Java Meets AI

Empowering Spring Developers to Build Intelligent Apps

Sandra Ahlgrimm
Senior Cloud Advocate, Microsoft

Timo Salm
Principal Solutions Engineer, Broadcom

May 2025

Microsoft & vmware by Broadcom

Who We Are

Sandra Ahlgrimm

Senior Cloud Advocate
Microsoft

X (Twitter): @sKriemhild
Bluesky: @sandraahlgrimm.bsky.social
LinkedIn: https://linkedin.com/in/sandraahlgrimm

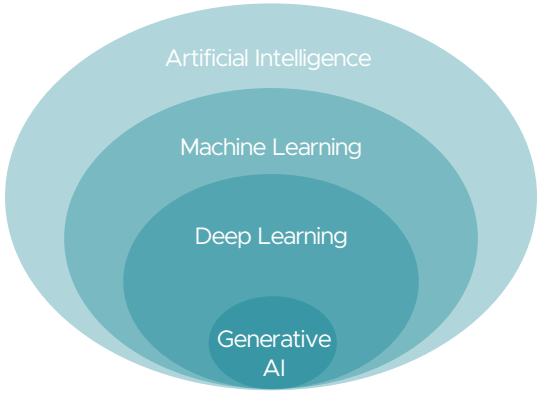
Timo Salm

Principal Solutions Engineer, Broadcom – EMEA
Broadcom

X (Twitter): @salmto
Bluesky: @timosalm.bsky.social
LinkedIn: https://linkedin.com/in/timosalm

Microsoft & vmware by Broadcom

(Generative) AI Fundamentals

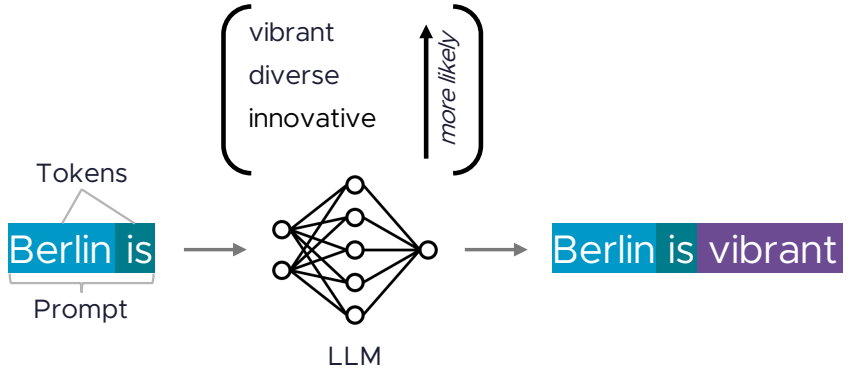


Artificial Intelligence: Machines are capable of performing cognitive functions typically associated with human minds

Generative AI is capable of generating text, images, or other data by utilizing models that learn patterns and structure of their training data

Foundation Model: A mathematical model trained on a huge amount of generic data that serves as the base for various generative tasks

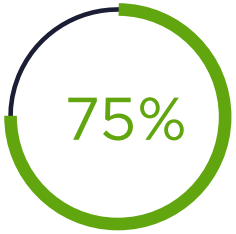
Large Language Models (LLMs): AI models specifically designed to understand and generate human language



spring® by VMware Tanzu

The most popular application development framework on the JVM

of Java developers utilize Spring Boot



like Spring because it's very stable, scalable, and secure

History of Spring

Created in 2003 as a lightweight alternative to address the complexity of the early J2EE specifications. Spring and Java/Jakarta EE are not in competition, they are complementary.

Why Spring?

- Focused on speed, simplicity, and productivity
- Enterprise and production-ready
- Extensive ecosystem
- Large, active developer community

Microsoft & vmware by Broadcom

Sources: https://www.jetbrains.com/lp/devecosystem-2023/java/ https://tanzu.vmware.com/content/ebooks/the-state-of-spring-2022

Why (Java) AI Frameworks?

