

SuperSonic 部署配置文档

作者：罗天成

日期：2024.08.15

1. 文档概述

1.1 目的

1.2 官方文档

[在线体验](#)

[配置安装](#)

2. 环境要求

2.1 硬件要求

2.2 软件要求

[相关库及语言](#)

[其他配套](#)

3. 安装步骤

3.1 docker启动

[部署前提](#)

[启动流程](#)

3.2 本地IDE启动

[修正错误（可能新版本已修复）](#)

[启动流程](#)

3.3 源代码编译包启动（未验证）

4. 配置说明

4.1 数据库配置

[官方说明](#)

[SQL脚本分析](#)

[配置流程](#)

[Mysql数据库链接成功：](#)

4.2 大模型配置

[官方文档](#)

[有哪些国内的大模型服务对接？](#)

[OpenAI模型配置参考](#)

4.3 数据集配置

4.4 日志输出设置

[log4j日志设置](#)

修改 `logback-spring.xml` 文件

[解释](#)

[重新启动项目](#)

[验证日志输出](#)

[5. 验证与测试](#)

[5.1 数据库测试](#)

[5.2 大模型测试](#)

[6. 相关资料](#)

1. 文档概述

1.1 目的

项目本身含有官方说明文档，其中包含在线体验、docker启动、本地配置启动等说明档案。但由于项目还在完善开发中，存在说明不够细致以及各种安装指引不明确等问题，因此撰写这份作为补充说明，便于后续开发安装配置。

1.2 官方文档

在线体验

<http://117.72.46.148:9080/>

<http://117.72.46.148:9080/>

配置安装

快速体验

快速体验 # SuperSonic内置用于DEMO的语义模型和智能助理，因而只需要以下三步即可快速体验。启动系统 # 下载相应版本release包 解压zip包，执行启动脚本：sh bin/supersonic-daemon.sh start 访问浏览器：http://localhost:9080 注意 启动之前请安装好Java环境(JDK1.8)，Windows系统请到bin目录下执行 <https://supersonicbi.github.io/docs/快速体验/>

2. 环境要求

2.1 硬件要求

任意windows和linux系统电脑均可，分别具有相应的配置脚本。

2.2 软件要求

相关库及语言

Java 8（验证可用） / 11

maven \geq 3.8.0

node j's

npm

pnpm

其他配套

Docker \geq 26.0.0

任意Java开发IDE（IDEA、VScode等）

任意数据库（Mysql 8.0/8.4等）

3. 安装步骤

以windows系统为例

3.1 docker启动

部署前提

下载安装Docker和Docker Compose，并启动docker

Docker版本：26.0.0+

Docker Compose：v2.26.1+

启动流程

1. 下载项目：`git clone https://github.com/tencentmusic/supersonic.git`
2. 启动服务：`docker-compose up -d`

注意：支持按照版本启动，不指定版本默认是：latest

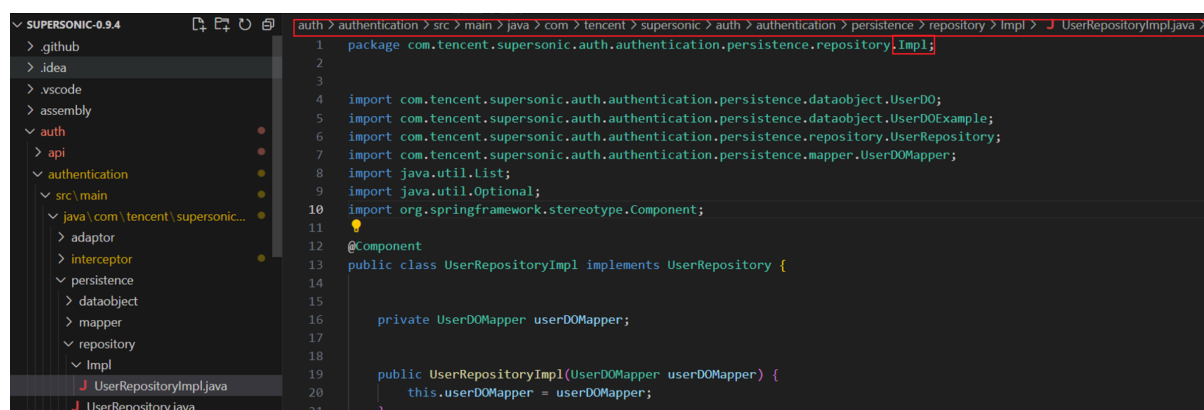
```
SUPERSONIC_VERSION=0.9.2-SNAPSHOT docker-compose up -d
```

3. 查看容器：`docker-compose ps`

4. 等待启动完成直接浏览器进入：<http://localhost:9080>
5. 查看进程日志信息：`docker exec -it supersonic_standalone bash`
6. 关闭服务：`docker-compose down`

3.2 本地IDE启动

修正错误（可能新版本已修复）



impl改大写：`package com.tencent.supersonic.auth.authentication.persistence.repository.Impl;`

启动流程

1. 下载项目：`git clone https://github.com/tencentmusic/supersonic.git`
2. 配置jdk以及maven必须：`java1.8 + maven ≥ 3.8.0`，参考IDE：IntelliJ IDEA（官方：后端编译需要JDK，推荐安装版本11）
3. 确保npm，nodejs保持最新即可，安装pnpm：`npm install -g pnpm`（官方：前端编译需要Node，推荐安装版本v20.16.0）
4. IDEA打开maven项目目录为根目录，等待识别project完成
5. `mvn clean install` 确保完成构建，大约2分钟
6. Windows：`.\assembly\bin\supersonic-build.bat webapp`
Linux：`sh assembly/bin/supersonic-build.sh webapp`
7. 检查是否构建完成 `.\assembly\build\supersonic-webapp.tar.gz`
8. 解压并移动生成的文件：

