Installation and Getting Started

JEUS 9

TMAXSOFT

Copyright

Company Information

TmaxSoft Co., Ltd.

TmaxTower 8-9F, 29, Hwangsaeul-ro 258 beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, South Korea

Website: https://www.tmaxsoft.com/en/

Restricted Rights Legend

Trademarks

Other products and company names are trademarks or registered trademarks of their respective owners.

The names of companies, systems, and products mentioned in this manual may not necessarily be indicated with a trademark symbol (™, ®).

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses: APACHE2.0, CDDL1.0, EDL1.0, OPEN SYMPHONY SOFTWARE1.1, TRILEAD-SSH2, Bouncy Castle, BSD, MIT, SIL OPEN FONT1.1

Detailed Information related to the license can be found in the following directory: ${INSTALL_PATH}/license/oss_licenses

Document History

Product Version	Guide Version	Date	Remarks
JEUS 9	3.1.1	2024-12-24	-

Contents

Part Ⅰ. JEUS Installation . . . 1
1. Installation Overview and Pre-Installation Tasks . . . 2
1.1. Overview . . . 2
1.2. System requirements . . . 2
1.3. Installation Order . . . 3
1.4. Installing Java JDK . . . 4
1.5. JEUS Licensing . . . 4
2. Installing and Uninstalling JEUS on UNIX . . . 5
2.1. Overview . . . 5
2.2. Installing . . . 5
2.3. Verifying Installation . . . 10
2.3.1. Directory Structure . . . 10
2.3.2. Environment Configurations . . . 14
2.3.3. Starting JEUS . . . 15
2.4. Uninstallation . . . 20
2.5. Verifying Uninstallation . . . 21
Part Ⅱ. Starting JEUS . . . 22
3. Introduction . . . 23
4. Configuring a JEUS System . . . 24
4.1. Overview . . . 24
4.2. Configuring Basic Environment Variables . . . 24
4.3. Adding and Configuring Managed Servers (MS) . . . 25
4.4. Adding Data Sources . . . 28
4.5. Starting and Stopping Servers . . . 29
5. Using WebTier . . . 31
5.1. Example . . . 31
5.2. Compilation . . . 33
5.3. Deploy . . . 34
5.4. Execution Result . . . 34
6. Using EJB . . . 37
6.1. Session Bean Example . . . 37
6.1.1. Sample Code . . . 37
6.1.2. Compilation . . . 39
6.1.3. Deployment . . . 39
6.1.4. Execution Result . . . 39
6.2. Example of Java Persistence API . . . 41
6.2.1. Sample File . . . 41
6.2.2. Compilation . . . 45

6.2.3. Deployment . . . 46
6.2.4. Execution Result . . . 46
7. Using Web Services . . . 48
7.1. Creating Web Services . . . 48
7.1.1. Creating a Web Service From Java . . . 48
7.1.2. Creating a Web Service From WSDL . . . 50
7.2. Building Web Service Clients . . . 52
7.2.1. Developing a Java SE Client . . . 52
Appendix A: Configuring IPv6 . . . 54
A.1. Introduction . . . 54
A.2. Configuring IPv6 Environment . . . 54
Appendix B: domain-config-template.properties Configuration . . . 57

Part Ⅰ. JEUS Installation

1. Installation Overview and Pre-Installation Tasks

This chapter briefly describes how to install JEUS and provides the system requirements and JDK environment configurations for the installation.

1.1. Overview

JEUS supports console mode for installing JEUS in UNIX/Linux environments.

The installation file performs the following actions.

- Displays the JEUS license agreement. Read it carefully, and agree with the terms to install JEUS.
- Copies JEUS configuration files and directories.
- Sets JEUS environment variables.

1.2. System requirements

The following are the hardware and software requirements for JEUS.

- System requirements

  JEUS requires the following.

Platform	Required Environment
AIX, Linux	JDK 8, JDK 11 or JDK 17 More than 2 GB of hard disk space available

- Supported platforms

  The following are the platforms supported by JEUS.

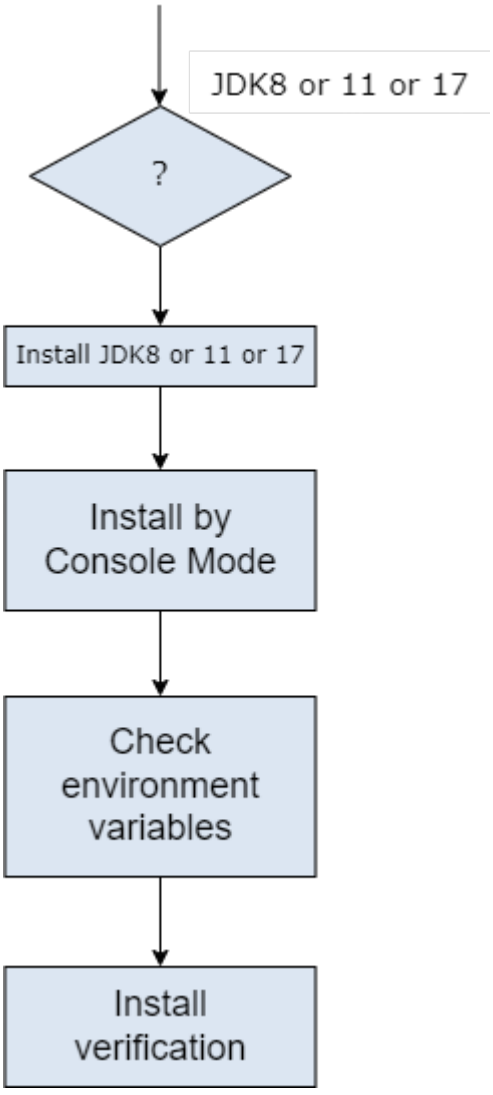
OS Version	CPU	RAM Memory	HardDisc Space	JDK Version
IBM AIX 5L, 6L, AIX 7L	RS6000 IBM pSeries(PowerPC)	1 GB	20 GB	8 or later
Linux series (Kernel 2.6 or later)	Intel x86 series k2.6 or later (k2.4 supported) Intel Itanium Series k2.6 or later IBM pSeries (PowerPC) k2.6 or later	1 GB	20 GB	8 or later

1.3. Installation Order

JEUS is installed in the following order on UNIX. Because the steps are different for each system, refer to each section for detailed information.

1. Install JDK 8, 11 or 17.
2. Install JEUS (copy files) in console mode.
3. Configure and verify the environment variables.
4. Verify JEUS installation.

The following figure shows the JEUS installation order.



JEUS Installation Order

A box in the previous figure represents an action. For detailed information, refer to the relevant section.

For information about how to install and uninstall JEUS in console mode on UNIX, refer to Install and Uninstall.

1.4. Installing Java JDK

To use JEUS, the Java SE Development Kit (JDK) must be installed.

Ensure that JDK 8, 11 or 17 is installed before installing JEUS. After installation, add the bin directory of the installed JDK to the environment variable PATH.

1.5. JEUS Licensing

A license key issued by TmaxSoft is required to use JEUS. A trial license is included with the version of JEUS installed by the installer by default. The license file is named 'license' and is in the 'JEUS_HOME/license' directory.

There are two license editions: Standard and Enterprise. They each have different features and license period.

To upgrade a license or to purchase a full license, contact a TmaxSoft sales representative or TmaxSoft directly. To use a license file, copy it to the 'JEUS_HOME/license' directory with the name of 'license'.

If the license file is not named 'license', rename it as 'license'.

2. Installing and Uninstalling JEUS on UNIX

This chapter describes how to install and uninstall JEUS.

2.1. Overview

Follow these steps to install/uninstall JEUS on UNIX/Linux in console mode.

The following are the steps for installing JEUS in console mode.

1. Install JDK 8, 11 or 17.
2. Install JEUS. (Copy files)
3. Verify the installation.
    - Verify that JEUS starts.
    - Configure and verify the environment variables.

1. If there is a JEUS patch available, it can be applied in the jext, jlext, and jnext in JEUS_HOME/lib.

2.2. Installing

This section describes how to install JEUS in console mode.

This section describes how to install JEUS in console mode (command line). First, execute the console installer.

The following are the steps for installing JEUS in console mode.