- Abstractions over working directly with LLM APIs
- Productivity features like structured output parsing, chat memory management, and vector store support
- Support for advanced AI patterns like RAG (Retrieval-Augmented Generation), Tool Calling, and AI Agents
- Stay within familiar programming languages and frameworks, like Java and Spring, rather than switching to, for example, Python and LangChain

JAVA

```java
var request = HttpRequest.newBuilder()
      .uri(URI.create ("https://api.openai.com/v1/chat/completions"))
      .header("Authorization", "Bearer YOUR_API_KEY")
      .header("Content-Type", "application/json")
      .POST(HttpRequest.BodyPublishers.ofString ("""
         {"model":"gpt-4","messages":[
              {"role":"user","content":"What is the capital of Germany?"}
         ]}
         """))
      .build();
var response = HttpClient.newHttpClient()
      .send(request, HttpResponse.BodyHandlers.ofString ());
var content = new ObjectMapper().readTree(response.body())
      .path("choices").get(0).path("message").path("content").asText();
```

Example: Raw OpenAI API Call with Java using HttpClient

Meet the Frameworks

LangChain4j

Initiated in early 2023 due to a lack of Java counterparts to Python libraries like LangChain, LlamaIndex

Framework-agnostic with integrations to Helidon, Quarkus, Spring Boot

1.0 GA since today

spring AI

Official Spring project since end of 2023

Aligns with Spring ecosystem design principles

Supports Spring Boot 3.4

Enterprise support available

1.0 GA May 20th, 2025

Semantic Kernel

Microsoft's cross-platform orchestration engine launched in March 2023

Evolving Java support in addition to C# and Python

Framework-agnostic

Enterprise support available

Demo

Sample Application

Recipe Finder *powered by OpenAi (Gpt-4o) & OpenAi (Dall-e-3)*

Ingredients (comma separated): Cheese ☐ Use available ingredients ☐ Prefer own recipes Find

Cheese And Herb Stuffed Pasta Shells

A delightful dish featuring pasta shells filled with a rich cheese and herb mixture, baked in a savory tomato sauce.

Ingredients

- 250g large pasta shells
- 200g ricotta cheese
- 100g mozzarella cheese, grated
- 50g parmesan cheese, grated
- 2 cloves garlic, minced
- 1 tablespoon fresh basil, chopped
- 1 tablespoon fresh parsley, chopped
- Salt and pepper, to taste
- 500g tomato passata
- 1 tablespoon olive oil
- 1 teaspoon dried oregano

Instructions

Preheat the oven to 180°C (350°F).
Cook the pasta shells in a large pot of boiling salted water until al dente. Drain and set aside to cool.
In a mixing bowl, combine ricotta cheese, half of the mozzarella, parmesan cheese, garlic, basil, parsley, salt, and pepper.
Fill each pasta shell with the cheese mixture and arrange in a baking dish.
In a separate bowl, mix the tomato passata with olive oil, oregano, salt, and pepper.
Pour the tomato sauce over the stuffed shells.
Sprinkle the remaining mozzarella cheese on top.

Microsoft & vmware by Broadcom

LangChain4j

Core APIs

`ChatModel`, `ImageModel`, etc. are low-level APIs to interact with AI models

`AiService` is a higher-level abstraction that hides the complexities of interacting with LLMs and other components

```java
interface Assistant {
    String chat(String userMessage);
}

@Autowired
ChatModel model;

var assistant = AiServices.create(Assistant.class, model);
var answer = assistant.chat("What is the capital of Germany?");
```

Example: Simple AiService

```java
@AiService
interface Assistant {
    @UserMessage("What is the capital of Germany?")
    String fetchCapitalOfGermany();
}

@Autowired
Assistant assistant;

var answer = assistant.fetchCapitalOfGermany();
```