- 检查生成的 `supersonic-webapp.tar.gz` 文件：
`ls assembly/build/supersonic-webapp.tar.gz`
- 解压并移动文件到 `standalone/target/classes/` 目录下：
`cd assembly/build`
`tar xvf supersonic-webapp.tar.gz`
`mv supersonic-webapp ../launchers/standalone/target/classes/`
- 确认 `supersonic-webapp` 目录存在于 `standalone/target/classes/` 目录下：
`ls ../launchers/standalone/target/classes/supersonic-webapp`
- 也可以全程手动移动和检查，默认会在 `standalone/target/classes/` 目录下创建 `webapp` 文件夹，内容基本一致但是无法后续启动

9. 运行 `StandaloneLauncher` 类：

IDE进入目录：

```
.\launchers\standalone\src\main\java\com\tencent\supersonic\StandaloneLauncher.java
```

- 在文件顶部，点击绿色的三角形按钮（Run）
- 或者右键点击 `StandaloneLauncher` 类，然后选择 `Run 'StandaloneLauncher.main()'`

10. 成功运行

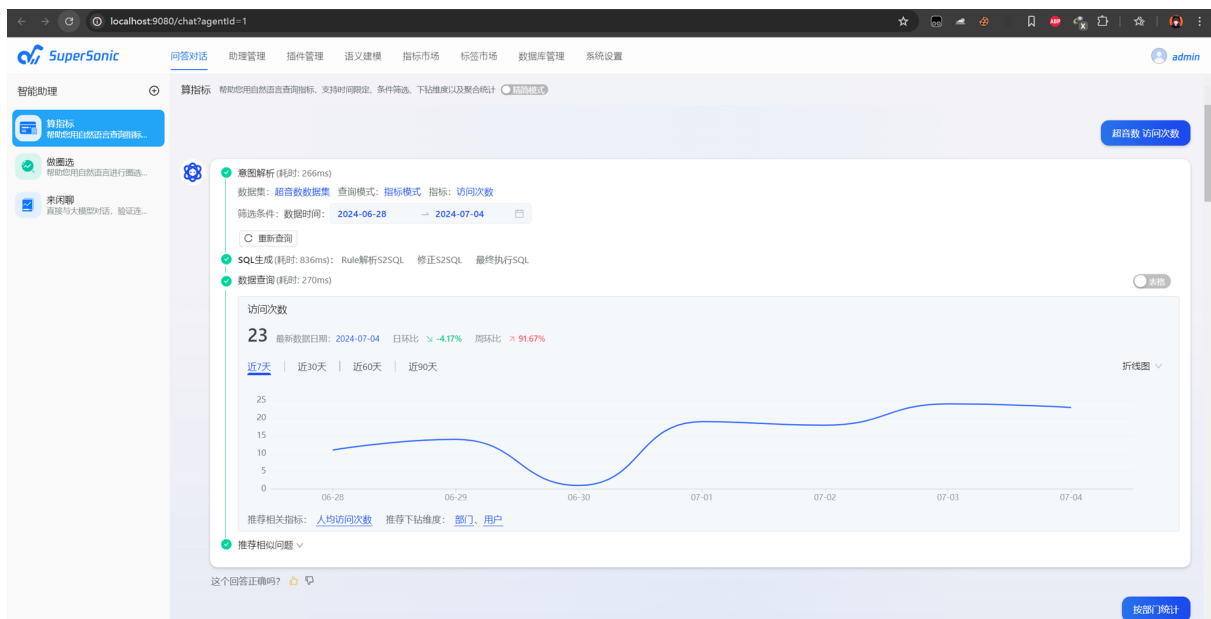
```

[INFO] [main] INFO C.e.s.h.chats.parser.llm.LLMRequestParser$8 - try generating query statement for dataset1...
[INFO] [main] INFO C.e.s.h.chats.parser.llm.LLMRequestParser$8 - currentTryRound=1, start runNextStep()
[INFO] [main] WARN dev.langchain4j.internal.RetryUtils$RetryException - Exception was thrown on attempt 1 of 3
[WARN] [main] java.util.concurrent.TimeoutException: Maximum number of tokens per request for demonstration purposes is 1000. If you wish to use more, please use your own OpenAI API key.
    at dev.langchain4j.model.openai.OpenAiChatModel.lambda$generate$6(OpenAiChatModel.java:184)
    at dev.langchain4j.internal.RetryUtils.withRetryPolicy(withRetryPolicy(RetryUtils.java:105))
    at dev.langchain4j.internal.RetryUtils.withRetry(RetryUtils.java:297)
    at dev.langchain4j.model.openai.OpenAiChatModel.generate(ChatOpenAiChatModel.java:263)
    at dev.langchain4j.model.openai.OpenAiChatModel.generate(ChatOpenAiChatModel.java:149)
    at dev.langchain4j.model.chat.ChatLanguageModel.generate(ChatLanguageModel.java:37)
    at com.tencent.supersonic.headless.chats.parser.llm.PassiveChatStrategy.lambda$generate$6(PassiveChatStrategy.java:49) ~[internal.jar]
    at java.util.concurrent.CompletableFuture$AsyncSupply.run(CompletableFuture.java:1770) ~[internal.jar]
    at java.util.concurrent.ForkJoinTask.execute(ForkJoinTask.java:711)
    at java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:289)
    at java.util.concurrent.ForkJoinTask.doExecute(ForkJoinTask.java:295)
    at java.util.concurrent.ForkJoinTask.awaitDone(ForkJoinTask.java:405) ~[internal.jar]
    at com.tencent.supersonic.headless.chats.parser.llm.PassiveChatStrategy.lambda$generate$6(PassiveChatStrategy.java:47)
    at com.tencent.supersonic.headless.chats.parser.llm.LLMRequestService.runNextStep(LLMRequestService.java:114)
    at com.tencent.supersonic.headless.chats.parser.llm.LLMRequestParser.tryParse(LLMRequestParser.java:49)
    at com.tencent.supersonic.headless.chats.parser.llm.LLMRequestParser.parse(LLMRequestParser.java:41)
    at com.tencent.supersonic.headless.server.web.service.impl.WorkflowServiceImpl.lambda$parseWorkflowParams$1(BrowserWorkflowServiceImpl.java:66)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at com.tencent.supersonic.headless.server.web.service.impl.WorkflowServiceImpl.parseWorkflowParams(BrowserWorkflowServiceImpl.java:65)
    at com.tencent.supersonic.headless.server.web.service.impl.WorkflowServiceImpl.startWorkflowFlow(BrowserWorkflowServiceImpl.java:51)