1. Insert and mount the installation CD. Go to the directory where jeus9000_unix_generic_ko.bin exists.
2. Open a console window and execute the following command to grant execution permissions for jeus9000_unix_generic_ko.bin.

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

3. Enter the following command in the console and press <Enter>.

```
[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. The JEUS License Agreement appears.

```
===============================================================================
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. To read the remainder of the JEUS license agreement, press <ENTER>.

```
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. Accept or decline the license agreement. Enter 'y' to accept or 'n' to decline the agreement, and then press <ENTER>.

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

7. Choose the platform on which to install JEUS.

```
===============================================================================
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. Choose the installation folder. Use the default path by pressing <ENTER>, or enter another path.

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

Enter the installation folder.

  Default Install Folder: /home/jeus

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

9. Enter the JDK path. If <ENTER> is pressed without specifying the path, the default JDK path will be used. Enter another path, if desired.

```
===============================================================================
Enter the JDK path.
```

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

Enter the JDK path:

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

10. Choose the installation type. Choose one of Master Server and Managed Server. The default is Master Server.

```
===============================================================================
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
   :
```

The following describes each installation type.

Item	Description
Master Server	Installs Master Server. Master Server manages Managed Servers.
Managed Server	Installs Managed Server.

11. In case of choosing Master Server in the step 10, choose an installation mode. The default is Production Mode.

```
===============================================================================
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:
```

The following describes each installation type.

Item	Description
Production Mode	Install JEUS in Production Mode. JEUS Hot Swap and Automatic Reloading are disabled. A warning message is displayed when a demo license is used.
Development Mode	Install JEUS in Development Mode. JEUS Hot Swap and Automatic Reloading are enabled.

12. In case of choosing Master Server in the step 10, enter the domain name.

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

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

Domain Name (Default: jeus_domain):
```

13. In case of choosing Master Server in the step 10, enter the JEUS administrator password. This password is used for the account, "administrator".

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

Enter the Password for the administrator account.


Input Password:
```

Enter the password carefully. It is saved in memory and required to start JEUS.

14. Check the installation summary. Press <ENTER> to start the installation.

```
================================================================================
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. Monitor the installation progress.

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

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

16. Confirm that the installation is completed successfully.

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

JEUS 9 has been successfully installed to:

     /home/jeus

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

2.3. Verifying Installation

After installing JEUS, you need to configure the environment variables and start JEUS to verify that it has been installed properly. Verify the following directory structure before setting the environment variables.

2.3.1. Directory Structure

The following is the directory structre of the installed JEUS.

```
{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
```

The following describes each directory and file.

{JEUS_HOME}

The root directory of JEUS. The actual directory name and location are selected during installation.

bin

Contains the scripts to start and stop servers. The scripts are startMasterServer, startManagedServer, and stopServer. This directory also contains the executable files including the JEUS console tool 'jeusadmin'.

derby

Contains Apache Derby, which helps users build sample applications or perform tests.

docs

Contains Javadoc documentations for the APIs provided by JEUS.

lib

Contains the libraries used to start JEUS. Users only need to access the shared directory.

Directory	Description
shared	Contains the libraries used by applications. Library information must be added to libraries.xml in order to use the libraries in the shared directory. Reference information about the library must also be specified in the JEUS deployment descriptor of the application that will use the library. For more information about shared libraries, refer to "Shared Libraries" in JEUS Applications & Deployment Guide.

license

Contains JEUS license files that are needed to execute JEUS.

setup

Contains the files needed to set up the environment to use JEUS after JEUS has been installed.

templates

Contains configuration file templates.

samples

Contains example files for JEUS.

webserver

Directory where the JEUS Web server is installed during the JEUS installation. For details, refer to "JEUS Web Engine Guide".

domains

Each domain contains a file named nodes.xml that contains the information about the nodes used from DOMAIN_HOME and JEUS_HOME.

DOMAIN_HOME contains the following files and directories:

- .applications

  Contains application files managed by the domain.

  These can only be added or deleted by using the **install-application** and **uninstall-application** commands. This is a JEUS system directory with restricted user access. For detailed information about each command, refer to "install-application" and "uninstall-application" in JEUS Reference Guide.

- .deploymentplans

  Contains deployment plan files managed by the domain.

  These can only be added or deleted by using the **install-deployment-plan** and **uninstall-deployment-plan** commands. This is a JEUS system directory with restricted user access. For detailed information about each command, refer to "install-deployment-plan" and "uninstall-deployment-plan" in JEUS Reference Guide.

- .libraries

  Contains library files managed by the domain.

  These can only be added or deleted by using the **install-library** and **uninstall-library** commands. This is a JEUS system directory with restricted user access. For detailed information about each command, refer to "install-library" and "uninstall-library" in JEUS Reference Guide.

- bin

  Contains the scripts to start or stop the Master Server and Managed Server of the domain. The functions of the scripts are the same as startMasterServer, startManagedServer, and stopServer scripts in the 'JEUS_HOME/bin' directory, except that the users don’t need to specify the file names.

- config

  Contains backup files that store changes to domain.xml, a domain configuration file. For detailed information about domain configurations, refer to "Changing Domain Settings" in JEUS Domain Guide.

Directory	Description
security	◦ SYSTEM_DOMAIN: Contains security domain files, including accounts.xml and policies.xml. Each XML file can be dynamically modified by using jeusadmin. For detailed information about security domain configurations, refer to "Configuring the Security System Domain" in JEUS Security Guide. ◦ security-domains.xml: Contains security domain configuration for JEUS. ◦ security.key: Contains the keys for symmetric key encryption. They are created when 'JEUS_HOME/bin/encryption' is executed. For detailed information about the security.key file, refer to "Configuring Password Security" in JEUS Security Guide. ◦ policy: Contains the Java permissions configuration file. This is used by Java SE Security Manager, separate from the JEUS security system.
servlet	◦ web.xml: This file is used when a web module does not have a separate web.xml file. By default, the file is empty. ◦ webcommon.xml: Settings that apply to all Web modules of the Web engines in the domain. For detailed information about the file, refer to "Directory Structure" in JEUS Web Engine Guide.

- lib/application

  Contains the shared application libraries for the domain.

  If a library conflicts with an application library in the SERVER_HOME directory, it is overridden

by 'SERVER_HOME/lib/application' and a warning message appears. For detailed information about the 'lib/application' directory, refer to "lib/application Directory" in JEUS Applications & Deployment Guide.

- servers

  Create the SERVER_HOME directory by using the server name in this directory. For detailed information about the directory structure, refer to "Server Directory Structure" in JEUS Server Guide.

Directory	Description
.workspace	Contains workspaces used by each JEUS server. Cannot be modified by the user.
bin	Contains scripts for starting and stopping the server. The scripts execute the same functions as those in 'JEUS_HOME/bin', but they do not require the domain and server names. ◦ Master Server: uses startMasterServer and stopServer. ◦ Managed Server: uses startManagedServer and stopserver.
lib/application	Contains application libraries for the server. This directory takes precedence over the domain-level library directory (DOMAIN_HOME/lib/application). If a library conflicts with an application library in the DOMAIN_HOME/lib/application, the file in this directory overrides that in DOMAIN_HOME and a warning message appears. For detailed information about lib/application, refer to "lib/application Directory" in JEUS Applications & Deployment Guide.
logs	Includes launcher logs, server logs, and access log files. For detailed information, refer to "Logging" in JEUS Server Guide.

2.3.2. Environment Configurations

The environment variables must be configured in order to use JEUS. Some environment variables are set during the installation, but they can be changed if needed. The PATH variable is set in the **.profile/.cshrc** file and other environment variables are set in the **$JEUS_HOME/bin/jeus.peoperties** file. If you want to set different environment variables for each server, create a **$JEUS_HOME/bin/<*SERVER_NAME*>.properties** file. When starting the server, use the [-server] option to specify the server name.

The following describes each environment variable.

Environment Variable	Description
PATH	System path. It must include: ◦ /home/jeus/bin ◦ /home/jeus/lib/system
JEUS_HOME	JEUS installation directory. (Example: /home/jeus)
JEUS_LIBPATH	JEUS library file path. (Example: /home/jeus/lib/system)
VM_TYPE	Option to use the Java HotSpot JVM. (Example: hotspot or old)
USERNAME	Administrator account ID.
PASSWORD	Administrator account password.
JAVA_HOME	Path to JDK. (Example: /usr/jdk17)
ANT_HOME	ANT installation directory. (Example: home/jeus/lib/etc/ant)
JAVA_ARGS	JDK parameters.
JAVA_VENDOR	JDK vendor. (Example: Sun, IBM, or HP)

The 'setenv' command is used to set the JEUS_HOME variables in the console.

