



ICT Training Center

Il tuo partner per la Formazione e la Trasformazione digitale della tua azienda



SPRING AI

GENERATIVE ARTIFICIAL INTELLIGENCE CON JAVA

Simone Scannapieco

Corso avanzato per Venis S.p.A, Venezia, Italia

Novembre 2025

DEFAULT CHAT MEMORY

- ⚠ LLM sono *stateless*
 - ➔ Non sono concepiti per ricordare conversazioni precedenti
 - ➔ Ogni interazione con LLM avviene senza supporto di una memoria di sistema
 - ⚠ Passaggio da *interazione domanda/risposta* a una *conversazione*???
- ➔ Approccio astratto Spring AI
 - ➔ *ChatMemory* decide la *strategia* di gestione della memoria
 - ➔ Quali messaggi salvare
 - ➔ Quando rimuovere un messaggio
 - ➔ *ChatMemoryRepository* si occupa del CRUD dei messaggi
- ➔ Strategie di gestione memoria (WIP)
 - 1 Salvataggio ultimi *n* messaggi
 - 2 Salvataggio messaggi entro una finestra temporale
 - 3 Salvataggio entro un numero prestabilito di *token*

- ➔ ChatMemory
 - ➔ MessageWindowChatMemory implementa la strategia 1
 - ➔ Finestra di *default* impostata a 20
 - ➔ Auto-configurazione di Spring AI per *bean* ChatMemory
 - ➔ Modificabile invocando `maxMessages()`

Configurazione statica

```
MessageWindowChatMemory memory = MessageWindowChatMemory.builder()  
    .maxMessages(10)  
    .build();
```

- ➔ ChatMemoryRepository
 - ➔ InMemoryChatMemoryRepository recupera e salva messaggi in memoria
 - ➔ Utilizzo di `ConcurrentHashMap` per gestione multi-sessione

- ➔ Come usare tutto questo con ChatClient?!
- ➔ *Chat memory advisor*
 - ➔ Anello di congiunzione fra
 - ➔ Sistema di gestione/CRUD memoria
 - ➔ Sistema di aggiunta di contesto al *prompt*
 - ➔ Implementazioni di `BaseChatMemoryAdvisor`
 - ➔ Recuperano lo storico della conversazione dalla memoria e la includono
 - ➔ nel *prompt* come lista di messaggi (`MessageChatMemoryAdvisor`)
 - ➔ nel *prompt* di sistema in formato testuale (`PromptChatMemoryAdvisor`)

Advisor	Formato storico	Caso utilizzo
<code>MessageChatMemoryAdvisor</code>	Lista di messaggi strutturati	Chatbot più performanti
<code>PromptChatMemoryAdvisor</code>	Testo puro	Limitazioni economiche

Setup minimale

```
/*
 * By automatically configuring a ChatMemory bean, Spring AI instantiates a InMemoryChatRepository and a
 * MessageWindowChatMemory under-the-hood. The only thing to decide is to use either a MessageChatMemoryAdvisor
 * or a PromptChatMemoryAdvisor on top of the ChatMemory bean.
 */

@Bean
public ChatClient chatClient(ChatModel chatModel, ChatMemory chatMemory) {

    ChatClient.Builder chatClientBuilder = ChatClient.builder(chatModel);

    return chatClientBuilder
        .defaultAdvisors(MessageChatMemoryAdvisor
            .builder(chatMemory)
            .build()
        )
        .build();

}

/*
 * Use the ChatClient as usual.
 */
@PostMapping("/client/ask/memory")
public Answer getMemoryAwareAnswer(@RequestBody Question question) {

    return new Answer(this.ChatClient
        .prompt()
        .user(question.question())
        .call()
        .content()
    );

}
```

- ➔ Configurazioni *bean per default chat memory* per ChatClient Ollama
 - 1 Modifica configurazioni di ChatClient per Ollama
 - 2 Modifica interfaccia e implementazione del servizio di risposta
 - 3 Modifica controllore MVC
 - 4 Test delle funzionalità con Postman/Insomnia