Example: AiService Spring Boot annotations and auto-configuration

LangChain4j

Demo

Core APIs and Features

Microsoft & vmware by Broadcom

Spring AI

Core APIs

`ChatModel`, `ImageModel`, etc. are low-level APIs to interact with AI models

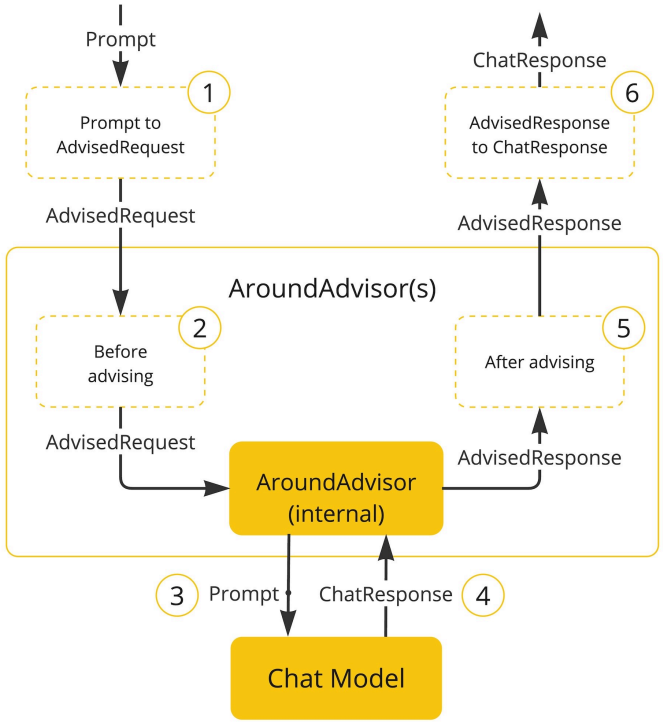
`ChatClient` offers a fluent API for communicating with an AI model

```java
@Autowired
ChatClient chatClient;

var answer = chatClient.prompt()
       .user("What is the capital of Germany?")
       .call()
       .content();
```

Example: Simple ChatClient usage

`Advisors` allow you to intercept, modify, and enhance AI interactions — for example, by appending contextual data to the prompt



