

# SpringAI

xbZhong

2025-09-24

[本页PDF](#)

## SpringAI

### 使用步骤

- 引入依赖
  - 引入 SpringAI 父工程依赖进行 SpringAI 其它依赖的统一版本管理

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.ai</groupId>
      <artifactId>spring-ai-bom</artifactId>
      <version>${spring-ai.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

- 引入模型对应的依赖，如ollama

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

- 配置模型

- 以ollama为例，在配置文件指定访问的url和模型名称，还可以加生成参数

```
spring:
  ai:
    ollama:
      base-url: 访问url
      chat:
        model: 模型名称
```

- 配置客户端

- 需要自定义配置类，通过第三方bean的注册方式注册客户端
- Bean对象的类名为 **ChatClient**，构造方法传入的参数为 **ollamaChatModel**，并且使用构建器 **builder** 创建对象

```
@Bean
public ChatClient chatClient(OllamaChatModel model){
    return ChatClient.builder(model)
        .defaultSystem("你是可爱的助手")
        .build();
}
```

- 在业务逻辑区使用**依赖注入**，注入`chatClient`，并进行信息发送

```
@Autowired
private ChatClient chatClient;
String content = chatClient.prompt()
    .user("你是谁")
    .call()
    .content();
```

## 常用方法

- `ChatClient.defaultSystem()`：设置系统提示词
- `ChatClient.defaultAdvisors()`：配置Advisor
- `chatClient.prompt()`：构建提示词
- `chatClient.user()`：构建用户提示词
- `chatClient.call()`：发送阻塞请求
- `chatClient.stream()`：发送流式生成请求，但是返回值需要声明为`Flux<String> content`
- `chatClient.content()`：获取生成的内容
- `chatClient.advisors()`：生成时使用的**环绕增强器**，可以向context增加上下文
  - `.advisors(a -> a.param(CHAT_MEMORY_CONVERSATION_ID_KEY, chatId))`：向context增加一个会话ID参数
    - `CHAT_MEMORY_CONVERSATION_ID_KEY`是由`AbstractChatMemoryAdvisor`抽象类声明的

## 会话日志

`SpringAI`利用**AOP**原理提供了AI会话时的拦截、增强等功能，也就是**Advisor**，在初始化`chatClient`时进行配置

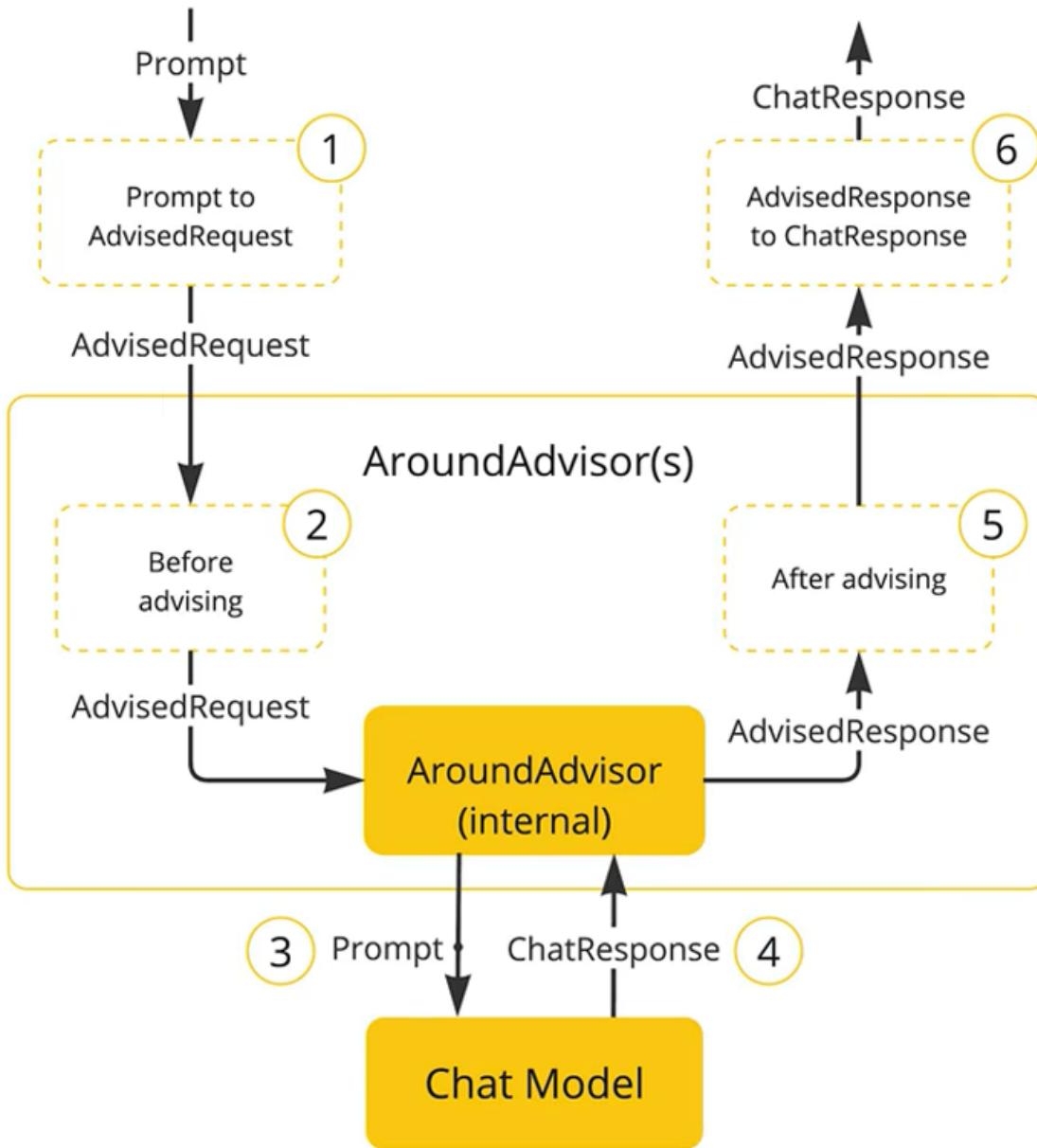


image-20250915194026653

### Advisor接口的实现类

- `SimpleLoggerAdvisor`: 记录日志
- `MessageChatMemoryAdvisor`: 实现会话记忆功能
- `QuestionAnswerAdvisor`: 实现RAG的功能

## 会话记忆

### 使用步骤：

- 定义会话存储方式
  - `SpringAI` 定义的会话记忆接口，我们可以来实现这个接口，自定义存储方式（可以存储在Redis、MongoDB等）

```
public interface ChatMemory{
    void add(String conversationId,List<Message> messages);
    List<Message> get(String conversationId,int lastN);
    void clear(String conversationId);
}
```

- SpringAI 默认定义的接口实现类：`InMemoryChatMemory`（默认存储在内存中）
- 配置会话记忆Advisor
  - 在 `ChatClient.defaultAdvisors()` 中进行配置
- 添加会话id
  - 使用 `chatClient.advisors()` 添加会话id参数

## 会话历史

实现步骤：

- 存储 `chatId`，可以存储在数据库
- 根据 `chatId` 调用 `ChatMemory` 的 `get` 方法获取消息列表进行返回