



APUSIC
固若长城
睿比世界

用户手册

金蝶Apusic应用服务器 V10.0.10标准版

版权所有 © 深圳市金蝶天燕云计算股份有限公司2026。保留所有权利。

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

 Apusic 是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

- 1 前言
 - 1.1 面向对象
- 2 版本变更说明
- 3 产品简介
 - 3.1 概述
 - 3.2 平台环境
 - 3.3 系统环境
 - 3.4 JavaEE规范
 - 3.5 术语
- 4 产品清单
- 5 许可证授权
 - 5.1 标准版获取特征码
 - 5.2 嵌入式部署获取特征码
- 6 安装和运行
 - 6.1 安装
 - 6.2 卸载
 - 6.3 运行
 - 6.3.1 在Linux环境下
 - 6.3.2 在 Windows环境下
 - 6.4 停止
 - 6.4.1 在Linux环境下
 - 6.4.2 在 Windows环境下
- 7 部署应用
- 8 取消部署应用
- 9 设置 JVM 内存
- 10 配置文件说明
 - 10.1 apusic.conf 文件
 - 10.1.1 listeners 配置 (监听器)
 - 10.1.2 resources 配置 (用户权限)
 - 10.1.3 executors 配置 (线程池)
 - 10.1.4 endpoints 配置 (网络 IO 和协议)
 - 10.1.4.1 SSLHostConfig 元素

- 10.1.4.2 Certificate 元素
- 10.1.5 services 配置 (虚拟主机和应用)
 - 10.1.5.1 Realm
 - 10.1.5.2 host
 - 10.1.5.3 application
- 10.1.6 servers 配置 (引用其他功能)
- 10.1.7 containers 配置 (监听器等)
- 10.2 其他配置
 - 10.2.1 共享库配置
 - 10.2.2 手动部署多个应用
 - 10.2.3 端口的修改
 - 10.2.4 开启 JMX
- 11 配置数据源
- 12 配置 HTTPS
 - 12.1 单向认证 -- 证书格式为 JKS
 - 12.2 双向认证 -- 证书格式为 JKS
 - 12.3 单向认证 -- 证书格式为 PEM
 - 12.4 双向认证 -- 证书格式为 PEM
- 13 国密配置说明
 - 13.1 环境说明
 - 13.2 应用服务器证书配置
 - 13.2.1 单向认证
 - 13.2.2 双向认证
 - 13.2.3 增加 endpoint 到 server
 - 13.3 启动应用服务器
 - 13.4 Springboot 中使用国密
 - 13.4.1 注册国密实现 Jar 包
 - 13.4.2 Pom 文件应用国密实现
 - 13.4.3 应用配置
 - 13.4.4 启动应用
- 14 日志配置
 - 14.1 系统日志配置
- 15 访问日志记录
 - 15.1 配置访问日志

- 15.2 访问控制
 - 15.2.1 远程地址
 - 15.2.2 远程主机名
 - 15.2.3 远程CIDR
- 16 集中配置说明
 - 16.1 使用 apollo 集中配置
 - 16.1.1 添加集中配置服务
 - 16.1.2 在 apollo 的配置
 - 16.1.2.1 apusic.conf 配置
 - 16.1.2.2 数据源的配置
 - 16.1.2.3 日志的配置
 - 16.1.2.4 JVM 的配置
 - 16.2 使用 ETCD 集中配置
 - 16.2.1 添加集中配置服务
 - 16.2.2 在 ETCD 的配置
 - 16.2.2.1 apusic.conf 的配置
 - 16.2.2.2 数据源的配置
 - 16.2.2.3 日志的配置
 - 16.2.2.4 JVM 的配置
- 17 配置 Session 存储
 - 17.1 配置本地内存存储
 - 17.2 配置本地文件存储
 - 17.3 配置数据库存储
- 18 集群方案
 - 18.1 Session 组播同步
 - 18.1.1 添加集群配置
 - 18.1.2 部署应用和设置 jvmRoute
 - 18.1.3 配置 Apache
 - 18.1.4 验证集群
 - 18.2 Session 集中式存储
 - 18.2.1 构建 Redis 插件
 - 18.2.1.1 系统构建
 - 18.2.2 添加 Redis 配置
- 19 与 Springboot 集成使用

- 19.1 前言
 - 19.1.1 面向对象
- 19.2 产品简介
 - 19.2.1 概述
 - 19.2.2 JavaEE规范
 - 19.2.3 平台环境
 - 19.2.4 系统环境
 - 19.2.5 术语
- 19.3 产品清单
 - 19.3.1 产品清单列表
 - 19.3.2 产品清单说明
- 19.4 安装
 - 19.4.1 基于源码方式构建
 - 19.4.1.1 按模块导入
 - 19.4.1.1.1 前置条件
 - 19.4.1.1.2 安装资源依赖
 - 19.4.1.1.3 引入资源依赖
 - 19.4.1.1.4 验证安装
 - 19.4.1.2 导入单个jar
 - 19.4.1.2.1 前置条件
 - 19.4.1.2.2 安装资源依赖
 - 19.4.1.2.3 引入资源依赖
 - 19.4.1.2.4 验证安装
 - 19.4.2 基于可执行jar替换
 - 19.4.2.1 前置条件
 - 19.4.2.2 删除弃用依赖
 - 19.4.2.3 引入资源依赖
- 19.5 授权认证
 - 19.5.1 项目目录启动（未打包状态）
 - 19.5.2 参数设置启动（未打包状态）
 - 19.5.3 项目目录启动（可执行jar包状态）
 - 19.5.4 参数设置启动（可执行jar包状态）
 - 19.5.5 集中授权
 - 19.5.6 特征码获取

- 19.6 容器功能说明
 - 19.6.1 安全管理
 - 19.6.1.1 HTTPS
 - 19.6.1.1.1 简介
 - 19.6.1.1.2 配置方法
 - 19.6.1.2 国密配置
 - 19.6.1.2.1 简介
 - 19.6.1.2.2 配置方法
 - 19.6.2 国际化支持
 - 19.6.2.1 简介
 - 19.6.2.2 配置方法
 - 19.6.3 日志管理
 - 19.6.3.1 应用日志
 - 19.6.3.1.1 简介
 - 19.6.3.1.2 使用方法
 - 19.6.3.2 访问日志
 - 19.6.3.2.1 简介
 - 19.6.3.2.2 使用方法
 - 19.6.4 监控支持
 - 19.6.4.1 简介
 - 19.6.4.2 使用方法
- 19.7 示例操作
 - 19.7.1 环境准备
 - 19.7.2 应用工程
 - 19.7.3 pom文件
 - 19.7.4 构建应用包
 - 19.7.5 运行应用
 - 19.7.6 访问应用
 - 19.7.7 以调试模式运行
- 20 附录
 - 20.1 附录A Web通用配置
 - 20.2 附录B 嵌入式部署日志参数
 - 20.3 附录C actuator详细说明
 - 20.3.1 配置

- 20.3.2 端口
 - 20.3.2.1 beans
 - 20.3.2.2 health
 - 20.3.2.3 conditions
 - 20.3.2.4 configprops
 - 20.3.2.5 env
 - 20.3.2.6 loggers
 - 20.3.2.6.1 设置日志级别
 - 20.3.2.7 threaddump
 - 20.3.2.8 metrics
 - 20.3.2.8.1 所有指标相关说明
 - 20.3.2.9 scheduledtasks
 - 20.3.2.10 mappings
- 20.4 附录D relaxed-path-chars和relaxed-query-chars配置示例
 - 20.4.1 Linux
 - 20.4.1.1 环境变量
 - 20.4.1.2 JVM参数
 - 20.4.2 Spring boot配置文件
 - 20.4.2.1 application.properties
 - 20.4.2.2 application.yaml

1 前言

本文档是金蝶Apsic应用服务器V10.0.10标准版的用户手册，详细介绍金蝶Apsic应用服务器V10.0.10标准版的基本安装以及配置，提供基本操作方法以及各功能的具体步骤。

1.1 面向对象

本手册主要面向对象为使用金蝶Apsic应用服务器V10.0.10 进行应用开发的开发人员，生成环境的系统管理员，应用发布人员，技术运维人员等。具备以下技能可能会更好理解和使用金蝶Apsic应用服务器V10.0.10用户手册内容：

- 熟悉Linux常用命令
- 基本的系统管理任务
- 安装和管理软件

2 版本变更说明

本手册根据产品实际更新情况同步更新，最新版本将会包括历史版本内容或作出对应的修改说明。

日期	手册版本	适用产品	更新说明
2024年9月	V1010A01F02	AAS标准版 V10.0.10	调整文中显示格式等问题
2023年12月	V1010A01F01	AAS标准版 V10.0.10	更新许可证授权码获取方式

3 产品简介

3.1 概述

金蝶 Apusic 应用服务器V10.0.10标准版是一款支持JavaEE规范的web应用服务器，支持全新Servlet和JSP规范。金蝶 Apusic 应用服务器V10.0.10标准版技术先进、性能稳定、轻量简便，快速和支持按需而变的业务需求，是应用系统高效开发的首选产品。

图1：金蝶Apusic应用服务器V10.0.10标准版架构图



3.2 平台环境

金蝶Apusic应用服务器V10.0.10标准版平台环境。

表1：金蝶Apusic应用服务器V10.0.10标准版平台环境

操作系统	说明
Windows	Windows7、Windows10、Windows Server
Linux	CentOS、Ubuntu、SUSE、Redhat
国产平台	银河麒麟系列、中标麒麟系列、普华、中科红旗、深度等

3.3 系统环境

金蝶Apusic应用服务器V10.0.10标准版系统环境。

表2：金蝶Apusic应用服务器V10.0.10标准版系统环境

系统组件	系统要求
Java环境	JDK11及以上
内存	至少需要 512MB 的内存，建议1024MB以上
磁盘空间	至少需要 1024MB 的磁盘空间，建议10G以上

3.4 JavaEE规范

金蝶Apusic应用服务器V10.0.10标准版产品规范。

表3：金蝶Apusic应用服务器V10.0.10标准版产品规范

规范/API	版本
Servlet	6.0
JSP	3.1
Expression Language	5.0
WebSocket	2.1

3.5 术语

AAS：金蝶Apusic应用服务器

AE：金蝶Apusic应用服务器标准版产品标识

APUSIC_HOME：金蝶Apusic应用服务器安装目录

4 产品清单

金蝶Apusic应用服务器V10.0.10标准版产品包说明，通常情况下打包成 `AAS-[aas_version]-[sp_version]-AE-[date].zip` 或 `AAS-[aas_version]-[sp_version]-AE-[date].tar.gz`。详看产品清单说明。

表4：金蝶Apusic应用服务器V10.0.10标准版产品包

产品包	说明
AAS-V10.0.10-SP10-AE-20251128.zip/AAS-V10.0.10-SP10-AE-20251128.tar.gz	金蝶Apusic应用服务器V10.0.10标准版安装包

产品包说明:

金蝶Apusic应用服务器V10.0.10标准版产品包说明。

表5：金蝶Apusic应用服务器V10.0.10标准版产品包说明

文件	说明
bin	应用服务器启动、停止等脚本所在的目录
conf	应用服务器配置文件所在的目录
lib	应用服务器JAR文件所在的目录
logs	应用服务器日志文件默认目录
plugins	应用服务器插件所在的目录
webapps	部署应用程序包的目录
webapps-javaee	部署支持JakartaEE9.1及以下版本的应用程序包的目录
temp	应用服务器临时文件存放的目录
work	存放JSP编译后产生的class文件
license.xml	产品包的授权文件

5 许可证授权

金蝶Apusic应用服务器需要有对应的许可证才能正常使用，通常情况下，金蝶天燕会根据用户购买的产品版本配套对应的许可证。如果产品将要过期或者已经过期，可向金蝶天燕对接人员申请许可证，与此同时将产品的特征码发给对接人员。

许可证位置为 `${APUSIC_HOME}/license.xml`。

5.1 标准版获取特征码

1. 可以运行bin目录下命令 `apusic` 获取，类似格式:

```
apusic -ac [ethname or ip]
```

其中 `[ethname or ip]` 表示可以输入ip地址或者mac地址的名称，如可以输入常用ip地址172.24.1.116作为参数获取软件特征码(适用windows、linux):

```
apusic -ac 172.24.1.116
```

输出格式如下，Auth Code=软件特征码:

```
Auth Code=SZTY263537626
```

```
le
[root@linux-2-42 bin]# ./apusic.sh -ac 172.24.2.42
Using APUSIC_BASE: /opt/testz/aams/AAMS-10.0.10
Using APUSIC_HOME: /opt/testz/aams/AAMS-10.0.10
Using APUSIC_TMPDIR: /opt/testz/aams/AAMS-10.0.10/temp
Using JRE_HOME: /opt/java/jdk-17.0.2
Using CLASSPATH: /opt/testz/aams/AAMS-10.0.10/bin/bootstrap.jar:/opt/testz/aams/AAMS-10.0.10/bin/aas-juli.jar
Using APUSIC_PID: /opt/testz/aams/AAMS-10.0.10/bin/pid
Using APUSIC_OPTS: -Djava.security.egd=file:/dev/urandom
Auth Code=SZTY1974666171
```

2. 也可以输入网卡名称获取特征码(适用linux，可以通过ifconfig获取网卡名称)

```
apusic -ac ens192
```

输出格式如下，Auth Code=软件特征码:

```
Auth Code=SZTY1640356587
```

5.2 嵌入式部署获取特征码

在与springboot整合的方式下，没有获取特征码的 `apusic` 命令，需要依赖用户的应用进行获取。

如果已经有临时的授权文件可以启动应用，则先把授权文件移除，然后在启动命令中增加JVM参数 `apusic-ac`，取值为ip地址或者网卡名称，类似如下,eth3为测试机器的网卡名称，需要更加实际的网络信息修改：

```
java -jar -Dapusic-ac=eth3 boot-quickstart-1.0.jar
```

由于没有授权文件，应用启动会停止，在停止前会打印特征码信息，类似如下，Auth Code=特征码内容：

```
Auth Code=SZTY263537626  
license is invalid.
```

6 安装和运行

下面介绍金蝶 Apusic 应用服务器V10.0.10标准版作为一个完整的产品进行使用，`${APUSIC_HOME}` 表示产品安装的根目录。

6.1 安装

准备安装包 `AAS-V10.0.10-SP10-AE-20251128.zip` 或 `AAS-V10.0.10-SP10-AE-20251128.tar.gz`。解压产品包到 `${APUSIC_HOME}` 目录下，如下图所示：

```
[root@linux-2-42 aams]# cd AAMS-10.0.10/
[root@linux-2-42 AAMS-10.0.10]# ls
bin  conf  lib  license.xml  logs  temp  webapps  webapps-javaee  work
```

6.2 卸载

将 `${APUSIC_HOME}` 目录下所有文件删除即可完成卸载操作。

6.3 运行

6.3.1 在Linux环境下

将应用程序打包文件，如`examples.war`，放置 `${APUSIC_HOME}/webapps/` 目录下；

1、进入安装目录 `${APUSIC_HOME}/bin`，执行 `apusic.sh run`。

```
[root@linux-2-42 bin]# ./apusic.sh run
Using APUSIC_BASE: /opt/testz/aams/AAMS-10.0.10
Using APUSIC_HOME: /opt/testz/aams/AAMS-10.0.10
Using APUSIC_TMPDIR: /opt/testz/aams/AAMS-10.0.10/temp
Using JRE_HOME: /opt/java/jdk-17.0.2
Using CLASSPATH: /opt/testz/aams/AAMS-10.0.10/bin/bootstrap.jar:/opt/testz/aams/AAMS-10.0.10/bin/aas-juli.jar
Using APUSIC_PID: /opt/testz/aams/AAMS-10.0.10/bin/pid
Using APUSIC_OPTS: -Djava.security.egd=file:/dev/urandom
22-Jan-2026 17:46:17.059 INFO [ApusicConsoleHandler] com.apusic.ams.core.AprLifecycleListener.lifecycleEvent The Apusic AAS Native
library which allows using OpenSSL was not found on the java.library.path: [/usr/java/packages/lib:/usr/lib64:/lib64:/lib:/usr/li
b]
22-Jan-2026 17:46:17.162 INFO [ApusicConsoleHandler] com.apusic.juli.logging.LogPrintStream.println Apusic Application Server Ente
rprise Edition 9.0
22-Jan-2026 17:46:17.164 INFO [ApusicConsoleHandler] com.apusic.juli.logging.LogPrintStream.println Licensed to 金蝶天燕2026临时授
权
22-Jan-2026 17:46:17.166 INFO [ApusicConsoleHandler] com.apusic.juli.logging.LogPrintStream.println Connctions limited to 10
22-Jan-2026 17:46:17.167 INFO [ApusicConsoleHandler] com.apusic.juli.logging.LogPrintStream.println Licensed from 2025-12-10 to 20
26-12-31
```

2、进入安装目录 `${APUSIC_HOME}/bin`，执行 `startapusic.sh`。

3、如果需要使用调试模式 (debug)，可在启动脚本，`apusic.sh` 中设置 `JPDA_OPTS`，如：

```
JPDA_OPTS="-  
agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000"
```

执行 `apusic.sh jpda start` 。

6.3.2 在 Windows环境下

- 1、将应用程序打包文件，如examples.war，放置 `${APUSIC_HOME}\webapps\` 目录下；进入安装目录 `${APUSIC_HOME}\bin` ，执行 `apusic.bat run` 。
- 2、进入安装目录 `${APUSIC_HOME}\bin` ，执行 `startapusic.bat` 。
- 3、如果需要使用调试模式 (debug) ，可在启动脚本， `apusic.bat` 中设置 `JPDA_OPTS` ，如：

```
JPDA_OPTS=-  
agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8000
```

执行 `apusic.bat jpda start` 。

如果 Windows 启动出现乱码，则可通过修改 `conf/logging.properties` 文件，把控制台日志编码设置为 `GBK` 。

```

-----1-----2-----3-----4-----5-----6-----7-----8-----9-----0-----1-----2-----
 7 # Handler specific properties.
 8 # Describes specific configuration info for Handlers.
 9 #####
10
11 1apusic.com.apusic.juli.AsyncFileHandler.level = FINE
12 1apusic.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
13 1apusic.com.apusic.juli.AsyncFileHandler.prefix = apusic.
14 1apusic.com.apusic.juli.AsyncFileHandler.maxDays = 90
15 1apusic.com.apusic.juli.AsyncFileHandler.encoding = UTF-8
16
17 2localhost.com.apusic.juli.AsyncFileHandler.level = FINE
18 2localhost.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
19 2localhost.com.apusic.juli.AsyncFileHandler.prefix = localhost.
20 2localhost.com.apusic.juli.AsyncFileHandler.maxDays = 90
21 2localhost.com.apusic.juli.AsyncFileHandler.encoding = UTF-8
22
23 3manager.com.apusic.juli.AsyncFileHandler.level = FINE
24 3manager.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
25 3manager.com.apusic.juli.AsyncFileHandler.prefix = manager.
26 3manager.com.apusic.juli.AsyncFileHandler.maxDays = 90
27 3manager.com.apusic.juli.AsyncFileHandler.encoding = UTF-8
28
29 4host-manager.com.apusic.juli.AsyncFileHandler.level = FINE
30 4host-manager.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
31 4host-manager.com.apusic.juli.AsyncFileHandler.prefix = host-manager.
32 4host-manager.com.apusic.juli.AsyncFileHandler.maxDays = 90
33 4host-manager.com.apusic.juli.AsyncFileHandler.encoding = UTF-8
34
35 java.util.logging.ConsoleHandler.level = FINE
36 java.util.logging.ConsoleHandler.formatter = com.apusic.juli.OneLineFormatter
37 java.util.logging.ConsoleHandler.encoding = GBK
38
39
40 #####
41 # Facility specific properties.
42 # Provides extra control for each logger.
43 #####
44
45 com.apusic.ams.core.ContainerBase.[Apusic].[localhost].level = INFO
46 com.apusic.ams.core.ContainerBase.[Apusic].[localhost].handlers = 2localhost.com.apusic.juli.AsyncFileHandler
47
48 com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/manager].level = INFO
49 com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/manager].handlers = 3manager.com.apusic.juli.AsyncFileHandler
50
51 com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/host-manager].level = INFO
52 com.apusic.ams.core.ContainerBase.[Apusic].[localhost].[/host-manager].handlers = 4host-manager.com.apusic.juli.AsyncFileHandler
53
54 "

```

启动完成后，在浏览器中输入 `http://<IP>:6888` 进行访问，如果是本机，则输入 `<http://localhost:6888>`，则会出现如下的界面表示成功：



6.4 停止

6.4.1 在Linux环境下

1、Ctrl + C

前端启动状态下，可以直接Ctrl + C退出，停止AAS。

2、stopapusic.sh

进入安装目录 `${APUSIC_HOME}/bin`，执行 `stopapusic.sh`。

3、kill进程

可以通过kill [进程号] 方式停止。

6.4.2 在 Windows环境下

1、Ctrl + C

前端启动状态下，可以直接Ctrl + C退出，停止AAS。

2、stopapusic.bat

进入安装目录 `${APUSIC_HOME}\aas\bin`，执行 `stopapusic.bat`。

3、停止进程

可以打开任务管理器，找到对应进程，停止运行。

7 部署应用

应用服务器支持部署JakartaEE10及以下版本相关Web规范的应用，根据不同JakartaEE版本将应用文件拷贝至对应的目录。

部署JakartaEE10相关Web规范的应用，只需要把应用拷贝到 `webapps` 目录下即可自动部署。该版本不支持目录部署。

```
[root@linux-2-42 bin]# cd /opt/testz/aams/AAMS-10.0.10/webapps/  
[root@linux-2-42 webapps]# ls  
examples  examples.war  ROOT
```

部署JakartaEE9.0及以下版本相关Web规范的应用，只需要把应用拷贝到 `webapps-javaee` 目录下即可自动部署。

```
[root@linux-2-42 webapps]# cd /opt/testz/aams/AAMS-10.0.10/webapps-javaee/  
[root@linux-2-42 webapps-javaee]# ls  
test1.war
```

8 取消部署应用

将应用程序文件从 `webapps` 或 `webapps-javaee` 目录下全部删除即可取消部署应用程序。

9 设置 JVM 内存

- Windows

在 `${APUSIC_HOME}/bin` 目录下创建 `setenv.bat` 文件, 添加 `MEMORY_JVMOPTS_DOMAIN` 进行设置。

```
set "MEMORY_JVMOPTS_DOMAIN=-Xmx2g -Xms2g -XX:NewRatio=2 -  
XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=256m"
```

- Linux

在 `${APUSIC_HOME}/bin` 目录下创建 `setenv.sh` 文件, 并添加 `**可运行权限**`, 在文件中添加 `MEMORY_JVMOPTS_DOMAIN` 进行设置。

```
MEMORY_JVMOPTS_DOMAIN="-Xmx2g -Xms2g -XX:NewRatio=2 -  
XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=256m"
```

10 配置文件说明

10.1 apusic.conf 文件

apusic.conf 是金蝶 Apusic 应用服务器的重要配置文件，可通过该文件设置端口、线程池大小、设置部署应用路径、性能调优等等。

默认配置如：

```
<config>
  <listeners>
    <listener name="aprLifecycleListener"
className="com.apusic.ams.core.AprLifecycleListener" />
    <listener name="jreMemoryLeakPreventionListener"
className="com.apusic.ams.core.JreMemoryLeakPreventionListener"/>
    <listener name="globalResourcesLifecycleListener"
className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener"/>
    <listener name="threadLocalLeakPreventionListener"
className="com.apusic.ams.core.ThreadLocalLeakPreventionListener"/>
  </listeners>
  <resources>
    <resource name="userDatabase" auth="Container"
type="com.apusic.ams.UserDatabase"
description="User database that can be updated and
saved"
factory="com.apusic.ams.users.MemoryUserDatabaseFactory"
pathname="conf/aas-users.xml" />
  </resources>
  <executors>
    <executor name="http-thread-pool" namePrefix="http-exec-"
maxThreads="200" minSpareThreads="25" maxQueueSize="5000"/>
  </executors>
  <endpoints>
    <endpoint name="ams-http"
```

```

        port="6888"
        protocol="HTTP/1.1"
        connectionTimeout="3000"
        redirectPort="6443"
        executor="http-thread-pool"
        maxConnections="10000"
        enableLookups="false"
        acceptCount="100"
        maxPostSize="104857600"
        maxHttpRequestSize="8192"
        compression="on"
        disableUploadTimeout="true"
        compressionMinSize="2048"

compressibleMimeType="text/html,text/plain,text/css,application/javascript-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image/png,image/gif,audio/mpeg,video/mp4"
        URIEncoding="utf-8"
        processorCache="20000"
        tcpNoDelay="true"
        keepAliveTimeout="15000"
        maxKeepAliveRequests="100">
</endpoint>
<endpoint name="ams-jmx"
        port="6868"
        auth="false"
        protocol="JMXRMI/2.0">
</endpoint>
</endpoints>
<services>
    <service name="ams-service" defaultHost="localhost">
        <realm className="com.apusic.ams.realm.LockOutRealm">
            <realm resourceName="userDatabase"
className="com.apusic.ams.realm.UserDatabaseRealm"/>
        </realm>
        <host name="localhost" appBase="webapps"

```

```

unpackWARs="true" autoDeploy="true">
    </host>
  </service>
</services>
<servers>
  <server name="ams-server" services="ams-service"
endpoints="ams-http" executors="http-thread-pool"/>
</servers>
<containers>
  <container name="container" port="-1" shutdown="shutdown"
servers="ams-server" resources="userDatabase"
listeners="aprLifecycleListener,jreMemoryLeakPreventionListener,globalF
  </containers>
</config>

```

10.1.1 listeners 配置 (监听器)

应用服务器的 `listener` (监听器) 是用于监听Web应用中某些对象的状态变化的组件，当监听范围的对象状态发生变化时，服务器会自动调用监听器对象中的方法作出相应的响应处理。

表6: 金蝶Apusic应用服务器V10.0.10标准版listeners配置说明

配置参数	说明	类名
<code>aprLifecycleListener</code>	使用 APR 进行接收客户端请求，在应用服务器初始化之前、初始化之前尝试初始化 APR 库，成功则使用 APR 接受处理客户端请求，应用服务器销毁之后，该监听器会做	<code>com.apusic.ams.core.AprLifecycleListener</code>

	APR 的清理工作	
jreMemoryLeakPreventionListener	防止 JRE 内存泄露	com.apusic.ams.core.JreMemoryLeakPreventionListener
globalResourcesLifecycleListener	初始化 JNDI 资源的 MBean	com.apusic.ams.mbeans.GlobalResourcesLifecycleListener
threadLocalLeakPreventionListener	防止 ThreadLocal 对象带来的内存泄漏	com.apusic.ams.core.ThreadLocalLeakPreventionListener

10.1.2 resources 配置 (用户权限)

应用服务器的用户权限配置主要涉及 `resources` 和用户信息的配置。可在 `apusic.conf` 配置 `resources`，指定资源的名称、认证方式、类型等属性；在 `aas-users.xml` 中配置用户信息，如用户名、密码、角色信息等；如果应用程序需要使用用户权限，通过 `java:comp/env` 前缀加上资源名称来引用配置的资源。

`userDatabase`：用户权限配置。

表7：金蝶Apusic应用服务器V10.0.10标准版resources配置说明

配置参数	说明	值类型	默认值
<code>name</code>	资源名称, 必须唯一。 如"userDatabase"表示配置用户权限	字符	userDatabase
<code>auth</code>	授权范围	字符	Container
<code>type</code>	类型	字符	com.apusic.ams.UserDatabase
<code>description</code>	描述	字符	User database that can be updated and saved
<code>factory</code>	资源工厂	字符	com.apusic.ams.users.MemoryUserDatabaseFactory

pathname	路径名称	字符	conf/aas-users.xml
----------	------	----	--------------------

10.1.3 executors 配置 (线程池)

应用服务器的 `executors` 一个用于定义和管理线程池的配置元素。线程池是执行任务的线程集合，它允许Web应用以并发方式处理多个请求，从而提高性能和响应速度。

```
<executor name="http-thread-pool" namePrefix="http-exec-"
maxThreads="200" minSpareThreads="25" maxQueueSize="5000" />
```

表8: 金蝶Apusic应用服务器V10.0.10标准版executors配置说明

配置参数	说明	值类型	默认值
name	线程池名称, 供配置文件其他地方引用, 必须唯一	字符	http-thread-pool
namePrefix	线程池里面的线程的前缀	字符	http-exec-
maxThreads	最大线程数	整型	200
minSpareThreads	最小空闲线程数	整型	25
maxIdleTime	当线程数大于 minSpareThreads 时, 空闲线程销毁前最大空闲时间, 单位毫秒	整型	60000
maxQueueSize	线程池队列最大任务数, 超过后将被拒绝	整型	Integer.MAX_VALUE

10.1.4 endpoints 配置 (网络 IO 和协议)

