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 base per Venis S.p.A, Venezia, Italia

Settembre 2025

Spring AI

Introduzione

- Ultimo progetto dell'ecosistema Spring
- Obiettivo: Integrare la GENAI nelle applicazioni Spring Boot
- Astrazione unificata per diversi *provider* AI
  - Anthropic (Claude)
  - OpenAI (GPT)
  - Microsoft (Azure OpenAI)
  - Google (AI Studio, Vertex AI)
- Integrazione nativa con l'ecosistema Spring esistente

- Problema: ogni *provider* AI ha API e modelli diversi
- Soluzione: interfaccia unificata e portabile
- Aderenza ai principi Spring cardine
  - IoC + DI (Inversion of Control and Dependency Injection)
  - AOP (Aspect Oriented Programming)
  - Astrazione
  - Configurazione dichiarativa e *in-code*
  - POJO
  - Modularità
- Integrazione con Spring Boot, Spring Security, Spring Data

Vantaggi di Spring AI

- Portabilità tra provider: cambio di provider AI senza modificare codice business
- Integrazione Spring nativa: sfrutta tutte le funzionalità dell'ecosistema
- Auto-configuration: configurazione automatica basata su dependencies
- Testabilità migliorata: mock e test integration semplificati
- Observability: metriche e monitoring integrati
- Gestione errori: retry policies e fallback automatici

Portabilità tra Provider
Un codice, molti provider

```java
// Configurazione per OpenAI
@Configuration
public class OpenAIConfig {
    @Bean
    public ChatClient chatClient() {
        return ChatClient.builder()
                .chatModel(new OpenAiChatModel(apiKey))
                .build();
    }
}

// Configurazione per Azure OpenAI
@Configuration
public class AzureConfig {
    @Bean
    public ChatClient chatClient() {
        return ChatClient.builder()
                .chatModel(new AzureOpenAiChatModel(endpoint, apiKey))
                .build();
    }
}
```

Auto-configuration
Zero boilerplate code

- application.properties definisce tutto
- @ConditionalOn* annotations per configurazione condizionale

```
# OpenAI Configuration
spring.ai.openai.api-key ${OPENAI_API_KEY}
spring.ai.openai.chat.model gpt-4
spring.ai.openai.chat.temperature 0.7

# Vector Store Configuration
spring.ai.vectorstore.redis.uri redis://localhost:6379
spring.ai.vectorstore.redis.index spring-ai-index
spring.ai.vectorstore.redis.prefix doc:

# Embedding Configuration
spring.ai.openai.embedding.model text-embedding-ada-002
```

```java
@SpringBootTest
class ChatServiceTest {

    @MockBean
    private ChatClient chatClient;

    @Autowired
    private ChatService chatService;

    @Test
    void shouldGenerateResponse() {
        // Given
        when(chatClient.prompt().user("Hello").call().content()
                .thenReturn("Hello! How can I help you?");

        // When
        String response   chatService.chat("Hello");

        // Then
```

Svantaggi e Limitazioni

Svantaggi di Spring AI
Considerazioni e limitazioni

- Progetto giovane: API ancora in evoluzione (versione 1.x)
- Astrazione overhead: layer aggiuntivo rispetto alle API native
- Dipendenza dall'ecosistema Spring: lock-in tecnologico
- Supporto provider limitato: non tutti i provider sono supportati
- Funzionalità avanzate: alcuni provider offrono feature specifiche non astratte
- Learning curve: necessario apprendere nuovi concetti AI oltre Spring

- Breaking changes: possibili modifiche incompatibili tra versioni
- Documentazione incompleta: esempi e best practices in sviluppo
- Community piccola: meno risorse e troubleshooting disponibili

> *Spring AI è attualmente in versione 1.0.1. Sebbene l'API principale sia stabile, alcune funzionalità potrebbero cambiare nelle future versioni minor.*

Considerazione per progetti enterprise

Overhead dell'Astrazione
Il prezzo della portabilità

```java
// Chiamata diretta OpenAI (senza Spring AI)
OpenAiService service = new OpenAiService("sk-...");
CompletionRequest request = CompletionRequest.builder()
    .model("gpt-4")
    .prompt("Explain Spring Boot")
    .temperature(0.7)
    .maxTokens(100)
    .build();
CompletionResult result = service.createCompletion(request);

// VS Spring AI (più semplice ma con overhead)
ChatClient client = ChatClient.builder().build();
String response = client.prompt()
    .user("Explain Spring Boot")
    .call()
    .content();
```

Semplicità vs Performance: trade-off da valutare