```
setenv JEUS_HOME “/home/jeus”
```

The following is an example of setting the system path.

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

Since the Java executable directory (**/usr/jdk17/bin**) is used by JEUS, it must be added before other environment variables.

2.3.3. Starting JEUS

Use the following steps to start JEUS in order to verify that JEUS has been installed properly.

1. Start the Master Server (MASTER) by entering 'startMasterServer' command at the console prompt. Default ID is 'administrator' and password is the input value during JEUS installation.

   The following is how to start the Master Server (MASTER) by executing the command.

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

Once the Master Server is started, you will see the message saying, 'Successfully started the server. The server state is now RUNNING'.

```
[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).
```

```
... Omitted

[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      |                                        |
+------------+-------+--------------+-----------+----------------------------------------+

... Omitted

[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. If an "Invalid License" message is displayed, there is a problem with the license. Obtain a license from TmaxSoft and copy it to the '$JEUS_HOME/license' directory.
2. Verify that all the steps have been completed successfully and that the environment variables have been configured correctly. In particular, check that the '/jeus/bin' directory is included in the system path so that the startMasterServer script can be executed.

2. Start Managed Server (MS). You can start the Managed Server by executing the 'startManagedServer' command.
   - **startManagedServer command**

     The following shows how to start the Managed Server (MS) using the startManagedServer command.

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

When you enter the command at the console prompt, the following message is displayed. Generally, the default ID is 'administrator' and the password is the value entered during the JEUS installation. If the JEUS MS successfully boots and starts, you will see the message, saying 'Successfully started the server. The server state is now RUNNING'.

```
[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).

... Omitted

[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. If an "Invalid License" message is displayed, there is a problem with the license. Obtain a license from TmaxSoft and copy it to the '$JEUS_HOME/license' directory.
2. Verify that all the steps have been completed successfully and that the environment variables have been configured correctly. In particular, check that the '/jeus/bin' directory is included in the system path so that the startManagedServer script can be executed.

3. Execute the **jeusadmin** command in another console window. Default ID is 'administrator' and password is the input value during JEUS installation.

```
[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. A message that JEUS has been started successfully will be displayed and the prompt shows that it is ready to accept user input.
5. Log in to jeusadmin at the console. The JEUS server can be controlled by using the **local-start-server** and **local-shutdown** commands of the tool. To terminate a JEUS server, enter **local-shutdown** command.

```
[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. Enter **exit** to exit jeusadmin.

```
offline>exit
```

2.4. Uninstallation

This section explains how to uninstall JEUS in console mode.

The following describes the steps for uninstalling JEUS in console mode.

1. Execute '$JEUS_HOME/UninstallerData/Uninstall' from where JEUS is installed to remove JEUS Core and the installation directories as shown in the following.

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

2. JEUS will be uninstalled. When the uninstallation is complete, a message that the uninstallation is complete is displayed.

```
===============================================================================
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. Verifying Uninstallation

Check that all JEUS directories and files have been removed. Any files created after the installation of JEUS will not be removed. These files need to be deleted manually.

Part II. Starting JEUS

3. Introduction

"Part III. Starting JEUS" is intended for users who want to develop programs by using JEUS, TmaxSoft's Web Application Server, for the first time. This part consists of the following chapters.

- **Configuring JEUS**

  Describes how to configure and operate JEUS.

- **Using WebTier**

  Describes how to package Web applications and how to execute servlet, JSP, JSTL, and JSF pages.

- **Using EJBs**

  Describes how to package and deploy EJB modules and how to use stateless session beans and the Java Persistence API.

- **Using Web Services**

  Describes how to package and deploy web services and how to use servlet and EJB endpoints.

TmaxSoft recommends the users to execute the examples in this document to fully understand JEUS Web Application Server.

For more detailed technical documentation, see "JEUS Server Guide", "JEUS EJB Guide", and "JEUS Web Engine Guide"

4. Configuring a JEUS System

This chapter describes how to set up and start JEUS and WebAdmin.

4.1. Overview

The following are the steps for configuring a JEUS system by using jeusadmin.

- Configuring basic environment variables
- Adding and configuring Managed Servers
- Adding data sources

4.2. Configuring Basic Environment Variables

JEUS's jeusadmin can be used to easily manage all JEUS components, configure the environment, perform monitoring, and manage applications.

Use the following steps to run jeusadmin.

1. Run `startMasterServer` from the command prompt to start JEUS Master Server.

   The following is an example of executing JEUS MASTER.

```
[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).

... Omitted

[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..
```

The startMasterServer script is in the 'JEUS_HOME/bin/' directory and must be set in the system path.

4.3. Adding and Configuring Managed Servers (MS)

The server instances that manage the actual application service engines and services are called Managed Servers (MSs). Multiple MSs can exist in a single domain. Applications are deployed to MSs, and the MSs provide the resources and services that the applications need.

Adding Managed Servers

Add new MSs and then add listeners to the MSs.

1. After connecting to jeusadmin, run **add-server** to add an MS.

```
[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".
```

The BASE listener is used to start the MSs. The BASE listener uses port 9736 by default. Because MASTER uses the same port, change the BASE listener to use another port.

2. When the server has been added, you can run **server-info** to check that the MS has been dynamically created.

```
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   |     |(Tue) PM   |        |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    |
|        |         |     |     |     |(Tue) PM   |        |           |        |
|        |         |     |     |     |03:06:43   |        |           |        |
|        |         |     |     |     |KST        |        |           |        |
+--------+---------+-----+-----+-----+-----------+--------+-----------+--------+
================================================================================
```

3. You can check the listener set through the **list-server-listener** command.

```
[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. You can update listener settings using the **modify-listener** command.

```
[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 |
+-------------------------------------+---------+------+
=============================================================
```

Configuring HTTP Listeners and Connectors

Only the **[Basic]** section needs to be configured to start an MS. However, additional HTTP Listener property settings are required to use the Web services.

1. After connecting to jeusadmin, run **add-listener** to add a listener.

```
[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]
...(Omitted)

[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.
```

The specified port must not be already in use by another listener.

2. Add a web connection using the **add-http-listener** command.

```
[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]
...(Omitted)

[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. Check the registered information using the **show-web-engine-configuration** command.

4.4. Adding Data Sources

JEUS can be used to configure the database connectivity by configuring data sources.

The following example uses the Apache Derby database that is included with JEUS. Apache Derby is located in the 'JEUS_HOME/derby' directory.

Execute the following command to start Derby.

```
JEUS_HOME\bin> startderby
```

Execute the following command to stop Derby.

```
JEUS_HOME\bin> stopderby
```

Derbyclient.jar, the Derby JDBC driver file, must be located in the 'JEUS_HOME/lib/datasource' directory to be able use Derby in JEUS. This file is included in JEUS by default. For detailed information about Derby, refer to http://db.apache.org/derby/.

The following example registers a data source named 'jdbc/sample' to use a database called 'sample'.

The following shows how to add a data source using the console tool.

1. Log in to JEUS as jeusadmin.

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

2. Add a data source to MASTER.

```
[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. Add the data source to an MS (server2).

```
[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. Starting and Stopping Servers

In JEUS, MS can be started with the startManagedServer script located in JEUS_HOME/bin, and can be terminated with the stopServer script.

1. Start the added MS using the **startManagedServer** script.

```
[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).
... Omitted
[2024.09.25 18:00:10][0] [launcher-1] [Launcher-0040] Successfully started the server[server2].
The server state is now RUNNING.
```

2. Shut down the MS using the **stopServer** script.

```
[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. Using WebTier

This chapter describes how to deploy servlet, JSP, JSTL, and JSF applications, and how to package and deploy WAR (Web application ARchive) modules.

5.1. Example

This section shows sample code for a web application and how to compile and deploy the code.

For more details, see "JEUS Server Guide", "JEUS Web Engine Guide", and "JEUS Web Service Guide"

The following sample servlet displays the message "Hello World!" in a web browser.

Example of WebTier: <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();
   }
}
```

The sample file is in the following directory.

