Real Alternative DBMS ALTIBASE, Since 1999

ALTIBASE & JEUS 연동 가이드

ALTIBASE 5

2010. 04

Document Control

Change Record

Date	Author	Change Reference
2010-01-28	swj0701	Created

Reviews

Date	Name (Position)

Distribution

Name	Location

목차

개요

본 문서는 ALTIBASE 5.3.3 과 JEUS6.0 을 연동하여 운영하기 위한 가이드 문서로써 JEUS6.0 을 설치하는 방법 및 환경변수 설정에 대해서 가이드를 제시한 후, ALTIBASE와 연동하는 방법에 대해서 기술하도록 한다.

JEUS 설치

JEUS를 설치하는 방법에는 콘솔모드를 이용한 설치 방법과 GUI 모드에서 설치하는 두 가지의 방법이 있으며, 본 절에서는 Unix/Linux 콘솔 모드에서 JEUS를 설치하는 과정과 Windows GUI 모드에서 JEUS를 설치하는 과정에 대해서 알아본다. JEUS6.0 은 JDK 5.0 Update 4(1.5.0_04) 이상이 설치되어 있어야 하며, 300M 이상의 하드디스크 여유공간을 필요로 한다.

기본 설치(콘솔모드)

다음의 하위 절은 텍스트 기반 쉘에서 실행하는 방법을 설명한다.

1. 티맥스소프트 홈페이지에서 Unix/Linux 버전의 JEUS 패키지를 다운로드 받는다.
2. JEUS를 설치할 서버에 업로드 한 후, jeus60-unix-generic.bin 파일이 있는 곳으로 이동한다.
3. 다른 곳에 있는 설치 콘솔 인스톨러가 실행이 가능하도록 하려면 jeus60-unix-generic.bin 파일의 실행 권한을 준다(chmod u+x jeus60-unix-generic.bin).
4. 콘솔로부터 jeus60-unix-generic.bin를 실행시킨다($ ~] ./ jeus60-unix-generic.bin).

```
$ ~] ./jeus60-unix-generic.bin
Preparing to install...
```

5. 해당 파일을 실행시키면 나타나는 License 부분은 확인하여 동의 한다.

```
PRESS <ENTER> TO CONTINUE:
DO YOU ACCEPT THE TERMS OF THIS LICENSE AGREEMENT? (Y/N):  Y
```

6. 운영체제의 종류를 선택한다.

```
Choose Platform
---------------
Choose current system ( platform-architecture )
1)HP-UX PA-RISC
2)HP-UX ITANIUM
3)Solaris Ultra-Sparc
4)Solaris x86
5)Solaris x64
6)AIX 5.x PowerPC
7)Linux ITANIUM
8)Linux x86
9)Linux x64
```

```
Quit) Quit Installer


Choose Current System (DEFAULT: 9):
```

7. 설치 디렉토리를 선택한다. 기본값으로 사용하려면 <엔터>를 누르고, 디렉토리 변경을 원하면 설치 경로를 입력한다.

```
Choose Install Folder
---------------------
Where would you like to install?
  Default Install Folder: $HOME/jeus6
ENTER AN ABSOLUTE PATH, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
        :
```

8. JEUS를 설치할 타입을 결정한다.

```
Choose Install Set
------------------
Please choose the Install Set to be installed by this installer.
 ->1- Full Install
   2- Typical
ENTER THE NUMBER FOR THE INSTALL SET, OR PRESS <ENTER> TO ACCEPT
THE DEFAULT
    :
```

9. JDK의 위치를 입력한다.

```
Choose JDK Folder
-----------------
Please Choose a Folder:
Input User JDK Folder (DEFAULT: /usr/jdk1.6.0_15):
```

10. 패스워드를 입력한다.

```
Enter the Password for the administrator account.
This password will be registered in JEUS as the first user.
Input Password::
Confirm Password::
```

11. 패스워드를 입력하면 JEUS 를 설치할 기본 정보에 대해서 요약해서 보여준다. <엔터>를 치면 설치가 완료된다.

```
Pre-Installation Summary
-------------------------
Please Review the Following Before Continuing:
Product Name: JEUS6.0
Install Folder: $HOME/jeus6
Install Set: Full Install
Disk Space Information (for Installation Target):
    Required:   267,088,590 bytes
    Available: 153,161,621,504 bytes
PRESS <ENTER> TO CONTINUE:
```

환경 변수 설정

JEUS는 다음과 같은 시스템 환경 변수들을 요구한다. 이 변수들은 설치 시 적용되며 환경변수 PATH는 환경변수 파일인 .profile, .cshrc 등에 기록하고 나머지 환경변수는 $JEUS_HOME /bin/jeus.properties 파일에 설정된다.

환경 변수	의미	설정값
PATH	시스템 경로	다음을 포함하고 있어야 한다. $JEUS_HOME/bin $JEUS_HOME/webserver/bin $JEUS_HOME/lib/system
JEUS_HOME	JEUS 설치 디렉토리	$HOME/jeus6/
JEUS_BASSPORT	JEUS가 사용할 네트워크 포트 가운데 가장 기본이 되는 포트(기본: 9763)	9763
JEUS_LIBPATH	JEUS 라이브러리 파일	$JEUS_HOME/lib/system
JAVA_HOME	JAVA2 설치 디렉토리	$JAVA_HOME

JEUS 디렉토리

다음의 항목들은 설치된 JEUS의 홈 디렉토리의 하위 디렉토리들이다.

```
$ ~ /jeus6] ls
bin       docs       logs       ThirdPartyLicenses.txt    webserver
```

```
config      lib         readme.txt      UninstallerData     workspace
derby       license     samples         webhome
```

- Bin: JEUS 실행파일이 포함된 디렉토리
- Config: XML 디스크립터 파일, 노드 설정파일, 보안설정 파일이 포함된 디렉토리
- Derby: 예제 실행 및 테스트를 위해 사용하는 데이터베이스
- Docs: JEUS 매뉴얼과 API 파일이 포함된 폴더
- Lib: JEUS에서 사용하는 라이브러리 파일의 폴더. JEUS 클래스 라이브러리 아카이브가 jeus.jar 에 포함되어 있다.
- License: JEUS 라이선스 파일의 폴더
- Logs: 로그 파일의 폴더
- Samples: 예제가 있는 폴더
- Sessiondb: JEUS에서 session server 사용 시 생성되는 폴더. 설치 과정에서 생성되지 않고 JEUS를 1 회 이상 부팅한 후에 생성된다.
- UninstallerData: Unistall 을 위한 폴더
- Webhome: EJB, Servlet, JSP 어플리케이션의 배치 폴더
- Webserver: Servlet Engine 리스너인 JEUS 웹 서버 폴더
- Workspace: JEUS가 사용하는 임시 폴더. Sessiondb 폴더와 마찬가지로 JEUS 1 회 이상 부팅 후에 생성된다.

JEUS 기동 확인

JEUS 설치가 정상적으로 완료되었음을 확인하기 위해서 다음 단계들을 수행한다.

1. 콘솔 프롬프트에 " jeus" 를 입력한다.

```
************************************************************

- JEUS Home : $HOME/jeus6

- JEUS Base Port : 9736

- Java Vendor : Sun

- Added Java Option :

************************************************************

+ /usr/jdk1.6.0_15/bin/java -server -Xmx512m -bootclasspath/p:/home1/swj/jeus6/lib/system/extension.jar
  -classpath /home1/swj/jeus6/lib/system/bootstrap.jar -Dsun.rmi.dgc.client.gcInterval=3600000 -
  Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.library.path=/home1/swj/jeus6/lib/system -
  Djava.endorsed.dirs=/home1/swj/jeus6/lib/endorsed -
  Djava.naming.factory.initial=jeus.jndi.JNSContextFactory -Djava.naming.factory.url.pkgs=jeus.jndi.jns.url -
  Djava.net.preferIPv4Stack=true -Djava.util.logging.config.file=/home1/swj/jeus6/bin/logging.properties -
  Djeus.home=/home1/swj/jeus6 -Djeus.baseport=9736 -Djeus.jvm.version=hotspot -Djeus.tm.checkReg=true -
  Djeus.tool.WebAdmin.locale.language=ko -Djeus.net.reuseAddress=true -
```

```
Djeus.properties.replicate=jeus,sun.rmi,java.util,java.net jeus.server.JeusBootstrapper

[2010.01.29 11:38:03][1][] [client-10] [MGR-0518] initializing virtual DNS table, enable : true, table : {rhel53-x64:21000=example, rhel53-x64:9736=rhel53-x64}

[2010.01.29 11:38:04][0][b216] [rhel53-x64-10] [MGR-0411] virtual host name of this manager : rhel53-x64

….

….

[2010.01.29 11:38:13][2][b216] [rhel53-x64-10] [JMX-0011] create MBean : JEUS:j2eeType=JeusService,jeusType=ContainerManagerService,JMXManager=rhel53-x64,JeusManager=rhel53-x64,name=rhel53-x64

[2010.01.29 11:38:13][2][b216] [rhel53-x64-10] [JMX-0011] create MBean : JEUS:j2eeType=JeusService,jeusType=ContainerMonitorService,JMXManager=rhel53-x64,JeusManager=rhel53-x64,name=rhel53-x64

[2010.01.29 11:38:13][2][b216] [rhel53-x64-10] [MGR-0248] JEUS Manager is READY
```

