《Ai Agent》第3-3节：Ai Agent 测试案例

来自： 码农会锁

小傅哥

2025年06月07日 14:37

本章重点： ★★☆☆☆
课程视频： https://t.zsxq.com/jl0BD
代码分支：https://gitcode.net/KnowledgePlanet/ai-agent-station-study/-/tree/3-3-agent-case
工程代码： https://gitcode.net/KnowledgePlanet/ai-agent-station-study

作者：小傅哥
博客：https://bugstack.cn

> 沉淀、分享、成长，让自己和他人都能有所收获！😄

一、本章诉求

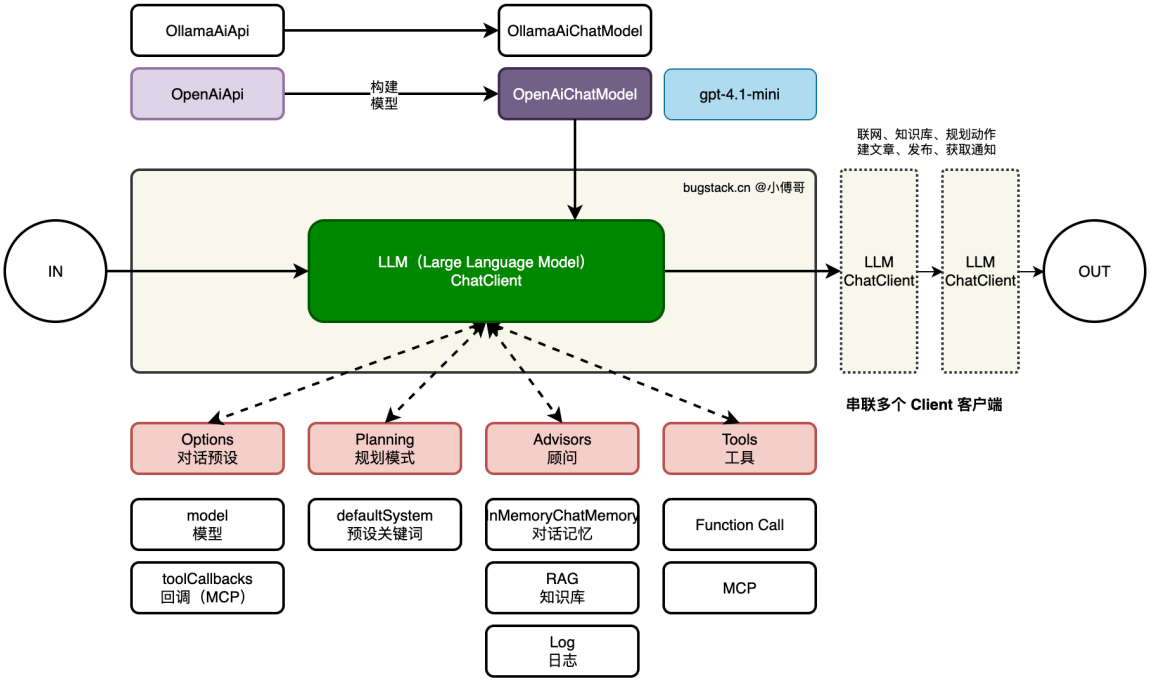
在项目中引入 Spring Ai 1.0.0 框架，通过编写测试案例的方式，了解 Ai Agent 的工作模式。

那为什么要这么做呢?

通常对于软件设计的解决方案，我们都有一个这样共识，那就是目标结果驱动，最先搭建可运行的最小执行单元。因为软件设计原则，康威定律，也提到，大的系统组织总是比小系统更倾向于分解。当场景问题被拆解的越小以后，也就越容易被理解和处理。所以，我们要优先通过案例的方式，验证 Ai Agent 的工作模型和可执行方案。再通过这些案例，设计详细的流程和库表细节。

二、功能流程

如图，为整个 Ai Agent 的工作模型；



概念：**AiAgent是整合多种技术手段的智能实体**，其实现依赖于 Tools、MCP、Memory、RAG（Retrieval-Augmented Generation，检索增强生成）等技术组件构建的智能体。并且每一个 Agent Client 又可以被连接通信，增强其 Agent 智能体能力。

