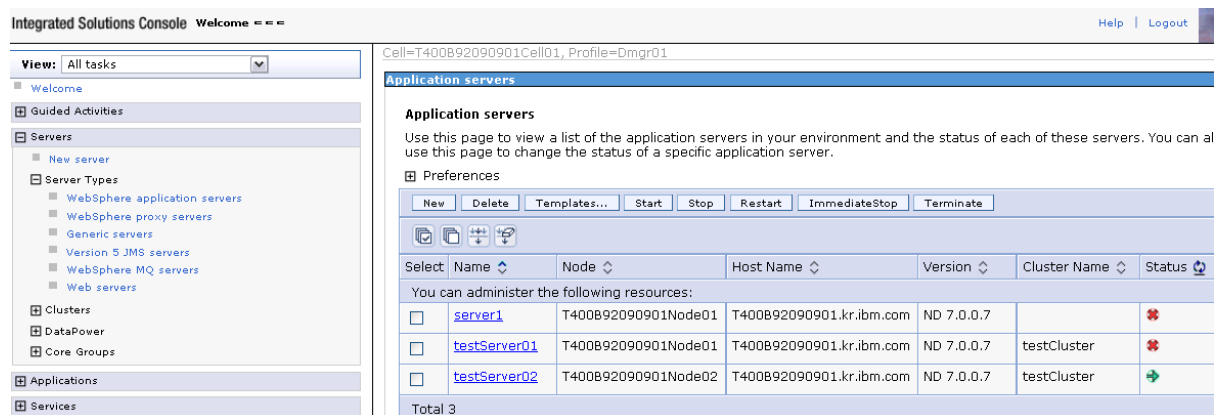


## 하나씩 쉽게 따라 해보는 IBM WebSphere Application Server(WAS) v7 – 18

이정운 ([juwlee@kr.ibm.com](mailto:juwlee@kr.ibm.com))

하나씩 쉽게 따라 해보는 IBM WAS v7 그 열 여덟번째 이야기를 시작합니다. 열 여덟번째 이야기는 지난 강좌와 연이어서 조금은 비슷하게 실제 관리콘솔 내의 WAS 서버에 대한 옵션 설명을 간단히 드리고자 합니다. 결국 WAS 에서 가장 중요한 것은 JVM instance 즉, WAS 서버입니다. 이를 각각의 환경에서 얼마나 Customize 하여 맞추어 사용하는 지에 따라, 시스템의 성능이나 운영에 많은 도움을 받을 수 있습니다. 그런데도 많은 WAS 운영자 분들은 프로젝트 중에 설정한 옵션을 그대로 변경없이 사용만 하고 있을 뿐, 실질적으로 이게 어떤 의미이며 해당 회사에서 어떤 value 를 주는지 이해하지 못한채 사용하는 면이 많습니다. 귀찮니즘이 많은 부분을 작용할 수도 있고, 운영환경을 변경해서 겪을 수 있는 위험을 감수하지 않기 위해서 일 수도 있습니다. 그렇지만 변경을 하지 않는다고 해도 WAS 의 해당 옵션을 이해하는 것은 WAS 를 이해하는데 한 발자국 더 나아갈 수 있다라고 생각합니다.

비록 WAS 서버의 모든 옵션을 여기서 다룰 수는 없겠지만, 가급적 중요한 옵션들 위주로 설명하여 실제로 사용하는 사용자나, 관리자 분들이 관심을 가질만한 내용을 녹여내도록 노력하도록 하겠습니다. 하나씩 쉽게 따라 해보는 IBM WAS v7 강좌의 처음은 초보자를 위한 것이지만, 첫번째 강의부터 차근 차근 잘 따라오셨다면 이제는 조금 더 내부적으로 파고 들어서 “당신도 WebSphere Application Server 전문가” 라는 말을 들을 수 있도록 진행하도록 노력하겠습니다. 뭐, 아직은 시작인지 얼마되지 않은지라 전문가라는 소리가 과분하기는 하지만 이제 전문가의 시작이라는 마음가짐을 가지며 강좌를 따라해보고, 복습해보고 테스트 해보시기 바라겠습니다.^ ^&



Integrated Solutions Console Welcome ==

Cell=T400B92090901Cell01, Profile=Dmgr01

**Application servers**

Use this page to view a list of the application servers in your environment and the status of each of these servers. You can also use this page to change the status of a specific application server.

Preferences

New Delete Templates... Start Stop Restart ImmediateStop Terminate

Select	Name	Node	Host Name	Version	Cluster Name	Status
<input type="checkbox"/>	server1	T400B92090901Node01	T400B92090901.kr.ibm.com	ND 7.0.0.7		✖
<input type="checkbox"/>	testServer01	T400B92090901Node01	T400B92090901.kr.ibm.com	ND 7.0.0.7	testCluster	✖
<input type="checkbox"/>	testServer02	T400B92090901Node02	T400B92090901.kr.ibm.com	ND 7.0.0.7	testCluster	✔

Total 3

## Part 1. 기본적인 WAS 서버 옵션

WAS 의 관리콘솔에서 Servers > Server types > WebSphere application servers 메뉴를 클릭하여 나타난 WAS 서버 중에서 하나의 WAS 서버를 선택해 보면 아시겠지만, 굉장히 많고 다양한 설정들을 확인하실 수 있습니다. 우선적으로 몇가지 기본적인 옵션들에 대해서 설명하도록 하겠습니다.

[Application servers](#) > testServer02

Use this page to configure an application server. An application server is a server that provides services required to run enterprise applications.

The screenshot shows the configuration page for testServer02. The 'Configuration' tab is active. The 'General Properties' section includes fields for Name (testServer02) and Node name (T400B92090901Node02). Under 'Run in development mode', the checkbox is checked. The 'Container Settings' section lists various container settings. The 'Applications' section lists installed applications. The 'Server-specific Application Settings' section shows 'Classloader policy' as Multiple and 'Class loading mode' as Classes loaded with parent class loader first.

**Run in development mode** 라는 옵션은 WAS 서버를 개발을 위한 개발모드로 실행할 것이냐를 묻는 옵션입니다. 개발 모드란 말 그대로 개발 환경을 위해 최적화 된 모드로 서버를 시작하는 것을 말합니다. 개발 단계에서는 그렇게 많이 필요로 하지 않는 검사 기능이나 최적화 기능등을 사용하지 않고 최대한 빠르고 가볍게 시작할 수 있는 상태를 말합니다. 이렇게 되면 산술적 수치로는 15% ~ 30% 정도의 WAS 서버 시작시간을 줄일 수 있습니다. 이 모드를 좀 더 들여다 보면, 결국은 JVM 특성의 사용자 옵션(Custom property) -Xverify:none 및 -Xquickstart 를 사용하도록 지정하는 것입니다. -Xverify:none 옵션은 Linking 시에 바이트코드(bytecode) 가 적절하게 맞는지 검사를 하게 되어있는데 이 부분을 생략하는 옵션이며, -Xquickstart 는 JIT 에서 컴파일이 일어나면 자동으로 최적화 작업이 이루어지는데 이 부분을 최소화 하는 옵션입니다. 따라서, 위의 개발 모드를 사용하게 되면 개발 시에 좀 더 빠른 WAS 재시작 및 반영을 확인할 수 있으며, 더 적은 메모리 풋프린트를 확인할 수 있습니다. 단, 가급적 용도에 맞게 개발 모드에서만 사용하시기를 바라며, 운영 시에는 해제되어야 좀 더 나은 성능과 안정성을 제공할 수 있다는 점을 잊지 마시길 바라겠습니다.

**Parallel start** 옵션은 병렬 시작 옵션을 선택할 것이냐를 묻는 옵션입니다. 병렬 시작이란 순차적 이 아니라 다수의 스레드(Multiple Thread) 를 통해서 서버 컴포넌트나 서비스, 어플리케이션을 병

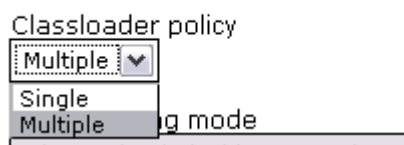
렬로 수행하는 것으로 WAS 서버에 대한 시작 시간을 줄일 수 있습니다. 병렬 시작 옵션을 선택하지 않으면 모두 순차적으로 시행되기 때문에 WAS 시작 시간이 조금 더 걸릴 수 있습니다.

다음으로 **Start components as needed** 옵션입니다. 해당 옵션은 어플리케이션 운영에 필요한 서버 컴포넌트들만 시작시키기 위한 옵션입니다. WAS 는 여러가지의 서버 컴포넌트를 가지고 있습니다. 웹 서버 컨테이너, SIP 컨테이너, EJB 컨테이너 등등 J2EE 표준에 맞추어 여러 컨테이너를 가지고 있는데 이를 불 필요하게 모두 시작시키는 것이 아니라 배포된 어플리케이션을 분석하여 필요한 컴포넌트만 선택적으로 시작시킬 수 있는 옵션입니다. 이전 까지 많은 개발자 분들이 말했던 WAS 에 대한 불평 및 불만 중에 가장 큰 것은 아마도 WAS 가 무겁고 느리다라는 이야기 일 것입니다. 그것은 이전 버전까지 WAS 서버를 시작하기 위해서 국제 표준에 따른 모든 컨테이너를 다 시작시키기 때문에 그러한 경향이 있을 수 있습니다. 그러나, 이번 WAS v7 의 본 옵션을 사용하게 되면 필요한 컴포넌트만 선택적으로 시작시키므로 WAS 서버에 대한 시작 시간을 단축시킬 수 있으며, 불필요한 컴포넌트를 시작시키지 않으므로 메모리 풋프린트를 줄일 수 있는, 좀 더 가벼운 WAS 로서의 장점을 줄 수 있습니다.

클래스 로더 옵션은 매우 중요하기 때문에 이번 강좌에서는 자세한 설명은 하지 않고 간단히만 이야기 하고 넘어가겠습니다. 추후에 하나의 강좌로 좀 더 자세히 다시 다루도록 하겠습니다. (언젠가는^^&~)

먼저 살펴볼 **Class loader policy** 에는 두 가지 옵션이 있습니다.

#### Server-specific Application Settings



그 두 가지는 Multiple 과 Single 옵션입니다. 우리가 기본적으로 사용하는 클래스 로더 정책은 Multiple 로서 해당 옵션은 클래스 로더 여러 개를 사용한다는 의미입니다. 이를 다시 말하면, 하나의 WAS 안에 여러 EAR 어플리케이션을 배포하면 각각의 EAR 어플리케이션 별로 클래스 로더가 만들어 집니다. 그러나, 만약 해당 옵션을 Single 로 변경하신다면 여러 EAR 어플리케이션을 배포하신다고 하여도 모든 어플리케이션은 하나의 클래스 로더에 올라가게 됩니다.