2. "JEUS Manager is READY" 라는 메시지가 출력이 되면 JEUS Manager가 정상적으로 로딩되어 JEUS가 대기 상태임을 나타낸다.
3. 다른 콘솔창에서 jeusadmin <node name>을 실행한다. 여기서, <node name>은 호스트 머신명이다. 로그인 시의 패스워드는 JEUS 설치할 때 지정을 해준 패스워드로 접속하면 된다.

```
$ ~] jeusadmin rhel53-x64

Login name>administrator

Password>

JEUS 6.0 (Fix#6) JEUS administration tool

rhel53-x64>
```

4. Jeusadmin 툴에 로그인을 한 후에 boot와 down 같은 명령으로 JEUS 서버를 제어할 수 있다.

```
rhel53-x64>boot

rhel53-x64 boot done

rhel53-x64_container1

rhel53-x64>


[2010.01.29 11:49:12][2][b216] [rhel53-x64-12] [MGR-0205] command : boot()

[2010.01.29 11:49:12][2][b216] [rhel53-x64-12] [MGR-0567] trying to start engine container[rhel53-x64_container1]

[2010.01.29 11:49:12][2][b216] [rhel53-x64-12] [MGR-0128] start engine container[rhel53-x64_container1] with command

…

[2010.01.29 11:49:23][2][b216] [container1-10] [MGR-0103] engine container[rhel53-x64_container1] is READY

[2010.01.29 11:49:23][2][b216] [container1-10] [MGR-0101] currently running engines
```

```
of engine container[rhel53-x64_container1] : [rhel53-x64_ws_engine1, rhel53-x64_jms_engine1, rhel53-x64_ejb_engine1, rhel53-x64_servlet_engine1]

[2010.01.29 11:49:23][0][b216] [rhel53-x64-13] [MGR-0303] engine container[rhel53-x64_container1] initialization successfully done [pid : 13360]

[2010.01.29 11:49:23][2][b216] [container1-15] [WEB-3384] unix(hth-0:9900:143) established

[2010.01.29 11:49:23][0][b216] [container1-15] [WEB-3347] worker(webtob1-hth0(localhost:9900)-w00:unix(hth-0:9900:143)) : connect successful
```

"connect successful" 메시지가 출력이 되면 정상적으로 JEUS가 부팅된 것이다.

```
rhel53-x64>down

Do you really want to shutdown the node [rhel53-x64]? (y : n):>y

The JEUS node [rhel53-x64] is down.

rhel53-x64>


[2010.01.29 13:36:27][0][b216] [rhel53-x64-12] [MGR-0207] command : down()

[2010.01.29 13:36:27][2][b216] [rhel53-x64-12] [MGR-0261] shutting down the jeus server ...

[2010.01.29 13:36:27][2][b216] [rhel53-x64-12] [MGR-0566] attempt to shutdown all running container

…

[2010.01.29 13:36:27][2][b216] [container1-17] OnePortServer[0.0.0.0/0.0.0.0:10501] is successfully shutdown.

[2010.01.29 13:36:27][0][b216] [container1-10] [MGR-0099] container rhel53-x64_container1 shutdown

[2010.01.29 13:36:28][0][b216] [rhel53-x64-12] [MGR-0141] engine container[rhel53-x64_container1] successfully stopped

[2010.01.29 13:36:28][0][b216] [rhel53-x64-12] [MGR-0558] all containers in the node shutdown successfully

[2010.01.29 13:36:28][0][b216] [rhel53-x64-12] [MGR-0262] all containers successfully shutdowned
```

"all containers successfully shutdowned" 메시지가 출력이 되면 JEUS가 정상적으로 Down 된 것이다.

5. JEUS 기동 후, 프로세스 확인

```
/home1/swj] ps -ef |grep jeus


swj          4833 24249    0 11:40 pts/9       00:00:00 /bin/sh /home1/swj/jeus6/bin/jeus

swj          4834    4833 20 11:40 pts/9       00:00:30 /usr/jdk1.6.0_15/bin/java -server -Xmx512m -Xbootclasspath/p:/home1/swj/jeus6/lib/system/extension.jar -classpath /home1/swj/jeus6/lib/system/bootstrap.jar -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.library.path=/home1/swj/jeus6/lib/system -
```

Djava.endorsed.dirs=/home1/swj/jeus6/lib/endorsed -
Djava.naming.factory.initial=jeus.jndi.JNSContextFactory -Djava.naming.factory.url.pkgs=jeus.jndi.jns.url -
Djava.net.preferIPv4Stack=true -Djava.util.logging.config.file=/home1/swj/jeus6/bin/logging.properties -
Djeus.home=/home1/swj/jeus6 -Djeus.baseport=9736 -Djeus.jvm.version=hotspot -
Djeus.tm.checkReg=true -Djeus.tool.WebAdmin.locale.language=ko -Djeus.net.reuseAddress=true -
Djeus.properties.replicate=jeus,sun.rmi,java.util,java.net jeus.server.JeusBootstrapper

swj 4991 4928 0 11:40 pts/4 00:00:00 /bin/sh /home1/swj/jeus6/bin/jeusadmin rhel53-x64

swj 4992 4991 1 11:40 pts/4 00:00:01 /usr/jdk1.6.0_15/bin/java -classpath
/home1/swj/jeus6/lib/system/bootstrap.jar -Djeus.tm.not_use=true -Djava.net.preferIPv4Stack=true -
Djava.library.path=/home1/swj/jeus6/lib/system:/home/jeus/jeus5/lib/system -
Djava.library.path=/home1/swj/jeus6/lib/system -Djeus.home=/home1/swj/jeus6 -Djeus.baseport=9736
-Djava.endorsed.dirs=/home1/swj/jeus6/lib/endorsed -
Djava.naming.factory.initial=jeus.jndi.JEUSContextFactory -Djava.naming.factory.url.pkgs=jeus.jndi.jns.url
-Djava.util.logging.config.file=/home1/swj/jeus6/bin/logging.properties jeus.server.Bootstrapper
jeus.server.manager.JeusCommander rhel53-x64

swj 5098 4834 22 11:41 pts/9 00:00:17 /usr/jdk1.6.0_15/jre/bin/java -Xms256m -Xmx512m -
XX:MaxPermSize=128m -server -Xbootclasspath/p:/home1/swj/jeus6/lib/system/extension.jar -classpath
/home1/swj/jeus6/lib/system/bootstrap.jar -
Djava.security.policy=/home1/swj/jeus6/config/security/policy -
Djava.util.logging.manager=jeus.util.logging.JeusLogManager -
Djava.library.path=/home1/swj/jeus6/lib/system -
Djava.endorsed.dirs=/home1/swj/jeus6/lib/endorsed -
Djeus.properties.replicate=jeus,sun.rmi,java.util,java.net -Djeus.net.reuseAddress=true -
Djeus.jvm.version=hotspot -Djava.util.logging.config.file=/home1/swj/jeus6/bin/logging.properties -
Dsun.rmi.dgc.server.gcInterval=3600000 -Djeus.home=/home1/swj/jeus6 -Djava.net.preferIPv4Stack=true
-Djeus.baseport=9736 -Djeus.ejb.operationTimeout=300000 -Djeus.tm.checkReg=true -
Dsun.rmi.dgc.client.gcInterval=3600000 -Djeus.tool.WebAdmin.locale.language=ko -
Djava.naming.factory.initial=jeus.jndi.JNSContextFactory -Djava.naming.factory.url.pkgs=jeus.jndi.jns.url -
Djeus.server.protectmode=false jeus.server.enginecontainer.EngineContainerBootstrapper rhel53-
x64_container1 rhel53-x64_ws_engine1 rhel53-x64_jms_engine1 rhel53-x64_ejb_engine1 rhel53-
x64_servlet_engine1

swj 5156 1 0 11:41 pts/9 00:00:00 wsm -I jeuservice_5155 -b 5155

swj 5157 1 0 11:41 pts/9 00:00:00 htl -I jeuservice_5155 -b 5155

swj 5158 1 0 11:41 pts/9 00:00:00 hth -I jeuservice_5155 -b 5155

swj 5159 1 0 11:41 pts/9 00:00:00 htmls -I jeuservice_5155 -b 5155 -s html

swj 5160 1 0 11:41 pts/9 00:00:00 cgis -I jeuservice_5155 -b 5155 -s cgi

swj 5162 1 0 11:41 pts/9 00:00:00 ssis -I jeuservice_5155 -b 5155 -s ssi

swj 5391 5252 0 11:42 pts/10 00:00:00 grep jeus

설치 시 유의사항

JEUS를 설치할 때의 유의 사항에 대해서 설명한다.