```

存在大模型接口设置问题，需要自行设置api，否则报错到达token上限（使用默认demo api仅1000 token长度限制）

11. 等待启动完成直接浏览器进入：<http://localhost:9080> 可以看到如下界面：



3.3 源代码编译包启动（未验证）

1. 下载相应版本source code
2. 执行编译脚本：`sh assembly/bin/supersonic-build.sh` 注：前端编译需要Node，推荐安装版本v20.16.0；后端编译需要JDK，推荐安装版本11
3. 编译完成后从 `assembly/build` 目录获取release包
4. 解压release包，`unzip supersonic-standalone-{revision}.zip`
5. 进入release目录，执行启动脚本 `sh bin/supersonic-daemon.sh start`
6. 访问浏览器：<http://localhost:9080/>

4. 配置说明

4.1 数据库配置

项目默认使用H2内存数据库，直接启动即可，但是无法保存信息，重新启动记录消失，如需要保存前序操作内容，需要自行配置相关数据库保存数据。

官方说明

配置DB

配置DB # 注意 系统默认使用H2内存数据库, 重启后会丢失数据, 若需要替换为自己的MySQL, 请按以下进行配置. 1. 执行SQL脚本 # 初次配置DB请依次执行conf/db下schema-mysql.sql、 data-mysql.sql, 这两个脚本均为最新表结构 若是已配置过DB并部署好的服务, 可参考sql-update.sql, 这里会注明每次功能改动需要改
≡ <https://supersonicbi.github.io/docs/系统部署/配置db/>

SQL脚本分析

SQL脚本分析

配置流程

以MySQL数据库为例：8.4LTS版本—8.0/5.7更优

1. 启动本地Mysql并链接

- 链接数据库：`mysql -h 127.0.0.1 -uroot -proot`
- 查看所有数据库：`show databases;`
- 查看该数据库下的表：`show tables;`

2. `CREATE DATABASE supersonic CHARACTER SET utf8 COLLATE utf8_unicode_ci;`

3. `USE supersonic;`

4. `SOURCE your_path\supersonic\launchers\standalone\src\main\resources\db\schema-mysql.sql;`

```
mysql> show tables;
+-----+
| Tables_in_supersonic |
+-----+
| artist                |
| files                 |
| genre                 |
| s2_agent              |
| s2_app                |
| s2_auth_groups        |
| s2_available_date_info |
| s2_canvas              |
| s2_chat               |
| s2_chat_config        |
| s2_chat_context       |
| s2_chat_memory        |
| s2_chat_parse         |
| s2_chat_query         |
| s2_chat_statistics    |
| s2_collect            |
| s2_data_set           |
| s2_database           |
| s2_dictionary_conf    |
| s2_dictionary_task    |
| s2_dimension          |
| s2_domain             |
| s2_metric             |
| s2_metric_query_default_config |
| s2_model              |
| s2_model_rela         |
| s2_plugin             |
| s2_pv_uv_statistic    |
| s2_query_rule         |
| s2_query_stat_info    |
| s2_semantic_pasre_info |
| s2_stay_time_statistic |
| s2_system_config      |
| s2_tag                |
| s2_tag_object         |
| s2_term               |
| s2_user               |
| s2_user_department    |
| singer               |
| song                 |
+-----+
40 rows in set (0.00 sec)
```

5. `SOURCE your_path\supersonic\launchers\standalone\src\main\resources\db\data-mysql.sql;`

```
ERROR 1366 (HY000): Incorrect string value: '\xAB\xE7\xB9\xE3\x80...' for column 'most_popular_in' at row 1
Query OK, 1 row affected (0.00 sec)

ERROR 1486 (22001): Data too long for column 'most_popular_in' at row 1
ERROR 1366 (HY000): Incorrect string value: '\xA1\x8C' for column 'g_name' at row 1
ERROR 1366 (HY000): Incorrect string value: '\x80\xA7' for column 'gender' at row 1
ERROR 1366 (HY000): Incorrect string value: '\x80\xA7' for column 'gender' at row 1
ERROR 1366 (HY000): Incorrect string value: '\x80\xA7' for column 'gender' at row 1
ERROR 1366 (HY000): Incorrect string value: '\x80\xA7' for column 'gender' at row 1
ERROR 1366 (HY000): Incorrect string value: '\x80\xA7' for column 'gender' at row 1
ERROR 1366 (HY000): Incorrect string value: '\x80\xA7' for column 'gender' at row 1
Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

ERROR 1366 (HY000): Incorrect string value: '\xA2\x8D \xE5\xB0\xBC...' for column 'song_name' at row 1
ERROR 1366 (HY000): Incorrect string value: '\xBA \xE5\xB8\x95\xE5...' for column 'song_name' at row 1
ERROR 1486 (22001): Data too long for column 'song_name' at row 1
ERROR 1486 (22001): Data too long for column 'song_name' at row 1
ERROR 1486 (22001): Data too long for column 'song_name' at row 1
ERROR 1366 (HY000): Incorrect string value: '\xAF\xAD' for column 'languages' at row 1
Query OK, 0 rows affected (0.00 sec)

mysql>
```