다음으로 살펴볼 것은 **Class loading mode** 입니다. 클래스로더 정책을 single 로 했을 경우에만 선택이 가능한 옵션으로 JDK 에서 기본적으로 사용되는 parant first 방식을 사용할 수 있으며, 필요에 의해서 parent last 방식으로 변경 가능합니다. Parent first 방식이란 클래스를 로드할 경우에 상위의 부모 클래스 로더에 동일 클래스가 이미 있을 경우에는 어플리케이션 클래스로더가 아니라 부모 클래스 로더에서 로딩하는 방식입니다. (항상 위로 던진다고 이해하시면 됩니다. 그리고 나서 없으면 자기 클래스 로더에서 로드하는 거죠.) 이 방식이 Java 의 기본 클래스 로딩 방식입니다. Parent last 방식은 parent first 방식과는 반대로 부모 클래스 로더에 동일 클래스가 있더라도 그냥 자기 클래스 로더에서 무조건 로딩하는 방식입니다. 이는 Java 클래스의 호출 구조, 즉 어플리케이션 아키텍처 상에서 상당히 중요한 문제이기 때문에 잘 고려하신 후에, 필요하다면 설정

변경을 하시기 바라겠습니다.


#### Server-specific Application Settings

---

Classloader policy

Single 

Class loading mode

Classes loaded with parent class loader first 

Classes loaded with parent class loader first

Classes loaded with local class loader first (parent last)

## Part 2. 웹 컨테이너 설정 옵션

다시 또 강조드리지만 WAS 는 결국 J2EE 표준을 준수하는 J2EE 컨테이너 입니다. J2EE 컨테이너 는 J2EE 컨테이너 라는 단일 컨테이너로 이루어진 것이 아니라 웹 컨테이너, EJB 컨테이너, SIP 컨테이너 등등의 여러가지 컨테이너를 종합하여 이루어지는 것 입니다. 이번 파트에서는 이 컨테이너 중 사용자 환경에서 많이 사용되는 컨테이너들의 옵션을 간략히 살펴 보는 시간을 가져보도록 하겠습니다.

처음으로 소개해드릴 것은 Session management 와 웹 컨테이너 입니다.

### Container Settings

- [Session management](#)

- ⊕ SIP Container Settings

- ⊖ Web Container Settings

- [Web container](#)

- [Web container transport chains](#)

**Session management** 옵션을 클릭하면 하단과 같은 세부 설정 메뉴를 보실 수 있습니다. Session management 옵션은 세션을 관리하기 위한 옵션이며 Session 을 위해 쿠키를 사용할지에 대한 여부라던가 세션 타임아웃 값 같은 것을 설정 할 수 있습니다. 이전 Session Clustering 강좌에서 어느 정도 이 부분에 대하여 설명을 했으므로 간단하게 넘어가도록 하겠습니다.

[Application servers](#) > [server1](#) > Session management

Use this page to configure session manager properties to control the behavior of Hypertext Transfer Protocol (HTTP) session support. These settings apply to both the SIP container and the Web container.

Configuration

#### General Properties

##### Session tracking mechanism:

- ☐ Enable SSL ID tracking
- ☒ [Enable cookies](#)
- ☐ Enable URL rewriting
  - ☐ Enable protocol switch rewriting

Maximum in-memory session count:  
 sessions

☒ Allow overflow

##### Session timeout:

- ☐ No timeout
- ☒ Set timeout  
 minutes

#### Additional Properties

- [Custom properties](#)
- [Distributed environment settings](#)

보다 더 중요한 것은 웹 컨테이너 입니다. 웹 컨테이너는 그 이름처럼 Web 에 관련된 처리를 하는 컨테이너로서, 기본적으로 HTTP 같은 정적 콘텐츠나 Servlet, JSP 같은 동적 콘텐츠를 처리합니다. 웹 컨테이너 설정을 보시면 이전에 첨부한 것처럼, 크게 두가지가 보이는 것을 확인하실 수

있습니다. 그중에 먼저 **Web container** 를 선택하면 다시 하단과 같은 세부 메뉴를 확인하실 수 있습니다.

[Application servers](#) > [server1](#) > **Web container**

Use this page to configure the Web container.

Configuration

#### General Properties

Default virtual host:

default\_host ▼

☐ Enable servlet caching

☐ Disable servlet request and response pooling

#### Additional Properties

- [Asynchronous Request Dispatching](#)
- [Custom properties](#)
- [Web container transport chains](#)
- [Session management](#)

**Enable servlet caching** 옵션은 말 그대로 Servlet caching 의 사용여부를 결정하는 옵션입니다. 본 옵션을 사용하면 Servlet 이 한 번 호출되면 output 을 caching 하여 성능상의 이점을 줄 수 있습니다. (별도의 추가 설정도 필요없이 옵션만으로 Servlet 캐싱을 가능하게 할 수 있습니다.)

**Disable servlet request and response pooling** 옵션도 해석 그대로 Servlet 에 대한 요청과 응답 객체를 pooling 하여 사용자에게 성능과 자원상의 이점을 주는 것을 끄는 옵션입니다. 보통 Servlet pooling 기능에 버그가 있다고 판단되었을 때 Servlet pooling 기능을 끄기 위해 사용되니 다만 잘 사용되어지지 않는 옵션입니다.

여기까지가 web container 에 관련된 옵션에 대한 설명이었고 다음으로 **web container transport chains** 옵션에 대한 설명을 진행하도록 하겠습니다. 해당 메뉴를 클릭하면 다음과 같은 전송 체인(transport chain) 에 대한 리스트를 확인할 수 있습니다. (전송 체인을 간단히 설명하면 전송을 위한 채널(channel) 들의 연결입니다. 전송에 관련되는 각 레이어들이 체인처럼 엮여있다고 전송 체인이라고 부르는 것이 아닌가 하는 것이 제 개인적인 생각입니다.^^&; 하단에 그림을 참고하시면 바로 이해하실 수 있습니다.)

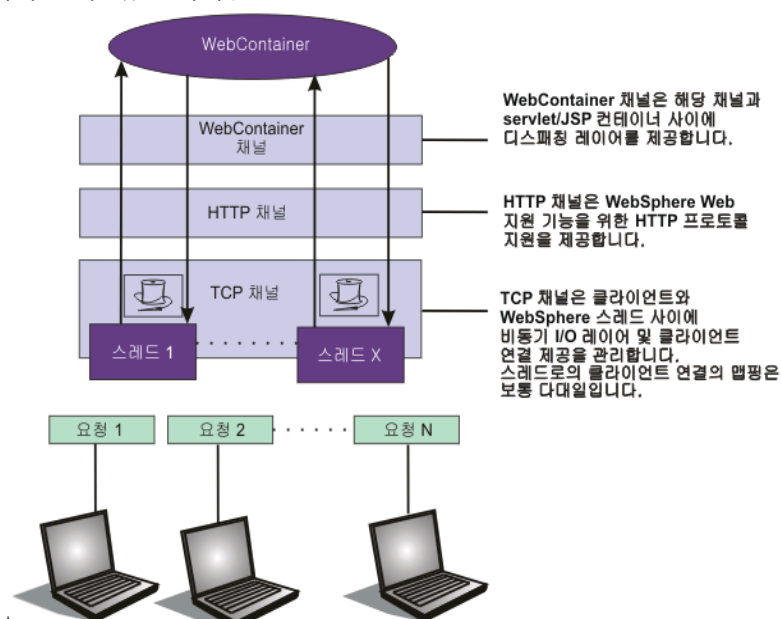


그림 1. 전송 채널 서비스

참조: IBM WAS Information Center

체인은 각 포트에 연결된 서비스 체인으로서 웹 컨테이너에서 서비스를 수행하는데 사용되는 기본 전송 체인은 WCInboundDefault 입니다.

(보안을 사용하셨다면 WCInboundDefaultSecure 전송 체인을 서비스에 이용합니다.)

[Application servers](#) > [server1](#) > [Web container transport chains](#)

Use this page to view and manage a transport chain. Transport chains represent network protocol stacks that are operating within a client or server.

Preferences

<div>New Delete</div> <div> </div>					
Select	Name	Enabled	Host	Port	SSL Enabled
You can administer the following resources:					
<input type="checkbox"/>	<a href="#">HttpQueueInboundDefault</a>	Enabled	*	9083	Disabled
<input type="checkbox"/>	<a href="#">HttpQueueInboundDefaultSecure</a>	Enabled	*	9446	Enabled
<input type="checkbox"/>	<a href="#">WCInboundAdmin</a>	Enabled	*	9064	Disabled
<input type="checkbox"/>	<a href="#">WCInboundAdminSecure</a>	Enabled	*	9047	Enabled
<input type="checkbox"/>	<a href="#">WCInboundDefault</a>	Enabled	*	9083	Disabled
<input type="checkbox"/>	<a href="#">WCInboundDefaultSecure</a>	Enabled	*	9446	Enabled
Total 6					

**WCInboundDefault** 세부 메뉴를 확인하기 위하여 해당 이름을 클릭하여 세부 메뉴로 들어갑니다.

[Application servers](#) > [server1](#) > [Web container transport chains](#) > [WCInboundDefault](#)

Use this page to view and manage a transport chain. Transport chains represent network protocol stacks that are operating within a client or server.

Configuration

#### General Properties

Name  
WCInboundDefault

☒ Enabled

#### Transport Channels

##### TCP inbound channel (TCP\_2)

Host \*  
Port 9083  
Thread pool WebContainer  
Maximum open connections 20000  
Inactivity timeout 60 seconds

##### HTTP inbound channel (HTTP\_2)

Use persistent keep alive connections Enabled  
Maximum persistent requests per connection 100  
Read timeout 60 seconds  
Write timeout 60 seconds  
Persistent timeout 30 seconds

##### Web container inbound channel (WCC\_2)

세부 메뉴로 들어가면 위의 그림과 같이 TCP inbound channel 과 HTTP inbound channel 등을 확인할 수 있습니다. 여기서 채널에 대해서 짚고 넘어가야 할 것 같은데 Websphere 에서 채널은 쉽게 연결(Connection) 을 의미한다고 생각하시면 됩니다. 보다 더 깊이 들어가면 연결도 각 network 계층, 즉 network layer 를 분리한 개념으로 이해하실 수 있습니다. TCP inbound 채널의 경우는 TCP network layer 의 inbound connection 에 대한 옵션이라고 보시면 됩니다. 마찬가지로 HTTP inbound 채널의 경우는 TCP layer 상위의 HTTP layer 의 inbound connection 에 대한 옵션

으로 이해하시면 됩니다. (약간의 network 개념과 프로토콜에 대한 이야기가 어렵긴 하지만 Web 을 하시면 TCP 나 HTTP 프로토콜에 대해서는 많이 들어서 대략 이해하실 수 있다고 생각합니다. 잘 이해가 안되시는 분들은 인터넷으로 보충학습을..^^&#x2192;)

이제 실제로 채널 내부의 옵션을 확인하시 위하여 **TCP inbound channel** 을 클릭하여 세부 내용을 확인합니다.