1. 운영체제 선택을 잘못하면 에러 발생

   JEUS 설치 시 운영체제를 선택하는 단계에서 서버의 특성에 맞지 않는 운영체제를 선택할 경우 정상적인 설치가 안 된다.

예) Linux x64 인 장비에서 Linux x86 을 선택한 경우

```
java.lang.UnsatisfiedLinkError: $HOME /jeus6/lib/system/libRunner.so: $HOME
/jeus6/lib/system/libRunner.so: cannot open shared object file: No such file or directory (Possible cause:
architecture word width mismatch)
at java.lang.ClassLoader$NativeLibrary.load(Native Method)
at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1751)
at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1676)
at java.lang.Runtime.loadLibrary0(Runtime.java:822)
at java.lang.System.loadLibrary(System.java:993)
at jeus.util.Runner.<clinit>(Runner.java:31)
at jeus.server.JeusServer.main(JeusServer.java:916)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:585)
at jeus.server.Bootstrapper.callMainMethod(Bootstrapper.java:299)
at jeus.server.Bootstrapper.callMain(Bootstrapper.java:371)
at jeus.server.Bootstrapper.main(Bootstrapper.java:365)
at jeus.server.JeusBootstrapper.main(JeusBootstrapper.java:8)
```

2. GLIBC 버전이 낮은 경우

   서버의 GLIBC 버전이 2.4 보다 하위 버전일 경우에 설치 시 에러가 발생한다. GBLIC의 버전은 다음의 명령어를 통해서 확인할 수 있다.

   OS가 리눅스인 경우에 해당되며, 윈도우 계열이나 유닉스 계열에서는 해당되지 않는다.

```
$~ ] rpm -qa |grep glibc
compat-glibc-2.3.4-2.26
compat-glibc-2.3.4-2.26
glibc-2.5-34
glibc-devel-2.5-34
compat-glibc-headers-2.3.4-2.26
glibc-common-2.5-34
glibc-headers-2.5-34
glibc-2.5-34
glibc-devel-2.5-34
```

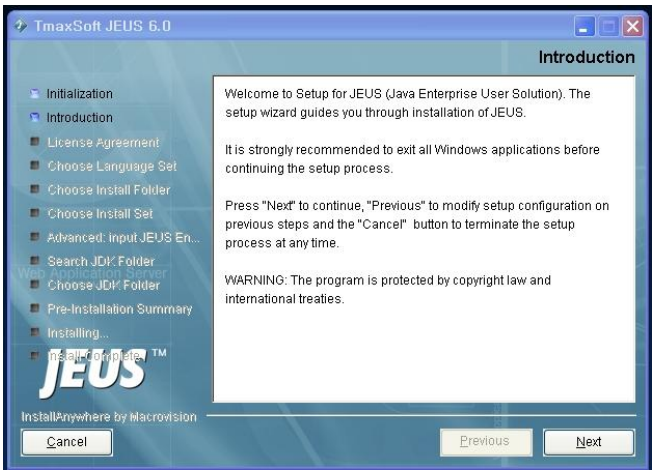
예) GLIBC 버전이 2.3 인 경우

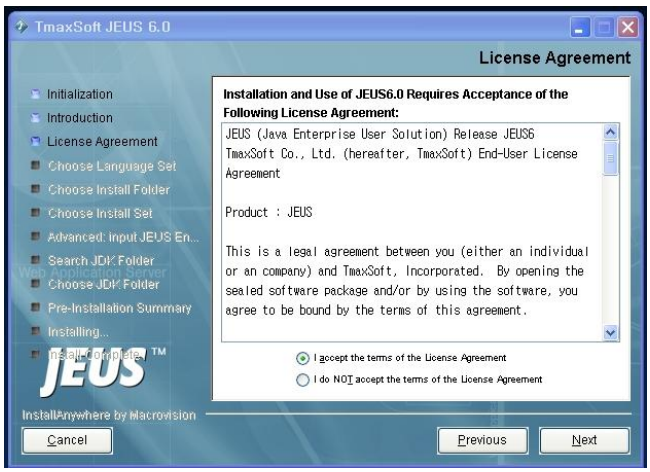
```
java.lang.UnsatisfiedLinkError: $HOME/jeus6/lib/system/libRunner.so: /lib64/tls/libc.so.6: version `GLIBC_2.4' not found (required by $HOME/jeus6/lib/system/libRunner.so)
at java.lang.ClassLoader$NativeLibrary.load(Native Method)
at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1751)
at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1676)
at java.lang.Runtime.loadLibrary0(Runtime.java:822)
at java.lang.System.loadLibrary(System.java:993)
at jeus.util.Runner.<clinit>(Runner.java:31)
at jeus.server.JeusServer.main(JeusServer.java:916)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:585)
at jeus.server.Bootstrapper.callMainMethod(Bootstrapper.java:299)
at jeus.server.Bootstrapper.callMain(Bootstrapper.java:371)
at jeus.server.Bootstrapper.main(Bootstrapper.java:365)
at jeus.server.JeusBootstrapper.main(JeusBootstrapper.java:8)
```

기본설치(GUI 모드)

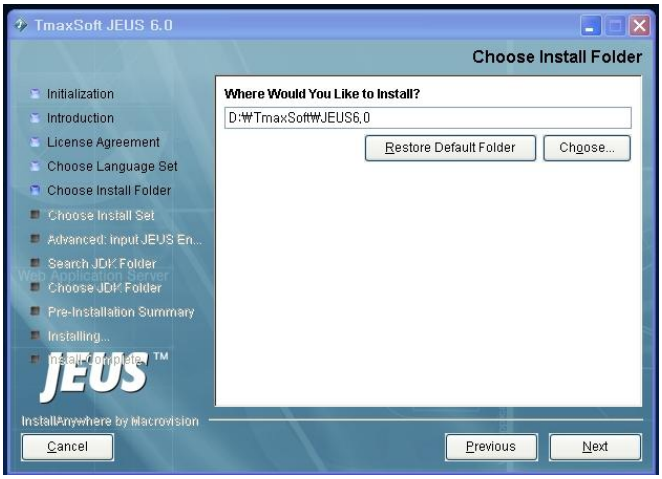
Windows GUI 모드에서 JEUS 를 설치하는 방법에 대해서 살펴보도록 한다.

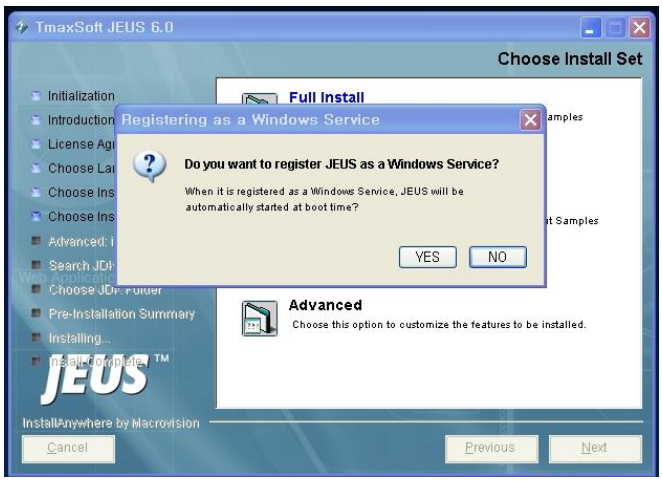
1. 티맥스 소프트에서 Jeus6.0-winx86.exe 실행파일을 다운로드 받아서 실행시킨다.