应用服务器的 `endpoints` 是连接器组件中的关键部分, 主要负责对外实现TCP/IP协议, 提供基础的网络I/O服务。

```
<endpoint name="ams-http" port="6888" protocol="HTTP/1.1"
connectionTimeout="20000" redirectPort="6443"
    executor="http-thread-pool" maxConnections="10000"
enableLookups="false" acceptCount="100" maxPostSize="104857600"
    maxHttpRequestSize="8192" compression="on"
disableUploadTimeout="true" compressionMinSize="2048"

compressableMimeType="text/html,text/plain,text/css,application/javascr
```

```
font-ttf,application/x-font-otf"
    URLEncodering="utf-8" processorCache="20000" tcpNoDelay="true"
keepAliveTimeout="15000" maxKeepAliveRequests="100">
    <SSLHostConfig truststoreType="JKS"
truststorePassword="changeit"
    truststoreFile="conf/cert/truststore.jks"
certificateVerification="required">
    <Certificate certificateKeystoreFile="keystore.jks"
certificateKeystoreType="JKS"
    certificateKeystorePassword="changeit" />
    </SSLHostConfig>
</endpoint>
```

`endpoint` 代表一个端口，接听服务器上一特定 TCP 端口，接收请求。

表9：金蝶Apusic应用服务器V10.0.10标准版endpoints配置说明

配置参数	说明	值类型	默认值
port	监听端口	整型	6888
protocol	处理协议，可选的有 HTTP/1.1、AJP/1.3	字符	HTTP1.1
connectionTimeout	连接超时时间, 读取 HTTP 请求行超时时间。单位毫秒	数字	3000
redirectPort	如果是 https, 重定向端口	整型	6443
executor	使用的线程池	字符	http-thread-pool
maxConnections	最大并发数	整型	10000
enableLookups	是否启用 DNS 查询, 如果设置 true, 则 request.getRemoteHost() 时, 将调用 DNS 查询, 否则只返回 IP 值	布尔值	false
acceptCount	backlog 大小, 达到 maxConnections 后, 最大队列数, 超出后, 请求将被拒绝	整型	100
maxPostSize	FORM URL 方式的最大提交数据。单位字节	数字	104857600

maxHTTPHeaderSize	请求或响应的最大 http header 大小。单位字节	数字	8192 (8KB)
compression	是否启用压缩, 可选值为: off,on,force	字符	on
compressionMinSize	压缩文件阈值, 小于该值不会进行压缩	整型	2048
compressableMimeType	逗号隔开的可压缩的 MIME 类型	字符	text/html,text/xml, text/plain,text/css,text/javascript,application/javascript,application/json,application/xml
connectionUploadTimeout	数据上传超时时间。仅当 disableUploadTimeout=false 时生效。单位毫秒	数字	
disableUploadTimeout	是否禁用上传超时	布尔值	true
URIEncoding	配置 url 中参数编码	字符	UTF-8
processorCache	缓存 Processor 的最大数量	整型	20000
tcpNoDelay	是否禁用了 Nagle 算法, 允许小包的发送	布尔值	true
keepAliveTimeout	keepalive 的超时时间。单位毫秒	数字	与 connectionTimeout 一致
maxKeepAliveRequests	保持 keepalive 的连接的个数, 超过就会关闭连接。-1 不限制	整型	15000
address	服务器监听地址	IP地址	0.0.0.0 或 ::
maxCookieCount	最大 Cookie 数量	整型	200
relaxedPathChars	允许请求行中的特殊字符	字符	
relaxedQueryChars	允许请求行中参数的特殊字符	字符	
useAsyncIO	是否使用异步 IO API	布尔值	true
useSendfile	是否使用 sendfile 功能。使用 sendfile 后将不压缩响应	布尔值	true
SSLEnabled	是否允许 SSL 通信	布尔值	false

10.1.4.1 SSLHostConfig 元素

当 Endpoint 允许 SSL 通信时，需要添加 `SSLHostConfig` 子元素，设置 SSL 通信相关的证书等属性。

表10：金蝶Apusic应用服务器V10.0.10标准版 SSLHostConfig配置说明

配置参数	说明	值类型	默认值
protocols	允许与客户端通信的协议名字。可以是以下的组合:SSLv2Hello/SSLv3/TLSv1/TLSv1.1/TLSv1.2/TLSv1.3/SMv1.1/all	字符	all
sslProtocol	服务器使用的 SSL 协议，用于创建 SSLContext	字符	TLS
truststoreFile	信任库文件，用于验证客户端证书	字符	
truststoreType	信任库类型	字符	JKS
truststorePassword	信任库密码	字符	
certificateVerification	证书验证，如果设置为： required: 客户端必须返回有效证书。 optional: 希望客户端返回证书，若客户端不发证书，也不报错。 none 不需要客户端证书，除非客户端请求 "CLIENT-CERT" 限制的资源	字符	none
ciphers	允许使用的密码套件	字符	HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!kRSA

10.1.4.2 Certificate 元素

服务器端证书设置元素。

表11：金蝶Apusic应用服务器V10.0.10标准版 Certificate配置说明

配置参数	说明	值类型	默认值
certificateFile	PEM 编码的证书文件	字符	
certificateChainFile	PEM 编码的证书链文件	字符	
certificateKeyFile	PEM 编码的密钥文件	字符	
certificateKeyPassword	密钥文件密码	字符	

certificateKeyAlias	服务器使用的密码和证书在密钥库的别名	字符	
certificateKeystoreFile	密钥库文件	字符	
certificateKeystorePassword	密钥库文件密码	字符	
certificateKeystoreType	密钥库文件类型	字符	JKS
type	证书类型，用于确认密码套件和证书是匹配的，必须是为 UNDEFINED,RSA,DSS 和 EC 之一。如果 SSLHostConfig 只有一个 Certificate 元素,type 属性不是必须的。若有多个 Certificate 元素，则必须指定 type 属性	字符	UNDEFINED

10.1.5 services 配置 (虚拟主机和应用)

应用服务器的 `service` 组件是 `server` 组件的子元素，代表一组共享Engine的Connector，负责管理它们的配置和生命周期，并接收和处理客户端请求。

```
<service name="ams-service" defaultHost="localhost">
  <realm className="com.apusic.ams.realm.LockOutRealm">
    <realm resourceName="userDatabase"
className="com.apusic.ams.realm.UserDatabaseRealm" />
  </realm>
  <host name="localhost" appBase="webapps" unpackWARs="true"
autoDeploy="true">
    <application path="/test-webapp"
docBase="D:\\work_test\\test-webapp" useHttpOnly="true" />
  </host>
</service>
```

表12: 金蝶Apusic应用服务器V10.0.10标准版 services配置说明

配置参数	说明	值类型	默认值
name	服务名称，必须全局唯一	字符	ams-service
defaultHost	默认虚拟主机名称	字符	localhost
jvmRoute	在负载均衡使用场景中，Session 粘滞时使用的标签，在集群节点中必须唯一。此标签将后缀在	字符	

Session ID 中，此后同一会话的请求将被同一 AAS 实例处理。		
--------------------------------------	--	--

10.1.5.1 Realm

Realm 元素代表这一个存储了用户名，密码及角色的库，用于认证及授权处理。

表13：金蝶Apusic应用服务器V10.0.10标准版realm配置说明

配置参数	说明	值类型	默认值
className	Java 类名，此类必须实现 com.apusic.ams.Realm	字符	

AAS 提供的默认实现有：

1.DataSource Database Realm: `com.apusic.ams.realm.DataSourceRealm`

DataSource Database Realm 通过 JNDI 命名的 JDBC 数据源连接到关系型数据库中，获取用户名，密码及对应的角色。

```
<Realm className="com.apusic.ams.realm.DataSourceRealm"
  dataSourceName="jdbc/authority"
  userTable="users" userNameCol="user_name" userCredCol="user_pass"
  userRoleTable="user_roles" roleNameCol="role_name"/>
```

表14：金蝶Apusic应用服务器V10.0.10标准版DataSource Database Realm配置说明

配置参数	说明	值类型	默认值
dataSourceName	JDBC 数据源的 JNDI 名字	字符	
userTable	用户表名，表中必须有 userNameCol 和 userCredCol 属性指定的列	字符	
userNameCol	用户名列	字符	
userCredCol	用户凭据列（例：密码）	字符	
userRoleTable	用户角色表，表中必须有 userNameCol 和 roleNameCol 属性指定的列	字符	
roleNameCol	角色列	字符	

2.UserDatabase Realm: `com.apusic.ams.realm.UserDatabaseRealm`

UserDatabase Realm 是可通过 JDNI 获取的实现了 UserDatabase 的认证域。

表15：金蝶Apusic应用服务器V10.0.10标准版UserDatabase Realm配置说明

配置参数	说明	值类型	默认值
resourceName	UserDatabase 资源的名字	字符	

****3.Combined Realm**:** `\com.apusic.ams.realm.CombinedRealm``

Combined Realm 组合域，可以合并多个域，通过一个或多个子域认证用户。

Combined Realm 子元素定义了它的子域，按子域定义的顺序认证用户，用户的角色将有第一个认证通过的域指定。

4.LockOut Realm: `com.apusic.ams.realm.LockOutRealm`

`LockOut Realm` 继承了组合域，并提供了锁定机制 - 当在某一时段内，用户验证失败多次后，将用户锁定。

表16：金蝶Apusic应用服务器V10.0.10标准版LockOut Realm配置说明

配置参数	说明	值类型	默认值
cacheSize	缓存验证失败用户数	整型	1000
failureCount	锁定前验证失败次数	整型	5
lockOutTime	用户锁死时间，在锁死时间内，验证失败将重置计时器，验证成功不起效。单位为秒	时间	300

10.1.5.2 host

`host` 元素代表一个虚拟主机，虚拟主机可以指定网络主机名（例如：www.mysite.com），[客户端可通过主机名访问服务器](#)。多个 `host` 元素可以嵌套在 `service` 元素中，必须有唯一一个 `host` 的 `name` 属性与 `service` 的 `defaultHost` 属性一致。

表17：金蝶Apusic应用服务器V10.0.10标准版host配置说明

配置参数	说明	值类型	默认值
name	虚拟名称，必须全局唯一，通常为在 DNS 注册的网络名	字符	
appBase	应用部署基路径	字符	webapps
unpackWARs	是否解压 war 进行部署	布尔值	true
autoDeploy	是否自动部署，如果为 true，AAS 将周期性的检查 appBase 目录，部署新发现的应用，重现部署更	布尔值	true

新过应用

10.1.5.3 application

应用服务器的 `application` 元素代表这一个应用，每个应用被打包成一个 Web Application Archive(WAR) 包或者解压缩的一个目录。`host` 元素下可以添加多个 `application` 元素，每个 `application` 的名字在 `host` 下必须唯一。

```
<application path="/test-webapp"
docBase="D:\\work_test\\test-webapp" useHttpOnly="true" />
```

表18: 金蝶Apusic应用服务器V10.0.10标准版application配置说明

配置参数	说明	值类型	默认值
path	应用上下文，访问路径。同一虚拟主机里，必须唯一。如果 docBase 在 host 的 appBase 内，则此属性必须不能设置，不然同一应用可能会被部署两次。	字符	
docBase	应用的根目录（应用以目录形式打包），或者应用（WAR 包）的路径，可以是绝对路径或者相对与 host 元素的 appBase 属性的相对路径	字符	
useHttpOnly	设置为 true 表示防止客户端脚本读取 JSESSIONID，防止 CSRF/XSS 恶意攻击	布尔值	true
cookie	是否允许 cookies 传递 Session ID	布尔值	true
crossContext	若设置为 true, 则 ServletContext.getContext() 能获取同一虚拟主机下其他应用的 ServletContext. 若设置为 false, 则 ServletContext.getContext() 返回 null.	布尔值	false
logEffectiveWebXml	若为 true, 服务器在应用启动时输出生效的 Web.xml 日志（INFO）等级，生效的 web.xml 包括默认 web.xml，应用提供的 web.xml、web-fragment.xml 和 annotations 解析结果	布尔值	false
override	若设置为 true，则忽略所有全局和 host 的默认 context 的设置	布尔值	true

reloadable	如果设置为 true, 则监控 /WEB-INF/classes/ 和 /WEB-INF/lib/, 若文件有变化, 则重新部署应用	布尔值	true
sessionCookieDomain	设置此应用中 Session cookies 的 domain 值	字符	
sessionCookieName	设置 Session cookies 的名字	字符	JSESSIONID
swallowOutput	若设置为 true, 应用中输出到 System.out 和 System.err 的内容将被重定向到应用的 logger 中	布尔值	false

10.1.6 servers 配置 (引用其他功能)

应用服务器的 `server` 组件负责定义全局服务器配置、管理服务生命周期以及监听服务器关闭命令

```
<servers>
  <server name="ams-server" services="ams-service" endpoints="ams-
http" executors="http-thread-pool" />
</servers>
```

表19: 金蝶Apusic应用服务器V10.0.10标准版servers配置说明

配置参数	说明	值类型	默认值
name	定义的server名称, 全局唯一	字符	
services	引用的服务, 仅可添加一个服务	字符	
endpoints	引用的网络和协议配置, 多个用逗号隔开	字符	
executors	引用的线程池, 多个用逗号隔开, 注意: 该 server 里面的 endpoints 里面引用的线程池也必须在这里引用, 如上面的 http-thread-pool	字符	

10.1.7 containers 配置 (监听器等)

应用服务器的 `containers` 用于接收客户端请求并返回响应。

```
<container name="container" port="-1" shutdown="shutdown"
servers="ams-server" resources="userDatabase"
```

```
listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventor"
/>
```

表20: 金蝶Apusic应用服务器V10.0.10标准版containers配置说明

配置参数	说明	值类型	默认值
name	container名称, 全局唯一	字符	
port	侦听端口, 用于关闭应用服务器, 不使用可设置为 -1, 建议不使用	整型	
servers	该应用服务器实例引用的服务器, 多个可用逗号隔开, 建议使用单个	字符	
resources	该应用服务器实例引用的资源, 比如用户名密码等, 多个可用逗号隔开	字符	
listeners	该应用服务器实例引用的监听器, 多个可用逗号隔开。引用的是文件开头配置的监听器	字符	

10.2 其他配置

10.2.1 共享库配置

共享库可以在多个应用中共享相同的类库, 降低内存的使用。配置的步骤如下:

1. 可以在 `${APUSIC_HOME}/` 目录下创建一个目录, 如 `sharelib`;
2. 目录中放置需要在多个应用中共享的 jar 包;
3. 配置 `${APUSIC_HOME}/conf` 下的 `apusic.properties`, 修改 `shared.loader` 的配置为:

```
shared.loader="${apusic.home}/sharelib/*.jar"
```

经过上面的步骤即可完成共享库配置, 应用则可以直接使用该 jar 的资源, 共享库的类库优先级低于应用中相同的类库优先级。

10.2.2 手动部署多个应用

在普通方式部署时, 应用除了直接拷贝到 `webapps` 目录自动进行部署外, 也可以通过修改配置文件手动部署: 修改 `apusic.conf` 文件中的 `host` 标签, 增加如下的类似配置, 其中部署了 2 个 `application`, 属性 `path` 为指定的上下文路径, `docBase` 是应用的目录:

```
<host name="localhost" appBase="webapps" unpackWARs="true"
autoDeploy="true">
```

```

    <application path="/test1" docBase="/opt/apps/test1.war"
useHttpOnly="true" />
    <application path="/test2" docBase="/opt/apps/test1.war"
useHttpOnly="true" />
</host>

```

10.2.3 端口的修改

修改 `apusic.conf` 文件中 `endpointname="ams-http"` 的 `port` 属性，默认为 `port="6888"`。

10.2.4 开启 JMX

JMX 服务默认为不开启，如果需要开启 JMX 服务，首先需要检查配置文件中是否存在 `ams-jmx` 服务配置，如下：

```

<endpoint name="ams-jmx" port="6868" auth="false"
protocol="JMXRMI/2.0">
</endpoint>

```

如果没有则需要添加。再需要在 `apusic.conf` 中找到如下的配置，把 `ams-jmx` 增加到 `endpoints` 属性中，如下配置：

```

<server name="ams-server" services="ams-service" endpoints="ams-
http,ams-jmx" executors="http-thread-pool" />

```

JMX 默认访问不进行授权验证，如果需要授权认证，则需要修改相关配置并制定授权文件：

修改 `ams-jmx` 服务属性 `auth` 设置为 `true`，配置密码和访问授权文件属性 `passFile` 和 `accessFile`，默认为 `conf` 目录下的 `jmxremote.password` 和 `jmxremote.access` 文件

`jmxremote.password` 文件内容类似：

```
controlRole changeit
```

其中 `controlRole` 为用户名，`changeit` 为密码

`jmxremote.access` 文件内容类似：

```
controlRole readonly
```

```
controlRole readwrite \  
    create
```

```
javax.management.monitor.* , javax.management.timer.* \  
    unregister
```

11 配置数据源

应用服务器提供配置数据源功能。

1.数据库连接池配置（在 `context.xml` 文件中新增内容，如下图）：

```
<Resource name="jdbc/test" auth="Container"
type="javax.sql.DataSource" driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://127.0.0.1:3306/test_platform?
useUnicode=true&characterEncoding=utf-8" username="root"
password="root" maxTotal="100" maxIdle="30"
maxWaitMillis="10000" />
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- The contents of this file will be loaded for each web application -->
3 <Context>
4
5 <!-- Default set of monitored resources. If one of these changes, the -->
6 <!-- web application will be reloaded. -->
7 <WatchedResource>WEB-INF/web.xml</WatchedResource>
8 <WatchedResource>WEB-INF/aas-web.xml</WatchedResource>
9 <WatchedResource>${apusic.base}/conf/web.xml</WatchedResource>
10
11 <!-- Uncomment this to disable session persistence across Aas restarts -->
12 <!--
13 <Manager pathname="" />
14 -->
15
16 <Resource name="jdbc/test"
17 auth="Container"
18 type="javax.sql.DataSource"
19 driverClassName="com.mysql.jdbc.Driver"
20 url="jdbc:mysql://127.0.0.1:3306/test_platform?useUnicode=true&characterEncoding=utf-8"
21 username="root"
22 password="root"
23 maxTotal="100"
24 maxIdle="30"
25 maxWaitMillis="10000"/>
26
27 </Context>
28
```

表21: 金蝶Apusic应用服务器V10.0.10标准版数据源配置说明

配置参数	说明	值类型	默认值
name	数据源 jndi 名称	字符	
auth	容器负责资源的连接	字符	
type	绑定类型	字符	

url	数据库连接 url	字符	
username	用户名	字符	
password	密码	字符	
maxTotal	最大连接数	字符	
maxIdle	最大空闲连接数	字符	
maxWaitMillis	初始化连接大小	字符	0
minIdle	最小空闲连接数	字符	
validationQuery	连接有效性检测语句	字符	
validationQueryTimeout	检测语句超时时间, -1表示不超时	字符	-1
testOnBorrow	true 或 false, 从池中拿连接的时, 是否检测连接有效性	布尔值	true
testOnReturn	true 或 false, 还回连接到池里面时, 是否检测连接有效性	布尔值	false
removeAbandonedOnBorrow	true 或 false, 是否移除失效的连接	布尔值	false
logAbandoned	true 或 false, 是否打印移除失效连接	布尔值	false
testWhileIdle	true 或 false, 连接空闲时, 是否检测连接有效性	布尔值	false
minEvictableIdleTimeMillis	连接空闲时间, 超过该时间会从池中移除 (单位为毫秒)	整型	
timeBetweenEvictionRunsMillis	每隔多少时间去检测一次空闲连接是否超时, 默认不检测 (单位为毫秒)	整型	

2.将数据源对应版本的驱动放入到 `${APUSIC_HOME}/lib/` 目录下。

****注意:****在代码中要引用 jndi 需要加上前缀: `java:comp/env`, 如上面的数据源的:

```
java:comp/env/jdbc/test
```

12 配置 HTTPS

12.1 单向认证 -- 证书格式为 JKS

注意：配置了新的 `endpoint` 需要在 `server` 中进行引用，配置了线程池也需要在 `server` 中进行引用。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1"
SSLEnabled="true" executor="http-thread-pool">
  <SSLHostConfig>
    <Certificate
certificateKeystoreFile="conf/cert/arsp_server_keystore.jks"
certificateKeystoreType="JKS"
      certificateKeystorePassword="changeit" />
    </SSLHostConfig>
  </endpoint>
```

表22：金蝶Apusic应用服务器V10.0.10标准版证书格式为JKS的单向认证方式配置说明

配置参数	说明	值类型	默认值
SSLEnabled	是否是 ssl 通讯	布尔值	
executor	使用的线程池	字符	
certificateKeystoreType	keystore 格式, JKS 或者 PKCS12	字符	
certificateKeystorePassword	keystore 密码	字符	
certificateKeystoreFile	keystore 文件	字符	

12.2 双向认证 -- 证书格式为 JKS

注意：配置了新的 `endpoint` 需要在 `server` 中进行引用，配置了线程池也需要在 `server` 中进行引用。

```
<endpoint name="ams-https" port="8443" protocol="HTTP/1.1"
SSLEnabled="true" executor="http-thread-pool">
  <SSLHostConfig truststoreType="JKS"
```

```

truststorePassword="changeit"
    truststoreFile="conf/cert/arsp_server_truststore.jks"
certificateVerification="required">
    <Certificate
certificateKeystoreFile="conf/cert/arsp_server_keystore.jks"
certificateKeystoreType="JKS"
        certificateKeystorePassword="changeit" />
    </SSLHostConfig>
</endpoint>

```

表23：金蝶Apusic应用服务器V10.0.10标准版证书格式为JKS的双向认证方式配置说明

配置参数	说明	值类型	默认值
SSLEnabled	是否是 ssl 通讯	布尔值	
executor	使用的线程池	字符	
truststoreType	信任库格式, JKS 或者 PKCS12	字符	
truststorePassword	信任库密码	字符	
truststoreFile	信任库文件	字符	
certificateVerification	是否需要验证客户端证书, 即是否双向认证, 默认不验证	字符	
certificateKeystoreType	keystore 格式, JKS 或者 PKCS12	字符	
certificateKeystorePassword	keystore 密码	字符	
certificateKeystoreFile	keystore 文件	字符	

12.3 单向认证 -- 证书格式为 PEM

注意：配置了新的 `endpoint` 需要在 `server` 中进行引用，配置了线程池也需要在 `server` 中进行引用。

```

<endpoint name="ams-https" port="8443" protocol="HTTP/1.1"
SSLEnabled="true" executor="http-thread-pool">
    <SSLHostConfig>

```

```

    <Certificate certificateKeyFile="conf/cert/private_key.pem"
certificateKeyPassword="changeit"
        certificateFile="conf/cert/public_key.pem"
certificateChainFile="conf/cert/chain.pem" type="RSA" />
    </SSLHostConfig>
</endpoint>

```

表24: 金蝶Apusic应用服务器V10.0.10标准版证书格式为PEM的单向认证方式配置说明

配置参数	说明	值类型	默认值
SSLEnabled	是否开启 ssl 通讯	布尔值	
executor	使用的线程池	字符	
type	证书类型	字符	
certificateKeyFile	私钥文件	字符	
certificateKeyPassword	私钥密码, 可选	字符	
certificateFile	公钥文件	字符	
certificateChainFile	证书链文件, 可选	字符	

12.4 双向认证 -- 证书格式为 PEM

```

<endpoint name="ams-https" port="8443" protocol="HTTP/1.1"
SSLEnabled="true" executor="http-thread-pool">
    <UpgradeProtocol
className="com.apusic.connector.http2.Http2Protocol" />
    <SSLHostConfig caCertificateFile="conf/cert/public_ca_key.pem"
certificateVerification="need">
        <Certificate certificateKeyFile="conf/cert/private_key.pem"
certificateKeyPassword="changeit"
            certificateFile="conf/cert/public_key.pem"
certificateChainFile="conf/cert/chain.pem" type="RSA" />
        </SSLHostConfig>
    </endpoint>

```

表25: 金蝶Apusic应用服务器V10.0.10标准版证书格式为PEM的双向认证方式配置说明

配置参数	说明	值类型	默认值
SSLEnabled	是否是 ssl 通讯	布尔值	
executor	使用的线程池	字符	
type	证书类型	字符	
certificateKeyFile	私钥文件	字符	
certificateKeyPassword	私钥密码, 可选	字符	
certificateFile	公钥文件	字符	
certificateChainFile	证书链文件, 可选	字符	
caCertificateFile	根证书文件, 可选	字符	
certificateVerification	是否需要验证客户端证书, 即是否双向认证	字符	

13 国密配置说明

13.1 环境说明

需要使用 JDK11 版本和国密浏览器（如 360 国密浏览器）。

关于国密证书的制作 [请参考](#)

13.2 应用服务器证书配置

13.2.1 单向认证

修改 `apusic.conf` 文件，增加如下的 `endpoint` 配置（如果已经存在，则进行修改）：

```
<endpoint name="https" port="6887" maxThreads="150"
SSLEnabled="true">
  <SSLHostConfig sslProtocol="SMv1.1" protocols="SMv1.1">
    <Certificate certificateKeystoreFile="conf/keystore.pfx"
certificateKeystorePassword="changeit"
certificateKeyPassword="changeit" type="EC"
certificateKeystoreType="PKCS12" />
  </SSLHostConfig>
</endpoint>
```

表26：金蝶Apusic应用服务器V10.0.10标准版国密证书的单向认证方式配置说明

配置参数	说明	值类型	默认值
SSLEnabled	是否是 ssl 通讯	布尔值	
certificateKeystoreFile	表示证书文件的路径	字符	
protocols	需要指定为 SMv1.1	字符	
sslProtocol	需要指定为 SMv1.1	字符	
certificateKeystorePassword	为证书库文件的密码	字符	
certificateKeyPassword	为证书的密码	字符	

type	证书类型	字符	
certificateKeystoreType	keystore类型, 如PKCS12	字符	

13.2.2 双向认证

修改 `apusic.conf` 文件, 增加如下的 `endpoint` 配置 (如果已经存在, 则进行修改):

```
<endpoint name="https" port="6887" maxThreads="150"
SSLEnabled="true">
  <SSLHostConfig sslProtocol="SMv1.1" protocols="SMv1.1"
truststoreFile="conf/truststore.jks"
  truststorePassword="changeit" certificateVerification="want"
truststoreType="JKS">
    <Certificate certificateKeystoreFile="conf/keystore.p12"
certificateKeystorePassword="changeit"
  certificateKeyPassword="changeit" type="EC"
certificateKeystoreType="PKCS12" />
  </SSLHostConfig>
</endpoint>
```

表27: 金蝶Apusic应用服务器V10.0.10标准版国密证书的双向认证方式配置说明

配置参数	说明	值类型	默认值
SSLEnabled	是否是 ssl 通讯	布尔值	
certificateKeystoreFile	表示证书文件的路径	字符	
protocols	需要指定为 SMv1.1	字符	
sslProtocol	需要指定为 SMv1.1	字符	
certificateKeystorePassword	为证书库文件的密码	字符	
certificateKeyPassword	为证书的密码	字符	
type	证书类型	字符	
certificateKeystoreType	keystore类型, 如PKCS12	字符	

truststoreFile	指定信任库文件，其类型是JKS	字符	
truststorePassword	信任库密码	字符	
certificateVerification	是否进行证书验证，双向认证时为 need 值，可选客户端认证设置为 want	字符	

13.2.3 增加 endpoint 到 server

修改 `apusic.conf` 文件，修改如下的配置，其中 `endpoints` 属性中的 `https` 就是上面配置单向认证或双向认证的 `endpoint` 的名字：

```
<server name="ams-server" services="ams-service"
endpoints="http,https" executors="http-thread-pool" />
```

13.3 启动应用服务器

通过上面的步骤配置完成后，通过命令 `apusic.sh gm start` 或 `apusic.sh gm run` 可以启动国密支持的容器。

备注：可以通过在启动脚本中增加参数 `-Djavax.net.debug=ssl:handshake` 来打印具体的 SSL 处理过程，方便分析可能存在的问题。

13.4 Springboot 中使用国密

支持springboot中使用国密证书，详情请参考《金蝶Apusic应用服务器V10.0.10嵌入式部署用户手册》。

13.4.1 注册国密实现 Jar 包

将 AAS 的国密实现 Jar 包 - `${APUSIC_HOME}/lib/aas-sm-3.0.jar` 注册到 Maven 库中，以备应用打包引用。

```
mvn install:install-file -Dfile=aas-sm-3.0.jar -DgroupId=com.apusic
-DartifactId=aas-sm -Dversion=3.0 -Dpackaging=jar
```

13.4.2 Pom 文件应用国密实现

在 `pom.xml` 文件中添加国密实现依赖。

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-sm</artifactId>
  <version>3.0</version>
</dependency>
```

13.4.3 应用配置

在 `application.properties` 文件中添加国密配置。如：

```
server.ssl.key-store = classpath:keystore.p12
server.ssl.key-store-type = PKCS12
server.ssl.key-store-password = changeit
server.ssl.key-password = changeit
server.ssl.protocol = SMv1.1
```