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

The following sample JSP program named 'snoop.jsp' shows information about the request it receives.

Example of WebTier: <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>
```

The sample file is in the following directory.

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

The following sample JSP program has the same functionality as `snoop.jsp` except that it uses JSTP and JSF.

Example of WebTier: <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>
```

JSP programs are automatically compiled by the servlet engine and don’t need to be compiled manually.

5.2. Compilation

The sample code can be built by using jant as in the following.

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

After the sample code has been built successfully, a WAR file named 'hello-servlet.war' will be created

in the `dist` folder.

5.3. Deploy

A packaged WAR application can be deployed by using the console tool.

In JEUS, there are two steps for deploying applications, installation and deployment.

For more information about deployment, see "JEUS Application & Deployment Guide"

The following describes how to deploy WAR applications.

Using the console tool

This section describes how to deploy and test a WAR application in the console.

1. Log in to JEUS as jeusadmin.

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

2. Install the application on MASTER.

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

3. Deploy the application to an MS (Server3).

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

4. Verify that the application has been deployed successfully.

5.4. Execution Result

This section describes how to execute the deployed servlets and JSPs.

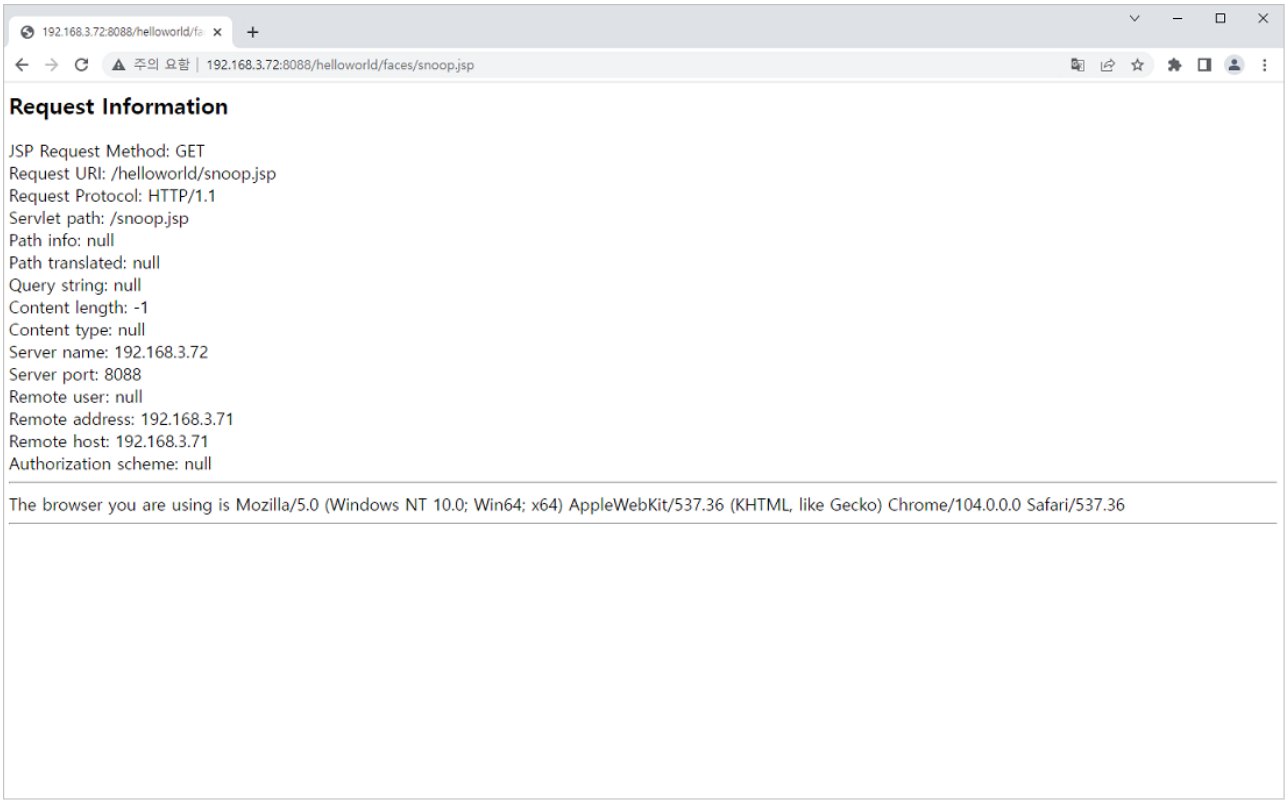
Executing a Deployed JSP

Deployed JSPs can be executed by:

- Opening snoop.jsp

  Open a browser window and enter the following URL into the address bar to open the 'snoop.jsp' page. Opening a JSP page for the first time takes a little longer because it is automatically compiled by the servlet engine.

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



Executing a JSP

- Opening snoop_jstl.jsp

  Open a browser window and enter the following URL into the address bar to open the snoop_jstl.jsp. The result of a page is the same with the result of opening snoop.jsp.

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

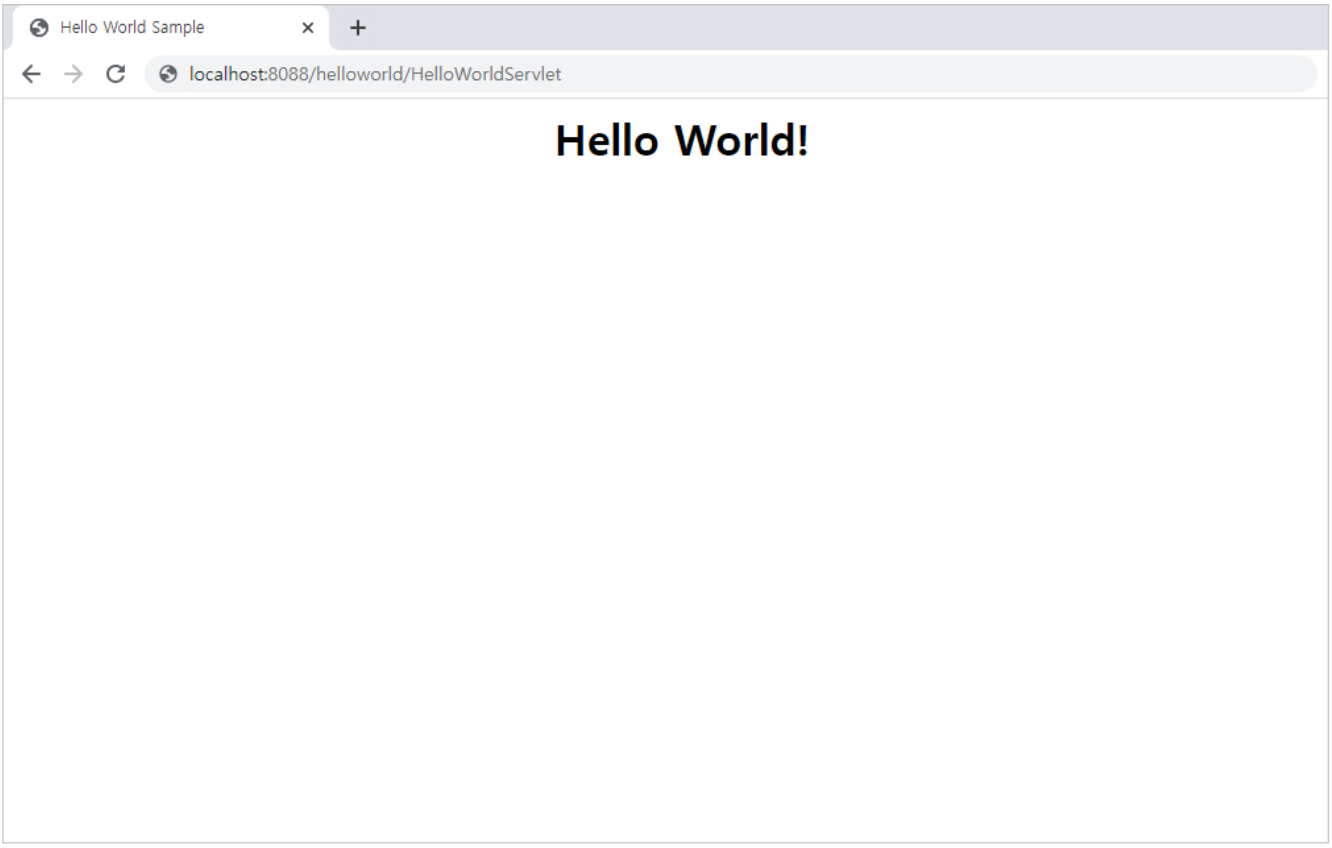
Executing a Deployed Servlet

To access the 'helloworld' servlet, open a browser window and enter the following URL into the address bar.

