+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 06/02/2026

Sign up

Spring AI Training: Developing AI Applications

2 days (14 hours)

Overview

Spring AI is the official framework of the Spring ecosystem, designed to easily consume and integrate generative AI models into enterprise applications.

One of the pillars of Spring AI is its focus on portability: it provides a single, unified API that abstracts away the different providers (OpenAI, Google Gemini, Mistral AI, or local models with Ollama). By leveraging the OpenAI protocol, which has become an industry standard, the framework applies the “1 code, N providers” philosophy.

The Spring AI training covers both the fundamental concepts of generative AI (tokens, context windows, temperature) and the framework’s advanced features such as Tool Calling, fine-grained history management (ChatMemory), and Retrieval-Augmented Generation (RAG).

Upon completing our Spring AI training, you will be able to develop intelligent Java applications connected to LLMs. You will know how to structure dialogues (system, user, and assistant messages) and retrieve responses directly as typed Java objects.

Like all our training courses, this one will introduce you to **the latest stable version** of the technology and its new features.

Objectives

- Understand how an LLM works (tokens, context window, temperature, models) and the vocabulary of the AI ecosystem
- Set up a Spring Boot project integrating Spring AI and interact with an AI model

- Leverage Spring AI’s portability to switch providers (OpenAI, Gemini, Groq, Ollama, Anthropic, Mistral) without rewriting code
- Manage conversation memory (ChatMemory) and multi-user sessions
- Retrieve structured responses (Java/Kotlin objects) rather than plain text
- Write effective prompts (prompt engineering) and choose the right prompt technique
- Give the AI advanced capabilities: internet access (grounding), calling business functions (Tool Calling), image generation
- Design a RAG (Retrieval Augmented Generation) solution to enable the AI to respond based on its own documents

Target Audience

- Java/Kotlin developers, back-end developers, software architects
- Technical leads looking to integrate AI models (LLMs) into their applications

Prerequisites

- Practical knowledge of the Spring/Spring Boot framework (dependency injection, starters, @RestController/@Controller, application.properties)
- Proficiency in Java or Kotlin
- No prior knowledge of Artificial Intelligence or Machine Learning is required
- Workstation equipped with IntelliJ IDEA and internet access

Software requirements

- IntelliJ IDEA
- JDK 17+
- Gradle & Spring Boot
- Free Gemini API key

Spring AI Training Program

[Day 1 - Morning]

Understanding Generative AI and LLMs

- Introduction to AI: LLMs, RAG, Agents
- Anatomy of an LLM: tokens (and token-based billing), the context window, the lack of memory between calls, temperature, the concept of a model, and "knowledge cutoff"

Introduction to Spring AI

- Spring AI: the official framework of the Spring ecosystem for using AI models
- The goal of portability: a single API for multiple providers
- The building blocks of Spring AI: ChatClient, Prompts & Templates, Structured Output, ChatMemory, Embeddings & Vector Stores, RAG (Advisors), Tool Calling, Image/Audio
- The OpenAI protocol as the new standard: "1 code, N providers"

[Day 1 - Afternoon]

The landscape of AI providers

- Overview of providers: OpenAI (ChatGPT), Google Gemini, Groq, Ollama (local), Anthropic (Claude), Mistral AI
- Selection criteria: cost, free tier, speed, privacy, multimodality
- OpenAI-compatible protocol vs. provider-native starter: advantages and limitations
- Spring AI's two OpenAI starters
- Hands-on - HelloWorld: First AI call via Spring AI
  - Creating a Spring Boot project (Web, Thymeleaf, Google GenAI), obtaining a free Gemini key, configuring application.properties, first prompt, and testing the model

Interacting with the model: ChatModel vs. ChatClient

- ChatModel (low-level API) vs ChatClient (user-friendly wrapper)
- Building a call: .prompt() / .call() / .content()

Conversation memory (ChatMemory)

