 실제로 중요한 것을 놓치지 않으려고 하는 것입니다. 정확한 장애에 대한 분석만이 문제를 가장 빠르게 해결 할 수 있습니다.

해당 프로세스는 기본 분석, 추가 분석, Trace 분석의 3단계로 나누었으며 해당 단계에서 어떠한 로그를 어떻게 보아야 하는지에 대한 이야기를 강좌로 진행할 예정입니다. 대부분의 장애는 기본 분석 부분에서 해결할 수 있으며, 메모리나 CPU 관련한 장애일 경우에는 추가 분석을 통해서, 마지막으로 Trace 분석까지 가는 경우는 제품상에서 가지고 있는 이슈일 확률이 높습니다. (반드시 그런 것은 아닙니다.)

Part 1. 기본 분석

기본 분석은 장애가 발생하면 제일 처음 이루어져야 하는 행위이며 대부분의 문제가 여기서 해결 될 수 있습니다. 아주 기본이긴 하지만 많이 간과하는 부분 중에 하나입니다. 예를 들어 제가 1번으로 이야기 한 것은 해당 장애의 현상 파악입니다. 장애가 발생하면 "장애가 발생했다. 로그 수집하고 봐야지." 라고 보통 판단하는데 가장 중요한 것은 현상을 먼저 파악하는 것입니다. 몇 시에 발생했으며, 어떠한 문제가 발생하는지, 어떤 특별한 조건이 발생하는지 등을 먼저 파악을 끝내놓아야 정확하게 다음 단계의 로그를 분석할 수 있습니다. 이러한 전제 조건 없이 바로 다음 단계로 넘어가게 된다면 잘못된 분석을 할 수도 있습니다. 따라서, 장애 시점, 장애 현상, 장애 반복 여부, 장애 조건 등을 먼저 확인한 후에 정확한 기본 장애 분석 절차에 따라 로그를 분석하시는 것이 도움이 됩니다.

기본적인 로그 분석 단계를 위하여 WebSphere 는 기본적으로 다음과 같은 다양한 로그들을 제공합니다. 이를 통해 서비스 상황을 추적하고, 장애 시 신속하고 정확하게 원인을 파악 할 수 있도록 도와드립니다. 참고적으로 로그에 기록 될 정보의 레벨은 관리 툴을 통해 조정됩니다.

로그 파일명	내용	용도
access_log	HTTP 를 이용하여 접근한 사용자 및 요청에 관한 내용을 포함	웹서버 장애 분석, 사용자 분석
cache.log	다이나믹 캐시의 사용 상황 관한 내용 포함	웹서버 장애 분석
plugin.log	웹서버 플러그인의 동작중 발생한 행위와 Error 메시지 등이 기록	웹서버와 WAS 간 통신 장애 분석
SystemOut.log	WAS 자체의 동작 메시지, 애플리케이션의 Console Out, Exception 및 Error 메시지 등이 기록	WAS 의 동작 분석
SystemErr.log	Exception 및 Error 내역만 기록	WAS 의 장애 분석
native_stdout.log	WAS 운영 중 O/S가 던진 Output 들이 기록	JVM 레벨의 동작 및 장애 분석
native_stderr.log	WAS 운영 중 O/S가 던진 Error 메시지 들이 기록	JVM 레벨의 장애 분석
activity.log	WAS에서 일어나는 모든 행위들을 기록한 바이너리 로그로써 Log Analyzer 라는 툴로 분석	WAS 의 동작 분석
trace.log	WAS 의 모든 동작을 상세히 기록	WAS 의 동작을 함수 스텝 단위로 분석
tranlog	WAS 자신이 자동 트랜잭션 복구를 위해 사용하는 바이너리 로그	2Phase commit 트랜잭션 자동 복구
heap dump	JVM 내 객체의 사용 상황을 기록한 덤프 로그	WAS 의 메모리 관련 장애 분석
thread dump	JVM 내 스레드의 동작 상황을 기록한 덤프 로그	WAS 의 Hang 및 서비스 지연 관련 장애 분석
FFDC	First Failure Data Captuer - 장애시의 WAS의 설정 및 환경, 운용 상황 등을 수집하여 만든 로그 파일들의 모음. IBM 서비스 센터에서 사용하기 위한 로그	WAS 의 장애 분석

먼저 살펴볼 것은 가장 많이 보는 로그인 SystemOut.log 입니다. 기본적으로 해당 로그에는 WAS 자체의 동작 메시지, 어플리케이션의 Console out, Exception 및 Error 메시지등이 기록됩니다. (해당 파일은 기본적으로 로그 저장 위치를 변경하지 않았다면 다음과 같은 디렉토리에 위치합니다. C:₩IBM₩WebSphere7₩AppServer₩profiles₩AppSrv03₩logs₩server1)

```
************ Start Display Current Environment ************
WebSphere Platform 7.0.0.15 [ND 7.0.0.15 cf151107.06][WXDOP 6.1.1.2 cf21023.56688] running with process name kr050578Node01Cell\kr050578Node03\server1 and
Host Operating System is Windows 7, version 6.1 build 7601 Service Pack 1
Java version = 1.6.0, Java Compiler = j9jit24, Java VM name = IBM J9 VM
was.install.root = C:\IBM\WebSphere7\AppServer
user.install.root = C:\IBM\WebSphere7\AppServer\profiles\AppSrv03
Java Home = C:\IBM\WebSphere7\AppServer\java\jre
ws.ext.dirs = C:\IBM\WebSphere7\AppServer/java/lib;C:\IBM\WebSphere7\AppServer\profiles\AppSrv03/classes;C:\IBM\WebSphere7\AppServer/classes;C:\IBM\WebSphe
Classpath = C:\IBM\WebSphere7\AppServer\profiles\AppSrv03/properties;C:\IBM\WebSphere7\AppServer/properties;C:\IBM\WebSphere7\AppServer/lib/startup.jar;C:\
Java Library path = C:\IBM\WebSphere7\AppServer\java\jre\bin;.;C:\IBM\WebSphere7\AppServer\bin;C:\IBM\WebSphere7\AppServer\java\bin;C:\IBM\WebSphere7\AppSe
************ End Display Current Environment ************
[11. 4. 9   1:50:43:238 KST] 00000000 ManagerAdmin  I   TRAS0017I: 시작 추적 상태는 *=info입니다.
[11. 4. 9   1:50:43:628 KST] 00000000 ManagerAdmin  I   TRAS0111I: 사용 중인 메시지 ID는 제공되지 않습니다.
[11. 4. 9   1:50:43:909 KST] 00000000 ModelMgr      I   WSVR0800I: 코어 구성 모델 초기화 중
[11. 4. 9   1:50:45:204 KST] 00000000 ComponentMeta I   WSVR0179I: 런타임 프로비저닝 기능이 사용 불가능합니다. 모든 컴포넌트가 시작됩니다.
[11. 4. 9   1:50:45:547 KST] 00000000 ProviderTrack I com.ibm.ffdc.osgi.ProviderTracker AddingService FFDC1007I: FFDC 프로바이더가 설치되었습니다: com.ibm.
[11. 4. 9   1:50:45:625 KST] 00000000 ProviderTrack I com.ibm.ffdc.osgi.ProviderTracker AddingService FFDC1007I: FFDC 프로바이더가 설치되었습니다: com.ibm.
[11. 4. 9   1:50:45:937 KST] 00000000 AdminInitiali A   ADMN0015I: 관리 서비스가 초기화되었습니다.
[11. 4. 9   1:50:46:873 KST] 00000000 PluginConfigS I   PLGC0057I: 플러그인 구성 서비스가 시작되었습니다.
[11. 4. 9   1:50:47:201 KST] 00000000 SSLComponentI I   CWPKI0001I: SSL 서비스가 구성을 초기화하는 중입니다.
[11. 4. 9   1:50:47:232 KST] 00000000 WSKeyStore    W   CWPKI0041W: 하나 이상의 키 스토어에서 기본 암호를 사용하고 있습니다.
[11. 4. 9   1:50:47:294 KST] 00000000 SSLConfigMana I   CWPKI0027I: HTTPS URL 연결에 대한 기본 호스트 이름 확인을 사용할 수 없습니다.
[11. 4. 9   1:50:47:341 KST] 00000000 SSLDiagnostic I   CWPKI0014I: SSL 컴포넌트의 FFDC 진단 모듈 com.ibm.ws.ssl.core.SSLDiagnosticModule이(가) 등록되었습니
[11. 4. 9   1:50:47:778 KST] 00000000 SSLComponentI I   CWPKI0002I: SSL 서비스 초기화가 완료되었습니다.
[11. 4. 9   1:50:47:809 KST] 00000000 DiagnosticCon I com.ibm.wsspi.rasdiag.DiagnosticConfigHome setStateCollectionSpec RASD0012I: Uninitialized Value에서
[11. 4. 9   1:50:47:825 KST] 00000000 PMIImpl        A   CWPMI1001I: PMI 사용 가능
[11. 4. 9   1:50:49:307 KST] 00000000 SibMessage     I   [:] CWSIU0000I: 릴리스: WAS70.SIB 레벨: o1050.07
[11. 4. 9   1:50:49:432 KST] 00000000 ODCManagerFac I   Factory set to <null>
[11. 4. 9   1:50:49:650 KST] 00000000 SecurityDM     I   SECJ0231I: 보안 컴포넌트의 FFDC 진단 모듈 com.ibm.ws.security.core.SecurityDM이(가) 등록되었습니다.
[11. 4. 9   1:50:49:728 KST] 00000000 distSecurityC I   SECJ6004I: 보안 감사가 사용 불가능합니다.
[11. 4. 9   1:50:49:744 KST] 00000000 distSecurityC I   SECJ0309I: Java 2 보안이 사용 불가능합니다.
[11. 4. 9   1:50:49:759 KST] 00000000 Configuration A   SECJ0215I: JAAS 로그인 프로바이더 구성 클래스를 com.ibm.ws.security.auth.login.Configuration(으)로
```

위에서 보시는 것처럼 SystemOut.log 는 text 형태로 구성되어 있으며 제일 상단에는 현재 수행되는 WebSphere 의 버전 이라던지 OS, JVM 버전, 환경 경로등과 같은 수행 환경에 대한 기본 정보들이 출력됩니다. 이후의 로그가 실제 기본 로그로서 다음과 같은 구조를 가집니다.

