JBoss AS Tuning & Slimming

by horanghi@gmail.com

목 차

2.13. Web-console과 jsr-177 extensions을 사용하지 않는 경우

2.14. 콘솔과 이메일 모니터 알림을 사용하지 않는 경우

2.15. JMX 의 rich property editors 를 이용하여 System properties 를 수정하지 않는 경우

2.16. 스케줄 서비스가 예제로 올라가있는데 필요없는 경우

2.17. JBoss 스케줄 매니너를 이용하지 않는 경우

2.18. 클라이언트사이드 트렌젝션관리또는 캐싱된 connections 을 이용하지 않는 경우

2.19. JBoss UUID 키 generation 을 이용하지 않는 경우 (종종 CMP 에서 사용)

2.20. HTTP를 통해 톰캣을 바로 이용하는 경우(mod_jk를 이용하지 않는 경우)

2.21. Mod_jk를 이용하여 접근하는 경우(직접 접근이 없이)

2.22. JMS Queues를 이용하지 않는 경우

2.23. CORBA/IIOP 를 이용하지 않는 경우

2.24. user-transaction-service.xml을 제거하려고 한다면

2.25. JSR-77 을 필요 없다면

2.26. 클라이언트 사이드 트렌젝션 관리를 하지 않는 다면 ₩

2.27. persistent MBean attributes 가 필요하지 않다면

2.28. RMI 클래스로딩을 이용하지 않는 경우 (클라이언트에서 서버로 codebases를 에로딩하지 않는 경우)

2.29. 로컬 JBoss 네이밍을 원한다면 (RMI 클라이언트 없이)

2.30. Thread pool을 제거하려면

2.31. BeanShell을 사용을 원하지 않는다면

2.32. JBoss 리스타트 없이 deploy 안의 파일을 hot deploy 하지 않으려면

1. Tunning Factor

1.1. Java Virtual Machine

VM 가비지 컬렉션을 튜닝 하거나 JDK 5 가비지 컬렉션을 튜닝 하여라.

x86 하드웨어인경우는 JRockit 을 이용해라.

64비트 하드웨어와 64비트 VM을 사용하는 것이 좋다.

그러면 heap size를 크게 잡을 수 있다. 2-4GB보다 크게 잡을 수 있다.

64비트를 최근 Sparc/Solaris 박스(Solaris 9 or later, Itanium with JDK 1.4,

or JDK 5 on Linux x64)에서도 가능하다.

64비트가 아닌경우는 최대32 비스 heap space를 넘기면 안된다.(2-4GB heap)

넘기게 되는 경우 오바된 heap 사이즈로 인해 가비지 컬렉션을 위한 메모리

스캔시간이 소요된다.

되도록이면 Sun 1.4 VM은 피해라. JDK 5의 경우 가비지컬렉션의 성능이 월등하게 좋아졌다. JRockit의 경우는 x86에서 뛰어나다.

-server 옵션을 이용해라. 하지만 -XX:ThreadStackSize=128k (Solaris) 이나

-Xss128k (every other platform) 중에 하나만 이용해라.

Solaris에서는 -Xss128k 을 하더라도 효과가 없다.

단지 thread의 스택 사이즈만 크게해 주는 것이 전부이다. 이렇게 함으로써 메모리사용이 적은 thread의 갯수를 늘릴 수 있다.

하지만 recurisive code로 인해 스택의 사용이 늘어나게 된다.

가비지 컬렉션의 이해가 필요하다면 load testing(JMeter)를 이용하여 해보면 된다.

두개이상의 멀티 프로세서를 이용한다면 parralled, concurrent 가비지컬렉션을 이용하라. JDK5에서는 대개 self-Tunning 이 되고 있다.

1.2. Tomcat

Tomcat 서버를 가볍게 하기위해서는 **jbossweb-tomcat5?.sar/server.xml** 를

수정하여라.

xml 문서의 이용되고 있는 connectors를 확인해보는 것이 좋다.

예를 들자면

```
<Connector port="8080" address="${jboss.bind.address}"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8443" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true"/>
```

충분하게 thread를 정했는지 (maxThreads) 예상되는 동시 사용 hit 갯수의 25%정도

더 많이 지정하는 것이 좋다.