[Application servers](#) > [server1](#) > [Web container transport chains](#) > [WCInboundDefault](#) > TCP inbound channel (TCP\_2)

Use this page to configure a TCP inbound channel for inbound network traffic.

Configuration

---

**General Properties**

\* Transport channel name  
TCP\_2

Port  
WC\_defaulthost (\*:9083)

Thread pool  
WebContainer

\* Maximum open connections  
20000

\* Inactivity timeout  
60 seconds

**Additional Properties**

■ [Custom properties](#)

**Related Items**

■ [Ports](#)

■ [Thread pools](#)

우선 **Port** 를 확인하실 수 있는데, 이는 해당 TCP connection 과 매핑된 포트입니다. 실질적으로 서비스를 하는 서비스 포트로 보시면 됩니다. (위의 예에서는 9083 으로 되어 있는데 이 경우라면 웹으로 요청을 받는 경우 9083 포트로 요청을 보내면 서비스를 받을 수 있다는 의미입니다.

예제: <http://127.0.0.1:9083/testServlet> )

그 다음이 **Thread pool** 옵션입니다. Java 에서 중요한 것중에 하나가 바로 실제 업무를 수행하는 이 스레드(Thread)죠. 스레드를 어떻게 이용하느냐가 성능과 자원 효율성을 좌지우지 할 수 있습니다. 이 옵션인 Thread pool 에 나온 이름의 실제의 스레드 풀에 매핑되는데, 해당 스레드 풀은 최대, 최소 값을 가지고 자동으로 조절이 됩니다. 이 값은 성능 테스트나 튜닝을 통해서 적절한 값으로 조절하여 사용하시기 바라겠습니다.

**Maximum open connections** 는 해당 port 로 열 수 있는 최대 TCP connection 의 개수를 의미합니다. 아무리 사용자의 요청이 많다고 해도 해당 WAS 의 웹 컨테이너에는 해당 개수 만큼의 TCP connection 이 맺어질 수 있으며 그 이상은 전부 거부 됩니다.

**Inactivity timeout** 은 자원의 효율성을 위한 옵션으로 TCP connection 이 연결된 이후에 아무런 활동이 없다면 지정된 시간 이후에 자동으로 close 합니다. 이 값이 60초로 되어 있다면 60초간 아무런 활동이 없는 TCP connection 은 자동으로 close 되는 것 입니다.



다음으로 **HTTP inbound channel** 을 클릭하여 세부 설정 옵션을 확인합니다.

[Application servers](#) > [server1](#) > [Web container transport chains](#) > [WCInboundDefault](#) > **HTTP inbound channel (HTTP\_2)**

Use this page to configure a channel for handling inbound HTTP requests from a remote client.

Configuration

### General Properties

★ Transport channel name

HTTP\_2

Discrimination weight

1

★ Read timeout

60

seconds

★ Write timeout

60

seconds

★ Persistent timeout

30

seconds

#### Persistent connections



Use persistent keep alive connections



Unlimited persistent requests per connection



Maximum persistent requests per connection

### Additional Properties

■ [Custom properties](#)

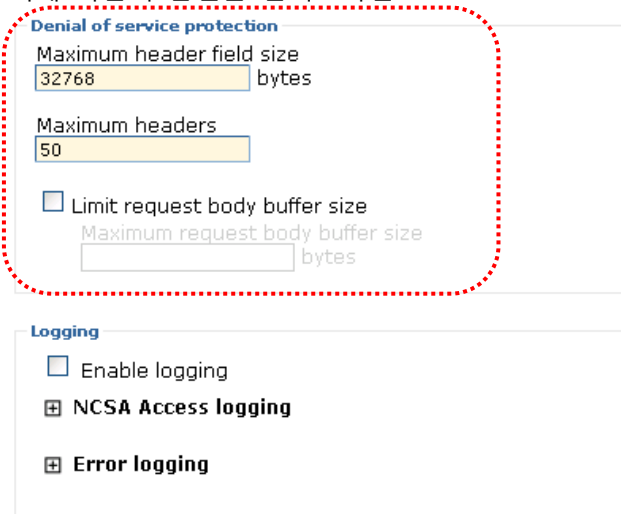
### Related Items

■ [NCSA access and HTTP error logging](#)

**Read timeout** 은 HTTP 단에서 요청을 읽기위한 대기시간의 최대값을 의미합니다. HTTP 요청에 대한 읽기가 처음으로 시작된 후, 소켓에 있는 요청을 모두 읽을 때까지의 대기시간 입니다. 이 시간을 넘어가면 해당 요청에 대해서 자동으로 Exception 이 발생하게 됩니다. 이와 마찬가지로, **Write timeout** 은 HTTP 응답 데이터의 전송이 완료될 때까지 대기시간의 최대값을 의미하며, **persistent timeout** 은 HTTP 요청 전송 후 idle 상태로 대기할 수 있는 최대값을 의미합니다.

**Persistent Connection** 은 지속 연결이라고도 하며 보통 KeepAlive 모드라고 불리는 HTTP 옵션의 사용여부를 결정하는 옵션입니다. Http 는 기본적으로 stateless 연결을 합니다.(무상태 방식이라고도 하며 연결에 대한 상태를 저장하지 않는 방식입니다.) 그렇기 때문에, 이 경우에는 요청에 따라 매번 연결을 맺고 끊고, 다시 맺는 비효율적인 활동이 일어납니다. 따라서, 이를 보완하고자 HTTP 1.1 스펙에서 KeepAlive 모드라는 것이 새롭게 소개가 되었습니다. KeepAlive 모드는 특별히 어려운 기술이라기 보다는 사용된 연결을 끊지 않고 그대로 재사용하는 것 입니다. 이를 통해 성능과 자원의 효율성을 높이는 것이고, 바로 이 옵션을 통해서 사용여부를 결정 할 수 있습니다.

이제 하단의 옵션을 살펴보자면 **Denial of service protection** 설정을 확인할 수 있습니다.



**Denial of service protection**

Maximum header field size  
32768 bytes

Maximum headers  
50

☐ Limit request body buffer size  
Maximum request body buffer size  
bytes

**Logging**

☐ Enable logging

☒ NCSA Access logging

☒ Error logging

DOS 라는 것을 들어보신 적이 있나요? 아니면 DDOS ? DOS 라는 것은 해킹의 한가지 기술로서 서비스 거부 공격을 말합니다. 정상적인 것 처럼 보이는 과부하 서비스를 서버로 보내서 해당 서비스를 처리하는데 모든 자원을 다 써서 다른 사용자가 원하는 서비스를 못 받게 하는 해킹 공격입니다. (DDOS는 여기에 분산을 추가한 것입니다. 즉 공격자를 아주 많이 분산시켜서 정상적인 사이즈의 서비스를 서버로 요청하여, 마찬가지로 서비스를 못 받게 하는 공격입니다.) 해당 옵션은 HTTP 프로토콜 단에서 직접 DOS 공격을 방지할 수 있는 옵션입니다.

그 다음으로 나온 Logging 카테고리는 WAS 에서 나오는 기본적인 로깅데이터 이외에 추가적으로 HTTP 요청에 대한 로깅을 가능하게 하는 옵션입니다. 그리 중요한게 아니니 이걸 간단하게 패스하도록 하겠습니다.

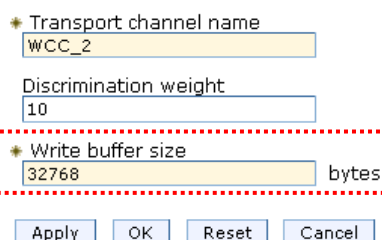
이제 웹 컨테이너 설정의 마지막으로 **Web container inbound channel** 을 선택하여 세부 설정 메뉴를 확인하도록 하겠습니다.

[Application servers](#) > [server1](#) > [Web container transport chains](#) > [WCInboundDefault](#) > **Web container inbound channel (WCC\_2)**

Use this page to view and configure an HTTP inbound channel. This type of transport channel handles inbound messages for servlets and JSP engines.

Configuration

#### General Properties



\* Transport channel name  
WCC\_2

Discrimination weight  
10

\* Write buffer size  
32768 bytes

Apply OK Reset Cancel

#### Additional Properties

☐ Custom properties

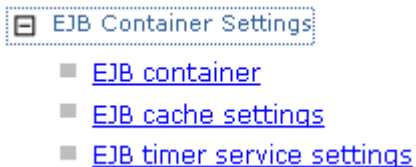
여기서 가장 중요한 옵션은 **Write buffer size** 옵션으로서 웹 컨테이너의 쓰기 버퍼 사이즈를 설

정하는 옵션입니다. 버퍼가 무엇에 쓰이는 것인지는 알고 계시죠? 데이터 쓰기를 예로 들면, Stream에서 데이터를 받아온다고 계속적으로 쓰는 것이 아니라 일정 크기의 버퍼에 우선 쓰고 나서 버퍼가 차면 실제로 버퍼단위로 데이터를 쓰는 것입니다. 이렇게 버퍼를 사용하게 되면 처리 속도를 높일 수 있습니다. 현재 해당 옵션은 기본적으로 32KB 로 설정 되어 있습니다.

### Part 3. EJB 컨테이너 설정 옵션

지금까지는 웹 컨테이너 설정 옵션을 살펴봤으니 이번 파트에서는 EJB 컨테이너 설정 옵션중에 중요한 것만 살펴보고 넘어가도록 하겠습니다. (이전에도 말씀드렸지만 이번 강좌에서 모든 설정을 이야기하기가 어렵기 때문에, 여기서 설명하지 않은 옵션들은 IBM WAS Information Center 에 친절히 설명되어 있으니 해당 내용을 참조하시기 바라겠습니다.)