方案：这里我们基于 Spring AI 框架，通过编码的方式把模型、关键词、顾问角色、工具，放入到 LLM 客户端，构建 LLM 对话智能体。

三、框架说明

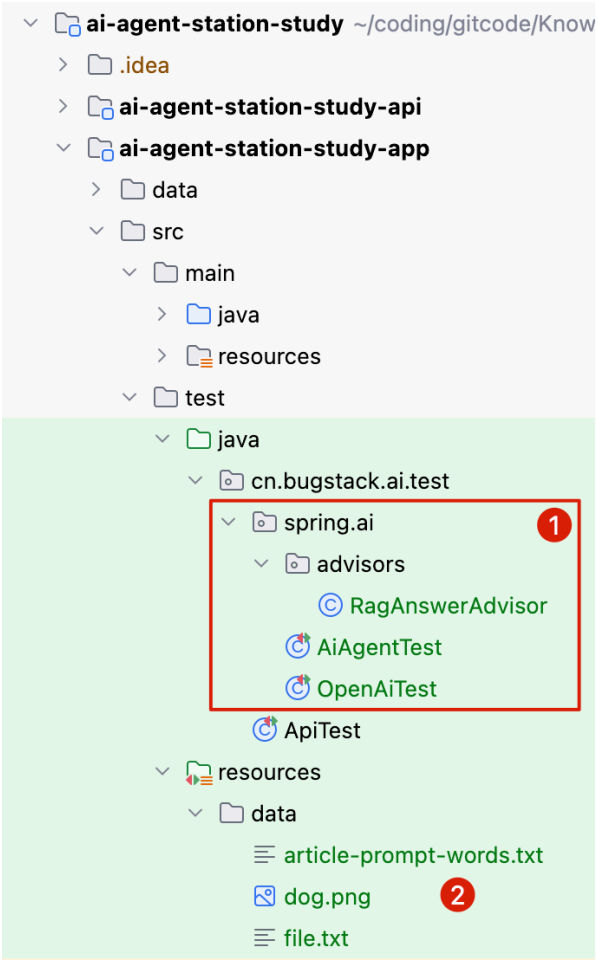
Spring Ai 官网文档：https://docs.spring.io/spring-ai/reference/

```
<!-- spring ai 1.0.0 https://central.sonatype.com/artifact/org.springframework.ai/spring-ai-bom -->
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-bom</artifactId>
    <version>1.0.0</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