보통 운영시 사용되는 thread의 갯수를 minSpareThreads 로 정하고

피크시점의 thread의 갯수를 maxSpareThreads 로 정한다.

minSpareThreads 는 서버가 스타트 되는 시점에서 waiting 상태로 기다리는 thread의 갯수를 뜻한다.

maxSpareThreads 는 minSpareThreads 갯수 이상으로 이용되는 경우 유지되는 thread 의 갯수라고 말할 수 있다.

불필요한 valves와 logging은 제거하는 것이 좋다. 또한 JBoss security를 사용하지 않는다면 security valve도 지운다.

JSP의 경우는 precompile해두는 것이 좋다.

/jbossweb-tomcat50.sar/conf/web.xml 의 개발모드는 끄는 것이 좋다.

1.3. RMI for Remote Invocations

디볼트 에서는 JBoss는 호출된 RMI 에 대해 새로운 thread를 만든다. 이러한 경우 효율성이 떨어지게 된다. 제한없이 connection을 생성하는 경우 성능에 상당한 저하를 보일 수 있다. 따라서 pooled invoker 에 스위칭하는 형태로 고려해 보아야 한다.

con/standardjboss.xml 를 수정하여라.

```
<invoker-mbean>jboss:service=invoker,type=jrmp</invoker-mbean>
```

에서

```
<invoker-mbean>jboss:service=invoker,type=pooled</invoker-mbean>
```

으로 수정하라.

1.4. PooledInvokerHA

JBoss는 pooled invoker 의 HA 버젼을 가진다. 이것은 **cluser-service.xml** 에서 정의된다.

사용하기위해서는 아래 태그를 server/all/deploy/cluster-service.xml 에 넣으면 된다.

```
<mbean code="org.jboss.invocation.pooled.server.PooledInvokerHA"
name="jboss:service=invoker,type=pooledha">
<attribute name="NumAcceptThreads">1</attribute>
<attribute name="MaxPoolSize">300</attribute>
<attribute name="ClientMaxPoolSize">300</attribute>
<attribute name="SocketTimeout">60000</attribute>
<attribute name="ServerBindAddress">${jboss.bind.address}</attribute>
<attribute name="ServerBindPort">4446</attribute>
<attribute name="ClientConnectAddress">${jboss.bind.address}</attribute>
<attribute name="ClientConnectPort">0</attribute>
<attribute name="EnableTcpNoDelay">false</attribute>

<depends optional-attribute- name="TransactionManagerService">

jboss:service=TransactionManager</depends>
</mbean>
```

그리고 나서 **server/all/conf/standardjboss.xml**를 수정한다.

jboss:service=invoker,type=jrmpha 를 jboss:service=invoker,type=pooledha 로

일반적으로 connect되는 thread의 갯수에 대해 고려해 보아야 한다.

1.5. Log4j

로그를 남기는 작업은 성능상에 상당한 영향을 준다. 따라서 JBossAS에 세팅되는 TRACE 로깅 레벨에 따라 상당히 느릴 수 있다. 간단하게 예를 들자면 레벨을 ERROR나 WARN으로 바꾸기만해도 속도가 눈에 띄게 빨라시는 것을 볼 수 있다.

JBoss 로그는 디볼트 상태로 INFO 로 정의 되어있다.(console 과 server.log) System.out을 쓰는 로깅은 되도록이면 자제하도록 해야 한다. 또한 로그레벨을 ERROR 로 쓰는 것을 고려해 볼 필요가 있다. JBoss는 운영중에 log4j 의 환경파일의 값을 바꾸어줌 으로써 적용이 가능하다. 또한 카테고리별로 따로 적용이 가능하다.

성능을 위해서는 console 로그를 끄는 것이 좋다. 끄기 위해서는 **/conf/log4j.xml** 의 파일을 수정하여라.

아래의 XML 태그를

```
<root>
<appender-ref ref=CONSOLE"/>
<appender-ref ref="FILE"/>
```

```
</root>
```

에서

```
<root>
<appender-ref ref="FILE"/>
</root>
```

으로 수정하면 된다.

