설치 및 시작하기

JEUS 9

TMAXSOFT

저작권 공지

제한된 권리

상표 공지

오픈 소스 소프트웨어 공지

본 제품의 일부 파일 또는 모듈은 다음의 라이선스를 준수합니다.: APACHE2.0, CDDL1.0, EDL1.0, OPEN SYMPHONY SOFTWARE1.1, TRILEAD-SSH2, Bouncy Castle, BSD, MIT, SIL OPEN FONT1.1

관련 상세 정보는 제품의 다음 디렉터리에 기재된 사항을 참고하시기 바랍니다. : ${INSTALL_PATH}/license/oss_licenses

기술 지원

홈페이지: https://www.tmaxsoft.com

기술 서비스 센터: 1544-8629

안내서 이력

제품 버전	안내서 버전	발행일	비고
JEUS 9	3.1.1	2024-09-27	-

목차

Part Ⅰ. JEUS 설치하기

1. 설치 개요 및 준비사항

본 장에서는 설치 과정에 대한 기본적인 설명과 설치 전에 준비할 시스템 환경 및 필요한 JDK 환경 구성에 대해서 설명한다.

1.1. 개요

UNIX 환경에서 JEUS를 설치하기 위해 UNIX/Linux 환경에서는 콘솔 모드를 통해 설치가 가능하다.

설치 파일은 다음의 작업을 진행한다.

- JEUS 라이선스 계약을 한다. 설치를 진행하려면 반드시 동의하여야 하므로 주의 깊게 읽어 보길 바란다.
- JEUS 구성 파일 및 디렉터리를 배치한다.
- JEUS의 환경변수를 설정한다.

1.2. 시스템 요구사항

다음은 JEUS를 설치하기 위한 시스템(H/W, S/W) 요구사항이다.

- 시스템 요구사항

  JEUS 설치를 위해서 필요한 H/W, S/W는 다음과 같다.

플랫폼	설치 시 필요 환경
AIX, Linux	JDK 8, JDK 11 또는 JDK 17 2GB 이상의 하드디스크 여유 공간

- 플랫폼 지원 환경

  플랫폼별 JEUS 동작에 필요한 표준 하드웨어 지원 환경은 다음과 같다.

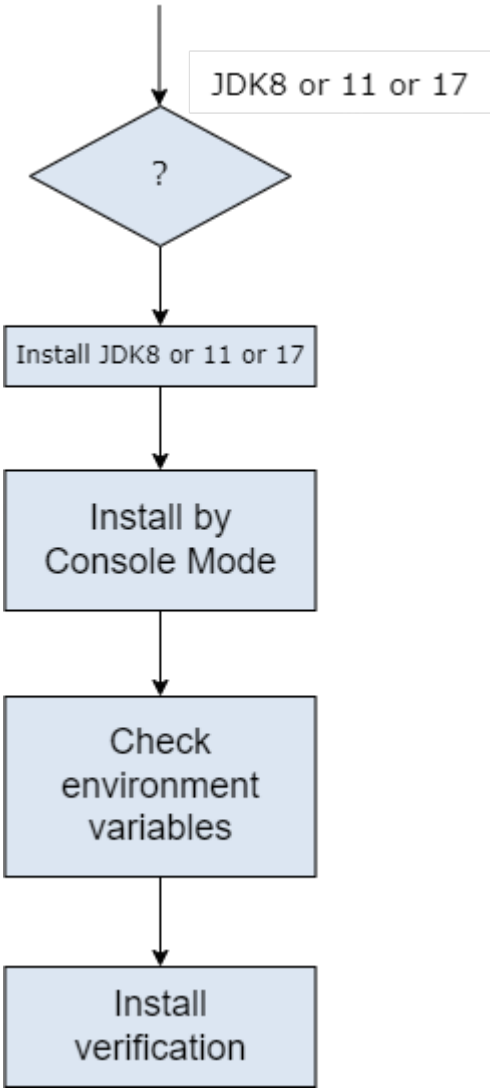
OS Version	CPU	RAM Memory	HardDisc Space	JDK Version
IBM AIX 5L, 6L, AIX 7L	RS6000 IBM pSeries(PowerPC)	1GB	20GB	JDK 8 이상
Linux 계열 (Kernel 2.6 이상)	Intel x86 series k2.6 이상(k2.4 지원) Intel Itanium Series k2.6 이상 IBM pSeries(PowerPC) k2.6 이상	1GB	20GB	JDK 8 이상

1.3. 설치 순서

다음은 UNIX에서 JEUS 설치 순서이다. 전체적인 설치 순서는 동일하고 일부 과정이 차이가 있으므로 자세한 내용은 각 절에서 확인한다.

1. JDK 8, JDK 11 또는 JDK 17을 설치한다.
2. 콘솔 모드에서 JEUS를 설치(파일 복사)한다.
3. 환경변수를 설정하고 검증한다.
4. 설치된 내용을 검증(동작 확인)한다.

다음은 JEUS 설치할 때 작업과 선택사항을 보여준다.



JEUS 설치 순서

위 그림의 작업 상자는 작업을 나타낸다. 자세한 설명은 각 절의 설명을 참고한다.

- UNIX에서 콘솔 모드를 이용한 JEUS 설치 및 제거는 설치와 제거를 각각 참고한다.

1.4. JDK 설치

JEUS를 사용하기 위해서는 JDK(Java SE Developement Kit)가 설치되어 있어야 한다.

JEUS 설치 전에는 JDK 8, JDK 11 또는 JDK 17이 설치되었는지 확인한다. 설치 후에는 환경변수 PATH에 설치된 JDK의 bin 디렉터리를 추가한다.

1.5. JEUS 라이선스 정책

JEUS를 사용하기 위해서는 TmaxSoft에서 발급하는 라이선스-키(License-Key) 파일이 있어야 한다. Installer로 설치한 JEUS에는 기본적으로 트라이얼 라이선스(Trial License)가 포함되어 있다. 라이선스 파일은 JEUS_HOME\license에 'license'라는 파일로 존재한다.

라이선스의 에디션에는 Standard, Enterprise 라이선스가 있으며 기능 및 사용 기한에 차이가 있다.

라이선스를 업그레이드하거나 정식 라이선스를 취득하기 위해서는 TmaxSoft 영업대표(Sales Representative)를 통하거나 직접 TmaxSoft에 문의해야 한다. 라이선스 파일을 사용하기 위해서는 JEUS_HOME\license 디렉터리 아래에 라이선스-키(License-Key) 파일을 license라는 이름으로 복사한다.

라이선스 파일명이 'license'가 아닌 경우 'license'로 변경해야 한다.

2. UNIX 환경에서 설치 및 제거

본 장에서는 JEUS를 설치하고 제거하는 과정에 대해서 설명한다.

2.1. 개요

UNIX/LINUX 환경에서 JEUS 설치/제거는 콘솔 모드에서 가능하다.

JEUS 설치는 다음의 과정으로 진행된다.

1. JDK 8, JDK 11 또는 JDK 17 설치
2. JEUS 설치(파일 복사)
3. 설치 확인
    - 기동 확인
    - 환경변수 설정과 검증

1. JEUS 패치가 있을 경우 JEUS_HOME/lib에 jext, jlext, jnext 디렉터리에 적용 및 확인이 가능하다.

2.2. 설치

본 절에서는 콘솔 모드에서 설치하는 과정에 대해서 설명한다.

콘솔 모드(명령 라인) UNIX/Linux 환경에서 JEUS를 설치하는 방법에 대해서 설명한다. 콘솔 모드에서 설치하기 위해서 Console Installer를 실행해야 한다.

다음은 텍스트 기반 셸을 실행하는 과정에 대한 설명이다.

1. JEUS 설치 CD를 넣고 마운트(mount)한다. CD의 jeus9000_unix_generic_ko.bin 파일이 위치한 디렉터리로 이동한다.
2. 다른 곳에 있는 설치 Console Installer가 실행이 가능하도록 하려면 설치 파일의 실행 권한을 다음과 같이 부여한다.

```
[was@localhost jeus]$ chmod u+x jeus9000_unix_generic_ko.bin
```

3. 콘솔에서 다음의 명령을 입력한 후 <ENTER> 키를 누른다.

```
[jeususer@matrix jeus]$ ./jeus9000_unix_generic_ko.bin
Preparing to install
Extracting the installation resources from the installer archive...
Configuring the installer for this system's environment...
```

```
Launching installer...

===============================================================================
JEUS 9                                          (created with InstallAnywhere)
-------------------------------------------------------------------------------

Preparing CONSOLE Mode Installation...
```

4. JEUS 라이선스 준수 화면이 나타난다.

```
===============================================================================
License Agreement
-----------------

Installation and Use of JEUS 9 Requires Acceptance of the Following License
Agreement:

JEUS (Java Enterprise User Solution) Release JEUS 9
TmaxSoft Co., Ltd. (hereafter, TmaxSoft) End-User License Agreement

Product : JEUS

This is a legal agreement between you (either an individual or an company) and
TmaxSoft, Incorporated.  By opening the sealed software package and/or by
using the software, you agree to be bound by the terms of this agreement.

TmaxSoft License
1.  Grant of License:  This TmaxSoft License Agreement ("License") permits you
to use one copy of the TmaxSoft product JEUS, on any single computer, provided
the software is in use on only one computer at any one time.  If this package
is a license pack, you may make and use additional copies of the software up
to the number of licensed copies authorized.  If you have multiple licenses
for the software, then at any time you may have as many copies of the software
in use as you have licenses.
  The software is "in use" on a computer when it is loaded into the temporary
memory (i.e., RAM) or installed into the permanent memory (e.g., hard disk,
CD-ROM, or other storage devices) of that computer, except that a copy
installed on a network server for the sole purpose of distribution to other
computers is not "in use".  If the anticipated number of users of the software
will exceed the number of applicable licenses, then you must have a reasonable
```

5. JEUS 라이선스 준수에 대한 다음 정보를 보기 위해 <ENTER> 키를 누른다.

```
PRESS <ENTER> TO CONTINUE:

mechanism or process in place to ensure that the number of persons using the
software concurrently does not exceed the number of licenses.

2.  Copyright:  The software (including any images, "applets," photographs,
animations, video, audio, music and text incorporated into the software) is
owned by TmaxSoft or its suppliers and international treaty provisions.
Therefore, you must treat the software like any other copyrighted materials
(e.g., a book or musical recording) except that you may either (a) make one
copy of the software solely for backup or archival purposes, or (b) transfer
the software to a single hard disk provided you keep the original solely for
```

```
backup or archival purposes.  You may not copy the printed materials
accompanying the software, nor print copies of any user documentation provided
in "online" or electronic form.

3.  Other restrictions: This license is your proof of license to exercise the
rights granted herein and must be retained by you. You may not rent, lease, or
transfer your rights under this license on a permanent basis provided you
transfer this license, the software, and all accompanying printed materials,
retain no copies, and the recipient agrees to the terms of this license. You
may not reverse engine, decompile, or disassemble the software, except to the
extent that the foregoing restriction is expressly prohibited by applicable
law.
```

6. 라이선스 동의 여부를 결정한다. 동의를 하면 'y'를 입력한 후 <ENTER> 키를 누르고, 아니면 'n'을 입력한 후 <ENTER> 키를 누른다.

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

7. 플랫폼 목록 중 설치할 플랫폼을 입력한다.

```
===============================================================================
Choose Platform
---------------

Choose the operating system and architecture :
1)AIX 5.x, 6.x, 7.x PowerPC
2)Linux x64
Quit) Quit Installer

Choose Current System (Default: 2):
```

8. 설치 디렉터리를 선택한다. 기본적으로 제공되는 값을 사용하려면 <ENTER> 키를 누르고, 디렉터리 변경을 원하면 설치 경로를 입력한다.

```
===============================================================================
Installation Folder
-------------------

Enter the installation folder.

  Default Install Folder: /home/jeus

ENTER AN ABSOLUTE PATH, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
      :
```

9. JDK의 위치를 입력한다. <ENTER> 키를 누르면 자동으로 인식한 JDK의 위치를 사용한다. 디렉터리를 변경하려면 설치 경로를 입력한다.

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

```
Enter the JDK path.
-------------------

Enter the JDK path:

Enter the JDK path (Default: /home/jdk-17.0.2):
```

10. Master Server와 Managed Server(MS) 중 설치할 서버를 선택한다. 기본적으로는 Master Server가 설정된다.

```
================================================================================
Installation type
-----------------

Please choose the Install Set to be installed by this installer.

  ->1- Master Server
    2- Managed Server

ENTER THE NUMBER FOR THE INSTALL SET, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
   :
```

다음은 각 설치 항목에 대한 설명이다.

항목	설명
Master Server	Master Server를 설치한다. Master Server는 Managed Server를 관리하기 위한 서버이다.
Managed Server	Managed Server를 설치한다.

11. Master Server를 선택한 경우 Installation Mode를 선택한다. 기본적으로는 Production Mode가 설정된다.

```
================================================================================
Installation Mode
-----------------

* Production Mode.
- Disables JEUS Hot Swap.
- Disables Automatic Reloading.
  - Displays a warning message and recommends using a full license if a demo
license is used.

* Development Mode.
- Enables JEUS Hot Swap.
  - Enables Automatic Reloading.

  ->1- Production Mode
    2- Development Mode
    3- Cancel

ENTER THE NUMBER OF THE DESIRED CHOICE, OR PRESS <ENTER> TO ACCEPT THE
```

```
   DEFAULT:
```

다음은 각 설치 항목에 대한 설명이다.

항목	설명
Production Mode	설치할 때 Production Mode로 설치한다. JEUS Hot Swap, Automatic Reloading을 사용하지 않는다. demo license가 사용되면 경고 메시지가 출력된다.
Development Mode	설치할 때 Development Mode로 설치한다. JEUS Hot Swap, Automatic Reloading을 사용한다.

12. Master Server를 선택한 경우 Master Server에 사용할 도메인 이름을 입력한다.

```
===============================================================================
input JEUS Environments :: JEUS Domain Name
----------------------------------------

Enter the
- JEUS Domain Name
- Enter alphanumeric characters (case-sensitive).

Domain Name (Default: jeus_domain):
```

13. Master Server를 선택한 경우 JEUS 관리자의 패스워드를 입력한다. 이 패스워드는 "administrator" ID로 할당된다.

```
===============================================================================
Password Input
--------------

Enter the Password for the administrator account.


Input Password:
```

패스워드는 JEUS를 기동하는 필수적인 요소이다. 입력한 패스워드는 기억장치에 기록되며, 주의 깊게 입력해야 한다.

14. 다음 화면은 설치에 관련한 요약정보를 보여준다. <ENTER> 키를 누르면 설치가 진행된다.

```
===============================================================================
Pre-Installation Summary
------------------------

Review the Following Before Continuing:

Product Name:
    JEUS 9
```

```
Install Folder:
    /home/jeus

Disk Space Information (for Installation Target):
    Required:      840,921,580 Bytes
    Available: 549,948,723,200 Bytes

PRESS <ENTER> TO CONTINUE:
```

15. 진행 바와 함께 설치가 진행된다.

```
===============================================================================
Installing...
-------------

 [==================|==================|==================|==================]
 [------------------|--------------
```

16. 설치가 완료되면 다음의 메시지가 출력된다.

```
===============================================================================
Installation Complete
---------------------

JEUS 9 has been successfully installed to:

     /home/jeus

PRESS <ENTER> TO EXIT THE INSTALLER:
```

2.3. 설치 확인

JEUS를 설치한 후 사용에 필요한 환경변수를 설정해야 한다. 환경변수를 설정하고 JEUS를 기동해서 설치가 정상적으로 이루어졌는지 확인한다. 환경변수를 설정하기 전에 JEUS가 설치된 경로에 디렉터리 구조를 확인한다.