13.4.4 启动应用

Springboot 应用中使用国密需要在应用启动中添加 `-Dcom.apusic.security.ssl.EnableGMTLS=true` 参数，例如：

```
java -Dcom.apusic.security.ssl.EnableGMTLS=true -jar springboot-gm-
demo.jar
```

14 日志配置

14.1 系统日志配置

通过配置系统日志文件配置日志信息，日志文件为 `logging.properties`。

```
handlers = 1apusic.com.apusic.juli.AsyncFileHandler, 2localhost.com.apusic.juli.AsyncFileHandler, com.apusic.juli.logging.ApusicConsoleHandler
.handlers = 1apusic.com.apusic.juli.AsyncFileHandler, com.apusic.juli.logging.ApusicConsoleHandler

#####
# Handler specific properties.
# Describes specific configuration info for Handlers.
#####

1apusic.com.apusic.juli.AsyncFileHandler.level = FINE
1apusic.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
1apusic.com.apusic.juli.AsyncFileHandler.prefix = apusic.
1apusic.com.apusic.juli.AsyncFileHandler.maxDays = 90
1apusic.com.apusic.juli.AsyncFileHandler.encoding = UTF-8

2localhost.com.apusic.juli.AsyncFileHandler.level = FINE
2localhost.com.apusic.juli.AsyncFileHandler.directory = ${apusic.base}/logs
2localhost.com.apusic.juli.AsyncFileHandler.prefix = localhost.
2localhost.com.apusic.juli.AsyncFileHandler.maxDays = 90
2localhost.com.apusic.juli.AsyncFileHandler.encoding = UTF-8

com.apusic.juli.logging.ApusicConsoleHandler.level = FINE
com.apusic.juli.logging.ApusicConsoleHandler.formatter = com.apusic.juli.OneLineFormatter
com.apusic.juli.logging.ApusicConsoleHandler.encoding = UTF-8

#####
# Facility specific properties.
# Provides extra control for each logger.
#####

com.apusic.ams.core.ContainerBase.[ams-service].[localhost].level = INFO
com.apusic.ams.core.ContainerBase.[ams-service].[localhost].handlers = 2localhost.com.apusic.juli.AsyncFileHandler

# For example, set the com.apusic.ams.util.LifecycleBase logger to log
# each component that extends LifecycleBase changing state:
#com.apusic.ams.util.LifecycleBase.level = FINE
```

15 访问日志记录

Valve元素表示将插入到引擎、主机或上下文的请求处理管道中的组件。单个值具有不同的处理能力，下文将分别进行描述。

15.1 配置访问日志

访问日志可配置在 `application` 或者 `service` 里面。配置在 `host` 里面则该 `host` 下的应用访问都会被记录日志。

```
<Valve className="com.apusic.ams.valves.AccessLogValve"
directory="logs" prefix="access_log" suffix=".txt"
pattern="common" />
```

```
<services>
  <service name="ams-service" defaultHost="localhost">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <Valve className="com.apusic.ams.valves.AccessLogValve" directory="logs" prefix="access_log" suffix=".txt" pattern="common" />
      <application path="/test2" docBase="/opt/testz/aams/demo/examples1.war" useHttpOnly="true" />
      <Valve className="com.apusic.ams.valves.RemoteAddrValve" allow="172.24.6.160" deny="" />
    </host>
  </service>
</services>
```

表28：金蝶Apusic应用服务器V10.0.10标准版访问日志配置说明

配置参数	说明	值类型	默认值
className	要使用的实现的Java类名，用默认访问日志值，则必须将其设置为com.apusic.ams.valves.AccessLogValve	字符	com.apusic.ams.valves.AccessLogValve
buffered	确定日志记录是否将被缓冲。如果设置为false，则每次请求后都会写入访问日志。	字符	true
directory	日志存储目录	字符	
prefix	日志文件前缀	字符	
suffix	日志文件后缀	字符	
fileDateFormat	文件时间格式，一般文件命名为 prefix（前缀）+ 时间格式（一般是按天算）+.+suffix（后	字符	

	缀)，如果想改为每小时一个文件可设置为 <code>fileDateFormat="yyyy-MM-dd.HH"</code>		
pattern	日志格式化布局，用于标识要记录的请求和响应中的各种信息字段，或通用或组合的单词，以选择标准格式。有关配置此属性的更多信息，请参阅下文。	字符	
conditionIf	启用条件日志记录。如果设置，则只有当 <code>ServletRequest.getAttribute()</code> 不为空时，才会记录请求。例如，如果此值设置为 <code>important</code> ，则只有当 <code>ServletRequest.getAttribute("important") != null</code> 时才是有效的。使用过滤器是一种简单的方法，可以在许多不同的请求上设置/取消设置 <code>ServletRequest</code> 中的属性	字符	
conditionUnless	启用条件日志记录。如果设置，则只有当 <code>ServletRequest.getAttribute()</code> 为 <code>null</code> 时，才会记录请求。例如，如果此值设置为 <code>junk</code> ，则只有当 <code>ServletRequest.getAttribute("junk") == null</code> 时，才会记录特定请求。使用过滤器是一种简单的方法，可以在许多不同的请求上设置/取消设置 <code>ServletRequest</code> 中的属性	字符	
encoding	用于写入日志文件的字符集。空字符串表示使用默认字符集	字符	UTF-8
ipv6Canonical	用于确定IPv6地址是否应以RFC 5952定义的规范表示格式表示。如果设置为 <code>true</code> ，则IPv6地址将以规范格式写入（例如， <code>2001:db8::1:0:0:1, ::1</code> ），否则将以完整形式表示（例如， <code>2001:db8:0:0:1:0:0:1, 0:0:0:0:0:0:0:1</code> ）	字符	false
locale	用于格式化访问日志行中时间戳的区域设置。使用显式 <code>SimpleDateFormat</code> 模式配置的任何时间戳（ <code>%{xxx}t</code> ）在此区域设置中格式化。默认情况下，使用Java进程的默认区域设置。不支持在 <code>AccessLogValve</code> 初始化后切换区域设置。任何使用通用日志格式（CLF）的时间戳都始终在区域设置 <code>en_US</code> 中格式化。	字符	
maxDays	在删除之前，将保留轮换访问日志的最大天数。如果未指定，将使用默认值 <code>-1</code> ，这意味着永远不会删除旧文件。	字符	-1

maxLogMessageBufferSize	日志消息缓冲区通常会被回收和重复使用。为了防止过度使用内存，如果缓冲区增长超过此大小，它将被丢弃。默认值为256个字符。这应该设置为大于典型的访问日志消息大小。	字符	256
renameOnRotate	默认情况下，对于可旋转日志，活动访问日志文件名将包含fileDateFormat中的当前时间戳。在旋转过程中，文件被关闭，并创建和使用名称中具有下一个时间戳的新文件。当将renameOnRotate设置为true时，时间戳不再是活动日志文件名的一部分。只有在旋转过程中，文件才会关闭，然后重命名以包含时间戳。这类似于大多数日志框架在进行基于时间的轮换时的行为	字符	false
requestAttributesEnabled	设置为true可检查是否存在请求属性（通常由RemoteIpValve等设置），这些属性应用于覆盖远程地址、远程主机、服务器端口和协议请求返回的值。如果未设置属性，或者此属性设置为false，则将使用请求中的值。如果未设置，将使用默认值false。	字符	false
rotatable	是否进行日志轮换。如果设置为false，则此文件永远不会旋转，fileDateFormat将被忽略	字符	true

pattern: 参考下面内容:

%a: 远程 IP 地址

%A: 本地 IP 地址

%b: 发送的字节数，不包括 HTTP 头，如果为零则显示为"_"

%B: 发送的字节数，不包括 HTTP 头

%F: 提交响应所用的时间（毫秒）

%h: 远程主机名（如果连接器的enableLookups为false，则为IP地址）

%H: 请求协议

%l (小写的 L): 来自identd的远程逻辑用户名（始终返回"-")

%m: 请求方法（GET、POST等）

%p: 接收此请求的本地端口

%q: 查询字符串 (如果存在, 则前缀为“? ”)

%r: 请求的第一行 (方法和请求URI)

%s: 响应的 HTTP 状态代码

%S: 用户会话 ID

%t: 日期和时间, 通用日志格式

%T: 处理请求所用的时间, 以秒为单位

%u: 已通过身份验证的远程用户 (如果有的话), 否则为“-” (如果需要, 则转义)

%U: 请求的 URL 路径

%v: 本地服务器名

%D: 处理请求所用的时间 (微秒)

%T: 处理请求所用的时间 (秒)

%I(大写的 i): 当前请求线程名称 (稍后可以与堆栈跟踪进行比较)

%X: 响应完成时的连接状态:

- X= 响应完成前连接已中止。
- += 发送响应后, 连接可能会保持活动状态。
- -= 发送响应后, 连接将关闭

还支持写入传入或传出头部、cookie、会话或请求属性以及特殊时间戳格式的信息。类似Apache语法, 以下每个占位符都可以与不同的 `xxx` 键一起使用多次:

- `%{xxx}a` : 写入远程地址 (客户端) (`xxx=remote`) 或连接对等方地址 (`xxx=peer`) 。
- `%{xxx}i` : 写入名称为xxx的传入头部的值 (如果需要, 则进行转义) 。
- `%{xxx}o` : 写入名称为xxx的传出头部的值 (如果需要, 则进行转义) 。
- `%{xxx}c` : 写入名称为xxx的cookie的值 (用逗号分隔, 如果需要, 则进行转义) 。
- `%{xxx}r` : 写入名称为xxx的ServletRequest属性的值 (如果需要, 则进行转义; 如果请求为null, 则值为??) 。
- `%{xxx}s` : 写入名称为xxx的HttpSession属性的值 (如果需要, 则进行转义; 如果请求为null, 则值为??) 。
- `%{xxx}p` : 写入本地 (服务器) 端口 (`xxx=local`) 或远程 (客户端) 端口 (`xxx=remote`) 。
- `%{xxx}t` : 使用增强的 `SimpleDateFormat` 模式xxx格式化请求结束时的时间戳。

- `%{xxx}T` : 使用单位xxx写入处理请求所花费的时间, 其中有效单位为ns (纳秒)、us (微秒)、ms (毫秒)、fracsec (秒的小数部分) 或s (整秒)。`%{s}T` 与 `%T` 等效, `%{us}T` 与 `%D` 等效。

在 `%{xxx}t` 中, 允许使用 `SimpleDateFormat` 支持的所有格式。此外, 还添加了以下扩展:

- `sec` : 自纪元 (epoch) 以来的秒数。
- `msec` : 自纪元 (epoch) 以来的毫秒数。
- `msec_frac` : 毫秒的小数部分。

这些格式不能与 `SimpleDateFormat` 格式在同一个格式标记中混合使用。此外, 还可以定义是记录请求开始时间的戳还是响应结束时间的戳:

- `begin` 或前缀 `begin:` : 选择请求开始时间。
- `end` 或前缀 `end:` : 选择响应结束时间。

通过在模式中添加多个 `%{xxx}t` 标记, 也可以同时记录这两个时间戳。

转义规则如下: 双引号 (") 被转义为", 反斜杠 (\) 被转义为\。对于\f、\n、\r和\t, 使用标准C转义规则。任何其他控制字符或代码点高于127的字符都使用标准Java Unicode转义 (\uXXXX) 进行编码。

简写模式 `pattern="common"` 对应由 `'%h %l %u %t "%r" %s %b'` 定义的通用日志格式 (Common Log Format)。

简写模式 `pattern="combined"` 在 `common` 模式的基础上, 将 `Referer` 和用户代理头 (User-Agent headers) 的值 (每个值都用双引号括起来) 追加到日志中, 其格式为 `"%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\""`。

使用未知模式标识符的字段将被记录为 `???X???`, 其中X是未知标识符。对于包含未知模式标识符和 `{xxx}` 键的字段, 将被记录为 `???`。

15.2 访问控制

15.2.1 远程地址

限制远程地址访问, 即允许将提交此请求的客户端的IP地址与一个或多个正则表达式进行比较, 并根据比较结果允许请求继续或拒绝处理来自此客户端的请求。

将 `addConnectorPort` 属性设置为true后, 可以在服务器连接器端口后附加一个分号 (;), 以便为每个连接器使用不同的表达式。

将 `usePeerAddress` 属性设置为true后, 该阀 (Valve) 将在其检查中使用连接对等地址。如果在AAS前面结合AJP协议或HTTP协议加 `RemoteIp (Valve | Filter)` 使用反向代理, 则这将与客户端IP不同。。

被拒绝的请求将收到状态码为403的响应。可以使用 `denyStatus` 属性来覆盖此状态码。

将 `invalidAuthenticationWhenDeny` 属性设置为`true`后，可以改变被拒绝请求的行为，不是拒绝请求，而是设置一个无效的身份验证头部。这与上下文属性 `preemptiveAuthentication="true"` 结合使用时非常有用。

注意：在使用此阀处理IPv6地址时有一个注意事项。此阀处理的IP地址格式取决于用于获取它的API。如果地址是通过Java套接字使用`Inet6Address`类获得的，则其格式将为 `x:x:x:x:x:x:x:x`。也就是说，localhost的IP地址将是 `0:0:0:0:0:0:0:1`，而不是更常用的 `::1`。请查阅您的访问日志以了解实际值。

远程地址的属性包括：

配置参数	说明	值类型	默认值
<code>classname</code>	要使用的实现的Java类名。必须设置为 <code>com.apusic.ams.valves.RemoteAddrValve</code>	字符	<code>com.apusic.ams.valves.RemoteAddrValve</code>
<code>allow</code>	允许远程访问的IP地址，如果指定了此属性，则远程地址必须匹配才能接受此请求。如果未指定此属性，则除非远程地址与拒绝模式匹配，否则将接受所有请求。	字符	
<code>deny</code>	禁止远程访问的IP地址。如果指定了此属性，则远程地址必须不匹配才能接受此请求。如果未指定此属性，则请求接受仅由 <code>allow</code> 属性决定。	字符	
<code>denyStatus</code>	拒绝请求时使用的HTTP响应状态码	字符	403
<code>addConnectorPort</code>	将服务器连接器端口附加到客户端IP地址后，并使用分号（";"）分隔。如果设置为 <code>true</code> ，则使用 <code>allow</code> 和 <code>deny</code> 配置的表达式将与 <code>ADDRESS;PORT</code> 进行比较。其中 <code>ADDRESS</code> 是客户端IP地址， <code>PORT</code> 是接收请求的AAS连接器端口。	字符	<code>false</code>
<code>invalidAuthenticationWhenDeny</code>	当应拒绝请求时，不拒绝请求而是设置无效的身份验证头。这仅当上下文具有属性 <code>preemptiveAuthentication="true"</code> 设置时才有效。不会覆盖已存在的身份验证头。实际上，这将触发身份验证而不是拒绝，即使应用程序没有配置安全约束也是如此。这可以与 <code>addConnectorPort</code> 结合使用，以便根据客户端和用于访问应用程序的连接器来触发身份验证	字符	
<code>usePeerAddress</code>	使用连接对等方地址而不是客户端IP地址。如果AAS前面使用了反向代理，并且与AJP协议或HTTP协议（加上 <code>RemoteIp (Valve Filter)</code> ）结合使用，那么它们将会有所不同。	字符	

配置示例：

只允许 172.24.6.* 访问。

```
<Valve className="com.apusic.ams.valves.RemoteAddrValve"
allow="172.24.6.*" deny="" />
```

```
<services>
  <service name="ams-service" defaultHost="localhost">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <Valve className="com.apusic.ams.valves.AccessLogValve" directory="logs" prefix="access_log" suffix=".txt" pattern="common" />
      <application path="/examples" docBase="/opt/testz/aams/examples.war" useHttpOnly="true" />
      <application path="/test2" docBase="/opt/testz/aams/demo/examples1.war" useHttpOnly="true" />
      <Valve className="com.apusic.ams.valves.RemoteAddrValve" allow="172.24.6.*" deny="" />
    </host>
  </service>
</services>
```

15.2.2 远程主机名

限制远程主机名访问，允许将提交此请求的客户端的主机名与一个或多个正则表达式进行比较，并根据比较结果允许请求继续或拒绝处理来自此客户端的请求。

在设置属性 `addConnectorPort` 为true后，可以在正则表达式后附加服务器连接器端口（使用分号“;”分隔），以便为每个连接器使用不同的表达式。

拒绝的请求将使用状态码403进行响应。可以使用属性 `denyStatus` 覆盖此状态码。

通过设置属性 `invalidAuthenticationWhenDeny` 为true，可以更改拒绝请求时的行为，不是拒绝请求，而是设置无效的身份验证头。这在与上下文属性`preemptiveAuthentication="true"`结合使用时很有用。

注意：此阈值处理由 `ServletRequest.getRemoteHost()` 方法返回的值。为了允许该方法返回正确的主机名，你必须在连接器上启用“DNS查找”功能。

远程主机名的属性包括：

配置参数	说明	值类型	默认值
classname	要使用的实现的Java类名。必须设置为com.apusic.ams.valves.RemoteHostValve	字符	com.apusic.ams.valves.RemoteHostValve
allow	允许远程访问的主机名，如果指定了此属性，则远程地址必须匹配才能接受此请求。如果未指定此属性，则除非远程主机名与拒绝模式匹配，否则将接受所有请求。	字符	

deny	禁止远程访问的主机名。如果指定了此属性，则远程主机名必须不匹配才能接受此请求。如果未指定此属性，则请求接受仅由allow属性决定。	字符	
denyStatus	拒绝请求时使用的HTTP响应状态码	字符	403
addConnectorPort	将服务器连接器端口附加到客户端主机名后，并使用分号(“;”)分隔。如果设置为true，则使用allow和deny配置的表达式将与HOSTNAME;PORT进行比较。其中HOSTNAME是客户端主机名，PORT是接收请求的AAS连接器端口。	字符	false
invalidAuthenticationWhenDeny	当应拒绝请求时，不拒绝请求而是设置无效的身份验证头。这仅当上下文具有属性preemptiveAuthentication="true"设置时才有效。不会覆盖已存在的身份验证头。实际上，这将触发身份验证而不是拒绝，即使应用程序没有配置安全约束也是如此。这可以与addConnectorPort结合使用，以便根据客户端和用于访问应用程序的连接器来触发身份验证	字符	

15.2.3 远程CIDR

限制网络掩码访问。允许将提交此请求的客户端的IP地址与一个或多个遵循CIDR表示法的网络掩码进行比较，并根据比较结果允许请求继续或拒绝处理来自此客户端的请求。支持IPv4和IPv6。

此值类似Apache的Order、Allow from和Deny from指令，但存在以下限制：

- Order总是遵循“allow, deny”的顺序；
- 不支持使用点分十进制表示法来指定网络掩码（即，不能写成 `192.168.1.0/255.255.255.0`，而必须写成 `192.168.1.0/24`）；
- 不支持使用快捷方式，如 `10.10.`（这相当于 `10.10.0.0/16`）；
- 正如值的名称所示，这是一个仅支持CIDR的值，因此不支持像 `.mydomain.com` 这样的子域表示法。

将 `addConnectorPort` 属性设置为 `true` 后，可以在服务器连接器端口后附加一个分号(;)，以便为每个连接器使用不同的表达式。

通过将 `usePeerAddress` 属性设置为`true`，该值将在检查中使用连接的对等地址。如果AAS前面使用了反向代理，并且与AJP协议或HTTP协议以及 `RemoteIp(Valve|Filter)` 结合使用，那么此对等地址将与客户端IP不同。

被拒绝的请求将收到状态码为403的响应。可以使用 `denyStatus` 属性来覆盖此默认状态码。

将 `invalidAuthenticationWhenDeny` 属性设置为`true`后，当请求被拒绝时，行为将更改为不直接拒绝，而是设置一个无效的身份验证头部。这与上下文属性 `preemptiveAuthentication="true"` 结合使用时非常有用。

此值的其他一些特性包括：

- 如果省略了CIDR前缀，此值将变为一个单IP值；
- 与Remote Host Valve不同，它可以处理压缩形式的IPv6地址（如：`::1`、`fe80::/71`等）。

远程CIDR的属性包括：

配置参数	说明	值类型	默认值
<code>classname</code>	要使用的实现的Java类名。必须设置为 <code>com.apusic.ams.valves.RemoteCIDRValve</code>	字符	<code>com.apusic.ams.valves.RemoteCIDRValve</code>
<code>allow</code>	一个由逗号分隔的IPv4或IPv6网络掩码或地址列表，用于与远程客户端的IP地址进行匹配。如果指定了此属性，则远程地址必须匹配才能接受此请求。如果未指定此属性，则除非远程IP与 <code>deny</code> 属性中的网络掩码匹配，否则将接受所有请求	字符	
<code>deny</code>	一个由逗号分隔的IPv4或IPv6网络掩码或地址列表，用于与远程客户端的IP地址进行匹配。如果指定了此属性，则远程地址必须不匹配才能接受此请求。如果未指定此属性，则请求接受仅由 <code>allow</code> 属性决定。	字符	
<code>denyStatus</code>	拒绝请求时使用的HTTP响应状态码	字符	403
<code>addConnectorPort</code>	将服务器连接器端口附加到客户端IP地址后，并使用分号（ <code>;</code> ）分隔。如果设置为 <code>true</code> ，则使用 <code>allow</code> 和 <code>deny</code> 配置的表达式将与 <code>ADDRESS;PORT</code> 进行比较。其中 <code>ADDRESS</code> 是客户端IP地址， <code>PORT</code> 是接收请求的AAS连接器端口。	字符	<code>false</code>
<code>invalidAuthenticationWhenDeny</code>	当应拒绝请求时，不拒绝请求而是设置无效的身份验证头。这仅当上下文具有属性 <code>preemptiveAuthentication="true"</code> 设置时才有效。不会覆盖已存在的身份验证头。实际上，这将触发身份验证而不是拒绝，即使应用程序没有配置安全约束也是如此。这可以与 <code>addConnectorPort</code> 结合使用，以便根据客户端和用于访问应用程序的连接器来触发身份验证	字符	
<code>usePeerAddress</code>	使用连接对等方地址而不是客户端IP地址。如果AAS前面使用了反向代理，并且与AJP协议或HT	字符	

TP协议（加上Remotep (Valve|Filter) ）结合使用，那么它们将会有所不同。

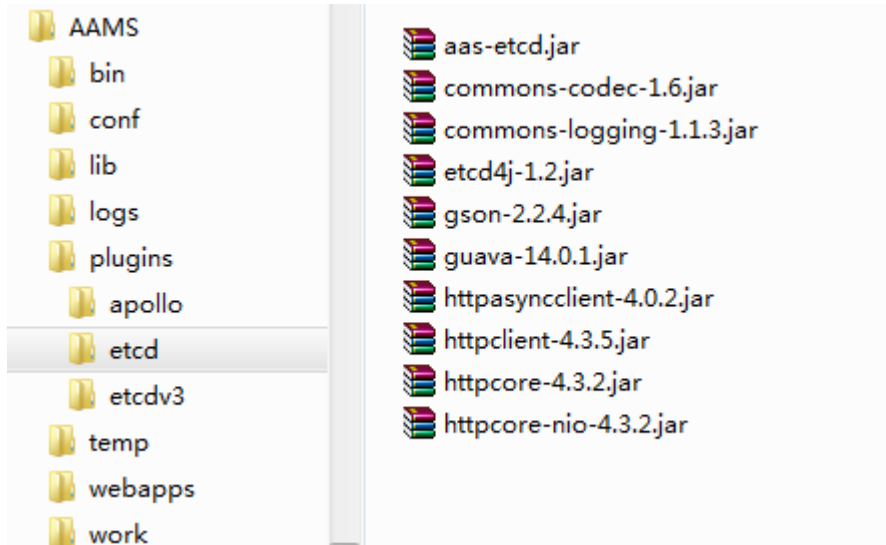
配置示例：

只允许从localhost连接的客户端访问。

```
<Valve className="com.apusic.ams.valves.RemoteCIDRValve"  
  allow="127.0.0.1, ::1"/>
```

16 集中配置说明

该版本可以支持集中配置软件包括 ETCD 和 apollo，后续会增加更多的支持。对集中配置软件的实现包，放在产品包的 `plugins` 目录下，如下面图所示：



备注：一般情况下一个用户场景只会使用到一个实现，可以删除其他的实现，以节约空间。

16.1 使用 apollo 集中配置

下面介绍如何在应用服务器中使用 apollo 软件进行集中配置。

16.1.1 添加集中配置服务

`apusic.conf` 配置文件需要增加 `plugins` 的配置，并指定 apollo 的实现，如下图所示：

```
<plugins>
  <plugin appId="10086"
    className="com.apusic.ams.config.listener.ApolloPluginLoader"
    cluster="sz-test"
      metaUrl="http://172.20.129.231:8080" namespace="test-
namespace" />
</plugins>
```

```

1 <?xml version="1.0" encoding="utf-8" standalone="no"?><config>
2
3
4   <plugins>
5     <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test"
6       metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
7   </plugins>
8
9   <listeners>
10    <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
11    <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
12    <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
13    <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
14    <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
15  </listeners>
16
17  <resources>
18    <resource auth="Container" description="User database that can be updated and saved"
19      factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase" pathname="conf/aas-users.xml"
20      type="com.apusic.ams.UserDatabase"/>
21  </resources>
22
23  <executors>
24    <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-"
25      prestartminSpareThreads="true"/>
26  </executors>
27
28  <endpoints>
29    <endpoint URIEncoding="utf-8" acceptCount="100"
30      compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image/png,image/gif,video/mp4" compression="on"
31      compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
32      executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192"
33      maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http" port="6999" processorCache="20000" protocol="HTTP/1.1"
34      redirectPort="7443" tcpNoDelay="true">
35    </endpoint>
36  </endpoints>
37
38  <services>
39    <service defaultHost="localhost" name="ams-service">
40      <realm className="com.apusic.ams.realm.LockOutRealm">
41        <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
42      </realm>
43      <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
44      </host>
45    </service>

```

其中 `className="com.apusic.ams.config.listener.ApolloPluginLoader"` 是加载 apollo 配置的类，属性包括了：

表30：金蝶Apusic应用服务器V10.0.10标准版与Apollo集成配置说明

配置参数	说明	值类型	默认值
appId	配置的应用id	字符	
cluster	配置属于的集群名称	字符	
metaUrl	Apollo 提供服务的地址	字符	
namespace	配置的命名空间	字符	
env	配置的环境（开发环境，正式环境等等）	字符	
idc	配置的数据中心	字符	

以上属性可通过配置文件配置，也可通过环境变量配置，也可通过系统变量设置

优先级是：环境变量 > 系统变量 > 配置文件

一般环境变量和系统变量的变量名称为大写的，环境变量（`System.getProperty("xxx")` 获取）和系统变量（`System.getenv("xxx")` 获取）的变量名为如下，每个变量名会与上面的属性对应：

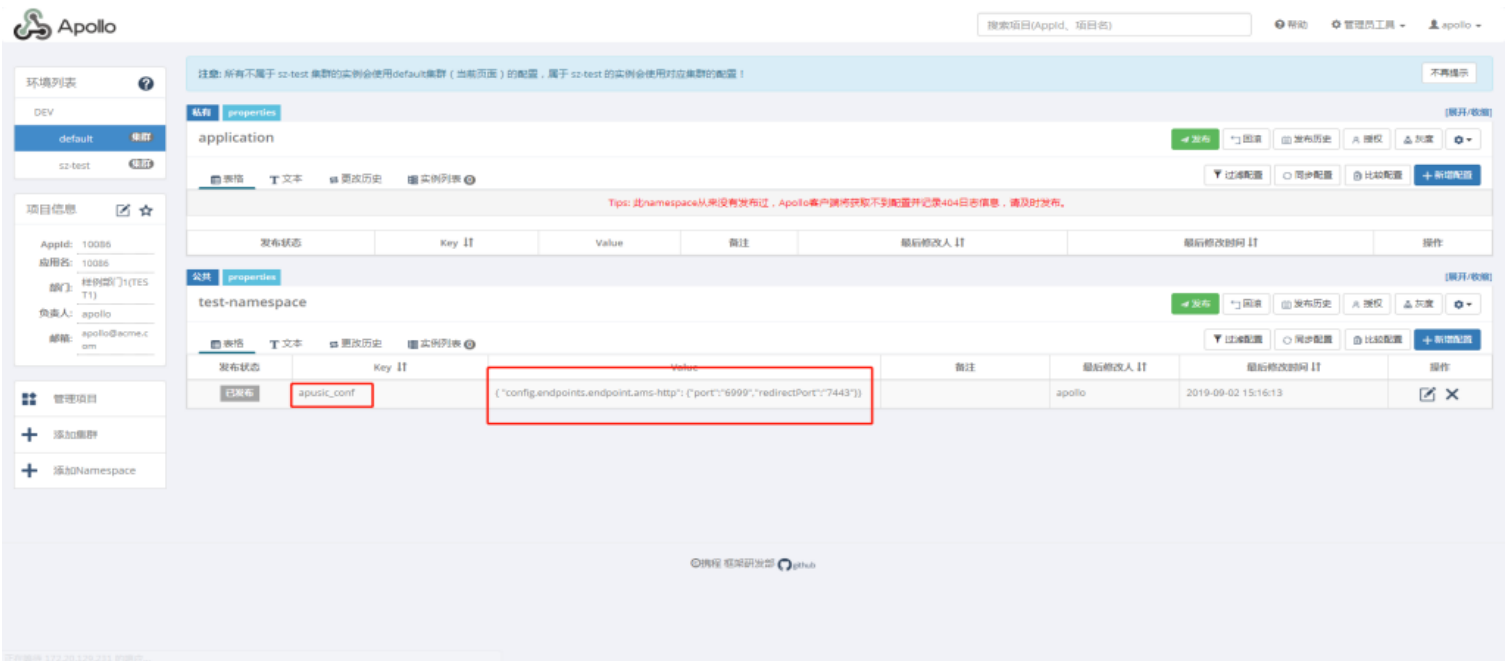
表31: 金蝶Apusic应用服务器V10.0.10标准版与Apollo集成环境变量配置说明