또한 아래와 같은 태그를 제거하라.

```
<appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
<errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
<param name="Target" value="System.out"/>
<param name="Threshold" value="INFO"/>
<layout class="org.apache.log4j.PatternLayout">
<!-- The default pattern: Date Priority [Category] Message₩n -->
<param name="ConversionPattern" value="%d{ABSOLUTE} %-5p [%c{1}] %m%n"/>
</layout>
</appender>
```

또한 로그 레벨을 바꾸기 위해서는

아래와 같이 패키지의 로그를 제한하는 것이 좋다.

```
<category name="org.apache">
<priority value="INFO"/>
</category>
```

에서

```
<!-- Limit org.jgroups category to INFO -->
<category name="org.jgroups">
<priority value="INFO"/>
</category>
```

--WARN

root 카테고리의 로그 레벨을 바꾸는 작업을 아래와 같이 해줄 수 있다.

```
<root>
<appender-ref ref="CONSOLE"/> <!-- you may have removed this earlier -->
<appender-ref ref="FILE"/>
</root>
```

에서

```
<root>
<priority value="ERROR" />
<appender-ref ref="CONSOLE"/> <!-- you may have removed this earlier -->
```

<appender-ref ref="FILE"/>
</root>

프로그램상에서 log.isDebugEnabled()이 true 인 경우만 로그를 남기게 해주는 것이 좋다.

1.6. Connection Pools

1. 개발상에 XA connection 이 정말 필요한 경우를 제외하곤 XA Connection 은 사용을 자제 하여야 한다. XA Connection 은 좋은 성능을 내지 않는다.

2. DB Connection을 체크하기 위한 수단으로 "ping" 설정을 할수 있는 Database를 사용하는 것이 좋다. 또는 계속적으로 전체를 Connection 체크 하기보단 Database (Vendor 제공하는) Fail-Over 가능한 Driver를 사용하는 것을 권장한다.

1.7. Deployment Scanner

Deployment Scanner 는 매 5초마다 스캔을 한다. 따라서 느린 시스템구성(예를 들자면 NTFS) 일 경우 영향을 주게 된다.

되도록 스캔하는 간격을 늘려주거나 아예 전혀 스캔하게 하지 않는 것도 성능에는 좋을 수 있습니다.

1.8. Stateless Session Beans

EJB 1.x - 2.x 의 SLSB 는 ill-advised pooling model 으로 운영된다. 간단하게 이야기하자면 Pool 형태로 먼저 만들어진 다음 가져다 쓰는 형태로 되어 있다. 디폴트는 최소 갯수가 10로 세팅되어 있다.

우리는 **server/../conf/standardjboss.xml** 를 수정할 수 있다.

<container-configuration>
<container-name>Standard Stateless SessionBean</container-name>
<call-logging>false</call-logging>
<invoker-proxy-binding-name>stateless-rmi-invoker</invoker-proxy-binding-name>
<container-interceptors> 을 찾을 수 있으며

수정은 아래와 같이 할 수 있다.

<container-pool-conf>
<MaximumSize>100</MaximumSize>
</container-pool-conf>
</container-configuration>
에서
<container-pool-conf>
<MinimumSize>100</MinimumSize>

```
<MaximumSize>100</MaximumSize>
<strictMaximumSize/>
<strictTimeout>30000</strictTimeout>
</container-pool-conf>
</container-configuration>
```

대부분의 서버 환경에서는 pool의 사이즈가 커지거나 작아지는 것이 좋지 않다. 메모리상에 조각이 많이 발생할 수 있고 또한 heap 의 문제보다도 문제가 발생하기 때문이다. 따라서 성능적인 측면에서 pool의 내부 갯수는 모든 요청에 서비스할 수 있도록 충분히 큰 것이 좋다.

1.9. CMP tuning

현재로써는 CMP 모델을 사용하는 것보단 대신 **JBossHibernate** 를 권장한다.