날자/시간	Thread ID	Component 명	구분 기호	메시지 코드 값	메시지

```
[11. 4. 9   1:50:43:238 KST] 00000000 ManagerAdmin  I   TRAS0017I: 시작 추적 상태는 *=info입니다.
```

구분기호
I : Information
W : Warning
A : Alarm
O : Output
E : Error

위와 같은 구조를 가지게 되므로 구조에 따라서 정확하게 문제 분석을 하면 됩니다. SystemOut.log 는 구조를 가지고 있지만 그 구조만 이해하면 직관적으로 읽을 수 있게 구성되어 있습니다. 보통 장애상황이 발생하게 되면 구분기호 'E' 를 위주로 해당 메시지를 살펴보시면 됩니다. 추가적으로 실제로 Exception 이나 Error 가 발생하게 되면 하단처럼 문제의 Stacktrace 를 같이 생성하게 되므로 이를 통해서 문제의 원인을 좀 더 쉽게 찾으실 수 있습니다.

```
[11. 5. 9   16:04:33:247 KST] 00000015 webapp        E com.ibm.ws.webcontainer.webapp.WebApp logServletError SRVE0293E: [Servlet 오류]-[/logon.jsp]: java.lang.NullPoi
  at com.ibm.ws.xd.visualizationengine.cacheservice.AdminSessionCounter.setCacheManagerActiveSessions(AdminSessionCounter.java:61)
  at com.ibm.ws.xd.visualizationengine.cacheservice.AdminSessionCounter.incrementSessionCount(AdminSessionCounter.java:20)
  at com.ibm.ws.console.xdoperations.AdminSessionListener.sessionCreated(AdminSessionListener.java:23)
  at com.ibm.ws.session.http.HttpSessionObserver.sessionCreated(HttpSessionObserver.java:111)
  at com.ibm.ws.session.SessionEventDispatcher.sessionCreated(SessionEventDispatcher.java:98)
  at com.ibm.ws.session.SessionManager.createISession(SessionManager.java:268)
  at com.ibm.ws.session.SessionManager.createSession(SessionManager.java:635)
  at com.ibm.ws.session.SessionContext.getIHttpSession(SessionContext.java:473)
  at com.ibm.ws.session.SessionContext.getIHttpSession(SessionContext.java:408)
  at com.ibm.ws.webcontainer.srt.SRTRequestContext.getSession(SRTRequestContext.java:89)
  at com.ibm.ws.webcontainer.srt.SRTServletRequest.getSession(SRTServletRequest.java:1772)
  at com.ibm.ws.webcontainer.srt.SRTServletRequest.getSession(SRTServletRequest.java:1759)
  at com.ibm.ws.console.core.servlet.WSCUrlFilter.doFilter(WSCUrlFilter.java:154)
```

그리고 문제를 분석할 때는 문제상황의 코드값과 Exception 을 먼저 살펴보시길 권고 드립니다.

```
webapp      E com.ibm.ws.webcontainer.webapp.WebApp logServletError SRVE0293E: [Servlet 오류]-[/logon.jsp]: java.lang.NullPointerException
```

IBM WAS v7.0 InfoCenter 에 보면 코드값 리스트를 확인할 수 있으며 이를 통해서 좀 더 쉽게 해당 문제의 원인과 취해야 할 조치를 찾으실 수 있습니다.

SRVE0308E: Filtering by asterisk is not allowed.

Explanation	Filtering by asterisk is not allowed because disabling serving all servlets by classname is handled by the com.ibm.ws.webcontainer.disallowserveservletsbyclassname custom property. This is an application error.
Action	Set com.ibm.ws.webcontainer.disallowserveservletsbyclassname to true to block serving all classes by classname.

SRVE0309E: Servlet on the blocked list: {0}

Explanation	The requested servlet is on the block list provided by the custom property com.ibm.ws.webcontainer.donotservebyclassname. This is an application error.
Action	If this is not the desired behavior, update the property com.ibm.ws.webcontainer.donotservebyclassname.

SRVE0310E: There was an error in a custom property: {1}

Explanation	System action is required.
Action	Verify the custom property is correct

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.messages.doc/com.ibm.ws.webcontainer.resources.Messages.html?resultof=%22%53%52%56%45%30%32%39%33%45%22%20%22%73%72%76%65%30%32%39%33%22%20

하나 덧붙여, 문제 상황을 만나실 때 마다 Google 검색의 힘을 적절하게 잘 활용하면 많은 도움이 된다는 점 기억하시기 바라겠습니다.^^& 특히, WebSphere 같은 경우는 전 세계에서 많은 고객사들이 이미 사용하고 있는 제품이기 때문에 비슷한 문제와 사례를 생각보다 쉽게 찾으실 수 있고 그러한 질문이나 답변들을 통해서 지금 겪고 있는 문제를 해결할 수도 있습니다.

SystemErr.log 는 SystemOut.log 와 동일한 구조의 로그이며 Exception 과 Error 만 별도로 기록됩니다. 다른 Output 이 나오지 않고 Exception 만 나오기 때문에 보기는 편하지만 문제 분석을 위해서 전반적인 흐름을 보기 어려우므로 저 같은 경우는 많이 안보게 되더군요.

다음으로 native_stdout.log 와 native_stderr.log 는 WAS 운영 중에 OS 가 던지는 Output 과 Error 가 기록됩니다. 보통 native_stdout.log 나 native_stderr.log 는 JVM 이슈이거나 native 관련 이슈일 경우에 분석하기 위한 데이터들이 로깅됩니다. 예를 들어 JVM 에서 memory allocation 부분에서 failure 가 발생하거나 crash 가 발생하게 되면 해당로그에 로그가 남게 되므로 이를 분석하여 문제의 원인을 찾고 해결할 수 있습니다. 이와 별도로, "-verbosegc" 같은 옵션을 사용하여 GC 분석을 작동시키게 되면 기본적으로 native_stderr.log 에 GC 데이터들이 기록 되어지며 solaris 환경에서 Javacore 를 생성하게 되면 별도의 파일로 만들어지는 것이 아니라 기본적으로 native_stdout.log 에 javacore 가 기록됩니다.
(왜 인지는 모르겠으나 운영체제 별로 달라서 생각 외로 많이 헷갈립니다. ^^&)

기본적인 문제의 경우에는 위에서 설명 드린 로그들을 확인함으로써 문제의 원인과 해결점을 찾을 수 있습니다. 이와 함께 초기 단계에서 가급적 같이 진행해 봐야 할 작업은 장애가 발생되기 직전 설정 변경이나 어플리케이션 변경 등의 작업이 있었는지 확인하는 작업입니다. 프로젝트가 끝나고 초창기의 WAS 환경을 제외하고는 실제 프로젝트가 안정적으로 돌아가고 난 이후에는 어플리케이션을 새로 배포하거나 설정 같은 것을 변경하는 작업 없이 장애가 발생하는 케이스가 많지 않습니다. 해당 작업들이 있는지 먼저 확인해서 적정한 설정이 반영되었는지 해당 설정이 이슈가 아닌지 확인하는 작업이 필요합니다.

참고 1) IBM WAS v7.0 인 경우 비록 영어이긴 하지만 기본적인 Troubleshooting 가이드를 인터넷상에서 제공하고 있습니다.
http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg27005324

Troubleshooting Guide for WebSphere Application Server

Product documentation

Abstract

The Troubleshooting Guide helps you get started on the troubleshooting process. It takes you through the process of identifying which component is causing the problem, finding the appropriate troubleshooting information, then collecting any necessary MustGather information, and finally submitting a problem to IBM® Support.

Content

To begin troubleshooting, select one of the following topics.

Note: Content is currently being developed for those components and problems that appear blank in the following Troubleshooting Guide.

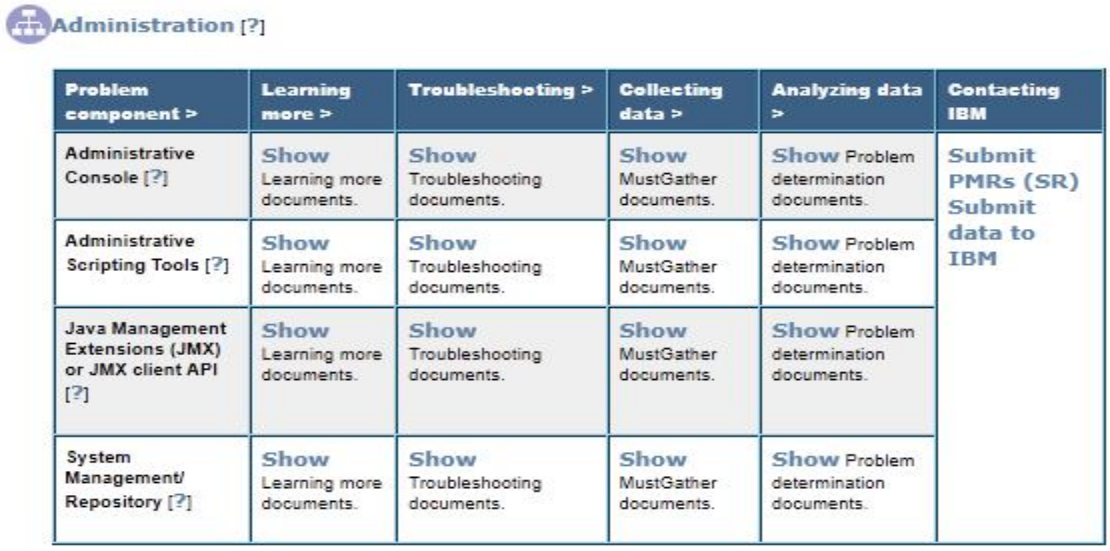
Administration [?]

Application Development [?]