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

Item	Description
http	HTTP protocol is used to access JEUS.
localhost	Server that provides the service is on the local machine.
8088	Port number of the HTTP listener created on the MS.
helloworld	Web application context path. This path is set in the <context-path> element of 'jeus-web-dd.xml'. By default, it has the same name as the WAR file.
HelloWorldServlet	URL pattern defined in the servlet.

The following message will appear if the servlet engine has started normally and the Hello World servlet has been deployed successfully.



Executing a WAR Module (Servlet)

6. Using EJB

This chapter provides examples of developing and deploying an entity using stateless session beans and the Java Persistence API.

6.1. Session Bean Example

A session bean consists of a business interface and a bean class.

6.1.1. Sample Code

Example of an EJB

The following example shows a stateless session bean that has a method that displays helloejb.

- Business Interface

  Example of Stateless Session Bean: <Hello.java>

```
package helloejb;

import jakarta.ejb.Remote;

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

- Bean Class

  Example of Stateless Session Bean: <HelloBean.java>

```
package helloejb;

import jakarta.ejb.Stateless;

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

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

The sample file is in the following path.

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

Example of Servlet Client

The following is the implementation of a servlet client that calls helloejb.

Example of Servlet Client: <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);
    }
}
```

The sample file is in the following path.

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

6.1.2. Compilation

The sample code can be built using jant as in the following.

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

After the sample code has been built successfully, an EAR file is created as '%JEUS_HOME%/samples/getting_started/helloejb/dist/helloejb.ear'.

6.1.3. Deployment

A packaged EJB application can be deployed using the console.

Using the console tool

This section describes how to deploy an EJB application in the console tool.

1. Find the helloejb.ear file that was created.
2. Log in to JEUS as jeusadmin.

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

3. Install the application on MASTER.

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

4. Deploy the application to an MS (Server 1).

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

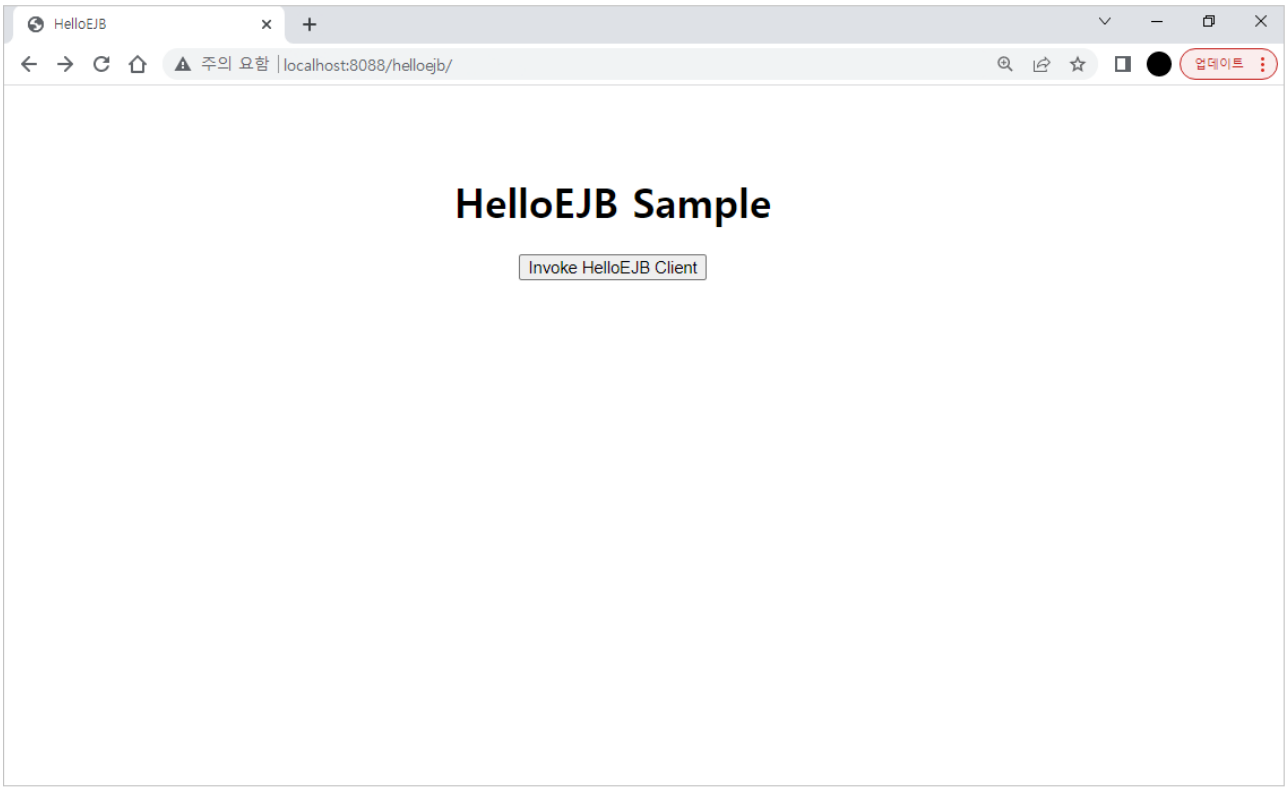
6.1.4. Execution Result

This section describes how to execute and test the deployed application.

To execute HelloEJB:

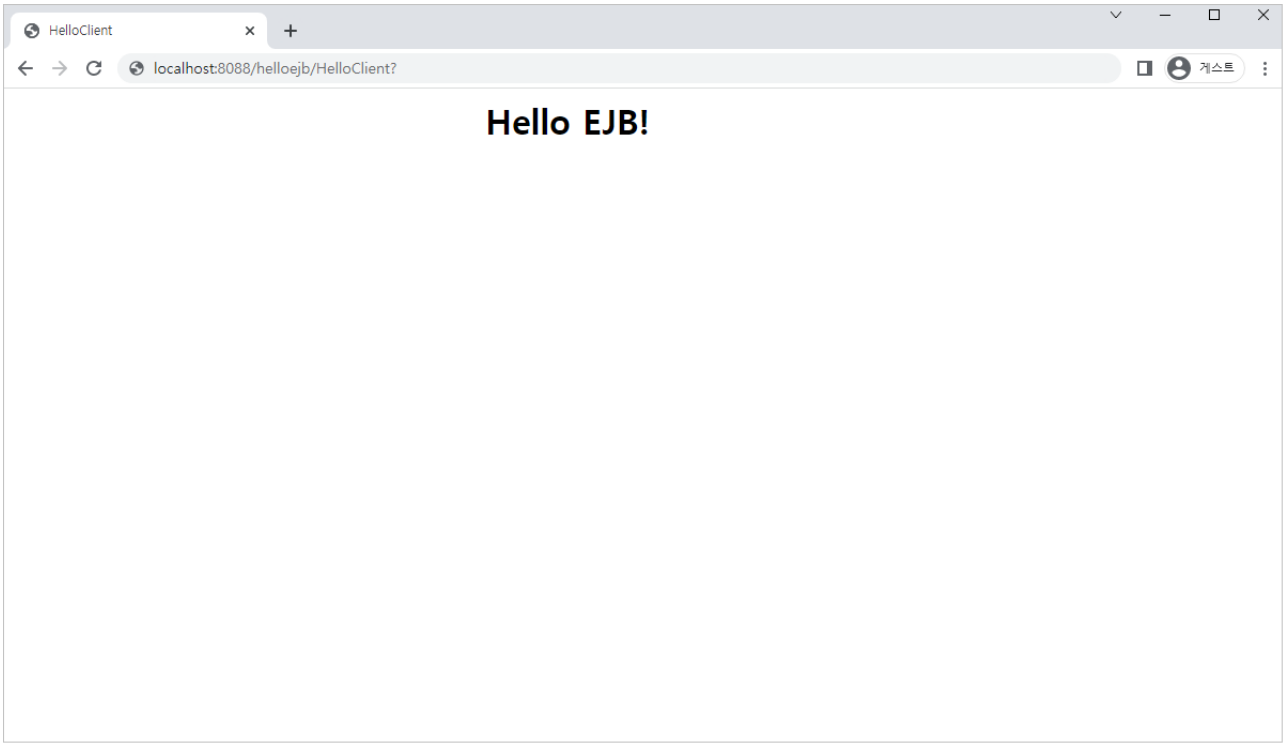
1. Open a browser window and enter the following URL into the address bar.

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



HelloEJB Client Page

2. Click **[Invoke HelloEJB Client]** to execute the HelloClient servlet that invokes the EJB. The following result will appear.



HelloEJB Servlet Execution Result

6.2. Example of Java Persistence API

This section describes how to develop an entity using the Java Persistence API and how to compile and deploy the entity.

6.2.1. Sample File

EJB Example

The sample code consists of a product entity and the ProductManager session bean that processes the entity.

- Entity

  Example of Java Persistence API: <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

  Example of Java Persistence API: <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

  Example of Java Persistence API: <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();
    }
}
```