참고자료

http://www.artima.com/forums/flat.jsp?forum=141&thread=24532

http://www.onjava.com/pub/a/onjava/2003/05/28/jboss_optimization.html

2. Slimming

2.1. 메일 서비스를 이용하지 않는 경우

server/slim/deploy/mail-service.xml 삭제하라.

server/slim/lib/mail* (mail-plugin.jar, mail.jar - JavaMail stuff) 을 삭제하라.

server/slim/lib/activation.jar (Java Activation Framework is used by JavaMail) 을 삭제하라.

2.2. the cache invalidation service을 이용하지 않는 경우

(이경우는 클러스터링 환경에서 CMP 옵션을 Commit Option A 을 이용하는 경우이다.)

server/slim/deloy/cache-invalidation-service.xml을 삭제하라.

2.3. J2EE client deployer service 를 이용하지 않는 경우

remove server/slim/deploy/client-deployer-service.xml 을 삭제하라

2.4. 통합 HAR 디플로이어와 Hibernate 세션관리 서비스를 이용하지 않는 경우

server/slim/deploy/hibernate-deployer-service.xml (HAR support) 을 삭제하라.

server/slim/lib/jboss-hibernate.jar (HAR support) 을 삭제하라.

server/slim/lib/hibernate2.jar (Hibernate itself) 을 삭제하라.

server/slim/lib/cglib-full-2.0.1.jar (used by Hibernate to create proxies of POJOs) 을 삭제하라.

server/slim/lib/odmg-3.0.jar 을 삭제하라.

2.5. Hypersonic 을 이용하지 않는 경우

JBossMQ에 Hypersonic가 DefaultDS 로 디플로이 되어 있음을 인지할 필요가 있다. 만약 다른 DS로 연결하려는 경우 Wiki를 참조하길 바란다.

server/slim/deploy/hsqldb-ds.xml 을 삭제하라.

server/slim/lib/hsqldb-plugin.jar 을 삭제하라.

server/slim/lib/hsqldb.jar 을 삭제하라.

2.6. JBossMQ를 이용하지 않는 경우

server/slim/deploy/jms 디렉토리 전체를 삭제하라

server/slim/lib/jbossmq.jar 을 삭제하라.

2.7. HTTPInvoker 를 사용하지 않는 경우

server/slim/deploy/http-invoker.sar 디렉토리 전체를 삭제하라.

2.8. XA Datasource를 사용하지 않는 경우 (분산 트랙젠션이 아닌경우)

server/slim/deploy/jboss-xa-jdbc.rar 을 삭제하라.

2.9. JMX-Console이 필요하지 않는 경우

server/slim/deploy/jmx-console.war 를 삭제하라.

2.10. RMI를 통한 JMX call이 없는 경우

server/slim/deploy/jmx-invoker-adaptor-server.sar 를 삭제하라.

server/slim/deploy/jmx-adaptor-plugin.jar 를 삭제하라.

2.11. web-console이 필요하지 않는 경우

server/slim/deploy/management/web-console.war 를 삭제하라.

2.12. JMX 를 위한 JSR-177 extensions이 필요하지 않는 경우

server/slim/deploy/management/console-mgr.sar 를 삭제하라.

2.13. Web-console과 jsr-177 extensions을 사용하지 않는 경우

server/slim/deploy/management 디렉토리 전체를 삭제하라.

2.14. 콘솔과 이메일 모니터 알림을 사용하지 않는 경우

server/slim/deploy/monitoring-service.xml 을 삭제하라.

server/slim/lib/jboss-monitoring.jar 을 삭제하라.

2.15. JMX 의 rich property editors 를 이용하여 System properties 를 수정하지 않는 경우

server/slim/deploy/properties-service.xml 을 삭제하라.

server/slim/lib/properties-plugin.jar 을 삭제하라.

2.16. 스케줄 서비스가 예제로 올라가있는데 필요없는 경우

server/slim/deploy/scheduler-service.xml 을 삭제하라.

2.17. JBoss 스케줄 매니너를 이용하지 않는 경우

server/slim/deploy/schedule-manager-service.xml 을 삭제하라.

server/slim/lib/scheduler-plugin* (scheduler-plugin.jar, scheduler-plugin-example.jar) 을 삭제하라.