2.3.1. 디렉터리 구조

다음은 JEUS를 설치했을 때의 전체 디렉터리 구조이다.

```
{JEUS_HOME}
    |-- bin
    |    |--[01]startMasterServer
    |    |--[01]startManagedServer
    |    |--[01]stopServer
    |    |--[01]jeusadmin
    |--derby
    |--docs
```

```
    |--lib
    |   |--shared
    |        |--[X]libraries.xml
    |--license
    |--setup
    |--templates
    |--samples
    |--webserver
    |--domains
          |--<domain_name>
                  |--.applications
                  |--.deploymentplans
                  |--.libraries
                  |--bin
                  |--config
                  |--lib
                  |   |--application
                  |--servers
                      |--<server_name>
                             |--.workspace
                             |     |--deployed
                             |     |--tmp
                             |     |--web-nio
                             |     |--tmlog
                             |--bin
                             |--lib
                             |   |--application
                             |--logs

* Legend
- [01]: binary or executable file
- [X] : XML document
- [J] : JAR file
- [T] : Text file
- [C] : Class file
- [V] : java source file
- [DD] : deployment descriptor
```

다음은 디렉터리와 파일의 설명이다.

{JEUS_HOME}

JEUS의 최상위 디렉터리로 실제 디렉터리 이름과 위치는 설치할 때 결정된다.

bin

서버의 시작 및 종료 스크립트인 startMasterServer, startManagedServer, stopServer와 JEUS 콘솔 툴(jeusadmin)과 같은 실행 파일들이 위치한다.

derby

샘플 애플리케이션이나 테스트에서 쉽게 사용할 수 있도록 Apache Derby를 포함시킨다.

docs

JEUS에서 제공하는 API에 대한 Javadoc이 존재한다.

lib

JEUS가 기동하는 데 필요한 라이브러리가 존재한다. shared 디렉터리를 제외한 나머지 디렉터리들은 사용자가 접근할 필요가 없다.

디렉터리	설명
shared	shared 디렉터리에는 애플리케이션에서 사용하는 라이브러리가 존재한다. shared 디렉터리의 라이브러리를 사용하려면 libraries.xml에 라이브러리의 정보를 추가해야 한다. 그리고 해당 라이브러리를 사용할 애플리케이션의 JEUS Deployment Descriptor(DD)에서 해당 라이브러리에 대한 레퍼런스 정보를 지정해야 한다. shared 라이브러리에 대한 자세한 설명은 JEUS Applications & Deployment 안내서의 "공유 라이브러리"를 참고한다.

license

JEUS 라이선스 파일이 위치한다. 라이선스 파일은 JEUS가 실행되기 위해서 반드시 필요한 파일이다.

setup

JEUS 설치 후 사용할 수 있도록 환경을 구축하기 위해 필요한 파일들이 위치한다.

templates

각종 설정과 환경 등의 template 파일이 위치한다.

samples

JEUS의 예제 파일들이 위치한다.

webserver

JEUS가 설치될 때 JEUS 웹 서버가 설치되는 디렉터리이다. 자세한 내용은 "JEUS Web Engine 안내서"를 참조한다.

domains

하위에 도메인별로 DOMAIN_HOME과 JEUS_HOME에서 사용하는 노드 정보가 포함된 nodes.xml이 존재한다.

다음의 디렉터리 및 파일들은 DOMAIN_HOME 아래에 위치한다.

- .applications

  해당 도메인에서 관리하는 애플리케이션 파일이 존재한다.

  install-application, **uninstall-application** 명령어를 통해 추가 및 삭제가 가능하다. JEUS가 사용하는 디렉터리로 사용자의 접근을 제한한다. 각 명령어에 대한 설명은 JEUS Reference 안내서의 "install-application"과 "uninstall-application"을 참고한다.

- .deploymentplans

  해당 도메인에서 관리하는 Deployment Plan 파일이 존재한다.

  install-deployment-plan, **uninstall-deployment-plan** 명령어를 통해 추가 및 삭제가 가능하다. JEUS가 사용하는 디렉터리로 사용자의 접근을 제한한다. 각 명령어에 대한 설명은 JEUS Reference

안내서의 "install-deployment-plan"과 "uninstall-deployment-plan"을 참고한다.

◦ .libraries

해당 도메인에서 관리하는 라이브러리 파일이 존재한다.

install-library, **uninstall-library** 명령어를 통해 추가 및 삭제가 가능하다. JEUS가 사용하는 디렉터리로 사용자의 접근을 제한한다. 각 명령어에 대한 설명은 JEUS Reference 안내서의 "install-library"와 "uninstall-library"를 참고한다.

◦ bin

해당 도메인에 속한 Master Server와 Managed Server의 시작 및 종료 스크립트가 위치한다. JEUS_HOME/bin의 startMasterServer, startManagedServer, stopServer와 동일한 기능을 수행하지만 도메인 이름을 설정할 필요가 없다.

◦ config

도메인의 설정 파일인 domain.xml이 변경된 경우 이전 이력을 위해 존재하는 백업 파일들이 위치한다. 도메인 설정에 대한 자세한 설명은 JEUS Domain 안내서의 "도메인 설정변경"을 참고한다.

구분	설명
security	◦ SYSTEM_DOMAIN : 도메인 단위로 적용되는 보안 도메인 파일인 accounts.xml, policies.xml이 존재하며, 각 XML 파일은 jeusadmin을 통해 동적 설정 변경이 가능하다. 보안 도메인 설정에 대한 자세한 설명은 JEUS Security 안내서의 "보안 도메인 설정"을 참고한다. ◦ security-domains.xml : JEUS의 보안 도메인에 대한 설정을 저장하는 파일이다. ◦ security.key : 대칭키 암호화 알고리듬에 대한 Key를 저장하는 파일로 JEUS_HOME/bin/encryption을 수행하면 생성된다. security.key 파일에 대한 자세한 설명은 JEUS Security 안내서의 "패스워드 보안 설정"을 참고한다. ◦ policy : Java permission 설정 파일이다. JEUS의 보안 시스템과는 별도로 Java SE Security Manager에서 사용된다.
servlet	◦ web.xml : web.xml을 개별적으로 가지고 있지 않은 경우 웹 엔진이 사용할 웹 모듈의 web.xml이다. 기본값은 빈 XML 파일이다. ◦ webcommon.xml : 도메인 내 서버의 웹 엔진의 모든 웹 모듈에 적용되는 공통 설정 파일이다. 설정에 대한 자세한 설명은 JEUS Web Engine 안내서의 "디렉터리 구조"를 참고한다.

◦ lib/application

도메인 전체에 적용하고 싶은 애플리케이션 라이브러리를 위치시키는 디렉터리다.

SERVER_HOME에 존재하는 애플리케이션 라이브러리와 충돌이 발생할 경우 SERVER_HOME/lib/application이 우선되고 경고 메시지가 남는다. lib/application 디렉터리에 대한 자세한 설명은 JEUS Applications & Deployment 안내서의 "lib/application 디렉터리"를 참고한다.

◦ servers

이 디렉터리 하위에 SERVER_HOME 디렉터리가 서버 이름으로 생성된다. SERVER_HOME 디렉터리 구조에 대한 자세한 설명은 JEUS Server 안내서의 "서버 디렉터리 구조"를 참고한다.

디렉터리	설명
.workspace	JEUS가 사용하는 서버별 공간으로 사용자가 변경해서는 안 된다.
bin	서버의 시작/종료 스크립트를 포함하고 있다. JEUS_HOME/bin의 스크립트와 동일한 기능을 수행하지만 도메인 이름과 서버 이름을 설정할 필요가 없다. ◦ Master Server일 경우 : startMasterServer/stopServer가 존재한다. ◦ Managed Server일 경우 : startManagedServer/stopserver가 존재한다.
lib/application	서버에 적용하고 싶은 애플리케이션 라이브러리가 존재한다. 도메인 범위의 라이브러리(DOMAIN_HOME/lib/application)보다 우선순위가 높다. 라이브러리가 충돌할 경우 이 디렉터리에 존재하는 파일이 적용되며 경고 메시지가 남는다. lib/application에 대한 자세한 설명은 JEUS Applications & Deployment 안내서의 "lib/application 디렉터리"를 참고한다.
logs	서버의 Launcher 로그, 서버 로그, 엑세스 로그 파일이 남는다. 자세한 내용은 JEUS Server 안내서의 "Logging"을 참고한다.

2.3.2. 환경설정

JEUS를 사용하기 위해서는 환경변수가 필요하다. 설치 과정에서 일부 환경변수를 설정하지만 경우에 따라서는 수정해서 사용한다. 해당 변수들은 설치할 때 환경변수 PATH는 **.profile/.cshrc**에 적용되고, 그 외 환경변수는 **$JEUS_HOME/bin/jeus.properties** 파일에 설정된다. 만약 서버 별로 서로 다른 환경변수를 설정하려면 **$JEUS_HOME/bin/<*SERVER_NAME*>.properties** 파일을 작성한다. 서버를 기동하는 경우 [-server] 옵션을 사용하여 서버 이름을 지정한다.

다음은 주요 환경변수에 대한 설명이다.

환경변수	설명
PATH	시스템 경로를 설정한다. 다음을 포함하고 있어야 한다. ◦ /home/jeus/bin ◦ /home/jeus/lib/system
JEUS_HOME	JEUS 설치 디렉터리를 설정한다. (예: /home/jeus)
JEUS_LIBPATH	JEUS 라이브러리 파일 경로를 설정한다. (예: /home/jeus/lib/system)
VM_TYPE	Java HotSpot JVM 사용 유무를 설정한다. (예: hotspot or old)
USERNAME	Administrator 계정의 ID를 설정한다.
PASSWORD	Administrator 패스워드를 설정한다.
JAVA_HOME	JDK 설치 디렉터리 경로를 설정한다. (예: /usr/jdk17)
ANT_HOME	ANT 설치 디렉터리 경로를 설정한다. (예: home/jeus/lib/etc/ant)

환경변수	설명
JAVA_ARGS	JDK 파라미터를 설정한다.
JAVA_VENDOR	JDK 벤더를 설정한다. (예: Sun, IBM, HP)

환경변수를 설정할 때 C 셸이라면 JEUS_HOME 변수를 설정하기 위해서는 'setenv'를 사용한다.

```
setenv JEUS_HOME “/home/jeus”
```

다음은 시스템 PATH 설정하는 예이다.

```
setenv PATH “${PATH}:/home/jeus/bin:
/home/jeus/lib/system”
```

Java의 실행 디렉터리(**/usr/jdk17/bin**)는 JEUS에서 사용하게 되므로 환경변수에 추가할 때는 앞 쪽에 추가하도록 한다.

2.3.3. 기동 확인

JEUS 설치가 정상적으로 완료되었는지 확인하기 위해 다음 단계를 진행해서 JEUS를 기동한다.

1. 콘솔 프롬프트에 startMasterServer 명령어를 입력해서 Master Server(MASTER)를 시작한다. 일반적으로 관리자의 계정은 'administrator'이고 패스워드는 JEUS를 설치할 때 입력한 값이다.

   다음은 명령어를 실행해서 Master Server(MASTER)를 시작하는 방법이다.

   ```
   startMasterServer -u <user_name> -p <password>
   ```

   기동이 완료되면 "Successfully started the server. The server state is now RUNNING."라는 메시지가 출력된다.

   ```
   [was@localhost ~]$ startMasterServer -u administrator -p <password>
   ************************************************************
     - JEUS Home          : /home/jeus
     - Added Java Option : -Djeus.io.buffer.size-per-pool=81920 -Djeus.cdi.enabled=false
   -Djeus.jms.server.manager.produce-wait-strategy-type=blocking
   -Djeus.servlet.sortWebinfLibraries=name_asc
   ************************************************************

   =============== JEUS LICENSE INFORMATION ================
   == VERSION : JEUS 9 (9.0.0.0-b15)
   == EDITION: Enterprise (Trial License)
   == NOTICE: This license restricts the number of allowed clients.
   == Max. Number of Clients: 5
   ==========================================================
   ```

```
[2024.09.25 17:54:09][1] [launcher-1] [Config-0153] DomainConfigServiceProvider is
jeus.service.descriptor.JEUSDomainDescriptorFile.
This license is not appropriate for product runtime mode. Replace the license with an appropriate
one.
[2024.09.25 17:54:10][1] [launcher-1] [Config-0157] SecurityDomainsConfigServiceProvider is
jeus.service.descriptor.SecurityDomainsDescriptorFile.
[2024.09.25 17:54:10][2] [launcher-1] [Launcher-0012] Starting the server [server1] with the
command
 /home/jdk-17.0.2/bin/java -Dserver1 -Xms1024m -Xmx1024m -XX:MetaspaceSize=128m
-XX:MaxMetaspaceSize=512m -Djeus.io.buffer.size-per-pool=81920 -Djeus.cdi.enabled=false
-Djeus.jms.server.manager.produce-wait-strategy-type=blocking
-Djeus.servlet.sortWebinfLibraries=name_asc -server
-Xbootclasspath/p:/home/jeus/lib/system/extension.jar -classpath
/home/jeus/lib/system/bootstrap.jar
-Djava.security.policy=/home/jeus/domains/jeus_domain/config/security/policy
-Djava.library.path=/home/jeus/lib/system:/home/webtob5004_B231_0_38//lib:/home/webtob5004_B231_0
_38//lib:/home/webtob5004_B231_0_38//lib: -Djava.endorsed.dirs=/home/jeus/lib/endorsed
-Djeus.properties.replicate=jeus,sun.rmi,java.util,java.net
-Djava.util.logging.config.file=/home/jeus/bin/logging.properties
-Dsun.rmi.dgc.server.gcInterval=3600000
-Djava.util.logging.manager=jeus.util.logging.JeusLogManager -Djeus.home=/home/jeus
-Djava.net.preferIPv4Stack=true -Djeus.tm.checkReg=true -Dsun.rmi.dgc.client.gcInterval=3600000
-Djeus.domain.name=jeus_domain -Djava.naming.factory.initial=jeus.jndi.JNSContextFactory
-Djava.naming.factory.url.pkgs=jeus.jndi.jns.url -Djeus.server.protectmode=false
-Dis.jeus.master=true -Dsun.net.http.errorstream.enableBuffering=true
-XX:+UnlockDiagnosticVMOptions -XX:+LogVMOutput
-XX:LogFile=/home/jeus/domains/jeus_domain/servers/server1/logs/jvm.log
jeus.server.admin.MasterServerBootstrapper -domain jeus_domain -u administrator -verbose -server
server1 .
[2024.09.25 17:54:10][2] [launcher-1] [Launcher-0014] The server[server1] is being started ...
[2024.09.25 17:54:10][1] [server1-1] [Config-0153] DomainConfigServiceProvider is
jeus.service.descriptor.JEUSDomainDescriptorFile.
[2024.09.25 17:54:10][1] [server1-1] [Config-0157] SecurityDomainsConfigServiceProvider is
jeus.service.descriptor.SecurityDomainsDescriptorFile.
[2024.09.25 17:54:12][2] [server1-1] [SERVER-0248] The JEUS server is STARTING.
[2024.09.25 17:54:12][0] [server1-1] [SERVER-0000] Version information - JEUS 9 (9.0.0.0-b15).

... 중략

[2024.09.25 17:54:13][2] [server1-1] [SERVER-0248] The JEUS server is STANDBY.
[2024.09.25 17:54:13][2] [server1-1] [SERVER-0248] The JEUS server is STARTING.
[2024.09.25 17:54:13][2] [server1-1] [WEB-3413] The web engine is ready to receive requests.
[2024.09.25 17:54:13][2] [server1-1] [NET-0002] Beginning to listen to
NonBlockingChannelAcceptor: qpsp1:8808.
[2024.09.25 17:54:13][2] [server1-1] [UNIFY-0100] Listener information

+------------+-------+--------------+-----------+----------------------------------------+
| Name       | SSL   | Address:Port | Protocol  | Virtual Listener                       |
+------------+-------+--------------+-----------+----------------------------------------+
| base       | false | 0.0.0.0:9736 | VIRTUAL   | ClassFTP                               |
|            |       |              |           | SecurityServer                         |
|            |       |              |           | JMXConnectionServer/JEUSMP_adminServer |
|            |       |              |           | JMXConnectionServer/JeusMBeanServer    |
|            |       |              |           | TransactionManager                     |
|            |       |              |           | JMSServiceChannel-default              |
|            |       |              |           | FileTransfer                           |
|            |       |              |           | JNDI                                   |
|            |       |              | HTTP      |                                        |
+------------+-------+--------------+-----------+----------------------------------------+
```

```
| http-server| false | 0.0.0.0:8088 | ProObject |                                      |
|            |       |              | HTTP      |                                      |
+------------+-------+--------------+-----------+--------------------------------------+

... 중략

[2024.09.25 17:54:29][2] [launcher-13] [Launcher-0034] The server[server1] initialization
completed successfully[pid : 473].
[2024.09.25 17:54:29][0] [launcher-1] [Launcher-0040] Successfully started the server[server1].
The server state is now RUNNING..**
```

1. “Invalid License” 메시지가 나타나면 라이선스가 잘못된 것이다. TmaxSoft로부터 라이선스를 취득하여 $JEUS_HOME/license 디렉터리에 복사한다.
2. 모든 단계들의 진행과 환경변수가 정확히 설정되어 있는지 확인한다. 특히, /jeus/bin 디렉터리가 시스템 경로(startMasterServer 스크립트가 실행될 수 있도록)에 있는 것을 확인한다.