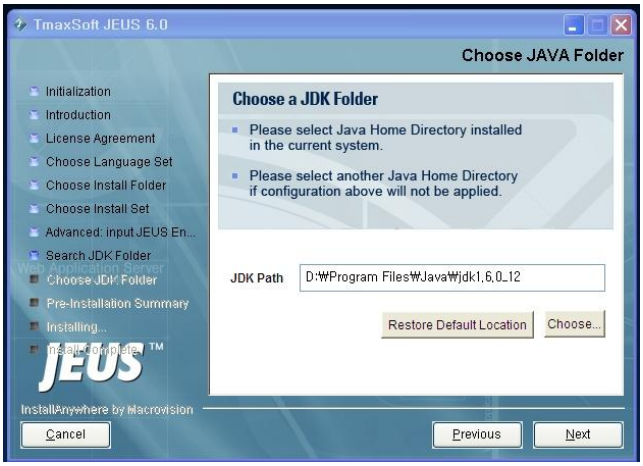


2. 설치할 경로를 지정한 후에 설치할 타입에 대해서 선택하여 설치를 계속한다. JEUS가 윈도우 시작 시에 자동으로 부팅 되도록 윈도우 서비스에 등록하려면 “YES”를 선택한다.

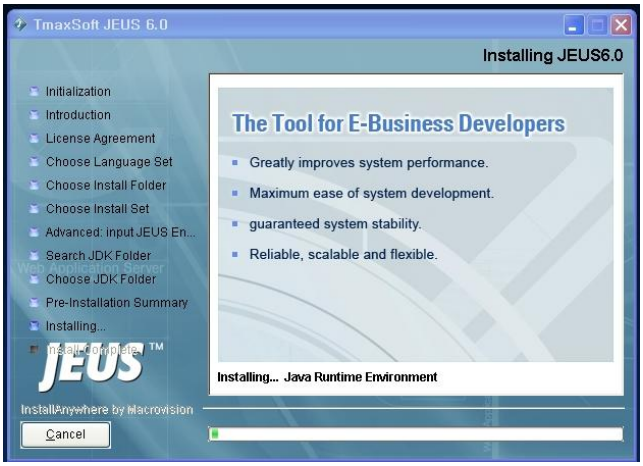


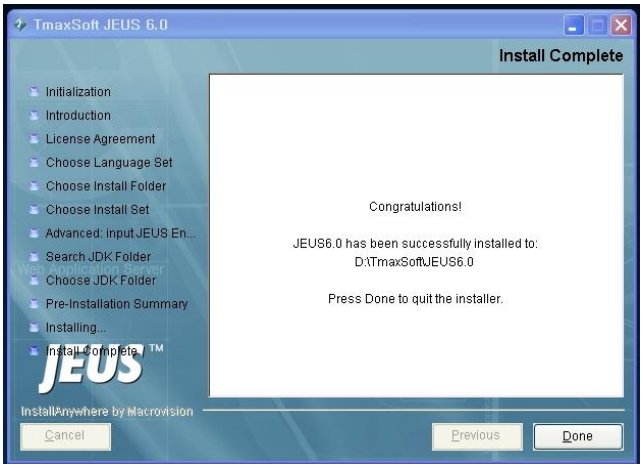


3. JDK가 설치된 디렉토리를 지정해 주고, 패스워드를 지정해준다.



4. 정상적으로 설치가 완료되면 다음과 같은 메시지가 출력된다.





환경변수 설정

환경변수 PTAH는 Windows 명령 프롬프트에 “set” 명령을 사용함으로써 정확하게 설정되었는지 확인 할 수 있다. 또한, 시작 → 설정 → 제어판 → 시스템 → 고급 → 환경변수에서 확인 가능하다.

나머지 환경변수는 %JEUS_HOME%\bin\jeus.properties.cmd 파일에 설정된다.

JEUS 기동 확인

Windows 환경에서의 JEUS 기동 확인은 명령 프롬프트에서 확인 할 수 있다. 확인하는 방법 및 명령어는 UNIX/LINUX 환경에서 소개한 내용과 동일하다.

ALTIBASE와 JEUS의 연동

ALTIBASE와 JEUS를 연동하여 사용하기 위한 설정 방법에 대해서 설명한다.

JDBC 드라이버

JDBC란 자바 응용프로그램에서 데이터베이스에 연결하여 여러 SQL을 실행할 수 있도록 제공하는 표준 인터페이스를 말한다.

이러한 JDBC 드라이버는 각 데이터베이스 벤더사가 제공하고 있으며, ALTIBASE에서는 Altibase.jar 파일로 제공하고 있다. 이 JDBC 드라이버는 $ALTIBASE_HOME/lib 디렉토리 안에 존재한다.

ALTIBASE 5 버전부터는 $ALTIBASE_HOME/lib 디렉토리에 Altibase.jar와 Altibase5.jar 파일이 존재하는데, Altibase.jar는 일반 ALTIBASE JDBC 드라이버 파일이며, Altibase5.jar는 ALTIBASE 5 버전과 그 이하의 버전을 함께 사용하고 싶을 때 사용하는 파일이다. 따라서 하나 이상의 ALTIBASE 와 연동하기를 원한다면 일반적으로 Altibase5.jar 파일을 사용한다.

JDBC 드라이버 설정

ALTIBASE 와 JEUS 를 연동하기 위해서는 ALIBASE에서 제공하는 JDBC 드라이버(Altibase.jar)를 $JEUS_HOME/lib/datasource 디렉토리에 위치시켜야 한다.

커넥션 풀링

커넥션 풀링(Connection Pooling)은 DB 커넥션의 캐쉬를 위한 하나의 프레임워크이다. 커넥션 풀이 시작될 때 특정한 수의 물리적 커넥션을 만들며 이는 어플리케이션 실행 중에 커넥션 생성을 위한 오버헤드를 줄여준다.

커넥션 풀의 이점은 다음과 같다.

1. 보다 높은 성능

   DB 커넥션 생성은 처리 과정이 느리다. 커넥션 풀 안에서의 모든 실제 커넥션들은 미리 만들어져 어플리케이션의 요청 처리를 위한 준비가 되어 있다. 커넥션을 더 이상 사용하지 않을 때에는 그것을 풀에 반환시켜서 커넥션 중단의 오버헤드를 감소시킬 수 있다.

2. 연결 관리

   동시 커넥션들의 수를 제어할 수 있다. 동시 커넥션들의 최대 수를 구성함으로써 DB의 동시 커넥션을 제한하는 작업을 효율적으로 할 수 있다.

데이터 소스

하나의 javax.sql.DataSource는 어플리케이션과 커넥션 풀 사이의 인터페이스이다. javax.sql.DataSource 객체는 DB 커넥션들의 팩토리로 볼 수 있으며 java.sql.DriverManager 보다 많은 이점을 제공한다.

아래는 데이터 소스들의 4 가지 타입들을 간략하게 정리하였다.

1. 기본 데이터 소스

   사용자들을 위해 커넥션을 반환한다. 커넥션 풀링이 이뤄지지 않기 때문에 커넥션 풀 형식에 비해 추가적인 오버헤드가 있을 수 있다.

2. 커넥션 풀 데이터 소스

   커넥션 풀에 저장된 커넥션을 얻어 응용 프로그램 등에 반환한다. JEUS 에서 제공하는 커넥션 풀링 기능을 이용하므로 DB에 매번 접속하여 커넥션을 가져오는 방식보다 오버헤드가 덜하다. Autocommit을 False로 하고 사용할 경우 어플리케이션이 직접 로컬 트랜잭션을 컨트롤 할 수 있다.

3. XA 데이터 소스

   분산/전역 트랜잭션에 이용되는 커넥션을 관리한다. 이 데이터 소스 형식은 2PC(2 Phase Commit)를 이용할 경우에만 사용하도록 한다. EJB나 Servlet에서 트랜잭션을 시작한 후, XA 데이터 소스로부터 얻어진 두 개 이상의 커넥션을 이용하여 작업을 시작하면 자동적으로 2PC-protocol이 시작되며, 이 커넥션은 트랜잭션이 끝난 이후에는 다시 사용할 수 없다.

   XADataSource를 사용할 경우에는 각각의 데이터 소스에 대해 트랜잭션 복구 기능이 지원된다.

   XA 데이터 소스를 사용한 DB 연동에 대해서는 ALTIBASE 5.3.3 API 매뉴얼을 참고하면 된다.

4. 로컬 XA 데이터 소스

   커넥션 풀 데이터 소스에서 얻은 커넥션을 Autocommit을 꺼서 로컬 트랜잭션이 항상 켜진 상태로 사용하고 커밋이나 롤백을 트랜잭션 매니저가 처리해준다. 이 때문에 로컬 트랜잭션을 XA에 참여하도록 에뮬레이션 할 수 있다. 참고로 로컬 XA 데이터 소스는 JDBC 드라이버가 XA 데이터 소스를 지원하지 않더라도 XA에는 참여시킬 필요가 있을 때 사용할 수 있다.

   로컬 XA 데이터 소스는 기능상의 제약으로 제대로 복구가 되지 않을 경우가 생길 수 있으며, 하나의 글로벌 트랜잭션 내에서 최대 하나의 로컬 XA 데이터 소스만 참여할 수 있다.