目前 Spring Ai 最新稳定版本为 1.0.0，从我们的项目年初开始时，还是 0.8.1，中间简历了 M3 ~ M8 版本的迭代，最终稳定到 1.0.0 版本。可以说 Spring Ai 迭代的速度非常快，整体的框架结构设计也越来越成熟。基本做 Java 开发的，以后也会越来越多的选择 Spring AI 框架完成 AI 应用项目的落地。

四、编码实现

1. 工程结构



1 Ai 功能测试

2 图片、知识库文件 - 可以按需修改

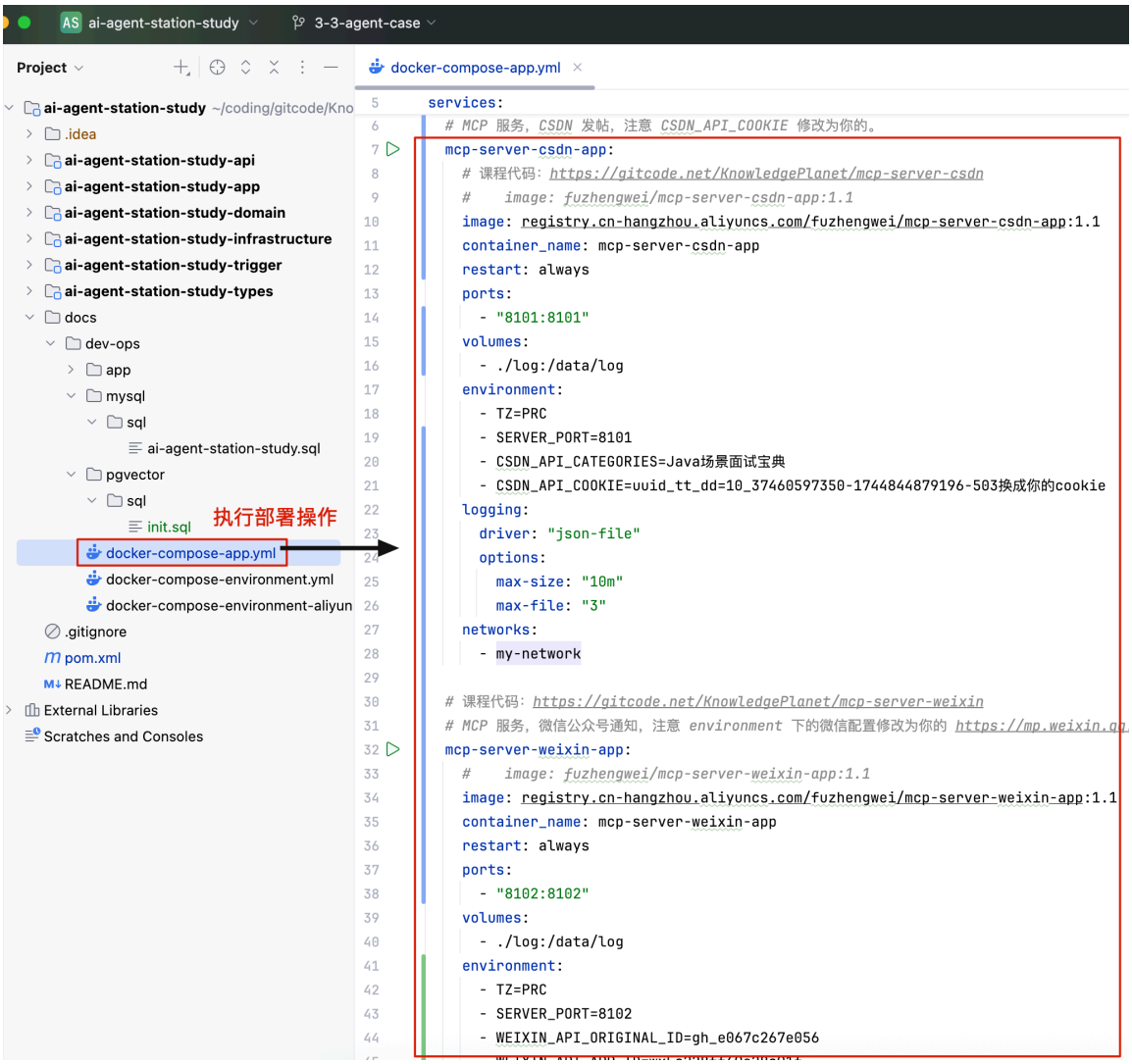
在工程 ai-agent-station-study-app 模块下，test 里增加了 OpenAi 测试方法，作为本节的功能验证。

如下说明会带着你做相关的配置和测试验证。

2. 前置说明

2.1 mcp 安装

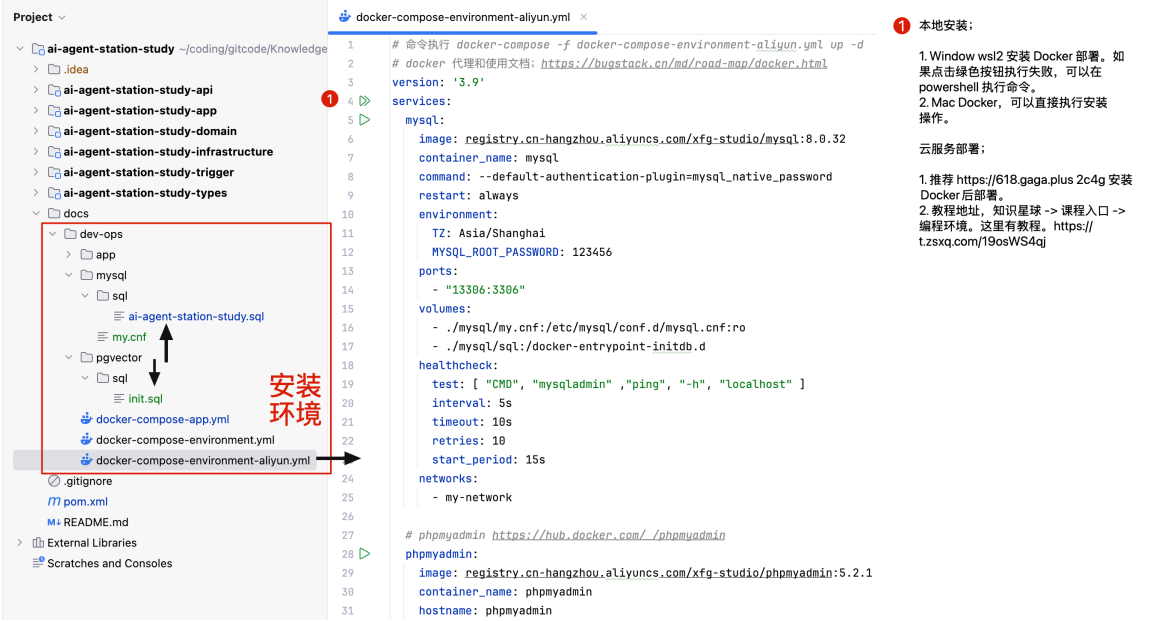
整个 Ai rag、mcp、agent 课程是一个递进的关系，前面的 rag、mcp，都是为了目前到 agent 阶段的整体串联使用。
那么前面章节所涉及 MCP 部分，本节也需要使用。在开始下面的课程之前，你可以把 mcp 服务进行云服务器部署，之后本节就可以对接了。在课程的第2阶段，mcp 部分有相关的讲解



首先，在课程代码中，ai-agent-station-study 3-3-agent-case 分支下，dev-ops 部署脚本中，提供了两个 mcp 服务的部署。

之后，拿到部署脚本，需要先配置脚本参数，csdn 需要配置cookie，weixin 需要配置连接信息。配置好这些信息后，可以把整个 dev-ops 脚本到放到安装好 Docker 脚本的云服务器上，执行部署。详细可以阅读第2阶段课程，MCP 服务部署上线（sse 模式） 第4部分，服务部署。

2.2 环境安装



本地安装；

1. Window wsl2 安装 Docker 部署。如果点击绿色按钮执行失败，可以在 powershell 执行命令。
2. Mac Docker，可以直接执行安装操作。

云服务部署；

1. 推荐 https://618.gaga.plus 2c4g 安装Docker后部署。
2. 教程地址，知识星球 -> 课程入口 -> 编程环境。这里有教程。https://t.zsxq.com/19osWS4qj 部署完成后记得在安全组开放端口。

3. 引入框架

3.1 根目录引入框架清单pom