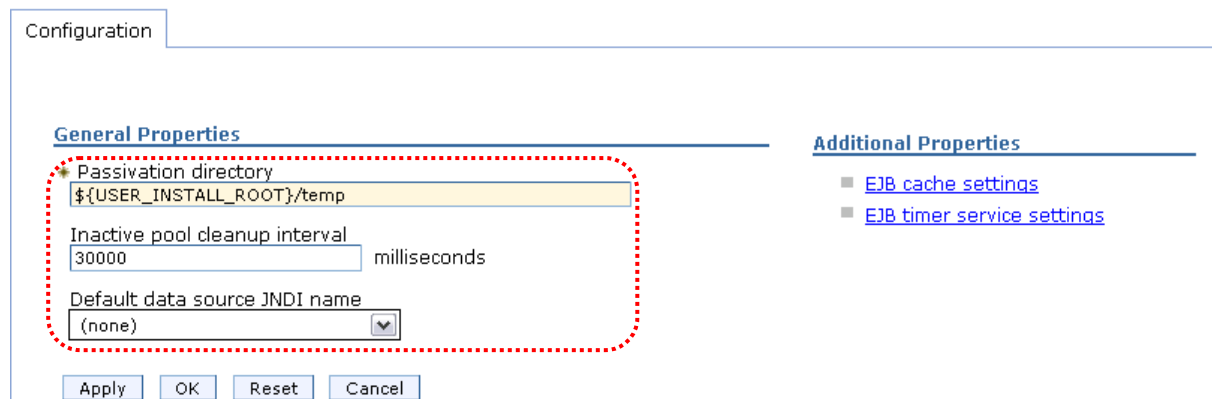
먼저 살펴볼 것은 **EJB Container Setting** 입니다. 먼저 살펴보기는 하지만 거의 수정하는 일이 없으므로 참고만 하시길 바라겠습니다. EJB Container Settings 를 열면 하단의 그림처럼 3가지의 큰 카테고리를 볼 수 있습니다.



먼저 **EJB container** 를 선택하면 하단과 같은 세부 설정 옵션을 확인할 수 있습니다.

[Application servers](#) > [server1](#) > **EJB container**

Specifies that an EJB container is a component of a J2EE application server that provides runtime services to EJB modules that can be deployed within it.



**Passivation directory** 는 EJB의 stateful session bean 을 사용할 경우 지속적 상태를 저장하기 위한 디렉토리를 지정합니다. 해당 디렉토리는 사전에 만들어야 하며 자동으로 생성되지는 않습니다.

**Inactive pool cleanup interval** 옵션은 stateless session 이나 entity bean pool 에 활동하지 않는 객체가 메모리 점유를 줄이기 위하여 삭제될 수 있는지 조사하는 주기를 설정합니다.

마지막으로 **Default data source JNDI name** 은 데이터 소스를 위한 기본 JNDI 이름으로 사용됩니다. 만약에 어플리케이션을 배치할 때 특별하게 데이터 소스를 지정하지 않았다면 해당 값으로 자동 매핑됩니다.

다음으로 살펴볼 것은 **EJB cache settings** 입니다. 이름에서 느끼셨겠지만 해당 옵션은 EJB 캐시를 위한 세부 설정을 위해 사용됩니다.

[Application servers](#) > [server1](#) > **EJB cache settings**

Use this page to configure the cache. Each EJB container maintains a cache of bean instances for ready access.

Configuration

---

**General Properties**

✱ Cleanup interval

3000

milliseconds

✱ Cache size

2053

buckets

Apply

OK

Reset

Cancel

**Cleanup interval** 은 컨테이너가 사용되지 않는 캐시 객체를 삭제하기 위한 주기를 지정합니다. 그리고 **Cache size** 는 해당 컨테이너가 사용하는 캐시의 크기를 지정합니다. (참고적으로 bucket 은 하나의 table 안의 row 로 이해하시면 편리합니다.)

마지막으로 **EJB timer service setting** 은 EJB 표준에서 제시하는 EJB timer service 를 이용하기 위해서 사용합니다.

[Application servers](#) > [server1](#) > **EJB timer service settings**

Use this page to configure and manage the EJB timer service for a specific EJB container.

Configuration

---

**General Properties**

**Scheduler Type**

☒ Use internal EJB timer service scheduler instance

Data source JNDI name

jdbc/DefaultEJBTimerDataSource

Data source alias

(none)

Table prefix

EJBTIMER\_

Poll interval

300

seconds

Number of timer threads

1

threads

☐ Use custom scheduler instance

Scheduler JNDI name

(none)

Apply

OK

Reset

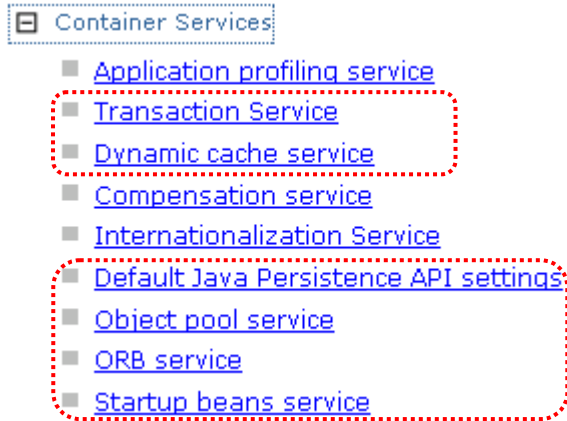
Cancel

**Related Items**

- [JAAS - J2C authentication data](#)
- [Schedulers](#)

#### Part 4. 컨테이너 서비스 설정 옵션

간단한 EJB container 에 대한 설명을 넘어오고, 다음으로 **Container services** 메뉴를 클릭하면 다양한 컨테이너 서비스 메뉴를 확인할 수 있습니다. 여기서도 마찬가지로 전체적으로 다 설명드리기 보다, 잘 사용되지 않는 것들도 있으므로 사용자들이 사용할 만한 중요한 것을 선별해서 설명해 드리도록 하겠습니다.



먼저 살펴볼 것은 가장 많이 사용되는 **Transaction service** 입니다. 해당 메뉴를 클릭하면 다음과 같은 세부메뉴를 살펴볼 수 있습니다.

[Application servers](#) > [server1](#) > Transaction service

Use this page to specify settings for the transaction service. The transaction service is a server runtime component that can coordinate updates to multiple resource managers to ensure atomic updates of data. Transactions are started and ended by applications or the container in which the applications are deployed.

**Transaction log directory** 는 트랜잭션 로그가 기록되는 디렉토리의 위치를 지정합니다. 이 옵션을 지정하지 않으면 기본적으로 `app_server_root/ tranlog/cell_ name/node_ name/server_ name` 디렉토리에 트랜잭션 로그가 기록됩니다. 트랜잭션 로그는 보통 XA 트랜잭션의 수행중에 예기치 않은 오류가 발생하는 경우를 대비하여 기록되고, 오류가 발생되면 해당 로그를 참조하여 WAS에

서 해당 트랜잭션을 복구할 수 있습니다.

**Total transaction lifetime timeout** 은 트랜잭션이 수행될 수 있는 최대 시간을 지정합니다. 해당 옵션에서 지정된 시간이 지나도 수행이 끝나지 않은 트랜잭션이 있다면 트랜잭션이 가지고 있는 타임아웃 플래그에 마크되며, 이는 곧 해당 트랜잭션이 무사히 완료 될 수 있다고 해도 roll back 된다는 것을 의미합니다. 단, 여기서 많은 분들이 오해하고 계시고 있는 점이 하나있는데 트랜잭션이 수행되는 중간에 여기서 지정된 타임아웃을 넘는다고 해도, 트랜잭션이 바로 중단되고 roll back 되는 것은 아닙니다. DB의 무결성을 훼손하지 않기 위하여 WAS에서는 가급적 강제 제어를 하지 않습니다. 오직 실질적인 타임아웃을 가지는 트랜잭션의 롤백은 트랜잭션 경계(boundary)에서 발생될 수 있습니다. 만약 무엇인가가 스레드에서 트랜잭션 경계로 이동하기 위한 처리를 막는다면 (행이나 데드락 같은 것들), 타임아웃 플래그가 이미 체크되어 있다고 해도 이것들은 트랜잭션 roll back 을 지연시킵니다. 그러나 타임아웃 발견은 타임아웃이 정확한 시간에 발견되는 것을 보장하기 위하여 다른 스레드에서 수행되므로 로그 상에서 타임아웃이 넘는다는 것을 확인할 수 있습니다. 다시 강조하지만 그렇다고 하여도 해당 트랜잭션이 바로 중단되지는 않습니다.

**Async response timeout** 은 인바운드 Web Services Atomic Transaction (WS-AT) 응답을 기다리는 대기시간의 최대 값을 의미하며 **Client inactivity timeout** 는 리모트 client 부터 트랜잭션 요청중에 client 가 활동하지 않을 수 있는 최대 시간을 의미합니다.

**Maximum transaction timeout** 은 트랜잭션이 서버에서 수행될 수 있는 최대 시간을 지정합니다. 이거 어디선가 들어본 이야기 같지 않나요? 본 강좌를 잘 읽었으면 아시겠지만 바로 위에서 이야기한 Total transaction lifetime timeout 과 동일한 이야기 입니다. 타임아웃의 최대 값은 Maximum transaction timeout 에 의해서 제한되어 지기 때문에 Maximum transaction timeout 은 반드시 Total transaction lifetime timeout 과 같거나 큰 값이어야 합니다. 예를 들어, total transaction timeout 을 500 으로 설정하고 maximum transaction timeout 을 300 으로 설정했다면 트랜잭션은 300 초 후에 타임아웃 됩니다.

트랜잭션 타임아웃에 대한 좀 더 세밀한 구분을 위해서는 하단의 표를 참조하시기 바라겠습니다.  
(참조: WAS v6.1 InfoCenter)

타임아웃	영향 받는 트랜잭션
Maximum transaction timeout	Total transaction lifetime timeout 이나 application component timeout 에 의해서 영향받지 않는 서버에서 돌고 있는 모든 트랜잭션들이 트랜잭션들은 클라이언트에서 들어온 것 같은 서버의 외부에서 들어온 트랜잭션도 포함하고 있습니다.
Total transaction lifetime timeout	Application component timeout 으로 영향 받지 않는 서버에서 생성된 모든 트랜잭션, 다시 말하면, 자체적으로 타임아웃을 설정하지 않은 연관된 어플리케이션 컴포넌트를 의미합니다.

## Application component timeout

어플리케이션 컴포넌트로 지정된 트랜잭션

만약 컴포넌트가 container-managed bean 이라면 컴포넌트를 위하여 배치 설명자에 타임아웃 값을 설정해야 하며 만약, 컴포넌트가 bean-managed bean 이라면 `UserTransaction.setTimeout` 메소드를 사용하여 프로그램적으로 타임아웃을 설정해야 합니다.

그 다음으로 살펴볼 것은 **Dynamic cache service** 입니다. 동적 캐시 서비스(Dynamic cache service)는 이전에 살펴본 Servlet 캐시 같은 것을 말하며 세부 메뉴를 살펴보면 아시겠지만 캐시 프로바이더나 사이즈를 변경할 수 있는 작업을 할 수 있습니다. 뿐만 아니라, 필요하면 enable disk offload 옵션을 이용해서 캐시를 보존하기 위하여 디스크에 저장할 수도 있습니다.

