

[illegible]

# SPRING AI

## GENERATIVE ARTIFICIAL INTELLIGENCE CON JAVA

Simone Scannapieco

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

Settembre 2025

### Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.



This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

- ➔ Tecnica per aumentare le capacità di ragionamento LLM
- ➔ Strategia: generare **passaggi di ragionamento intermedi**
- ⚠ Utilizzare *few-shot* se la *task* risulta particolarmente complessa
- ➔ Vantaggi
  - ➔ Efficace con minimo sforzo (*prompting* semplice)
  - ➔ LLM più conosciuti hanno capacità di CoT come proprietà emergente
  - ➔ Facile capire *dove* sbaglia (se sbaglia) (approccio *explainable AI*)
  - ➔ Approccio CoT più robusto in caso di cambio di LLM
- ➔ Svantaggi
  - ➔ Costoso e lento (generazione di molti *token* per i passaggi)

## This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page, providing a template for writing or drawing. There are no margins, text, or other markings on the paper.

- ➔ Tecnica utilizzata in combinata ad altre tecniche di *prompting* che coinvolgono il **ragionamento**
- ➔ Capacità di ragionamento LLM basate su un approccio di **non reale comprensione**
- ➔ Prono a risultati finali variabili (**inconsistenza**)
- ➔ Strategia: generare diversi **percorsi di ragionamento**
  - ➔ *Prompt* fornito al LLM più volte
  - ⚠ Aumentare la temperatura del modello per farlo spaziare nel ragionamento
  - ➔ Estrazione del risultato per ogni risposta
    - Meccanismo di *voting* che sceglie la risposta più comune
- ➔ Vantaggi
  - ➔ Mitiga l'inconsistenza intrinseca dei LLM
- ➔ Svantaggi
  - ➔ Costoso e lento (replica il lavoro di altre tecniche di ragionamento)

[illegible]

-  Simone Scannapieco

[illegible]

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

public enum ArtifactGenre {

    STORIA("Storico"),
    FANTASCIENZA("Fantascienza"),
    FANTASY("Fantasy"),
    AVVENTURA("Avventura"),
    ...

    private String artifactGenre;

    ArtifactGenre(String artifactGenre) {
        this.artifactGenre = artifactGenre;
    }

    public String getArtifactGenre() {
        return artifactGenre;
    }

}
```

### Note

[illegible]

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

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

public record ArtifactGenreResponse(ArtifactGenre genre, String reasoning) {}
```

[illegible]



```
DESCRIZIONE: {descrizione}
Classifica la descrizione precedente di {artefatto} in {generi_possibili}.
Ragiona passo per passo e spiega il motivo.
```

### Note

[illegible]

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

import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.ArtifactRequest;
import it.venis.ai.spring.demo.model.DefinitionRequest;
import it.venis.ai.spring.demo.model.DefinitionResponse;
import it.venis.ai.spring.demo.model.Question;

public interface GeminiFromClientService {

    String getAnswerFromClient(String question);

    Answer getAnswerFromClient(Question question);

    Answer getDefinitionFromClient(DefinitionRequest definitionRequest);

    Answer getCustomFormatDefinitionFromClient(DefinitionRequest definitionRequest);

    Answer getJSONUserFormatDefinitionFromClient(DefinitionRequest definitionRequest);

    DefinitionResponse getJSONOutputConverterFormatDefinitionFromClient(DefinitionRequest definitionRequest);

    Answer getSentimentForArtifact(ArtifactRequest artifactRequest);

    Answer getNERinYAMLForArtifact(ArtifactRequest artifactRequest);

    Answer getSuggestionForArtifact(ArtifactRequest artifactRequest);

    Artifact getGeneratedArtifact(ArtifactRequest artifactRequest, Integer numChoices, Integer numParagraphs);

    Answer getGenreForArtifact(ArtifactRequest artifactRequest);

}
```

### Note

[illegible]

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

import java.util.Map;
import org.springframework.ai.chat.client.ChatClient;
import org.springframework.ai.template.st.StTemplateRenderer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.Resource;
import org.springframework.stereotype.Service;
import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.DefinitionRequest;
import it.venis.ai.spring.demo.model.Question;

@Service
public class GeminiFromClientServiceImpl implements GeminiFromClientService {

    private final ChatClient chatClient;

    public GeminiFromClientServiceImpl(ChatClient.Builder chatClientBuilder) {
        this.chatClient = chatClientBuilder.build();
    }

    @Override
    public String getAnswerFromClient(String question) {
        return this.chatClient.prompt()
            .user(question)
            .call()
            .content();
    }
}
```

## Note

[illegible]

```

...
@Value("classpath:templates/get-artifact-genre-prompt.st")
private Resource artifactGenrePrompt;

@Override
public Answer getGenreForArtifact(ArtifactRequest artifactRequest) {
    List<ArtifactGenreResponse> responses = new ArrayList<ArtifactGenreResponse>();
    for (int i = 0; i < 5; i++) {
        List<ArtifactGenreResponse> genreResponse = this.chatClient.prompt()
            .options(ChatOptions.builder()
                .model("gemini-2.0-flash")
                .temperature(2.0)
                //.topP(1.0)
                //.topK(30)
                .maxTokens(1024)
                //.frequencyPenalty(0.1)
                //.presencePenalty(0.1)
                .build())
            .user(u -> u.text(this.artifactGenrePrompt)
                .params(Map.of("descrizione", artifactRequest.artifact().body(),
                    "artefatto", artifactRequest.artifact().type(),
                    "generi_possibili", ArtifactGenre.values())))
            .templateRenderer(StTemplateRenderer.builder().startDelimiterToken('{')
                .endDelimiterToken('}')
                .build())
            .call()
            .entity(new ParameterizedTypeReference<List<ArtifactGenreResponse>>() {});
        responses.addAll(genreResponse);
    }
    return new Answer(responses.stream().collect(Collectors.groupingBy(s -> s.genre(),
        Collectors.counting()))
        .toString());
}
}

```

### Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

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

import org.springframework.web.bind.annotation.RestController;
...
import it.venis.ai.spring.demo.services.GeminiFromClientService;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

@RestController
public class QuestionFromClientController {

    private final GeminiFromClientService geminiService;

    ...

    @PostMapping("/client/generate")
    public Artifact getGeneratedArtifact(@RequestBody ArtifactRequest artifactRequest,
        @RequestParam(required = true, defaultValue = "3") Integer numChoices,
        @RequestParam(required = true, defaultValue = "1") Integer numParagraphs) {

        return this.geminiService.getGeneratedArtifact(artifactRequest, numChoices, numParagraphs);
    }

    @PostMapping("/client/genre")
    public Answer getGeneratedArtifact(@RequestBody ArtifactRequest artifactRequest) {

        return this.geminiService.getGenreForArtifact(artifactRequest);
    }
}
```

### Note

[illegible]

- 13 / 14

[illegible]

14 / 14

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.