配置参数	说明	值类型	默认值
AAS_CONFIG_APOLLO_APP_ID	配置的应用id	字符	
AAS_CONFIG_APOLLO_CLUSTER	配置属于的集群名称	字符	
AAS_CONFIG_APOLLO_META_SERVER	Apollo 提供服务的地址	字符	
AAS_CONFIG_APOLLO_NAMESPACE	配置的命名空间	字符	
AAS_CONFIG_APOLLO_ENV	配置的环境（开发环境，正式环境等等）	字符	
AAS_CONFIG_APOLLO_IDC	配置的数据中心	字符	

16.1.2 在 apollo 的配置

16.1.2.1 apusic.conf 配置

与配置中心整合后，应用服务器的配置文件（`apusic.conf` 文件）可通过设置 `key` 为 `apusic_config`，`value` 为要改变的值进行修改，如下图：



其中 key 必须为 apusic_config , value 必须是 json 字符串。

```
{
  "config.endpoints.endpoint.ams-http": {
    "port": "6999",
    "redirectPort": "7443"
  }
}
```

config.endpoints.endpoint.ams-http 拆分对应这 config > endpoints > endpoint > ams-http
对应这 xml 的层级结构:

```

1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2
3 <config>
4   <plugins>
5     <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
6   </plugins>
7
8   <listeners>
9     <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
10    <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
11    <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
12    <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
13    <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
14  </listeners>
15
16  <resources>
17    <resource auth="Container" description="User database that can be updated and saved" factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase"
18      pathName="conf/aas-users.xml" type="com.apusic.ams.UserDatabase"/>
19  </resources>
20
21  <executors>
22    <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-" prestartminSpareThreads="true"/>
23  </executors>
24
25  <endpoints>
26    <endpoint URIEncoding="utf-8" acceptCount="100"
27      compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image
28      /png,image/gif,audio/mpeg,video/mp4" compression="on" compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
29      executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192" maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http"
30      port="6999" processorCache="20000" protocol="HTTP/1.1" redirectPort="7443" tcpNoDelay="true">
31    </endpoint>
32  </endpoints>
33
34  <services>
35    <service defaultHost="localhost" name="ams-service">
36      <realm className="com.apusic.ams.realm.LockOutRealm">
37        <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
38      </realm>
39    </service>
40  </services>
41
42  <servers>
43    <server endpoints="ams-http" executors="http-thread-pool" name="ams-server" services="ams-service"/>
44  </servers>
45
46  <containers>
47    <container listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"
48      name="container" port="-1" resources="userDatabase" servers="ams-server" shutdown="shutdown"/>
49  </containers>
50 </config>

```

port 和 redirectPort 对应着属性。

```

{
  "config.endpoints.endpoint.ams-http": {
    "port": "6999",
    "redirectPort": "7443"
  }
}

```

上图的意思就是，修改名称为 `ams-http` 的 `xml` 标签为 `endpoint` 的 `port` 属性的值为 `6999`，`redirectPort` 的属性值为 `7443`。

16.1.2.2 数据源的配置

通过 apollo 软件的操作界面进行配置，数据的格式可以参考 ETCD 章节下的说明。

16.1.2.3 日志的配置

通过 apollo 软件的操作界面进行配置，数据的格式可以参考 ETCD 章节下的说明。

16.1.2.4 JVM 的配置

通过 apollo 软件的操作界面进行配置，数据的格式可以参考 ETCD 章节下的说明。

16.2 使用 ETCD 集中配置

16.2.1 添加集中配置服务

apusic.conf 配置文件需要增加 plugins 的配置，并指定 etcd 的配置实现。

要支持 ETCD2，则需要指定实现为：

```
<plugins>
  <plugin
    className="com.apusic.ams.config.listener.EtcdPluginLoader"
    clusterId="cluster" host="172.18.100.74"
      port="2379" namespace="test" />
</plugins>
```

要支持 ETCD3 则需要指定为：

```
<plugins>
  <plugin
    className="com.apusic.ams.config.listener.EtcdV3PluginLoader"
    clusterId="cluster" host="172.18.100.74"
      port="2379" namespace="test" />
</plugins>
```

示例如下图所示：

```

1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2
3 <config>
4 <plugins>
5   <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
6
7   <plugin className="com.apusic.ams.config.listener.EtcdPluginLoader" host="xxx" port="xxx" namespace="test-namespace" clusterId="xxx"/>
8 </plugins>
9
10 <listeners>
11   <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
12   <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
13   <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
14   <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
15   <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
16 </listeners>
17
18 <resources>
19   <resource auth="Container" description="User database that can be updated and saved" factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase"
20     pathname="conf/aas-users.xml" type="com.apusic.ams.UserDatabase"/>
21 </resources>
22
23 <executors>
24   <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-" prestartminSpareThreads="true"/>
25 </executors>
26
27 <endpoints>
28   <endpoint URIEncoding="utf-8" acceptCount="100"
29     compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image
30     /png,image/gif,audio/mpeg,video/mp4" compression="on" compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
31     executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192" maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http"
32     port="6999" processorCache="20000" protocol="HTTP/1.1" redirectPort="7443" tcpNoDelay="true">
33 </endpoint>
34 </endpoints>
35
36 <services>
37   <service defaultHost="localhost" name="ams-service">
38     <realm className="com.apusic.ams.realm.LockOutRealm">
39       <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
40     </realm>
41     <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
42     </host>
43   </service>
44 </services>
45
46 <servers>
47   <server endpoints="ams-http" executors="http-thread-pool" name="ams-server" services="ams-service"/>
48 </servers>
49
50 <containers>
51   <container listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"
52     name="container" port="-1" resources="userDatabase" servers="ams-server" shutdown="shutdown"/>
53 </containers>
54 </config>

```

其中 `className="com.apusic.ams.config.listener.EtcdPluginLoader"` 是加载 etcd 配置的类，属性包括了属性说明：

表32：金蝶Apusic应用服务器V10.0.10标准版与ETCD集成配置说明

配置参数	说明	值类型	默认值
host	安装 etcd 的服务器 IP 地址, 必须	字符	
port	etcd 服务的客户端访问端口, 必须	字符	
namespace	应用的命名空间, 必须	字符	
clusterId	应用的集群标识, 必须	字符	

以上属性可通过配置文件配置，也可通过环境变量配置，也可通过系统变量设置

优先级是：环境变量 > 系统变量 > 配置文件

一般环境变量和系统变量的变量名称为大写的，环境变量（`System.getProperty("xxx")` 获取）和系统变量（`System.getenv("xxx")` 获取）的变量名为如下，每个变量名会与上面的属性对应：

表33：金蝶Apusic应用服务器V10.0.10标准版与ETCD集成环境变量配置说明

配置参数	说明	值类型	默认值
AAS_CONFIG_ETCD_HOST	安装 etcd 的服务器 IP 地址, 必须	字符	
AAS_CONFIG_ETCD_PORT	etcd 服务的客户端访问端口, 必须	字符	
AAS_CONFIG_ETCD_NAMESPACE	应用的命名空间, 必须	字符	
AAS_CONFIG_ETCD_CLUSTERID	应用的集群标识, 必须	字符	

16.2.2 在 ETCD 的配置

16.2.2.1 apusic.conf 的配置

与配置中心整合后，应用服务器的配置文件（`apusic.conf` 文件）可通过设置 `key` 为 `apusic_config`，`value` 为要改变的值，进行修改。

在 ETCD 服务器中设置如下的值，则表示修改端口为 6999 和 7443：

```
./etcdctl put /test/cluster/aamsconfig/apusic_config
'{"config.endpoints.endpoint.ams-http":
{"port":"6999","redirectPort":"7443"}}'
```

配置信息最后保存在 `conf` 目录下的 `apusic.conf` 文件。

`config.endpoints.endpoint.ams-http` 拆分对应这 `config` > `endpoints` > `endpoint` > `ams-http` 对应这 xml 的层级结构：

```

1 <?xml version="1.0" encoding="utf-8" standalone="no"?>
2
3 <config>
4   <plugins>
5     <plugin appId="10086" className="com.apusic.ams.config.listener.ApolloPluginLoader" cluster="sz-test" metaUrl="http://172.20.129.231:8080" namespace="test-namespace"/>
6   </plugins>
7
8   <listeners>
9     <listener className="com.apusic.ams.startup.VersionLoggerListener" name="versionLoggerListener"/>
10    <listener className="com.apusic.ams.core.AprLifecycleListener" name="aprLifecycleListener"/>
11    <listener className="com.apusic.ams.core.JreMemoryLeakPreventionListener" name="jreMemoryLeakPreventionListener"/>
12    <listener className="com.apusic.ams.mbeans.GlobalResourcesLifecycleListener" name="globalResourcesLifecycleListener"/>
13    <listener className="com.apusic.ams.core.ThreadLocalLeakPreventionListener" name="threadLocalLeakPreventionListener"/>
14  </listeners>
15
16  <resources>
17    <resource auth="Container" description="User database that can be updated and saved" factory="com.apusic.ams.users.MemoryUserDatabaseFactory" name="userDatabase"
18      pathName="conf/aas-users.xml" type="com.apusic.ams.UserDatabase"/>
19  </resources>
20
21  <executors>
22    <executor maxQueueSize="5000" maxThreads="200" minSpareThreads="25" name="http-thread-pool" namePrefix="http-exec-" prestartminSpareThreads="true"/>
23  </executors>
24
25  <endpoints>
26    <endpoint URIEncoding="utf-8" acceptCount="100"
27      compressableMimeType="text/html,text/plain,text/css,application/javascript,application/json,application/x-font-ttf,application/x-font-otf,image/svg+xml,image/jpeg,image
28      /png,image/gif,audio/mpeg,video/mp4" compression="on" compressionMinSize="2048" connectionTimeout="20000" disableUploadTimeout="true" enableLookups="false"
29      executor="http-thread-pool" keepAliveTimeout="15000" maxConnections="10000" maxHttpHeaderSize="8192" maxKeepAliveRequests="100" maxPostSize="104857600" name="ams-http"
30      port="6999" processorCache="20000" protocol="HTTP/1.1" redirectPort="7443" tcpNoDelay="true">
31    </endpoint>
32  </endpoints>
33
34  <services>
35    <service defaultHost="localhost" name="ams-service">
36      <realm className="com.apusic.ams.realm.LockOutRealm">
37        <realm className="com.apusic.ams.realm.UserDatabaseRealm" resourceName="userDatabase"/>
38      </realm>
39      <host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
40      </host>
41    </service>
42  </services>
43
44  <servers>
45    <server endpoints="ams-http" executors="http-thread-pool" name="ams-server" services="ams-service"/>
46  </servers>
47
48  <containers>
49    <container listeners="versionLoggerListener,aprLifecycleListener,jreMemoryLeakPreventionListener,globalResourcesLifecycleListener,threadLocalLeakPreventionListener"
50      name="container" port="-1" resources="userDatabase" servers="ams-server" shutdown="shutdown"/>
51  </containers>
52 </config>

```

port 和 redirectPort 对应着属性。

```

{
  "config.endpoints.endpoint.ams-http": {
    "port": "6999",
    "redirectPort": "7443"
  }
}

```

上图的意思就是，修改名称为 `ams-http` 的 `xml` 标签为 `endpoint` 的 `port` 属性的值为 `6999`，`redirectPort` 的属性值为 `7443`。

16.2.2.2 数据源的配置

与配置中心整合后，数据源配置的 `key` 为 `datasource_config`，它是一个 `json` 格式的字符串。可以在 `etcd` 中增加如下的配置：

```
./etcdctl put /test/cluster/aamsconfig/datasource_config
'{"jdbc/test":
{"auth":"Container","type":"javax.sql.DataSource","maxActive":"100","us
```

其中 `jdbc/test` 为数据源的名称，如果存在该名称，则进行修改动作，如果不存在该名称，则进行添加的动作。配置信息最后保存在 `conf` 目录下的 `context.xml` 文件。

16.2.2.3 日志的配置

与配置中心整合后，日志的配置的 `key` 为 `logging_conf`，它是一个 json 格式的字符串。可以在 etcd 中增加如下类似的配置，修改多个配置，则增加多个 json 格式的 `key-value`：

```
./etcdctl put /test/cluster/aamsconfig/datasource_config
'{"com.apusic.connector.http2.level":"FINER"}'
```

配置信息最后保存在 `conf` 目录下的 `logging.properties` 文件

16.2.2.4 JVM 的配置

与配置中心整合后，日志的配置的 `key` 为 `jvm_opt`，它是一个普通格式的字符串，不是一个 json 字符串，配置的格式类似如下：

```
./etcdctl put /test/cluster/aamsconfig/jvm_opt '-server -
Djava.net.preferIPv4Stack=true -Xms513m -Xmx1023m -
XX:MaxPermSize=257m'
```

配置信息最后保存在 `bin` 目录下的 `apusic.bat` (window 下) 或 `apusic.sh` (linux 下) 文件。

17 配置 Session 存储

AAS 单机方式有两种 Session 存储方式：`StandardManager`（默认）、`PersistentManager`。

- `StandardManager` 方式把 Session 放在内存中，停止时才把内存中的 Session 保存到文件，再次启动时从文件读取。
- `PersistentManager` 方式又分为文件或者数据库存储。

17.1 配置本地内存存储

默认的 Session 存储就是本地内存存储（`StandardManager`），该方式 Session 是放在内存中，停止时才把内存中的 Session 保存到文件，再次启动时从文件读取。

配置如下：

```
<Manager maxActiveSession="-1">
</Manager>
```

```
<services>
  <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="UserDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <application path="/test" docBase="E:\work_test\test" useHttpOnly="true">
        <Manager
          maxActiveSession="-1">
        </Manager>
      </application>
    </host>
```

参数说明：

表34：金蝶Apusic应用服务器V10.0.10标准版配置本地内存存储参数说明

配置参数	说明	值类型	默认值
maxActiveSession	最大会话数，-1 表示不限制，超过后再创建 Session 会直接抛异常	整型	-1

17.2 配置本地文件存储

```

<services>
  <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <application path="/test" docBase="E:\work_test\test" useHttpOnly="true">
        <Manager className="com.apusic.ams.session.PersistentManager"
          saveOnRestart="true"
          maxActiveSessions="1"
          minIdleSwap="-1"
          maxIdleSwap="-1"
          maxIdleBackup="60">
          <Store className="com.apusic.ams.session.FileStore"/>
        </Manager>
      </application>
    </host>
  </service>
</services>

```

表35：金蝶Apusic应用服务器V10.0.10标准版配置本地文件储存参数说明

配置参数	说明	值类型	默认值
saveOnRestart	true/false;配置服务重启对 Session 的处理，若为 true，则关闭前把有效的 Session 保存，启动后重新载入	布尔值	
maxActiveSessions	活动状态 Session 的最大数，为 -1 时则不限制，否则 Session Manager 将会把超出的 Session 对象转移到 Session Store 中	整型	
minIdleSwap	Session 不活动的最短时间，超过该时间，Session Manager 可能会把该 Session 对象转移到 Session Store 中，单位为秒，该 Session 将不在内存中，如果配置了 minIdleSwap 则两个条件都需要满足	整型	
maxIdleSwap	Session 不活动的最长时间，超过该时间，Session Manager 将会把该 Session 对象转移到 Session Store 中，该 Session 将不在内存中。如果配置了 minIdleSwap 则两个条件都需要满足	整型	
maxIdleBackup	Session 不活动的最长时间，超过该时间，Session Manager 将会把该 Session 对象备份到 Session Store 中，但该 Session 对象依然存在内存中	整型	

Store标签配置如下：

表36：金蝶Apusic应用服务器V10.0.10标准版配置本地文件储存store标签配置参数说明

配置参数	说明	值类型	默认值
className	储存方式类名	字符	com.apusic.ams.session. FileStore

17.3 配置数据库存储

```
<Manager className="com.apusic.ams.session.PersistentManager"
saveOnRestart="true" maxActiveSessions="1"
  minIdleSwap="-1" maxIdleSwap="-1" maxIdleBackup="60">
  <Store className="com.apusic.ams.session.JDBCStore"
driverName="com.mysql.jdbc.Driver"
  connectionURL="jdbc:mysql://127.0.0.1:3306/test_platform?
useUnicode=true&characterEncoding=utf-8"
  connectionName="root" connectionPassword="changeit"
sessionTable="ams_session" sessionIdCol="session_id"
  sessionDataCol="session_data"
sessionValidCol="session_valid" sessionMaxInactiveCol="max_inactive"
  sessionLastAccessedCol="last_access"
sessionAppCol="app_name" />
</Manager>
```

注意：需要自己创建表。

表37：金蝶Apusic应用服务器V10.0.10标准版配置数据库储存参数说明

配置参数	说明	值类型	默认值
saveOnRestart	配置服务重启对 Session 的处理，若为 true，则关闭前把有效的 Session 保存，启动后重新载入	布尔值	
maxActiveSessions	活动状态 Session 的最大数，为 -1 时则不限制，否则 Session Manager 将会把超出的 Session 对象转移到 Session Store 中	整型	
minIdleSwap	Session 不活动的最短时间，超过该时间，Session Manager 可能会把该 Session 对象转移到 Session Store 中，单位为秒，该 Session 将	整型	

	不在内存中，如果配置了 minIdleSwap 则两个条件都需要满足		
maxIdleSwap	Session 不活动的最长时间，超过该时间，Session Manager 将会把该 Session 对象转移到 Session Store 中，该 Session 将不在内存中。如果配置了 minIdleSwap 则两个条件都需要满足	整型	
maxIdleBackup	Session 不活动的最长时间，超过该时间，Session Manager 将会把该 Session 对象备份到 Session Store 中，但该 Session 对象依然存在内存中	整型	

Store标签中配置如下：

表38：金蝶Apusic应用服务器V10.0.10标准版配置数据域文件储存store标签参数说明

配置参数	说明	值类型	默认值
className	储存方式类名	字符	com.apusic.ams.session.JDBCStore
driverName	数据库驱动类名，根据数据源类型填写	字符	
connectionURL	连接数据源URL，根据数据源连接方式填写	字符	
connectionName	连接数据源用户名，需要有远程连接权限	字符	
connectionPassword	连接数据源用户名密码	字符	
sessionTable	session储存表名，需要在数据库中提前创建	字符	
sessionIdCol	session储存ID列名，需要在数据库中提前创建	字符	
sessionDataCol	session储存数据列名，需要在数据库中提前创建	字符	
sessionValidCol	session储存值列名，需要在数据库中提前创建	字符	
sessionMaxInactiveCol	非活动最大值列名，需要在数据库中提前创建	字符	
sessionLastAccessedCol	最后储存数据列名，需要在数据库中提前创建	字符	

sessionAppCol	应用名称列名，需要在数据库中提前创建	字符	
---------------	--------------------	----	--

18 集群方案

18.1 Session 组播同步

Session 同步需要满足以下几个要求：

1. Session 的属性必须实现 `java.io.Serializable` 接口
2. 负载均衡器需要配置会话粘滞模式 (sticky Session mode) ，测试时可不开启会话粘滞
3. 多台机器时间需要同步
4. 保证多播端口可用
5. 应用的 `web.xml` 需要加入 `<distributable/>` ;
6. 应用使用文件夹方式部署

18.1.1 添加集群配置

在需要加入集群的应用服务器中的 `apusic.conf` 文件的 `host` 或者 `service` 里面添加如下内容 (注意：在 `service` 下添加表示对所有虚拟主机均启用集群功能，否则对某个虚拟主机启用集群功能)

```
<Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
channelSendOptions="6">
  <Manager className="com.apusic.ams.ha.session.BackupManager"
expireSessionsOnShutdown="false"
  notifyListenersOnReplication="true" />
  <Channel className="com.apusic.ams.tribes.group.GroupChannel">
    <Membership
className="com.apusic.ams.tribes.membership.McastService"
address="228.0.0.4" port="45564"
      frequency="500" dropTime="3000" />
    <Receiver
className="com.apusic.ams.tribes.transport.nio.NioReceiver"
address="172.20.50.43" port="5000"
      selectorTimeout="100" maxThreads="6" />
    <Sender
className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
      <Transport
className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"
/>>
  </Channel>
</Cluster>
```

```
        </Sender>
        <Interceptor
className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"
/>
        <Interceptor
className="com.apusic.ams.tribes.group.interceptors.MessageDispatchIntercepto
/>
        <Interceptor
className="com.apusic.ams.tribes.group.interceptors.ThroughputIntercepto
/>
    </Channel>
    <Valve className="com.apusic.ams.ha.tcp.ReplicationValve"
filter="" />
    <Valve className="com.apusic.ams.ha.session.JvmRouteBinderValve"
/>
    <ClusterListener
className="com.apusic.ams.ha.session.ClusterSessionListener" />
</Cluster>
```

```

<services>
  <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
    <realm className="com.apusic.ams.realm.LockOutRealm">
      <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
    </realm>
    <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
      <application path="/test" docBase="E:\work_test\test" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext"/>
    </host>
    <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
      channelSendOptions="6">
      <Manager className="com.apusic.ams.ha.session.DeltaManager"
        expireSessionsOnShutdown="false"
        notifyListenersOnReplication="true"/>
      <Channel className="com.apusic.ams.tribes.group.GroupChannel">
        <Membership className="com.apusic.ams.tribes.membership.McastService"
          address="228.0.0.4"
          port="45564"
          frequency="500"
          dropTime="3000"/>
        <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
          address="auto"
          port="5000"
          selectorTimeout="100"
          maxThreads="6"/>
        <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
          <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
        </Sender>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
        <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
      </Channel>
      <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
      <Valve className="com.apusic.ams.ha.session.JvmRouteBinderValve"/>
      <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
    </Cluster>
  </service>
</services>
<servers>
  <server name="ams-server" services="ams-service" endpoints="ams-http" executors="http-thread-pool"/>
</servers>

```

注意：需要把 `NioReceiver` 里面的 `address` 修改为当前机器的 IP 地址。

重要配置详解：

Cluster: 应用服务器集群配置。

表39: 金蝶Apusic应用服务器V10.0.10标准版集群配置cluster标签说明

配置参数	说明	值类型	默认值
className	实现集群功能的类名称	字符	com.apusic.ams.ha.tcp.SimpleTcpCluster
channelSendOptions	消息发送选项，可以设置为 2（确认发送），4（同步发送），8（异步发送）以及它们的逻辑或运算结果，默认是 8，如果希望异步发送且确认发送结果，那么设置为 2+8=10	字符	

Manager: 集群会话管理器。

表40: 金蝶Apusic应用服务器V10.0.10标准版集群配置manager标签说明

配置参数	说明	值类型	默认值
className	集群会话管理器类，会话改变了会同步给集群中的其它节点	字符	com.apusic.ams.ha.session.BackupManager
expireSessionsOnShutdown	设置为 true 时，一个节点关闭，将导致集群下的所有 Session 失效	布尔值	false
notifyListenersOnReplication	如果希望当集群中的节点上复制或删除会话属性时通知会话监听器就设置 notifyListenersOnReplication 为 true	布尔值	
mapSendOptions	为 6 表示 BackupManager 同步发送消息，参考 "Cluster"的 channelSendOptions	字符	

Channel: 是主组件，`channel`管理一组子组件，并和它们一起组成了AAS实例间的通讯框架。在AAS集群中，`DeltaManager`通过`SimpleTcpCluster`调用channel来实现信息传递，而`BackupManager`自己调用`channel`以及子组件这些组件来实现信息传递。`ReplicatedContext`也会调用`channel`传递`context`属性。

表41: 金蝶Apsusic应用服务器V10.0.10标准版集群配置channel标签说明

配置参数	说明	值类型	默认值
className	Channel类名	字符	com.apusic.ams.tribes.group.GroupChannel

Membership: 集群节点配置。

表42: 金蝶Apsusic应用服务器V10.0.10标准版集群配置membership标签说明

配置参数	说明	值类型	默认值
className	集群节点信息管理类名	字符	com.apusic.ams.tribes.membership.McastServicer
address	多播地址	字符	
port	多播端口	字符	
frequency	发送心跳时间间隔，单位毫秒	整型	
dropTime	该 dropTime 配置的时间内没有收到某个心跳，那么这个心跳所属节点就会从当前	整型	

	节点维护的节点列表删除, 单位毫秒		
--	----------------------	--	--

****Receiver:**** 接收器，负责接收消息。

表43：金蝶Apusic应用服务器V10.0.10标准版集群配置receiver标签说明

配置参数	说明	值类型	默认值
className	消息接收类名	字符	com.apusic.ams.tribes.transport.nio.NioReceiver
address	节点监听地址	字符	
port	节点监听端口	字符	
selectorTimeout	接收器的 selector 超时时间	整型	
maxThreads	最大线程数	整型	

Sender: 发送器，负责将消息发送给其他节点的 Receiver。

表44：金蝶Apusic应用服务器V10.0.10标准版集群配置sender标签说明

配置参数	说明	值类型	默认值
className	Sender类名	字符	com.apusic.ams.tribes.transport.ReplicationTransmitter

****Interceptor:**** 拦截器。

表45：金蝶Apusic应用服务器V10.0.10标准版集群配置interceptor标签说明

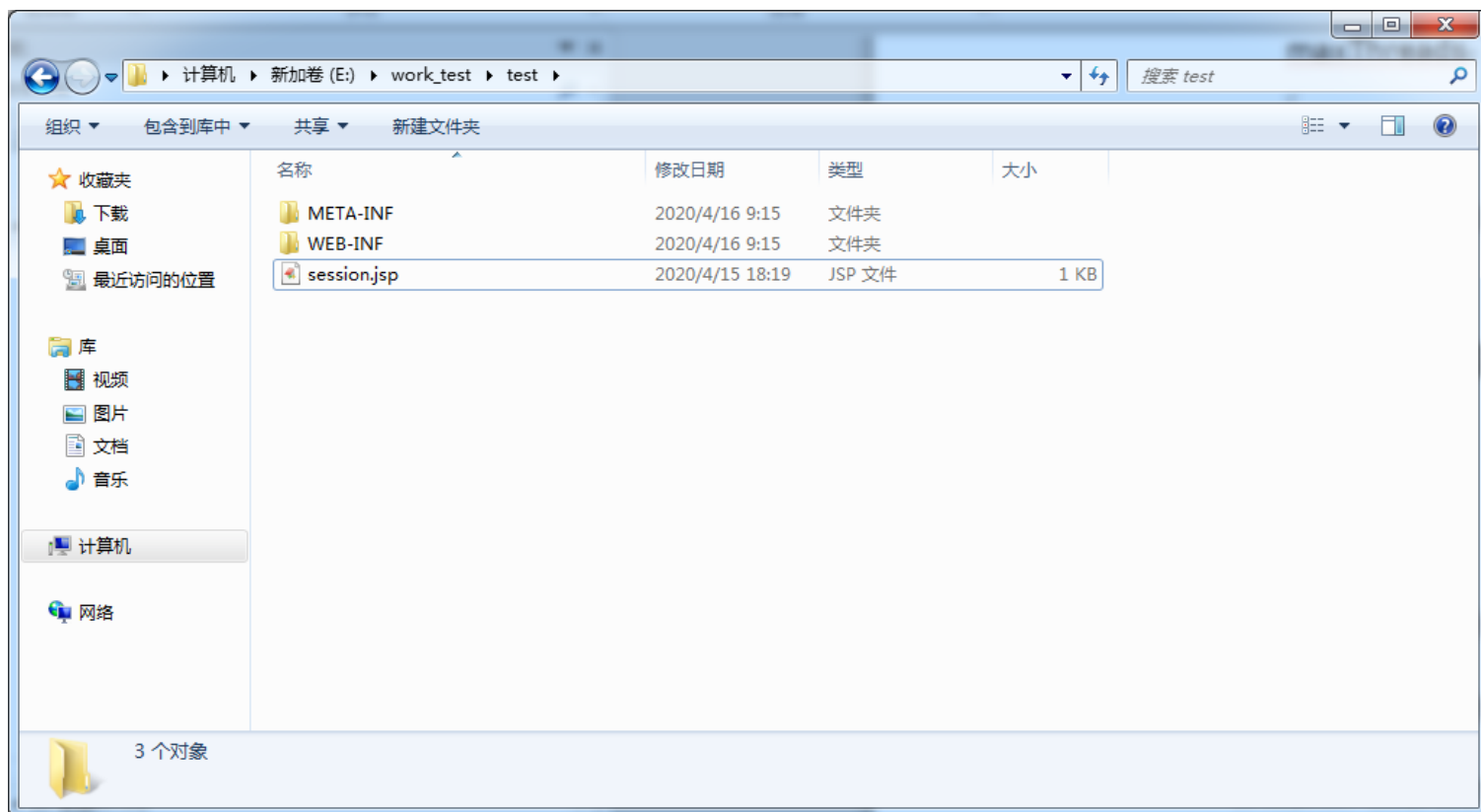
配置参数	说明	值类型	默认值
className	Interceptor类名	字符	