(存在部分数据导入问题，问题应为列长度不足，不影响整体数据暂保持不处理)

6. 配置YAML文件：将conf下 `application-local.yaml` 文件
(`supersonic\launchers\standalone\src\main\resources\application-local.yaml`) 默认的
DB - H2配置, 替换为本地MySQL或其他在线数据库


```

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/supersonic?useUnicode=true
    username: root
    password: root
  h2:
    console:
      path: /h2-console/semantic
      enabled: false
  config:
    import:
      - classpath:langchain4j-local.yaml

```

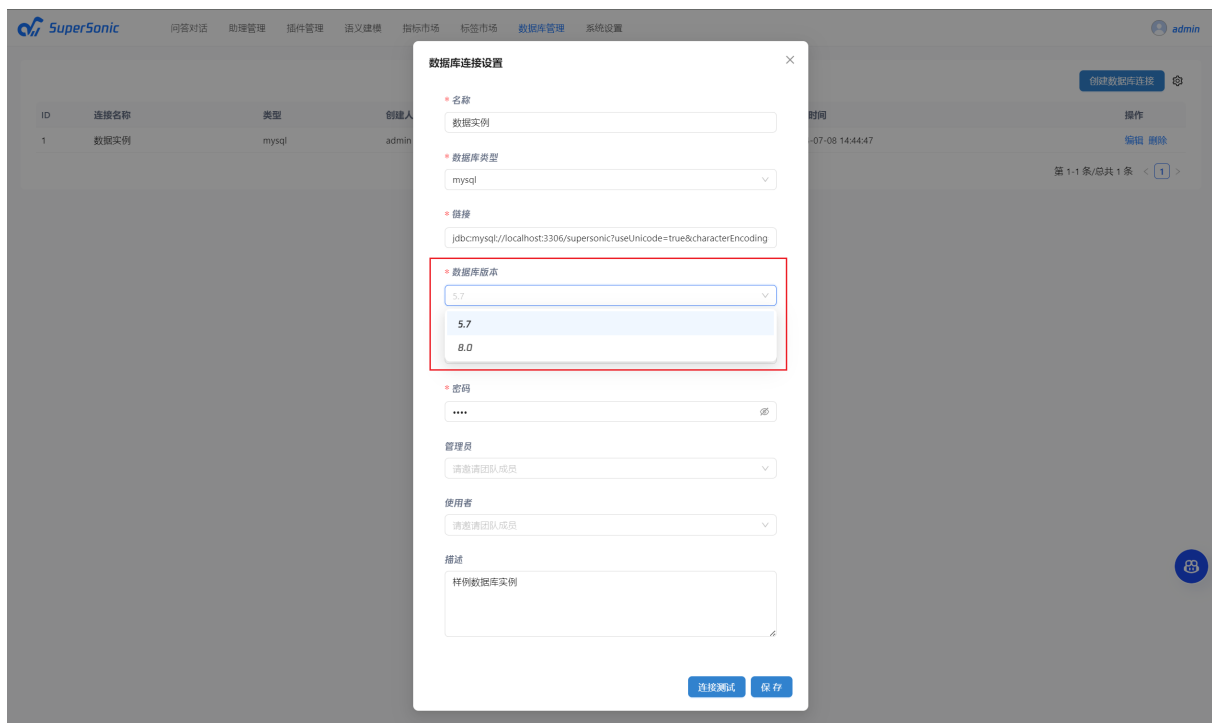
7. 进入网页设置Mysql服务器版本：

The screenshot displays the SuperSonic web application interface. The top navigation bar includes links for '问答对话' (Q&A Dialogue), '助理管理' (Assistant Management), '插件管理' (Plugin Management), '语义建模' (Semantic Modeling), '指标市场' (Indicator Market), '标签市场' (Tag Market), '数据库管理' (Database Management - highlighted with a red box), and '系统设置' (System Settings). The '数据库管理' section shows a table with the following data:

ID	连接名称	类型	创建人	描述	更新时间	操作
1	数据实例	mysql	admin	样例数据库实例	2024-07-08 14:44:47	编辑 删除

The '操作' (Operations) column for the first row contains two links: '编辑' (Edit) and '删除' (Delete). The '编辑' link is highlighted with a red box.