The sample file is in the following path.

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

Example of Servlet Client

A servlet client that saves and processes data in a database by using the ProductManager EJB is implemented as in the following.

Example of Servlet Client: <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);
    }
}
```

The sample file is in the following path.

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

6.2.2. Compilation

The following are the steps for compiling the sample code using jant. When it is successfully built and the database configuration is completed, the packaged module can be deployed.

1. Execute the **jant build** command from the directory of the sample code file.

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

2. After the sample code has been built correctly, the 'dist\hellojpa.ear' EAR file is created.

Since this example requires a database, Derby must be running. Configure the database with the 'jdbc/sample' data source. For detailed information, refer to System Configuration.

After Derby has been started, create the following database table. In this example, a database named 'sample' is used.

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

3. Create the database table by executing the **jant setup** command as in the following.

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

6.2.3. Deployment

A packaged EJB application can be deployed using the console tool.

Using the console tool

This section describes how to deploy an EJB application in the console.

1. Log in to JEUS as jeusadmin.

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

2. Install the application on MASTER.

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

3. Deploy the application to an MS (Server1).

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

4. Verify that the application has been deployed successfully.

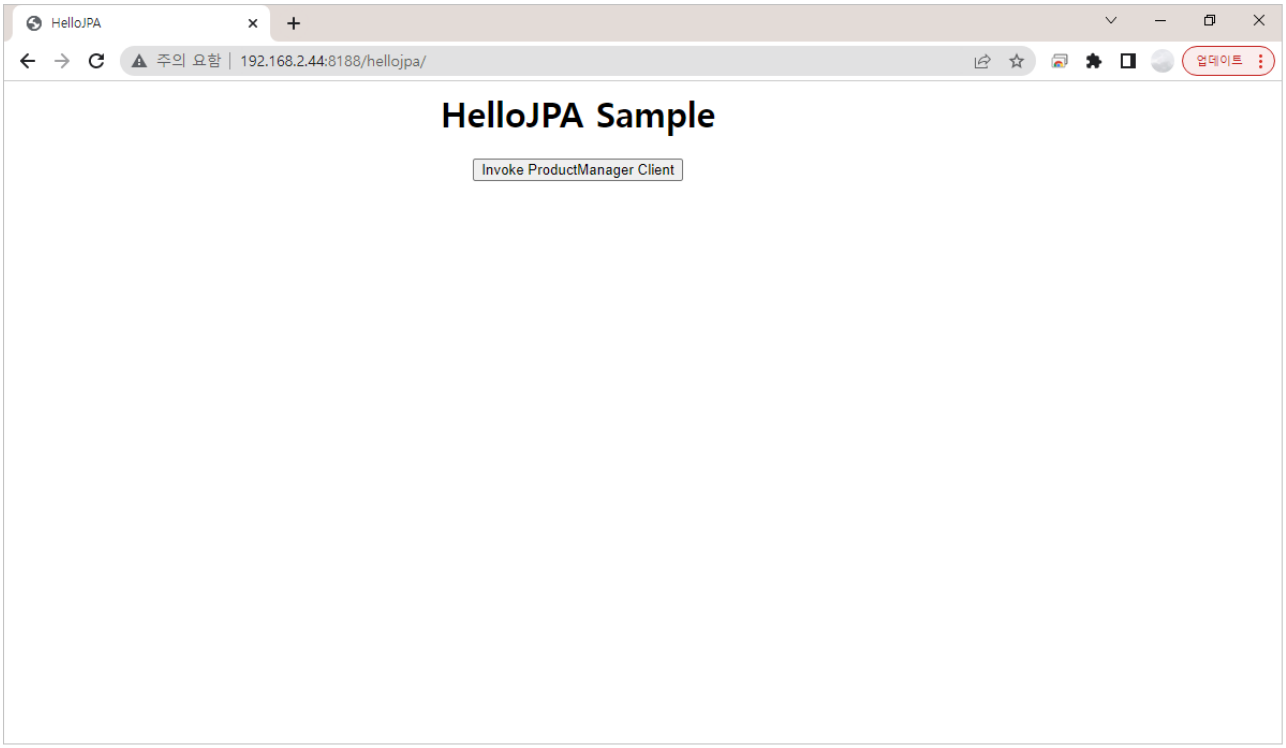
6.2.4. Execution Result

This section describes how to execute and test the deployed application.

To execute HelloJPA:

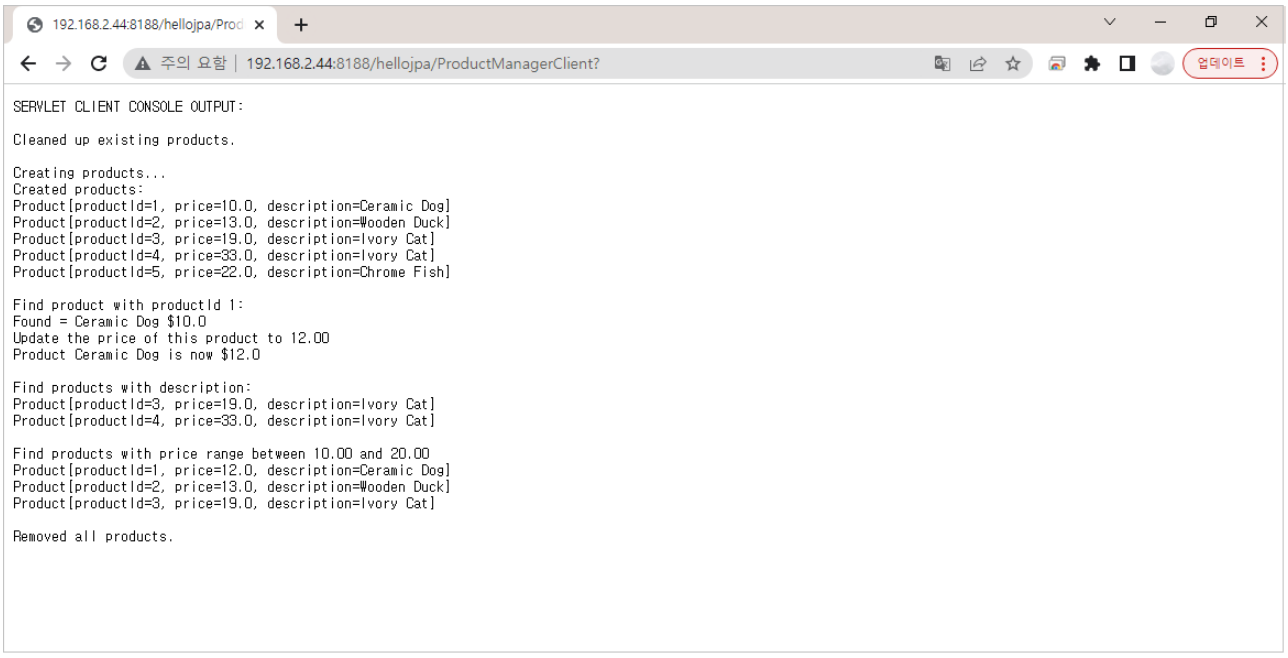
1. Open a browser window and enter the following URL into the address bar.

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



HelloJPA Client Page

2. Click **[Invoke ProductManager Client]** to execute the HelloJPA servlet that invokes the EJB. The following result will appear.



HelloJPA Servlet Execution Result

7. Using Web Services

This chapter describes how to create a web service and client using JAX-WS 2.2, and how to call the web service.

7.1. Creating Web Services

JEUS 9 supports Jakarta EE 8 style web services, and JAX-WS 2.2 is the core of Java EE 6 web services.

JAX-WS replaces the existing JAX-RPC features. Since JAXB 2.2 and later versions fully support all XML schema types, Java to XML mappings can be well defined without having to rely on the JAX-RPC specification.

In addition, a new web service model called Java Web Services 2.0, which includes SAAJ 1.3 that enables direct handling of SOAP 1.2 messages, has been added to JEUS.

7.1.1. Creating a Web Service From Java

Developing a Web service from an existing Java class involves:

1. **Creating a service implementation bean that includes the web service annotations.**

   The following constraints apply when creating service implementation beans:

   - jakarta.jws.WebService annotation must be included to indicate that the class is a service implementation bean.
   - The arguments and the return type of web service methods must be compatible with the mapping definition between JAXB 2.0 and XML schema.
   - The arguments and return type of a web service method should not implement the java.rmi.Remote interface directly or indirectly.

   A method's arguments, return type, and binding method can be customized by using the annotations defined in the jakarta.jws package. The following example shows how to create a service implementation bean.

   The sample web service implementation class is in the following path

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

   Example of Web Service (From Java): <AddNumbersImpl.java>

   ```
   package fromjava.server;