18.1.2 部署应用和设置 jvmRoute

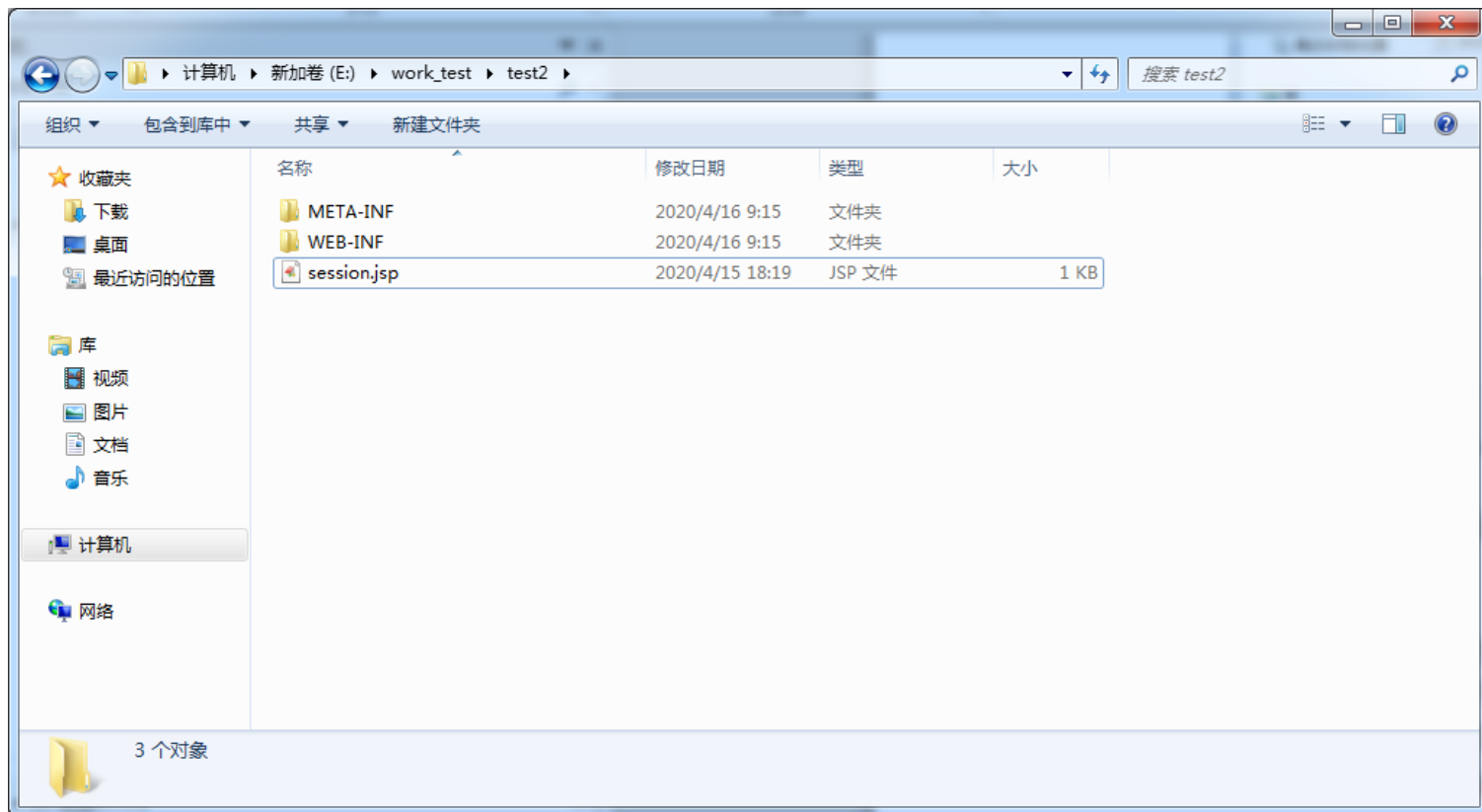
1. 测试应用

`test.war` 是用于测试集群的应用。里面只有一个 `session.jsp`，需要把应用解压进行部署。

节点 1 是解压到 e:/work_test/test



节点 2 是解压到 `e:/work_test/test2`



`session.jsp` 内容:

```

1 <%@ page import="java.text.SimpleDateFormat" %>
2 <%@ page import="java.util.Date" %>
3 <%@ page import="java.util.logging.Logger" %>
4 <%@ page language="java" contentType="text/html; charset=UTF-8"
5   pageEncoding="UTF-8"%>
6 <%
7   Logger logger = Logger.getLogger("test");
8   String str = (String) request.getSession().getAttribute("test");
9   if(null == str) {
10    str = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
11    request.getSession().setAttribute("test", str);
12    logger.info("session put: " + str);
13   } else {
14    logger.info("session get: " + str);
15   }
16 %>

```

往 Session 里面获取数据，没有数据再往里放数据。

修改 web.xml 文件，添加 <distributable/>。

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID">
3   <display-name>test</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12  <distributable/>
13 </web-app>

```

2. 部署应用和修改配置

修改 apusic.conf 文件的 service 加上 jvmRoute="nodeXXX"。

确保每个 jvmRoute 都是不同。还需要在 application 标签加上
className="com.apusic.ams.ha.context.ReplicatedContext"。

这里只有两个节点，分别为 node01、node02

```

85
86 <services>
87 <service name="ams-service" defaultHost="localhost" jvmRoute="node02">
88 <realm className="com.apusic.ams.realm.LockOutRealm">
89 <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
90 </realm>
91 <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
92 <application path="/test" docBase="E:\work_test\test2" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext">
93 </host>
94 <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
95 channelSendOptions="6">
96
97 <Manager className="com.apusic.ams.ha.session.DeltaManager"
98 expireSessionsOnShutdown="false"
99 notifyListenersOnReplication="true"/>
100
101 <Channel className="com.apusic.ams.tribes.group.GroupChannel">
102 <Membership className="com.apusic.ams.tribes.membership.McastService"
103 address="228.0.0.4"
104 port="45564"
105 frequency="500"
106 dropTime="3000"/>
107 <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
108 address="auto"
109 port="5000"
110 selectorTimeout="100"
111 maxThreads="6"/>
112
113 <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
114 <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
115 </Sender>
116 <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
117 <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
118 <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
119 </Channel>
120
121 <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
122 <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
123 </Cluster>
124 </service>
125 </services>

```

```

84 </endpoints>
85
86 <services>
87 <service name="ams-service" defaultHost="localhost" jvmRoute="node01">
88 <realm className="com.apusic.ams.realm.LockOutRealm">
89 <realm resourceName="userDatabase" className="com.apusic.ams.realm.UserDatabaseRealm"/>
90 </realm>
91 <host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
92 <application path="/test" docBase="E:\work_test\test" useHttpOnly="true" className="com.apusic.ams.ha.context.ReplicatedContext">
93 </host>
94 <Cluster className="com.apusic.ams.ha.tcp.SimpleTcpCluster"
95 channelSendOptions="6">
96 <Manager className="com.apusic.ams.ha.session.DeltaManager"
97 expireSessionsOnShutdown="false"
98 notifyListenersOnReplication="true"/>
99 <Channel className="com.apusic.ams.tribes.group.GroupChannel">
100 <Membership className="com.apusic.ams.tribes.membership.McastService"
101 address="228.0.0.4"
102 port="45564"
103 frequency="500"
104 dropTime="3000"/>
105 <Receiver className="com.apusic.ams.tribes.transport.nio.NioReceiver"
106 address="auto"
107 port="5000"
108 selectorTimeout="100"
109 maxThreads="6"/>
110 <Sender className="com.apusic.ams.tribes.transport.ReplicationTransmitter">
111 <Transport className="com.apusic.ams.tribes.transport.nio.PooledParallelSender"/>
112 </Sender>
113 <Interceptor className="com.apusic.ams.tribes.group.interceptors.TcpFailureDetector"/>
114 <Interceptor className="com.apusic.ams.tribes.group.interceptors.MessageDispatch15Interceptor"/>
115 <Interceptor className="com.apusic.ams.tribes.group.interceptors.ThroughputInterceptor"/>
116 </Channel>
117 <Valve className="com.apusic.ams.ha.tcp.ReplicationValve" filter=""/>
118 <Valve className="com.apusic.ams.ha.session.JvmRouteBinderValve"/>
119 <ClusterListener className="com.apusic.ams.ha.session.ClusterSessionListener"/>
120 </Cluster>
121 </service>
122 </services>
123

```

18.1.3 配置 Apache

安装好 Apache 后，向配置文件 `httpd.conf` 末尾添加如下内容：


```

C:\Windows\System32\cmd.exe - apusic.bat run
E:\work_test\aa.ms\bin>
E:\work_test\aa.ms\bin>
E:\work_test\aa.ms\bin>
E:\work_test\aa.ms\bin>
E:\work_test\aa.ms\bin>
E:\work_test\aa.ms\bin>apusic.bat run
Using APUSIC_BASE: "E:\work_test\aa.ms"
Using APUSIC_HOME: "E:\work_test\aa.ms"
Using APUSIC_TMPDIR: "E:\work_test\aa.ms\temp"
Using JRE_HOME: "F:\tools\jdk1.8.0_201"
Using CLASSPATH: "E:\work_test\aa.ms\bin\bootstrap.jar;E:\work_test\aa.ms\bin\aa.ms-juli.jar"
16-Apr-2020 14:54:23.837 [main] com.apusic.ams.startup.Apusic.initDirs 在E:\work_test\aa.ms\temp\找不到指定的临时文件夹
16-Apr-2020 14:54:24.134 [main] com.apusic.ams.startup.VersionLoggerListener.log Server. 服务器版本: Apusic AAS/10.1.0
16-Apr-2020 14:54:24.136 [main] com.apusic.ams.startup.VersionLoggerListener.log 服务器构建: Aug 27 2019 01:38:38 UTC
16-Apr-2020 14:54:24.138 [main] com.apusic.ams.startup.VersionLoggerListener.log 服务器版本号: 10.1.0.0
16-Apr-2020 14:54:24.140 [main] com.apusic.ams.startup.VersionLoggerListener.log OS Name: Windows 7
16-Apr-2020 14:54:24.141 [main] com.apusic.ams.startup.VersionLoggerListener.log OS 版本: 6.1
16-Apr-2020 14:54:24.143 [main] com.apusic.ams.startup.VersionLoggerListener.log 架构: amd64
16-Apr-2020 14:54:24.144 [main] com.apusic.ams.startup.VersionLoggerListener.log Java 环境变量: F:\tools\jdk1.8.0_201\jre
16-Apr-2020 14:54:24.146 [main] com.apusic.ams.startup.VersionLoggerListener.log JUM 版本: 1.8.0_201-b09
16-Apr-2020 14:54:24.148 [main] com.apusic.ams.startup.VersionLoggerListener.log JUM 供应商: Oracle Corporation
16-Apr-2020 14:54:24.150 [main] com.apusic.ams.startup.VersionLoggerListener.log CATALINA_BASE: E:\work_test\aa.ms
16-Apr-2020 14:54:24.152 [main] com.apusic.ams.startup.VersionLoggerListener.log CATALINA_HOME: E:\work_test\aa.ms
16-Apr-2020 14:54:24.154 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=
16-Apr-2020 14:54:24.156 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=com.apusic.logging.manager
16-Apr-2020 14:54:24.159 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
16-Apr-2020 14:54:24.161 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dfile.encoding=UTF-8
16-Apr-2020 14:54:24.162 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=com.apusic.protocol.handler
16-Apr-2020 14:54:24.165 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dignore.endorsed.dirs=
16-Apr-2020 14:54:24.167 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dapusic.base=E:\work_test\aa.ms
16-Apr-2020 14:54:24.169 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dapusic.home=E:\work_test\aa.ms
16-Apr-2020 14:54:24.171 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=E:\work_test\aa.ms\temp
16-Apr-2020 14:54:24.173 [main] com.apusic.ams.core.AprLifecycleListener.lifecycleEvent The APR based Apusic AAS Native library which was successfully loaded
16-Apr-2020 14:54:24.414 [main] com.apusic.connector.AbstractProtocol.init 初始化协议处理器 ["http-nio-8888"]
16-Apr-2020 14:54:24.491 [main] com.apusic.ams.startup.Apusic.load 服务器在[652]毫秒内初始化
16-Apr-2020 14:54:24.515 [main] com.apusic.ams.core.StandardService.startInternal Starting service [ams-server]
16-Apr-2020 14:54:24.519 [main] com.apusic.ams.core.StandardEngine.startInternal 正在启动 Servlet 引擎: [Apusic AAS/10.1.0]
16-Apr-2020 14:54:24.528 [main] com.apusic.ams.tcp.SimpleTcpCluster.startInternal Cluster is about to start
16-Apr-2020 14:54:24.548 [main] com.apusic.ams.tribes.transport.ReceiverBase.bind 服务器套接字接收器绑定到: [/172.20.50.43:4001]
16-Apr-2020 14:54:24.559 [main] com.apusic.ams.tribes.membership.McastServiceImpl.setupSocket 设置集群多播超时时间: [500]
16-Apr-2020 14:54:24.563 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to allow members to join
16-Apr-2020 14:54:24.688 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:25.565 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:25.577 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to allow members to join
16-Apr-2020 14:54:25.593 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:26.592 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.407 [main] com.apusic.jasper.servlet.TldScanner.scanJars 至少有一个JAR被扫描用于TLD但尚未包含TLD。为此记录器启用
16-Apr-2020 14:54:27.421 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.423 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.432 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.539 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.589 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.611 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.620 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.627 [main] com.apusic.ams.startup.Apusic.start Server startup in [3,133] milliseconds

```

节点 node01 输出节点加入信息:

```

管理: C:\Windows\System32\cmd.exe - apusic.bat run
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>
G:\project\saas\saas\output\build\bin>apusic.bat run
Using APUSIC_BASE: "G:\project\saas\saas\output\build"
Using APUSIC_HOME: "G:\project\saas\saas\output\build"
Using APUSIC_TMPDIR: "G:\project\saas\saas\output\build\temp"
Using JRE_HOME: "F:\tools\jdk1.8.0_201"
Using CLASSPATH: "G:\project\saas\saas\output\build\bin\bootstrap.jar;G:\project\saas\saas\output\build\bin\saas-juli.jar"
16-Apr-2020 14:52:20.204 [main] com.apusic.ans.startup.VersionLoggerListener.log Server 服务器版本: Apusic AAS/10.1.0
16-Apr-2020 14:52:20.209 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器构建: Aug 27 2019 01:38:38 UTC
16-Apr-2020 14:52:20.211 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器版本号: 10.1.0.0
16-Apr-2020 14:52:20.212 [main] com.apusic.ans.startup.VersionLoggerListener.log OS Name: Windows 7
16-Apr-2020 14:52:20.214 [main] com.apusic.ans.startup.VersionLoggerListener.log OS 版本: 6.1
16-Apr-2020 14:52:20.215 [main] com.apusic.ans.startup.VersionLoggerListener.log 架构: amd64
16-Apr-2020 14:52:20.217 [main] com.apusic.ans.startup.VersionLoggerListener.log Java 环境变量: F:\tools\jdk1.8.0_201\jre
16-Apr-2020 14:52:20.219 [main] com.apusic.ans.startup.VersionLoggerListener.log JVM 版本: 1.8.0_201-b09
16-Apr-2020 14:52:20.221 [main] com.apusic.ans.startup.VersionLoggerListener.log JVM 供应商: Oracle Corporation
16-Apr-2020 14:52:20.222 [main] com.apusic.ans.startup.VersionLoggerListener.log CATALINA_BASE: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.226 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=G:\project\saas\saas\output\build\conf\logging.p
16-Apr-2020 14:52:20.229 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=com.apusic.juli.ClassLoaderLogManager
16-Apr-2020 14:52:20.232 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
16-Apr-2020 14:52:20.234 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dfile.encoding=UTF-8
16-Apr-2020 14:52:20.238 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=com.apusic.ans.webresources
16-Apr-2020 14:52:20.240 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dignore.endorsed.dirs=
16-Apr-2020 14:52:20.242 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.base=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.245 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.home=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.247 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=G:\project\saas\saas\output\build\temp
16-Apr-2020 14:52:20.249 [main] com.apusic.ans.core.AppLifecycleListener.lifecycleEvent The APR based Apusic AAS Native Library which allows optimal performance in production environme
Windows\System32\cmd;C:\Windows\System32\WindowsPowerShell\v1.0\;F:\tools\apache-maven-3.6.0\bin;F:\软件安装\UltraEdit;F:\tools\git\cmd;F:\tools\apache-ant-1.10.5\bin;F:\tools\gradle-5.4.1
16-Apr-2020 14:52:20.494 [main] com.apusic.ans.startup.Apusic.load 服务器在[651]毫秒内初始化
16-Apr-2020 14:52:20.593 [main] com.apusic.ans.core.StandardService.startInternal Starting service [ans-server]
16-Apr-2020 14:52:20.597 [main] com.apusic.ans.core.StandardEngine.startInternal 正在启动 Servlet 引擎: [Apusic AAS/10.1.0]
16-Apr-2020 14:52:20.607 [main] com.apusic.ans.ha.tcp.SimpleTcpCluster.startInternal Cluster is about to start
16-Apr-2020 14:52:20.626 [main] com.apusic.ans.tribes.transport.ReceiverBase.bind 服务器套接字接收器绑定到: [/172.20.50.43:4000]
16-Apr-2020 14:52:20.636 [main] com.apusic.ans.tribes.membership.McastServiceImpl.setupSocket 设置集群多播超时时间: [500]
16-Apr-2020 14:52:20.640 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership, start level:[4]
16-Apr-2020 14:52:21.643 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established, start level:[4]
16-Apr-2020 14:52:21.657 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership, start level:[8]
16-Apr-2020 14:52:22.668 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established, start level:[8]
16-Apr-2020 14:52:23.491 [main] com.apusic.jasper.servlet.TldScanner.scanJars 至少有一个JAR被扫描用于TLD但尚未包含TLD。为此记录器启用调试日志记录,以获取已扫描但未在其中找到TLD的完整J
16-Apr-2020 14:52:23.505 [main] com.apusic.ans.ha.session.DeltaManager.startInternal 将管理器[localhost#test]注册到名为[ans-service]的集群元素[Engine]
16-Apr-2020 14:52:23.508 [main] com.apusic.ans.ha.session.DeltaManager.startInternal Starting clustering manager at [localhost#test]
16-Apr-2020 14:52:23.510 [main] com.apusic.ans.ha.session.DeltaManager.getAllClusterSessions Manager [localhost#test]: skipping state transfer. No members active in cluster group.
16-Apr-2020 14:52:23.533 [main] com.apusic.ans.tribes.tipis.AbstractReplicatedMap.init Initializing AbstractReplicatedMap with context name:[test]
16-Apr-2020 14:52:23.536 [main] com.apusic.ans.tribes.tipis.AbstractReplicatedMap.init AbstractReplicatedMap[localhost#test] initialization was completed in [3] ms.
16-Apr-2020 14:52:23.540 [main] com.apusic.ans.ha.session.JvmRouteBinderValue.startInternal JvmRouteBinderValue 启动
16-Apr-2020 14:52:23.548 [main] com.apusic.connector.AbstractProtocol.start 开始协议处理句柄["http-nio-6888"]
16-Apr-2020 14:52:23.548 [main] com.apusic.ans.startup.Apusic.start Server startup in [2,982] milliseconds
16-Apr-2020 14:52:23.554 [main] com.apusic.ans.tribes.io.BufferPool.getBufferPool Created a buffer pool with max size:[104857600] bytes of type:
16-Apr-2020 14:52:57.786 [Memberships-MemberAdded.] com.apusic.ans.ha.tcp.SimpleTcpCluster.memberAdded Replication member added:[com.apusic.ans.tribes.membership.MemberImpltcp://172.
16-Apr-2020 14:54:21.286 [TribeTask-Receiver[ans-service-Channel]-6] com.apusic.ans.tribes.tipis.ReplicatedMap.memberDisappeared 成员[com.apusic.ans.tribes.membership.MemberImpltcp://172.
16-Apr-2020 14:54:21.287 [TribeTask-Receiver[ans-service-Channel]-6] com.apusic.ans.tribes.tipis.ReplicatedMap.memberDisappeared 成员[com.apusic.ans.tribes.membership.MemberImpltcp://172.
16-Apr-2020 14:54:21.389 [Memberships-MemberDisappeared.] com.apusic.ans.ha.tcp.SimpleTcpCluster.memberDisappeared 收到成员消失:[com.apusic.ans.tribes.membership.MemberImpltcp://172.
16-Apr-2020 14:54:25.579 [Memberships-MemberAdded.] com.apusic.ans.ha.tcp.SimpleTcpCluster.memberAdded Replication member added:[com.apusic.ans.tribes.membership.MemberImpltcp://172.

```

访问 <http://172.20.50.43:8080/test/session.jsp>, 多刷新几次:

```

管理: C:\Windows\System32\cmd.exe - apusic.bat run
Using JRE_HOME: "F:\tools\jdk1.8.0_201"
Using CLASSPATH: "G:\project\saas\saas\output\build\bin\bootstrap.jar;G:\project\saas\saas\output\build\bin\saas-juli.jar"
16-Apr-2020 14:52:20.204 [main] com.apusic.ans.startup.VersionLoggerListener.log Server 服务器版本: Apusic AAS/10.1.0
16-Apr-2020 14:52:20.209 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器构建: Aug 27 2019 01:38:38 UTC
16-Apr-2020 14:52:20.211 [main] com.apusic.ans.startup.VersionLoggerListener.log 服务器版本号: 10.1.0.0
16-Apr-2020 14:52:20.212 [main] com.apusic.ans.startup.VersionLoggerListener.log OS Name: Windows 7
16-Apr-2020 14:52:20.214 [main] com.apusic.ans.startup.VersionLoggerListener.log OS 版本: 6.1
16-Apr-2020 14:52:20.215 [main] com.apusic.ans.startup.VersionLoggerListener.log 架构: amd64
16-Apr-2020 14:52:20.217 [main] com.apusic.ans.startup.VersionLoggerListener.log Java 环境变量: F:\tools\jdk1.8.0_201\jre
16-Apr-2020 14:52:20.219 [main] com.apusic.ans.startup.VersionLoggerListener.log JVM 版本: 1.8.0_201-b09
16-Apr-2020 14:52:20.221 [main] com.apusic.ans.startup.VersionLoggerListener.log JVM 供应商: Oracle Corporation
16-Apr-2020 14:52:20.222 [main] com.apusic.ans.startup.VersionLoggerListener.log CATALINA_BASE: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.226 [main] com.apusic.ans.startup.VersionLoggerListener.log CATALINA_HOME: G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.229 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=G:\project\saas\saas\output\build\conf\logging.p
16-Apr-2020 14:52:20.232 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
16-Apr-2020 14:52:20.234 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dfile.encoding=UTF-8
16-Apr-2020 14:52:20.236 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=com.apusic.ans.webresources
16-Apr-2020 14:52:20.238 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dignore.endorsed.dirs=
16-Apr-2020 14:52:20.240 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.base=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.242 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.home=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.245 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Dapusic.home=G:\project\saas\saas\output\build
16-Apr-2020 14:52:20.247 [main] com.apusic.ans.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=G:\project\saas\saas\output\build\temp
16-Apr-2020 14:52:20.249 [main] com.apusic.ans.core.AppLifecycleListener lifecycleEvent The APR based Apusic AAS Native library which allows optimal performance in production environme
Windows\System32\Nhaem;C:\Windows\System32\WindowsPowerShell\v1.0\F:\tools\apache-naven-3.6.0\bin\F:\tools\apache-ant-1.10.5\bin\F:\tools\gradle-5.4.1
16-Apr-2020 14:52:20.494 [main] com.apusic.connector.AbstractProtocol.init 初始化协议处理器 ["http-nio-6888"]
16-Apr-2020 14:52:20.570 [main] com.apusic.ans.startup.Apusic.start Server 服务器在 [651]毫秒内初始化
16-Apr-2020 14:52:20.593 [main] com.apusic.ans.core.StandardService.startInternal Starting service [ans-server]
16-Apr-2020 14:52:20.597 [main] com.apusic.ans.core.StandardEngine.startInternal 正在启动 Servlet 引擎; [Apusic AAS/10.1.0]
16-Apr-2020 14:52:20.607 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal Starting cluster is about to start
16-Apr-2020 14:52:20.626 [main] com.apusic.ans.tribes.transport.ReceiverBase.bind 服务器套接字接收器绑定到: [/172.20.50.43:4000]
16-Apr-2020 14:52:20.636 [main] com.apusic.ans.tribes.membership.McastServiceImpl.setupSocket 设置集群多播超时时间: [500]
16-Apr-2020 14:52:20.640 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership, start level:[4]
16-Apr-2020 14:52:21.643 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established, start level:[4]
16-Apr-2020 14:52:21.657 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to establish cluster membership, start level:[8]
16-Apr-2020 14:52:22.668 [main] com.apusic.ans.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established, start level:[8]
16-Apr-2020 14:52:23.491 [main] com.apusic.jasper.servlet.TldScanner.scanJars 至少有一个JAR被扫描用于TLD但尚未包含TLD。为此记录器启用调试日志记录,以获取已扫描但未在其中找到TLD的完整
16-Apr-2020 14:52:23.505 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal 将管理器 [localhost#test]注册到名为 [ans-service]的集群元素 [Engine]
16-Apr-2020 14:52:23.508 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal Starting clustering manager at [localhost#test]
16-Apr-2020 14:52:23.510 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal skipping state transfer. No members active in cluster group.
16-Apr-2020 14:52:23.533 [main] com.apusic.ans.tribes.tipis.AbstractReplicatedMap.init Initializing AbstractReplicatedMap with context name:[test]
16-Apr-2020 14:52:23.536 [main] com.apusic.ans.tribes.tipis.AbstractReplicatedMap.init AbstractReplicatedMap[test] initialization was completed in [3] ms.
16-Apr-2020 14:52:23.540 [main] com.apusic.ans.tcp.SimpleTcpCluster.startInternal Starting clustering manager at [localhost#test]
16-Apr-2020 14:52:23.548 [main] com.apusic.connector.AbstractProtocol.start 开始协议处理句柄 ["http-nio-6888"] 启动
16-Apr-2020 14:52:23.554 [main] com.apusic.ans.startup.Apusic.start Server startup in [2.982] milliseconds
16-Apr-2020 14:52:57.706 [ Tribes-Task-Receiver[ans-service-Channel]-1] com.apusic.ans.tribes.io.BufferPool.getBufferPool Created a buffer pool with max size:[104857600] bytes of type:
16-Apr-2020 14:52:58.506 [ Membership-MemberAdded.1] com.apusic.ans.tcp.SimpleTcpCluster.memberAdded Replication member added: [com.apusic.ans.tribes.membership.MemberImpltcp://172.
16-Apr-2020 14:54:21.286 [ Tribes-Task-Receiver[ans-service-Channel]-6] com.apusic.ans.tribes.tipis.ReplicatedMap.memberDisappeared 成员 [com.apusic.ans.tribes.membership.MemberImpltcp://172.
16-Apr-2020 14:54:21.287 [ Tribes-Task-Receiver[ans-service-Channel]-6] com.apusic.ans.tribes.tipis.ReplicatedMap.memberDisappeared map 条目的重定位在 [0] ms内完成。
16-Apr-2020 14:54:21.309 [ Membership-MemberDisappeared.1] com.apusic.ans.tcp.SimpleTcpCluster.memberDisappeared 收到成员消失: [com.apusic.ans.tribes.membership.MemberImpltcp://172.
16-Apr-2020 14:54:21.329 [ Membership-MemberAdded.1] com.apusic.ans.tcp.SimpleTcpCluster.memberAdded Replication member added: [com.apusic.ans.tribes.membership.MemberImpltcp://172.
16-Apr-2020 14:55:56.558 [ Http-exec-1] org.apache.jsp.session_jsp._jspService session put: 2020-04-16 14:55:56
16-Apr-2020 14:56:00.195 [ Http-exec-2] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.082 [ Http-exec-4] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.429 [ Http-exec-5] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.766 [ Http-exec-6] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:42.122 [ Http-exec-7] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:42.460 [ Http-exec-8] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56

```

