dgroove
Tecnologia emozionale.

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

Chat Memory

Concetti avanzati

- Come tenere traccia di diverse conversazioni?
  - `ChatMemory` definita secondo una logica multi-conversazione
    - ⚠ `CONVERSATION_ID` chiave che il sistema ricerca nel contesto per CRUD storico messaggistica
    - ⚠ `DEFAULT_CONVERSATION_ID` valore di *default* fornito alla chiave se non popolata programmaticamente

Interfaccia Chat Memory

```
public interface ChatMemory {
    String DEFAULT_CONVERSATION_ID = "default";
    /**
     * The key to retrieve the chat memory conversation id from the context.
     */
    String CONVERSATION_ID = "chat_memory_conversation_id";
    ...
```

- Come sovrascrivere il valore di *default*?
  - Agire a livello di *chat memory advisor*
  - Utilizzo delle *advisor specifications* di `ChatClient` (`ChatClient$AdvisorSpec`)

Interfaccia per ogni *chat memory advisor*

```
public interface BaseChatMemoryAdvisor extends BaseAdvisor {
    /**
     * Retrieve the conversation ID from the given context or return the default conversation ID when not found.
     */
    default String getConversationId(Map<String, Object> context, String defaultConversationId) {
        Assert.notNull(context, "context cannot be null");
        Assert.noNullElements(context.keySet().toArray(), "context cannot contain null keys");
        Assert.hasText(defaultConversationId, "defaultConversationId cannot be null or empty");
        return context.containsKey(ChatMemory.CONVERSATION_ID) ? context.get(ChatMemory.CONVERSATION_ID).toString()
                : defaultConversationId;
    }

}
```

Customizzazione *id* conversazione

```
String conversation_id = "custom_conversation_id";

ChatClient
    .prompt()
    .advisors(advisorSpec -> advisorSpec.param(ChatMemory.CONVERSATION_ID, conversation_id))
    .build();
```

- Configurazione *per-user chat memory* per `ChatClient` Ollama
  1. Creazione modello per richiesta domanda con *username*
  2. Modifica interfaccia e implementazione del servizio di risposta
  3. Modifica controllore MVC
  4. *Test* delle funzionalità con Postman/Insomnia

Modello richiesta con *username*

```java
package it.venis.ai.spring.demo.model;

import java.util.Objects;
import java.util.UUID;

public record QuestionRequest(UUID id, String username, Question body) {

    public QuestionRequest(String username, Question body) {

        this(UUID.randomUUID(), username, body);

    }

    @Override
    public String username() {

        return Objects.requireNonNullElse(this.username, "default");

    }

}
```

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;
import it.venis.ai.spring.demo.model.QuestionRequest;

public interface QuestionService {

    Answer getGeminiAnswer(Question question);

    Answer getOllamaAnswer(Question question);

    Answer getOllamaDefaultAnswer(Question question);

    Answer getOllamaMemoryAwareAnswer(Question question);

    Answer getOllamaPerUserMemoryAwareAnswer(QuestionRequest request);

}
```

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 getOllamaPerUserMemoryAwareAnswer(QuestionRequest request) {
        return new Answer(this.ollamaMemoryChatClient
                .prompt()
                .advisors(advisorSpec -> advisorSpec.param(ChatMemory.CONVERSATION_ID, request.username()))
                .user(request.body().question())
                .call()
                .content());
    }
}
```

Implementazione controllore REST

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

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.Question;
import it.venis.ai.spring.demo.model.QuestionRequest;
import it.venis.ai.spring.demo.services.QuestionService;

@RestController
public class QuestionController {

    ...

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

        return this.service.getOllamaMemoryAwareAnswer(question);

    }

    @PostMapping("/ollama/ask/memory/user")
    public Answer getOllamaPerUserMemoryAwareAnswer(@RequestBody QuestionRequest request) {

        return this.service.getOllamaPerUserMemoryAwareAnswer(request);

    }

}
```

https://github.com/simonescannapieco/spring-ai-advanced-dgroove-venis-code.git

Branch: `5-spring-ai-gemini-ollama-per-user-chat-memory`