- 检查数据库配置是否正确并更改数据库版本为8.0：



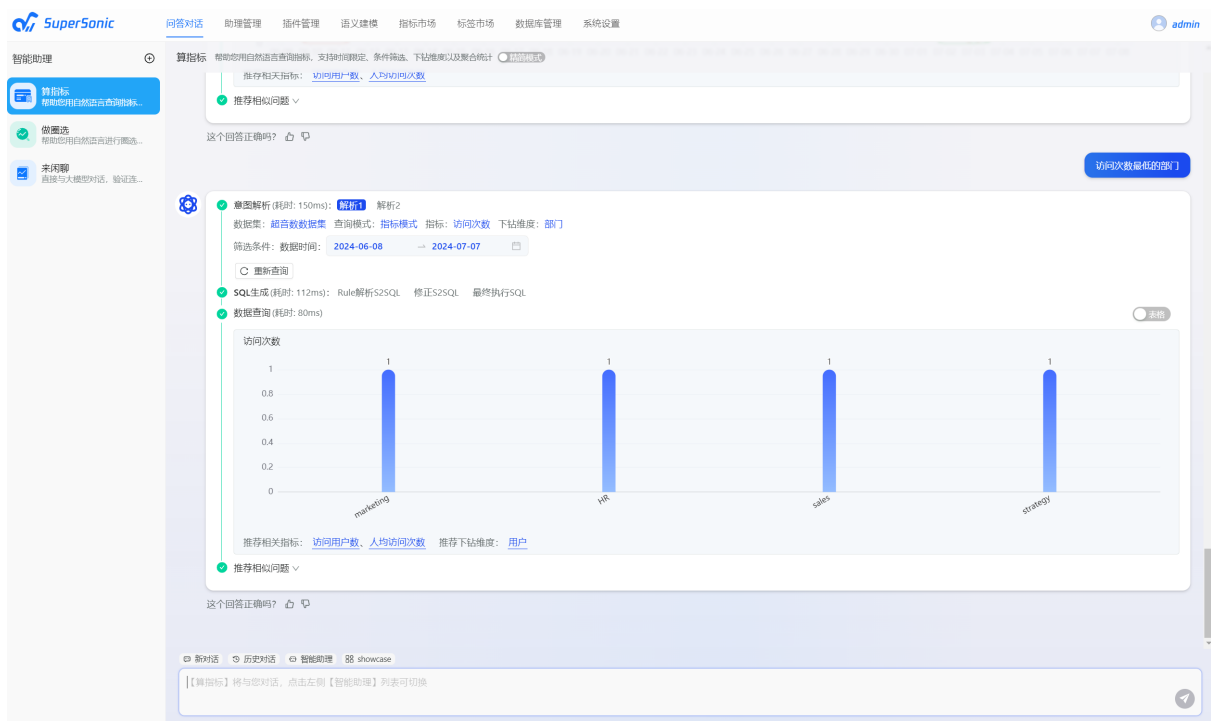
描述

样例数据库实例

连接测试

保存

Mysql数据库链接成功：



4.2 大模型配置

官方文档

非openai设置参考：<https://supersonicbi.github.io/docs/系统部署/配置llm/>

有哪些国内的大模型服务对接？

A: 当前我们验证过一些主流大模型服务，其申请链接如下表所示：

提供商	API申请链接	推荐模型
智谱AI	https://open.bigmodel.cn/api/paas/v4	glm-4
阿里云	https://dashscope.aliyuncs.com/compatible-mode/v1	qwen-max
幻方	https://api.deepseek.com	deepseek-chat
月之暗面	https://api.moonshot.cn/v1	moonshot-v1-8k

OpenAI模型配置参考

1. 创建OpenAI的api端口获取密钥：

OpenAI Platform

Explore developer resources, tutorials, API docs, and dynamic examples to get the most out of OpenAI's platform.

 <https://platform.openai.com/api-keys>

OpenAI Platform



2. 测试链接：`curl https://api.openai.com/v1/models -H "Authorization: Bearer your_api_key"`

3. 测试api返回：

```
import openai

openai.api_key = "your_api_key"

response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
        {"role": "system", "content": "You are a helpful assistant"},
        {"role": "user", "content": "Say this is a test"}
    ]
)

print(response.choices[0].message['content'])
```

4. 进入路径：`your_path\supersonic\launchers\standalone\src\main\resources\langchain4j-local.yaml`

5. 设置 `langchain4j-local.yaml`

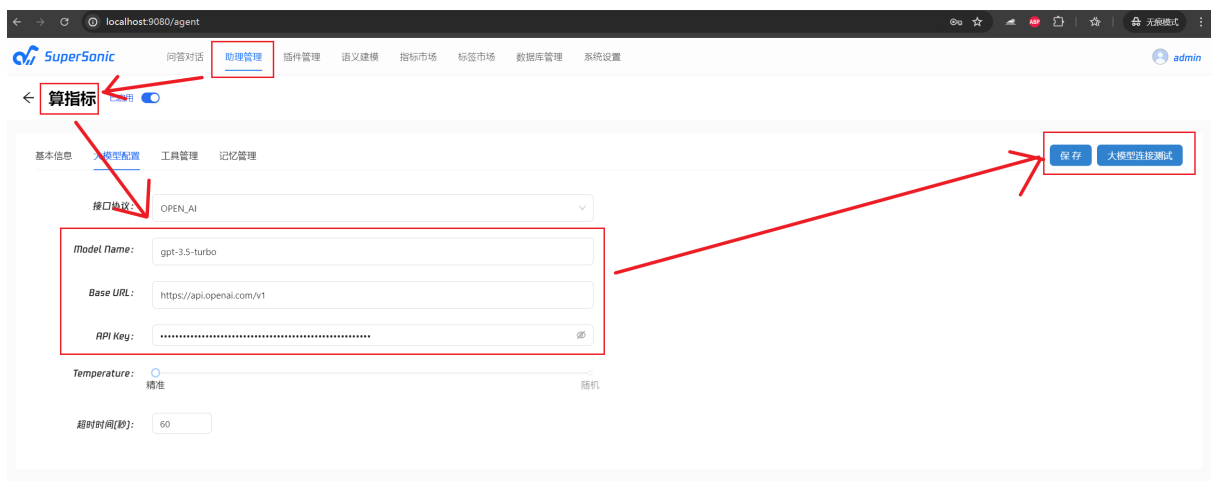
```
langchain4j:
  # Replace `open_ai` with ollama/hipu/azure/dashscope as needed
  # Note:
  # 1. `open_ai` is commonly used to connect to cloud-based models
  # 2. `ollama` is commonly used to connect to local models.
  open-ai:
    chat-model:
      # It is recommended to replace with your API key in production
      # Note: The default API key `demo` is provided by langchain4j
```

```
#          which limits 1000 tokens per request.
#
#  base-url: ${OPENAI_API_BASE:https://api.openai.com/v1}
#  api-key:  ${OPENAI_API_KEY:demo}
#  model-name: ${OPENAI_MODEL_NAME:gpt-3.5-turbo}
#  temperature: ${OPENAI_TEMPERATURE:0.0}
#  timeout:  ${OPENAI_TIMEOUT:PT60S}

base-url: ${OPENAI_API_BASE:https://api.openai.com/v1}
api-key:  ${OPENAI_API_KEY:your_api_key}
model-name: ${OPENAI_MODEL_NAME:gpt-3.5-turbo}
temperature: ${OPENAI_TEMPERATURE:0.0}
timeout:  ${OPENAI_TIMEOUT:PT60S}
```