spring AI

Demo

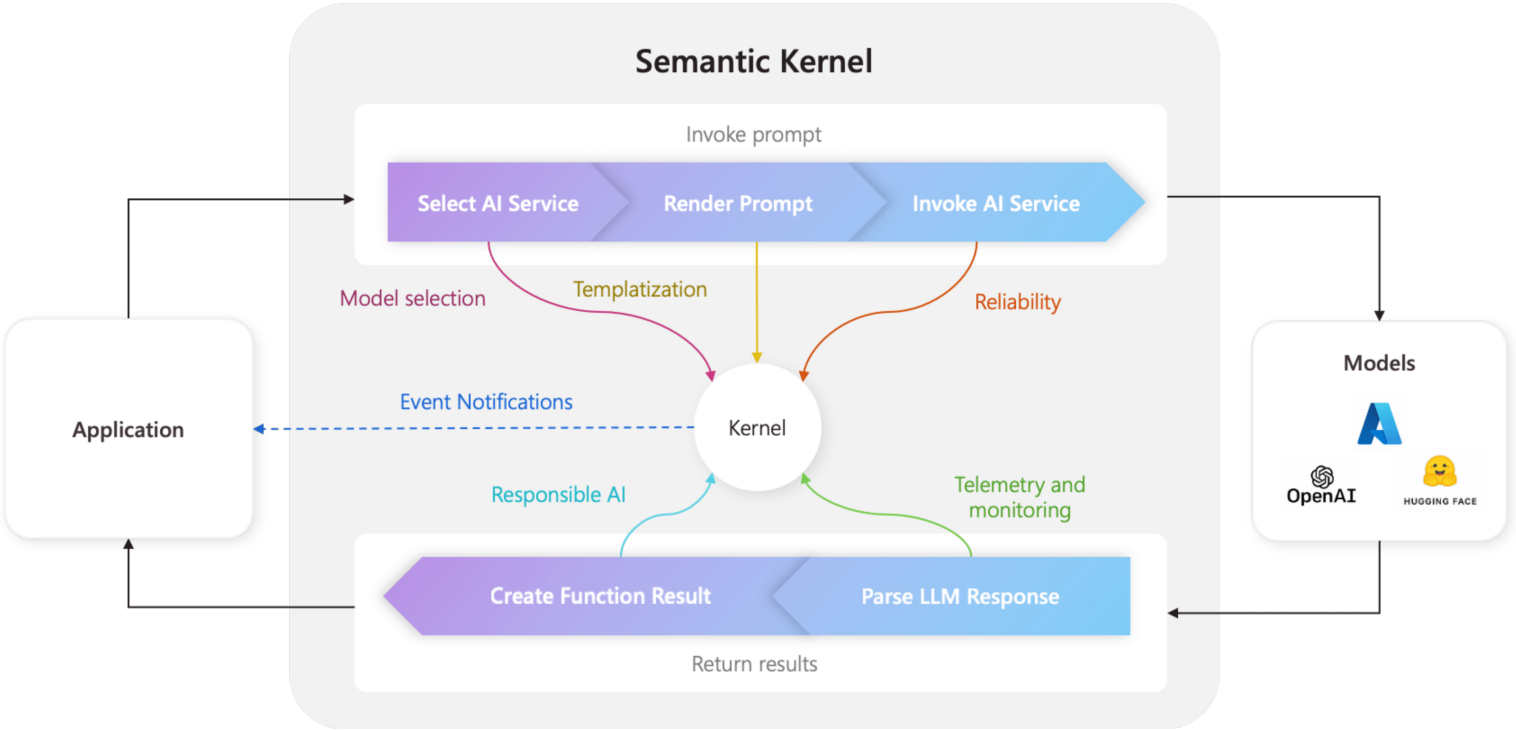
Core APIs and Features

Microsoft & vmware by Broadcom

Semantic Kernel

Core APIs

`Kernel` is central component and DI[1] container managing all AI and other services and plugins[2]



[1] *Dependency Injection* [2] *More details on slide 19*

Semantic Kernel

Demo

Core APIs and Features

Microsoft & vmware by Broadcom

Comparison of Core Aspects

Aspect	LangChain4j	Spring AI	Semantic Kernel
Documentation	Good	Excellent	Incomplete and outdated
Types of models	Wide	Wide	No Image Model
AI provider and model support	Wide	Wide	OpenAI, Azure OpenAI, Hugging Face
Response Streaming	Yes	Yes	No
Spring Boot Autoconfiguration	Good	Excellent	No (sample available)
Structured output	Yes	Yes	No
Prompt templating with variables	Yes	Yes	Yes
Multimodality	Yes	Yes	No
Logging & Observability (AI interactions)	Yes	Yes	Yes
AI model eval testing support	Yes	Yes	No
Enhanced Kotlin support	Yes	Yes	No
Framework Support	Helidon, Quarkus, Spring Boot	Only Spring	Agnostic
Ease of use	Requires familiarity	Spring developer friendly	Medium

Adapting Foundation Models

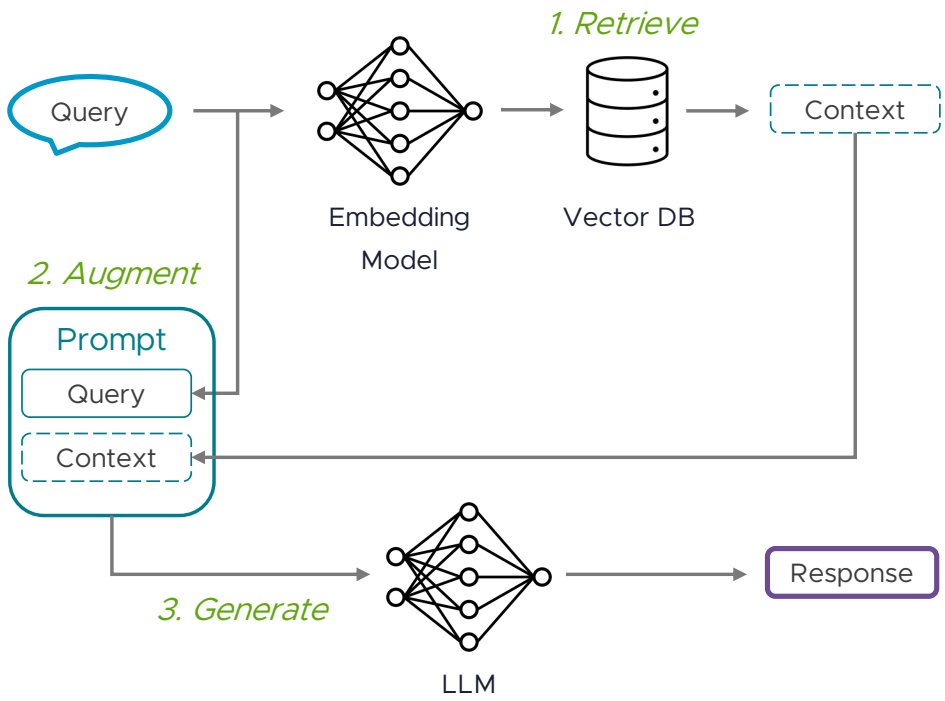
Popular techniques

Fine-Tuning: Further refining a model with specific data to improve its performance on a particular type of task ← *requires significant computational resources*

Prompt Engineering: Designing effective input prompts to guide a generative model's outputs (e.g. with Few-Shot Prompting, Chain-of-Thought Prompting, or In-Context Learning)