[Application servers](#) > [server1](#) > [Dynamic cache service](#)

The dynamic cache service consolidates caching activities to improve application performance. By caching the response from servlets, Web services, Java(TM) Server Pages (JSP) files, and WebSphere(R) Application Server commands, the application server does not have to perform the same computations and back-end queries multiple times.

Configuration

**General Properties**

To enable servlet caching, go to [Web container](#) settings.

To enable portlet caching, go to [Portlet container](#) settings.

Cache provider  
Default dynamic cache

Cache size  
2000

Default priority  
1

**Memory Cache Size**

Limit memory cache size

Memory cache size  
MB

High threshold  
95 %

Low threshold  
80 %

**Disk Cache settings**

Enable disk offload

**Additional Properties**

External cache groups

Custom properties

Apply OK Cancel

뒤이어 **Default Java Persistence API settings** 입니다. 하단의 세부 메뉴를 확인하시면 아시겠지만 J2EE 5.0 부터 사용되는 JPA 프로바이더를 WebSphere 에서 구현한 것을 사용할 것인지, 오픈 소스인 Apache 에서 구현한 것을 사용할지 등을 선택하기 위한 설정입니다.

[Application servers](#) > [server1](#) > [Default Java Persistence API settings](#)

The Java(TM) Persistence API (JPA) defines the management of persistence and object-relational mapping for the Java(TM) Platform, Enterprise Edition (Java(TM) EE) environment. Use this page to configure the default JPA settings for this server. These JPA settings are used for the persistence unit of an application only when the application does not define the JPA settings for that persistence unit. Application persistence settings always override the settings on this page.

Configuration

**General Properties**

Default persistence provider

Select a default persistence provider that is included with WebSphere Application Server

Default persistence provider  
com.ibm.websphere.persistence.PersistenceProviderImpl

com.ibm.websphere.persistence.PersistenceProviderImpl

org.apache.openjpa.persistence.PersistenceProviderImpl

Default JTA data source JNDI name  
(none)

Default non-JTA data source JNDI name  
(none)

**Related Items**

Data Sources

Apply OK Reset Cancel



**Object pool service** 옵션은 글자 그대로 객체 풀 서비스를 사용할 것이냐를 지정하는 옵션입니다. 기본적으로 사용에 체크되어 있으며 성능에 영향을 줄 수 있는 옵션입니다.

[Application servers](#) > [server1](#) > **Object pool service**

Many Java objects can be created once, used, and then reused. The object pool service manages object pool resources that are used by the application server.

Configuration

**General Properties**

☒ Enable service at server startup

**Additional Properties**

■ [Custom properties](#)

Apply OK Reset Cancel

그 다음 컴포넌트 서비스를 체크해보면 **ORB service** 입니다. ORB 서비스는 EJB 를 이용하실 때 상당히 중요한 서비스 입니다. 그것은 EJB 가 결국, Object 이며 ORB 서비스 라는 것은 Object Request Broker 서비스를 의미합니다. 다시말하면 EJB 라는 객체의 요청에 대한 브로커 서비스이죠.^^&; EJB 는 객체에 대한 요청을 CORBA 방식으로 전달 하기 때문에 EJB를 사용하는 어플리케이션의 경우에는 상당히 중요합니다.

[Application servers](#) > [server1](#) > **ORB service**

Use this page to configure the object request broker (ORB).

Configuration

**General Properties**

\* Request timeout  
180 seconds

\* Request retries count  
1 retries

\* Request retries delay  
0 milliseconds

\* Connection cache maximum  
240 connections

\* Connection cache minimum  
100 connections

☐ ORB tracing

\* Locate request timeout  
180 seconds

먼저, **Request timeout** 은 요청 메시지에 대한 타임아웃 값을 설정합니다. 커맨드라인에서 시스템 설정(system property)인 `com.ibm.CORBA.RequestTimeout` 을 이용해서 설정하는 것도 가능합니다.

**Request retries count** 는 서버가 실패되었을 경우에 ORB 가 요청을 전달하려고 시도하는 최대 회수를 지정합니다. 그리고 Request retries delay 는 위의 옵션에서 재시도를 하기 전에 얼마간의 지연 후에 재시도 할 것인지 시간을 설정하는 옵션입니다.

☒ Pass by reference

**Thread Pool Settings**

☐ Use the [ORB.thread.pool](#) settings associated with the Thread Pool Manager (recommended).

☒ Use the [Thread Pool Settings](#) directly associated with the ORB service.

**Pass by reference** 옵션은 ORB 가 파라미터를 어떠한 방식으로 보낼지를 설정하는 것입니다. 해당 옵션을 선택하면 값을 전달하는(객체를 복사해서 전달하는) 것을 피하고 레퍼런스를 통해서 파라미터를 전달 합니다. 객체를 복사해서 전달하는 것보다는 당연히 메모리 레퍼런스를 통해서 전달하는 것이 성능상의 이점이 있습니다. 그렇기 때문에, EJB 어플리케이션에서 성능을 위해 많이 사용되는 튜닝 파라미터 중에 하나입니다.

다음으로 **Thread Pool Setting** 옵션은 ORB 객체들에 대한 스레드 풀 설정을 위한 옵션입니다. 풀(pool)에 대한 개념은 이미 아시겠지만 간단히 다시 설명 드리자면, 자원에 대한 요청이 있을 경우 해당 요청이 올 때마다 해당 객체를 매번 만들고, 없애고 하는 것이 아니라 풀 이라는 개념을 활용하는 것입니다. 사용되지 않는 객체를 바로 없애는 것이 아니라 풀에 보관해 두었다가 새로운 요청이 들어오면 이미 만들어서 사용되었지만 지금은 사용되지 않아 풀에 보관해 두었던 객체를 반환해 줍니다. 이를 통해서 한정된 객체를 계속 재사용 하는 것입니다. 이와 같은 방법으로 생성 및 삭제에 대한 오버헤드를 없앨 수 있고, 객체가 무한정 증가하는 일이 없으므로 성능 및 자원을 최적화 할 수 있습니다. 하단의 세부 설정 메뉴를 확인하시면 아시겠지만 스레드 풀의 최소, 최대 값을 설정할 수 있으며 Thread inactivity timeout 을 통해서 활동이 없을 경우에 지정된 일정 시간 이후에 해당 Thread 에 대한 자원을 해제하게 할 수도 있습니다. 그리고 Allow thread allocation beyond maximum thread size 옵션을 체크하시면 최대 값을 넘은 요청이 들어와도 허용할 수 있습니다. (단, 이 경우에는 당연히 자원 활용률이 떨어질 수 있습니다.)

[Application servers](#) > [server1](#) > [ORB service](#) > [ORB.thread.pool](#)

Use this page to specify a thread pool for the server to use. A thread pool enables server components to reuse threads instead of creating new threads at run time. Creating new threads is typically a time and resource intensive operation.

Configuration

#### General Properties

Name  
ORB.thread.pool

Description

Minimum Size  
10 threads

Maximum Size  
50 threads

Thread inactivity timeout  
3500 milliseconds

☐ Allow thread allocation beyond maximum thread size

Apply OK Reset Cancel

#### Additional Properties

Custom properties

이번 파트에서 마지막으로 확인할 서비스는 **Startup beans service** 입니다. 해당 서비스는 WebSphere 에서 제공하는 Startup beans 서비스를 사용할 경우에 설정합니다. Startup beans 란 EJB 이면서 해당 어플리케이션이 시작되면 자동으로 시작되는 beans 를 의미합니다. 아시겠지만 Java 의 특징은 lazy loding 입니다. 즉, 요청이 있어야만 그때 메모리에 해당 class 를 로드한다는 것이죠. 그렇기 때문에 Startup beans 의 경우는 보통 요청 전에 메모리에 로드 할 것이 있거나 사전에 수행할 일이 있으면 많이 사용되는 beans 입니다.

[Application servers](#) > [server1](#) > **Startup beans service**

Startup beans are session beans that run business logic through the invocation of start and stop methods when applications start and stop. If the startup beans service is disabled, the automatic invocation of the start and stop methods does not occur for deployed startup beans when the parent application starts or stops. This service is disabled by default. Enable this service only when you want to use startup beans.

Configuration

#### General Properties

☐ Enable service at server startup

Apply

OK

Reset

Cancel

#### Additional Properties

■ [Custom properties](#)

## Part 5. JVM 설정 옵션

이번 파트에서는 사람들이 가장 많이 변경할 수 밖에 없는 JVM 옵션에 대해서 설명하도록 하겠습니다. 이미 몇 번 강조해 드렸지만, WAS 서버는 결국 JVM instance 이기 때문에 JVM 에 대한 옵션 설정이 가장 빈번하고 중요한 작업 중에 하나입니다.

Server Infrastructure 밑의 Java and Process Management 를 클릭하면 다음과 같은 3개의 카테고리를 확인할 수 있습니다. **Class loader** 카테고리는 현재의 클래스 로더 구조에서 새로운 사용자 클래스 로더를 등록할 수 있는 옵션입니다. 사용자 클래스 로더는 Sheard library 와 연결하여 클래스 로더 가장 상위에 해당 라이브러리나 클래스를 위치시킬 때 보통 사용되지만 그렇게 많이 사용되는 케이스는 없습니다. 따라서 여기서는 간단히 언급만 하고 넘어가도록 하겠습니다. 본격적인 것은 다음 카테고리인 Process definition 부터 다루도록 하겠습니다.

### Server Infrastructure

#### Java and Process Management

- [Class loader](#)
- [Process definition](#)
- [Process execution](#)

#### Administration

**Process definition** 카테고리를 클릭하면 하단의 그림과 같은 세부 메뉴를 확인할 수 있습니다. 세부 메뉴중에서 제일 핵심은 바로 Java Virtual Machine 입니다. 바로 JVM 이죠.^.^&;

[Application servers](#) > [server1](#) > **Process definition**

Use this page to configure a process definition. A process definition defines the command line information necessary to start or initialize a process.

Configuration

#### General Properties

Executable name

Executable arguments

#### Additional Properties

- [Java Virtual Machine](#)
- [Environment Entries](#)
- [Process execution](#)
- [Process Logs](#)
- [Logging and tracing](#)

Java virtual Machine 메뉴를 클릭하면 하단처럼 세부 설정 메뉴를 확인하실 수 있습니다.

[Application servers](#) > [server1](#) > [Process definition](#) > [Java Virtual Machine](#)

Use this page to configure advanced Java(TM) virtual machine settings.