6. 保存并启动supersonic，启动方式见前

7. 登录并按照如下顺序配置：



4.3 数据集配置

系统自带默认数据集Supersonic/超音速，自行数据集配置参考官方文档

SuperSonic

组装数据集 # 1. 挑选指标维度 # 如下图，创建数据集时，首先填写好数据集的名称等基本信息，然后即可进入到下一步，可以选择相同主题域下的所有模型，并挑选符合场景需要的指标维度加入到数据集中。注意 加入到数据集的指标维度所在的模型必须已在构建模型时配置好关联关系，否则将会提示报错。 2. 查询设置 #
<https://supersonicbi.github.io/docs/headless-bi/组装数据集/>

Headless BI

概念

连接数据库

构建模型

创建指标

管理指标

组装数据集

配置数据权限

标签

4.4 日志输出设置

默认使用INFO级别输出，会缺少大量中间信息输出，需要手动调整

log4j日志设置

目录：`\supersonic\launchers\standalone\src\main\resources\logback-spring.xml`

`logback-spring.xml` 是Logback配置文件的一种形式。如果您的项目使用的是Spring框架并且配置文件被命名为 `logback-spring.xml`，那么它将由Spring Boot自动加载。

要增加日志的详细程度，将日志级别设置为 `DEBUG`，可以按以下步骤进行：

修改 `logback-spring.xml` 文件

1. 将 `<root>` 元素中的 `level` 属性修改为 `DEBUG`。
2. 确保您感兴趣的 `logger` 元素也设置为 `DEBUG`。

以下是修改后的 `logback-spring.xml` 示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="true">
  <contextName>logback</contextName>
  <!--      <property name="LOG_PATH" value="${logback.logdir:-logs}"/>-->
```

```

<property name="LOG_PATH" value="${LOG_PATH:-logs}"/>
<property name="LOG_APPNAME" value="chat"/>
<!-- 输出到控制台 -->
<appender name="consoleLog" class="ch.qos.logback.core.
ConsoleAppender">
    <encoder>
        <pattern>%d{HH:mm:ss} [%thread] %-5level %logger{36} %line - %msg%n</pattern>
    </encoder>
</appender>

    <appender name="fileInfoLog" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <File>${LOG_PATH}/info.${LOG_APPNAME}.log</File>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <FileNamePattern>${LOG_PATH}/info.${LOG_APPNAME}.%d{yyyy-MM-dd}.log.gz</FileNamePattern>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <charset>UTF-8</charset>
            <pattern>%d [%thread] %-5level [%X{traceId}] %logger{36} %line - %msg%n</pattern>
        </encoder>
    </appender>

    <appender name="fileErrorLog" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>ERROR</level>
        </filter>
        <File>${LOG_PATH}/error.${LOG_APPNAME}.log</File>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <FileNamePattern>${LOG_PATH}/error.${LOG_APPNAME}.%d{yyyy-MM-dd}.log.gz</FileNamePattern>

```

```

        <maxHistory>90</maxHistory>
    </rollingPolicy>
    <encoder>
        <charset>UTF-8</charset>
        <pattern>%d [%thread] %-5level [%X{traceId}] %logger{36} %line - %msg%n</pattern>
    </encoder>
</appender>

    <appender name="serviceLog" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <File>${LOG_PATH}/serviceinfo.${LOG_APPNAME}.log</File>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <FileNamePattern>${LOG_PATH}/serviceinfo.${LOG_APPNAME}.%d{yyyy-MM-dd}.log.gz</FileNamePattern>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <charset>UTF-8</charset>
            <pattern>%d [%thread] %-5level [%X{traceId}] %logger{36} %line - %msg%n</pattern>
        </encoder>
    </appender>

    <logger name="com.tencent.supersonic" level="DEBUG" additivity="true">
        <appender-ref ref="serviceLog"/>
    </logger>

    <!-- 业务日志输出 -->
    <appender name="keyPipelineAppender" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <File>${LOG_PATH}/keyPipeline.log</File>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${LOG_PATH}/keyPipeline.%d{yyy

```