2. Managed Server(MS)를 시작한다. Managed Server는 startManagedServer 명령어를 실행할 수 있다.
   - **startManagedServer 명령어 실행**

     다음은 startManagedServer 명령어를 이용해 Managed Server(MS)를 기동하는 방법이다.

     ```
     startManagedServer -domain <domain_name> -server <server_name> -u <user_name> -p <
     password>
     ```

     콘솔 프롬프트에 명령어를 입력하면 다음 메시지를 출력한다. 일반적으로 관리자의 계정은 'administrator'이고 패스워드는 JEUS를 설치할 때 입력한 값이다. JEUS MS가 정상적으로 부트되어 기동되면 "Successfully started the server. The server state is now RUNNING."라는 메시지가 출력된다.

     ```
     [was@localhost ~]$ startManagedServer -domain jeus_domain -server server2 -u administrator -p
     <password>
     ************************************************************
       - JEUS Home         : /home/jeus
       - Added Java Option : -Djeus.io.buffer.size-per-pool=81920 -Djeus.cdi.enabled=false
     -Djeus.jms.server.manager.produce-wait-strategy-type=blocking
     -Djeus.servlet.sortWebinfLibraries=name_asc
     ************************************************************

     =============== JEUS LICENSE INFORMATION ================
     == VERSION : JEUS 9 (9.0.0.0-b15)
     == EDITION: Enterprise (Trial License)
     == NOTICE: This license restricts the number of allowed clients.
     == Max. Number of Clients: 5
     ==========================================================
     [2024.09.25 17:55:40][2] [launcher-1] [SERVER-0201] Successfully connected to the JEUS Master
     Server(localhost:9736).
     [2024.09.25 17:55:40][2] [launcher-1] [Launcher-0058] All local configurations are up-to-
     date.
     ```

```
[2024.09.25 17:55:40][1] [launcher-1] [Config-0157] SecurityDomainsConfigServiceProvider is
jeus.service.descriptor.SecurityDomainsDescriptorFile.
[2024.09.25 17:55:41][1] [launcher-1] [Config-0153] DomainConfigServiceProvider is
jeus.service.descriptor.JEUSDomainDescriptorFile.
This license is not appropriate for product runtime mode. Replace the license with an
appropriate one.
[2024.09.25 17:55:41][2] [launcher-1] [Launcher-0012] Starting the server [server2] with the
command
 /home/jdk-17.0.2/bin/java -Dserver2 -Djeus.io.buffer.size-per-pool=81920
-Djeus.cdi.enabled=false -Djeus.jms.server.manager.produce-wait-strategy-type=blocking
-Djeus.servlet.sortWebinfLibraries=name_asc -server
-Xbootclasspath/p:/home/jeus/lib/system/extension.jar -classpath
/home/jeus/lib/system/bootstrap.jar
-Djava.security.policy=/home/jeus/domains/jeus_domain/config/security/policy
-Djava.library.path=/home/jeus/lib/system:/home/webtob5004_B231_0_38//lib:/home/webtob5004_B2
31_0_38//lib: -Djava.endorsed.dirs=/home/jeus/lib/endorsed
-Djeus.properties.replicate=jeus,sun.rmi,java.util,java.net
-Djava.util.logging.config.file=/home/jeus/bin/logging.properties
-Dsun.rmi.dgc.server.gcInterval=3600000
-Djava.util.logging.manager=jeus.util.logging.JeusLogManager -Djeus.home=/home/jeus
-Djava.net.preferIPv4Stack=true -Djeus.tm.checkReg=true
-Dsun.rmi.dgc.client.gcInterval=3600000 -Djeus.domain.name=jeus_domain
-Djava.naming.factory.initial=jeus.jndi.JNSContextFactory
-Djava.naming.factory.url.pkgs=jeus.jndi.jns.url -Djeus.server.protectmode=false
-Djeus.master.port=9736 -Djeus.master.host=localhost -Djeus.master.protocol=http
-XX:+UnlockDiagnosticVMOptions -XX:+LogVMOutput
-XX:LogFile=/home/jeus/domains/jeus_domain/servers/server2/logs/jvm.log
jeus.server.ServerBootstrapper -domain jeus_domain -server server2 -u administrator -verbose
.
[2024.09.25 17:55:41][2] [launcher-1] [Launcher-0014] The server[server2] is being started
...
[2024.09.25 17:55:42][1] [server2-1] [Config-0153] DomainConfigServiceProvider is
jeus.service.descriptor.JEUSDomainDescriptorFile.
[2024.09.25 17:55:42][1] [server2-1] [Config-0157] SecurityDomainsConfigServiceProvider is
jeus.service.descriptor.SecurityDomainsDescriptorFile.
[2024.09.25 17:55:43][2] [server2-1] [SERVER-0248] The JEUS server is STARTING.
[2024.09.25 17:55:43][0] [server2-1] [SERVER-0000] Version information - JEUS 9 (9.0.0.0-
b15).

... 중략

[2024.09.25 17:55:45][2] [server2-50] [WEB-3484] ServletContext[name=healthcheck,
path=/health, ctime=Mon Sep 25 17:55:45 KST 2024, apptime=1682326357716, index=1682326357716]
started successfully.
[2024.09.25 17:55:45][2] [server2-50] [Deploy-0099] Successfully started the
application[healthcheck, 1682326357716].
[2024.09.25 17:55:45][0] [server2-1] [SERVER-0242] Successfully started the server.
[2024.09.25 17:55:45][2] [server2-1] [SERVER-0248] The JEUS server is RUNNING.
[2024.09.25 17:55:45][2] [server2-1] [SERVER-0401] The elapsed time to start: 3626ms.
[2024.09.25 17:55:45][2] [launcher-14] [Launcher-0034] The server[server2] initialization
completed successfully[pid : 792].
[2024.09.25 17:55:45][0] [launcher-1] [Launcher-0040] Successfully started the
server[server2]. The server state is now RUNNING.**
```

1. “Invalid License” 메시지가 나타나면 라이선스가 잘못된 것이다.
   TmaxSoft로부터 라이선스를 취득하여 $JEUS_HOME/license 디렉터리에 복사한다.

2. 모든 단계들의 진행과 환경변수가 정확히 설정되어 있는지 확인한다. 특히, /jeus/bin 디렉터리가 시스템 경로(startManagedServer 스크립트가 실행될 수 있도록)에 있는 것을 확인한다.

3. JEUS의 관리 콘솔에 접속하기 위해 다른 콘솔 창에서 **jeusadmin** 명령어를 실행한다. 일반적으로 관리자의 계정은 'administrator'이고 패스워드는 JEUS를 설치할 때 입력한 값이다.

```
[was@localhost ~]$ jeusadmin -u administrator -p <password>
Attempting to connect to 127.0.0.1:9736.
The connection has been established to JEUS Master Server [adminServer] in the domain
[jeus_domain].
JEUS 9 Administration Tool
To view help, use the 'help' command.
[MASTER]jeus_domain.adminServer>
```

4. 잠시 후 프롬프트가 다시 뜨면 JEUS가 제대로 기동되었고, 다시 명령어를 받을 수 있는 상태가 되었다는 것을 나타낸다.

5. 콘솔에서 **jeusadmin** 툴에 로그인을 한 후에 **local-start-server**와 **local-shutdown** 명령어로 JEUS 서버를 제어할 수 있다. JEUS 서버를 다운시키려면 **local-shutdown** 명령어를 실행한다.

```
[MASTER]jeus_domain.adminServer>local-shutdown
Executing this command affects the service. Do you want to continue? (y/n)y
The server [adminServer] has been shut down successfully.
```

6. jeusadmin을 종료하기 위해서 **exit** 명령어를 실행한다.

```
offline>exit
```

2.4. 제거

본 절에서는 콘솔 모드에서 제거하는 과정에 대해서 설명한다.

다음은 콘솔 모드에서 JEUS를 삭제하는 과정에 대한 설명이다.

1. JEUS가 설치된 경로에서 $JEUS_HOME/UninstallerData/Uninstall을 실행하여 JEUS Core와 JEUS 설치 디렉터리를 삭제한다.

```
[was@localhost ~ UninstallerData]$./Uninstall
```

2. 삭제 과정이 진행된다. 제거가 완료되면 완료되었다는 메시지가 표시된다.

```
===============================================================================
JEUS 9                                        (created with InstallAnywhere)
```

```
-------------------------------------------------------------------------------

Preparing CONSOLE Mode Uninstallation...

===============================================================================
Uninstall JEUS 9
------------------

About to uninstall...

JEUS 9

This will remove features installed by InstallAnywhere.  It will not remove
files and folders created after the installation.

PRESS <ENTER> TO CONTINUE:
===============================================================================

Check JEUS process...
-
===============================================================================
Uninstalling...
---------------

...*
*
************************
************************
************************
************************
...*
*
************************
************************
************************
************************
...*
*
************************
************************
************************
************************
...

===============================================================================
Uninstallation Complete
-----------------------

All items were successfully uninstalled.
```

2.5. 제거 확인

JEUS가 설치된 경로에 디렉터리와 설치된 파일이 삭제된 것을 확인한다. JEUS가 설치된 후 생성된 파일은 삭제되지 않으므로 해당 파일은 수동으로 삭제해야 한다.

Part Ⅱ. JEUS 시작하기

3. 개요

본 부분은 TmaxSoft의 Web Application Server(WAS)인 JEUS를 사용하여 프로그램을 처음 작성하는 사용자를 위한 Part로 다음의 내용으로 구성된다.

- **시스템 설정**

  JEUS의 기본적인 환경설정 방법과 운영 방법에 대해 설명한다.

- **WebTier 사용**

  웹 애플리케이션의 패키징 방법을 설명하고 예제 Servlet, JSP, JSTL, JSF 페이지를 실행한다.

- **EJB 사용**

  EJB 모듈의 패키징 및 배치 방법, Stateless Session Bean과 Java Persistence API의 이용 방법에 대해 설명한다.

- **웹 서비스 사용**

  웹 서비스의 패키징 및 배치 방법, 서블릿 기반 End-point와 EJB 기반 Endpoint의 이용 방법에 대해 설명한다.

각 부분은 순서에 따라 진행하도록 되어 있으므로 JEUS Web Application Server에 대하여 완벽히 이해하려면 이 문서에 담겨있는 내용을 실제로 실행해 보는 것을 권장한다.

좀 더 상세한 기술적인 내용의 문서를 보려면 "JEUS Server 안내서", "JEUS EJB 안내서", "JEUS Web Engine 안내서" 등을 참고한다.

4. 시스템 설정

본 장에서는 JEUS의 환경설정 및 기동 방법에 대한 기본적인 내용을 설명한다.

4.1. 개요

Jeusadmin을 사용해서 시스템을 구성하는 방법은 다음의 순서로 설명한다.

- 기본 환경설정
- Managed Server(MS)의 추가와 설정 방법
- 데이터소스 추가

4.2. 기본 환경설정

JEUS Jeusadmin 콘솔 통해 JEUS의 모든 요소를 관리할 수 있는 서비스를 제공한다. 사용자는 쉽게 JEUS에 접근하여 시스템 설정과 모니터링 및 애플리케이션을 관리할 수 있다.

다음 순서에 따라 Jeusadmin을 실행한다.

1. startMasterServer 커맨드 실행하여 JEUS MASTER를 실행한다.

   다음은 JEUS MASTER를 실행하는 예제이다.

```
[was@localhost bin]$ startMasterServer -u administrator -p <password>
************************************************************
  - JEUS Home         : /home/jeus
- Added Java Option : -Djeus.io.buffer.size-per-pool=81920 -Djeus.cdi.enabled=false
-Djeus.jms.server.manager.produce-wait-strategy-type=blocking
-Djeus.servlet.sortWebinfLibraries=name_asc
************************************************************

=============== JEUS LICENSE INFORMATION ================
== VERSION : JEUS 9 Fix#0 (9.0.0.0-b15)
== EDITION: Enterprise (Trial License)
== NOTICE: This license restricts the number of allowed clients.
== Max. Number of Clients: 5
==========================================================
[2024.09.25 17:54:09][1] [launcher-1] [Config-0153] DomainConfigServiceProvider is
jeus.service.descriptor.JEUSDomainDescriptorFile.
This license is not appropriate for product runtime mode. Replace the license with an appropriate
one.
[2024.09.25 17:54:10][1] [launcher-1] [Config-0157] SecurityDomainsConfigServiceProvider is
jeus.service.descriptor.SecurityDomainsDescriptorFile.
[2024.09.25 17:54:10][2] [launcher-1] [Launcher-0012] Starting the server [server1] with the
command
 /home/jdk-17.0.2/bin/java -Dserver1 -Xms1024m -Xmx1024m -XX:MetaspaceSize=128m
-XX:MaxMetaspaceSize=512m -Djeus.io.buffer.size-per-pool=81920 -Djeus.cdi.enabled=false
-Djeus.jms.server.manager.produce-wait-strategy-type=blocking
-Djeus.servlet.sortWebinfLibraries=name_asc -server
```

```
-Xbootclasspath/p:/home/jeus/lib/system/extension.jar -classpath
/home/jeus/lib/system/bootstrap.jar
-Djava.security.policy=/home/jeus/domains/jeus_domain/config/security/policy
-Djava.library.path=/home/jeus/lib/system:/home/webtob5004_B231_0_38//lib:/home/webtob5004_B231_0
_38//lib:/home/webtob5004_B231_0_38//lib: -Djava.endorsed.dirs=/home/jeus/lib/endorsed
-Djeus.properties.replicate=jeus,sun.rmi,java.util,java.net
-Djava.util.logging.config.file=/home/jeus/bin/logging.properties
-Dsun.rmi.dgc.server.gcInterval=3600000
-Djava.util.logging.manager=jeus.util.logging.JeusLogManager -Djeus.home=/home/jeus
-Djava.net.preferIPv4Stack=true -Djeus.tm.checkReg=true -Dsun.rmi.dgc.client.gcInterval=3600000
-Djeus.domain.name=jeus_domain -Djava.naming.factory.initial=jeus.jndi.JNSContextFactory
-Djava.naming.factory.url.pkgs=jeus.jndi.jns.url -Djeus.server.protectmode=false
-Dis.jeus.master=true -Dsun.net.http.errorstream.enableBuffering=true
-XX:+UnlockDiagnosticVMOptions -XX:+LogVMOutput
-XX:LogFile=/home/jeus/domains/jeus_domain/servers/server1/logs/jvm.log
jeus.server.admin.MasterServerBootstrapper -domain jeus_domain -u administrator -verbose -server
server1 .
[2024.09.25 17:54:10][2] [launcher-1] [Launcher-0014] The server[server1] is being started ...
[2024.09.25 17:54:10][1] [server1-1] [Config-0153] DomainConfigServiceProvider is
jeus.service.descriptor.JEUSDomainDescriptorFile.
[2024.09.25 17:54:10][1] [server1-1] [Config-0157] SecurityDomainsConfigServiceProvider is
jeus.service.descriptor.SecurityDomainsDescriptorFile.
[2024.09.25 17:54:12][2] [server1-1] [SERVER-0248] The JEUS server is STARTING.
[2024.09.25 17:54:12][0] [server1-1] [SERVER-0000] Version information - JEUS 9 Fix#0 (9.0.0.0-
b15).

... 중략

[2024.09.25 17:54:29][2] [launcher-13] [Launcher-0034] The server[server1] initialization
completed successfully[pid : 473].
[2024.09.25 17:54:29][0] [launcher-1] [Launcher-0040] Successfully started the server[server1].
The server state is now RUNNING..
```

startMasterServer 스크립트는 JEUS_HOME/bin/ 디렉터리에 위치하며 시스템 경로(path)에 설정되어 있어야 한다.

