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

Tool calling

Method as Tool - Interazione con DB

- Servizio MAT programmatico per gestione *ticketing help desk* (H2 DB)
  1. Modifica dipendenze di progetto
  2. Modifica configurazione applicativo
  3. Creazione *string template* di sistema per `ChatClient` con capacità *help desk*
  4. Creazione *entity* per tabella DB
  5. Creazione modello per CRUD a DB per *ticket*
  6. Creazione interfaccia JPA *repository*
  7. Implementazione servizio di CRUD a DB
  8. Configurazione `ChatClient` per *help desk tools*
  9. Modifica controllore MVC
  10. *Test* delle funzionalità con Postman/Insomnia

Dipendenze di progetto

```
...
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <scope>annotationProcessor</scope>
</dependency>
...
```

Configurazione applicativo

```
spring:
    ...
    jpa:
        hibernate:
            ddl-auto: update
```

File `templates/get-help-desk-system-prompt.st`

```
Sei un assistente virtuale di helpdesk, responsabile di assistere i clienti con i loro problemi.
Il tuo obiettivo principale è fornire soluzioni chiare, accurate e utili ai problemi più comuni.
Se il problema di un utente non può essere risolto tramite la tua risposta, offriti di:
- Creare un nuovo ticket helpdesk per il problema segnalato dal cliente
- Verificare lo stato dei ticket helpdesk esistenti
Cerca sempre di essere cortese, proattivo e orientato alle soluzioni.
Aderisci alle seguenti linee guida:
- Se esiste già un ticket relativo allo stesso problema per lo stesso cliente, non creare duplicati
- Se non sei sicuro della natura del problema prima di creare il ticket, chiedi al cliente di spiegare meglio la
problematica o chiedi maggiuori informazioni
- Chiedi sempre conferma esplicita al cliente prima di aprire un nuovo ticket
- Quando rispondi riguardo allo stato dei problemi esistenti, mantieni la risposta breve con lo stato del ticket
e il tempo stimato di risoluzione (ETA).
- Usa i tool forniti dall'utente per fornire la tua risposta
```

Entity tabella in DB

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

import java.time.LocalDateTime;

import jakarta.persistence.*;
import lombok.*;

@Entity
@Getter
@Setter
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "helpdesk_tickets")
public class HelpDeskTicket {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;

    private String issue;

    private String status; // e.g., OPEN, IN_PROGRESS, CLOSED

    private LocalDateTime createdAt;

    private LocalDateTime eta;

}
```

Modello CRUD *ticket* a DB

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

public record TicketRequest(String issue) {

}
```

JPA *repository*

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

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import it.venis.ai.spring.demo.entity.HelpDeskTicket;

public interface HelpDeskTicketRepository extends JpaRepository<HelpDeskTicket, Long> {

    List<HelpDeskTicket> findByUsername(String username);

}
```

Servizio CRUD DB

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

...

@Service
@RequiredArgsConstructor
public class HelpDeskTicketService {

    private final HelpDeskTicketRepository helpDeskTicketRepository;

    public HelpDeskTicket createTicket(TicketRequest ticketInput, String username) {

        HelpDeskTicket ticket = HelpDeskTicket.builder()
                .issue(ticketInput.issue())
                .username(username)
                .status("OPEN")
                .createdAt(LocalDateTime.now())
                .eta(LocalDateTime.now().plusDays(7))
                .build();

        return helpDeskTicketRepository.save(ticket);

    }

    public List<HelpDeskTicket> getTicketsByUsername(String username) {

        return helpDeskTicketRepository.findByUsername(username);

    }

}
```

Help desk tools - I

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

...

@Component
@RequiredArgsConstructor
public class HelpDeskTools {

    private static final Logger LOGGER = LoggerFactory.getLogger(HelpDeskTools.class);

    private final HelpDeskTicketService service;

    @Tool(name = "createTicket",
          description = "Genera un nuovo ticket di assistenza.",
          returnDirect = true)
    String createTicket(
        @ToolParam(description = "Il problema segnalato dall'utente.") TicketRequest ticketRequest,
                                                                       ToolContext toolContext) {

        String username = (String) toolContext.getContext().get("username");
        LOGGER.info("Creazione ticket di assistenza per utente {} con dettagli {}", username, ticketRequest);
        HelpDeskTicket savedTicket = service.createTicket(ticketRequest,username);
        LOGGER.info("Ticket creato con successo. Ticket ID: {}, Username: {}",
                    savedTicket.getId(),
                    savedTicket.getUsername()
                   );
        return "Ticket #" + savedTicket.getId() + " creato con successo per utente " + savedTicket.getUsername();

    }

    ...
```

Help desk tools - II

```
    ...