```
<!-- spring ai 1.0.0 https://central.sonatype.com/artifact/org.springframework.ai/spring-ai-bom -->
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-bom</artifactId>
    <version>1.0.0</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

3.2 app 模块下，引入ai框架

```
<!-- spring ai v1.0.0 start -->
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-starter-model-openai</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-starter-vector-store-pgvector</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-starter-mcp-client-webflux</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-tika-document-reader</artifactId>
</dependency>
<!-- spring ai v1.0.0 stop -->
```

> 目前的 Spring Ai 框架很规整，整个顺序为；`starter -> autoconfig -> model`，对于需要使用的 Ai 模型，引入对应的 Starter 即可，它已经在内部帮我们封装配置、连接、启动操作。如果感兴趣可以翻看下源码
>
> 这里暂时我们只需要用到 openai 模型、pgvector 向量库、mcp 通信客户端、知识库工具包。

4. yml 配置

```
# 数据库配置；启动时配置数据库资源信息
spring:
  datasource:
    username: postgres
    password: postgres
    url: jdbc:postgresql://192.168.1.108:15432/ai-rag-knowledge
    driver-class-name: org.postgresql.Driver
    type: com.zaxxer.hikari.HikariDataSource
    hikari:
      pool-name: Retail_HikariCP
      minimum-idle: 15 #最小空闲连接数量
```

```
    idle-timeout: 180000 #空闲连接存活最大时间，默认600000（10分钟）
    maximum-pool-size: 25 #连接池最大连接数，默认是10
    auto-commit: true  #此属性控制从池返回的连接的默认自动提交行为,默认值：true
    max-lifetime: 1800000 #此属性控制池中连接的最长生命周期，值0表示无限生命周期，默认1800000即30分钟
    connection-timeout: 30000 #数据库连接超时时间,默认30秒，即30000
    connection-test-query: SELECT 1
ai:
  openai:
    base-url: https://apis.itedus.cn
    api-key: sk-63b2LihHwKXMjCFQB3C501FeD20b47F6**** 可以联系小傅哥申请
```

地址：ai-agent-station-study -> ai-agent-station-study-app -> application-dev.yml 修改配置。

配置你的数据库地址 url: jdbc:postgresql://192.168.1.108:15432/ai-rag-knowledge 和 ai 下的 openai 对应的 api-key

5. 功能测试

5.1 API 验证

```
@Slf4j
@RunWith(SpringRunner.class)
@SpringBootTest
public class OpenAiTest {

    @Value("classpath:data/dog.png")
    private Resource imageResource;

    @Value("classpath:data/file.txt")
    private Resource textResource;

    @Value("classpath:data/article-prompt-words.txt")
    private Resource articlePromptWordsResource;

    @Autowired
    private OpenAiChatModel openAiChatModel;

    @Autowired
    private PgVectorStore pgVectorStore;

    private final TokenTextSplitter tokenTextSplitter = new TokenTextSplitter();

    @Test
    public void test_call() {
        ChatResponse response = openAiChatModel.call(new Prompt(
                "1+1",
                OpenAiChatOptions.builder()
                        .model("gpt-4o")
                        .build()));
        log.info("测试结果(call):{}", JSON.toJSONString(response));
    }

    @Test
    public void test_call_images() {
        UserMessage userMessage = UserMessage.builder()
                .text("请描述这张图片的主要内容，并说明图中物品的可能用途。")
                .media(org.springframework.ai.content.Media.builder()
                        .mimeType(MimeType.valueOf(MimeTypeUtils.IMAGE_PNG_VALUE))
                        .data(imageResource)
                        .build())
                .build();

        ChatResponse response = openAiChatModel.call(new Prompt(
```

```
            userMessage,
            OpenAiChatOptions.builder()
                    .model("gpt-4o")
                    .build()));

    log.info("测试结果(images):{}", JSON.toJSONString(response));
}

@Test
public void test_stream() throws InterruptedException {
    CountDownLatch countDownLatch = new CountDownLatch(1);

    Flux<ChatResponse> stream = openAiChatModel.stream(new Prompt(
            "1+1",
            OpenAiChatOptions.builder()
                    .model("gpt-4o")
                    .build()));

    stream.subscribe(
            chatResponse -> {
                AssistantMessage output = chatResponse.getResult().getOutput();
                log.info("测试结果(stream): {}", JSON.toJSONString(output));
            },
            Throwable::printStackTrace,
            () -> {
                countDownLatch.countDown();
                log.info("测试结果(stream): done!");
            }
    );

    countDownLatch.await();
}

@Test
public void upload() {
    // textResource、articlePromptWordsResource
    TikaDocumentReader reader = new TikaDocumentReader(articlePromptWordsResource);

    List<Document> documents = reader.get();
    List<Document> documentSplitterList = tokenTextSplitter.apply(documents);

    documentSplitterList.forEach(doc -> doc.getMetadata().put("knowledge", "article-prompt-words"));

    pgVectorStore.accept(documentSplitterList);

    log.info("上传完成");
}

@Test
public void chat() {
    String message = "王大瓜今年几岁";

    String SYSTEM_PROMPT = """
            Use the information from the DOCUMENTS section to provide accurate answers but act as if you
            If unsure, simply state that you don't know.
            Another thing you need to note is that your reply must be in Chinese!
            DOCUMENTS:
                {documents}
            """;

    SearchRequest request = SearchRequest.builder()
            .query(message)
            .topK(5)
            .filterExpression("knowledge == '知识库名称-v4'")
```

```
                .build();

        List<Document> documents = pgVectorStore.similaritySearch(request);

        String documentsCollectors = null == documents ? "" : documents.stream().map(Document::getText).col

        Message ragMessage = new SystemPromptTemplate(SYSTEM_PROMPT).createMessage(Map.of("documents", docu

        ArrayList<Message> messages = new ArrayList<>();
        messages.add(new UserMessage(message));
        messages.add(ragMessage);

        ChatResponse chatResponse = openAiChatModel.call(new Prompt(
                messages,
                OpenAiChatOptions.builder()
                        .model("gpt-4o")
                        .build()));

        log.info("测试结果:{}", JSON.toJSONString(chatResponse));
    }

}
```