2.18. 클라이언트사이드 트렌젝션관리또는 캐싱된 connections 을 이용하지 않는 경우

server/slim/deploy/user-service.xml 을 삭제하라.

2.19. JBoss UUID 키 generation 을 이용하지 않는 경우 (종종 CMP 에서 사용)

server/slim/deploy/uuid-key-generator.sar 를 삭제하라.

server/slim/lib/autonumber-plugin.jar 를 삭제하라.

2.20. HTTP를 통해 톰캣을 바로 이용하는 경우(mod_jk를 이용하지 않는 경우)

server/slim/deploy/jbossweb-tomcat5x.sar/server.xml 파일 안의 아래 내용을 제거하라.

```
<!-- A AJP 1.3 Connector on port 8009 -->
<Connector port="8009" address="${jboss.bind.address}"
enableLookups="false" redirectPort="8443" debug="0"
protocol="AJP/1.3"/>
```

2.21. Mod_jk를 이용하여 접근하는 경우(직접 접근이 없이)

server/slim/deploy/jbossweb-tomcat5x.sar/server.xml 파일 안의 아래 내용을 제거하라.

```
<!-- A HTTP/1.1 Connector on port 8080 -->
<Connector port="8080" address="${jboss.bind.address}"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8443" acceptCount="100"
connectionTimeout="20000" disableUploadTimeout="true"/>
```

2.22. JMS Queues를 이용하지 않는 경우

server/slim/conf/jboss-service.xml 파일 안의 아래 내용을 제거하라.

<mbean code="org.jboss.management.j2ee.LocalJBossServerDomain" 아래 <attribute name="JMSService">jboss.mq:service=DestinationManager</attribute> 을 제거하라.

2.23. CORBA/IIOP 를 이용하지 않는 경우

server/slim/conf/jboss-service.xml 파일 안의 아래 내용을 제거하라.

<mbean code="org.jboss.management.j2ee.LocalJBossServerDomain" 아래

<attribute name="RMI_IIOPService">jboss:service=CorbaORB</attribute> 을 제거하라.

2.24. user-transaction-service.xml을 제거하려고 한다면

server/slim/conf/jboss-service.xml 파일 안의 아래 내용을 제거하라.

<mbean code="org.jboss.management.j2ee.LocalJBossServerDomain" MBean 아래

<attribute

name="UserTransactionService">jboss:service=ClientUserTransaction</attribute> 을 제거하라.

2.25. JSR-77 을 필요 없다면

server/slim/conf/jboss-service.xml 파일 안의 아래 내용을 제거하라

```
<!-- ===================================================== -->
<!-- JSR-77 Single JBoss Server Management Domain -->
<!-- ===================================================== -->
<mbean code="org.jboss.management.j2ee.LocalJBossServerDomain"
name="jboss.management.local:j2eeType=J2EEDomain,name=Manager">
<attribute name="MainDeployer">jboss.system:service=MainDeployer</attribute>
<attribute name="SARDeployer">jboss.system:service=ServiceDeployer</attribute>
<!-- <attribute name="EARDeployer">jboss.j2ee:service=EARDeployer</attribute>-
->
<attribute name="EJBDeployer">jboss.ejb:service=EJBDeployer</attribute>
<attribute name="RARDeployer">jboss.jca:service=RARDeployer</attribute>
<attribute
name="CMDeployer">jboss.jca:service=ConnectionFactoryDeployer</attribute>
<attribute name="WARDeployer">jboss.web:service=WebServer</attribute>
<attribute name="MailService">jboss:service=Mail</attribute>
<!--                                                    <attribute
name="JMSService">jboss.mq:service=DestinationManager</attribute>-->
<attribute name="JNDIService">jboss:service=Naming</attribute>
<attribute name="JTAService">jboss:service=TransactionManager</attribute>
<!--                                                    <attribute
name="UserTransactionService">jboss:service=ClientUserTransaction</attribute>
<attribute name="RMI_IIOPService">jboss:service=CorbaORB</attribute>-->
</mbean>
```