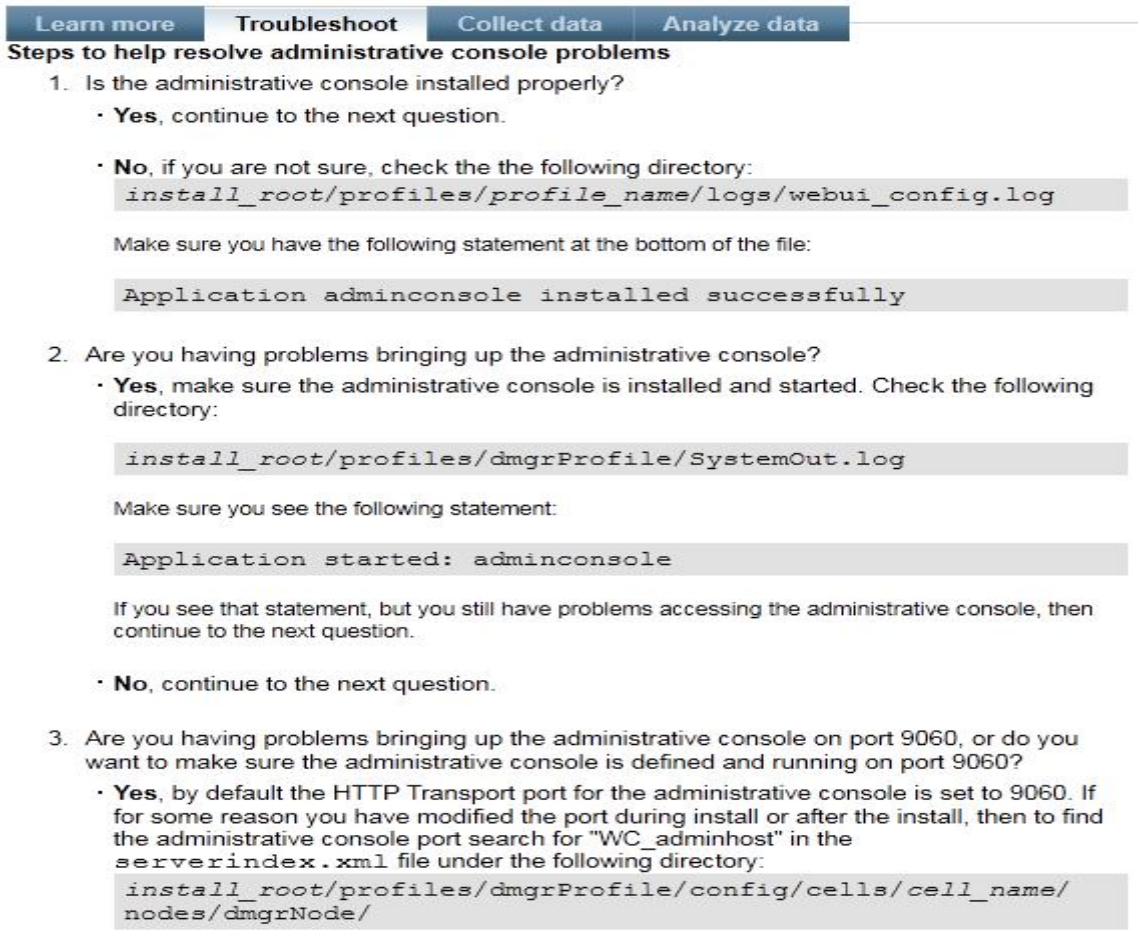
Architecture [?]

Installation [?]

예를 들어 해당 문제가 관리 콘솔 상에서 문제라면 관리 콘솔 부분에서 문제해결 파트를 클릭하면 됩니다.



해당 부분을 클릭하면 하단과 같은 기본적으로 하나씩 단계별로 yes, no 문답식으로 현재 상태를 체크해볼 수 있는 TroubleShooting 가이드를 확인할 수 있습니다. 이를 통해 기본적인 문제 분석 및 해결을 시도해 볼 수 있습니다.



Part 2. 추가 분석 - javacore

이전 파트에서 이야기 드린 일반적인 로그들을 통한 분석과 설정 분석을 했는데도 문제의 원인이나 해결점을 찾지 못했을 경우에 다음 스텝으로 진행하는 것은 추가 분석작업입니다. 여기서는 기본 생성되는 로그가 아니라 필요에 의해서 해당 로그나 덤프를 생성하여 분석을 진행하며 대표적으로 많이 사용되는 것은 Javacore 와 Heapdump 입니다.

Javacore 는 JVM 관련 문제를 분석하기 위한 기본적인 dump 로서 Thread 관련 정보들을 담고 있으므로 보통 Thread dump 라고도 이야기 합니다. 해당 문제가 발생한 시점에 Thread 들의 snapshot 을 text 파일 형태로 출력해 놓은 것으로 이해하시면 됩니다. 다시 좀 더 쉽게 설명 드리자면 JVM 위에서 수없이 많이 돌아가는 Thread 들을 마치 카메라로 찍듯이 찍어서 Text 파일 형태로 만들어 놓은 것입니다. 그러기 때문에 순간적인 Thread 의 상태를 확인할 수 있으며 lock 이나 monitor 관련 이슈나 hang 이슈를 분석하기 위해서는 여러 번 Javacore 를 발생하는 것이 중요합니다.

AIX 와 같은 Unix 환경에서 Javacore 는 kill -3 PID(수행되는 WAS 서버의 PID) 명령을 통해서 생성할 수 있으며 하단과 같이 스크립트를 사용하여 OS 환경에 관계없이 생성할 수도 있습니다.

스크립트를 이용하여 Javacore(thread dump) 강제로 생성하기(Jacl)

```
set jvm [$AdminControl completeObjectName type=JVM,process=server1,*]
$AdminControl invoke $jvm dumpThreads
```

Javacore 는 보통 javacore 이름에 PID 와 time 이 붙는 txt 형식으로 파일이 생성되며 기본적으로는 해당 서버가 속해있는 profiles 디렉토리 밑에 생성되며 text editor 를 통해서 열어보면 다음과 같은 Text 형식으로 되어 있습니다.

```
NULL           ------------------------------------------------------------------------
0SECTION       TITLE subcomponent dump routine
NULL           ===============================
1TISIGINFO     Dump Event "user" (00004000) received
1TIDATETIME    Date:                2010/04/14 at 20:45:28
1TIFILENAME    Javacore filename:   /kimjeh/WebSphereV6.1/AppServer/profiles/juwleeApp01/javacore.20100414.204528.1724892.0001.txt
NULL           ------------------------------------------------------------------------
0SECTION       GPINFO subcomponent dump routine
NULL           ================================
2XHOSLEVEL     OS Level         : AIX 5.3
2XHCPUS        Processors -
3XHCPUARCH       Architecture   : ppc64
3XHNUMCPUS       How Many       : 8
NULL
1XHERROR2      Register dump section only produced for SIGSEGV, SIGILL or SIGFPE.
NULL
NULL           ------------------------------------------------------------------------
0SECTION       ENVINFO subcomponent dump routine
NULL           ================================
1CIJAVAVERSION J2RE 5.0 IBM J9 2.3 AIX ppc64-64 build j9vmap6423-20080315
1CIVMVERSION   VM build 20080314_17962_BHdSMr
1CIJITVERSION  JIT enabled - 20080130_0718ifx2_r8
1CIRUNNINGAS   Running as a standalone JVM
1CICMDLINE     /kimjeh/WebSphereV6.1/AppServer/java/bin/java -Declipse.security -Dwas.status.socket=63652 -Dosgi.install.area=/kimj
1CIJAVAHOMEDIR Java Home Dir:   /kimjeh/WebSphereV6.1/AppServer/java/jre
1CIJAVADLLDIR  Java DLL Dir:    /kimjeh/WebSphereV6.1/AppServer/java/jre/bin
1CISYSCP       Sys Classpath:   /kimjeh/WebSphereV6.1/AppServer/java/jre/lib/ext/ibmorb.jar;/kimjeh/WebSphereV6.1/AppServer/java/j
1CIUSERARGS    UserArgs:
2CIUSERARG               -Xjcl:jclscar_23
2CIUSERARG               -Dcom.ibm.oti.vm.bootstrap.library.path=/kimjeh/WebSphereV6.1/AppServer/java/jre/bin
2CIUSERARG               -Dsun.boot.library.path=/kimjeh/WebSphereV6.1/AppServer/java/jre/bin
2CIUSERARG               -Djava.library.path=/kimjeh/WebSphereV6.1/AppServer/java/jre/bin:/kimjeh/WebSphereV6.1/AppServer/java/jre/
```

해당 Javacore 의 경우 제일 좌측의 숫자는 nesting level (0,1,2,3) 이며 그 뒤로 Section 구분자가 다음과 같이 나오며 마지막은 설명이 나옵니다.

CI
Command-line interpreter
CL
Class loader
LK
Locking
ST
Storage (Memory management)
TI
Title
XE
Execution engine

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp?topic=/com.ibm.java.doc.diagnostics.60/html/javadump_interpret.html

예를 들어 하단과 같은 경우는 nesting level 이 1 이며 Commend-line interpreter Section 이며 Javaversion 정보라고 해석하면 됩니다.

1CIJAVAVERSION J2RE 5.0 IBM J9 2.3 AIX ppc64-64 build j9vmap6423-20080315

위와 같은 구조를 이해했다면 우선 Javacore 에서 제일 처음 확인해 볼 부분은 Title 입니다. Title 에는 Javacore 가 어떠한 이유에서 생성되었는지에 대한 기본 정보가 나옵니다. 기본 사용자가 kill -3 PID 를 이용해서 생성한 경우에는 하단과 같은 user event 를 받았다고 나오며

```
Dump Event "user" (00004000) received
```

System 이 Error 를 받아서 Javacore 를 생성한 경우에는 하단처럼 "systhrow" 라는 문구와 함께 해당 Error 가 표시됩니다. (가용한 Heap 이 부족해서 OutOfMemoryError 를 받은 예제)

```
Dump Event "systhrow" (00040000) Detail : "java/lang/OutOfMemoryError":
```

다음으로 System 의 문제로 인해서 general protection fault (gpf) 같은 이벤트를 받은 경우에는 하단처럼 나옵니다.

```
Dump Event "gpf" (00002000) received
```

이와 같이 Title 에 나타난 기본 정보를 이용해서 Javacore 가 발생한 근본 원인을 확인합니다.