    @Tool(description = "Recupera lo stato dei ticket di assistenza a partire da uno specifico username.")
    List<HelpDeskTicket> getTicketStatus(ToolContext toolContext) {

        String username = (String) toolContext.getContext().get("username");
        LOGGER.info("Recupero ticket per utente: {}", username);
        List<HelpDeskTicket> tickets =  service.getTicketsByUsername(username);
        LOGGER.info("Trovati {} ticket di assistenza per utente {}", tickets.size(), username);
        return tickets;

    }

}
```

Configurazione `ChatClient` - I

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

...

@Configuration
public class HelpDeskChatClientConfig {

    @Value("classpath:/templates/get-help-desk-system-prompt.st")
    Resource helpDeskSystemPrompt;

    @Bean
    public ChatClient geminiHelpDeskToolsChatClient(
        OpenAiChatModel geminiChatModel,
        HelpDeskTools helpDeskTools,
        ChatMemory chatMemory,
        @Qualifier("messageChatMemoryAdvisor") BaseChatMemoryAdvisor chatMemoryAdvisor) {

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

        return chatClientBuilder
                .defaultTools(helpDeskTools)
                .defaultAdvisors(List.of(new SimpleLoggerAdvisor(), chatMemoryAdvisor))
                .defaultSystem(helpDeskSystemPrompt)
                .build();
    }

    ...
```

Configurazione `ChatClient` - II

```
    ...

    @Bean
    public ChatClient ollamaHelpDeskToolsChatClient(
        OllamaChatModel ollamaChatModel,
        HelpDeskTools helpDeskTools,
        ChatMemory chatMemory,
        @Qualifier("messageChatMemoryAdvisor") BaseChatMemoryAdvisor chatMemoryAdvisor) {

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

        return chatClientBuilder
                .defaultTools(helpDeskTools)
                .defaultAdvisors(List.of(new SimpleLoggerAdvisor(),
                                         new OllamaCostSavingsAdvisor(),
                                         chatMemoryAdvisor))
                .defaultSystem(helpDeskSystemPrompt)
                .build();
    }

}
```

Controllore MVC - I

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

...

@RestController
@Configuration
public class QuestionController {

    ...
    private final ChatClient geminiHelpDeskToolsChatClient;
    private final ChatClient ollamaHelpDeskToolsChatClient;

    public QuestionController(QuestionService service,
            RAGService ragService,
            TimeToolsService timeToolsService,
            @Qualifier("geminiWeatherToolsChatClient") ChatClient geminiWeatherToolsChatClient,
            @Qualifier("ollamaWeatherToolsChatClient") ChatClient ollamaWeatherToolsChatClient,
            @Qualifier("geminiHelpDeskToolsChatClient") ChatClient geminiHelpDeskToolsChatClient,
            @Qualifier("ollamaHelpDeskToolsChatClient") ChatClient ollamaHelpDeskToolsChatClient) {

        this.service = service;
        this.ragService = ragService;
        this.timeToolsService = timeToolsService;
        this.geminiWeatherToolsChatClient = geminiWeatherToolsChatClient;
        this.ollamaWeatherToolsChatClient = ollamaWeatherToolsChatClient;
        this.geminiHelpDeskToolsChatClient = geminiHelpDeskToolsChatClient;
        this.ollamaHelpDeskToolsChatClient = ollamaHelpDeskToolsChatClient;

    }

    ...
```

Controllore MVC - II

```
    ...

    @PostMapping("/gemini/ask/help-desk-tools/help-desk")
    public Answer getGeminiHelpDeskToolAnswer(@RequestHeader("username") String username,
            @RequestParam("message") String message) {
        return new Answer(
            this.geminiHelpDeskToolsChatClient.prompt()
                .advisors(a -> a.param(ChatMemory.CONVERSATION_ID, username))
                .user(message)
                .toolContext(Map.of("username", username))
                .call().content()
            );
    }

    @PostMapping("/ollama/ask/help-desk-tools/help-desk")
    public Answer getOllamaHelpDeskToolAnswer(@RequestHeader("username") String username,
            @RequestParam("message") String message) {
        return new Answer(
            this.ollamaHelpDeskToolsChatClient.prompt()
                .advisors(a -> a.param(ChatMemory.CONVERSATION_ID, username))
                .user(message)
                .toolContext(Map.of("username", username))
                .call().content()
            );
    }

}
```

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

Branch: `14-spring-ai-gemini-ollama-help-desk-tools`