   import jakarta.jws.WebService;

   @WebService
   ```

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

JAX-WS provides the convenience of developing a web service from an existing Java class by adding the @WebService annotation to define the class as a web service.

2. **Creating Artifacts that are Portable Across Vendors**

   After a service implementation bean has been created and compiled, portable artifacts, which are portable across vendors, need to be generated. Portable artifacts are the files generated by a JAX-WS tool that complies with the JAX-WS specification and are portable across vendors. The artifacts include java classes and WSDL that contain information required for accurate Java-to-WSDL mappings.

   JEUS provides a console script called **wsgen**, and it is located in the 'JEUS_HOME/bin' directory.

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

   Portable artifacts are created in the previous path. A WSDL can also be created by using the '-wsdl' option with wsgen. However, this option should not be used because JAX-WS does not need to include a WSDL for a web service endpoint.

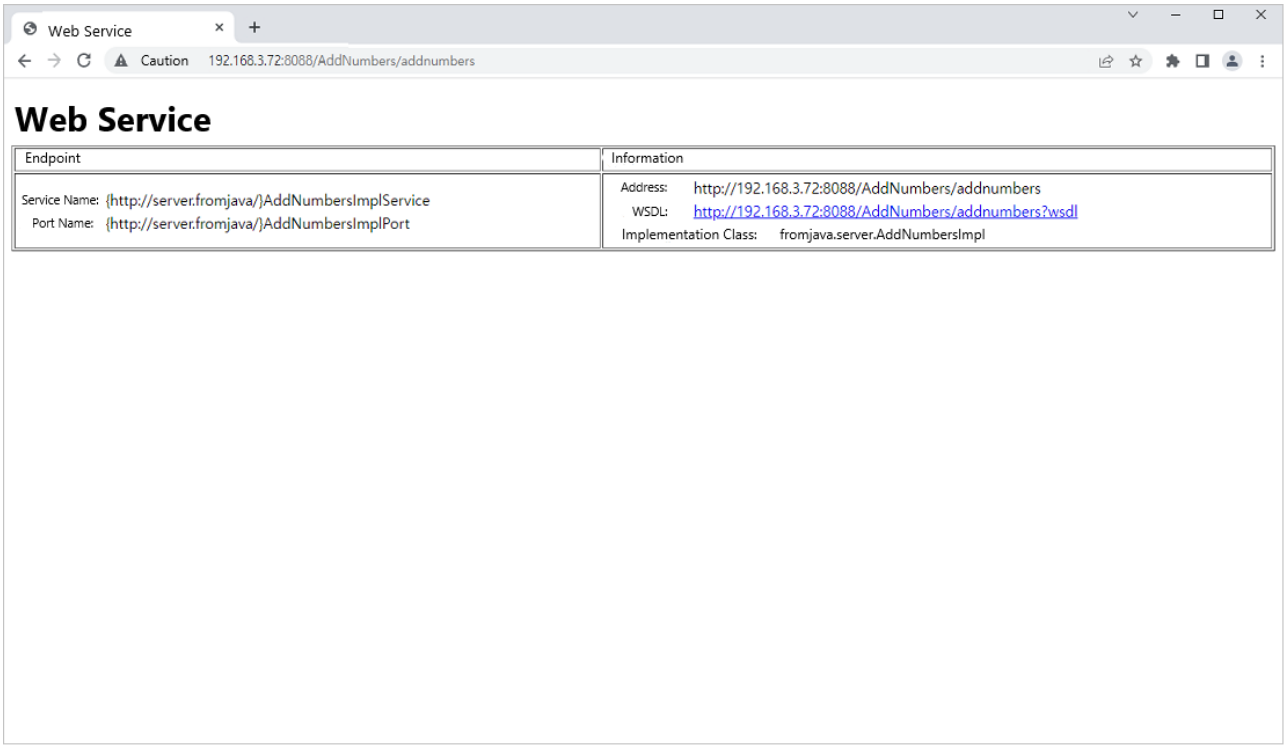
3. **Packaging and Deploying a Web Service**

   Packaging a web service involves packaging the service implementation bean with the Java classes and deployment descriptors that are referenced by the service implementation bean. They are packaged as a WAR file. The following example only includes the fromjava.server.jaxws.AddNumbers, fromjava.server.jaxws.AddNumbersResponse classes. These classes must be in the 'WEB-INF/classes' folder. After being packaged into the AddNumbers.war file, they are deployed to JEUS.

   The web service can be accessed at the following URL.

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

   After the web service has been compiled successfully, the result can be verified in the web browser.



Successful Deployment of a Web Service Created from Java

Simple Way to Execute a Web Service Created from Java

The sample web service can easily be executed in JEUS in the following way.

The tasks, from service packaging to client program execution, can be easily performed by executing the **jant** command from the 'JEUS_HOME/samples/getting_started/webservices/from_java/' directory.

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

After the web service has been compiled successfully, the result can be verified in the web browser.

7.1.2. Creating a Web Service From WSDL

The JAX-RPC programming model can be used to develop a web service from existing Java classes. When starting from a WSDL to create a web service, SOAP messages must be defined and shared through the WSDL. Java classes are created according to the defined message types.

In general, developing a web service from a WSDL involves:

1. **Generating a Service Endpoint Interface.**

   In this step, Java interface and class files for the web service are created from the public WSDL by using the wsimport console script provided by JEUS. The script is in the 'JEUS_HOME/bin' directory.

   Execute the following command from the 'JEUS_HOME/samples/getting_started/webservices/from_wsdl' directory.

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

Artifacts that include service endpoint interfaces and service definition classes will be created in the specified path.

The following example shows the created service endpoint interface. The interface includes JAX-WS annotation.

Example of Web Service (From WSDL): <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. **Implementing a Service Endpoint Interface**

   After a service endpoint interface has been created, a service implementation bean must be implemented with the actual logic. Add the @jakarta.jws.WebService Annotation to the service implementation bean. This annotation must have an endpoint interface member property that defines the service endpoint interface.

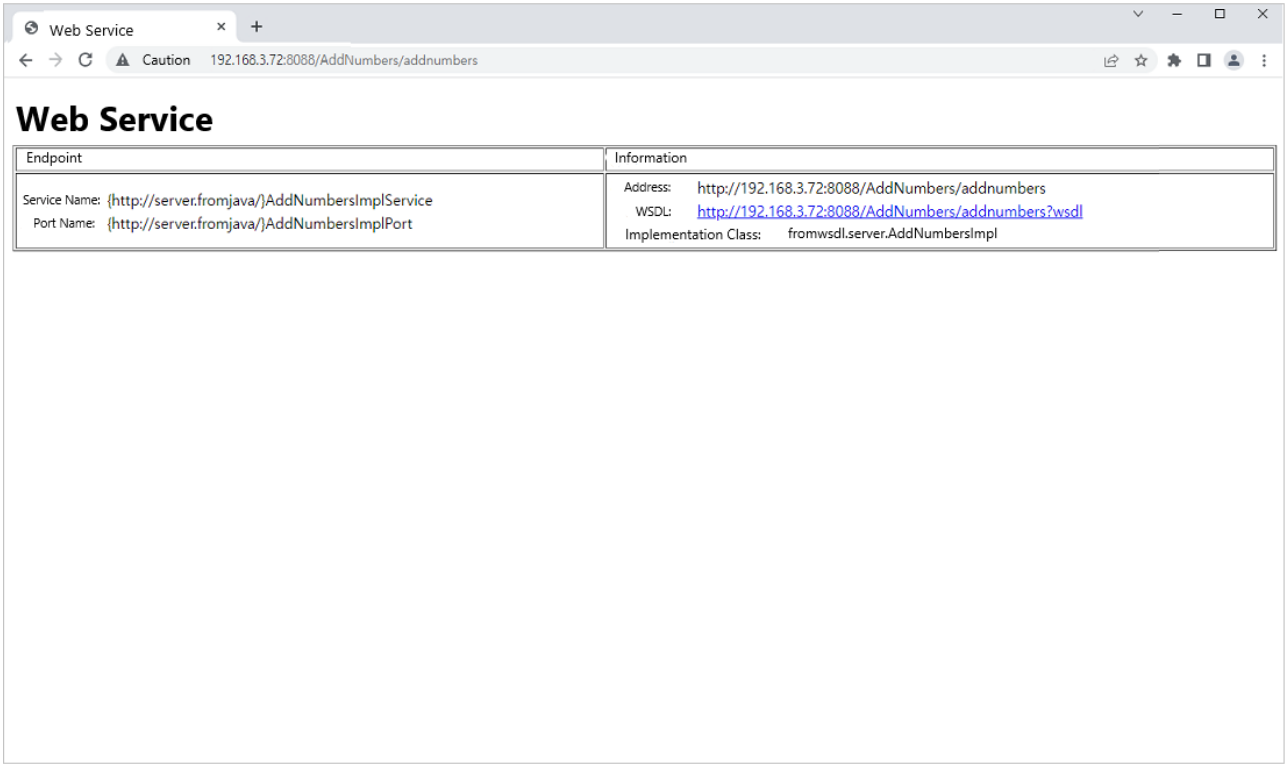
3. **Packaging and Deploying a Web Service**

   Packaging a web service created from WSDL into a WAR file is similar to packaging a web service created from Java classes.

   Generate the portable artifacts required for web service execution. Save the artifacts in the 'WEB-INF/classes' directory. The artifacts include the service endpoint interfaces created by the wsimport script and the service implementation beans. Next, package them into a WAR file. For example, if the WAR package's name is AddNumbers.war, the package can be accessed at the following address after being deployed to JEUS. If the web service has already been created from Java and deployed, it must be removed and then redeployed because they have the same context names.

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

   Use this URL to verify that the web service has been deployed successfully.



Successful Deployment of a Web Service Created from WSDL

Simple Way to Execute a Web Service Created from WSDL

The sample web service can easily be executed in JEUS in the following way.

The tasks, from service packaging to client program execution, can be easily performed by executing the **jant** command from the 'JEUS_HOME/samples/getting_started/webservices/from_wsdl/' directory.

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

After the web service has been compiled successfully, the result can be verified in the web browser.

7.2. Building Web Service Clients

After the web service has been deployed, it can be accessed by client programs. Clients are classified into Java SE clients and Jakarta EE 8 clients based on their operating environments.

This section only describes the Java SE clients that run like regular Java programs.

7.2.1. Developing a Java SE Client

In JAX-WS, a dynamic proxy that corresponds to a web service endpoint is created internally and used as a service endpoint interface implementation. Client programs can invoke a web service method, which is defined by a service endpoint interface, by using the proxy.

Portable artifacts must be generated to allow this. The **wsimport** script tool must be used to create the required artifacts from WSDL.

The following example illustrates how to use wsimport. Run the following command from the 'JEUS_HOME/samples/getting_started/webservices/from_wsdl' directory.

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

After the service endpoint interfaces and the service classes have been created, write a client Java program that uses them.

The following is an example of a web service client program. The example is similar to the sample web service client program provided for JAX-RPC. The sample program creates a service instance and obtains a proxy object that implements the service endpoint interface from the service instance.

The sample code is in the 'JEUS_HOME/samples/getting_started/webservices/from_wsdl/src/java/fromwsdl/client' directory.

Java SE Client: <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!");
        }
    }
}
```