Configuration

Runtime

General Properties

Classpath

Boot Classpath

☐ Verbose class loading

☐ Verbose garbage collection

☐ Verbose JNI

Initial heap size

 MB

Maximum heap size

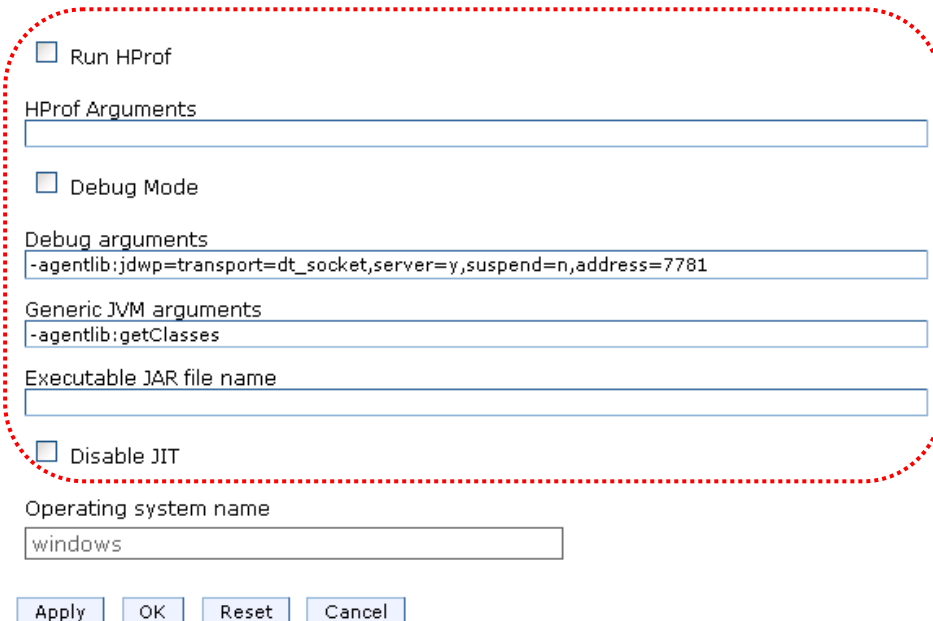
 MB

먼저 **Classpath** 옵션은 Java 를 하시는 분들은 바로 아시겠지만 Class 에 대한 Classpath 를 추가할 수 있는 메뉴입니다.

다음은 **Verbose class loading**, **Verbose garbage collection**, **Verbose JNI** 옵션입니다. 동일하게 앞에 Verbose 가 있으므로 세 개의 옵션의 의미를 쉽게 파악하실 수 있으실 것입니다. Verbose class loading 은 class loading 을 로깅하는 것이며, Verbose garbage collection 은 GC(garbage collection) 을, Verbose JNI 는 네이티브 메소드 호출을 로깅합니다.

**Initial heap size** 와 **Maximum heap size** 옵션은 이름 그대로 최초의 JVM의 heap 사이즈와 최대로 늘어날 수 있는 JVM 의 heap 사이즈를 설정 할 수 있습니다. JVM 에서 가장 민감한 것 중에 하나가 Memory, 특히 heap 사이즈 입니다. 따라서 JVM heap 을 얼마나 잘 튜닝하냐가 성능에 가장 민감한 포인트중의 하나입니다. 너무 적거나 많지 않은 JVM heap 설정을 해야하는데, 말

은 쉽지만 상당히 어려운 일이므로 운영환경이라면 오픈하기 전에 많은 테스트를 통해서 튜닝하 시기를 바라겠습니다. (사실, 성능 말고도 이부분을 잘 튜닝해야 하는 이유는 쉽사리 OOM(Out of Memory) Exception 같은 것들을 야기시킬 수도 있기 때문입니다.)



**Run HProf** 는 옵션 그대로 HProf 프로파일을 지원하는 옵션입니다. 다른 사용자 지정 프로파일을 사용하실려면 HProf Arguments 에 별도 설정을 통하여 사용가능 합니다.

그 뒤로는 **Generic JVM arguments** 옵션입니다. 해당 옵션이 바로 JVM 의 사용자 설정(Custom property) 을 넣을 수 있는 옵션입니다. -D, -X 나 -XX 로 시작되는 다양한 JVM 의 사용자 설정 을 입력할 수 있습니다. 예를 들어 그냥 해당 옵션에 '-Xnoclassgc' 이런 식으로 사용자 설정을 넣기만 하면 됩니다. (해당 옵션은 class gc 를 하지 않는다는 JVM 의 사용자 설정입니다.)

JVM 설정의 마지막으로 **Disable JIT** 옵션은 just-in-time (JIT) 컴파일러를 사용하지 않는다는 옵션 입니다. JIT의 경우는 JVM 의 성능을 높이기 위한 하나의 방법으로 나온 것으로 많이 사용되는 코드를 미리 기계어로 컴파일하여 성능을 높이는 방법입니다. Java 는 기본적으로 interpreter 언어입니다. 즉, Java 코드를 컴파일해서 바로 실행하는 것이 아니라 Java 코드를 컴파일 하면 실행 이 가능한 기계어가 나오는 것이 아니라 바이트코드 라는 중간 코드가 나옵니다. 바이트코드는 특정 컴퓨터를 위한 기계어가 아니고 컴퓨터 아키텍처상 다른 기종으로 이식이 가능한 언어입니다. 이처럼 바이트코드라는 중간 코드를 만들어 내는 것은 Java 가 가지고 있는 “write once run any where” 라는 사상 때문입니다. 한번 만든 코드를 모든 곳(운영체제)에서 사용하겠다는 의미죠. 그렇기 때문에 이 바이트코드는 다시 JVM 에서 해석되거나 실행됩니다. 이런 작업을 거치기 때문에 바로 해당 운영환경에 맞는 최적화된 기계어로 번역되어 실행되는 C 같은 프로그램 보다는 성능상의 이슈가 있습니다. 이런 성능상의 이슈를 극복하기 위해 다양한 방법들이 동원되었으며 그 중의 하나가 바로 JIT 컴파일러 입니다. JIT 컴파일러는 바이트코드가 실행될 때 더 나은 성능을 위하여 코드의 일부나 전체를 기계어(native machine code) 로 변환하여 사용합니다. 바이트코

드 해석의 장점을 유지한 채 정적인 컴파일의 성능을 도달하거나 능가하기 위함입니다.

**Disable JIT** 옵션은 과거 버전의 JDK 의 경우 JIT 컴파일러에 문제가 많아서 많이 사용되었으나 최신 버전의 JDK 에서는 거의 문제가 발생하지 않으므로 해당 옵션을 사용할 일은 거의 없을 것입니다.

그 뒤로 살펴볼 것은 **Process execution** 에 대한 세부 설정 옵션입니다. 여기서 어느정도 감을 잡으셨겠지만 Process definition 안에도 Process execution 이라는 옵션이 있고 이번 파트의 제일 처음의 Java and Process Management 카테고리 안에도 Process execution 이 있습니다. 이는 동일 메뉴이며 관리의 편의성을 위한 구조에 따라서 Java and Process Management 카테고리에서도 확인하고 설정을 변경할 수 있으며 Process definition 에서도 다른 JVM 옵션과 함께 작업할 수 있도록 배려(?)한 것입니다.

[Application servers](#) > [server1](#) > [Process definition](#) > **Process execution**

Use this page to configure additional process execution settings. These settings are not used on the Microsoft(R) Windows(R) platform.

Configuration

---

**General Properties**

Process Priority  
20

UMASK  
022

Run As User  
[Empty]

Run As Group  
[Empty]

Run In Process Group  
0

Apply OK Reset Cancel

**Run As User, Run As Group** 을 사용하여 어떤 사용자와 그룹으로 WAS 를 시작하여도 지정된 사용자와 그룹으로 권한을 제어할 수 있습니다. 예를 들어서, root 권한을 이용해서 WAS 를 시작하여도, 이 옵션을 사용하면 해당 옵션에서 사용된 사용자와 권한으로 WAS 서버를 시작하게 됩니다.

그 뒤에 있는 카테고리는 **Logging and Tracing** 입니다. 해당 카테고리를 클릭하면 하단과 같이 각 로깅 설정 메뉴를 확인할 수 있습니다.

[Application servers](#) > [server1](#) > [Process definition](#) > **Logging and Tracing**

Use this page to select a system log to configure, or to specify a log detail level for components and groups of components. Use log levels to control which events are processed by Java logging.

**General Properties**

- [Diagnostic Trace](#)
- [JVM Logs](#)
- [Process Logs](#)
- [IBM Service Logs](#)
- [Change Log Detail Levels](#)





다음으로 **JVM Logs** 옵션입니다. 해당 옵션은 JVM 로그에 대한 설정을 할 수 있습니다. JVM 로그라는 것은 결국 WAS 로그와 동일한 이야기 입니다.

[Application servers](#) > [server1](#) > [Process definition](#) > [Logging and Tracing](#) > **JVM Logs**

Use this page to view and modify the settings for the Java virtual machine (JVM) System.out and System.err logs for a managed process. The JVM logs are created by redirecting the System.out and System.err streams of the JVM to independent log files. The System.out log is used to monitor the health of the running application server. The System.err log contains exception stack trace information that is used to perform problem analysis. One set of JVM logs exists for each application server and all of its applications. JVM logs are also created for the deployment manager and each node manager. Changes on the Configuration panel apply when the server is restarted. Changes on the Runtime panel apply immediately.

Configuration Runtime

General Properties

System.out

\* File Name:  
\${SERVER\_LOG\_ROOT}/System.out

File Formatting  
Basic (Compatible)

Log File Rotation

☒ File Size  
Maximum Size  
1 MB

☐ Time  
Start Time  
24  
Repeat Time  
24 hours

Maximum Number of Historical Log Files. Number in range 1 through 200.  
1

Installed Application Output

☒ Show application print statements

☒ Format print statements

JVM 로그 옵션 중에 **Log File Rotation** 카테고리 안에 포함된 옵션들은 모두 로그 파일을 어떻게 로테이션 할 것인가에 대한 옵션입니다. 여기서 로테이션이라는 의미는 1개의 로그 파일을 사용하는 것이 아니라 지정된 개수의 로그파일을 돌아가면서 사용하는 것을 의미합니다. 예를 들어 File Size 가 일정 이상되고, Maximum Number of Historical Log Files 에 지정된 개수가 1 개 초과 라면 해당 로그 뒤에 시간을 붙여서 새로 만든 다음에 본래의 JVM 로그는 내용을 클리어 하여 새로 로깅하게 됩니다. 이런 식으로 예전 로그를 Maximum Number of Historical Log Files 에 지정된 개수만큼만 남기게 되는 것입니다. 지정된 개수를 넘게 되는 내용이 로깅되면 순차적으로 처음에 만들었던 로그가 자동으로 삭제되고 로그 파일의 개수를 유지합니다. 여기서는 파일 사이즈 대신에 시간 주기를 지정할 수도 있으며, 파일 사이즈와 시간 주기를 같이 사용하여 로그 파일 로테이션을 구분할 수도 있습니다.