4.3. Managed Server(MS)의 추가와 설정

MS는 실제 애플리케이션을 서비스하기 위한 엔진들과 여러 서비스들을 관장하는 서버 인스턴스를 의미한다. MS는 도메인에 여러 개 존재할 수 있다. MS의 주요 역할은 사용자가 deploy하는 애플리케이션을 서비스하고, 애플리케이션이 필요로 하는 리소스나 서비스를 제공하는 것이다.

Managed Server 추가

다음은 새로운 MS를 추가하고, 추가된 MS에 리스너를 추가하는 방법이다.

1. jeusadmin 접속하여 **add-server** 명령어를 통해 MS를 추가한다.

```
[MASTER]jeus_domain.server1>add-server server2
Successfully performed the ADD operation for server (server2).
 NOTICE : base-addr [0.0.0.0] base-port [9736] http-port [8088]
```

```
Check the results using "list-servers or add-server".
```

MS를 기동하려면 'BASE'라는 리스너를 이용하는데, 기본값이 '9736'으로 설정된다. 이는 MASTER와 값이 동일하여 정상적으로 기동되지 않을 수 있으므로 변경해야 한다.

2. MS 추가가 완료되면 **server-info** 명령어를 통해 동적 설정으로 MS가 생성된 것을 확인할 수 있다.

```
Information about Domain (jeus_domain)
================================================================================
+--------+---------+-----+-----+-----+-----------+--------+-----------+--------+
| Server |  Status |Node | PID | Clu |   Latest  |  Need  |   Listen  |Running |
|        |         |Name |     |ster |Start Time |   to   |   Ports   |Engines |
|        |         |     |     |     |     /     |Restart |           |        |
|        |         |     |     |     | Shutdown  |        |           |        |
|        |         |     |     |     |   Time    |        |           |        |
+--------+---------+-----+-----+-----+-----------+--------+-----------+--------+
| adminS | RUNNING | N/A | 291 | N/A |2024-09-24 | false  | base-0.0. | jms,   |
|erver   |(00:38:2 |     |34   |     |(화) 오후  |        |0.0:9736   |web, ejb|
|(*)     |7)       |     |     |     |03:06:43   |        | http-serv |        |
|        |         |     |     |     |KST        |        |er-0.0.0.0 |        |
|        |         |     |     |     |           |        |:8808      |        |
+--------+---------+-----+-----+-----+-----------+--------+-----------+--------+
| server2| SHUTDOWN| N/A | N/A | N/A |2024-09-24 | N/A    | N/A       | N/A    |
|        |         |     |     |     |(화) 오후  |        |           |        |
|        |         |     |     |     |03:06:43   |        |           |        |
|        |         |     |     |     |KST        |        |           |        |
+--------+---------+-----+-----+-----+-----------+--------+-----------+--------+
================================================================================
```

3. **list-server-listener** 명령어를 통해 설정한 Listener 확인이 가능하다.

```
[MASTER]jeus_domain.server1>list-server-listeners -server server2
============================================================
+-------------------------------------+---------+------+
|             listener-name           | address | port |
+-------------------------------------+---------+------+
| base                                | 0.0.0.0 | 9736 |
| http-server                         | 0.0.0.0 | 8088 |
+-------------------------------------+---------+------+
============================================================
```

4. **modify-listener** 명령어를 통해 Listener 관련 수정을 진행할 수 있다.

```
[MASTER]jeus_domain.server1>modify-listener -server server2 -name base -port 9512
Executed successfully, but some configurations were not applied dynamically. It might be
necessary to restart the server.
Check the result using 'list-server-listeners -server server2 -name base.

[MASTER]jeus_domain.server1>list-server-listeners -server server2
============================================================
```

```
+-------------------------------------+---------+------+
|             listener-name           | address | port |
+-------------------------------------+---------+------+
| base                                | 0.0.0.0 | 9512 |
| http-server                         | 0.0.0.0 | 8088 |
+-------------------------------------+---------+------+
============================================================
```

HTTP 리스너 및 커넥터 설정

새로 추가된 MS를 구동하기 위한 설정은 기본 설정으로 충분하며, 추가로 HTTP 리스너 추가하여 웹 엔진을 이용한 서비스하기 위해서는 리스너 및 커넥터 설정을 추가적으로 진행해야 한다.

1. jeusadmin 접속하여 **add-listener** 명령어를 사용하여 리스너를 추가할 수 있다.

```
[MASTER]jeus_domain.server1>help add-listener
NAMES
    add-listener
        Adds a new server listener with the given properties.
ALIAS
    addlistener, createlistener
USAGE
    add-listener -server <server-name>
                 -name <listener-name>
                 [-addr <address>]
                 -port <port>
                 [-selectors <selectors>]
                 [-dual]
                 [-backlog <backlog>]
                 [-timeout <read-timeout>]
                 [-keepaliveTimeout <keepalive-timeout>]
                 [-rt,--reservedthreads <reserved-threads>]
                 [-f,--forceLock]
...(생략)

[MASTER]jeus_domain.server1>add-listener -server server2 -name testListener -port 8777
Executed successfully, but some configurations were not applied dynamically. It might be
necessary to restart the server.
Check the result using 'list-server-listeners -server server2 -name testListener.
```

다른 리스너에서 사용하는 Port와 중복되지 않도록 설정한다.

2. **add-http-listener** 명령어를 사용하여 Web Connection을 추가한다.

```
[MASTER]jeus_domain.server1>help add-http-listener
NAMES
    add-http-listener
        Add HTTP listener.
ALIAS
    addhttpl
USAGE
    add-http-listener [-cluster <cluster-name> | -server <server-name>]
```

```
                    [-f,--forceLock]
                    -name <web-connection-name>
                    -tmin <minimum-thread-num>
                    [-tmax <maximum-thread-num>]
                    [-tidle <max-idle-time>]
                    [-qs <max-queue-size>]
                    -slref <server-listener-ref-name>
                    [-http2]
...(생략)

[MASTER]jeus_domain.server1>add-http-listener -server server2 -name testHttpListener -tmin 10
-tmax 20 -slref testListener
Successfully changed only the XML.
Restart the server to apply the changes.
For detailed web connection information, use the 'show-web-engine-configuration -cn' command.
```

3. **show-web-engine-configuration** 명령어를 통해 등록된 정보를 확인한다.

4.4. 데이터소스 추가

데이터소스(Datasource)는 데이터베이스와 Jakarta EE 프로그램을 연결하기 위해 사용된다.

예제에서는 JEUS에 기본적으로 포함되어 있는 Apache Derby 데이터베이스를 사용한다. Apache Derby는 JEUS_HOME\derby에 포함되어 있다. 만약 Derby가 실행되어 있지 않다면 다음과 같이 실행한다.

UNIX/Linux의 명령 프롬프트에서 다음과 같이 실행한다.

```
JEUS_HOME\bin> startderby
```

추후에 Derby를 종료하기 위해서는 다음과 같이 실행한다.

```
JEUS_HOME\bin> stopderby
```

Derby를 JEUS에서 사용하려면 Derby의 JDBC 드라이버 파일인 derbyclient.jar가 JEUS_HOME\lib\datasource에 위치해야 한다(기본적으로 포함되어 있다). Derby에 대한 자세한 내용은 http://db.apache.org/derby/를 참고한다.

예제에서는 sample이라는 데이터베이스를 jdbc/sample이라는 데이터소스 이름으로 사용한다.

다음은 콘솔 툴을 사용하여 데이터소스를 추가하는 방법에 대한 설명이다.

1. jeusadmin으로 JEUS에 접속한다.

```
jeusadmin –u jeus –p <password>
```

2. 데이터소스를 MASTER에 추가한다.

```
[MASTER]jeus9.server1>add-data-source -id datasource1 -en jdbc/sample -dscn
org.apache.derby.jdbc.ClientConnectionPoolDataSource -dst ConnectionPoolDataSource -vendor others
-sn localhost -pn 1527 -dn sample -user app -pw app -prop
"ConnectionAttributes:java.lang.String=;create=true"
Successfully performed the ADD operation for data source [datasource1] to domain.
Check the results using "add-data-source".
```

3. 데이터소스를 MS(server2)에 추가한다.

```
[MASTER]jeus9.server1>add-data-sources-to-server -server server2 -ids datasource1
Successfully performed the ADD operation for data sources to the server [server2].
Check the results using "add-data-sources-to-server -server server2".
```

4.5. 서버 기동 및 종료

JEUS에서 MS는 JEUS_HOME/bin에 있는 startManagedServer 스크립트로 기동할 수 있고, stopServer 스크립트로 종료할 수 있다.

1. **startManagedServer** 스크립트를 사용하여 추가한 MS 기동한다.

```
[was@localhost bin]$ startManagedServer -u administrator -p <password> -server server2
+++************************************************************
  - JEUS Home         : /home/jeus
- Added Java Option : -Djeus.io.buffer.size-per-pool=81920 -Djeus.cdi.enabled=false
-Djeus.jms.server.manager.produce-wait-strategy-type=blocking
-Djeus.servlet.sortWebinfLibraries=name_asc
************************************************************+++

=============== JEUS LICENSE INFORMATION ================
== VERSION : JEUS 9 Fix#0 (9.0.0.0-b15)
== EDITION: Enterprise (Trial License)
== NOTICE: This license restricts the number of allowed clients.
== Max. Number of Clients: 5
==========================================================
[2023.04.25 18:00:05][2] [launcher-1] [SERVER-0201] Successfully connected to the JEUS Master
Server(localhost:9736).
... 중략
[2024.09.25 18:00:10][0] [launcher-1] [Launcher-0040] Successfully started the server[server2].
The server state is now RUNNING.
```

2. **stopServer** 스크립트를 사용하여 기동한 MS를 종료한다.

```
[was@localhost bin]$ stopServer -u administrator -p <password> -server server2
Attempting to connect to 127.0.0.1:9736.
The connection has been established to JEUS Master Server [server1] in the domain [jeus_domain].
Stop server message to server [server2] was successfully sent.
```

5. WebTier 사용하기

본 장에서는 예제를 통해서 Servlet, JSP, JSTL, JSF 애플리케이션의 배치와 WAR(Web Application ARchive) 모듈의 패키징과 배치에 대해서 설명한다.

5.1. 예제

본 절에서는 웹 애플리케이션의 간단한 예제 코드를 작성하고, 해당 소스의 컴파일과 배치 과정을 설명한다.

자세한 내용은 "JEUS Server 안내서", "JEUS Web Engine 안내서", "JEUS Web Service 안내서"를 참고한다.

다음의 HelloWorldServlet.java는 웹 브라우저에 간단히 "Hello World!"라는 메시지를 출력하는 예제 서블릿이다.

WebTier 예제 : <HelloWorldServlet.java>

```
import java.io.*;

import jakarta.servlet.*;
import jakarta.servlet.http.*;

public class HelloWorldServlet extends HttpServlet
{
   public void doGet(HttpServletRequest req, HttpServletResponse res)
               throws IOException, ServletException
   {
      res.setContentType("text/html");

      PrintWriter out = res.getWriter();
      out.println("<HTML>");
      out.println("<HEAD>");
      out.println("<TITLE>Hello World Sample</TITLE>");
      out.println("</HEAD>");
      out.println("<BODY>");
      out.println("<CENTER><H1>Hello World!</H1></CENTER>");
      out.println("</BODY>");
      out.println("</HTML>");
      out.close();
   }
}
```

예제는 다음 위치의 디렉터리에서 찾을 수 있다.

```
JEUS_HOME/samples/getting_started/helloservlet/src/java
```

다음의 snoop.jsp는 요청을 받았을 때 Request에 대한 몇 가지 정보를 보여주는 snoop.jsp라는 샘플 JSP 프로그램이다.

WebTier 예제 : <snoop.jsp>

```
<html>

<body bgcolor="white">

<h2> Request Information </h2>
<font size="4">
JSP Request Method: <%= request.getMethod() %>
<br>
Request URI: <%= request.getRequestURI() %>
<br>
Request Protocol: <%= request.getProtocol() %>
<br>
Servlet path: <%= request.getServletPath() %>
<br>
Path info: <%= request.getPathInfo() %>
<br>
Path translated: <%= request.getPathTranslated() %>
<br>
Query string: <%= request.getQueryString() %>
<br>
Content length: <%= request.getContentLength() %>
<br>
Content type: <%= request.getContentType() %>
<br>
Server name: <%= request.getServerName() %>
<br>
Server port: <%= request.getServerPort() %>
<br>
Remote user: <%= request.getRemoteUser() %>
<br>
Remote address: <%= request.getRemoteAddr() %>
<br>
Remote host: <%= request.getRemoteHost() %>
<br>
Authorization scheme: <%= request.getAuthType() %>
<hr>
The browser you are using is <%= request.getHeader("User-Agent") %>
<hr>
</font>
</body>
</html>
```

예제는 다음 위치의 디렉터리에서 찾을 수 있다.

```
JEUS_HOME/samples/getting_started/helloservlet/web
```

다음은 snoop.jsp와 똑같은 일을 하지만 JSTL과 JSF를 사용한 snoop_jstl.jsp라는 샘플 JSP 프로그램이다.

WebTier 예제 : <snoop_jstl.jsp>

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<html>
```

```
<body>
<h2> Request Information </h2>
<font size="4">
    <c:set var="req" value="${pageContext.request}"/>
    JSP Request Method: <c:out value="${req.method}"/>
    <br/>
    Request Protocol: <c:out value="${req.protocol}"/>
    <br/>
    Servlet path: <c:out value="${req.servletPath}"/>
    <br/>
    Path info: <c:out value="${req.pathInfo}"/>
    <br/>
    Path translated: <c:out value="${req.pathTranslated}"/>
    <br/>
    Query string: <c:out value="${req.queryString}"/>
    <br/>
    Content length: <c:out value="${req.contentLength}"/>
    <br/>
    Content type: <c:out value="${req.contentType}"/>
    <br/>
    Server name: <c:out value="${req.serverName}"/>
    <br/>
    Server port: <c:out value="${req.serverPort}"/>
    <br/>
    Remote user: <c:out value="${req.remoteUser}"/>
    <br/>
    Remote address: <c:out value="${req.remoteAddr}"/>
    <br/>
    Remote host: <c:out value="${req.remoteHost}"/>
    <br/>
    Authorization scheme: <c:out value="${req.authType}"/>
    <hr/>
    <f:view>
    The browser you are using is <h:outputText value=
    "#{header['User-Agent']}"/>
    </f:view>
    <hr/>
</font>
</body>
</html>
```

이 예제는 JSP이므로 사용자가 컴파일할 필요없이 서블릿 엔진이 자동으로 컴파일한다.

5.2. 컴파일

작성된 예제는 jant를 이용하여 다음과 같이 빌드할 수 있다.