```

C:\Windows\System32\cmd.exe - apusic.bat run

E:\work_test\aa.ms\bin>
E:\work_test\aa.ms\bin>apusic.bat run
Using APUSIC_BASE: "E:\work_test\aa.ms"
Using APUSIC_HOME: "E:\work_test\aa.ms"
Using APUSIC_TMPDIR: "E:\work_test\aa.ms\temp"
Using JRE_HOME: "F:\tools\jdk1.8.0_201"
Using CLASSPATH: "E:\work_test\aa.ms\bin\bootstrap.jar;E:\work_test\aa.ms\bin\aa.ms-juli.jar"
16-Apr-2020 14:54:23.837 [main] com.apusic.ams.startup.Apusic.initDirs 在E:\work_test\aa.ms\temp找不到指定的临时文件夹
16-Apr-2020 14:54:24.134 [main] com.apusic.ams.startup.VersionLoggerListener.log Server: 服务器版本: Apusic AAS/10.1.0
16-Apr-2020 14:54:24.136 [main] com.apusic.ams.startup.VersionLoggerListener.log 服务器构建: Aug 27 2019 01:38:38 UTC
16-Apr-2020 14:54:24.138 [main] com.apusic.ams.startup.VersionLoggerListener.log 服务器版本号: 10.1.0.0
16-Apr-2020 14:54:24.140 [main] com.apusic.ams.startup.VersionLoggerListener.log OS Name: Windows 7
16-Apr-2020 14:54:24.141 [main] com.apusic.ams.startup.VersionLoggerListener.log OS 版本: 6.1
16-Apr-2020 14:54:24.143 [main] com.apusic.ams.startup.VersionLoggerListener.log 架构: amd64
16-Apr-2020 14:54:24.144 [main] com.apusic.ams.startup.VersionLoggerListener.log Java 环境变量: F:\tools\jdk1.8.0_201\jre
16-Apr-2020 14:54:24.146 [main] com.apusic.ams.startup.VersionLoggerListener.log JVM 版本: 1.8.0_201-b09
16-Apr-2020 14:54:24.148 [main] com.apusic.ams.startup.VersionLoggerListener.log JVM 供应商: Oracle Corporation
16-Apr-2020 14:54:24.150 [main] com.apusic.ams.startup.VersionLoggerListener.log CATALINA_BASE: E:\work_test\aa.ms
16-Apr-2020 14:54:24.152 [main] com.apusic.ams.startup.VersionLoggerListener.log CATALINA_HOME: E:\work_test\aa.ms
16-Apr-2020 14:54:24.154 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file
16-Apr-2020 14:54:24.156 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=co
16-Apr-2020 14:54:24.159 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2
16-Apr-2020 14:54:24.161 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dfile.encoding=UTF-8
16-Apr-2020 14:54:24.162 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.protocol.handler.pkgs=c
16-Apr-2020 14:54:24.165 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dignore.endorsed.dirs=
16-Apr-2020 14:54:24.167 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dapusic.base=E:\work_test\aa.ms
16-Apr-2020 14:54:24.169 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Dapusic.home=E:\work_test\aa.ms
16-Apr-2020 14:54:24.171 [main] com.apusic.ams.startup.VersionLoggerListener.log Command line argument: -Djava.io.tmpdir=E:\work_test\
16-Apr-2020 14:54:24.173 [main] com.apusic.ams.core.AprLifecycleListener.lifecycleEvent The APR based Apusic AAS Native library which
indows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;F:\tools\apache-maven-3.6.0\bin;F:\软件安装\UltraEdit;F:\tools\Git\cmd;F:
16-Apr-2020 14:54:24.414 [main] com.apusic.connector.AbstractProtocol.init 初始化协议处理器 ["http-nio-8888"]
16-Apr-2020 14:54:24.491 [main] com.apusic.ams.startup.Apusic.load 服务器在16521毫秒内初始化
16-Apr-2020 14:54:24.515 [main] com.apusic.ams.core.StandardService.startInternal Starting service [ams-server]
16-Apr-2020 14:54:24.519 [main] com.apusic.ams.core.StandardEngine.startInternal 正在启动 Servlet 引擎: [Apusic AAS/10.1.0]
16-Apr-2020 14:54:24.528 [main] com.apusic.ams.ha.tcp.SimpleTcpCluster.startInternal Cluster is about to start
16-Apr-2020 14:54:24.548 [main] com.apusic.ams.tribes.transport.ReceiverBase.bind 服务器套接字接收器绑定到: [/172.20.50.43:4001]
16-Apr-2020 14:54:24.559 [main] com.apusic.ams.tribes.membership.McastServiceImpl.setupSocket 设置集群多播超时时间: [500]
16-Apr-2020 14:54:24.563 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to e
16-Apr-2020 14:54:24.688 [Membership-MemberAdded.] com.apusic.ams.ha.tcp.SimpleTcpCluster.memberAdded Replication member added: [com.ap
16-Apr-2020 14:54:25.565 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:25.577 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Sleeping for [1000] milliseconds to e
16-Apr-2020 14:54:25.593 [Tribes-Task-Receiver\ams-service-Channel-1] com.apusic.ams.tribes.io.BufferPool.getBufferPool Created a buf
16-Apr-2020 14:54:26.592 [main] com.apusic.ams.tribes.membership.McastServiceImpl.waitForMembers Done sleeping, membership established
16-Apr-2020 14:54:27.407 [main] com.apusic.jasper.servlet.TldScanner.scanJars 至少有一个JAR被扫描用于TLD但尚未包含TLD。为此记录器启用
16-Apr-2020 14:54:27.421 [main] com.apusic.ams.ha.session.DeltaManager.startInternal 将管理器 [localhost#/test] 注册到名为 [ams-service]
16-Apr-2020 14:54:27.423 [main] com.apusic.ams.ha.session.DeltaManager.startInternal Starting clustering manager at [localhost#/test]
16-Apr-2020 14:54:27.432 [main] com.apusic.ams.ha.session.DeltaManager.getAllClusterSessions Manager [localhost#/test], requesting ses
16-Apr-2020 14:54:27.539 [main] com.apusic.ams.ha.session.DeltaManager.waitForSendAllSessions 管理器 [localhost#/test]; 在 [204-16 下午
16-Apr-2020 14:54:27.589 [main] com.apusic.ams.tribes.tipis.AbstractReplicatedMap.init Initializing AbstractReplicatedMap with context
16-Apr-2020 14:54:27.611 [main] com.apusic.ams.tribes.tipis.AbstractReplicatedMap.init AbstractReplicatedMap [test] initialization was
16-Apr-2020 14:54:27.620 [main] com.apusic.connector.AbstractProtocol.start 开始协议处理句柄 ["http-nio-8888"]
16-Apr-2020 14:54:27.627 [main] com.apusic.ams.startup.Apusic.start Server startup in [3,133] milliseconds
16-Apr-2020 14:55:58.862 [http-exec-1] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 14:56:01.619 [http-exec-2] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.255 [http-exec-4] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.590 [http-exec-5] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:41.937 [http-exec-6] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:42.274 [http-exec-7] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56
16-Apr-2020 15:05:42.643 [http-exec-8] org.apache.jsp.session_jsp._jspService session get: 2020-04-16 14:55:56

```

18.2 Session 集中式存储

支持使用 Redis 进行 Session 集中式存储。

18.2.1 构建 Redis 插件

18.2.1.1 系统构建

应用服务器安装包里 `./plugins/redis/` 下有相关 jar 文件，修改

`${APUSIC_HOME}/conf/apusic.properties` 文件 `common.loader` 项，如下：

```
common.loader="${apusic.base}/lib","${apusic.base}/lib/*.jar","${apusic
```

18.2.2 添加 Redis 配置

修改 `${APUSIC_HOME}/conf/context.xml` 配置文件增加 `Valve` , `Manager` 配置, 要将 `Valve` 配置在 `Manager` 之前如:

```
<Valve
  className="com.apusic.ams.session.redisessions.RedisSessionHandlerValv
/>
<Manager
  className="com.apusic.ams.session.redisessions.RedisSessionManager"
  host="localhost" port="6379" database="0" />
```

如果 Redis 使用了主从、哨兵、集群模式, `${APUSIC_HOME}/conf/context.xml` 中的 `Manager` 配置对应修改。如 Redis 使用了哨兵模式, `Manager` 配置如下:

```
<Manager
  className="com.apusic.ams.session.redisessions.RedisSessionManager"
  password="root" sentinelMaster="mymaster"
    sentinels="172.20.140.21:26379,172.20.140.22:26379" database="0"
/>
```

Manager 参数说明:

表46: 金蝶Apusic应用服务器V10.0.10标准版session集中式配置Manager标签说明

参数	说明
host	Redis 服务器地址
port	端口
sentinelMaster	Redis server 名字
sentinels	哨兵, 用逗号分割开的“host:port”
password	Redis 密码

blockWhenExhausted	连接耗尽时是否阻塞, false 报异常,ture 阻塞直到超时, 默认 true
connectionPoolMaxIdle	资源池允许最大空闲的连接数, 默认 8
connectionPoolMaxTotal	资源池中最大连接数, 默认 8
connectionPoolMinIdle	资源池确保最少空闲的连接数, 默认 0
database	DB 索引
jmxEnabled	是否启用 pool 的 jmx 管理功能, 默认 true
lifo	是否启用后进先出, 默认 true
maxWaitMillis	获取连接时的最大等待毫秒数 (如果设置为阻塞时 BlockWhenExhausted), 如果超时就抛异常, 小于零: 阻塞不确定的时间, 默认 -1
minEvictableIdleTimeMillis	逐出连接的最小空闲时间 默认 1800000 毫秒 (30 分钟)
numTestsPerEvictionRun	每次逐出检查时 逐出的最大数目 如果为负数就是 : 1/abs(n), 默认 3
sessionPersistPolicies	"SAVE_ON_CHANGE": 每次调用 session.setAttribute() 或者 session.removeAttribute() 时, Session 都会存储到 Redis。 "ALWAYS_SAVE_AFTER_REQUEST": 请求结束后, 报错 Session, 不管是否检测到 Session 属性变化。 "DEFAULT": 默认。当检测到 Session 属性变化时, 才保存
testOnBorrow	向资源池借用连接时是否做连接有效性检测 (ping), 无效连接会被移除, 默认 false
testOnCreate	向资源池创建连接时是否做连接有效性检测 (ping)
testOnReturn	向资源池归还连接时是否做连接有效性检测 (ping), 无效连接会被移除, 默认 false
testWhileIdle	在空闲时检查有效性, 默认 false
timeBetweenEvictionRunsMillis	逐出扫描的时间间隔 (毫秒) 如果为负数, 则不运行逐出线程, 默认 -1
timeout	读取超时

19 与 Springboot 集成使用

19.1 前言

本部分为金蝶 Apusic 应用服务器V10.0.10嵌入式部署使用说明，详细介绍金蝶 Apusic 应用服务器V10.0.10嵌入式部署安装和相关配置方法。

19.1.1 面向对象

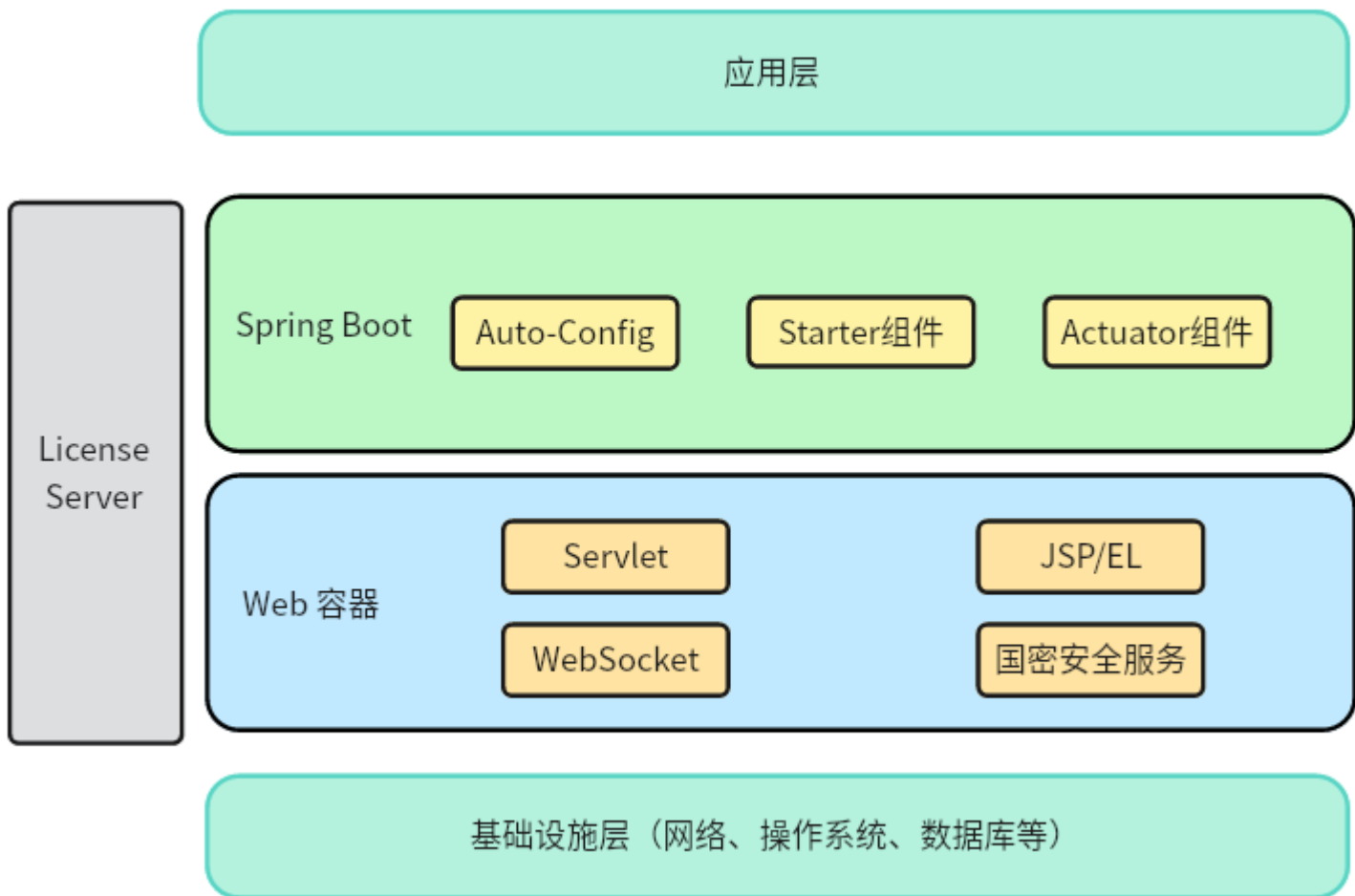
本部分主要面向对象为使用金蝶 Apusic 应用服务器V10.0.10嵌入式部署进行应用开发的开发人员，以及相关管理人员和运维人员。

19.2 产品简介

19.2.1 概述

金蝶 Apusic 应用服务器V10.0.10嵌入式部署是一款同时支持JavaEE企业级标准与 React 响应编程标准的嵌入式应用服务器，集成Spring Boot框架使用。

图2：金蝶Apusic应用服务器V10.0.10嵌入式部署架构图



19.2.2 JavaEE规范

金蝶Apusic应用服务器V10.0.10嵌入式部署产品规范。

表47：金蝶Apusic应用服务器V10.0.10嵌入式部署产品规范

规范/API	版本
Servlet	6.0
JSP	3.1
Expression Language	5.0
WebSocket	2.1

19.2.3 平台环境

金蝶Apusic应用服务器V10.0.10嵌入式部署平台环境。

表48：金蝶Apusic应用服务器V10.0.10嵌入式部署平台环境

操作系统	说明
Windows	Windows7、Windows10、Windows Server
Linux	CentOS、Ubuntu、SUSE、Redhat
国产平台	银河麒麟系列、中标麒麟系列、普华、中科红旗、深度等

19.2.4 系统环境

金蝶Apusic应用服务器V10.0.10嵌入式部署系统环境。

表49：金蝶Apusic应用服务器V10.0.10嵌入式部署系统环境

系统组件	系统要求
Java环境	JDK17及以上
内存	至少需要 512MB 的内存，建议1024MB以上
磁盘空间	至少需要 1024MB 的磁盘空间，建议10G以上

19.2.5 术语

AAS：金蝶Apusic应用服务器

AAMS-[aas_version]-embed：金蝶Apusic应用服务器嵌入式部署产品标识

APUSIC_HOME：金蝶Apusic应用服务器安装目录

19.3 产品清单

金蝶Apusic应用服务器V10.0.10嵌入式部署产品包说明，通常情况下打包成 `AAMS-[aas_version]-embed.tar.gz`。主要包括starter、websocked、国密等资源包。详看产品清单说明。

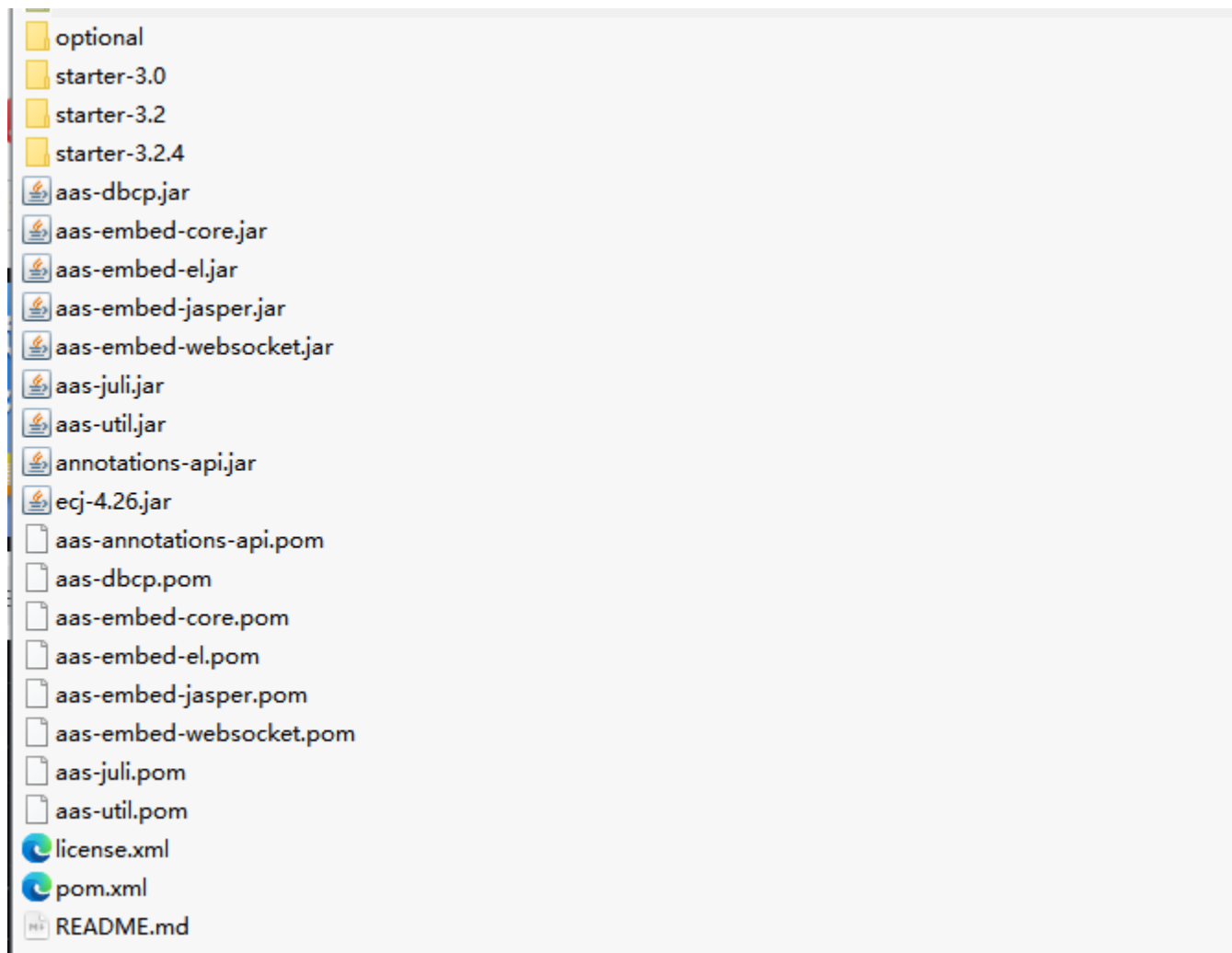
表50：金蝶Apusic应用服务器V10.0.10嵌入式部署产品包

产品包	说明
AAMS-10.0.9-embed.tar.gz	金蝶Apusic应用服务器V10.0.10嵌入式部署相关模块

19.3.1 产品清单列表

金蝶Apusic应用服务器V10.0.10嵌入式部署产品包包括starter、相关资源模块、可选资源模块等内容。

图3：金蝶Apusic应用服务器V10.0.10嵌入式部署产品清单列表



19.3.2 产品清单说明

金蝶Apusic应用服务器V10.0.10嵌入式部署产品清单说明。

表51：金蝶Apusic应用服务器V10.0.10嵌入式部署产品清单说明

文件名	说明
optional	可选组件，包括aas-sm-3.0.jar（国密模块）和acls-client-1.0.5.jar（集中授权模块）。
starter-3.0	springboot3.0版本的starter；aams-spring-boot-starter-3.0.0.jar为starter，aams-spring-boot-starter-all-3.0.0.jar为集成模块
starter-3.2	springboot3.2版本的starter；aams-spring-boot-starter-3.2.0.jar为starter，aams-spring-boot-starter-all-3.2.0.jar为集成模块

starter-3.2.4	springboot3.2.4版本的starter; aams-spring-boot-starter-3.2.4.jar为starter, aams-spring-boot-starter-all-3.2.4.jar为集成模块
annotations-api.jar、aas-annotations-api.pom	annotations-api模块的安装文件
ecj-4.26.jar	编译jsp的相关模块的安装文件
aas-dbcj.jar、aas-dbcj.pom	aas-dbcj模块的安装文件
aas-embed-core.jar、aas-embed-core.pom	aas-embed-core模块的安装文件
aas-embed-el.pom、aas-embed-el.jar	aas-embed-el模块的安装文件
aas-embed-jasper.jar、aas-embed-jasper.pom	aas-embed-jasper模块的安装文件
aas-embed-programmatic.pom、aas-embed-programmatic.jar	aas-embed-programmatic模块的安装文件
aas-embed-websocket.jar、aas-embed-websocket.pom	aas-embed-websocket模块的安装文件
aas-util.jar	一些简单的工具类的安装文件
license.xml	产品授权文件
pom.xml	产品pom文件
README.md	注册aas相关jar文件使用说明

19.4 安装

安装方式有两种：一种是基于应用系统源码重新构建可执行jar包，另一种是基于可执行jar包进行替换。

需注意，导入jar包时应根据jar包实际名称进行修改。

19.4.1 基于源码方式构建

19.4.1.1 按模块导入

应用服务器支持按模块导入jar的方式，可将aas、starter分模块导入，产品包提供了各版本的starter以及可选的模块。

19.4.1.1.1 前置条件

安装jdk环境，版本需要在JDK17及以上；

安装maven环境；

需要下载产品包 `AAMS-[aas_version]-embed.tar.gz` 到本地。

19.4.1.1.2 安装资源依赖

(1) AAS相关包导入

解压 `AAMS-10.0.9-embed.tar.gz` 后进入 `AAMS-10.0.9-embed`，通过命令行进入解压目录，随后输入命令：

`mvn initialize` 即可。

```
E:\AAMS-10.0.9-embed>mvn initialize
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.apusic:jars-init >-----
[INFO] Building jars-init 1.0
[INFO]    from pom.xml
[INFO] -----[ pom ]-----
```

(2) starter相关包导入

解压 `AAMS-10.0.9-embed.tar.gz` 后进入 `AAMS-[aas_version]-embed/starter-[starter_version]`，`[starter_version]` 为 `springboot starter` 的版本标识，可以为 `3.0/3.2/3.2.4`。通过命令行进入解压目录，随后输入以下命令即可。

示例为导入 `starter-3.0`。

导入 `jar` 文件 `aams-spring-boot-starter-3.0.0.jar`；`groupId` 为 `com.apusic`；`artifactId` 为 `aams-spring-boot-starter`；`version` 为 `3.0.0`，版本号与 `pom.xml` 中定义的一致，通常在 `jar` 文件打包时文件名称已标明；`pomFile` 为 `pom.xml`；`packaging` 为 `jar`：

```
mvn install:install-file -Dfile=aams-spring-boot-starter-3.0.0.jar -
DgroupId=com.apusic -DartifactId=aams-spring-boot-starter -
Dversion=3.0.0 -DpomFile=pom.xml -Dpackaging=jar
```

```
E:\AAMS-10.0.9-embed\starter-3.0>mvn install:install-file -Dfile=aams-spring-boot-starter-3
.0.0.jar -DgroupId=com.apusic -DartifactId=aams-spring-boot-starter -Dversion=3.0.0 -DpomFile=pom.xml -Dpackaging=jar
[INFO] Scanning for projects...
Downloading from alimaven: http://maven.aliyun.com/nexus/content/groups/public/org/apache/maven/plugins/maven-compiler-p
ugin/3.11.0/maven-compiler-plugin-3.11.0.pom
Downloaded from alimaven: http://maven.aliyun.com/nexus/content/groups/public/org/apache/maven/plugins/maven-compiler-pl
ugin/3.11.0/maven-compiler-plugin-3.11.0.pom (9.8 kB at 9.9 kB/s)
Downloading from alimaven: http://maven.aliyun.com/nexus/content/groups/public/org/apache/maven/plugins/maven-plugins/39
/maven-plugins-39.pom
Downloaded from alimaven: http://maven.aliyun.com/nexus/content/groups/public/org/apache/maven/plugins/maven-plugins/39/
maven-plugins-39.pom (8.1 kB at 17 kB/s)
Downloading from alimaven: http://maven.aliyun.com/nexus/content/groups/public/org/apache/maven/plugins/maven-compiler-
```

19.4.1.1.3 引入资源依赖

`aams-spring-boot-starter`

使用金蝶Apusic应用服务器V10.0.10嵌入式部署作为运行容器，可以在 pom.xml 中添加 aams-spring-boot-starter 和金蝶Apusic应用服务器V10.0.10嵌入式部署的依赖，同时排除tomcat 的依赖。

由于 spring-boot-starter-web 插件默认带了 tomcat 的 starter，所以在 pom.xml 中需要排除 tomcat 的 starter；以 starter-3.0 为例，如下配置：

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aams-spring-boot-starter</artifactId>
  <version>3.0.0</version>
</dependency>
```

其他模块

在 pom.xml 引入其他模块的方式如以下代码所示，也可以参考产品包的对应的pom文件。

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-embed-jasper</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-embed-websocket</artifactId>
  <version>10.0.9</version>
</dependency>
```

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-embed-core</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-dbc</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-embed-el</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-annotations-api</artifactId>
  <version>10.0.9</version>
</dependency>
```

19.4.1.1.4 验证安装

访问 maven 的本地仓库地址“`.../[repository_name]/com/apusic/`”，可以在对应仓库地址 中找到上述的 jar 包，此时即表示 jar 包部署到本地仓库成功。

名称	修改日期	类型	大小
aams-spring-boot-starter	2024/5/24 15:14	文件夹	
aas-annotations-api	2024/5/24 15:07	文件夹	
aas-dbcsp	2024/5/24 15:07	文件夹	
aas-embed-core	2024/5/24 15:07	文件夹	
aas-embed-el	2024/5/24 15:07	文件夹	
aas-embed-jasper	2024/5/24 15:07	文件夹	
aas-embed-websocket	2024/5/24 15:07	文件夹	
aas-juli	2024/5/24 15:07	文件夹	
aas-sm	2024/5/24 15:07	文件夹	
aas-util	2024/5/24 15:07	文件夹	
acls-client	2024/5/24 15:07	文件夹	

19.4.1.2 导入单个jar

为了方便与应用快速集成，应用服务器支持导入单个jar，该jar包含了aas和starter相关的jar。通常命名为 `aams-spring-boot-starter-all-[aas_version].jar`。

19.4.1.2.1 前置条件

安装jdk环境，版本需要在JDK17及以上；

安装maven环境；

需要下载产品包 `AAMS-[aas_version]-embed.tar.gz` 到本地。

19.4.1.2.2 安装资源依赖

解压 `AAMS-10.0.9-embed.tar.gz` 后进入 `AAMS-[aas_version]-embed/starter-[starter_version]`。

以 `starter-3.0` 为例，输入以下命令即可。

导入 jar 文件 `aams-spring-boot-starter-all-3.0.0.jar`；`groupId` 为 `com.apusic`；`artifactId` 为 `aams-spring-boot-starter-all`；`version` 为 `3.0.0`，版本号通常在jar文件打包时文件名称已标明；`packaging` 为 `jar`：

```
mvn install:install-file -Dfile=aams-spring-boot-starter-all-3.0.0.jar
-DgroupId=com.apusic -DartifactId=aams-spring-boot-starter-all -
Dversion=3.0.0 -Dpackaging=jar
```