그 중에 gpf 이벤트의 경우는 SIGSEGV 또는 SIGILL과 같이 얘기치 않은 충돌 현상이 발생할 때 받게 되는 이벤트 이며 이럴 경우 GPInfo 항목에 OS 기본 정보에 추가하여 왜 gpf 이벤트가 발생했는지에 대한 보다 자세한 failure 관련 내용이 하단처럼 나오게 됩니다.

```
NULL           ------------------------------------------------------------------------
0SECTION       TITLE subcomponent dump routine
NULL           ===============================
1TISIGINFO     Dump Event "gpf" (00002000) received
1TIDATETIME    Date:                2011/02/11 at 11:08:41
1TIFILENAME    Javacore filename:    /wcs/dump/javacore.20110211.110828.14221524.0004.txt
1TIREQFLAGS    Request Flags: 0x81 (exclusive+preempt)
1TIPREPSTATE   Prep State: 0x0
1TIPREPINFO    Exclusive VM access not taken: data may not be consistent across javacore sections
NULL           ------------------------------------------------------------------------
0SECTION       GPINFO subcomponent dump routine
NULL           ================================
2XHOSLEVEL     OS Level        : AIX 6.1
2XHCPUS        Processors -
3XHCPUARCH       Architecture   : ppc64
3XHNUMCPUS       How Many       : 20
3XHNUMASUP       NUMA is either not supported or has been disabled by user
NULL
1XHEXCPCODE    J9Generic_Signal_Number: 00000004
1XHEXCPCODE    Signal_Number: 0000000B
1XHEXCPCODE    Error_Value: 00000000
1XHEXCPCODE    Signal_Code: 00000033
1XHEXCPCODE    Handler1: 09001000A09438A8
1XHEXCPCODE    Handler2: 09001000A093AE58
NULL
1XHEXCPMODULE  Module: /usr/lib/libc.a
1XHEXCPMODULE  Module_base_address: 0900000000000A00
.....................
1XHFLAGS       VM flags:0000000000040000
```

방금 설명한 것처럼 GPInfo 정보에 좀더 자세한 gpf 이벤트 정보가 나오게 되는데 위의 예제의 경우에는 Signal_Number: 0000000B 정보가 출력되어 있기 때문에 이에 SIGSEGV (Segmentation Violation)이 덤프가 발생한 원인인 것을 확인할 수 있습니다. JVM 에서 사용되는 Signal 에 대한 정보는 다음과 같습니다.

Table 1. Signals used by the JVM

Signal Name	Signal type	Description	Disabled by -Xrs	Disabled by -Xrs:sync
SIGBUS (7)	Exception	Incorrect access to memory (data misalignment)	Yes	Yes
SIGSEGV (11)	Exception	Incorrect access to memory (write to inaccessible memory)	Yes	Yes
SIGILL (4)	Exception	Illegal instruction (attempt to call an unknown machine instruction)	Yes	Yes
SIGFPE (8)	Exception	Floating point exception (divide by zero)	Yes	Yes
SIGABRT (6)	Error	Abnormal termination. The JVM raises this signal whenever it detects a JVM fault.	Yes	Yes
SIGINT (2)	Interrupt	Interactive attention (CTRL-C). JVM exits normally.	Yes	No
SIGTERM (15)	Interrupt	Termination request. JVM will exit normally.	Yes	No
SIGHUP (1)	Interrupt	Hang up. JVM exits normally.	Yes	No
SIGQUIT (3)	Control	By default, this triggers a Javadump.	Yes	No
No Name (40)	Control	An AIX® reserved signal. Used by the AIX JVM for internal control purposes.	Yes	No
SIGRECONFIG (58)	Control	Reserved to detect any change in the number of CPUs, processing capacity, or physical memory.	Yes	No
SIGTRAP (5)	Control	Used by the JIT.	Yes	Yes
SIGRTMIN (50)	Control	Used by the JVM for internal control purposes.	No	No
SIGRTMAX (2)	Control	Used by the SDK.	No	No
SIGCHLD (17)	Control	Used by the SDK for internal control.	No	No

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/topic/com.ibm.java.doc.user.aix32.60/user/sighand.html?resultof=%22%53%69%67%6e%61%6c%5f%4e%75%6d%62%65%72%22%20

이에 추가하여 다음으로 확인해 볼 것이 VM flags 입니다. 현재 위에서 보여드린 Javacore 상에서 VM flags:0000000000040000 이기 때문에 하단의 코드 리스트를 확인해보면 JNI component 에서 crash 가 발생된 것으로 확인할 수 있습니다.

Major component	Code number
INTERPRETER	0x10000
GC	0x20000
GROW_STACK	0x30000
JNI	0x40000
JIT_CODEGEN	0x50000
BCVERIFY	0x60000
RTVERIFY	0x70000
SHAREDCLASSES	0x80000

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/topic/com.ibm.java.doc.diagnostics.60/diag/tools/javadump_tags_info.html?resultof=%22%53%69%67%6e%61%6c%5f%4e%75%6d%62%65%72%22%20

뿐만 아니라 문제 시점의 모듈을 같이 출력하여 주므로 해당 문제가 발생한 모듈은 '/usr/lib/libc.a' 으로 유추할 수 있습니다.

이처럼 Javacore 의 앞부분에는 문제가 발생한 원인 정보들이 로깅되어 있기 때문에 지금과 같은 형식으로 기본분석이 가능합니다.

다음으로 Javacore 에서 확인해보는 항목은 ENVINFO 항목입니다. 해당 항목은 JDK 버전과 환경 설정 정보들이 출력됩니다. 특히, JVM 에 대한 사용자 설정정보들이 하단과 같이 나오니 해당 항목을 확인해서 JVM 기본 설정상에 이슈가 있는지의 여부를 확인해 볼 수 있습니다.

```
UserArgs:
          -Xjcl:jclscar_23
          -Dcom.ibm.oti.vm.bootstrap.library.path=/kimjeh/WebSphereV6.1/AppServer/java/jre/bin
          -Dsun.boot.library.path=/kimjeh/WebSphereV6.1/AppServer/java/jre/bin
          -Djava.library.path=/kimjeh/WebSphereV6.1/AppServer/java/jre/bin:/kimjeh/WebSphereV6.
          -Djava.home=/kimjeh/WebSphereV6.1/AppServer/java/jre
          -Djava.ext.dirs=/kimjeh/WebSphereV6.1/AppServer/java/jre/lib/ext
          -Duser.dir=/kimjeh/WebSphereV6.1/AppServer/profiles/juwleeApp01
          _j2se_j9=70912 0x09001000A28B23E8
          vfprintf 0x0000000110001930
          -Declipse.security
          -Dwas.status.socket=63652
          -Dosgi.install.area=/kimjeh/WebSphereV6.1/AppServer
          -Dosgi.configuration.area=/kimjeh/WebSphereV6.1/AppServer/profiles/juwleeApp01/config
          -Djava.awt.headless=true
          -Dosgi.framework.extensions=com.ibm.cds
          -Xshareclasses:name=websphere61_%g,groupAccess,nonFatal
          -Xscmx50M
          -Xbootclasspath/p:/kimjeh/WebSphereV6.1/AppServer/java/jre/lib/ext/ibmorb.jar:/kimjeh
          -Dibm.websphere.internalClassAccessMode=allow
          -verbose:gc
          -Xms256m
          -Xmx512m
```

가장 많이 문제 분석을 위해서 확인하는 항목은 MEMINFO 입니다. Javacore 가 생성된 시점으로 Heap 의 할당된 영역이 얼마이고 free 영역이 얼마인지 확인할 수 있습니다. 이 정보에 따라서

Heap 부족으로 인한 OutOfMemory 발생여부를 확인할 수 있습니다.

```
0SECTION       MEMINFO subcomponent dump routine
NULL           ================================
1STHEAPFREE    Bytes of Heap Space Free: 9367528
1STHEAPALLOC   Bytes of Heap Space Allocated: 10000000
```

참고로 Meminfo 에 나오는 정보는 16 진법으로 표시된 Hex 값이므로 할당된 Heap 영역을 10진수로 변환하면 268,435,456 으로 256MB 인 것을 확인할 수 있습니다.

만약 Heap 의 free 영역이 부족해서 OutOfMemory 가 발생했다면 해당 Javacore 의 GC History 항목에서 직전의 GC 정보를 확인하여 큰 객체를 할당하다가 OOM 이 발생하는지 작은 객체들을 할당함에도 불구하고 Heap 이 모자라서 OOM 이 발생했는지를 확인할 수 있습니다.

그 다음으로 확인하는 것은 Locks 항목으로 JVM 의 lock 과 Monitor 정보들을 출력해 줍니다. 보통 다음과 같이 어떤 Thread 가 현재 monitor 를 가지고 있으며 해당 monitor 를 받기 위해서 대기 중인 Thread 들의 리스트를 하단처럼 확인할 수 있습니다.

```
2LKMONINUSE     sys_mon_t:0x5CF12B30 infl_mon_t: 0x5CF12B70:
3LKMONOBJECT      org/apache/commons/pool/impl/GenericObjectPool@18B8F378/18B8F384: owner "WebContainer : 288" (0x68CD7B00), entry count 1
3LKWAITERQ          Waiting to enter:
3LKWAITER             "WebContainer : 251" (0x000D8D00)
3LKWAITER             "WebContainer : 340" (0x000D9100)
3LKWAITER             "WebContainer : 325" (0x61270200)
3LKWAITER             "WebContainer : 169" (0x61270600)
3LKWAITER             "WebContainer : 209" (0x61270A00)
3LKWAITER             "WebContainer : 326" (0x60465900)
3LKWAITER             "WebContainer : 253" (0x61797600)
3LKWAITER             "WebContainer : 157" (0x61BFA500)
3LKWAITER             "WebContainer : 212" (0x61BFA900)
3LKWAITER             "WebContainer : 257" (0x63E3DD00)
3LKWAITER             "WebContainer : 308" (0x62020100)
3LKWAITER             "WebContainer : 324" (0x62020500)
3LKWAITER             "WebContainer : 198" (0x6598A900)
3LKWAITER             "WebContainer : 327" (0x6598B500)
3LKWAITER             "WebContainer : 292" (0x6598B900)
3LKWAITER             "WebContainer : 311" (0x66882200)
3LKWAITER             "WebContainer : 328" (0x65BC9E00)
3LKWAITER             "WebContainer : 342" (0x65BCA600)
```

따라서 이를 통해 Dead lock 이라던지, monitor 를 제대로 반환하지 않아서 monitor 를 가지기 위해 긴 시간 대기하는 현상과 같은 시스템 지연 문제를 확인할 수 있습니다.