**Installed Application Output** 카테고리에 해당하는 옵션은 Java의 System.out 이나 System.err API 를 이용한 어플리케이션의 출력에 관련된 설정입니다. Show application print statements 옵션 은 print 나 println 같은 메소드의 출력에 대한 로깅여부를 결정할 수 있는 설정이며 Format print statements 옵션은 어플리케이션에서 API 를 이용해서 찍는 출력이라고 해도 WebSphere 에 맞는 포맷으로 맞춰줄 수 있는 옵션입니다.

다음에 나오는 **Process Logs** 는 프로세스에 대한 native 로그의 파일을 지정하는 옵션입니다. 또한 여러 로그의 위치도 지정할 수 있습니다.

[Application servers](#) > [server1](#) > [Process definition](#) > **Process Logs**

Use this page to view or modify settings to specify the files to which standard out and standard error streams write. The process logs are created by redirecting the standard out and standard error streams of a process to independent log files. Native code writes to the process logs. These logs can also contain information that relates to problems in native code or diagnostic information written by the JVM. One set of process logs is created for each application server and all of its applications. Process logs are also created for the deployment manager and each node manager. Changes on the Configuration panel apply when the server is restarted. Changes on the Runtime panel apply immediately.

Configuration **Runtime**

**General Properties**

\* Stdout File Name  
\${SERVER\_LOG\_ROOT}/native\_stdout.log

\* Stderr File Name  
\${SERVER\_LOG\_ROOT}/native\_stderr.log

Apply OK Reset Cancel

마지막으로 살펴볼 수 있는 로그는 **IBM Service Logs** 로 향후 장애나 문제상황이 발생했을 경우 그 원인을 분석하기 위하여 Lab 에서 분석할 수 있는 서비스에 대한 로그 입니다. 즉, Lab 에서 필요한 데이터로 포맷자체가 Encoding 되어 있기 때문에 굳이 확인하고 보실 필요는 없습니다.

[Application servers](#) > [server1](#) > [Process definition](#) > [Logging and Tracing](#) > **IBM Service Logs**

Use this page to configure the IBM service log, also known as the activity log. The IBM service log contains both the application server messages that are written to the System.out stream and special messages that contain extended service information that you can use to analyze problems. One service log exists for all Java virtual machines (JVMs) on a node, including all application servers and their node agent, if present. A separate activity log is created for a deployment manager in its own logs directory. The IBM Service log is maintained in a binary format. Use the Log Analyzer or Showlog tool to view the IBM service log.

Configuration

**General Properties**

☒ Enable service log

\* File Name:  
\${LOG\_ROOT}/activity.log

\* Maximum File Size  
2 MB

☒ Enable Correlation ID

Apply OK Reset Cancel

## Part 6. 기타 설정 옵션

길고 긴 서버 관련 설정 강좌의 마지막은 기타 설정 관련 옵션입니다. 기타라고는 하지만 중요한 것도 있으니 길게 느껴지시더라도 조금만 더 참고 강좌를 끝까지 잘 마무리 하시기 바라겠습니다. 거의 다 왔습니다. 파이팅! ^^&;

기타 설정 옵션 파트에서 처음으로 소개할 것은 **Ports** 입니다. 하단의 첨부된 그림을 보시면 아시겠지만 WAS 의 경우는 다양한 포트를 사용합니다. 해당 포트가 무엇과 매핑되어 있는지 확인할 수 있으며 포트 변경도 가능합니다.

### Communications

#### Ports

Port Name	Port	Details
BOOTSTRAP_ADDRESS	2814	
SOAP_CONNECTOR_ADDRESS	8884	
ORB_LISTENER_ADDRESS	9102	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9415	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9414	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9413	
WC_adminhost	9064	
WC_defaulthost	9083	
DCS_UNICAST_ADDRESS	9358	
WC_adminhost_secure	9047	
WC_defaulthost_secure	9446	
SIP_DEFAULTHOST	5067	
SIP_DEFAULTHOST_SECURE	5066	
SIB_ENDPOINT_ADDRESS	7280	
SIB_ENDPOINT_SECURE_ADDRESS	7289	
SIB_MQ_ENDPOINT_ADDRESS	5561	
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5581	
IPC_CONNECTOR_ADDRESS	9636	

여기서 중요한 포트만 간단하게 말씀드리자면 BOOTSTRAP\_ADDRESS 포트는 부트스트랩 서비스 포트 입니다. 리모트 EJB 호출을 하거나 JMS Message Engine 을 찾는 경우등 대부분의 서비스에서 맨 먼저 접근하는 포트입니다. 만약 리모트 EJB 호출을 위해 BOOTSTRAP\_ADDRESS 포트로 접근을 하면 EJB 호출을 위한 ORB\_LISTENER\_ADDRESS 포트를 리턴해주고 다시 해당 포트를 이용해서 실제적으로 EJB 호출이 이루어집니다. 마찬가지로 JMS Message Engine 을 찾기 위해 BOOTSTRAP\_ADDRESS 포트로 접근을 하면 JMS Message Engine 에 접속할 수 있는 SIB\_ENDPOINT\_ADDRESS 포트를 리턴해주고, 이 포트를 이용해서 실제적으로 JMS Message Engine 에 접속할 수 있습니다. 결국, BOOTSTRAP\_ADDRESS 포트는 다양한 서비스에 대한 대표적

인 게이트웨이 같은 포트입니다.

WC\_defaulthost 와 WC\_defaulthost\_secure 는 실제적인 HTTP 서비스 포트 입니다. 여기서 맵핑 된 해당 포트로 요청을 보내야지만 HTTP 나 JSP, Servlet 등의 서비스를 호출 할 수 있습니다. 즉, <http://127.0.0.1:9080/ServletTest> 이런 식의 URL 에서 지정되는 9080 포트가 바로 WC\_defaulthost 입니다. WC\_defaulthost 와 WC\_defaulthost\_secure 의 차이는 보안을 사용하느냐 안하느냐의 차이 입니다. 다시 이해하기 쉽게 말하면 SSL 을 적용한 HTTPS 프로토콜의 경우에는 WC\_defaulthost\_secure 포트를 사용하며 그렇지 않은 경우에는 WC\_defaulthost 포트를 사용하게 됩니다.

WC\_adminhost 와 WC\_adminhost\_secure 포트는 관리콘솔에 접근하기 위한 포트이며 WC\_adminhost 와 WC\_adminhost\_secure 포트의 차이는 위에서 말한 것과 동일하게 보안 사용 여부의 차이입니다.

다음 설정에 관련된 옵션은 Performance 에 관련된 **PMI(Performance Monitoring Infrastructure)** 와 **Performance and Diagnostic Advisor configuration** 입니다. 두 가지 모두 이전에는 Tivoli 라는 브랜드의 별도 제품이었으나 시간이 지나면서 WAS 내부에 포함된 제품입니다.

#### Performance

- [Performance Monitoring Infrastructure \(PMI\)](#)
- [Performance and Diagnostic Advisor Configuration](#)

PMI 의 경우는 말 그대로 성능 모니터링을 할 수 있는 기능입니다. 즉, WAS 에서 성능에 관련된 포인트들을 그래프나 표로 실시간 확인하실 수 있습니다. 하단의 세부 설정 메뉴에서도 확인할 수 있지만 서버에 있는 PMI 설정 옵션은 PMI 사용 여부라던가, 어느 단계까지 자세하게 모니터링 할 것인가를 설정할 수 있습니다.

[Application servers](#) > [server1](#) > **Performance Monitoring Infrastructure (PMI)**

Use this page to configure Performance Monitoring Infrastructure (PMI)

Runtime

Configuration

General Properties

☒ Enable Performance Monitoring Infrastructure (PMI)

☐ Use sequential counter updates

Currently monitored statistic set

☐ None  
No statistics are enabled.

☒ Basic  
☐ Provides basic monitoring, including Java EE and the top 38 statistics.

☐ Extended  
☐ Provides extended monitoring, including the basic level of monitoring plus workload monitor, performance advisor, and Tivoli resource models.

☐ All  
☐ All statistics are enabled.

☐ Custom  
Provides fine-grained control to selectively enable statistics.

Apply

OK

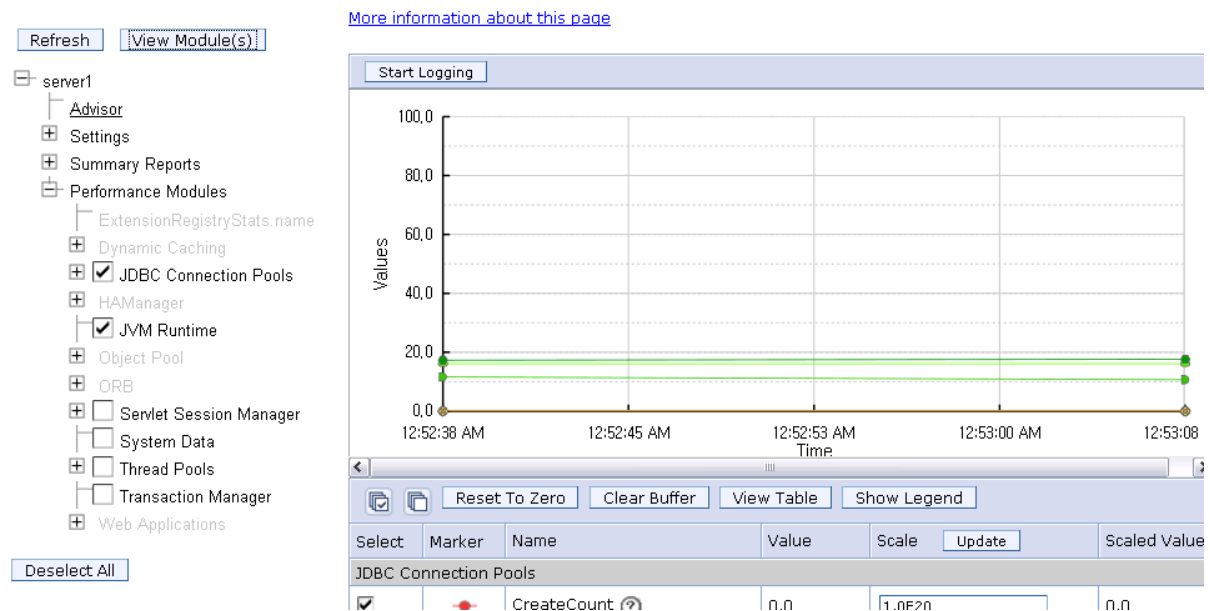
Reset

Cancel

만약 이 메뉴에서 PMI 를 enable 로 설정했다면 관리콘솔에서 Monitoring and Tuning > Performance Viewer > Current activity 메뉴를 클릭하여 하단과 같은 성능 모니터링 화면을 확인할 수 있습니다.