2.26. 클라이언트 사이드 트렌젝션 관리를 하지 않는 다면 ₩

server/slim/conf/jboss-service.xml 파일 안의 아래 내용을 제거하라

```
<!--
| UserTransaction support.
-->
<mbean code="org.jboss.tm.usertx.server.ClientUserTransactionService"
name="jboss:service=ClientUserTransaction"
xmbean-dd="resource:xmdesc/ClientUserTransaction-xmbean.xml">
<depends>
<mbean code="org.jboss.invocation.jrmp.server.JRMPProxyFactory"
name="jboss:service=proxyFactory,target=ClientUserTransactionFactory">
<attribute name="InvokerName">jboss:service=invoker,type=jrmp</attribute>
<attribute name="TargetName">jboss:service=ClientUserTransaction</attribute>
<attribute name="JndiName">UserTransactionSessionFactory</attribute>
<attribute name="ExportedInterface">
org.jboss.tm.usertx.interfaces.UserTransactionSessionFactory
</attribute>
<attribute name="ClientInterceptors">
<interceptors>
<interceptor>org.jboss.proxy.ClientMethodInterceptor</interceptor>
<interceptor>org.jboss.invocation.InvokerInterceptor</interceptor>
</interceptors>
</attribute>
<depends>jboss:service=invoker,type=jrmp</depends>
</mbean>
</depends>
<depends optional-attribute-name="TxProxyName">
<mbean code="org.jboss.invocation.jrmp.server.JRMPProxyFactory"
name="jboss:service=proxyFactory,target=ClientUserTransaction">
<attribute name="InvokerName">jboss:service=invoker,type=jrmp</attribute>
<attribute name="TargetName">jboss:service=ClientUserTransaction</attribute>
<attribute name="JndiName"></attribute>
<attribute name="ExportedInterface">
org.jboss.tm.usertx.interfaces.UserTransactionSession
</attribute>
<attribute name="ClientInterceptors">
<interceptors>
```

```
<interceptor>org.jboss.proxy.ClientMethodInterceptor</interceptor>
<interceptor>org.jboss.invocation.InvokerInterceptor</interceptor>
</interceptors>
</attribute>
<depends>jboss:service=invoker,type=jrmp</depends>
</mbean>
</depends>
</mbean>
```

상단의 내용을 제거한 후 server/slim/conf/xmdesc/ClientUserTransaction-xmbean.xml을 제거 할수 있다.

2.27. persistent MBean attributes 가 필요하지 않다면

server/slim/conf/jboss-service.xml 파일 안의 아래 내용을 제거하라

```
<!--
========================================================================
== -->
<!-- XMBean Persistence -->
<!--
========================================================================
== -->
<mbean code="org.jboss.system.pm.AttributePersistenceService"
name="jboss:service=AttributePersistenceService"
xmbean-dd="resource:xmdesc/AttributePersistenceService-xmbean.xml">
<attribute name="AttributePersistenceManagerClass">
org.jboss.system.pm.XMLAttributePersistenceManager
</attribute>
<attribute name="AttributePersistenceManagerConfig">
<data-directory>data/xmbean-attrs</data-directory>
</attribute>
<attribute name="ApmDestroyOnServiceStop">false</attribute>
<attribute name="VersionTag"></attribute>
</mbean>
```

상단의 내용을 제거한 후

server/slim/conf/xmdec/xmdesc/AttributePersistenceService-xmbean.xml 을 제거할 수 있다.

2.28. RMI 클래스로딩을 이용하지 않는 경우 (클라이언트에서 서버로 codebases를 에 로딩하지 않는 경우)

server/slim/conf/jboss-service.xml 파일 안의 아래 내용을 제거하라

```
<!--
========================================================================
== -->
<!-- JBoss RMI Classloader - only install when available -->
<!--
========================================================================
== -->
<mbean code="org.jboss.util.property.jmx.SystemPropertyClassValue"
name="jboss.rmi:type=RMIClassLoader">
<attribute name="Property">java.rmi.server.RMIClassLoaderSpi</attribute>
<attribute name="ClassName">org.jboss.system.JBossRMIClassLoader</attribute>
</mbean>
```

과

```
<!--
========================================================================
== -->
<!-- Class Loading -->
<!--
========================================================================
== -->
```

```
<mbean code="org.jboss.web.WebService"
name="jboss:service=WebService">
<attribute name="Port">8083</attribute>
<!-- Should resources and non-EJB classes be downloadable -->
<attribute name="DownloadServerClasses">true</attribute>
<attribute name="Host">${jboss.bind.address}</attribute>
<attribute name="BindAddress">${jboss.bind.address}</attribute>
</mbean>
```

을 삭제하라.

