dgroove
Tecnologia emozionale.

ICT Training Center

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

Note

Spring AI

Generative Artificial Intelligence con Java

Simone Scannapieco

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

Novembre 2025

Note

Tool calling

Function as Tool

Note

- Servizio FAT programmatico per richiesta temperatura (*mock*)
  1. Definizione modelli ed enumeratori per `WeatherRequest`, `TemperatureResponse` e `Unit`
  1. Creazione *tool* per richiesta temperatura
  2. Configurazione *client* Ollama e Gemini per utilizzo *tool*
  3. Creazione interfaccia e implementazione servizio
  4. *Test* delle funzionalità con Postman/Insomnia

Note

Enumeratore per unità di temperatura

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

public enum Unit {

    C,
    F;

}
```

Modello per inoltro dati per tempo atmosferico

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

import it.venis.ai.spring.demo.data.Unit;

public record WeatherRequest(String location, Unit unit) {}
```

Modello per risposta da tempo atmosferico

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

import it.venis.ai.spring.demo.data.Unit;

public record TemperatureResponse(double temp, Unit unit) {}
```


Note

Funzione di richiesta temperatura

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

import java.util.function.Function;

import it.venis.ai.spring.demo.data.Unit;
import it.venis.ai.spring.demo.model.WeatherRequest;
import it.venis.ai.spring.demo.model.TemperatureResponse;

public class TemperatureService implements Function<WeatherRequest, TemperatureResponse> {

    public TemperatureResponse apply(WeatherRequest request) {
        return new TemperatureResponse(30.0, Unit.C);
    }

}
```


Note

Configurazione Ollama e Gemini per *tool calling* - I

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

...

@Configuration
public class WeatherToolsConfig {

    public static final String GET_TEMP_IN_LOCATION_FUNCTION_NAME = "getTemperatureInLocation";

    ToolCallback toolCallback = FunctionToolCallback
        .builder(GET_TEMP_IN_LOCATION_FUNCTION_NAME, new TemperatureService())
        .description("Ottieni la temperatura corrente nella località specificata.")
        .inputType(WeatherRequest.class)
        .toolMetadata(ToolMetadata.builder()
            .returnDirect(true)
            .build())
        .build();

    @Bean
    public ChatClient geminiWeatherToolsChatClient(OpenAiChatModel geminiChatModel) {

        ChatClient.Builder chatClientBuilder = ChatClient.builder(geminiChatModel);
        return chatClientBuilder
                .defaultToolCallbacks(toolCallback)
                .defaultAdvisors(List.of(new SimpleLoggerAdvisor()))
                .defaultSystem(
                    """
                        Sei un assistente AI di nome WeatherGeminiBot, addestrato per fornire
                        informazioni meteo alle persone.
                        Utilizza quando possibile il tool 'getTemperatureInLocation'.
                    """)
                .build();
    }
    ...
```


Note

Configurazione Ollama e Gemini per *tool calling* - II

```
    ...

    @Bean
    public ChatClient ollamaWeatherToolsChatClient(OllamaChatModel ollamaChatModel) {

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

        return chatClientBuilder
                .defaultToolCallbacks(toolCallback)
                .defaultAdvisors(List.of(new SimpleLoggerAdvisor(), new OllamaCostSavingsAdvisor()))
                .defaultSystem(
                    """
                        Sei un assistente AI di nome WeatherLlamaBot, addestrato per fornire
                        informazioni meteo alle persone.
                    """)
                .build();
    }

}
```


Note

Implementazione controllore REST - I

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

...

@RestController
@Configuration
public class QuestionController {

    private final QuestionService service;
    private final RAGService ragService;
    private final TimeToolsService timeToolsService;

    private final ChatClient geminiWeatherToolsChatClient;
    private final ChatClient ollamaWeatherToolsChatClient;

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

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

    }

    ...
```


Note

Implementazione controllore REST - II

```
    ...

    @PostMapping("/gemini/ask/weather-tools/temperature")
    public TemperatureResponse getGeminiTemperatureToolAnswer(@RequestBody WeatherRequest request) {

        return this.geminiWeatherToolsChatClient
                .prompt()
                .call()
                .entity(TemperatureResponse.class);
    }

    @PostMapping("/ollama/ask/weather-tools/temperature")
    public TemperatureResponse getOllamaTemperatureToolAnswer(@RequestBody WeatherRequest request) {

        return this.ollamaWeatherToolsChatClient
                .prompt()
                .call()
                .entity(TemperatureResponse.class);
    }

}
```


Note

Codice
Branch di riferimento

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

Branch: 13-spring-ai-gemini-ollama-weather-tools

Note