다음으로는 Javacore 에서 가장 많이 확인하면서 javacore 를 생성해서 확인하고자 하는 근본 목적인 Thread 항목입니다. 초기에 말씀 드린 것처럼 Thread 항목에는 생성 시점의 Thread 의 snapshot 을 찍어서 Text 형태로 만들어 둔 것입니다. 따라서, 각 Thread 마다 상태가 있고 상태 정보의 정확한 의미는 하단과 같습니다.

- R - Runnable - the thread is able to run when given the chance.
- CW - Condition Wait - the thread is waiting. For example, because:
  - A sleep() call is made
  - The thread has been blocked for I/O
  - A synchronized method of an object locked by another thread has been called
  - The thread is synchronizing with another thread with a join() call
- S - Suspended - the thread has been suspended by another thread.
- Z - Zombie - the thread has been killed.
- P - Parked - the thread has been parked by the new concurrency API (java.util.concurrent).
- B - Blocked - the thread is waiting to obtain a lock that something else currently owns.

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp?topic=/com.ibm.java.doc.diagnostics.60/html/javadump_interpret.html

예를 들어 하단의 Thread 는 WebContainer : 339 라는 Thread 이며 javacore 생성 당시에 상태는 R 이므로 Runnable 입니다. 즉 javacore 생성 시점에 수행중인 thread 라는 의미입니다.

```
"WebContainer : 339" J9VMThread:0x0000000033271900, j9thread_t:0x0000000121B2DA20, java/lang/Thread:0x00000000498E3760, state:R, prio=5
      (native thread ID:0x2A50285, native priority:0x5, native policy:UNKNOWN)
     Java callstack:
         at com/xxxxxxxxx/JniSearcher.GetColumn(Native Method)
         at com/xxxxxxxxx/JniSearcher.GetColumn(Bytecode PC:7)
         at com/xxxxxxxx/search/action/OzxxxxxAction.getOzxxxxBandi(Bytecode PC:1133)
         at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method)
         at sun/reflect/NativeMethodAccessorImpl.invoke(Bytecode PC:83(Compiled Code))
         at sun/reflect/DelegatingMethodAccessorImpl.invoke(Bytecode PC:6(Compiled Code))
         at java/lang/reflect/Method.invoke(Bytecode PC:6(Compiled Code)) [EOF]
```

Thread 항목에 대한 구분이 가능하게 되면 맨 먼저 살펴 보아야 할 것은 Current thread 입니다. Current thread 는 말 그대로 Javacore 생성 시점에 돌고 있던 thread 이므로 만약 thread 때문에 문제가 발생했다면 해당 thread 에서 발생될 확률이 가장 높습니다. 또한, 반복적인 문제상황이 발생해서 여러 개의 Javacore 를 확인했는데 Current thread 가 모두 같은 Thread 를 지칭하고 있다면 역시나 문제는 해당 thread 에서 기인했을 가능성이 높습니다. Current thread 항목을 확인한 후에 전체 Thread 리스트를 확인하면서 불필요하게 수행되고 있거나 너무 많은 중복 등 문제를 야기할 만한 Thread 가 있는지 확인을 해야 합니다.

마지막으로 Classes 항목정보가 나오는데 이 정보는 classloader 에 실질적으로 로드된 class 의 세부정보입니다. 하단과 같이 어떤 classloader 가 몇 개의 library 와 class 를 실제적으로 load 하고 있는지에 대한 내용이 표시됩니다.

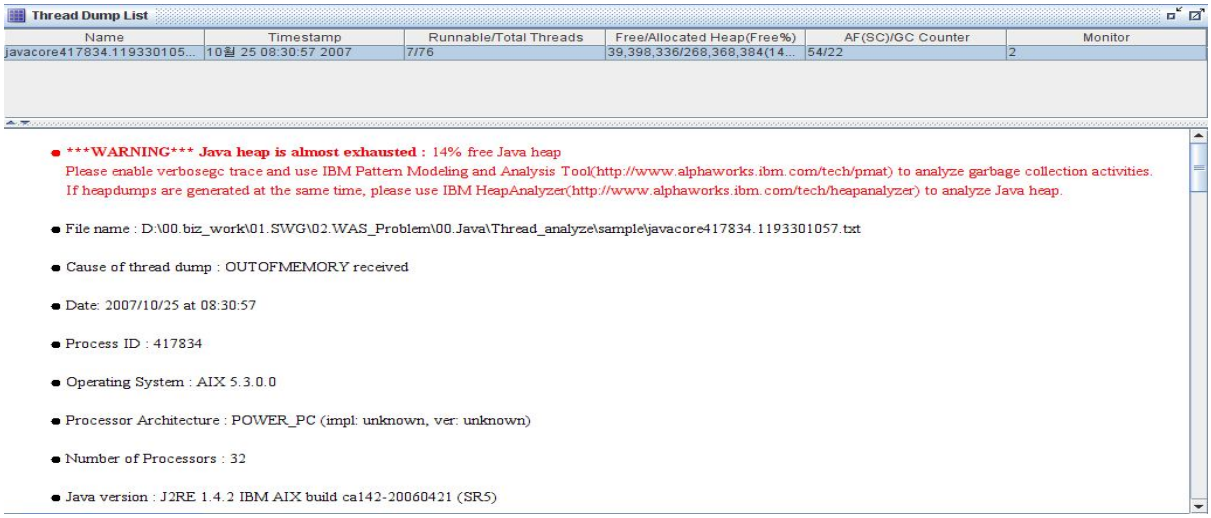
```
1CLTEXTCLLOS    Classloader summaries
1CLTEXTCLLSS      12345678: 1=primordial,2=extension,3=shareable,4=middleware,5=system,6=trusted,7=application,8=delegating
2CLTEXTCLLOADER   p---st-- Loader *System*(0x00EED1B8)
3CLNMBRLOADEDLIB    Number of loaded libraries 5
3CLNMBRLOADEDCL     Number of loaded classes 3566
2CLTEXTCLLOADER   -x--st-- Loader sun/misc/Launcher$ExtClassLoader(0x00ECD0B0), Parent *none*(0x00000000)
3CLNMBRLOADEDLIB    Number of loaded libraries 0
3CLNMBRLOADEDCL     Number of loaded classes 85
2CLTEXTCLLOADER   -----ta- Loader sun/misc/Launcher$AppClassLoader(0x00EED448), Parent sun/misc/Launcher$ExtClassLoader(0x00ECD0B0)
3CLNMBRLOADEDLIB    Number of loaded libraries 0
3CLNMBRLOADEDCL     Number of loaded classes 240
```

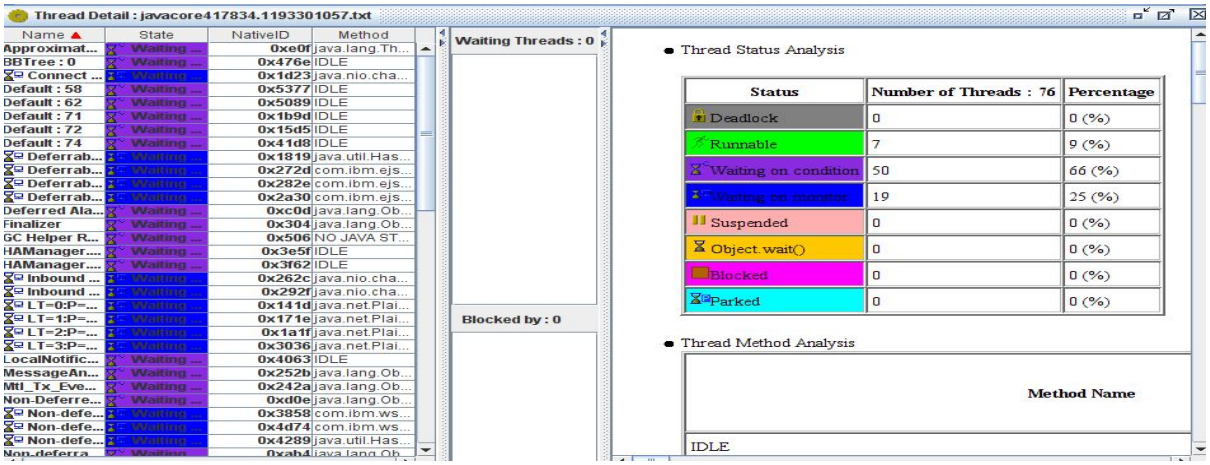
Classloader 요약 항목 다음에는 실제적으로 해당 classloader 에 load 된 class 항목을 하단처럼 확인할 수 있습니다.