```
%JEUS_HOME%/samples/getting_started/helloservlet>jant build
```

빌드가 정상적으로 완료되면 dist 폴더 아래에 hello-servlet.war 애플리케이션 WAR 파일이 생성된다.

5.3. Deploy

패키징된 WAR 모듈은 콘솔을 사용해서 deploy한다.

JEUS에서는 애플리케이션의 배포 과정인 install과 deploy 과정에 대한 설명이다.

Deploy에 대한 자세한 내용은 자세한 내용은 "JEUS Application & Deployment 안내서"를 참고한다.

다음은 이전에 사용한 방법과 동일하게 WAR 모듈을 deploy하는 과정에 대한 설명이다.

콘솔 툴 사용

콘솔 툴(jeusadmin)을 사용하여 웹 모듈을 deploy하는 방법은 다음과 같다.

1. jeusadmin으로 JEUS에 접속한다.

```
jeusadmin –u jeus –p <password>
```

2. 애플리케이션을 다음과 같이 MASTER에 install한다.

```
[MASTER]domain1.adminServer>install-application -id helloworld
C:\TmaxSoft\JEUS\samples\getting_started\helloservlet\dist\hello-servlet.war
Successfully installed application[helloworld].
```

3. 애플리케이션을 다음과 같이 MS(server3)에 deploy한다.

```
[MASTER]domain1.adminServer>deploy helloworld -servers server3
deploy the application for the application [helloworld] succeeded.
```

4. 모듈이 정상적으로 deploy되었는지 확인한다.

5.4. 실행 및 결과

본 절에서는 deploy된 JSP와 서블릿을 사용하는 방법을 설명한다.

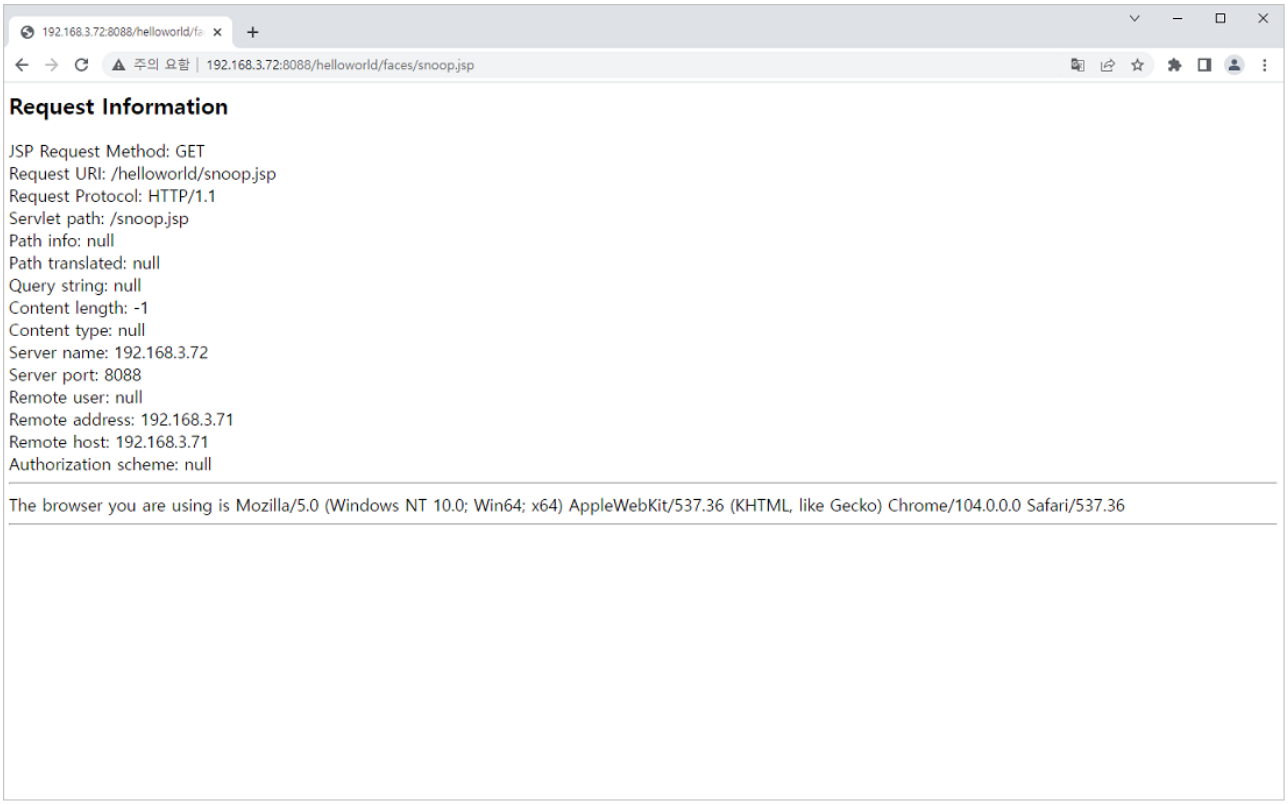
Deploy된 JSP 사용 방법

deploy된 JSP를 사용하는 방법은 다음과 같다.

- snoop.jsp 페이지 호출

snoop.jsp 페이지를 호출하려면 다음의 주소를 웹 브라우저 주소 창을 통해 호출한다(JSP의 경우 최초로 호출할 때 서블릿 엔진이 자동으로 컴파일을 수행하므로 약간 늦게 실행된다).

```
http://localhost:8088/helloworld/faces/snoop.jsp
```



WAR 모듈 JSP 호출

- snoop_jstl.jsp 페이지 호출

  snoop_jstl.jsp 페이지를 호출하려면 다음 주소를 주소 창을 통해 호출한다. 화면 결과는 snoop.jsp의 호출 결과와 동일하다.

  ```
  http://localhost:8088/helloworld/snoop_jstl.jsp
  ```

Deploy된 서블릿 사용 방법

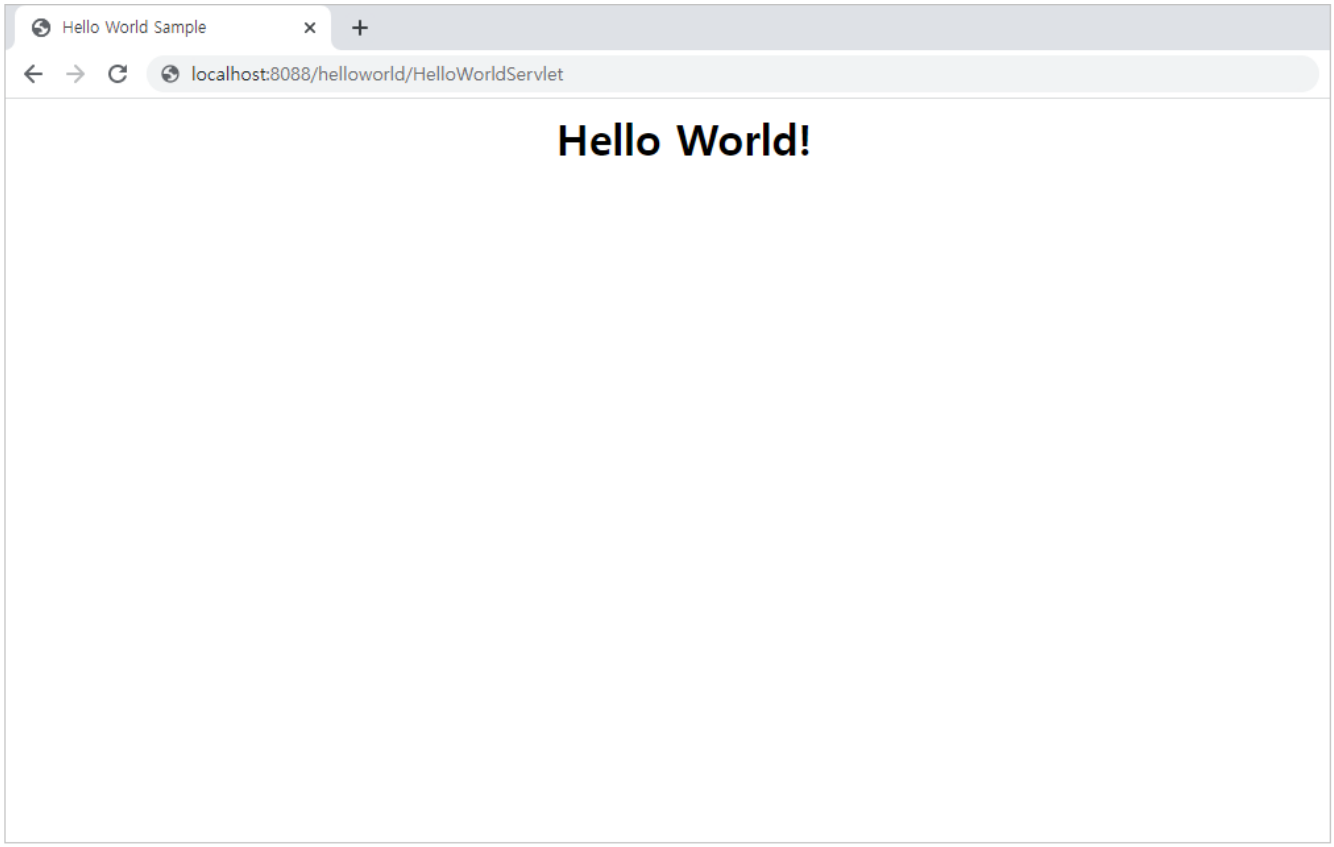
'helloworld' 서블릿을 호출하기 위해서는 브라우저의 주소 창에 다음과 같이 입력한다.

```
http://localhost:8088/helloworld/HelloWorldServlet
```

항목	설명
http	JEUS에 접속할 때 사용하는 HTTP 프로토콜을 의미한다.
localhost	서비스를 제공하는 서버가 브라우저와 동일한 자신의 주소에 있음을 의미한다.
8088	MS 내에 생성한 HTTP 리스너의 포트 번호이다.

항목	설명
helloworld	웹 애플리케이션의 컨텍스트용 Request path이다. 이 Request path는 jeus-web-dd.xml의 <context-path> element에 설정되며, 이 값을 지정하지 않을 경우 WAR 모듈 파일의 이름과 동일하다.
HelloWorldServlet	서블릿에 정의된 URL 패턴이다.

서블릿 엔진이 정상적으로 기동된 상태이고, Hello World 서블릿이 정상적으로 deploy되었다면 다음과 같은 화면이 나타난다.



WAR 모듈 서블릿 호출

6. EJB 사용하기

본 장에서는 예제를 이용해서 Stateless Session Bean과 Java Persistence API를 이용하여 Entity를 개발하고 deploy하는 과정에 대해 설명한다.

6.1. Session Bean 예제

Session Bean은 기본적으로 Business Interface와 Bean Class로 구성된다.

6.1.1. 예제

EJB 예제

다음의 예제 코드는 helloejb를 출력하는 Business 메소드를 가진 간단한 Stateless Session Bean 예제이다.

- Business Interface

  Stateless Session Bean 예제 : <Hello.java>

```
package helloejb;

import jakarta.ejb.Remote;

@Remote
public interface Hello {
    String sayHello();
}
```

- Bean Class

  Stateless Session Bean 예제 : <HelloBean.java>

```
package helloejb;

import jakarta.ejb.Stateless;

@Stateless(mappedName="helloejb.Hello")
public class HelloBean implements Hello {

    public String sayHello() {
        return "Hello EJB!";
    }
}
```

예제 코드는 다음의 디렉터리에서 찾아볼 수 있다.

```
JEUS_HOME/samples/getting_started/helloejb/helloejb-ejb/src/java/helloejb
```

서블릿 클라이언트 예제

helloejb를 호출하는 서블릿 클라이언트를 다음과 같이 구현한다.

서블릿 클라이언트 예제 : <HelloClient.java>

```
package helloejb;

import java.io.*;
import jakarta.ejb.EJB;

import jakarta.servlet.*;
import jakarta.servlet.http.*;

public class HelloClient extends HttpServlet {
    @EJB(mappedName="helloejb.Hello")
    private Hello hello;

    protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        try {
            // Call session bean business method.
            String msg = hello.sayHello();

            response.setContentType("text/html");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>HelloClient</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<center><h1>" + msg + "</h1></center>");
            out.println("</body>");
            out.println("</html>");
            out.close();
        } catch(Exception ex){
            response.setContentType("text/plain");
            ex.printStackTrace(out);
        }
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

예제 코드는 다음 위치의 디렉터리에서 찾아볼 수 있다.

```
JEUS_HOME/samples/getting_started/helloejb/helloejb-war/src/java/helloejb
```

6.1.2. 컴파일

작성된 예제는 Ant를 이용하여 다음과 같이 빌드할 수 있다.

```
%JEUS_HOME%/samples/getting_started/helloejb>jant build
```

빌드가 정상적으로 완료되면 %JEUS_HOME%/samples/getting_started/helloejb/dist/helloejb.ear 애플리케이션 EAR 파일이 생성된다.

6.1.3. Deploy

패키징한 EJB 애플리케이션을 콘솔 툴로 deploy할 수 있다.

콘솔 툴 사용

콘솔 툴을 사용하여 deploy하는 과정은 다음과 같다.

1. 위에서 작성한 helloejb.ear 파일을 임의의 경로에 위치시킨다.
2. jeusadmin으로 JEUS에 접속한다.

   ```
   jeusadmin –u jeus –p <password>
   ```

3. 다음과 같이 실행하여 애플리케이션을 MASTER에 install한다.

   ```
   [MASTER]domain1.adminServer>install-application -id helloejb
   C:\TmaxSoft\JEUS\samples\getting_started\helloejb\dist\helloejb.ear
   Successfully installed application[helloejb].
   ```

4. 다음과 같이 실행하여 애플리케이션을 MS(server1)에 deploy한다.

   ```
   [MASTER]domain1.adminServer>deploy helloejb -servers server1
   Succeeded to deploy the application : helloejb
   ```

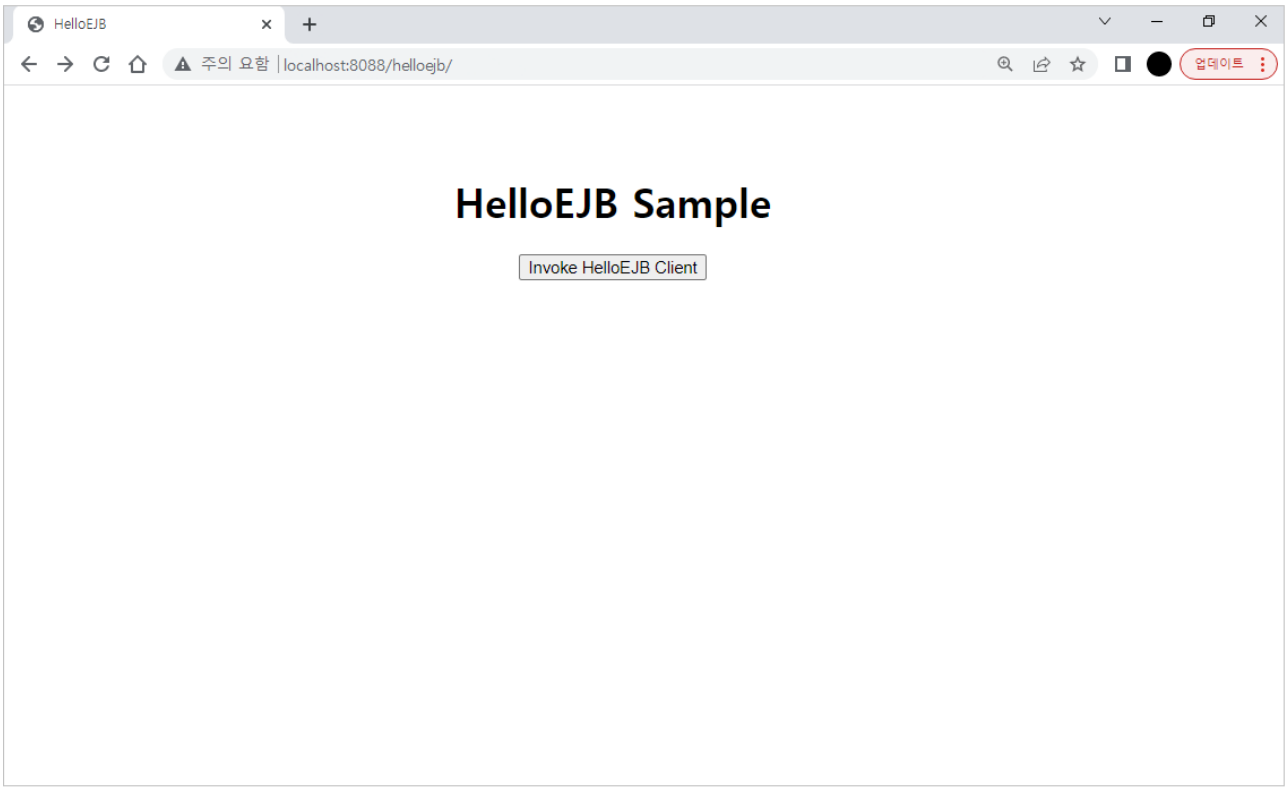
6.1.4. 실행 및 결과

deploy가 완료되면 작성된 화면를 실행해서 결과를 확인한다.