데이터 소스 구성

Altibase와 JEUS를 연동하기 위해서 JEUSMain.xml에 데이터 소스를 설정할 수 있다. Javax.sql.DataSource의 속성들은 각 드라이버별로 다르기 때문에 사용하기 원하는 드라이버의 특성을 파악하고 그 특성에 맞게 설정해야 한다.

다음의 XML 태그들은 <resource><data-source>…<database> XML 태그의 하위 태그로 사용할 수 있다.

태그	설명

Vender	DB 벤더의 이름 (oracle, mssql, db2, sybase, tibero, others). ALTIBASE는 others로 설정하도록 한다.
Export-name	JNDI에 바인딩 될 이름. 이 이름으로 데이터 소스 객체가 바인드된다. 사용자가 임의로 지정해주면 된다.
Data-source-class-name	JDBC 드라이버 별 데이터 소스 클래스 이름. ABConnectionPoolDataSource, BlackboxConnectionPoolDataSource 등의 설정 방법을 결정한다.
Data-source-type	“DataSource”, “ConnectionPoolDataSource”, “XADataSource”, “LocalXAData- Source” 값 중에 하나. 사용하려는 데이터 소스 타입에 맞게 선택한다.
Data-source-name	데이터 소스의 이름. 드라이버 벤더에 의존적이며 일반적으로 DataSourceClass-Name 값과 동일하다.
Database-name	DB의 이름 (예: mydb)
Service-name	Oracle inet 드라이버 사용시만 사용하며 Oracle Database의 SID 값. ALTIBASE와 연동 시에는 사용 안함.
description	데이터 소스를 설명하는 내용의 텍스트.
Network-protocol	DB에 연결할 때 사용되는 프로토콜
Password	사용자의 암호
User	사용자 이름
Port-number	DB 리스너의 포트번호
Server -name	DB 가 운용중인 서버의 DNS 이름이나 IP주소
Driver-type	Oracle의 경우 드라이버의 타입(ex. thin, oci)
Property	JDBC 커스텀 프로퍼티 지정 BlackboxConnectionPoolDataSource로 설정할 경우에 이 항목에 DriverClassName과 URL, USER, PASSWORD 를 지정해준다.
Connection-pool	커넥션 풀링에 특화된 내용을 설정해준다.
Auto-commit	커넥션에 지정될 auto commit 값을 지정한다. true, false로 지정한다. 로컬 XA 데이터 소스나 XA 데이터 소스의 경우에는 커넥션이 트랜잭션에 연동되어 있지 않을 경우에만 적용한다.
Action-on-connection-leak	컴포넌트(주로 Stateless 컴포넌트 - Servlet/JSP, Stateless 세션빈, MDB)에서 사용한 JDBC 커넥션에 대한 로깅이나 반환 액션을 설정한다. 설정하지 않았을 경우 기본 동작은 엔진 컨테이너에 설정한 invocation-manager-action을 따른다.

데이터 소스 설정

ALTIBASE와 JEUS를 연동하여 사용하기 위해 데이터 소스를 설정하는 방법에는 콘솔모드에서 직접 XML 파일을 수정하는 방법과 WebAdmin 을 이용한 GUI 모드에서 설정하는 두 가지의 방법을 제공한다.

데이터 소스 설정 시에 커넥션 풀을 구성하여 관리하는 방법에는 ABConnectionPoolDataSource 방법과 BlackboxConnectionPoolDataSource 방법 두 가지가 있다. ABConnectionPoolDataSource는 JEUS가 아니라 ALTIBASE의 JDBC 드라이버에서 커넥션 풀을 관리하는 방식이고, BlackboxConnectionPoolDataSource는 JEUS에서 커넥션 풀을 관리하는 방식이다.

데이터 소스를 설정하는 두 가지 방법을 이용하여 ABConnectionPoolDataSource 방법과 BlackboxConnectionPoolDataSource 방법으로 커넥션 풀을 구성해보도록 한다.

1. BlackboxConnectionPoolDataSource 방법으로 설정

   - JEUSMain.xml 파일을 직접 수정하여 설정

     파일 편집기를 이용하여 JEUSMain.xml 파일을 열고, <resource> ~ </resource>의 내용을 추가하여 준다.

```
<jeus-system>
…
 <resource>
    <data-source>
        <database>
            <vendor>others</vendor>
            <export-name>datasource1</export-name>
            <data-source-class-
name>jeus.jdbc.driver.blackbox.BlackboxConnectionPoolDataSource</data-source-class-
name>
            <data-source-type>ConnectionPoolDataSource</data-source-type>
            <property>
                <name>DriverClassName</name>
                <type>java.lang.String</type>
                <value>Altibase.jdbc.driver.AltibaseDriver</value>
            </property>
            <property>
                <name>URL</name>
                <type>java.lang.String</type>
                <value>jdbc:Altibase://127.0.0.1:20300/mydb</value>
              </property>
              <property>
```

```
                    <name>User</name>
                    <type>java.lang.String</type>
                    <value>sys</value>
                </property>
                <property>
                    <name>Password</name>
                    <type>java.lang.String</type>
                    <value>manager</value>
                </property>
            <connection-pool>
                <pooling>
                    <min>2</min>
                    <max>30</max>
                    <step>1</step>
                    <period>3600000</period>
                </pooling>
            </connection-pool>
        </database>
    </data-source>
 </resource>
...
</jeus-system>
```

- WebAdmin을 이용하여 설정

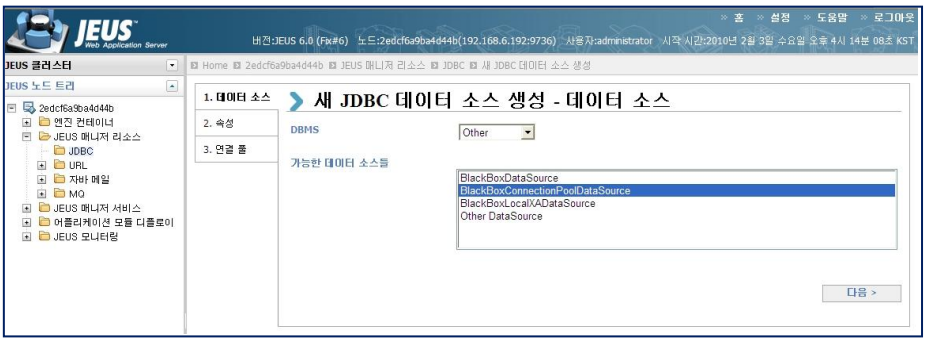
WebAdmin을 이용하여 BlackboxConnectionPoolDataSource 방법으로 커넥션 풀을 구성하는 방법을 알아보도록 한다.

WebAdmin은 http://localhost:9744/WebAdmin 로 접속할 수 있으며, 기본적으로 USER는 Administrator이고, 패스워드는 설치 시에 설정한 값이 된다. 여기서 9744 라는 값은 JEUS_BASSPORT + 8 을 한 값이다.

WebAdmin의 JEUS 매니저 메뉴에서는 원하는 노드를 체크하여 부트, 다운, 종료시키는 기능을 제공하고 있다.

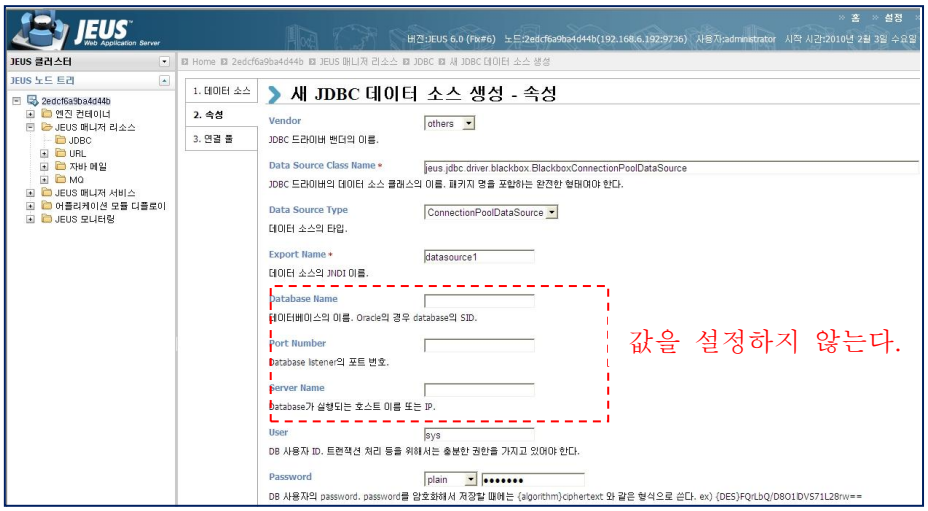
WebAdmin으로 해당 노드를 부팅 시킨 후에, ALTIBASE와의 연동을 위해 JEUS 매니저 리소스 → JDBC 메뉴를 클릭하여 새로운 데이터 소스를 생성한다.