在 pom.xml 引入其他模块的方式如以下代码所示。

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-embed-jasper</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-embed-websocket</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-embed-core</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-dbc</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-embed-el</artifactId>
  <version>10.0.9</version>
</dependency>
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-annotations-api</artifactId>
  <version>10.0.9</version>
</dependency>
```

19.4.1.2.4 验证安装

访问 maven 的本地仓库地址“`.../repository/com/apusic/`”，可以在对应仓库地址 中找到上述的 jar 包，此时即表示 jar 包部署到本地仓库成功。

19.4.2 基于可执行jar替换

若应用已经编译为可运行资源 jar 包，且采用的是 Spring Boot 内置的 Tomcat 服务器或其他应用服务器，需要把 Tomcat 嵌入式或其他应用服务器嵌入式的 jar 包删除，然后根据不同的 Spring Boot 版本把 AAS嵌入式所需的 jar 包添加进去。

19.4.2.1 前置条件

应用项目已开发完成，且编译为可运行资源jar包。

19.4.2.2 删除弃用依赖

以Tomcat嵌入式应用为例。

使用rar 或 7zip 等压缩工具打开已编译完成的 jar 包。

进入" `BOOT-INF/lib` " 目录，删除如下jar包。

```
tomcat-embed-core-10.1.1.jar
tomcat-embed-el-10.1.1.jar
tomcat-embed-websocket-10.1.1.jar
```

19.4.2.3 引入资源依赖

以引入starter-3.0为例。解压项目文件， `jar -xvf xxx.jar` ，随后将产品包中 `aas-embed-core.jar,aas-embed-el.jar,aas-embed-websocket.jar, aams-spring-boot-starter-3.0.0.jar` 放入 `BOOT-INF/lib` 文件夹。重新打包即可。

19.5 授权认证

金蝶 Apusic 应用服务器嵌入式部署启动需要依赖授权文件进行校验，可联系金蝶天燕销售人员申领授权文件 (license.xml)。授权认证方式有项目目录启动 (未打包状态)、参数设置启动 (未打包状态) 和参数设置启动 (可执行jar包状态) 等。

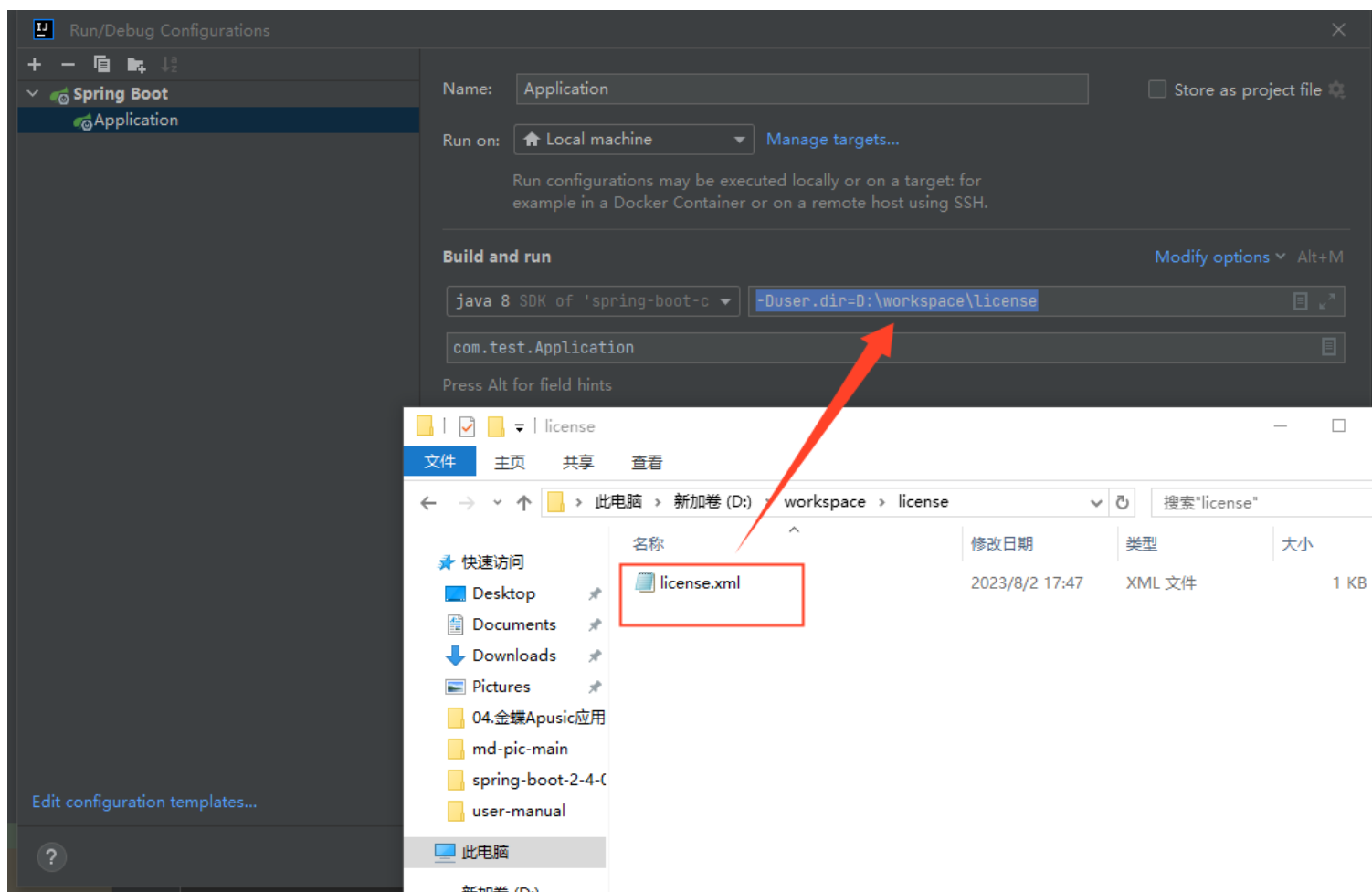
19.5.1 项目目录启动 (未打包状态)

将 `license.xml` 文件放在项目启动文件目录下，如下图所示。

名称	修改日期	类型	大小
.idea	2023/8/21 14:54	文件夹	
LOG_FILE_PATH_IS_UNDEFINED	2023/8/21 14:54	文件夹	
src	2023/8/16 10:13	文件夹	
target	2023/8/16 13:35	文件夹	
license.xml	2023/8/2 17:47	XML 文件	1 KB
pom.xml	2023/8/17 16:41	XML 文件	6 KB

19.5.2 参数设置启动（未打包状态）

打开idea，找到Run->Edit Configuration->Spring Boot中的项目启动类，在右边的VM options中输入 `-Duser.dir=xxxxxxx`，`xxxxxxx` 表示 `license.xml` 文件所在的目录，如下图所示。



19.5.3 项目目录启动（可执行jar包状态）

将 `license.xml` 文件放在项目启动文件目录下，如下图所示。

/opt/AAS/		
Name	Size (KB)	Last modified
..		
springboot-helloworld-0.0.1-S...	23 536	2024-09-23 14:35
license.xml	1	2024-09-23 14:36

19.5.4 参数设置启动（可执行jar包状态）

当项目已经打包为可执行jar包状态时，可以通过一下命令对项目进行授权认证。

```
java -Dlicpath=[license.xml文件的绝对路径] -jar [可执行jar包]
```

19.5.5 集中授权

应用服务器支持集中授权，通过连接授权中心，由授权中心统一下发管理授权文件，实现集中授权。

需要将集中授权模块导入到本地库。

导入 jar 文件 acls-client-1.0.5.jar；groupId 为 com.apusic；artifactId 为 acls-client；version 为 1.0.5，版本号通常在jar文件打包时文件名称已标明；packaging 为 jar：

```
mvn install:install-file -Dfile=optional/acls-client-1.0.5.jar -
DgroupId=com.apusic -DartifactId=acls-client -Dversion=1.0.5 -
Dpackaging=jar
```

```

D:\AAMS-10.0.9-embed>mvn install:install-file -Dfile=optional/acls-client-1.0.5.jar -Dgroup
Id=com.apusic -DartifactId=acls-client -Dversion=1.0.5 -Dpackaging=jar
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.apusic:jars-init >-----
[INFO] Building jars-init 1.0
[INFO]   from pom.xml
[INFO] -----[ pom ]-----
[INFO] --- install:2.5.2:install-file (default-cli) @ jars-init ---
[INFO] pom.xml not found in acls-client-1.0.5.jar
[INFO] Installing D:\AAMS-10.0.9-embed\optional\acls-client-1.0.5.jar to D:\software\maven\
apache-maven-3.9.2\mvn_resp\com\apusic\acls-client\1.0.5\acls-client-1.0.5.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.654 s
[INFO] Finished at: 2024-05-27T15:21:05+08:00
[INFO]

```

在pom文件引入。

```
<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>acls-client</artifactId>
  <version>1.0.5</version>
</dependency>
```

连接授权中心。以通过JVM参数启动为例，更多连接方法请参考《金蝶Apusic许可授权中心用户手册》。

- `apusic_acls_enable`：是否开启授权中心认证，取值为 `true` 或 `false`，为 `true` 则表示开启授权中心认证。没有该参数或该参数值为 `false`，都表示没有开启授权中心认证
- `apusic_acls_authUrls`：授权中心的地址，可设置多个授权地址，格式为 `ip1:port1,ip2:port2`，如果一个授权地址链接失败，会轮询其他的地址；如果开启授权中心认证，则为必填参数，其中端口为授权中心的https端口
- `apusic_acls_ns`：设置该实例所属的命名空间名称，可选参数；默认值为 `public`，具体的命名空间可以在授权中心管理控制台-系统管理-授权管理查看
- `apusic_acls_tenant`：设置该实例所属的租户名称，可选参数

```
java -jar -Dapusic_acls_ns=深圳市金蝶天燕云计算股份有限公司_TEMP -
Dapusic_acls_enable=true -Dapusic_acls_authUrls=172.24.3.116:6869
spring-boot-jsp.jar
```

19.5.6 特征码获取

一般情况下产品会在出厂时自带了有效的授权文件，如果产品将要过期或者已过期，可以与金蝶天燕对接人员联系申请授权文件。与此同时需要把产品的特征码发给金蝶天燕对接人员，产品特征码获取方式参考以下步骤。

先把授权文件移除，然后在启动命令中增加JVM参数 `apusic-ac`，取值为ip地址或者网卡名称，类似如下，`eth3`为测试机器的网卡名称，需要更加实际的网络信息修改：

```
java -jar -Dapusic-ac=eth3 boot-quickstart-1.0.jar
```

由于没有授权文件，应用启动会停止，在停止前会打印特征码信息，类似如下，Auth Code=[特征码内容]：

```
Auth Code=SZTY263537626

license is invalid.
```

19.6 容器功能说明

19.6.1 安全管理

介绍安全管理相关配置。

19.6.1.1 HTTPS

19.6.1.1.1 简介

HTTPS（全称：Hypertext Transfer Protocol Secure），是由 HTTP 加上 TLS/SSL 协议构建的可进行加密传输、身份认证的网络协议，可以有效提高网站安全性。金蝶 Apusic 应用服务器嵌入式部署支持HTTPS。

19.6.1.1.2 配置方法

使用HTTPS时，可以通过Spring Boot 配置加载对应 `application.properties`、`application.yml` 等文件加载配置项。

`application.properties` 配置如下：

```
#https配置
#应用服务器监听端口
server.port=8089
#是否开启SSL协议，为true表示启用
server.ssl.enabled=true
#配置SSL证书目录
server.ssl.key-store=classpath:keystore.jks
#SSL证书类型，根据证书实际配置
server.ssl.key-store-type=JKS
#私钥密码，根据证书实际配置
server.ssl.key-password=xxxxxxx
#密钥库文件密码，根据证书实际配置
server.ssl.key-store-password=xxxxxxx
#服务器使用的密码和证书在密钥库的别名，根据证书实际配置
server.ssl.key-alias=apusicuserkey
#客户端证书验证，值为 need,want,none 之一
server.ssl.client-auth=none
#信任库文件，用于验证客户端证书
server.ssl.trust-store=classpath:truststore.jks
#信任库密码，根据证书实际配置
server.ssl.trust-store-password=xxxxxxx
#信任库类型
server.ssl.trust-store-type=JKS
```

#允许使用的密码套件

```
server.ssl.ciphers=*
```

19.6.1.2 国密配置

19.6.1.2.1 简介

金蝶 Apusic 应用服务器V10.0.10嵌入式部署支持国密SSL协议通信，支持国家密码局认定的国产密码算法。

19.6.1.2.2 配置方法

注册国密实现jar包

将金蝶Apusic应用服务器V10.0.10嵌入式部署的国密实现 Jar 包 `${APUSIC_HOME}/optional/aas-sm-3.0.jar` 注册到 Maven 库中，以备应用打包引用。

导入 jar 文件 `aas-sm-3.0.jar`；groupId 为 `com.apusic`；artifactId 为 `aas-sm`；version 为 `3.0`，版本号通常在jar文件打包时文件名称已标明；packaging 为 `jar`：

```
mvn install:install-file -Dfile=aas-sm-3.0.jar -DgroupId=com.apusic -DartifactId=aas-sm -Dversion=3.0 -Dpackaging=jar
```

```

D:\AAMS-10.0.9-embed>mvn install:install-file -Dfile=optional/aas-sm-3.0.jar -DgroupId=com.apusic -DartifactId=aas-sm -Dversion=3.0 -Dpackaging=jar
[INFO] Scanning for projects...
[INFO] -----< com.apusic:jars-init >-----
[INFO] Building jars-init 1.0
[INFO]   from pom.xml
[INFO] -----[ pom ]-----
[INFO] --- install:2.5.2:install-file (default-cli) @ jars-init ---
[INFO] pom.xml not found in aas-sm-3.0.jar
[INFO] Installing D:\AAMS-10.0.9-embed\optional\aas-sm-3.0.jar to D:\software\maven\apache-maven-3.9.2\mvn_resp\com\apusic\aas-sm\3.0\aas-sm-3.0.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.011 s
[INFO] Finished at: 2024-05-27T14:05:52+08:00
[INFO] -----

```

引入依赖

在 `pom.xml` 文件中添加国密实现依赖。

```

<dependency>
  <groupId>com.apusic</groupId>
  <artifactId>aas-sm</artifactId>

```

```
<version>3.0</version>
</dependency>
```

配置文件

引入依赖之后，需要在项目的 `application.properties` 或 `application.yml` 添加国密SSL配置。

`application.properties` 配置如下：

```
#国密配置：
#表示SSL通信协议使用国密通信协议
server.ssl.protocol=SMv1.1
#国密证书私钥类型
server.ssl.key-store-type=PKCS12
#配置国密密钥库文件
server.ssl.key-store=classpath:keystore.p12
#私钥密码
server.ssl.key-password=xxxxxxx
#密钥库文件密码
server.ssl.key-store-password=xxxxxx
```

启动应用

Springboot 应用中使用国密需要在应用启动中添加 `-Dcom.apusic.security.ssl.EnableGMTLS=true` 和 `-Djava.security.egd=file:/dev/./urandom` 参数，参考以下示例 `springboot-gm-demo.jar` 启动方式：

```
java -Dcom.apusic.security.ssl.EnableGMTLS=true -
Djava.security.egd=file:/dev/./urandom -jar springboot-gm-demo.jar
```

访问方法

项目配置国密SSL协议之后，需要使用支持国密SSL协议的浏览器访问，例如360安全浏览器、奇安信等。浏览器需要开启国密通信功能。

以360安全浏览器为例，进入“设置”-“选项”-“安全设置”，勾选“国密通信协议”，设置“国密SSL通讯白名单”，重启浏览器。

选项

基本设置

界面设置

标签设置

鼠标手势

快捷键

高级设置

实验室

安全设置

核心安全防护

- 多层系统防护 文件，驱动，进程，注册表多层防护，保护系统安全
- 沙箱防护 页面沙箱防护，防止未知病毒侵害
- 挖矿防护 禁止执行数字货币挖矿脚本，防止硬件资源被利用
- 隔离防护 网页，代码，插件，CPU进程隔离
- 系统级防注入 防恶意代码注入，保护浏览器安全

隐私安全设置

清理上网痕迹设置...

管理HTTPS/SSL证书...

- 检查服务器证书吊销状态
- 拦截证书风险
- 开启安全键盘功能，输入密码时使用虚拟键盘，可防止密码被键盘记录器窃取
- 自动停用来源不明的扩展
- 开启“禁止跟踪(DNT)”功能

国密通信协议

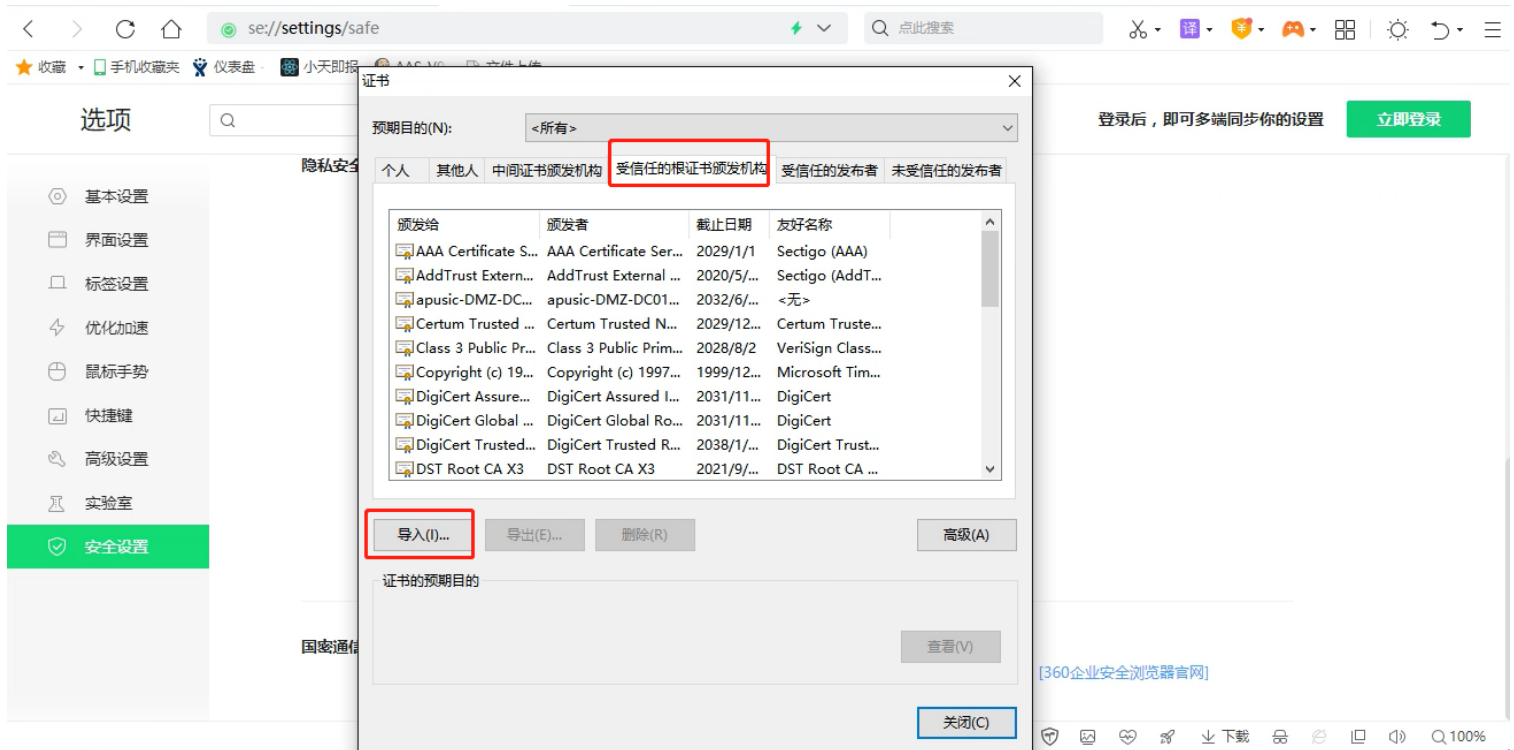
- 启用国密SSL协议支持

国密SSL通讯白名单 HTTPS协议优先走国密SSL协议

根证书管理

信任的根证书

将根证书(如CA.cert.pem)导入到浏览器中，重启浏览器。再访问应用程序。



19.6.2 国际化支持

19.6.2.1 简介

为方便不同区域用户使用金蝶Apusic应用服务器V10.0.10嵌入式部署支持，金蝶Apusic应用服务器V10.0.10嵌入式部署提供国际化支持。

19.6.2.2 配置方法

默认日志输出采用的是英文格式输出。如需改成中文格式需要在 spring boot 的配置文件中编写代码定义日志输出格式配置，如下所示。

```
@Bean
public Locale locale() {
    return Locale.CHINA;
}
```

19.6.3 日志管理

19.6.3.1 应用日志

19.6.3.1.1 简介

金蝶Apusic应用服务器V10.0.10嵌入式部署支持日志功能，支持集成Spring Boot方法实现。

19.6.3.1.2 使用方法

可以通过Spring Boot 配置加载对应 `application.properties` 或 `application.yml` 配置文件，配置Spring Boot 标准日志配置信息。

`application.properties` 配置如下：

```
#日志配置
#关闭日志输出
#logging.level.root: OFF
#日志级别
logging.level.root=info
#使用相对路径的方式设置日志输出的位置
#logging.file.path=logs
#控制台日志输出格式
logging.pattern.console=%d{yyyy-MM-dd hh:mm:ss} [%thread] %-5level
%logger{50} - %msg%n
#日志文件输出格式
logging.pattern.file=%d{yyyy-MM-dd} === [%thread] === %-5level ===
%logger{50} === - %msg%n
#定义日志文件名称
logging.file.name=logs/aams-log-test.log
```

19.6.3.2 访问日志

19.6.3.2.1 简介

金蝶Apusic应用服务器V10.0.10嵌入式部署支持访问日志记录输出能力，集成Spring Boot方法配置对接。

19.6.3.2.2 使用方法

可以通过Spring Boot 配置加载对应 `application.properties` 或 `application.yml` 配置文件，配置日志信息。

`application.properties` 配置如下：

```
server.aas.basedir=./
server.aas.accesslog.enabled=true
```

19.6.4 监控支持

19.6.4.1 简介

金蝶Apusic应用服务器V10.0.10嵌入式部署提供监控功能，自带了springboot actuator 框架，监控对应资源。

19.6.4.2 使用方法

在项目 `pom.xml` 中引入以下依赖。

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

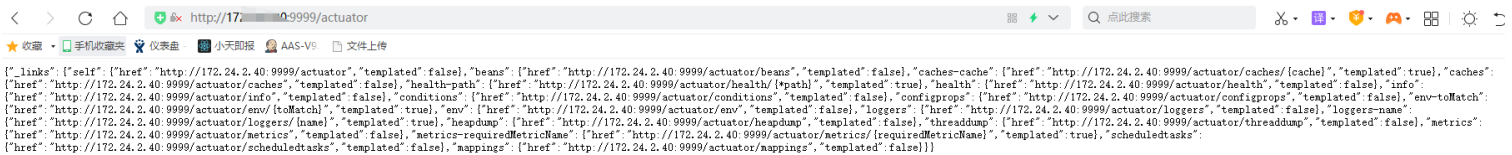
开启监控

在 spring boot 的配置文件中配置开启监控，在 `application.properties` 配置如下。

```
#actuator端口 如果不配置做默认使用上面server.port端口
management.server.port=9999
#默认值访问health,info端点,用*可以包含全部端点
management.endpoints.web.exposure.include=*
#修改访问路径
management.endpoints.web.base-path=/actuator
```

查看监控

启动服务，格式为 `http://[IP]:[management.server.port]/[management.endpoints.web.base-path]`。如访问 `http://[IP]:9999/actuator`。



访问目录描述

路径	说明
<code>/actuator</code>	actuator服务访问上下文
<code>/actuator/configprops</code>	描述配置属性如何注入Bean

/actuator/beans	描述配置应用程序上下文里全部Bean以及其关系
/actuator/conditions	提供自动配置，记录自动配置通过情况
/actuator/env	获取全部环境属性
/actuator/health	报告应用程序的监控指标
/actuator/loggers	获取loggers
/actuator/heapdump	获取heapdump
/actuator/metrics	报告各种应用程序度量信息，比如内存用量和HTTP请求计数
/actuator/threaddump	获取threaddump
/actuator/httptrace	提供基本的http请求跟踪信息
/actuator/mappings	报告全部URL路径
/actuator/scheduledtasks	获取计划任务

19.7 示例操作

示例 `spring-boot-jsp` 使用AAS作为springboot应用的运行容器。

19.7.1 环境准备

应用工程：使用的应用工程

maven环境：安装有可用的maven环境，用于构建应用

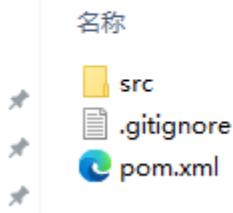
Java环境：准备有可用的JDK，版本需要在JDK17及以上

AAS授权文件：准备对应的AAS授权文件，或者对接的Apusic授权中心

19.7.2 应用工程

示例应用springboot版本为3.3.0，如下：

spring-boot-demo-jsp



19.7.3 pom文件

pom文件如下:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>test</groupId>
  <artifactId>springboot-demo-jsp</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>

  <properties>
    <failOnMissingWebXml>>false</failOnMissingWebXml>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
      <version>3.3.0</version>
      <exclusions>
        <exclusion>
          <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-  
tomcat</artifactId>  
    </exclusion>  
</exclusions>  
</dependency>  
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-test</artifactId>  
    <version>3.3.0</version>  
    <scope>test</scope>  
    <exclusions>  
        <exclusion>  
            <groupId>com.jayway.jsonpath</groupId>  
            <artifactId>json-path</artifactId>  
        </exclusion>  
    </exclusions>  
</dependency>  
  
<dependency>  
    <groupId>com.apusic</groupId>  
    <artifactId>aams-spring-boot-starter</artifactId>  
    <version>3.3.0</version>  
</dependency>  
  
<dependency>  
    <groupId>com.apusic</groupId>  
    <artifactId>aas-embed-jasper</artifactId>  
    <version>10.0.9</version>  
</dependency>  
  
<dependency>  
    <groupId>jakarta.servlet.jsp.jstl</groupId>  
    <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>  
    <version>3.0.0</version>  
</dependency>  
  
<dependency>  
    <groupId>jakarta.el</groupId>
```

```
<artifactId>jakarta.el-api</artifactId>
<version>5.0.0</version>
</dependency>
<dependency>
  <groupId>org.glassfish.web</groupId>
  <artifactId>jakarta.servlet.jsp.jstl</artifactId>
  <version>3.0.0</version>
</dependency>
</dependencies>
<build>
  <finalName>spring-boot-jsp</finalName>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <version>3.3.0</version>
      <executions>
        <execution>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
  <resources>
    <resource>
      <directory>src/main/java</directory>
      <includes>
        <include>*/*.properties</include>
      </includes>
    </resource>
    <resource>
      <directory>src/main/resources</directory>
    </resource>
    <resource>
```

```
        <directory>src/main/webapp</directory>
        <targetPath>META-INF/resources</targetPath>
        <includes>
            <include>*/**</include>
        </includes>
    </resource>
</resources>
</build>
</project>
```

19.7.4 构建应用包

进入工程目录下，使用maven构建，执行 `mvn package` 。

19.7.5 运行应用

运行前需要将授权文件 `license.xml` 放置与应用 `spring-boot-jsp.jar` 同一目录，执行：

```
java -jar spring-boot-jsp.jar
```

19.7.6 访问应用

客户端访问<http://IP:PORT/MyServlet>。如<http://localhost/8080/MyServlet>。

19.7.7 以调试模式运行

如果需要启动调试模式 (debug) ，执行：

```
java -
agentlib:jdwp=transport=dt_socket,address=8000,server=y,suspend=n -jar
spring-boot-jsp.jar
```