다음은 웹 클라이언트를 실행하는 과정에 대한 설명이다.

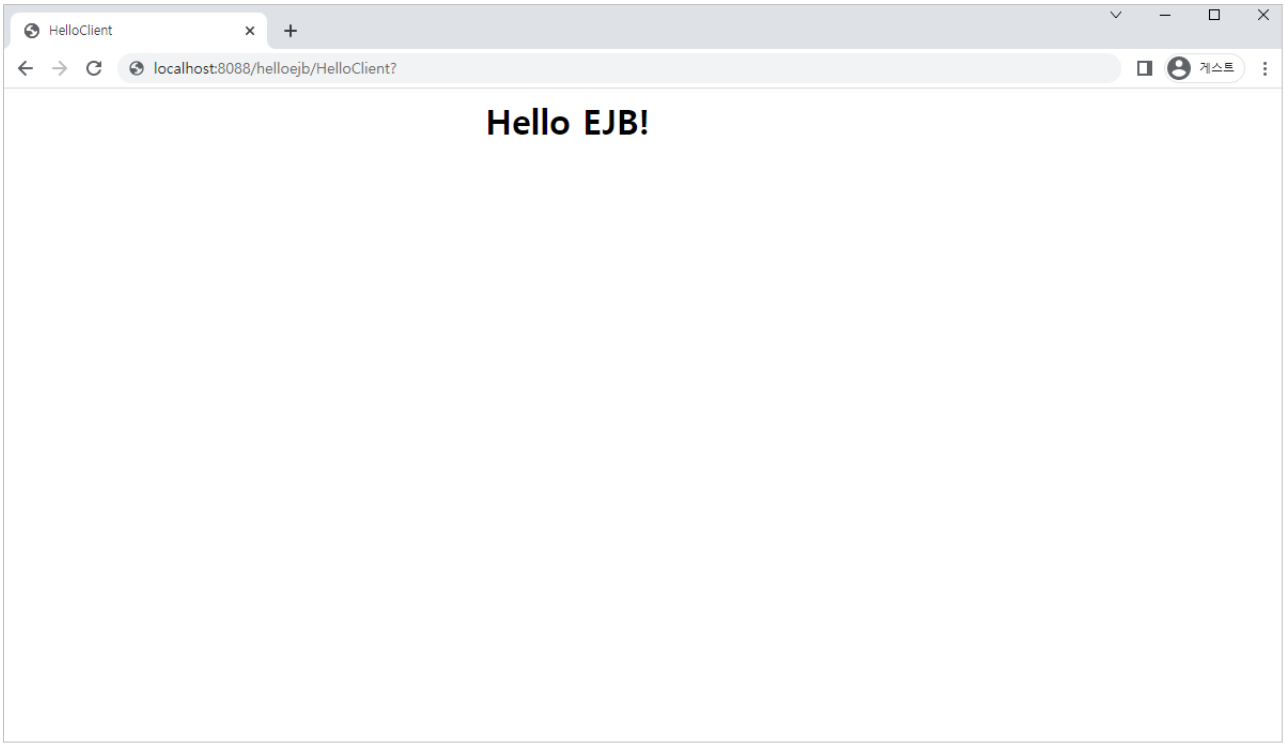
1. 웹 브라우저를 통해 다음 URL로 접속하면 HelloEJB 웹 클라이언트로 접속할 수 있다.

```
http://localhost:8088/helloejb/
```



HelloEJB 클라이언트 화면

2. 화면의 **[Invoke HelloEJB Client]** 버튼을 클릭하면 EJB를 호출하는 HelloClient 서블릿을 실행시키며 다음과 같은 결과를 확인할 수 있다.



HelloEJB 서블릿 클라이언트 수행 결과

6.2. Java Persistence API 예제

본 절에서는 Java Persistence API를 통해 Entity를 작성하고 컴파일하여 deploy하는 과정을 설명한다.

6.2.1. 예제

EJB 예제

예제는 Product Entity와 이를 다루는 EJB인 ProductManager Session Bean으로 구성되어 있다.

- Entity

Java Persistence API 예제 : <Product.java>

```
package hellojpa;

import java.io.Serializable;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQuery;

@Entity
@NamedQuery(name="findAllProducts", query="SELECT p FROM Product p")
public class Product implements Serializable {
    @Id
    private String productId;
    private double price;
    private String description;

    public Product() {
    }

    public Product(String productId, double price,
        String description){
        this.productId = productId;
        this.price = price;
        this.description = description;
    }

    public String getProductId() {
        return productId;
    }

    public void setProductId(String id) {
        this.productId = id;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
```

```
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String toString() {
        return "Product[productId=" + productId + ", price=" +
            price + ", description=" + description + "]";
    }
}
```

- Business Interface

Java Persistence API 예제 : <ProductManager.java>

```
package hellojpa;

import java.util.Collection;
import jakarta.ejb.Local;

@Local
public interface ProductManager {
    Product createProduct(String productId, double price, String desc);

    Product getProduct(String productId);

    Collection findAllProducts();

    Collection findProductsByDescription(String desc);

    Collection findProductsInRange(double low, double high);

    void updateProduct(Product product);

    void removeProduct(Product product);

    void removeAllProducts();
}
```

- Bean Class

Java Persistence API 예제 : <ProductManagerBean.java>

```
package hellojpa;

import java.util.Collection;
import jakarta.ejb.Stateless;
import jakarta.persistence.EntityManager;
import jakarta.persistence.PersistenceContext;
import jakarta.persistence.Query;

@Stateless(mappedName="hellojpa.ProductManager")
public class ProductManagerBean implements ProductManager {
```

```
    @PersistenceContext
    private EntityManager em;

    public ProductManagerBean() {
    }

    public Product createProduct(String productId, double price, String desc){
        Product product = new Product(productId, price, desc);
        em.persist(product);
        return product;
    }

    public Product getProduct(String productId){
        return (Product)em.find(Product.class, productId);
    }

    public Collection findAllProducts() {
        return em.createNamedQuery("findAllProducts").getResultList();
    }

    public Collection findProductsByDescription(String desc){
        Query query = em.createQuery("SELECT p FROM Product p WHERE
            p.description=:desc");
        query.setParameter("desc", desc);
        return query.getResultList();
    }

    public Collection findProductsInRange(double low, double high){
        Query query = em.createQuery("SELECT p FROM Product p WHERE
            p.price between :low and :high");
        query.setParameter("low", low).setParameter("high", high);
        return query.getResultList();
    }

    public void updateProduct(Product product){
        Product managed = em.merge(product);
        em.flush();
    }

    public void removeProduct(Product product){
        Product managed = em.merge(product);
        em.remove(managed);
    }

    public void removeAllProducts(){
        em.createQuery("DELETE FROM Product p").executeUpdate();
    }
}
```

예제 코드는 다음의 디렉터리에서 찾아볼 수 있다.

```
JEUS_HOME/samples/getting_started/hellojpa/hellojpa-ejb/src/java/hellojpa
```

서블릿 클라이언트 예제

ProductManager EJB를 사용하여 데이터베이스에 데이터를 저장하고 데이터를 다루는 서블릿 클라이언트를 다음과 같이 구현한다.

서블릿 클라이언트 예제 : <ProductManagerClient.java>

```
package hellojpa;

import java.io.*;
import java.util.Collection;
import jakarta.ejb.EJB;

import jakarta.servlet.*;
import jakarta.servlet.http.*;

public class ProductManagerClient extends HttpServlet {
    @EJB
    private ProductManager productManager;

    protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        out.println("SERVLET CLIENT CONSOLE OUTPUT:\n");

        productManager.removeAllProducts();
        out.println("Cleaned up existing products.\n");

        out.println("Creating products...");
        Product p1 = productManager.createProduct("1", 10.00, "Ceramic Dog");
        Product p2 = productManager.createProduct("2", 13.00, "Wooden Duck");
        Product p3 = productManager.createProduct("3", 19.00, "Ivory Cat");
        Product p4 = productManager.createProduct("4", 33.00, "Ivory Cat");
        Product p5 = productManager.createProduct("5", 22.00, "Chrome Fish");

        Collection products;

        out.println("Created products:");
        products = productManager.findAllProducts();
        for(Object product : products){
             out.println(product);
        }
        out.println();

        out.println("Find product with productId 1:");
        Product pp1 = productManager.getProduct("1");
        out.println("Found = " + pp1.getDescription() + " $" + pp1.getPrice());

        out.println("Update the price of this product to 12.00");
        pp1.setPrice(12.00);
        productManager.updateProduct(pp1);

        Product pp2 = productManager.getProduct("1");
        out.println("Product " + pp2.getDescription() + " is now $" + pp2.getPrice());
        out.println();

        out.println("Find products with description:");
```

```
        products = productManager.findProductsByDescription("Ivory Cat");
        for(Object product : products){
            out.println(product);
        }
        out.println();

        out.println("Find products with price range between 10.00 and 20.00");
        products = productManager.findProductsInRange(10.00, 20.00);
        for(Object product : products){
            out.println(product);
        }
        out.println();

        out.println("Removed all products.");
        productManager.removeProduct(p1);
        productManager.removeProduct(p2);
        productManager.removeProduct(p3);
        productManager.removeProduct(p4);
        productManager.removeProduct(p5);

        out.close();
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

예제코드는 다음의 디렉터리에서 찾아볼 수 있다.

```
JEUS_HOME/samples/getting_started/hellojpa/hellojpa-war/src/java/hellojpa
```

6.2.2. 컴파일

jant를 이용하여 예제 코드를 다음과 같이 빌드할 수 있다. 빌드와 데이터베이스 설정이 완료되었다면 패키징된 모듈을 deploy할 준비가 완료되었다.

1. 예제가 작성된 경로에서 다음과 같이 **ant build** 명령어를 실행한다.

   ```
   C:\TmaxSoft\JEUS\samples\getting_started\hellojpa>jant build
   ```

2. 빌드가 정상적으로 완료되면 dist\hellojpa.ear 애플리케이션 EAR 파일이 생성된다.

이번 예제는 데이터베이스를 필요로 하므로 Derby를 시작시키도록 한다. 또한 데이터베이스가 jdbc/sample 데이터소스로 설정되어 있어야 한다. 자세한 내용은 시스템 설정을 참고한다.

Derby가 시작되었다면 다음과 같이 데이터베이스 테이블을 생성한다. 본 예제에서는 "sample"이라는 데이터베이스를 사용할 것이다.

```
CREATE TABLE PRODUCT (PRODUCTID VARCHAR(255) NOT NULL, PRICE FLOAT,
    DESCRIPTION VARCHAR(255), PRIMARY KEY (PRODUCTID));
```

3. 다음과 같이 **ant setup** 명령어를 실행해서 데이터베이스 테이블을 생성한다.

```
C:\TmaxSoft\JEUS\samples\getting_started\hellojpa>jant setup
```

6.2.3. Deploy

패키징한 EJB 애플리케이션을 콘솔 툴로 deploy할 수 있다.

콘솔 툴 사용

콘솔 툴을 사용하여 deploy하는 방법은 다음과 같다.

1. jeusadmin으로 JEUS에 접속한다.

```
jeusadmin –u jeus –p <password>
```

2. 애플리케이션을 MASTER에 install한다.

```
[MASTER]domain1.adminServer>install-application -id hellojpa
C:\TmaxSoft\JEUS\samples\getting_started\hellojpa\dist\hellojpa.ear
Successfully installed application[hellojpa].
```

3. 애플리케이션을 MS(server1)에 deploy한다.