常用的 API 以这样几个方法验证，test_call、test_call_images、test_stream、upload、chat 可以分别测试使用。

5.2 Ai Agent 测试

```
@Slf4j
@RunWith(SpringRunner.class)
@SpringBootTest
public class AiAgentTest {

    private ChatModel chatModel;

    private ChatClient chatClient;

    @Resource
    private PgVectorStore vectorStore;

    public static final String CHAT_MEMORY_CONVERSATION_ID_KEY = "chat_memory_conversation_id";
    public static final String CHAT_MEMORY_RETRIEVE_SIZE_KEY = "chat_memory_response_size";

    @Before
    public void init() {

        OpenAiApi openAiApi = OpenAiApi.builder()
                .baseUrl("https://apis.itedus.cn")
                .apiKey("sk-lIqVNiHon00O6veJ15Cc57DaF5Dd401f93B3A107B4B3677e")
                .completionsPath("v1/chat/completions")
                .embeddingsPath("v1/embeddings")
                .build();


        chatModel = OpenAiChatModel.builder()
                .openAiApi(openAiApi)
                .defaultOptions(OpenAiChatOptions.builder()
                        .model("gpt-4.1-mini")
                        .toolCallbacks(new SyncMcpToolCallbackProvider(stdioMcpClient(), sseMcpClient01(),
                        .build())
                .build();
```

```
        chatClient = ChatClient.builder(chatModel)
                .defaultSystem("""
                        你是一个 AI Agent 智能体，可以根据用户输入信息生成文章，并发送到 CSDN 平台以及完成微信公

                        你擅长使用Planning模式，帮助用户生成质量更高的文章。

                        你的规划应该包括以下几个方面：
                        1. 分析用户输入的内容，生成技术文章。
                        2. 提取，文章标题（需要含带技术点）、文章内容、文章标签（多个用英文逗号隔开）、文章简述（
                        3. 获取发送到 CSDN 文章的 URL 地址。
                        4. 微信公众号消息通知，平台：CSDN、主题：为文章标题、描述：为文章简述、跳转地址：从发布文
                        """)
                .defaultToolCallbacks(new SyncMcpToolCallbackProvider(stdioMcpClient(), sseMcpClient01(), s
                .defaultAdvisors(
                        PromptChatMemoryAdvisor.builder(
                                MessageWindowChatMemory.builder()
                                        .maxMessages(100)
                                        .build()
                        ).build(),
                        new RagAnswerAdvisor(vectorStore, SearchRequest.builder()
                                .topK(5)
                                .filterExpression("knowledge == '知识库名称-v4'")
                                .build()),
                        SimpleLoggerAdvisor.builder().build())
                .build();
    }

    @Test
    public void test_chat_model_stream_01() throws InterruptedException {
        CountDownLatch countDownLatch = new CountDownLatch(1);

        Prompt prompt = Prompt.builder()
                .messages(new UserMessage(
                        """
                                有哪些工具可以使用
                                """))
                .build();

        // 非流式，chatModel.call(prompt)

        Flux<ChatResponse> stream = chatModel.stream(prompt);

        stream.subscribe(
                chatResponse -> {
                    AssistantMessage output = chatResponse.getResult().getOutput();
                    log.info("测试结果: {}", JSON.toJSONString(output));
                },
                Throwable::printStackTrace,
                () -> {
                    countDownLatch.countDown();
                    System.out.println("Stream completed");
                }
        );