DBMS 벤더를 “Others”로 선택하고, 데이터 소스를 “BlackboxConnectionPoolDataSource”로 선택한 후에 다음 단계로 넘어간다.



새 JDBC 데이터 소스 생성을 할 때, 각각의 속성 값을 설정해주면 된다. 기본적으로 DBMS 벤더 명과 데이터 소스를 선택하면 다음과 같이 “Vender”, “Data Source Class Name”, “Data Source Type”, “Export Name” 이 설정되어 있다. 해당 속성 값의 변경이 필요한 경우만 수정하여 사용하고 그 외에는 그냥 사용하면 된다.

“Database Name”, “Port Name”, “Server Name” 속성 값은 설정하지 말고, 밑에 나오는 “Property” 항목에 설정하고 다음 단계로 넘어간다. (ABConnectionPoolDataSource나 XADataSource로 선택하여 구성할 시에는 “Property”항목이 아닌 “Database Name”, “Port Name”, “Server Name”에 값을 설정해주면 된다.)



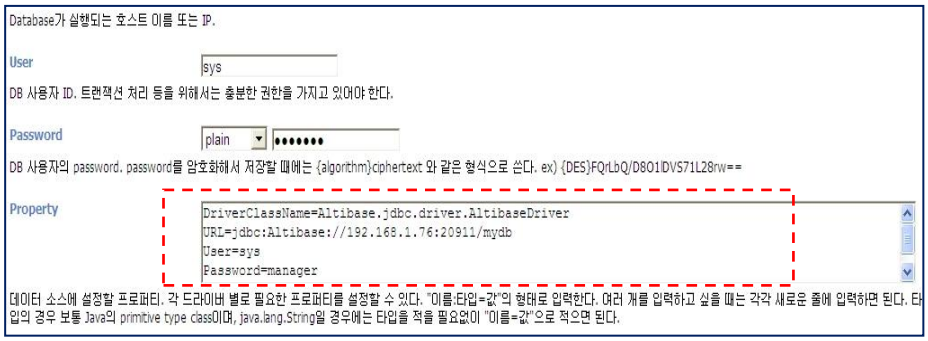
"Property" 속성 값을 다음과 같이 설정해주면 된다.

DriverClassName=Altibase.jdbc.driver.AltibaseDriver

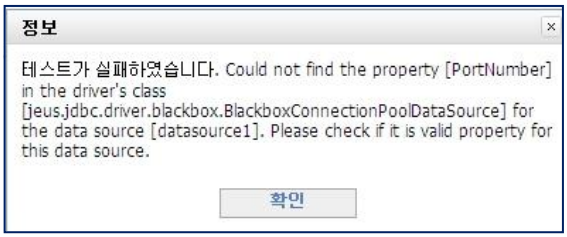
URL= jdbc:Altibase://server_ip:server_port/dbname

USER=sys

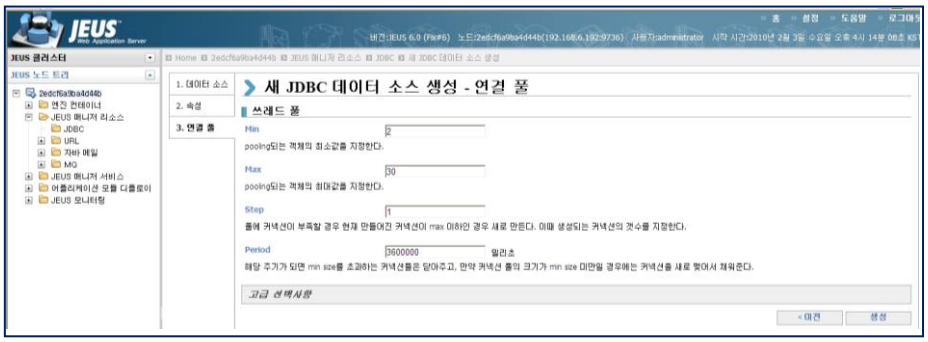
PASSWORD=manager



"Property" 항목에 아래의 값을 설정해주지 않거나, "Database Name", "Port Name", "Server Name"의 값들 중 하나의 값이라도 설정이 되면 에러가 발생한다.



커넥션 풀의 Min/Max 값을 설정해주면 기본적인 설정이 끝나게 된다.



생성이 끝이 난 뒤 테스트 및 바인드를 할 수 있으며 바인드를 해 줘야 해당 설정이 반영이 된다.

2. ABConnectionPoolDataSource 방법으로 설정

- JEUSMain.xml 파일을 직접 수정하여 설정

  파일 편집기를 이용하여 JEUSMain.xml 파일을 열고, <resource> ~ </resource>의 내용을 추가하여 준다.

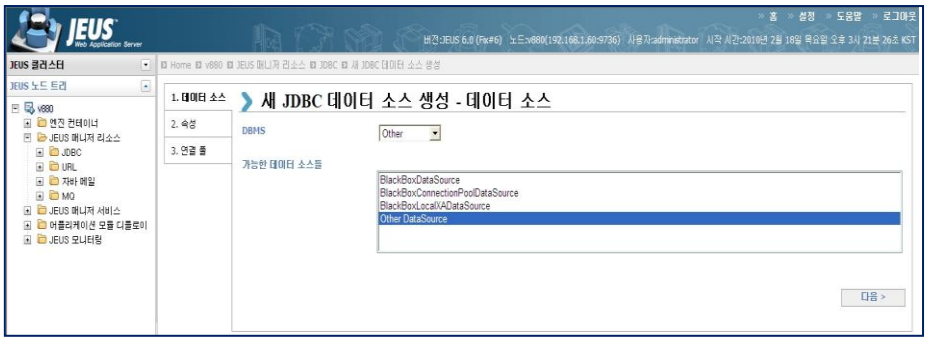
```
<jeus-system>
…
 <resource>
  <data-source>
   <database>
        <vendor>others</vendor>
        <export-name>Datasource1</export-name>
        <data-source-class-ame>Altibase.jdbc.driver.ABConnectionPoolDataSource</data-source-class-name>
        <data-source-type>ConnectionPoolDataSource</data-source-type>
        <database-name>mydb</database-name>
        <port-number>20300</port-number>
        <server-name>127.0.0.1</server-name>
        <user>sys</user>
        <password>manager</password>
```

```
        <auto-commit>true</auto-commit>
        <property>
          <name>Encoding</name>
          <type>java.lang.String</type>
          <value>KSC5601</value>
        </property>
        <property>
          <name>maxPoolSize</name>
          <type>java.lang.Integer</type>
          <value>30</value>
        </property>
        <connection-pool>
          <pooling>
            <min>2</min>
            <max>30</max>
            <step>1</step>
            <period>3600000</period>
          </pooling>
        </connection-pool>
      </database>
    </data-source>
  </resource>
…
</jeus-system>
```

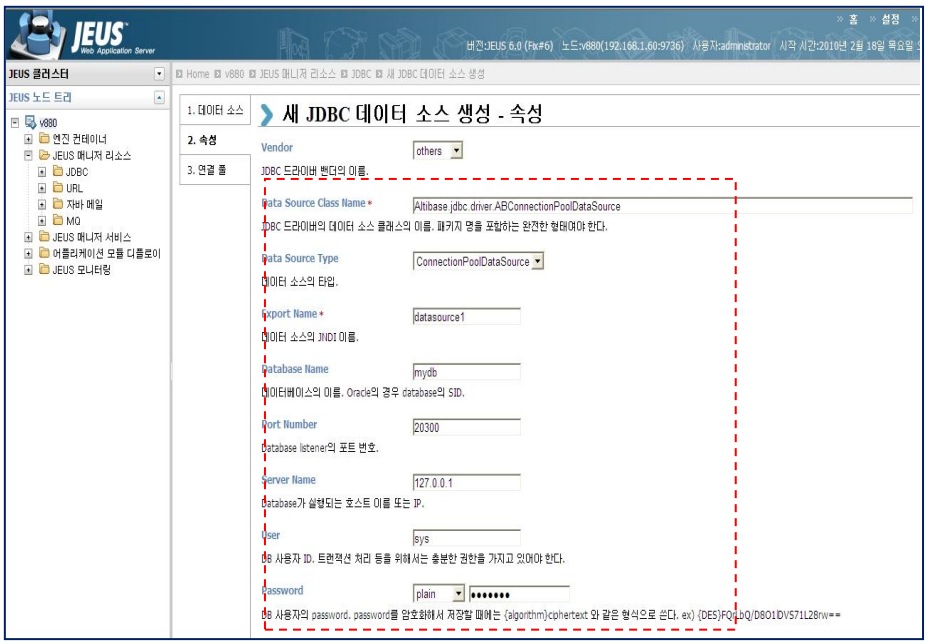
- WebAdmin을 사용하여 설정

  JEUS 에서 제공하는 WebAdmin을 통해서 JEUS의 구동, 종료 및 데이터 소스 설정이 가능하다. WebAdmin을 통해서 설정한 속성 값들은 JEUSMain.xml에 저장이 된다. 속성 값의 설정 이전과 이후의 JEUSMain.xml 파일을 비교해보면 <resource> ~ </resource> 사이에 ALTIBASE의 속성 값이 수정되어 있는 것을 확인할 수 있다.