- The LLM is "stateless": demonstration of the lack of memory
- The 3 layers of memory: ChatMemoryRepository (WHERE to store), ChatMemory (WHAT to return), MessageChatMemoryAdvisor (WHEN to apply)
- In-memory storage (InMemory) vs. persistent storage (JDBC)
- Message window strategy (MessageWindowChatMemory) and cost control
- Practical Exercise - History Management
  - Detect lack of memory, connect a MessageChatMemoryAdvisor, manage a conversation per user via the HTTP session ID

Metadata and Internet access

- Leveraging ChatResponse: use of tokens, finishReason, model used, rate limit.
- Provide Internet access to the model: the concept of "grounding".
- Practical Exercise - Internet & Metadata
  - Enable Internet search in the prompt and analyze the response metadata

[Day 2 - Morning]

Structuring the dialogue

- Roles: system / user / assistant messages
- Working with objects: directly retrieving a Java/Kotlin object (Structured Output with .entity(), handling lists, and limits (the LLM may not return valid JSON)

The Art of Prompt Engineering

- The 5 questions of a good prompt: Role, Task, Context, Constraints, Format
- The iterative refinement process
- Types of prompts: zero-shot, one-shot, few-shot, search, creative, expansion, summary, template, meta-prompt, iterative, translation, narrative
- Practical Exercise - Prompt Engineering: "Prompt Golf"
  - Getting an accurate result with the shortest possible prompt
- Hands-on Exercise - Applied Prompt Engineering: Creating a Sales Page
  - Applying prompt techniques to a real-world scenario

[Day 2 - Afternoon]

Tool Calling (Function Calling)

- Principle: Provide the LLM with business functions it can request to call (weather, database, email sending, calculations, etc.)
- The LLM never executes the code: it requests the call (finishReason = TOOL_CALLS), Spring AI makes the call and returns the result
- The @Tool and @ToolParam annotations; the importance of the name and description
- The returnDirect option and the incompatibility between Tool Calling and Internet access depending on the provider
- Practical Exercise - Tool Calling
  - Exposing a business function and letting the AI call it

Image generation

- Image generation with Spring AI
- The Gemini image model ("Nano Banana"): image transformation and generation
- Practical Exercise - Image Generation/Transformation
  - Generate and transform an image from a prompt and a file

RAG: Retrieval-Augmented Generation

- The principle of RAG: getting AI to respond based on YOUR documents
- Ingestion phase (offline): chunking, conversion to vectors (embeddings), storage in a VectorStore
- Query phase: search for relevant chunks based on semantic similarity (top-k, cosine)
- VectorStores: SimpleVectorStore (in-memory), PGVector, Redis, Chroma, Milvus...
- Implementation with Spring AI: RagConfig, IngestionService, QuestionAnswerAdvisor
- Practical Exercise - RAG
  - Feeding an AI with documents (ingestion) and querying it about their content

Further reading

Target Companies

This training is intended for both individuals and companies, large or small, wishing to train their teams in new advanced IT technologies or to acquire specific business knowledge or modern methods.

Assessment upon enrollment

The pre-training assessment complies with Qualiopi quality standards. Upon final registration, the learner receives a self-assessment questionnaire that allows us to evaluate their estimated proficiency with various types of technologies, as well as their expectations and personal goals for the upcoming training, within the limits imposed by the selected format. This questionnaire also allows us to anticipate certain connection or internal security issues within the company (intra-company or virtual classroom) that could pose challenges for monitoring and ensuring the smooth running of the training session.

Teaching Methods

Practical Course: 60% Practical, 40% Theory. Training materials distributed in digital format to all participants.

Organization

The course alternates between theoretical input from the trainer, supported by examples and reflection sessions, and group work.

Assessment

At the end of the session, a multiple-choice questionnaire is used to verify that the skills have been properly acquired.

Certification

A certificate will be issued to each trainee who has completed the entire training program.