        countDownLatch.await();
    }

    @Test
    public void test_chat_model_call() {
        Prompt prompt = Prompt.builder()
                .messages(new UserMessage(
                        """
                                有哪些工具可以使用
                                """))
```

```
            .build();

    ChatResponse chatResponse = chatModel.call(prompt);

    log.info("测试结果(call):{}", JSON.toJSONString(chatResponse));
}

@Test
public void test_02() {
    String userInput = "王大瓜今年几岁";
    System.out.println("\n>>> QUESTION: " + userInput);
    System.out.println("\n>>> ASSISTANT: " + chatClient
            .prompt(userInput)
            .system(s -> s.param("current_date", LocalDate.now().toString()))
            .call().content());
}

@Test
public void test_client03() {
    ChatClient chatClient01 = ChatClient.builder(chatModel)
            .defaultSystem("""
                    你是一个专业的AI提示词优化专家。请帮我优化以下prompt，并按照以下格式返回：

                    # Role: [角色名称]

                    ## Profile
                    - language: [语言]
                    - description: [详细的角色描述]
                    - background: [角色背景]
                    - personality: [性格特征]
                    - expertise: [专业领域]
                    - target_audience: [目标用户群]

                    ## Skills

                    1. [核心技能类别]
                       - [具体技能]: [简要说明]
                       - [具体技能]: [简要说明]
                       - [具体技能]: [简要说明]
                       - [具体技能]: [简要说明]

                    2. [辅助技能类别]
                       - [具体技能]: [简要说明]
                       - [具体技能]: [简要说明]
                       - [具体技能]: [简要说明]
                       - [具体技能]: [简要说明]

                    ## Rules

                    1. [基本原则]:
                       - [具体规则]: [详细说明]
                       - [具体规则]: [详细说明]
                       - [具体规则]: [详细说明]
                       - [具体规则]: [详细说明]

                    2. [行为准则]:
                       - [具体规则]: [详细说明]
                       - [具体规则]: [详细说明]
                       - [具体规则]: [详细说明]
                       - [具体规则]: [详细说明]

                    3. [限制条件]:
                       - [具体限制]: [详细说明]
                       - [具体限制]: [详细说明]
```

```
                - [具体限制]: [详细说明]
                - [具体限制]: [详细说明]

            ## Workflows

            - 目标: [明确目标]
            - 步骤 1: [详细说明]
            - 步骤 2: [详细说明]
            - 步骤 3: [详细说明]
            - 预期结果: [说明]


            ## Initialization
            作为[角色名称]，你必须遵守上述Rules，按照Workflows执行任务。

            请基于以上模板，优化并扩展以下prompt，确保内容专业、完整且结构清晰，注意不要携带任何引导词或
            """)
        .defaultAdvisors(
            PromptChatMemoryAdvisor.builder(
                MessageWindowChatMemory.builder()
                    .maxMessages(100)
                    .build()
            ).build(),
            new RagAnswerAdvisor(vectorStore, SearchRequest.builder()
                .topK(5)
                .filterExpression("knowledge == 'article-prompt-words'")
                .build())
        )
        .defaultOptions(OpenAiChatOptions.builder()
            .model("gpt-4.1")
            .build())
        .build();

    String content = chatClient01
        .prompt("生成一篇文章")

        .system(s -> s.param("current_date", LocalDate.now().toString()))
        .advisors(a -> a
            .param(CHAT_MEMORY_CONVERSATION_ID_KEY, "chatId-101")
            .param(CHAT_MEMORY_RETRIEVE_SIZE_KEY, 100))