새로운 JDBC데이터 소스를 생성하기 위해서 “새 JDBC 데이터 소스 생성”을 클릭한다.



DBMS를 “Other”로 선택한 후에 “Other DataSource”를 선택한다.



“Data Source Class Name” ~ “Password” 항목들을 설정해준다. “Data Source Class Name”를 설정할 때는 ABConnectionPoolDataSource로 입력하면 안되고, JDBC 드라이버의 데이터 소스 클래스의 이름. 패키지 명을 포함하는 완전한 형태로 입력해야 한다(Altibase.jdbc.dirver.ABConnectionPoolDataSource)

연결풀 설정 및 바인드, 테스트는 BlackboxConnectionPoolDataSource의 설정을 참고하면 된다.

데이터 소스 설정 시 유의 사항

1. ABConnectionPoolDataSource 사용 시 유의 사항
   - Data-Source-Class-Name으로 ABConnectionPoolDataSource를 설정한 경우에는 “Encoding”, “maxPoolSize” 등의 프로퍼티를 지정하여 사용할 수 있지만,

BlackboxConnectionPoolDataSource 사용 시에는 해당 프로퍼티를 사용할 수 없다.

2. Deadlock 발생

- ABConnectionPool 사용시 initialPoolSize를 사용하면 lock이 걸리는 문제가 발생하기도 한다. 이 문제는 JEUS에서 사용해야 할 Pool 의 값들을 ALTIBASE JDBC 드라이버에서 사용하면서 발생하는 문제로 해결하기 위해서는 initialPoolSize 값을 설정하지 않고 MaxPoolSize 값만 설정해서 사용하면 된다.

```
<property>
  <name>encoding</name>
  <type>java.lang.String</type>
  <value>KSC5601</value>
</property>
<property>
     <name>maxPoolSize</name>
     <type>java.lang.Integer</type>
     <value>10</value>
 </property>
```

체크 쿼리

어플리케이션이 JDBC 커넥션 요청을 했을 때, 특정 SELECT 쿼리를 보내서 커넥션의 상태를 점검하는 기능이다. JDBC 커넥션의 내부적인 에러로 인한 끊김, 방화벽에 의한 소켓 끊김 현상 등을 체크할 때 유용하다. 점검이 실패하면 물리적 커넥션을 새로 만들어서 그에 대한 핸들을 어플리케이션으로 리턴해 준다.

체크 쿼리 기능은 크게 두 가지로 설정할 수 있다. 첫째로 단순히 설정상의 <check-query> 태그에 쿼리문을 넣는 방법이 있고, <check-query-class> 태그를 이용하여 기능을 확장할 수도 있다.

JEUSMain.xml 파일의 <database> ~ </database> 사이에 직접 추가해도 되며, WebAdmin을 사용하는 경우에는 Datasource를 생성하거나 수정 시에 [연결풀] → [고급 선택사항]을 클릭한 후에 해당 값을 설정하면 된다.

1. Check-query 설정

   체크 쿼리를 위한 쿼리문은 DB에 업데이트를 가하는 명령이 아닌 단순히 쿼리만을 위한 명령어를 넣어야 한다.

```
<check-query>select 1 from dual</check-query>
```

2. Check-query-timeout 설정(msec)

check-query 를 수행했을 때 DB 서버의 응답이 없는 경우 무한정 대기하는 상황이 발생할 수도 있다. 이런 경우를 피하기 위해서 check-query-timeout 을 설정할 수 있는데, 설정값은 msec 이며, 1000msec 보다 적을 경우에는 0 으로 설정된다.

```
< check-query-timeout>20000</check-query-timeout>
```

3. Non-validation-interval 설정(msec)

   check-query의 잦은 사용으로 오버헤드가 발생하게 된다면 Non-validation-interval을 설정하여 해결할 수 있다.

   check-query를 수행할 때의 시각과 가장 마지막에 커넥션을 사용한 시각의 차이가 Non-validation-interval의 설정값 사이에 있다면 check-query를 수행하지 않도록 한다.

```
<non-validation-interval>10000</non-validation-interval>
```

4. check-query 에 대한 destroy 정책 설정

   사용자는 check-query가 실패했을 경우 해당 커넥션 풀에 있는 나머지 커넥션들에 대한 destroy 정책을 다음과 같이 결정할 수 있다.

   - FailedConnectionOnly: check-query가 실패한 커넥션만 버린다. 기본 설정값.
   - AllConnections: 나머지 커넥션들도 모두 버린다.

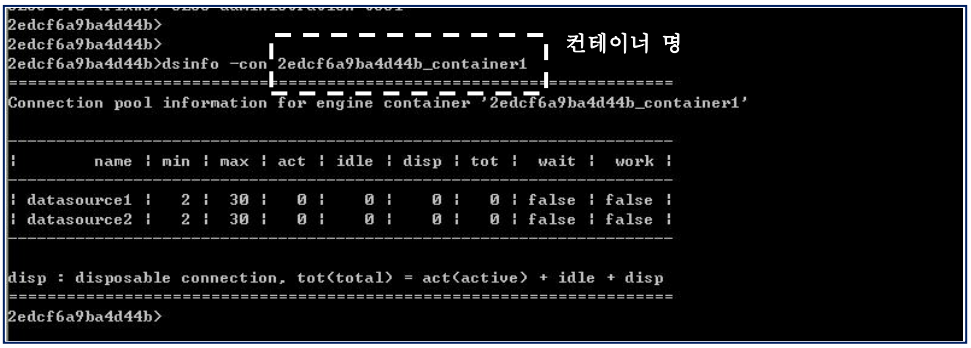
```
<destroy-policy-on-check-query>AllConnections</destroy-policy-on-check-query>
```

커넥션 풀 모니터링

커넥션 풀을 모니터링 하는 방법은 jeusadmin을 이용하는 방법과 WebAdmin을 이용하는 방법이 있다. 이번 절에서는 jeusadmin을 이용하여 커넥션 풀을 모니터링 하는 방법에 대해서 설명한다.

jeusadmin에서는 엔진 컨테이너에 구성된 JDBC 커넥션 풀을 모니터링 하기 위해서 'dsinfo'라는 명령어를 사용한다. 컨테이너에 구성된 모든 커넥션 풀을 모니터링 하기 위해서는 컨테이너 명을 명시해주고, 하나의 커넥션 풀을 모니터링 하기 위해서는 컨테이너에 생성한 데이터 소스 명(Export Name)을 명시해주면 된다.

```
Jeusadmin> Dsinfo –con 컨테이너 명 데이터 소스 명(export name)
```



항목	설명
Name	DB풀의 export name
Min	풀안에서 유지되는 커넥션의 최소 크기
Max	풀안에서 유지되는 커넥션의 최대 크기
Act	어플리케이션이 사용하고 있는 커넥션의 수
Idle	현재 풀에 들어있는 커넥션의 수
Disp	한번 사용하고 버리는 커넥션의 총 개수
Tot	DB 커넥션의 총 수(active+idle+disposable connection)
Wait	풀에 커넥션이 비었을 경우, 쓰레드를 기다리게 할 것인지를 결정한다. "true" 일 경우 기다리게 하고, "false"일 경우 풀과 상관없는 커넥션을 만들어준다.
work	만약 DB 풀이 활성화 상태이면 "true"이고 비활성화이거나 아직 생성되지 않은 상태이면 "false"이다.