20 附录

20.1 附录A Web通用配置

配置参数	说明	值类型
server.address	应用服务器监听地址	
server.port	应用服务器监听端口	8080
server.error.include-binding-errors	配置在错误响应里是否包括数据绑定错误信息，可设项为： always:无论在什么情况下，错误响应中都会包含数据绑定错误信息。 never:错误响应中永远不会包含数据绑定错误信息。 on-param:只有当请求中包含 trace 参数时，错误响应才会包含数据绑定错误信息	never
server.error.include-exception	控制在错误响应中是否包含异常类的完整名称。可设项为"true"或"false"	FALSE
server.error.include-message	用于控制在错误响应中是否包含异常消息的配置属性。可设项为： always:无论在何种情况下，错误响应中都会包含异常消息。 never:错误响应中永远不会包含异常消息。 on-param: 只有当请求中包含 trace 参数时，错误响应才会包含异常消息	never
server.error.include-stacktrace	用于控制在错误响应里是否包含异常堆栈跟踪信息。可设项为： always:无论在什么情况下，错误响应中都会包含异常堆栈跟踪信息。 never:错误响应中永远不会包含异常堆栈跟踪信息。 on-param: 只有当请求中包含 trace 参数时，错误响应才会包含异常堆栈跟踪信息	never
server.error.path	配置错误处理路径	/error
server.error.whitelabel.enabled	当服务器发生错误时，是否启用浏览器显示的默认错误页面	TRUE

server.forward-headers-strategy	处理X-Forwarded-* headers的策略	
server.server-header	设置 HTTP 响应头 Server 的值	
server.max-http-header-size	设置最大 HTTP 消息头大小	8192
server.shutdown	服务器支持的关闭类型	immediate
server.http2.enable	是否启用http2	false
server.ssl.enabled	是否允许 SSL 通信	true
server.ssl.ciphers	允许使用的密码套件	
server.ssl.client-auth	客户端证书验证, 值为 need,want,none 之一	
server.ssl.enabled-protocols	允许与客户端通信的协议名字	
server.ssl.key-alias	服务器使用的密码和证书在密钥库的别名	
server.ssl.key-password	密钥库密钥密码	
server.ssl.key-store	密钥库文件	
server.ssl.key-store-password	密钥库文件密码	
server.ssl.key-store-provider	密钥储存文件的提供者	
server.ssl.key-store-type	密钥库文件类型	
server.ssl.protocol	服务器使用的 SSL 协议, 用于创建 SSLContext	TLS
server.ssl.trust-store	信任库文件, 用于验证客户端证书	
server.ssl.trust-store-password	信任库密码	
server.ssl.trust-store-provider	信任储存的提供者	
server.ssl.trust-store-type	信任库类型	
server.ssl.bundle	已配置SSL bundle的名称	
server.ssl.certificate	pem编码的SSL证书文件的路径	
server.ssl.certificate-private-key	SSL证书的pem编码私钥文件的路径	
server.ssl.trust-certificate	pem编码的SSL认证机构文件的路径	

server.ssl.trust-certificate-private-key	SSL证书颁发机构的pem编码私钥文件的路径	
server.servlet.context-parameters.*	Servlet上下文初始化参数	
server.servlet.contextPath	应用上下文	
server.servlet.application-display-name	应用显示名称	application
server.servlet.encoding.charset	HTTP请求和响应的字符集。如果没有明确设置，则添加到“Content-Type”头	utf-8
server.servlet.encoding.force	是否在HTTP请求和响应上强制编码为配置的字符集	
server.servlet.encoding.force-request	是否强制HTTP请求的编码为配置的字符集	
server.servlet.encoding.force-response	是否强制HTTP响应的编码为配置的字符集	
server.servlet.encoding.mapping	将区域设置映射到响应编码的字符集	
server.servlet.jsp.class-name	用于 JSP 的 Servlet 的类名	
server.servlet.jsp.init-parameters	用于配置 JSP Servlet 的 Init 参数	
server.servlet.jsp.registered	JSP Servlet 是否已注册	
server.servlet.session.persistent	是否报错 Session 在文件中	false
server.servlet.session.store-dir	Session 存储路径	
server.servlet.session.timeout	Session 失效时间	3600ms
server.servlet.session.tracking-modes	配置会话跟踪模式。可设项为:COOKIE、URL、SSL	3600ms
server.servlet.session.persistent	是否在重启之间保留会话数据	FALSE
server.servlet.session.store-dir	用于存储会话数据的目录	
server.servlet.session.cookie.domain	设置该 Cookie 的域名	
server.servlet.session.cookie.name	设置 Cookie 的名称	JSESSIONID
server.servlet.session.cookie.path	Cookie 的使用路径	

server.servlet.session.cookie.comment	cookie的注释	
server.servlet.session.cookie.http-only	Cookie 的访问仅限于 HTTP (或 HTTPS) 请求	
server.servlet.session.cookie.secure	是否始终将cookie标记为安全	
server.servlet.session.cookie.max-age	cookie的最大使用时间	
server.servlet.register-default-servlet	是否向容器注册默认Servlet	
server.servlet.session.cookie.same-site	cookie的SameSite设置	
server.reactive.session.timeout	会话超时	30m
server.reactive.session.cookie	与server.servlet.session.cookie部分一致	
server.aas.accept-count	已连接但应用服务器未处理数	100
server.aas.additioonal-tld-skip-patterns	在进行TLD扫描时, 需要忽略与jar匹配的其他模式, 这些模式以逗号分隔	
server.aas.basedir	应用服务器基础路径, 若不设置则使用临时文件夹	
server.aas.background-processor-delay	调用backgroundProcess方法之间的延迟	
server.aas.connection-timeout	连接超时时间	
server.compression.enabled	是否启用压缩	false
server.compression.excluded-user-agents	设置不压缩的客户端	
server.compression.mime-types	压缩的文件类型	text/html,text/xml,text/plain,text/css,text/javascript,application/javascript,application/json,application/xml
server.compression.min-response-size	超过该大小的文件才启用压缩	2048

server.aas.disable-upload-timeout	是否禁用上传超时	false
server.aas.enable-lookups	是否启用 DNS 查询以将远程 IP 地址解析为域名，在访问日志中记录客户端的主机名	true
server.aas.head-case-sensitive	head是否大小写敏感	true
server.aas.keepalive-timeout	在连接关闭之前等待另一个HTTP请求的时间	30000ms
server.aas.license-path	License 路径	
server.aas.max-connections	最大并发个数	10000
server.aas.max-http-form-post-size	HTTP post请求中表单内容的最大大小	2097152
server.aas.max-keepalive-requests	在连接关闭之前可以通过管道传输的HTTP请求的最大数量，-1 表示不限制	100
server.aas.max-response-header-size	HTTP响应头最大大小	8kb
server.aas.max-request-header-size	HTTP请求头最大大小	0
server.aas.max-swallow-size	请求体大小限制	20971520
server.aas.mbeanregistry.enabled	是否启用AAS的MBean注册中心	false
server.aas.processor-cache	可以保留在缓存中并在后续请求中重用的空闲处理器的最大数目, springboot2.4及以上版本支持	200
server.aas.redirect-context-root	是否应该通过在路径后面添加一个/来重定向到上下文根目录的请求	200
server.aas.reject-illegal-header	拒绝任何包含不符合 HTTP 标准的头的请求	true
server.aas.relaxed-path-chars	允许请求行中的特殊字符 其余配置方式： 1. 系统环境变量： AAS_RELAXED_PATH_CHARS; 2. JVM参数：-Drelaxed-path-char 优先级：系统环境变量 > JVM参数 > Spring boot配置文件，配置示例请阅读附录D	

server.aas.use-sendfile	是否使用 sendfile 功能	true
-------------------------	------------------	------

20.2 附录B 嵌入式部署日志参数

参数	说明	默认值
server.aas.accesslog.enabled	是否打印 accesslog 日志，为 true 时为打印	false
server.aas.accesslog.pattern	日志格式	common
server.aas.accesslog.directory	日志文件存放的路径	logs
server.aas.accesslog.condition-if	是否只有在“ServletRequest.getAttribute (conditionIf) ”不返回null时才启用请求日志记录。	
server.ras.accesslog.condition-unless	是否只有在“ServletRequest.getAttribute (conditionUnless) ”返回null时才启用请求日志记录。	
server.aas.accesslog.prefix	日志名称前缀	access_log
server.aas.accesslog.suffix	日志名称后缀	.log
server.aas.accesslog.encoding	日志文件使用的字符集	
server.aas.accesslog.locale	用于格式化日志条目和日志文件名后缀中的时间戳的区域设置	true
server.aas.accesslog.check-exists	是否检查日志文件是否存在，以便在外部进程重命名它时可以重新创建它	false
server.aas.accesslog.rotate	是否启用日志轮转	true
server.aas.accesslog.rename-on-rotate	是否推迟在文件名中包含日期戳，直到轮换	false
server.aas.accesslog.max-days	删除访问日志文件前保留的天数	-1
server.aas.accesslog.file-date-format	日志文件名中的日期格式	.yyyy-MM-dd
server.aas.accesslog.ipv6-canonical	是否使用RFC 5952定义的IPv6规范表示格式	false

server.aas.accesslog.request-attributes-enabled	是否记录请求的IP地址、主机名、协议和端口	false
server.aas.accesslog.buffered	是否缓冲日志记录。设置 true 时，将异步记录日志	true

20.3 附录C actuator详细说明

20.3.1 配置

参数	说明	默认值
management.endpoints.web.exposure.include	选择暴露端口名(*为暴露所有端口); 输入端口名1,端口名2,...,端口名n即可选择性暴露多个端口	Health,info
management.endpoints.web.exposure.exclude	隐藏端口; 输入端口名1,端口名2,...,端口名n即可选择性隐藏多个端口	
management.server.port	actuator开放的端口, 它用于提供应用程序的健康状况、性能指标等管理信息。	8080
management.endpoints.web.base-path	通过更改此属性的值, 可以更改管理访问 actuator的 URL 路径。例如, 如果将 management.endpoints.web.base-path 设置为 /management, 则应用程序的管理端点 URL 将变为 http://localhost:8080/management/health	/actuator

20.3.2 端口

路径	说明	详情
/actuator	actuator服务访问上下文	
/actuator/beans	beans 端点提供有关应用程序 bean 的信息	详见beans
/actuator/caches	查看应用程序的缓存	详见caches
/actuator/health	查看应用程序的健康状态	详见health
/actuator/info	提供有关应用程序的一般信息 (可自定义)	
/actuator/conditions	提供有关配置和自动配置类条件评估的信息	详见conditions

/actuator/configprops	提供有关应用程序@ConfigurationPropertiesBean的信息	详见configprops
/actuator/env	提供有关应用程序的整个环境	详见env
/actuator/loggers	提供对应用程序记录器及其级别配置的访问	详见loggers
/actuator/heapdump	提供来自应用程序 JVM 的heap dump	响应是HPROF格式的二进制数据
/actuator/threaddump	提供来自应用程序 JVM 的thread dump	详见threaddump
/actuator/metrics	提供对应用程序指标的访问	详见metrics
/actuator/scheduledtasks	提供有关应用程序计划任务的信息	详见scheduledtasks
/actuator/mappings	提供有关应用程序请求映射的信息	详见mappings

20.3.2.1 beans

自定义一个bean，然后运行程序后，进入 /actuator/beans 搜索test即可查看到test的相关信息。

```
@Configuration
public class HelloWorld {
    @Bean
    public void test() {
        System.out.println("hello, world!");
    }
}
```

```
{
  "context": {
    "application": {
      "beans": {
        ....
        "test": {
          "aliases": [],
          "scope": "singleton",
          "type": "void",
          "resource": "class path resource
```

```
[com/apusic/demo/bean/HelloWorld.class]",
    "dependencies": []
  },
  ...
}
}
}
}
```

```
Body Cookies Headers (5) Test Results Status: 200 OK Time: 34 ms Size: 73.9 KB Save Response
Pretty Raw Preview Visualize JSON
655     "dependencies": []
656   },
657   "defaultViewResolver": {
658     "aliases": [],
659     "scope": "singleton",
660     "type": "org.springframework.web.servlet.view.InternalResourceViewResolver",
661     "resource": "class path resource [org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoConfiguration$WebMvcAutoConfigurationAdapter.class]",
662     "dependencies": []
663   },
664   "test": {
665     "aliases": [],
666     "scope": "singleton",
667     "type": "void",
668     "resource": "class path resource [com/apusic/demo/bean/HelloWorld.class]",
669     "dependencies": []
670   },
671   "routerFunctionMapping": {
672     "aliases": [],
673     "scope": "singleton"
```

详细参数介绍如下:

路径	类型	说明
contexts	Object	由 id 键控的应用程序上下文
contexts.*.parentId	String	父应用程序上下文的 ID (如果有)
contexts.*.beans	Object	应用程序上下文中的 Bean 按名称键入
contexts.*.beans.*.aliases	Array	任何别名的名称
contexts.*.beans.*.scope	String	Bean 的范围
contexts.*.beans.*.type	String	完全合格的 bean 类型
contexts.*.beans.*.resource	String	定义 bean 的资源 (如果有)
contexts.*.beans.*.dependencies	Array	任何依赖项的名称

20.3.2.2 health

修改配置信息显示health端口的详细内容。

```
management.endpoint.health.show-details=always #
never/always/when_authorized,默认为never
```

此时再次访问 `/actuator/health` 端口查看详细内容，只有当components中所有组件均为UP，应用程序的健康状态才为UP。（eg. 当添加数据源但是数据库服务未开启时组件中有关服务会显示DOWN，此时应用程序的健康状态为DOWN）

```
{
  "status": "UP",
  "components": {
    "diskSpace": {
      "status": "UP",
      "details": {
        "total": 107371032576,
        "free": 97080827904,
        "threshold": 10485760,
        "exists": true
      }
    },
    "ping": {
      "status": "UP"
    }
  }
}
```

20.3.2.3 conditions

进入 `/actuator/conditions` 即可看到类似以下信息的响应。

```
{
  "contexts" : {
    "application" : {
      "positiveMatches" : {
        ...
        "EndpointAutoConfiguration#endpointOperationParameterMapper"
      }
    }
  }
}
```


contexts.*.negativeMatches.*.matched.[] .message	String	条件匹配的原因的详细信息。
contexts.*.unconditionalClasses	Array	无条件自动配置类的名称（如果有）。
contexts.*.parentId	String	父应用程序上下文的 ID（如果有）。

20.3.2.4 configprops

进入 /actuator/configprops 即可看到类似以下信息的响应。

```
{
  "contexts": {
    "application": {
      "beans": {
        "management.endpoints.web-org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties": {
          "prefix": "management.endpoints.web",
          "properties": {
            "pathMapping": {},
            "exposure": {
              "include": [
                "*"
              ],
              "exclude": []
            },
            "basePath": "/actuator"
          },
          "inputs": {
            "pathMapping": {},
            "exposure": {
              "include": [
                {
                  "value": "*",
                  "origin": "class path resource [application.properties] - 1:43"
                }
              ],
              "exclude": []
            },
            "basePath": {}
          }
        }
      }
    }
  }
}
```

结合我们的配置文件，从红框中可以看到我们在配置文件中对应用程序的相关配置。

```
management.endpoints.web.exposure.include=*
```

响应详情如下:

路径	类型	描述
contexts	Object	由 id 键控的应用程序上下文。
contexts.*.beans.*	Object	@ConfigurationProperties 通过 bean 名称键入的 beans。

contexts.*.beans.*.prefix	String	应用于 bean 属性名称的前缀。
contexts.*.beans.*.properties	Object	bean 的属性作为名称-值对。
contexts.*.beans.*.inputs	Object	绑定到此 bean 时使用的配置属性的来源和值。
contexts.*.parentId	String	父应用程序上下文的 ID (如果有)。

20.3.2.5 env

进入 /actuator/env 路径下, 即可看到如下页面:

```
{
  "activeProfiles": [],
  "propertySources": [
    {
      "name": "server.ports",
      "properties": {
        "local.server.port": {
          "value": "Complex property type java.lang.Integer"
        }
      }
    },
    {
      "name": "servletContextInitParams",
      "properties": {}
    },
    {
      "name": "systemProperties",
      "properties": {
        "java.runtime.name": {
          "value": "OpenJDK Runtime Environment"
        },
        "spring.output.ansi.enabled": {
          "value": "always"
        },
        "sun.boot.library.path": {
          "value": "D:\\workspace\\jdk8\\jre\\bin"
        },
        "java.vm.version": {
          "value": "25.382-b05"
        }
      }
    }
  ]
}
```

从图中可以看到我们一些系统配置以及服务端口值的类型。

路径	类型	描述
property	Object	来自环境的属性 (如果发现)
property.source	String	属性来源的名称

property.value	String	属性的值
activeProfiles	Array	活动配置文件的名称（如果有）
propertySources	Array	属性来源按优先顺序排列
propertySources.[].name	String	属性来源的名称
propertySources.[].property	Object	属性来源中的属性
propertySources.[].property.value	Varies	属性值
propertySources.[].property.origin	String	属性来源

20.3.2.6 loggers

进入 `actuator/loggers` 路径可以看到以下信息。

```

"levels": [
  "OFF",
  "ERROR",
  "WARN",
  "INFO",
  "DEBUG",
  "TRACE"
],
"loggers": {
  "ROOT": {
    "configuredLevel": "INFO",
    "effectiveLevel": "INFO"
  },
  "com": {
    "configuredLevel": null,
    "effectiveLevel": "INFO"
  },
  "com.apusic": {
    "configuredLevel": null,
    "effectiveLevel": "INFO"
  },
  "com.apusic.ams": {
    "configuredLevel": null,
    "effectiveLevel": "INFO"
  }
}

```

从图中可以看到日志的相关等级，不同名称的记录器的有效级别，响应格式如下。

路径	类型	描述
----	----	----

levels	Array	日志系统支持的级别
loggers	Object	按名称键入的记录器
groups	Object	按名称键入的记录器组
loggers.*.configuredLevel	String	记录器的配置级别（如果有）
loggers.*.effectiveLevel	String	记录器的有效级别
groups.*.configuredLevel	String	记录器组的配置级别（如果有）
groups.*.members	Array	属于该组的记录器

可以通过 `/actuator/loggers/{logger.name}` 访问单个记录器的相关信息。

该端口下还有以下接口：

路径	描述	
<code>/actuator/loggers/{logger.name}</code>	访问单个记录器的相关信息	
<code>/actuator/loggers/{group.name}</code>	访问某个组下的记录器的相关信息	

20.3.2.6.1 设置日志级别

设置单个记录器

示例：我们选择 `com.apusic` 这个记录器进行设置，在设置前进入

`http://localhost:9999/actuator/loggers/com.apusic` 可以查看当前记录器的相关信息。

HTTP http://localhost:9999/actuator/loggers/com.apusic

GET http://localhost:9999/actuator/loggers/com.apusic

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

Key
Key

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ↕

```

1  {
2    "configuredLevel": null,
3    "effectiveLevel": "INFO"
4  }

```

然后需改请求为 POST，在Headers和Body中分别加入 Content-Type: application/json 和 {"configuredLevel":"debug"}，当Body中为空时即为清除日志级别。

HTTP http://localhost:9999/actuator/loggers/com.apusic

POST http://localhost:9999/actuator/loggers/com.apusic

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Headers 8 hidden

Key	Value
<input checked="" type="checkbox"/> Content-Type	application/json
Key	Value

HTTP <http://localhost:9999/actuator/loggers/com.apusic>

POST <http://localhost:9999/actuator/loggers/com.apusic>

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary **JSON** ▾

```
1 {"configuredLevel": "debug"}
```

然后发送请求后，再次发送GET请求可以看到日志级别已经变更为DEBUG。

HTTP <http://localhost:9999/actuator/loggers/com.apusic>

GET <http://localhost:9999/actuator/loggers/com.apusic>

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary **JSON** ▾

```
1 {"configuredLevel": "debug"}
```

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize **JSON** ▾

```
1 {
2   "configuredLevel": "DEBUG",
3   "effectiveLevel": "DEBUG"
4 }
```

设置组的日志级别

与设置单个记录器同理，只需要将路径中的 `{logger.name}` 替换为 `{group.name}` 即可。

20.3.2.7 threaddump

进入 `/actuator/threaddump` 之后可以看到以下页面:

Body Cookies Headers (5) Test Results

Pretty

Raw

Preview

Visualize

JSON



```

1  {
2    "threads": [
3      {
4        "threadName": "http-nio-9999-exec-10",
5        "threadId": 47,
6        "blockedTime": -1,
7        "blockedCount": 4,
8        "waitedTime": -1,
9        "waitedCount": 6,
10       "lockName": "java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject@47a58040",
11       "lockOwnerId": -1,
12       "lockOwnerName": null,
13       "inNative": false,
14       "suspended": false,
15       "threadState": "WAITING",
16       "stackTrace": [
17         {
18           "methodName": "park",
19           "fileName": "Unsafe.java",
20           "lineNumber": -2,
21           "className": "sun.misc.Unsafe",
22           "nativeMethod": true
23         },

```

20.3.2.8 metrics

进入 `/actuator/metrics` 之后可以看到以下页面

```

{
  "names": [
    "http.server.requests",
    "jvm.buffer.count",
    "jvm.buffer.memory.used",
    "jvm.buffer.total.capacity",
    "jvm.classes.loaded",
    "jvm.classes.unloaded",
    "jvm.gc.live.data.size",
    "jvm.gc.max.data.size",
    "jvm.gc.memory.allocated",
    "jvm.gc.memory.promoted",
    "jvm.gc.pause",
    "jvm.memory.committed",
    "jvm.memory.max",

```

```

    "jvm.memory.used",
    "jvm.threads.daemon",
    "jvm.threads.live",
    "jvm.threads.peak",
    "jvm.threads.states",
    "logback.events",
    "process.cpu.usage",
    "process.start.time",
    "process.uptime",
    "system.cpu.count",
    "system.cpu.usage"
  ]
}

```

以 `/actuator/metrics/{metric.name}` 方式发送GET请求，可以得到具体的指标的信息，以 `jvm.gc.max.data.size` 为例，浏览器会输出以下响应信息。

```

{
  "name": "jvm.gc.max.data.size",
  "description": "Max size of long-lived heap memory pool",
  "baseUnit": "bytes",
  "measurements": [
    {
      "statistic": "VALUE",
      "value": 1.431830528E9
    }
  ],
  "availableTags": []
}

```

响应结构如下:

路径	类型	描述
name	String	指标名称
description	String	指标描述

baseUnit	String	公制的基本单位
measurements	Array	指标的测量
measurements[].statistic	String	测量统计。(TOTAL、 、 、 、 、 、)。TOTAL_TIME_COUNT_ _ _ _MAX``VALUE``UNKNOWN``ACTIVE_TASKS``DURATION
measurements[].value	Number	测量值
availableTags	Array	可用于向下钻取的标签
availableTags[].tag	String	标签的名称
availableTags[].values	Array	标签的可能值

20.3.2.8.1 所有指标相关说明

指标	说明
http.server.requests	提供了有关应用程序处理的每个HTTP请求的信息，如请求计数、请求持续时间等
jvm.buffer.count	提供了有关JVM缓冲区使用情况的信息，如缓冲区数量、缓冲区使用率等
jvm.buffer.memory.used	已从JVM中卸载的类的数量，可以帮助开发人员了解应用程序的类加载器的性能
jvm.buffer.total.capacity	活跃对象的大小，可以帮助开发人员识别内存泄漏问题和调整JVM堆大小。
jvm.classes.loaded	堆的最大大小，可以帮助开发人员了解JVM堆的大小限制，并根据需要进行调整
jvm.classes.unloaded	已分配内存的总量，可以帮助开发人员了解应用程序的内存使用情况，并进行优化
jvm.gc.live.data.size	活跃对象的大小，可以帮助开发人员识别内存泄漏问题和调整JVM堆大小
jvm.gc.max.data.size	堆的最大大小，可以帮助开发人员了解JVM堆的大小限制，并根据需要进行调整
jvm.gc.memory.allocated	已分配内存的总量，这可以帮助开发人员了解应用程序的内存使用情况，并进行优化
jvm.gc.memory.promoted	通过垃圾回收已提升的对象大小，可以帮助开发人员了解垃圾回收器的性能，并调整堆大小和垃圾回收策略
jvm.gc.pause	垃圾回收暂停的次数和持续时间，可以帮助开发人员了解垃圾回收器的性能，并调整堆大小和垃圾回收策略
jvm.memory.committed	已提交的内存量，可以帮助开发人员了解应用程序的内存使用情况，并进行优化
jvm.memory.max	最大可用内存量，可以帮助开发人员了解JVM堆的大小限制，并根据需要进行调整
jvm.memory.used	已使用的内存量，可以帮助开发人员了解应用程序的内存使用情况，并进行优化

jvm.threads.daemon	守护线程的数量，可以帮助开发人员了解应用程序中守护线程的使用情况，并进行优化
jvm.threads.live	活动线程的数量，以帮助开发人员了解应用程序中线程的使用情况，并进行优化
jvm.threads.peak	峰值线程数，可以帮助开发人员了解应用程序中线程使用的最高峰值，并进行优化
jvm.threads.states	每种线程状态的数量，可以帮助开发人员了解应用程序中线程状态的分布情况，并进行优化
logback.events	Logback日志事件的数量，可以帮助开发人员了解应用程序中日志事件的数量和类型，并进行优化
process.cpu.usage	进程CPU使用率，可以帮助开发人员了解应用程序在CPU方面的性能，并进行优化
process.start.time	进程启动时间，可以帮助开发人员了解应用程序运行时间和稳定性
process.uptime	进程运行时间，可以帮助开发人员了解应用程序运行时间和稳定性
system.cpu.count	系统CPU核心数，可以帮助开发人员了解系统硬件规格，并优化应用程序性能
system.cpu.usage	系统CPU使用率，可以帮助开发人员了解系统在CPU方面的性能，并进行优化

20.3.2.9 scheduledtasks

给 `/actuator/scheduledtasks` 发送GET请求，可以看到如下界面:

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ↕

```

1  {
2    "cron": [],
3    "fixedDelay": [],
4    "fixedRate": [],
5    "custom": []
6  }

```

此时我们没有设置定时任务，响应的结构如下:

路径	类型	描述
cron	Array	Cron任务（如果有）。
cron[].runnable.target	String	将要执行的目标。
cron[].expression	String	Cron表达式。

fixedDelay	Array	修复延迟任务（如果有）。
fixedDelay. [].runnable.target	String	将要执行的目标。
fixedDelay [].initialDelay	Number	第一次执行之前的延迟（以毫秒为单位）。
fixedDelay [].interval	Number	上次执行结束与下一次执行开始之间的时间间隔（以毫秒为单位）。
fixedRate	Array	固定时间任务（如果有）。
fixedRate [].runnable.target	String	将要执行的目标。
fixedRate [].interval	Number	每次执行开始之间的时间间隔（以毫秒为单位）。
fixedRate [].initialDelay	Number	第一次执行之前的延迟（以毫秒为单位）。
custom	Array	具有自定义触发器的任务（如果有）。
custom [].runnable.target	String	将要执行的目标。
custom [].trigger	String	触发任务。

20.3.2.10 mappings

给 /actuator/mappings 发送GET请求，可以看到如下界面：

```

"contexts": {
  "application": {
    "mappings": {
      "dispatcherServlets": {
        "dispatcherServlet": [
          {
            "handler": "Actuator web endpoint 'loggers-name'",
            "predicate": "{POST [/actuator/loggers/{name}], consumes [application/vnd.spring-boot.actuator.v3+json | application/vnd.spring-boot.actuator.v2+json | application/json]}",
            "details": {
              "handlerMethod": {
                "className": "org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.OperationHandler",
                "name": "handle",
                "descriptor": "(Ljava/lang/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;"
              },
              "requestMappingConditions": {
                "consumes": [
                  {
                    "mediaType": "application/vnd.spring-boot.actuator.v3+json",
                    "negated": false
                  },
                  {
                    "mediaType": "application/vnd.spring-boot.actuator.v2+json",
                    "negated": false
                  },
                  {
                    "mediaType": "application/json",
                    "negated": false
                  }
                ]
              }
            }
          }
        ]
      }
    }
  }
}

```

20.4 附录D relaxed-path-chars和relaxed-query-chars配置示例

20.4.1 Linux

20.4.1.1 环境变量

```
export AAS_RELAXED_PATH_CHARS="\\"<>[\\]^`{|}"
export AAS_RELAXED_QUERY_CHARS="\\"<>[\\]^`{|}"
```

20.4.1.2 JVM参数

```
JAVA_OPTS="$JAVA_OPTS $JSSE_OPTS -Drelaxed-path-chars='\\"<>[\\]^`{|}' -Drelaxed-query-chars='\\"<>[\\]^`{|}'"
```

20.4.2 Spring boot配置文件

20.4.2.1 application.properties

```
server.aas.relaxed-path-chars=\<, >, [, ], \, ^, `, {, |, } # 允许特殊字符
server.aas.relaxed-query-chars=\<, >, [, ], \, ^, `, {, |, } # 允许特殊字符
```

20.4.2.2 application.yml

```
server:
  aas:
    relaxed-path-chars: ["\\", "<", ">", "[", "]", "\\", "^",
    "`", "{", "}"] # 允许特殊字符
server:
  aas:
    relaxed-query-chars: ["\\", "<", ">", "[", "]", "\\", "^",
    "`", "{", "}"] # 允许特殊字符
```

全国统一服务热线
4008-555-800



金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年,前身为“金蝶中间件公司”,是金蝶集团旗下新一代软件基础云平台服务商,云计算国家标准制定企业,国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业,也是“两网一站四库十二金”国家重点工程的基础平台提供商,产品广泛应用于政府、军工、金融、能源等关键行业,累计服务客户总数超过10万家。

Apusic
金蝶天燕

云计算国家标准制定企业
金蝶集团旗下基础软件企业
信息技术应用创新核心企业
官网: www.apusic.com