```
[MASTER]domain1.adminServer>deploy hellojpa -servers server1
Succeeded to deploy the application : hellojpa
```

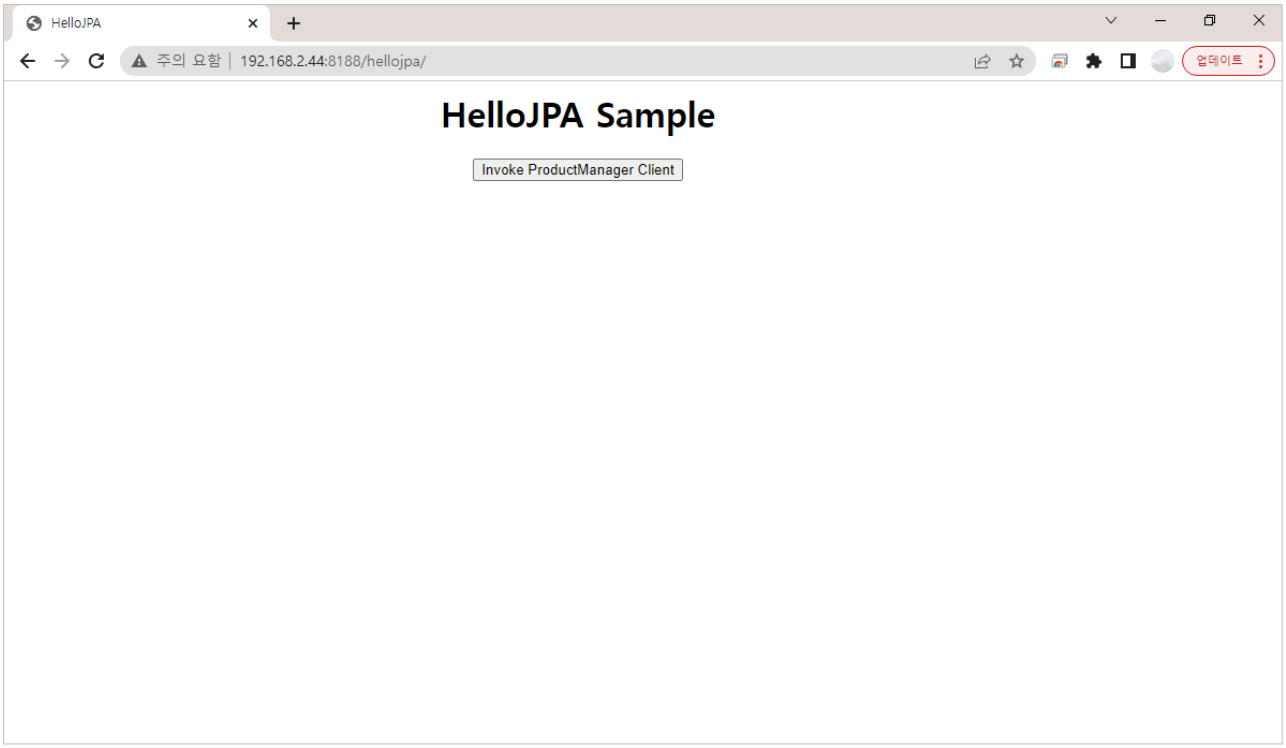
4. 모듈이 정상적으로 deploy되었는지 확인한다.

6.2.4. 실행 및 결과

배포가 완료되면 다음의 과정으로 작성된 화면을 실행해서 결과를 확인한다.

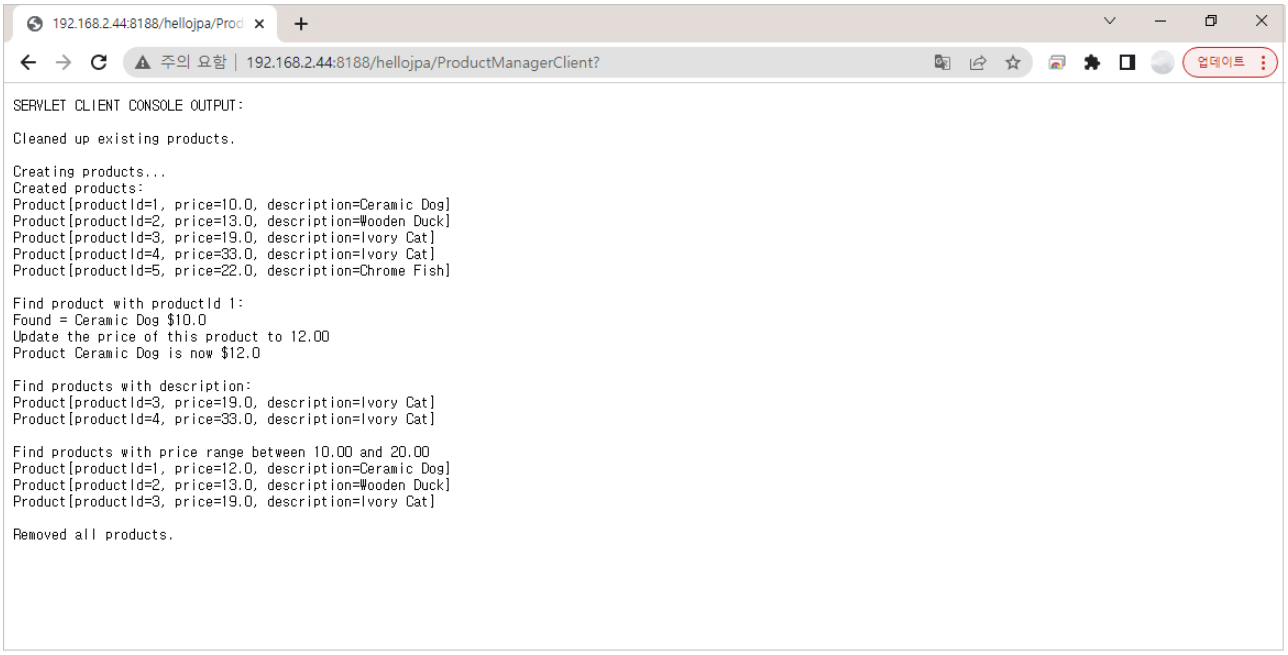
1. 웹 브라우저를 통해 다음 URL로 접속하면 HelloJPA 웹 클라이언트로 접속할 수 있다.

```
http://localhost:8088/hellojpa/
```



HelloJPA 클라이언트 화면

2. 화면의 **[Invoke ProductManager Client]** 버튼을 클릭하면 EJB에 요청을 내리는 서블릿 클라이언트를 수행하고 다음과 같이 결과가 표시된다.



HelloJPA 서블릿 클라이언트 수행 결과

7. 웹 서비스 사용하기

본 장에서는 JAX-WS 2.2를 활용한 웹 서비스의 생성과 클라이언트의 작성, 그리고 웹 서비스를 호출하는 내용을 설명한다.

7.1. 웹 서비스 생성

JEUS 9에서는 Jakarta EE 8 방식의 웹 서비스를 지원하며 JAX-WS 2.2는 Java EE 6 웹 서비스의 핵심이다.

JAX-WS는 기존 JAX-RPC를 대체하는 수단으로 설계되었다. 새로운 모습으로 등장한 JAX-WS의 배경으로는 JAXB 2.0의 등장에 있으며, JAXB 2.2부터는 모든 XML 스키마 타입을 완전히 지원하게 됨으로써 기존에 존재했던 Java 타입과 XML 타입 간의 매핑을 보다 명확하게 정의할 수 있게 되었다. 기존에 JAX-RPC 스펙에 존재하던 Java 타입과 XML 타입 간의 매핑에 대한 의존성을 제거할 수 있게한 원동력이 되었다.

또한 SOAP 1.2 메시지를 직접 다룰 수 있게 하는 SAAJ 1.3까지 포함하여, Java Web Services 2.0이라는 새로운 웹 서비스 모델이 등장하게 된다.

7.1.1. From Java 방식

From Java 방식의 웹 서비스 생성은 기본적으로 다음과 같은 절차를 따른다.

1. **웹 서비스 Annotation이 포함된 서비스 구현 Bean의 작성**

   서비스 구현 Bean을 작성할 경우에는 다음과 같은 몇 가지의 필수 제약 조건을 따르게 된다.

   - jakarta.jws.WebService Annotation을 포함시켜서 이 클래스가 서비스 구현 Bean임을 명시한다.
   - 웹 서비스 메소드의 인자와 리턴 타입은 JAXB 2.0의 Java와 XML 스키마 간의 매핑 정의와 호환되어야 한다.
   - 웹 서비스 메소드의 인자와 리턴 타입은 java.rmi.Remote 인터페이스를 직접 또는 간접적으로 구현해서는 안 된다.

   이와 같은 요소들 외에도, jakarta.jws 패키지에 정의되어 있는 여러 Annotation을 활용하면 메소드의 인자와 리턴 타입, 바인딩 방식 등의 커스터마이징이 가능하다. 간단한 예제 코드를 통해 실제 서비스 구현 Bean의 작성 방법을 설명한다.

   다음은 간단한 웹 서비스를 구현한 클래스에 대한 예제로 코드는 다음의 경로에서 찾아볼 수 있다.

   ```
   JEUS_HOME/samples/getting_started/webservices/from_java/src/java/fromjava/server
   ```

   웹 서비스 예제(From Java 방식) : <AddNumbersImpl.java>

   ```
   package fromjava.server;

   import jakarta.jws.WebService;

   @WebService
   public class AddNumbersImpl {
   ```

```
    public int addNumbers(int number1, int number2) {
        return number1 + number2;
    }
}
```

JAX-WS에서는 기존 Java 클래스의 웹 서비스로의 공개가 아주 용이하며, 위 <AddNumbersImpl.java> 에서 보는 바와 같이 @WebService Annotation을 추가함으로써 기존의 Java 클래스를 쉽게 웹 서비스로 변환하여 만들 수 있다.

2. **타 벤더 간에 이식 가능한 산출물(Portable Artifact)의 생성**

   서비스 구현 Bean을 작성하고 컴파일까지 수행했다면, JAX-WS 런타임에서 사용할 타 벤더 간에 이식 가능한 산출물의 생성 작업이 필요하다. 여기서 타 벤더 간에 이식 가능한 산출물은 JAX-WS 스펙을 준수하는 모든 벤더에서 사용할 수 있는 JAX-WS 툴을 통해 만들어낸 산출물임을 의미한다. Java의 파라미터를 실제 WSDL의 메시지로 정확하게 매핑하기 위한 정보 등을 담고 있는 Java 클래스와 WSDL 등이 여기에 포함된다.

   JEUS에서는 **wsgen**이라는 콘솔 스크립트를 제공하며, 이 스크립트는 JEUS_HOME/bin 디렉터리 하위에 존재한다.

   ```
   wsgen –cp <classpath> -d <destination_dir> fromjava.server.AddNumbersImpl
   ```

   위와 같이 명령어를 실행하면 지정한 경로에 이식 가능한 산출물이 생성됨을 확인할 수 있다. 위 스크립트를 실행할 때 –wsdl 옵션을 설정하면 WSDL까지 생성이 가능하나 JAX-WS에서는 웹 서비스 Endpoint에 WSDL을 포함하지 않아도 되므로 여기서 –wsdl 옵션을 설정하지 않도록 주의한다. 다음은 이 산출물들을 묶어서 서버에 배치하는 작업을 진행한다.

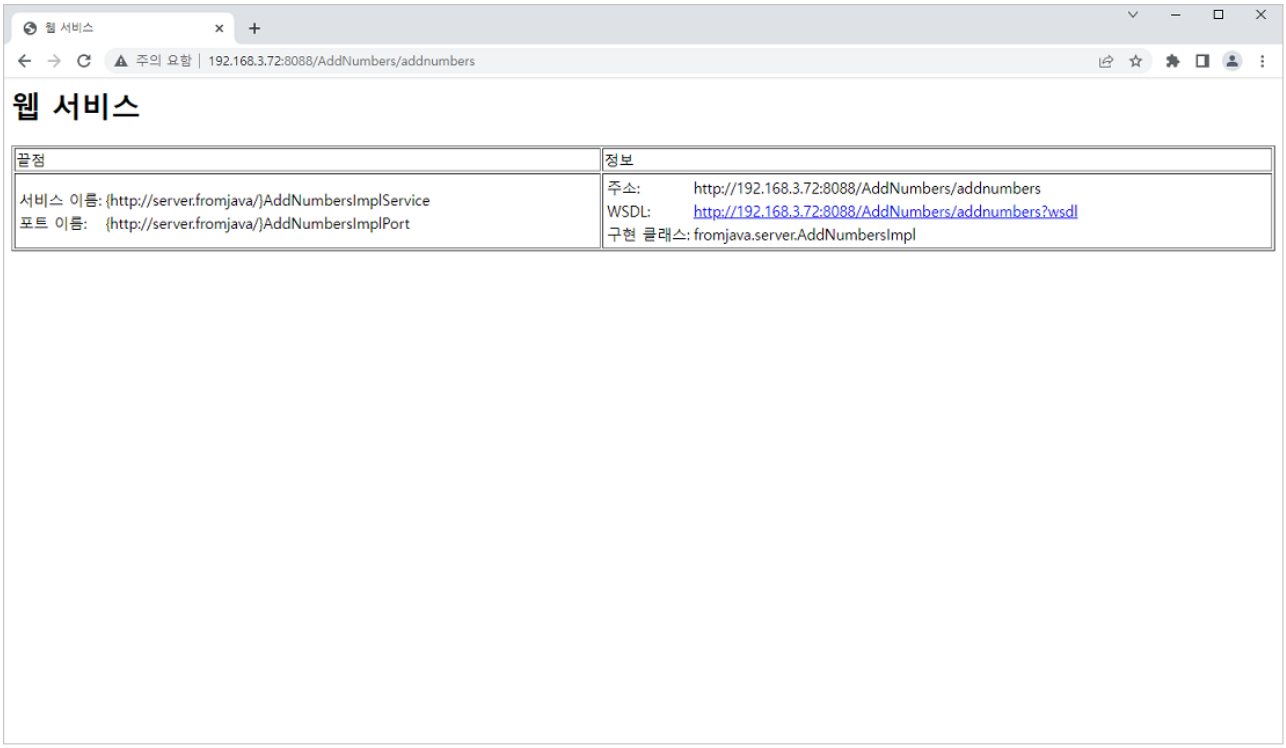
3. **웹 서비스 패키징과 배치**

   현재 작성하고 있는 웹 서비스를 패키징한다는 것은 서비스 구현 Bean과 서비스 구현 Bean이 참조하고 있는 Java 클래스와 부수적인 DD를 WAR 형식으로 묶는 것을 의미한다. 여기서는 이미 앞에서 작성했던 파일인 fromjava.server.AddNumbersImpl 클래스와 wsgen 스크립트를 통해 생성하였던 fromjava.server.jaxws.AddNumbers, fromjava.server.jaxws.AddNumbersResponse 클래스만을 포함한다. 이 클래스들은 WEB-INF/classes 하위에 포함시키면 된다. 패키징할 파일 이름을 AddNumbers.war로 하여 WAR 패키징을 한 다음, JEUS에 배치한다.

   실제 이 서비스를 호출할 수 있는 HTTP 주소는 다음과 같다.

   ```
   http://localhost:8088/AddNumbers/addnumbers
   ```

   해당 주소로 웹 브라우저에서 호출하면 다음과 같이 성공적으로 웹 서비스가 배치되어 있는 것을 확인할 수 있다.



성공적으로 배치된 From Java 방식의 웹 서비스

샘플 예제 쉽게 실행하기

JEUS에서는 웹 서비스 예제를 쉽게 실행해 볼 수 있는 방법을 제공한다.

JEUS_HOME/samples/getting_started/webservices/from_java/ 아래에서 다음과 같이 **jant** 명령어를 실행하면 서비스 패키징부터 클라이언트 실행까지 쉽게 할 수 있다.

```
%JEUS_HOME%/samples/getting_started/webservices/from_java> jant
```

빌드가 정상적으로 되면 웹 브라우저가 열리면서 결과를 확인할 수 있다.

7.1.2. From WSDL 방식

From Java 방식의 웹 서비스 방식이 이미 작성한 Java RPC 모델을 웹 서비스로 공개하는 것이 초점이라면 From WSDL 방식의 웹 서비스는 서로 간에 통신할 SOAP 메시지를 먼저 정의하고 WSDL를 통해 그 정보를 공유한 다음, 그 정의된 메시지 타입에 맞게 Java 클래스를 생성하는 것이 초점이라 할 수 있다.

일반적인 From WSDL 방식의 웹 서비스 생성은 다음과 같은 과정으로 진행된다.

1. **서비스 Endpoint 인터페이스(Service Endpoint Interface)의 생성**

   이 과정에서는 이미 공개되어 있는 WSDL을 기반으로 웹 서비스의 Java 인터페이스 파일과 Java 클래스 파일들을 생성한다. 이때 JEUS에서 제공하는 wsimport라는 콘솔 스크립트를 사용하고, 그 스크립트는 JEUS_HOME/bin이라는 위치에 존재한다.

   JEUS_HOME/samples/getting_started/webservices/from_wsdl 디렉터리에서 다음과 같이 실행한다.