Tool Calling: Allows you to register your own functions to connect the model to the APIs of external systems

Retrieval-Augmented Generation (RAG) enhances the output of models by incorporating relevant external information from external data sources



Anatomy of a RAG System

Demo

Sample Application

Tool Calling & RAG

Microsoft & vmware by Broadcom

Recipe Finder *powered by OpenAi (Gpt-4o) & OpenAi (Dall-e-3)*

Ingredients (comma separated): Cheese ☐ Use available ingredients ☐ Prefer own recipes Find

Cheese And Herb Stuffed Pasta Shells

A delightful dish featuring pasta shells filled with a rich cheese and herb mixture, baked in a savory tomato sauce.

Ingredients

- 250g large pasta shells
- 200g ricotta cheese
- 100g mozzarella cheese, grated
- 50g parmesan cheese, grated
- 2 cloves garlic, minced
- 1 tablespoon fresh basil, chopped
- 1 tablespoon fresh parsley, chopped
- Salt and pepper, to taste
- 500g tomato passata
- 1 tablespoon olive oil
- 1 teaspoon dried oregano

Instructions

Preheat the oven to 180°C (350°F).

Cook the pasta shells in a large pot of boiling salted water until al dente. Drain and set aside to cool.

In a mixing bowl, combine ricotta cheese, half of the mozzarella, parmesan cheese, garlic, basil, parsley, salt, and pepper.

Fill each pasta shell with the cheese mixture and arrange in a baking dish.

In a separate bowl, mix the tomato passata with olive oil, oregano, salt, and pepper.

Pour the tomato sauce over the stuffed shells.

Sprinkle the remaining mozzarella cheese on top.

LangChain4j

APIs for Advanced Patterns

Tool Calling

`ToolSpecification` API to provide information about Tools

Annotate Java methods with `@Tool` for automatic conversion to `ToolSpecifications`

```java
@Tool("Fetches the current temperature in a city")
Double fetchCurrentTemperature(@P("Name of the city") String city) {
    ...
}
```

Example: Defining a callable tool with annotations

RAG

`DocumentLoader`s to load documents from various sources

`DocumentParser`s to parse different file formats like PDFs

`DocumentSplitter`s to split documents into smaller units

`EmbeddingModel`s convert text to vectors, which can be stored in an `EmbeddingStore`.

`EmbeddingStoreIngestor` is a pipeline for ingesting `Documents` into an `EmbeddingStore`

`EmbeddingStoreContentRetriever` implements naive RAG

Additional APIs for advanced RAG

LangChain4j

Demo

Tool Calling and Retrieval-Augmented Generation

Microsoft & vmware by Broadcom

Spring AI

APIs for Advanced Patterns

Tool Calling

Tools are modeled via the `ToolCallback` interface

Annotate Java methods with `@Tool` for automatic conversion to a `ToolDefinition`

```java
@Tool(description = "Fetches the current temperature in a city")
Double fetchCurrentTemperature(
    @ToolParam(description = "Name of the city") String city) {
    ...
}
```

Example: Defining a callable tool with annotations

RAG

`DocumentReader`s load and parse documents from various sources

`DocumentTransformer`s transform documents, like splitting them into smaller units

`DocumentWriter`s are preparing documents for storage

`EmbeddingModel`s convert text to vectors, which can be stored in a `VectorStore`.

`QuestionAnswerAdvisor` implements naive RAG

`RetrievalAugmentationAdvisor` and additional APIs to implement advanced RAG

spring AI

Demo

Tool Calling and Retrieval-Augmented Generation

Microsoft & vmware by Broadcom

Semantic Kernel

APIs for Advanced Patterns

Tool Calling

A `KernelPlugin` is a class with functions the `Kernel` can execute

Annotate Java methods with `@DefineKernelFunction` to define them as callable functions in the Kernel

```java
@DefineKernelFunction(name = "fetch_current_temperature",
    description = "Fetches the current temperature in a city",
    returnType = "java.lang.Double")
Double fetchCurrentTemperature(@KernelFunctionParameter(name = "city",
    description = "Name of the city")) {
    ...
}
```

Example: Defining a callable tool with annotations

RAG

Libraries such as `Apache PDFBox` are used to parse and transform documents for storage

`EmbeddingGenerationService`s convert text to vectors, which can be stored in a `VectorStore`