```

y-MM-dd}.log</fileNamePattern>
    <maxHistory>30</maxHistory>
    <cleanHistoryOnStart>true</cleanHistoryOnStart>
</rollingPolicy>
<encoder>
    <charset>UTF-8</charset>
    <pattern>%d [%thread] %-5level [%X{traceId}] %l
ogger{36} %line - %msg%n</pattern>
</encoder>
    <!--<filter class="ch.qos.logback.classic.filter.L
evelFilter">-->
        <!--<level>DEBUG</level>-->
    <!--</filter>-->
</appender>
<!--keyPipeline相关日志-->
<logger name="keyPipeline" level="DEBUG" additivity="fa
lse">
    <appender-ref ref="keyPipelineAppender"/>
</logger>

<root level="DEBUG">
    <appender-ref ref="fileInfoLog"/>
    <appender-ref ref="fileErrorLog"/>
    <appender-ref ref="consoleLog"/>
</root>
</configuration>

```

解释

全部更改为DEBUG级别

重新启动项目

完成配置修改后，重新编译并启动您的项目，查看控制台输出和日志文件，确认日志级别设置已生效。

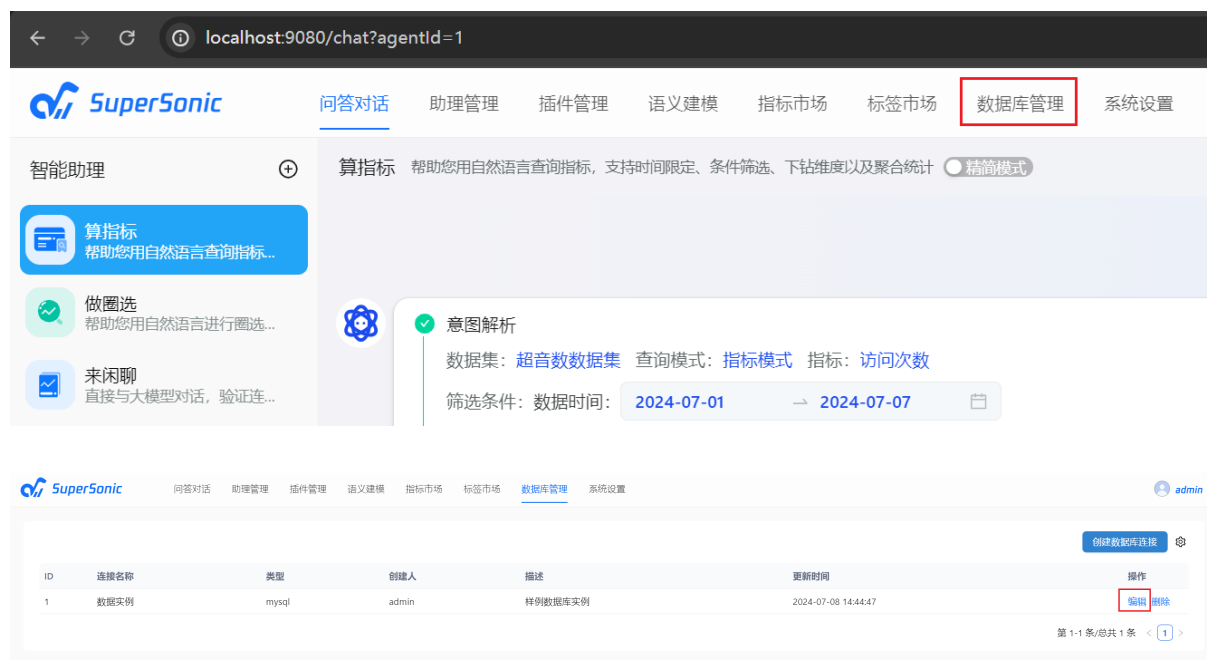
验证日志输出

确认您在控制台和日志文件中可以看到 `DEBUG` 级别的日志消息，包括之前无法看到的 `"before handleNoMetric, sql:{}"` 和 `"after handleNoMetric, sql:{}"` 日志输出。

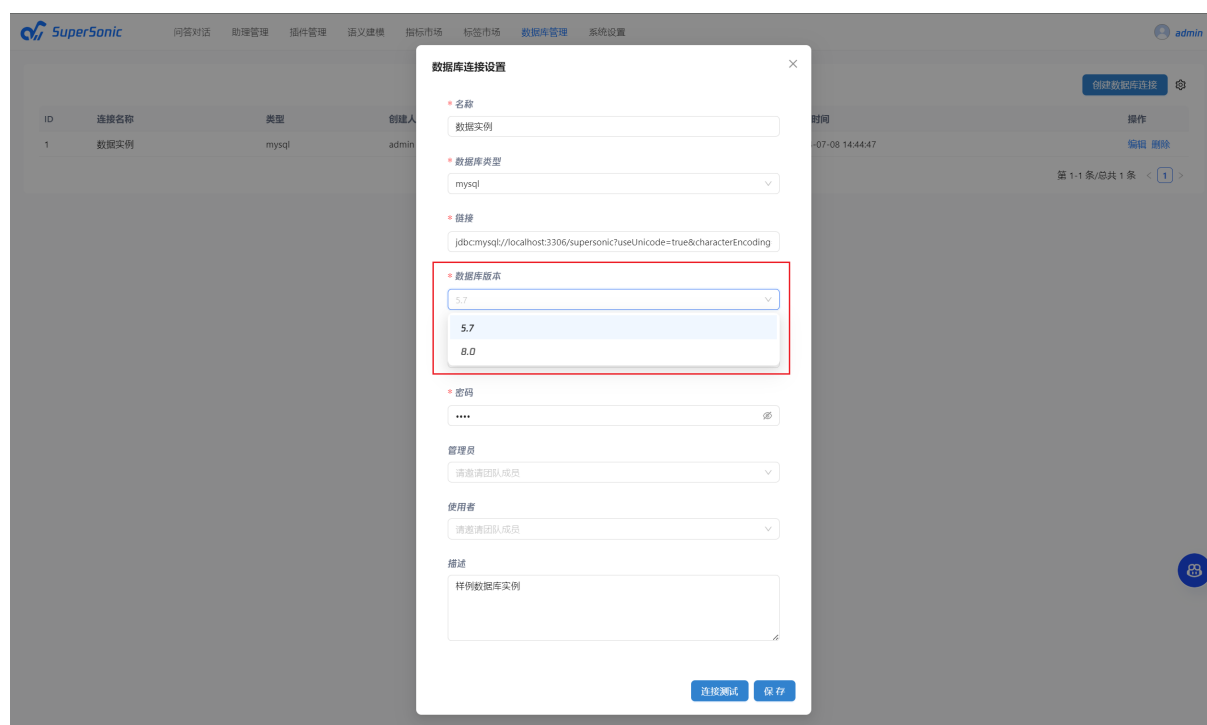
5. 验证与测试

5.1 数据库测试

入网页设置Mysql服务器版本：



- 检查数据库配置是否正确并更改数据库版本为8.0：



描述

样例数据库实例

连接测试

保存

以及直接控制台代码测试数据库内容

- 链接数据库：`mysql -h 127.0.0.1 -uroot -proot`
- 查看所有数据库：`show databases;`
- 查看该数据库下的表：`show tables;` 等

5.2 大模型测试

控制台测试/新调试创建接口：

库管理



连接成功