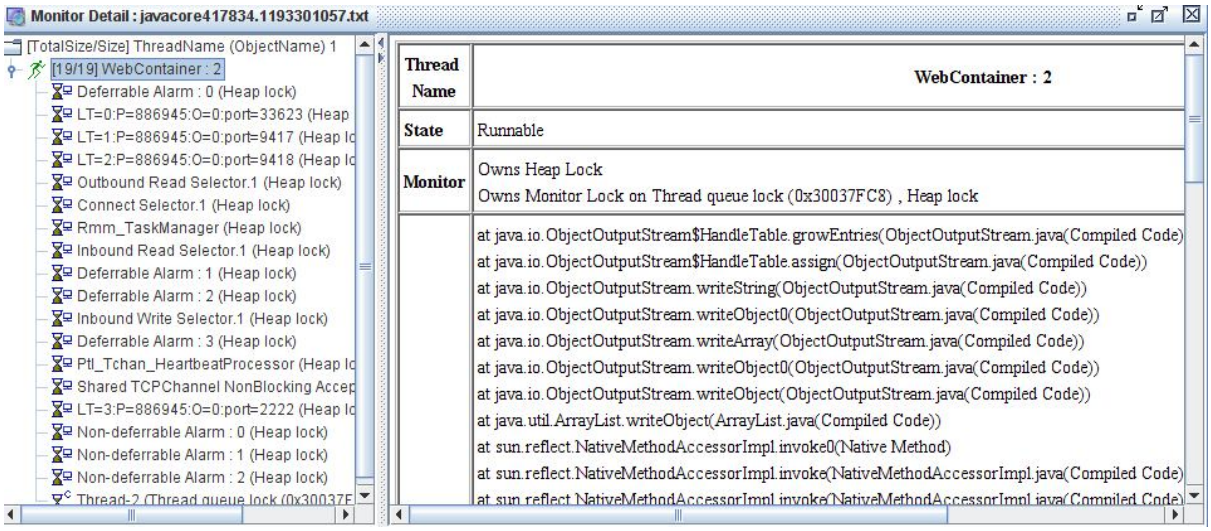
```
1CLTEXTCLLOD    ClassLoader loaded classes
2CLTEXTCLLOAD     Loader *System*(0x00EED1B8)
3CLTEXTCLASS        sun/reflect/UnsafeQualifiedBooleanFieldAccessorImpl(0x65D5F080)
3CLTEXTCLASS        sun/reflect/UnsafeStaticIntegerFieldAccessorImpl(0x65D5F3D8)
3CLTEXTCLASS        sun/reflect/UnsafeQualifiedStaticBooleanFieldAccessorImpl(0x65D5F768)
3CLTEXTCLASS        sun/net/www/http/KeepAliveCache$1(0x65D5FA78)
3CLTEXTCLASS        sun/net/www/http/KeepAliveEntry(0x65D5FC10)
3CLTEXTCLASS        java/util/logging/Filter(0x65D5FCF8)
3CLTEXTCLASS        [[[D(0x65D5FDC0)
3CLTEXTCLASS        sun/io/MalformedInputException(0x65D5FEA0)
3CLTEXTCLASS        sun/io/UnknownCharacterException(0x65D60000)
3CLTEXTCLASS        sun/text/resources/LocaleData$2(0x65D60138)
3CLTEXTCLASS        org/apache/xerces/dom/DeferredElementDefinitionImpl(0x65D603C0)
3CLTEXTCLASS        java/sql/BatchUpdateException(0x65D60760)
3CLTEXTCLASS        sun/io/CharToByteDBCS_ASCII(0x65D60998)
3CLTEXTCLASS        sun/io/CharToByteCp970(0x65D60E00)
3CLTEXTCLASS        com/ibm/jsse2/eb(0x65D610B8)
3CLTEXTCLASS        javax/net/ssl/SSLProtocolException(0x65D61E08)
3CLTEXTCLASS        javax/net/ssl/SSLHandshakeException(0x65D61F58)
3CLTEXTCLASS        com/ibm/jsse2/z(0x65D62098)
3CLTEXTCLASS        com/ibm/jsse2/z$d_(0x65D623F8)
3CLTEXTCLASS        com/ibm/jsse2/rb(0x65D627D8)
3CLTEXTCLASS        com/ibm/jsse2/z$g_(0x65D62A78)
3CLTEXTCLASS        com/ibm/jsse2/z$a_(0x65D62E40)
3CLTEXTCLASS        com/ibm/jsse2/z$h_(0x65D63218)
3CLTEXTCLASS        com/ibm/jsse2/cb(0x65D63418)
```

따라서 해당 항목은 Memory leak 문제가 발생했을 경우 불필요하게 중복 load 된 class 들이 없는지 분석하거나 해결할 때 주로 찾아보는 항목입니다.

지금까진 Javacore 를 Text 파일 그대로 확인해서 분석했는데 이는 약간의 구식 분석 방법이며 IBM Thread and Monitor Dump Analyzer for java 와 같은 무료 툴을 사용하면 좀 더 쉽게 분석을 할 수 있게 하단처럼 내용을 정리해서 그래픽을 이용하여 직관적으로 보여줍니다.







따라서, 이러한 툴을 활용하면 보다 쉽게 javacore 를 분석할 수 있습니다.

Part 3. 추가 분석 - heapdump

이번 파트에서 분석을 할 때 사용해볼 덤프는 Heapdump 입니다. Heapdump 는 Javacore 와 비슷한 snapshot 정보이지만 Thread 가 아니라 해당 시점의 Heap 영역의 snapshot 정보를 담은 덤프입니다. 아시겠지만 JVM 은 거의 모든 정보를 heap 영역에 두고 활용합니다. 따라서 해당 정보를 잘 분석하게 되면 이전 단계에서 해결하지 못한 문제도 해결할 수 있는 실마리를 찾을 수 있습니다. (보통은 Memory leak 과 같은 문제를 해결하기 위해서 많이 분석합니다.) Heapdump 는 text 형태로 되어 있지 않기 때문에 Text 로 직접 분석할 수는 없고 IBM Heap analyzer 와 같은 무료 분석 툴을 활용하여 분석합니다.

Heapdump 를 생성하려면 하단과 같은 옵션을 사용한 상태에서 signal(kill -3 과 같은) 을 주거나 javacore 와 같이 스크립트를 이용해서 생성할 수 있습니다.

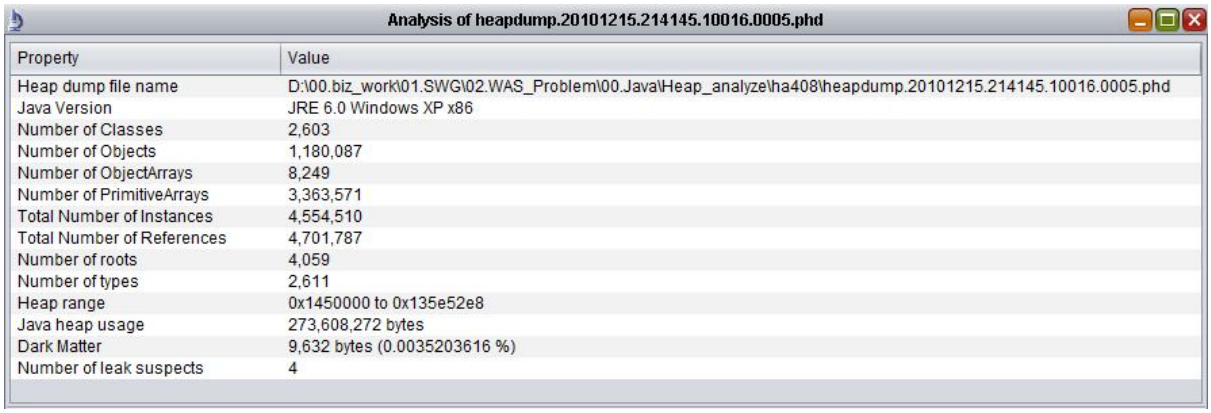
Environment Variable	Usage Information
IBM_HEAPDUMP IBM_HEAP_DUMP	Setting either of these to any value (such as true) enables heap dump production by means of signals.
IBM_HEAPDUMPDIR	The default location into which the Heapdump will be written.
JAVA_DUMP_OPTS	Use this environment variable to control the conditions under which Heapdumps (and other dumps) are produced. See Dump agent environment variables for more information .
IBM_HEAPDUMP_OUTOFMEMORY	By setting this environment variable to false, you disable Heapdumps for an OutOfMemory condition.
IBM_JAVA_HEAPDUMP_TEST	Use this environment variable to cause the JVM to generate both phd and text versions of Heapdumps. Equivalent to **opts=PHD+CLASSIC** on the **-Xdump:heap** option.
IBM_JAVA_HEAPDUMP_TEXT	Use this environment variable to cause the JVM to generate a text (human readable) Heapdump. Equivalent to **opts=CLASSIC** on the **-Xdump:heap** option.

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp?topic=/com.ibm.java.doc.diagnostics.60/diag/tools/heapdump_env.html

스크립트를 이용하여 Heapdump 강제로 생성하기(Jacl)

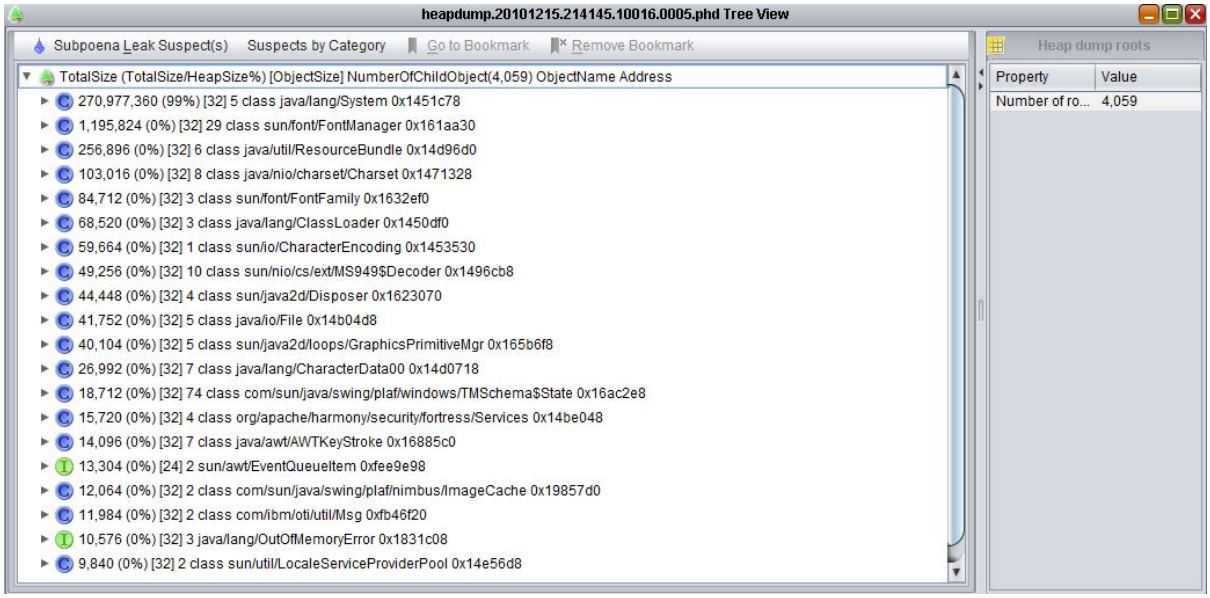
```
set objectName [$AdminControl queryNames WebSphere:type=JVM,process=server1,node=pcrea01Node02,*]
$AdminControl invoke $objectName generateHeapDump
```

Heapdump 를 분석하려면 이전에 언급드린 것처럼 IBM Heap analyzer 를 실행시킨 후에 분석하고자 하는 Heapdump 파일을 불러옵니다. 그러면 하단과 같이 heap 안의 class 의 개수 object 의 개수등과 같은 기본 정보들이 보여집니다.