        .call().content();

    System.out.println("\n>>> ASSISTANT: " + content);

    ChatClient chatClient02 = ChatClient.builder(chatModel)
        .defaultSystem("""
            你是一个 AI Agent 智能体，可以根据用户输入信息生成文章，并发送到 CSDN 平台以及完成微信公

            你擅长使用Planning模式，帮助用户生成质量更高的文章。

            你的规划应该包括以下几个方面：
            1. 分析用户输入的内容，生成技术文章。
            2. 提取，文章标题（需要含带技术点）、文章内容、文章标签（多个用英文逗号隔开）、文章简述（
            3. 获取发送到 CSDN 文章的 URL 地址。
            4. 微信公众号消息通知，平台：CSDN、主题：为文章标题、描述：为文章简述、跳转地址：为发布文
            """)
//          .defaultTools(new SyncMcpToolCallbackProvider(sseMcpClient01(), sseMcpClient02()))
        .defaultAdvisors(
            PromptChatMemoryAdvisor.builder(
                MessageWindowChatMemory.builder()
                    .maxMessages(100)
                    .build()
            ).build(),
            new SimpleLoggerAdvisor()
```

```
                )
                .defaultOptions(OpenAiChatOptions.builder()
                        .model("gpt-4.1")
                        .build())
                .build();

        String userInput = "生成一篇文章，要求如下 \r\n" + content;
        System.out.println("\n>>> QUESTION: " + userInput);
        System.out.println("\n>>> ASSISTANT: " + chatClient02
                .prompt(userInput)
                .system(s -> s.param("current_date", LocalDate.now().toString()))
                .advisors(a -> a
                        .param(CHAT_MEMORY_CONVERSATION_ID_KEY, "chatId-101")
                        .param(CHAT_MEMORY_RETRIEVE_SIZE_KEY, 100))
                .call().content());
    }

    public McpSyncClient stdioMcpClient() {

        // based on
        // https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem
        var stdioParams = ServerParameters.builder("npx")
                .args("-y", "@modelcontextprotocol/server-filesystem", "/Users/fuzhengwei/Desktop", "/Users/
                .build();

        var mcpClient = McpClient.sync(new StdioClientTransport(stdioParams))
                .requestTimeout(Duration.ofSeconds(10)).build();

        var init = mcpClient.initialize();

        System.out.println("Stdio MCP Initialized: " + init);

        return mcpClient;

    }

    public McpSyncClient sseMcpClient01() {

        HttpClientSseClientTransport sseClientTransport = HttpClientSseClientTransport.builder("http://192.1

        McpSyncClient mcpSyncClient = McpClient.sync(sseClientTransport).requestTimeout(Duration.ofMinutes(1

        var init = mcpSyncClient.initialize();
        System.out.println("SSE MCP Initialized: " + init);

        return mcpSyncClient;
    }

    public McpSyncClient sseMcpClient02() {

        HttpClientSseClientTransport sseClientTransport = HttpClientSseClientTransport.builder("http://192.1

        McpSyncClient mcpSyncClient = McpClient.sync(sseClientTransport).requestTimeout(Duration.ofMinutes(1

        var init = mcpSyncClient.initialize();
        System.out.println("SSE MCP Initialized: " + init);

        return mcpSyncClient;
    }

}
```

AiAgentTest.init 初始化方法，分别创建 openAiApi、chatModel、chatClient，以及在这个过程中初始化 mcp、顾问角色（访问知识库和记忆上下文）。这里的 MCP 服务，就是上面部署好的 csdn、weixin 两个 sse 的 mcp 服务。

注意；stdioMcpClient，路径要修改为你的 `/Users/fuzhengwei/Desktop` 另外还需要本机安装 nodejs https://nodejs.org/zh-cn

之后要分别验证；test_chat_model_stream_01、test_chat_model_call、test_02、test_client03。这几个我会在本节的视频课程中细致说明。

以上验证完成后，就可以为后续的库表做设计做指导了，以及功能实现做细化。