2.29. 로컬 JBoss 네이밍을 원한다면 (RMI 클라이언트 없이)

server/slim/conf/jboss-service.xml 을 열어 아래와 같이 바꾸면 된다.

Before

```
<!-- ===================================================== -->
<!-- JNDI -->
<!-- ===================================================== -->
<mbean code="org.jboss.naming.NamingService"
name="jboss:service=Naming"          xmbean-dd="resource:xmdesc/NamingService-
xmbean.xml">
...
<!-- The listening port for the bootstrap JNP service. Set this to -1
to run the NamingService without the JNP invoker listening port.
-->
<attribute name="Port">1099</attribute>
...
<!-- The port of the RMI naming service, 0 == anonymous -->
<attribute name="RmiPort">1098</attribute>
...
</mbean>
```

After

```
<!-- ===================================================== -->
<!-- JNDI -->
<!-- ===================================================== -->
<mbean code="org.jboss.naming.NamingService"
name="jboss:service=Naming"          xmbean-dd="resource:xmdesc/NamingService-
xmbean.xml">
...
<!-- The listening port for the bootstrap JNP service. Set this to -1
to run the NamingService without the JNP invoker listening port.
-->
<attribute name="Port">-1</attribute>
...
<!-- The port of the RMI naming service, 0 == anonymous -->
<attribute name="RmiPort">0</attribute>
...
</mbean>
```

RMI port는 대게 선택하는 항목이다. 그렇지만 만드시 1098에 바인딩하지는 않아도 된다.

2.30. Thread pool을 제거하려면

아래와 같은 pool 찾는 부분을 제거하고

```
<depends optional-attribute-name="LookupPool"
proxy-type="attribute">jboss.system:service=ThreadPool</depends>
```

pool을 정의한 mbean을 제거한다.

```
<!-- A Thread pool service -->
<mbean code="org.jboss.util.threadpool.BasicThreadPool"
name="jboss.system:service=ThreadPool">
<attribute name="Name">JBoss System Threads</attribute>
<attribute name="ThreadGroupName">System Threads</attribute>
<attribute name="KeepAliveTime">60000</attribute>
<attribute name="MinimumPoolSize">1</attribute>
<attribute name="MaximumPoolSize">10</attribute>
<attribute name="MaximumQueueSize">1000</attribute>
<attribute name="BlockingMode">run</attribute>
</mbean>
```

2.31. BeanShell을 사용을 원하지 않는다면

server/slim/conf/jboss-service.xml 파일을 열어 아래와 같은 xml을 삭제하라.

```
<mbean code="org.jboss.varia.deployment.BeanShellSubDeployer"
name="jboss.scripts:service=BSHDeployer">
</mbean>
```

또한 server/slim/bsh* (bsh-deployer.jar, bsh-1.3.0.jar) 을 삭제하라.

2.32. JBoss 리스타트 없이 deploy 안의 파일을 hot deploy 하지 않으려면

server/slim/conf/jboss-service.xml 파일을 열어 아래와 같이 수정하라.

```
<!-- An mbean for hot deployment/undeployment of archives. -->
<mbean code="org.jboss.deployment.scanner.URLDeploymentScanner"
name="jboss.deployment:type=DeploymentScanner,flavor=URL">
...
```

```
<attribute name="ScanPeriod">5000</attribute>
...
</mbean>
```

에서

```
<!-- An mbean for hot deployment/undeployment of archives. -->
<mbean code="org.jboss.deployment.scanner.URLDeploymentScanner"
```

```
name="jboss.deployment:type=DeploymentScanner,flavor=URL">
...

<attribute name="ScanPeriod">5000</attribute>
<attribute name="ScanEnabled">False</attribute>
...
</mbean>
```