Property	Value
Heap dump file name	D:\00.biz_work\01.SWG\02.WAS_Problem\00.Java\Heap_analyze\ha408\heapdump.20101215.214145.10016.0005.phd
Java Version	JRE 6.0 Windows XP x86
Number of Classes	2,603
Number of Objects	1,180,087
Number of ObjectArrays	8,249
Number of PrimitiveArrays	3,363,571
Total Number of Instances	4,554,510
Total Number of References	4,701,787
Number of roots	4,059
Number of types	2,611
Heap range	0x1450000 to 0x135e52e8
Java heap usage	273,608,272 bytes
Dark Matter	9,632 bytes (0.0035203616 %)
Number of leak suspects	4

메뉴에서 Analysis > Tree view 를 선택하여 Tree 형태로 되어 있는 heap 안의 참조관계 트리를 확인합니다.

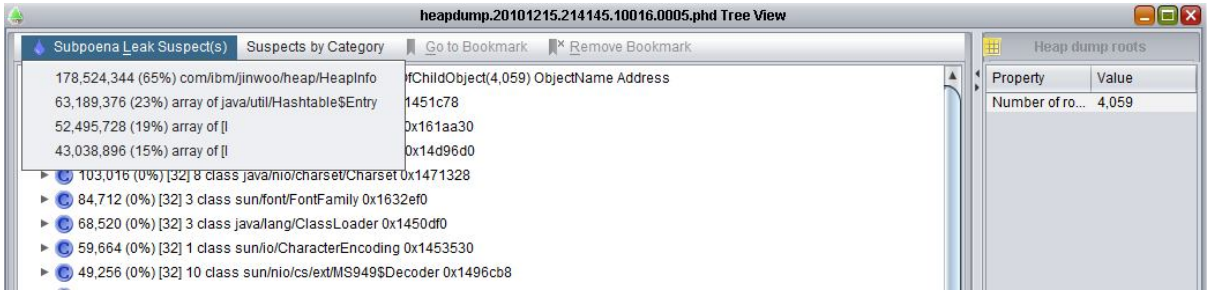


여기서 각각의 아이콘은 다음과 같이 class, instance, array 를 의미합니다.

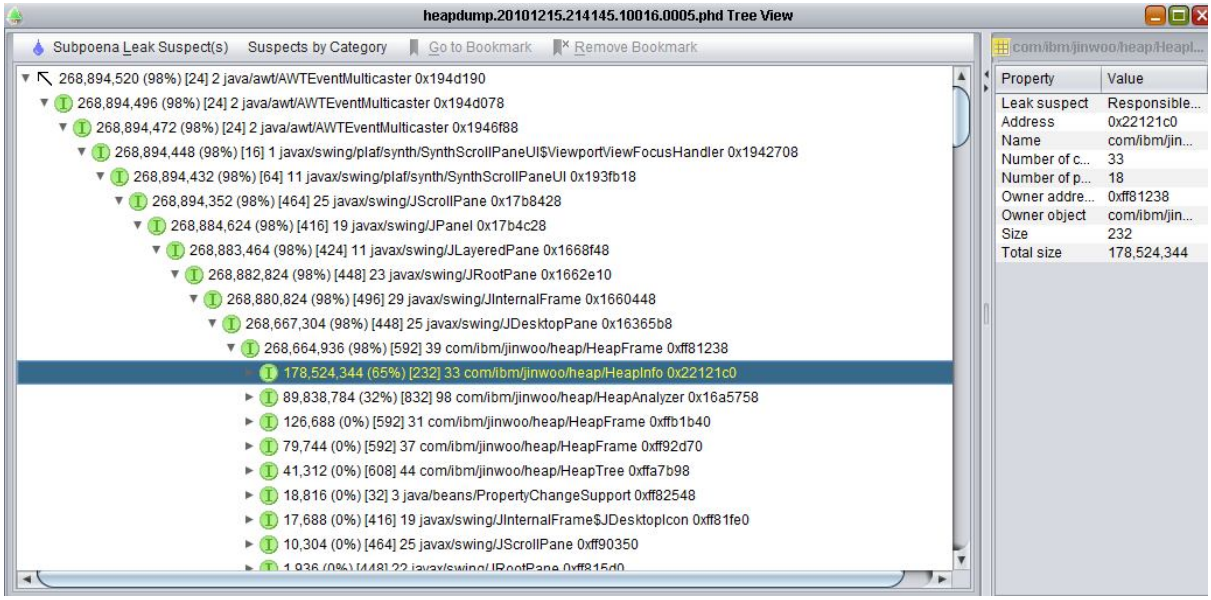
그리고 각 항목은 다음과 같은 구조로 되어 있으므로 각 내용은 하단의 구조를 이용해서 해석이 가능합니다.

TotalSize(TotalSize/HeapSize%)[ObjectSize] NumberOfChildObject(Number of root objects) Name Address

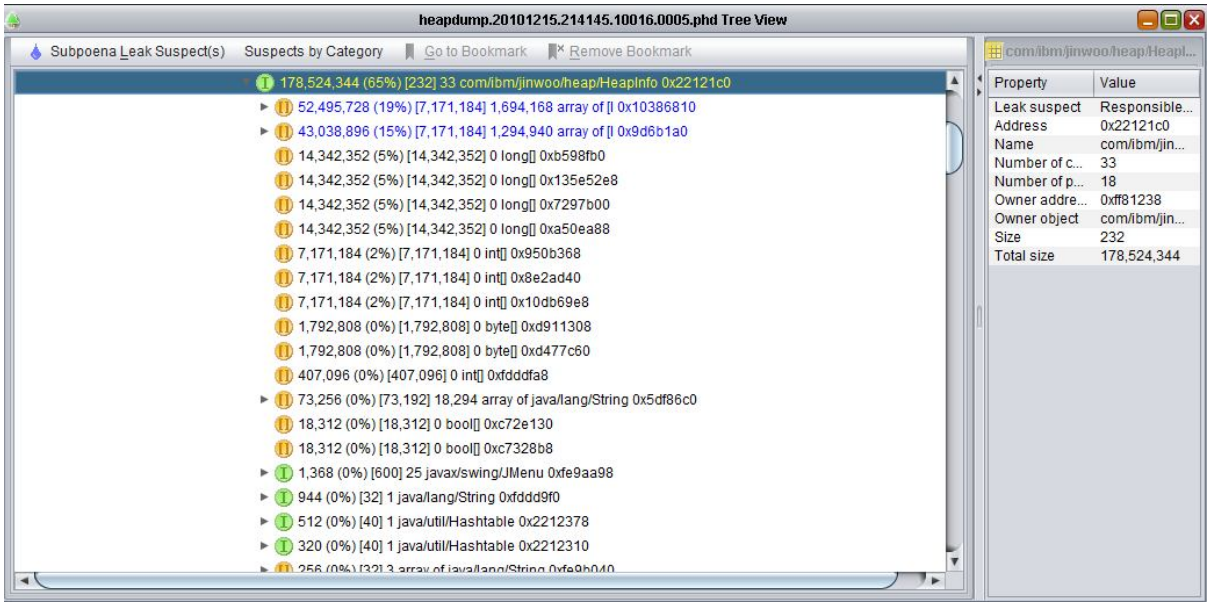
Heap 안의 참조관계 리스트에서 leak suspect 메뉴를 선택하면 프로그램적으로 툴에서 의심되는 leak 용의자들을 찾아서 보여줍니다.



이중에 하나를 클릭하면 다음과 같이 직접 해당하는 객체의 위치로 이동됩니다.



좀 더 자세하게 분석하려면 해당 객체를 클릭하면 그 하단의 참조관계를 계속적으로 보여줍니다.



보시는 것처럼 해당 heapdump 분석에서 현재 instance 밑에 7MB 짜리 int[] 라던가 14MB 짜리 long[] 이 여러 개 달려서 178MB 를 소모하고 있기 때문에 이 부분을 leak 이라고 의심하고 포인팅 한 것입니다. 이게 만약 실제 상황이라면 지칭된 instance 가 실제 호출되는 소스를 찾아서 실제로 int[] 나 long[] 이 이렇게 많이 호출되고 정상적인지, 아니면 GC 가 제대로 안되어서 불필요하게 leak 이 발생하는지 확인해 볼 수 있습니다.

위와 같은 방식으로 각 class, instance, array 의 참조 관계 트리를 따라가 보면서 Heap 영역 안에 어떤 class 나 instance 가 얼마의 사이즈로 얼마나 많은 개수가 담겨져 있는가를 확인 및 분석할 수 있습니다. 이를 통하여 Heap 영역을 많이 점유하는 instance 를 찾아내기도 하고 Memory leak 부분을 찾을 수 도 있습니다. (다만, 제 생각으로 이 부분의 분석은 약간의 반복작업과 노하우가 조금 필요합니다.^^&)

참고로 Javacore 분석에서도 이야기 했는데 heapdump 는 해당 시점의 snapshot 이기 때문에 순간적인 정보만이 남아있습니다. 따라서, 만약 Memory leak 과 같은 점진적인 문제를 분석하고 싶으시다면 시간차를 두고 여러 개의 heapdump 를 생성하여 실제적으로 늘어나는 class 나 instance 를 찾는 방법을 활용하셔야 합니다.