```
5 response = openai.ChatCompletion.create(  
6     model="gpt-3.5-turbo",  
7     messages=[  
8         {"role": "system", "content": "你是一个助手"},  
9         {"role": "user", "content": "请问你是什么大模型"}  
10    ]  
11 )
```

```
D:\JAVA\test\Supersonic-Anaylsis>C:/Users/timlu/AppData/Local/Programs/Python/Python310/python.exe d:/JAVA/test/Supersonic-Anaylsis/test-api.py  
我是一个语言模型助手，可以回答你关于各种主题的问题，并提供一些帮助和建议。如果您有任何问题或需要帮助，请随时告诉我！
```

测试代码及结果：

```
import openai  
  
openai.api_key = "YOUR_API_KEY"
```

```

response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
        {"role": "system", "content": "You are a helpful assistant"},
        {"role": "user", "content": "Say this is a test"}
    ]
)

print(response.choices[0].message['content'])

# response = openai.Model.list()
# print(response)

```

超音数近半年哪个月的访问次数汇总最高

意图解析 (耗时: 3006ms): **解析1** 解析2 解析3
数据集: 超音数数据集 查询模式: 指标模式 指标: 访问次数
筛选条件: 数据时间: 2024-01-10 → 2024-07-10

重新查询

SQL生成 (耗时: 118ms): **Schema映射** Few-shot示例 LLM解析S2SQL 修正S2SQL 最终执行SQL
名称: 访问用户数、用户、部门、访问次数、停留时长、人均访问次数、页面、数据日期

数据查询 (耗时: 38ms)

date	访问次数
6	390

推荐相似问题 ▾

这个回答正确吗?

LLM配置使用说明：

SuperSonic

配置助理 # 1. 信息配置 # 助理的配置主要分为三个部分: 基本信息的配置 大模型连接配置 工具配置

1.1 基本信息的配置 # 名称: 即助理的名称 支持联想: 默认开启, 开启后, 用户在问答对话中输入问题时, 会进行联想提示 开启多轮: 默认关闭, 开启后, 用户在问答对话时, 可自动保存上下文, 提供多轮对话的效果 示例问题:

≡ <https://supersonicbi.github.io/docs/chat-bi/配置助理/>

最终直接进行“来闲聊”模块对话即可验证是否连接成功。

6. 相关资料

Docker部署

部署前提 # 下载安装Docker和Docker Compose，并启动docker Docker版本：26.0.0+ Docker Compose：v2.26.1+ 注意 1 如果报如下错: Cannot connect to the Docker daemon at unix:///Users/lexluo/.docker/run/docker.sock. Is the docker daemon running? 说明docker没有启动 2
≡ <https://supersonicbi.github.io/docs/系统部署/docker部署/>

https://github.com/tencentmusic/supersonic/blob/master/README_CN.md

<https://www.jianshu.com/p/c51d92a9f91d>

windows安装npm教程_npm 安装-CSDN博客

文章浏览阅读10w+次，点赞187次，收藏436次。在使用之前，先类掌握3个东西，明白它们是用来干什么的：npm: nodejs 下的包管理器。webpack: 它主要用途是通过CommonJS 的语法把所有浏览器端需要发布的静态资源作相应的准备，比如资源的合并和打包。vue-cli: 用户生成Vue工程模板。（帮你快速开始一个


 <https://blog.csdn.net/zhouyan8603/article/details/109039732>

下载 | Node.js 中文网

 <https://nodejs.cn/download/>

pnpm 基本详细使用教程（安装、卸载、使用、可能遇到的问题及解决办法）-CSDN博客

文章浏览阅读6.6w次，点赞120次，收藏240次。本文详细介绍了PNPM8.x版本的安装过程，包括通过npm和HomeBrew安装，以及如何配置镜像源、修改默认安装路径。重点讲述了如何处理可能遇到的环境变量设置问题，如pnpmsetup、手动配置和验证全局安装路径。

 https://blog.csdn.net/m0_56416743/article/details/136122153