The following result will be displayed when the client program has been compiled and executed successfully.

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

Appendix A: Configuring IPv6

This appendix describes how to configure IPv6.

A.1. Introduction

Use the following steps to configure JEUS in IPv6 environment.

1. Modify the jeus.properties, jeusadmin, startMasterServer, startManagedServer, stopServer, startderby, stopderby, mcastReceiver, and mcastSender scripts.
2. Modify domain.xml.
3. Verify operation.

The loopback address must be set to "::1" in the 'hosts' file of the server.

The 'hosts' file is located in the following paths, depending on the OS.

- UNIX

  ```
  /etc
  ```

The following is a sample 'hosts' file.

hosts

```
[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. Configuring IPv6 Environment

This section describes the environment configurations for IPv6. IPv6 environment configurations are different for each file.

Changing a File in JEUS_HOME/bin

- File name: startMasterServer, startManagedServer

  Add -Djava.net.preferIPv6Addresses=true and -Djava.net.preferIPv4Stack=false where -classpath is written.

◦ Example

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

- File name: mcastReceiver, mcastSender
  ◦ Existing environment variable

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

  ◦ Updated environment variables

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

- File name: jeusadmin, stopServer
  ◦ Existing environment variable

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

  ◦ Updated environment variables

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

- File name: jeus.properties
  ◦ Existing environment variable

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

  ◦ Updated environment variables

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

- File name: startderby, stopderby
  ◦ Existing environment variable

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

- Updated environment variables

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

Changing domain.xml

Modify and add JEUS_HOME/domains/DOMAIN_NAME/config/domain.xml file as shown in the following example.

1. If the address of listen-address is set to IPv4, then change it to IPv6.

```
<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. If the address of heartbeat-address is set to IPv4, then change it to IPv6 as shown in the following example. If no configuration has been set, then add the following information.

```
<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. Add java.net.preferIPv4Stack=false and java.net.preferIPv6Addresses=true to the vm-option for each server.

```
<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 Configuration

This appendix describes how to configure the domain-config-template.properties file required for JEUS installation.

The following is the default configuration screen of the domain-config-template.properties file.

```
#================================================================================
# [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
```

The following describes the key configuration items.

Item	Description
domain	Domain name.

Item	Description
productionmode	◦ true: Install in Production Mode. JEUS Hot Swap and Automatic Reloading are not used. A warning message is displayed if a demo license is used. ◦ false: Install in Development Mode. JEUS Hot Swap and Automatic Reloading are used.
domain.admin.server.name	Master Server name.
domain.admimn.server.jvm.config	Jvm option of the Master Server.
domain.admin.server.jeus.port	BASE Port of the Master Server.
domain.admin.server.http.port	HTTP port of the Master Server.
transport.type	Sets the transport type. ◦ DUMMY ◦ HYBRID ◦ MESH ◦ TREE
transport.address, transport.port	Address and port value required for the selected transport type.
jeus.password	Password for the administrator account.
jeus.username	ID for the administrator account.
nodename	Sets the information for the server classified under the node.