Configurazione Gemini + Ollama - I

```
package it.venis.ai.spring.demo.config;

...

@Configuration
public class MemoryChatClientConfig {

    /*
     * Done under-the-hood via Spring auto-configuration.
     */
    @Bean
    public ChatMemoryRepository chatMemoryRepository() {
        return new InMemoryChatMemoryRepository();
    }

    /*
     * Custom bean for chat memory, based on the (auto-)configured chat memory repository.
     * Done under-the-hood via Spring auto-configuration with maxMessages = 20.
     */
    @Bean
    public ChatMemory chatMemory(ChatMemoryRepository chatMemoryRepository) {
        return MessageWindowChatMemory.builder()
            .chatMemoryRepository(chatMemoryRepository)
            .maxMessages(10)
            .build();
    }

    ...
}
```

Configurazione Gemini + Ollama - II

```
...

/*
 * Custom bean for message chat memory advisor, based on the configured chat memory.
 */
@Bean
public BaseChatMemoryAdvisor messageChatMemoryAdvisor(ChatMemory chatMemory) {
    return MessageChatMemoryAdvisor.builder(chatMemory)
        .order(1)
        .build();
}

/*
 * Custom bean for prompt chat memory advisor, based on the configured chat memory.
 */
@Bean
public BaseChatMemoryAdvisor promptChatMemoryAdvisor(ChatMemory chatMemory) {
    return PromptChatMemoryAdvisor.builder(chatMemory)
        .order(1)
        .build();
}

...
```

Configurazione Gemini + Ollama - III

```
...

/*
 * Custom chat client based on the configured chat advisor.
 */
@Bean
public ChatClient ollamaMemoryChatClient(OllamaChatModel ollamaChatModel,
    ChatMemory chatMemory,
    @Qualifier("promptChatMemoryAdvisor") BaseChatMemoryAdvisor chatMemoryAdvisor) {

    ChatClient.Builder chatClientBuilder = ChatClient.builder(ollamaChatModel);

    return chatClientBuilder
        .defaultAdvisors(List.of(
            new SimpleLoggerAdvisor(),
            new OllamaCostSavingsAdvisor(),
            chatMemoryAdvisor))
        .defaultSystem(
            """
            Sei un assistente AI di nome LLamaMemoryBot, addestrato per intrattenere una
            conversazione con un umano.
            """)
        .defaultOptions(ChatOptions.builder()
            .temperature(0.1)
            .build())
        .build();

}
```

Interfaccia servizio

```
package it.venis.ai.spring.demo.services;

import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.Question;

public interface QuestionService {

    Answer getGeminiAnswer(Question question);

    Answer getOllamaAnswer(Question question);

    Answer getOllamaDefaultAnswer(Question question);

    Answer getOllamaMemoryAwareAnswer(Question question);

}
```

Implementazione servizio

```
package it.venis.ai.spring.demo.services;

...

@Service
@Configuration
public class QuestionServiceImpl implements QuestionService {

    private final ChatClient geminiChatClient;
    private final ChatClient ollamaChatClient;
    private final ChatClient ollamaMemoryChatClient;

    public QuestionServiceImpl(@Qualifier("geminiChatClient") ChatClient geminiChatClient,
                              @Qualifier("ollamaChatClient") ChatClient ollamaChatClient,
                              @Qualifier("ollamaMemoryChatClient") ChatClient ollamaMemoryChatClient) {

        this.geminiChatClient = geminiChatClient;
        this.ollamaChatClient = ollamaChatClient;
        this.ollamaMemoryChatClient = ollamaMemoryChatClient;
    }

    ...

    @Override
    public Answer getOllamaMemoryAwareAnswer(Question question) {

        return new Answer(this.ollamaMemoryChatClient.prompt()
            .user(question.question())
            .call()
            .content());
    }
}
```

Implementazione controllore REST

```
package it.venis.ai.spring.demo.controllers;

...

@RestController
public class QuestionController {

    private final QuestionService service;

    public QuestionController(QuestionService service) {

        this.service = service;
    }

    ...

    @PostMapping("/ollama/ask/default")
    public Answer ollamaAskDefaultQuestion(@RequestBody Question question) {

        return this.service.getOllamaDefaultAnswer(question);
    }

    @PostMapping("/ollama/ask/memory")
    public Answer getOllamaMemoryAwareAnswer(@RequestBody Question question) {

        return this.service.getOllamaMemoryAwareAnswer(question);
    }

}
```

<https://github.com/simonescannapieco/spring-ai-advanced-dgroove-venis-code.git>
Branch: 4-spring-ai-gemini-ollama-default-chat-memory