```
wsimport -keep -p fromwsdl.server -d ./build/classes  ./web/WEB-INF/wsdl/AddNumbers.wsdl
```

위와 같이 명령어를 실행하면, 지정한 경로에 서비스 Endpoint 인터페이스와 서비스 정의 클래스를 포함한 여러 산출물들이 생성된다.

이 중 생성된 서비스 Endpoint 인터페이스는 다음과 같다. 이렇게 생성된 서비스 Endpoint 인터페이스는 JAX-WS 런타임에 사용될 정보들을 Annotation 형태로 포함하고 있다.

웹 서비스 예제(From WSDL 방식) : <AddNumbersImpl.java>

```
package fromwsdl.server;

@jakarta.jws.WebService(endpointInterface = "fromwsdl.server.AddNumbersPortType",
wsdlLocation = "WEB-INF/wsdl/AddNumbers.wsdl",
targetNamespace = "urn:AddNumbers", serviceName = "AddNumbersService",
portName = "AddNumbersPort")
public class AddNumbersImpl {

    public int addNumbers(int number1, int number2) {
        return number1 + number2;
    }
}
```

2. **서비스 Endpoint 인터페이스의 구현**

   서비스 Endpoint 인터페이스가 생성이 되었다면, 이 Endpoint 인터페이스를 구현하는 실제 비즈니스 로직을 가지고 있는 서비스 구현 Bean을 작성한다. 서비스 Endpoint 인터페이스를 구현한 서비스 구현 Bean을 작성할 경우에는 @jakarta.jws.WebService Annotation을 추가해야 하며, 이 Annotation은 서비스 Endpoint 인터페이스를 명시한 endpointInterface 멤버를 속성으로 가지고 있어야 한다.

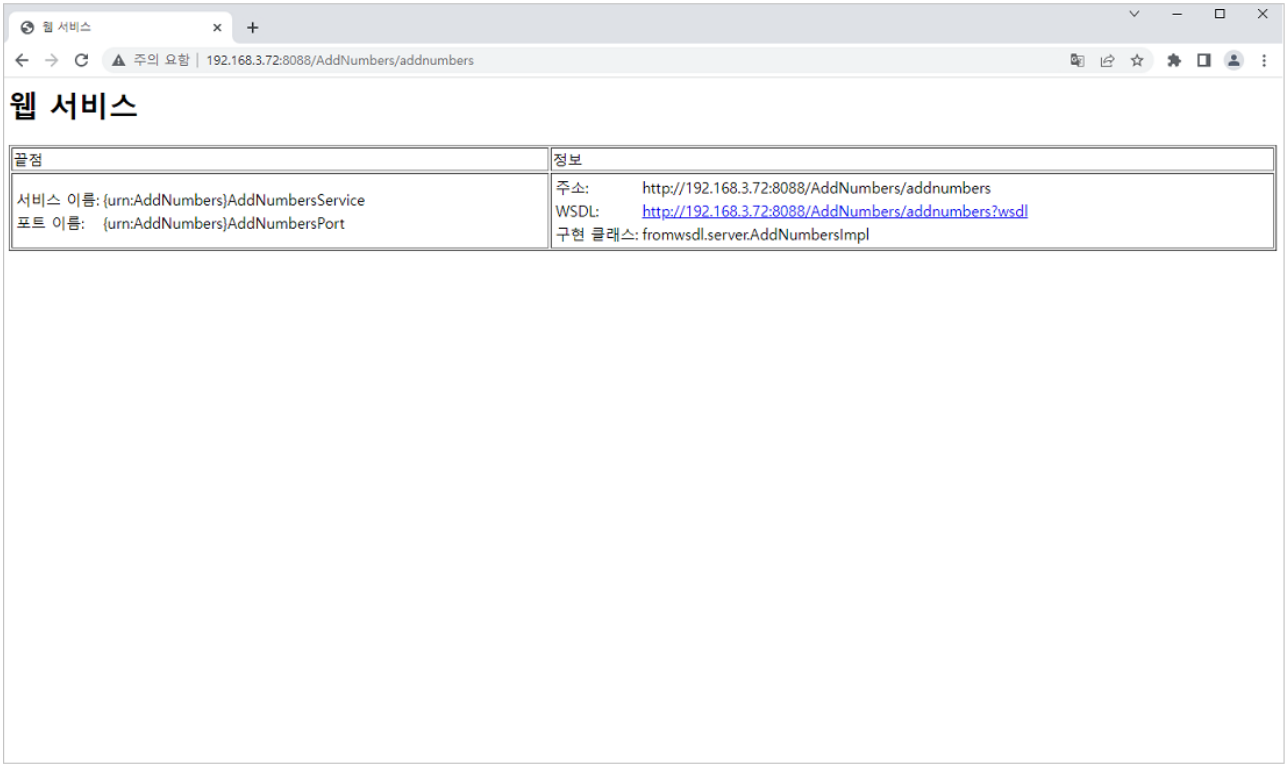
3. **웹 서비스 패키징과 배치**

   웹 서비스로 공개하기 위해 WAR 형태로 패키징하는 작업은 이전의 From Java 방식의 웹 서비스의 패키징 및 배치 작업과 유사하다.

   wsimport 스크립트를 통해 생성한 서비스 Endpoint 인터페이스를 포함한 몇 가지의 산출물들과 서비스 구현 Bean을 WEB-INF/classes에 위치시키고 WAR 형태로 패키징한다. 이때의 WAR 패키지의 이름을 AddNumbers.war라고 할 경우 이를 JEUS에 배치하게 되면 다음과 같은 주소로 접근할 수 있다. From Java 방식으로 이미 배치하였다면 컨텍스트 이름이 동일하므로 이를 제거한 후 다시 배치해야 한다.

   ```
   http://localhost:8088/AddNumbers/addnumbers
   ```

   실제 이 주소로 웹 브라우저에서 호출하면 다음과 같이 성공적으로 웹 서비스가 배치되어 있는 것을 확인할 수 있다.



성공적으로 배치된 From WSDL 방식의 웹 서비스

샘플 예제 쉽게 실행하기

JEUS에서는 웹서비스 예제를 쉽게 실행해 볼 수 있는 방법을 제공한다.

JEUS_HOME/samples/getting_started/webservices/from_wsdl/ 아래에서 다음과 같이 **jant** 명령어를 실행하면 서비스 패키징부터 클라이언트 실행까지 쉽게 할 수 있다.

```
%JEUS_HOME%/samples/getting_started/webservices/from_wsdl> jant
```

빌드가 정상적으로 되면 웹 브라우저가 열리면서 결과를 확인할 수 있다.

7.2. 웹 서비스 클라이언트 작성

웹 서비스에 접근하기 위해서는 웹 서비스 클라이언트를 작성해야 하는데 웹 서비스 클라이언트 프로그램의 구동 환경에 따라 Java SE 클라이언트와 Jakarta EE 8 클라이언트로 구분할 수 있다.

본 절에서는 일반 Java 프로그램과 동일하게 구동되는 Java SE 클라이언트에 대해서만 다루도록 하겠다.

7.2.1. Java SE 클라이언트의 작성

JAX-WS에서는 기본적으로 웹 서비스 Endpoint에 대응하는 동적 프록시를 내부적으로 생성하여 마치 이를 서비스 Endpoint 인터페이스의 구현체인 것처럼 사용할 수 있다. 따라서 클라이언트 프로그램 개발자는 생성된 프록시를 통해 서비스 Endpoint 인터페이스로 정의된 웹 서비스 메소드를 호출할 수 있다.

이와 같은 작업을 진행하기 위해서는 서비스 Endpoint 인터페이스와 같은 몇 가지의 산출물들이 필요한데, 이때 필요한 산출물들을 WSDL로부터 생성하기 위해서는 From WSDL 방식의 웹 서비스 생성에서와 마찬가지로 **wsimport** 툴을 사용해야 한다.

다음은 wsimport 사용 예로 JEUS_HOME/samples/getting_started/webservices/from_wsdl 디렉터리에서 실행한다.

```
wsimport -p fromwsdl.client -d ./build/classes http://localhost:8088/AddNumbers/addnumbers?wsdl
```

서비스 Endpoint 인터페이스와 서비스 클래스가 생성이 되었다면, 이를 사용하는 클라이언트 Java 프로그램을 작성한다.

다음은 웹 서비스 클라이언트 프로그래밍의 한 예를 보여준다. 이 부분은 기존 JAX-RPC에서 제공하던 웹 서비스 클라이언트 프로그래밍 예와 비슷하다. 기본적으로 서비스 인스턴스를 생성한 다음, 그 서비스 인스턴스로부터 서비스 Endpoint 인터페이스를 구현한 프록시 객체를 얻어오는 것이 주요 작업이다. 클라이언트 런타임은 내부적으로는 많이 달라졌지만, Java SE 클라이언트에서는 유사하다고 할 수 있다.

예제 코드는 JEUS_HOME/samples/getting_started/webservices/from_wsdl/src/java/fromwsdl/client에서 찾아볼 수 있다.

Java SE 클라이언트 : <AddNumbersClient.java>

```
package fromwsdl.client;

public class AddNumbersClient {

    public static void main(String[] args) {
        AddNumbersPortType port = new AddNumbersService().getAddNumbersPort();

        int number1 = 10;
        int number2 = 20;

        System.out.println("################################################");
        System.out.println("### JAX-WS Webservices examples - fromwsdl ###");
        System.out.println("################################################");
        System.out.println("Testing Java class webservices from WSDL...");
        int result = port.addNumbers(number1, number2);
        if (result == 30) {
            System.out.println("Success!");
        }
    }
}
```

위와 같이 클라이언트 프로그램을 작성한 후 실제 이를 컴파일하여 실행하면 다음과 같은 결과가 나타난다.

```
[java] ################################################
[java] ### JAX-WS Webservices examples - fromwsdl ###
[java] ################################################
[java] Testing Java class webservices from WSDL...
[java] Success!
```

Appendix A: IPv6 설정

본 부록에서는 IPv6 설정에 대해 설명한다.

A.1. 소개

IPv6 환경에서 JEUS의 설정은 다음과 같은 절차로 이루어진다.

1. jeus.properties, jeusadmin, startMasterServer, startManagedServer, stopServer, startderby, stopderby, mcastReceiver, mcastSender 스크립트 변경
2. domain.xml 변경
3. 동작 확인

설정하려는 서버의 hosts 파일에 loopback address는 '::1'로 설정되어 있어야 한다.

hosts 파일은 운영체제에 따라 다음의 경로에 위치한다.

- UNIX 계열

```
/etc
```

hosts 파일에 대한 예제는 다음과 같다.

hosts 파일

```
[jeusqa@ipv6linux /home/jeusqa]$ cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
#127.0.0.1              localhost
#::1    localhost6.localdomain6 localhost6
::1     localhost
```

A.2. IPv6 환경설정

IPv6 사용을 위한 환경설정에 대해 설명한다. IPv6 환경은 파일에 따라 설정 방법이 다르다.

JEUS_HOME/bin에 존재하는 파일의 변경

- 파일명 : startMasterServer, startManagedServer

  -classpath 가 작성되어 있는 부분에 -Djava.net.preferIPv6Addresses=true와 -Djava.net.preferIPv4Stack=false 내용을 추가한다.

  - 예제

```
-classpath .........                                                \
-Djava.net.preferIPv6Addresses=true                                 \
-Djava.net.preferIPv4Stack=false                                    \
```

- 파일명 : mcastReceiver, mcastSender
  - 기존

```
-Djava.net.preferIPv4Stack=true                                  \
```

  - 변경

```
-Djava.net.preferIPv6Addresses=true                                 \
-Djava.net.preferIPv4Stack=false                                    \
```

- 파일명 : jeusadmin, stopServer
  - 기존

```
"${JAVA_HOME}/bin/java" -classpath "${BOOTSTRAP_CLASSPATH}" ${TOOL_OPTION}      \
```

  - 변경

```
"${JAVA_HOME}/bin/java" -classpath "${BOOTSTRAP_CLASSPATH}" ${TOOL_OPTION}      \
-Djava.net.preferIPv6Addresses=true                                 \
-Djava.net.preferIPv4Stack=false                                    \
```

- 파일명 : jeus.properties
  - 기존

```
TOOL_OPTION="-Djeus.tm.not_use=true  -Djava.net.preferIPv4Stack=true"
```

  - 변경

```
TOOL_OPTION="-Djeus.tm.not_use=true -Djava.net.preferIPv6Addresses=true
-Djava.net.preferIPv4Stack=false"
```

- 파일명 : startderby, stopderby
  - 기존

```
-Dderby.system.home="${JEUS_HOME}/derby/databases"              \
```

◦ 변경

```
-Dderby.system.home="${JEUS_HOME}/derby/databases"              \
-Djava.net.preferIPv6Addresses=true                             \
-Djava.net.preferIPv4Stack=false                                \
```

domain.xml의 변경

JEUS_HOME/domains/DOMAIN_NAME/config/domain.xml 파일의 내용을 아래와 같이 변경 및 추가한다.

1. listen-address의 주소가 IPv4로 설정되어 있다면 IPv6로 변경한다.

```
<listeners>
   <base>BASE</base>
   <listener>
   <name>BASE</name>
   <listen-address>0:0:0:0:0:0:0:0</listen-address>
   <listen-port>9736</listen-port>
   <selectors>1</selectors>
   <use-dual-selector>false</use-dual-selector>
   <backlog>128</backlog>
   <select-timeout>120000</select-timeout>
   <read-timeout>30000</read-timeout>
   <reserved-thread-num>0</reserved-thread-num>
</listener>
```

2. heartbeat-address의 address가 IPv4로 설정되어 있다면 아래와 같이 IPv6로 변경하여야 하며, 설정이 없다면 아래의 내용을 추가한다.

```
<group-communication-info>
   <heartbeat-address>FF02:0:0:0:0:0:0:0</heartbeat-address>
   <heartbeat-port>3030</heartbeat-port>
   <use-virtual-multicast>false</use-virtual-multicast>
</group-communication-info>
```

3. 각 서버별로 vm-option에 java.net.preferIPv4Stack=false와 java.net.preferIPv6Addresses=true를 추가한다.

```
<jvm-config>
   <jvm-option>-Xmx1024m -XX:MaxPermSize=256m</jvm-option>
   <jvm-option>java.net.preferIPv4Stack=false</jvm-option>
   <jvm-option>java.net.preferIPv6Addresses=true</jvm-option>
</jvm-config>
```

Appendix B: domain-config-template.properties 설정

본 부록에서는 JEUS를 설치할 때 필요한 domain-config-template.properties 파일 설정에 대해 설명한다.

다음은 domain-config-template.properties 파일의 기본 설정 화면이다.

```
#================================================================================
# [Default configuration template]
# This template will be used when generating default domain-configurations via admin
# tool(e.g. create-domain).
#
# System admin can modify this to change the default template if needed.
# Do not modify option name.
#================================================================================

# Default option values. You can input your options.
domain=domain1
productionmode=true
domain.admin.server.name=adminServer
cloud.server.name=server
domain.admin.server.jvm.config=-Xmx1024m -XX:MaxMetaspaceSize=512m
domain.admin.server.jeus.port=9736
domain.admin.server.http.port=8088
transport.type=HYBRID
transport.address=230.30.1.1
transport.port=12488
# password's plain text is jeus.
# If you want to set encrypted password, change it by set-password command with algorithm option in
jeusadmin
jeus.password={SHA-256}UyhKRdViLWdFJefDZhJXWtqIJ55ByAl4jldD6hlcuIg=
jeus.username=jeus
# Node configuration
nodename=node1
# Other configuration
jeus.lang=ko
jvm.vendor=Sun

# If you want to set native library folder manually, define "source" to name of folder in
JEUS_HOME/setup/lib_native
#source=sunos_64

# target xsd file for config (default: jeus-domain.xsd,security-domains.xsd,jeus-nodes.xsd,jeus-po-
service-model.xsd)
source.schemas=jeus-domain.xsd,security-domains.xsd,jeus-nodes.xsd,jeus-po-service-model.xsd
```

다음은 domain-config-template.properties 주요 설정 항목에 대한 설명이다.

항목	설명
domain	사용할 domain의 이름을 정하는 항목이다.

항목	설명
productionmode	◦ true : 설치할 때 Production Mode로 설치한다. JEUS Hot Swap, Automatic Reloading을 사용하지 않는다. demo license가 사용되면 경고 메시지가 출력된다. ◦ false : 설치할 때 Development Mode로 설치한다. JEUS Hot Swap, Automatic Reloading을 사용한다
domain.admin.server.name	MasterServer의 이름을 정하는 항목이다.
domain.admin.server.jvm.config	MasterServer의 Jvm Option을 설정하는 항목이다.
domain.admin.server.jeus.port	MasterServer의 BASE Port을 설정하는 항목이다.
domain.admin.server.http.port	MasterServer의 HTTP Port를 설정하는 항목이다.
transport.type	transport type을 설정한다. ◦ DUMMY ◦ HYBRID ◦ MESH ◦ TREE
transport.address, transport.port	선택한 transport의 종류에 따라서 그 동작에 필요한 주소 및 포트 값을 설정하는 항목이다.
jeus.password	관리자 계정의 패스워드를 설정하는 항목이다.
jeus.username	관리자 계정의 아이디를 설정하는 항목이다.
nodename	해당 노드로 분류된 서버에 대한 정보를 설정하는 항목이다.