Part 4. Trace 분석

마지막 파트는 Trace 분석이라는 이름을 달아놨지만 실질적으로 사용자가 직접 분석하기 보다는 보통 IBM Lab 에 분석을 요청하기 위한 Trace 를 받는 작업이 될 듯 합니다. 그러나 시간이 허락하신다면 해당 trace 를 분석해 보면 실제적으로 보기 어려운 내부 component 의 동작 방식 뿐만 아니라 IBM Lab 의 답변 전에도 문제 해결의 실마리를 찾을 수 있습니다.

우선적으로 장애 상황별로 분석해야할 Trace 가 다르기 때문에 하단의 링크를 통해서 큰 범주의 장애 상황과 연관된 Trace 를 확인합니다.

http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21145599

MustGather document:

- Administrative Console (all non-scripting)
- Administrative Scripting Tools (for example: wsadmin or ANT)
- Application Client
- Application Server Toolkit
- Classloader
- Crash
- Data Replication Services
- DB Connections/Connection Pooling
- Deploy (for example: AAT or ANT or EAR/WAR/JAR)
- Double Byte Character Set (DBCS)
- Dynamic Cache
- Edge Component
- EJB Container
- Feature pack for Service Component Architecture (SCA)
- General
- High Availability (HA)
- HTTP Transport
- IBM HTTP Server
- Install
- Java Management Extensions (JMX) or JMX client API
- Java Message Service (JMS)
- Java Security (JSSE/JCE)
- Java Transaction Service (JTS)
- Java SDK
- JavaMail
- JavaServer Faces (JSF)
- JavaServer Pages (JSP)
- JNDI/Naming
- Migration
- Object Level Trace/Distributed Debugger (OLT/DD)
- Object Request Broker (ORB)
- Out of Memory
- PD tools (for example Log Analyzer)
- Performance, hang, or high CPU
- Plug-in
- PMI/Performance Tools
- Programming Model Extensions (PME)
- Proxy server

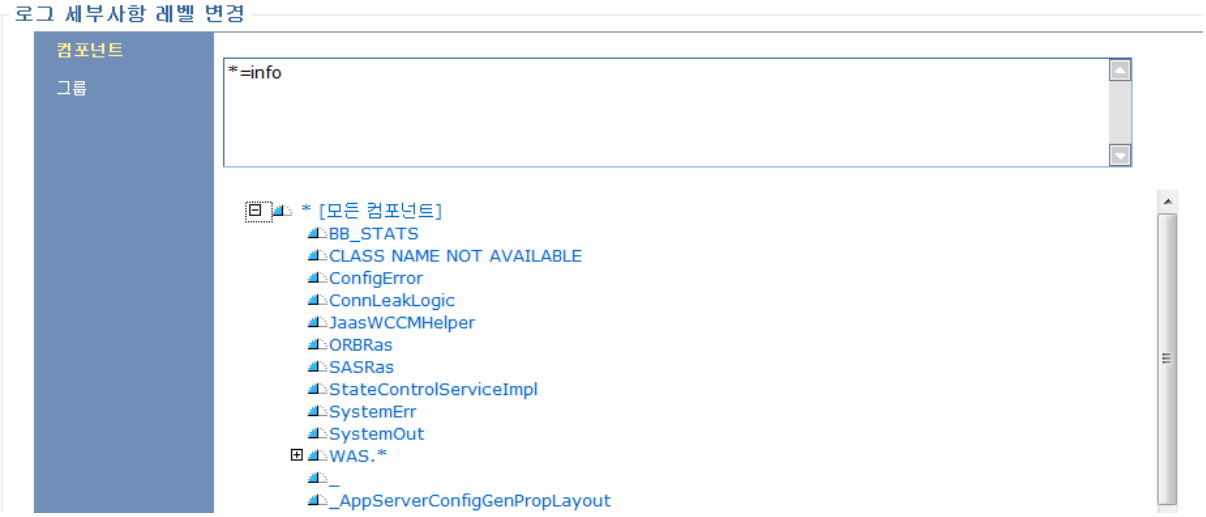
관련 범주의 해당 링크로 들어가서 collecting data manually 에 나와있는 지시대로 로그 수집 및 Trace 세팅을 하면 됩니다. (하단의 내용은 예제)

2. Enable and collect Application Server traces with the following trace string:

```
com.ibm.ws.classloader.*=all
```

For more details, see How to set up a trace.

마지막으로 Trace 생성하는 방법을 좀 보여드리면, 관리콘솔에서 Server > 해당 서버명 > 로깅 및 추적 > 로그 세부사항 레벨 변경 메뉴에서 Runtime 탭에 해당 Trace 세팅을 입력하면 실시간으로 필요한 Trace 를 생성할 수 있습니다. (해당 Trace 정보는 trace.log 파일에 생성됩니다.)



참고로 생성된 Trace 인 Trace.log 파일은 하단과 같은 Text 형태로서 직접 확인이 가능합니다.

```
[08. 12. 24   12:27:31:983 KST] 0000000a ODCProperties 3   oldPriority=-1 newPriority=0
[08. 12. 24   12:27:31:983 KST] 0000000a ODCNodeImpl   3   property changed: true
[08. 12. 24   12:27:31:983 KST] 0000000a ODCTreeImpl   3   in transaction --> TreeBuilder.init, local=true, genNo=2
[08. 12. 24   12:27:31:983 KST] 0000000a ODCTreeImpl   3   top=TreeBuilder.init, local=true, genNo=2
[08. 12. 24   12:27:31:983 KST] 0000000a ODCNodeImpl   <  setProperty Exit
[08. 12. 24   12:27:31:983 KST] 0000000a TreeBuilder   3   custom property: name=MOP.MaxZoneFactor is removed
[08. 12. 24   12:27:31:983 KST] 0000000a TreeBuilder   3   custom property: name=MOP.closeToMaxFraction is removed
[08. 12. 24   12:27:31:983 KST] 0000000a ODCNodeImpl   3   setProperty: local=true, desc=cell:MOP.closeToMaxFraction, value=null
[08. 12. 24   12:27:31:983 KST] 0000000a ODCTreeImpl   3   top=TreeBuilder.init, local=true, genNo=2
[08. 12. 24   12:27:31:983 KST] 0000000a ODCNodeImpl   >  setProperty Entry
                                setProperty MOP.closeToMaxFraction on /cell/smbia01Cell01 /cell/smbia01Cell01, oldVal=null, newVal=null
[08. 12. 24   12:27:31:983 KST] 0000000a ODCProperties 3   oldPriority=-1 newPriority=0
[08. 12. 24   12:27:31:983 KST] 0000000a ODCNodeImpl   3   property changed: true
[08. 12. 24   12:27:31:983 KST] 0000000a ODCTreeImpl   3   in transaction --> TreeBuilder.init, local=true, genNo=2
[08. 12. 24   12:27:31:984 KST] 0000000a ODCTreeImpl   3   top=TreeBuilder.init, local=true, genNo=2
[08. 12. 24   12:27:31:984 KST] 0000000a ODCNodeImpl   <  setProperty Exit
[08. 12. 24   12:27:31:984 KST] 0000000a TreeBuilder   3   custom property: name=MOP.closeToMaxFraction is removed
[08. 12. 24   12:27:31:984 KST] 0000000a TreeBuilder   3   custom property: name=profilerGoodnessBlend0 is removed
[08. 12. 24   12:27:31:984 KST] 0000000a ODCNodeImpl   3   setProperty: local=true, desc=cell:profilerGoodnessBlend0, value=null
[08. 12. 24   12:27:31:984 KST] 0000000a ODCTreeImpl   3   top=TreeBuilder.init, local=true, genNo=2
[08. 12. 24   12:27:31:984 KST] 0000000a ODCNodeImpl   >  setProperty Entry
                                setProperty profilerGoodnessBlend0 on /cell/smbia01Cell01 /cell/smbia01Cell01, oldVal=null, newVal=null
```

여기까지 해서 하나씩 쉽게 따라 해보는 IBM WAS v7 의 마지막 강좌인 WebSphere 에서 문제상황과 분석(장애처리) 에 대한 기본 정리가 된 것 같습니다. 조금 어려울 수 있지만 한번 따라서 해보시고 익숙해 지시면 그리 어렵지 않다는 것을 이해할 수 있으실 것입니다. 실제 운영환경에서는 장애처리 부분은 벤더사나 협력업체에서 담당자가 직접 방문해서 해결해주는 경우가 많지만 그래도 IBM WAS 를 운영하는 사람에게는 기본적으로 숙지하고 이해해야할 내용이라고 생각합니다. 그렇게 하다보면 보다 빠른 장애 처리뿐만 아니라 시스템의 안정적인 운영 측면에서도 더 많은 것을 얻으실 수 있으실 것입니다. 물론 IBM WAS 나 JVM 의 이해를 더 한층 높일 수 도 있습니다.

마무리

하다보니 어느새 3년이 걸렸네요. 이제사 미흡하지만 3부작 26편으로 구성된 하나씩 쉽게 따라 해보는 IBM WAS v7 강좌의 마무리를 짓습니다. 업무외 시간을 이용해서 만들다 보니 오랜시간이 걸리긴 했지만 저도 만들면서 많은 것을 배우고 얻게 된 의미있는 강좌였던 것 같습니다. 제가 받았던 좋은 경험처럼 이 강의를 보시는 모든 분들이 WebSphere 전문가가 되면 좋겠다는 바램을 가지면서 이만 진짜 마무리 하도록 하겠습니다. 모두 행복하세요... ^^&;

참고 1) IBM WebSphere Application Server v7.0 InfoCenter
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/welcome_nd.html

참고 2) IBM JAVA v6.0 InfoCenter
http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp

참고 3) MustGather: Read first for WebSphere Application Server
http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21145599