[Tivoli Performance Viewer](#) > **server1**

Use this page to view and refresh performance data for the selected server, change user and log settings, and view summary reports and information c performance modules.



Performance and Diagnostic Advisor configuration 은 Performance and Diagnostic Advisor 에 관련된 설정을 할 수 있습니다. Performance and Diagnostic Advisor 는 성능 데이터를 수집하여 분석 및 진단, 가이드를 제공하는 기능입니다.

[Application servers](#) > **server1** > Performance and Diagnostic Advisor Configuration

The Performance and Diagnostic Advisor analyzes PMI data and receives notifications regarding performance and diagnostic information from components. Use this page to specify settings for the Performance and Diagnostic Advisor. Performance issues can be related to memory leaks in the system. Use the Memory Dump Diagnostic for Java tool, a separate memory leak analysis utility, for detecting memory leaks.

Runtime Configuration

**General Properties**

☐ Enable Performance and Diagnostic Advisor Framework (Runtime Performance Advisor)

Calculation Interval  
4 minutes

Maximum warning sequence  
1

\* Number of processors  
2

\* Minimum CPU For Working System  
50

\* CPU Saturated  
90

Apply OK Reset Cancel

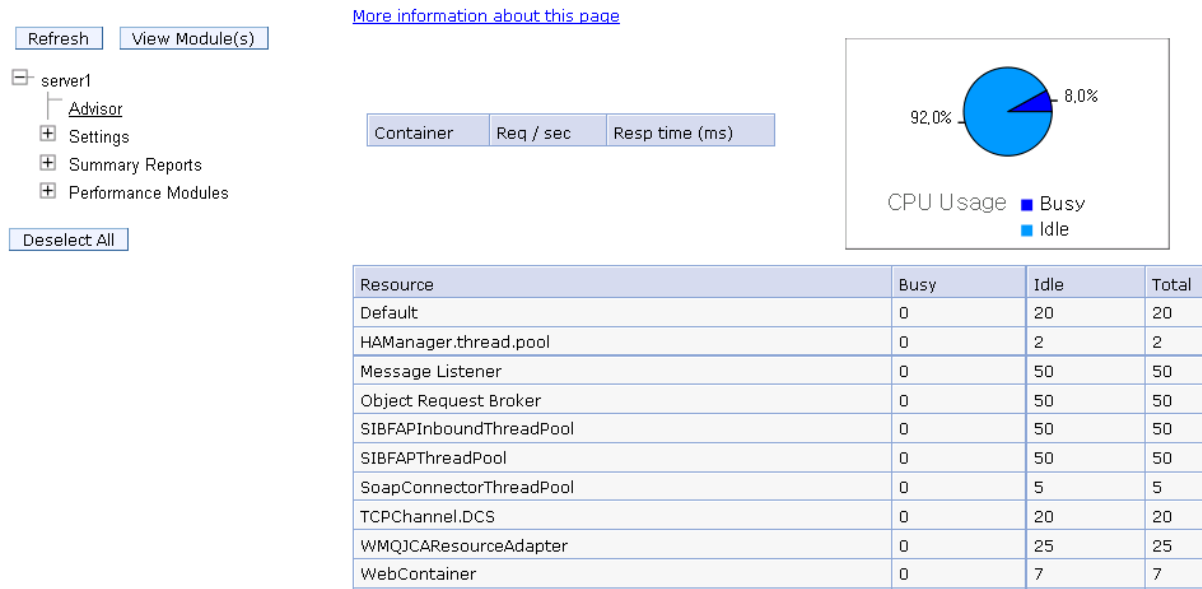
**Additional Properties**

- [Performance and Diagnostic Advice configuration](#)

만약 이 메뉴에서 Performance and Diagnostic Advisor 을 Enable 한다면 관리콘솔에서 Monitoring and Tuning > Performance Viewer > Current activity 메뉴를 클릭하여 Advisor 를 선택 하면 하단의 그림과 같은 분석 현황표와 가이드를 제공받을 수 있습니다.

[Tivoli Performance Viewer](#) > **server1**

Use this page to view and refresh performance data for the selected server, change user and log settings, and view summary reports and information performance modules.



이제 거의 이번 강좌의 마무리가 보이네요. 마지막으로 살펴볼 옵션은 Additional Properties 카테고리입니다.

### Additional Properties

- [Class loader viewer service](#)
- [Endpoint listeners](#)
- [Debugging service](#)
- [Thread pools](#)
- [Reliable messaging state](#)
- [Web server plug-in properties](#)

먼저 **Class loader viewer service** 옵션입니다.

[Application servers](#) > **server1** > **Class loader viewer service**

Use this page to enable or disable a class loader viewer service that tracks the classes loaded.

Configuration

### General Properties

☐ Enable service at server startup

Apply

OK

Reset

Cancel

Class loader viewer 서비스란 말 그대로 클래스 로더의 구조를 관리콘솔에서 보여주는 서비스 입니다. 해당 서비스를 시작 시키고 Troubleshooting > Class loader viewer 메뉴를 선택하면 하단처럼 클래스 로더의 구조를 확인할 수 있으며, 또한 하나의 class 가 어느 클래스 로더에 의해서 로드되었는지도 확인할 수 있습니다. 클래스를 못 찾는다는 Exception 이 발생했을 경우에 해결을 위해서 자주 사용되는 기능입니다.

### [Enterprise Applications Topology](#) > **Class loader viewer**

Use this page to examine the class loaders visible to a Web module (.war file) or enterprise application. You can use this page to determine which class loaders loaded files of a module and to diagnose problems with class loading.

Hierarchy

Search Order

Export

Table View

Search

#### **ClassLoader - Search Order**

- 1 - JDK Extension - sun.misc.Launcher\$ExtClassLoader
- 2 - JDK Application - sun.misc.Launcher\$AppClassLoader
- 3 - OSGI - org.eclipse.osgi.internal.baseadaptor.DefaultClassLoader
- 4 - Extension - com.ibm.ws.bootstrap.ExtClassLoader
- 5 - WAS Protection Class Loader - com.ibm.ws.classloader.ProtectionClassLoader
- 6 - Server-associated - com.ibm.ws.classloader.ExtJarClassLoader
- 7 - Module - com.ibm.ws.classloader.CompoundClassLoader

그 다음으로 **Debugging Service** 는 디버깅 서비스에 관련된 옵션입니다.

### [Application servers](#) > [server1](#) > **Debugging service**

Specifies a model of the attributes needed for debugging a JVM and various components, such as the BSF manager.

Configuration

#### **General Properties**

☐ Enable service at server startup

\* JVM debug port

7781

\* JVM debug arguments

-agentlib:jdwp=transport=dt\_soc

#### **Debug class filters**

com.ibm.servlet.\*  
com.ibm.ws.\*  
com.ibm.som.\*  
com.ibm.CORBA.\*  
com.ibm.debug.\*

Add>

Apply

OK

Reset

Cancel

The additional properties will not be available until the general properties for this item are applied or saved.

#### **Additional Properties**

■ Custom properties

**Thread pools** 메뉴는 해당 WAS 에서 사용되는 모든 스레드 풀을 확인할 수 있으며 여기서 직접 해당 스레드 풀의 최대, 최소 값을 수정할 수 있습니다.

[Application servers](#) > [server1](#) > **Thread pools**

Use this page to specify a thread pool for the server to use. A thread pool enables server components to reuse threads instead of creating new threads at run time. Creating new threads is typically a time and resource intensive operation.

Preferences

New Delete				
Select	Name	Description	Minimum Size	Maximum Size
You can administer the following resources:				
<input type="checkbox"/>	<a href="#">Default</a>		20	20
<input type="checkbox"/>	<a href="#">ORB.thread.pool</a>		10	50
<input type="checkbox"/>	<a href="#">SIBFAPInboundThreadPool</a>	Service integration bus FAP inbound channel thread pool	4	50
<input type="checkbox"/>	<a href="#">SIBFAPThreadPool</a>	Service integration bus FAP outbound channel thread pool	4	50
<input type="checkbox"/>	<a href="#">SIBJMSRAThreadPool</a>	Service Integration Bus JMS Resource Adapter thread pool	35	41
<input type="checkbox"/>	<a href="#">TCPChannel.DCS</a>		5	20
<input type="checkbox"/>	<a href="#">WMQCommonServices</a>	WebSphere MQ common services thread pool	1	40

마지막으로 **Web server plug-in properties** 옵션은 웹 서버를 위한 Plugin-cfg.xml 파일을 만들어 배포할 때 해당 WAS 서버에 대한 속성을 변경을 원하면 사용되는 옵션입니다.

[Application servers](#) > [server1](#) > **Web server plug-in properties**

Use this page to configure application server properties for a Web Server plug-in.

Configuration

### Web server plug-in properties

Server Role  
Primary

**Connection timeout**

☒ Use connection timeout  
Connection timeout  
0 seconds

**Read/Write timeout**

☒ Use read/write timeout  
Read/Write timeout  
0 seconds

**Maximum number of connections that can be handled by the Application server**

☐ Use maximum number of connections  
Maximum number of connections  
0 connections



위의 첨부된 그림에서 확인할 수 있는 것 처럼 해당 WAS 서버를 Primary 로 할지 Backup 으로 할지, Connection 타임아웃 이나 서버에서 제어할 수 있는 connection 의 최대 수치를 지정할 경우에 해당 옵션을 이용해서 웹 서버에 배포되는 Plugin-cfg.xml 파일을 제어할 수 있습니다.

여기까지 잘 이해가 되시나요? 다른 강좌에 비해 조금 긴 강좌이며 각각의 옵션에 대한 개략적인 설명 위주라 조금은 지루하셨나요? 그래도 어느 틈에 휘리릭... 이번 강좌의 마무리까지 왔네요. 수고하셨습니다. ^^& 비록 지루하고 긴 강의가 되었을 수도 있겠지만, WAS 서버에 대한 옵션을 간략히 살펴보고 대략적인 기능을 이해할 수 있어야 실전에서 필요한 옵션이 어느 것인지 감을 찾을 수 있습니다. 비록 바쁘시더라도 가급적 한, 두 번 계속 읽어보시고 현재 환경에서 도움이 될만한 옵션이 무엇인지, 어떻게 조정하면 좋을지에 대한 고민도 한번 해 보면, 보다 더 나은 강좌가 될 것이라고 생각합니다. 그럼 이번 강좌도 여기서 마무리 하도록 하겠습니다. 이만~~~~~ 휘리릭... ^^&

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](http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/welcome_nd.html)

참고 2) IBM WebSphere Application Server v7.0 InfoCenter

- Application server setting

[http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/urun\\_rappsvr.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/urun_rappsvr.html)

※이 자료의 저작권은 작성자에게 있으며 유포는 자유로이 허용되나 상업적으로 이용은 금합니다.