또한, 'dsconinfo' 명령어를 사용해서 각각의 데이터 소스 별로 현재 커넥션의 상태를 파악하거나 관련 통계를 볼 수 있다.

```
Jeusadmin> dsconinfo –con 컨테이너 명 데이터 소스 명(export name)
```

```
2edcf6a9ba4d44b>dsconinfo -con 2edcf6a9ba4d44b_container1 datasource1


================================================================

Connection information list for the engine container[2edcf6a9ba4d44b_container1]

----------------------------------------------------------------

| id | state | state-time(ms) | use-count | type |

----------------------------------------------------------------

| datasource1-1 | idle | 10000 | 0 | pooled |

----------------------------------------------------------------

================================================================
```

항목	설명
Id	해당 컨테이너의 데이터 소스 내에서 각 connection 별로 붙인 고유한 값
State	커넥션의 상태를 나타내며 active와 idle로 나뉜다. Active일 경우 현재 사용중인 커넥션을 의미한다.
Usecount	Open, close 짝이 몇 번 일어났는가를 의미한다.
State time	커넥션이 현재 상태로 바뀐 후 지속된 시간을 의미한다.

type	풀링된 커넥션인지 disposable 커넥션인지 구분한다.

데이터 소스의 Export Name을 알고 있으면 jeusadmin을 통해서 사용 가능한 풀인지 설정 테스트를 해볼 수 있다.

```
Jeusadmin> testdsconfig 데이터 소스 명(export name)
```

```
2edcf6a9ba4d44b>testdsconfig datasource1
Configuration is valid. You can use it.
```

샘플 예제

위에서 ALTIBASE와 JEUS의 연동 방법에 대해서 알아봤다면, 본 절에서는 위 Node 설정을 이용하여 Altibase에 접속 및 쿼리를 수행함으로써 ALTIBASE와 JEUS의 연동을 확인해본다.

Pool 사용 샘플 예제

```
<%@ page import="javax.naming.*" %>
<%@ page import="java.sql.*" %>
<%@ page import="javax.sql.*" %>
<%
    Connection con=null;
    Statement   st=null;
    ResultSet   rs=null;
    try
      {
          InitialContext ctx = new InitialContext();
          DataSource ds = (DataSource)ctx.lookup("DataSource1");
          con=ds.getConnection();
          st=con.createStatement();
          rs=st.executeQuery("select *   from dual");
          while(rs.next())
            {
                out.println("TABLE_NAME :" + rs.getString(1)+"<br>");
            }
      }
    catch(Exception e)
      {
          out.println("Error:" + e.getMessage());
          e.printStackTrace();
      }
    finally
      {
          if(rs!=null)rs.close();
          if(st!=null)st.close();
          if(con!=null)con.close();
```

```
        }
%>
```

getConnection 사용 샘플 예제

```
<%@ page import="java.util.*"%>
<%@ page import="java.sql.*"%>
<%
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    String db_url = "jdbc:Altibase://127.0.0.1:20300/mydb";
    String db_user = "sys";
    String db_passwd = "manager";
    String enc = "KO16KSC5601";
    Properties props = new Properties();
    props.put("user", db_user);
    props.put("password", db_passwd);
    props.put("encoding", enc);

    try {
        try {
            Class.forName("Altibase.jdbc.driver.AltibaseDriver");
            conn = DriverManager.getConnection(db_url, props);
            out.print(" Connection ok" + "<br>" );
        } catch (Exception e) {
            out.println("### CONN ERROR=>" + e.toString() + "###" + "<br>");
        }
        String Query = "select * from dual";
        pstmt = conn.prepareStatement(Query);
        String get_1 = null;
```

```
            try {
                rs = pstmt.executeQuery();
                while(rs.next()) {
                        get_1 = rs.getString(1);
                        out.println(" get_1 = " + get_1 + "<br>");
                  }
            } catch (Exception e) {
                 out.println("### SELECT ERROR=>" + e.toString() + "###" + "<br>");
            }
        } catch (Exception e) {
             out.println("### ERROR=>" + e.toString() + "###" + "<br>");
             try {
                   conn.rollback();
             } catch(Exception ex) {
                 out.println("### rollback ERROR=>" +    ex.toString()    + "###" + "<br>");
             }    // end of try
        } finally {
            try {
                 conn.close();
            } catch(Exception ex) {
                 out.println("### close ERROR=>" +    ex.toString()    + "###" + "<br>");
            }    // end of try
       }    // end of try
%>
```

실행 방법

WebAdmin을 사용하여 JSP 파일을 디플로이한 후에 테스트하는 방법에 대해서 설명한다.

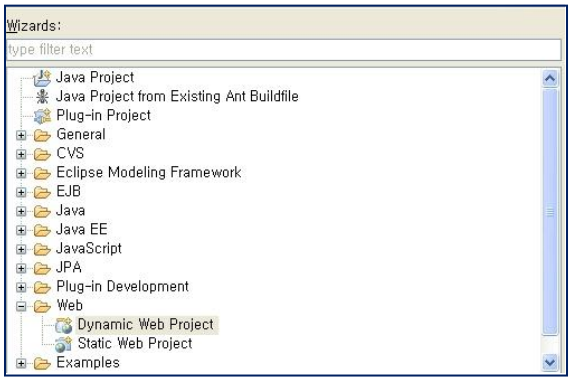
디플로이는 어플리케이션의 서비스들을 시작하기 위해서 JEUS에 모듈 파일을 올리고 제어하는 모든 동작을 일컫는 작업을 의미한다.

JEUS에는 EJB 모듈(.jar 파일), 웹 어플리케이션 모듈(.war 파일), 리소스 어댑터 모듈(.rar 파일) 등을 업로드하여 디플로이 할 수 있으며, 하나의 모듈로 구성된 Standalone 모듈도 Java EE 어플리케이션의 한 종류로 디플로이 할 수 있다.

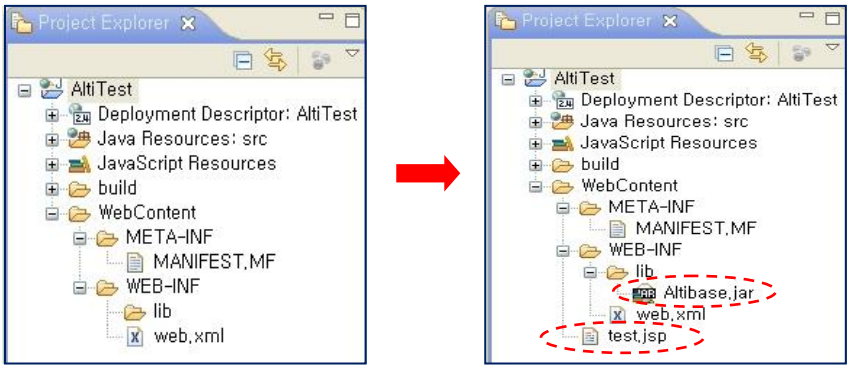
샘플 테스트에서는 위에서 제공한 샘플 예제를 이용하여 WAR 파일을 작성하고, JEUS 서버에 디플로이하여 실행시키는 방법에 대하여 설명한다. 다른 모듈들을 업로드하여 디플로이 하는 방법에 대해서는 JEUS 매뉴얼 중 “Deployment” 부분을 참조하면 된다.

샘플 테스트에서 사용하는 WAR 파일은 이클립스를 사용하여 생성하였다.

1. 이클립스를 실행시켜서 [File] → [New] → [Project] 를 선택한 후에 다음의 Wizard 에서 [Web] → "Dynamic Web Project" 를 선택하여 프로젝트 이름을 설정한다.



2. 다음과 같이 프로젝트가 생성이 되면, 위의 샘플 예제를 JSP 파일로 생성하여 "WebContent" 디렉토리 밑에 추가해준다.
Lib 폴더에는 Altibase.jar 파일을 추가해준다.



3. JSP 파일과 Altibase.jar 파일을 추가했으면, [File] → [Export] 를 선택하고, [Web] → "War file" 를 선택하여 생성할 경로를 지정해주면 WAR 파일이 생성된다.

4. WAR 파일이 생성되면, WebAdmin 노드 트리에서 "어플리케이션 모듈 디플로이" → 파일 업로드 탭을 선택하여 해당 WAR 파일을 업로드 한다.

5. 파일이 업로드 된 후에 업로드 한 모듈을 디플로이 한다. 디플로이가 된 모듈의 이름(WAR 파일의 이름)이 예제 파일을 실행시키는 경로가 된다.

6. 디플로이가 완료되면 인터넷 창을 띄워서 다음과 같은 URL을 입력하여 실행시키면 된다.

http://server_ip:8088/모듈이름/jsp 파일 이름 예) http://192.168.1.76:8088/AltiTest/test.jsp

알티베이스㈜
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 1008 호
02-2082-1000
http://www.altibase.com

대전사무소
대전광역시 서구 둔산동 921
주은리더스텔 901 호
042-489-0330

기술지원본부
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 908 호
02-2082-1000

기술지원센터
02-2082-1114
http://support.altibase.com