`VectorStoreTextSearch` to retrieve relvant content for a prompt

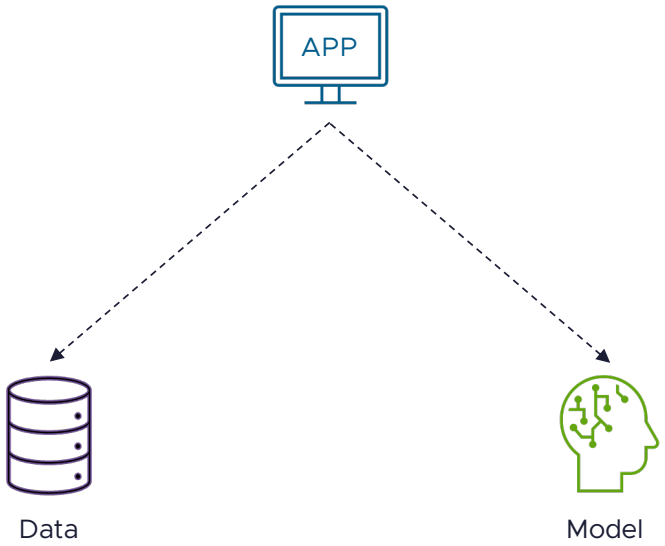
Semantic Kernel

Demo

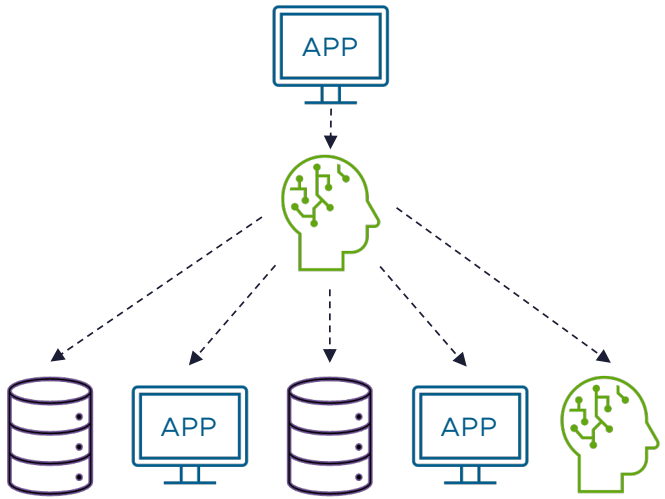
Tool Calling and Retrieval-Augmented Generation

Microsoft & vmware by Broadcom

Agentic AI

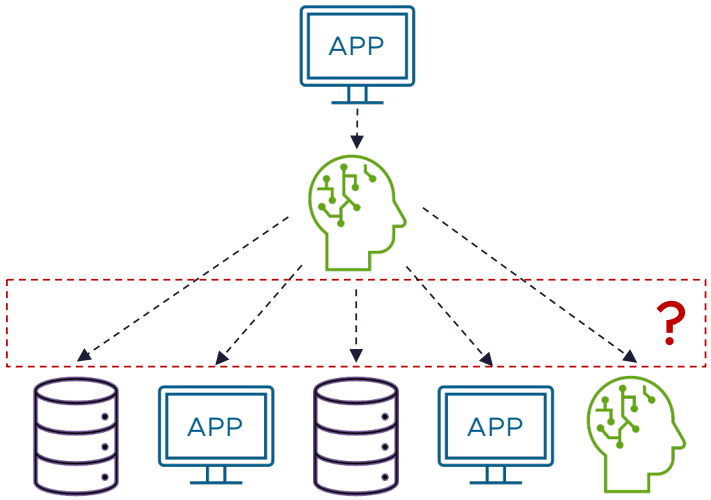


Intelligent Applications



Agentic Applications

Model Context Protocol (MCP)



Agentic Applications

The Model Context Protocol provides a standardized way to connect AI models to different data sources and tools

Advanced Features Comparison

Aspect	LangChain4j	Spring AI	Semantic Kernel
Tool Calling	Yes	Yes	Yes
RAG	Seamless	Seamless	Basic
Vector Database Integration	Yes	Yes	Yes
ETL Pipeline (injest data into Vector DB)	Yes	Yes	Yes
Advanced RAG (with Routing etc.)	Yes	Yes	No
MCP	Only Clients	Yes	Only Clients

Summary

- Enterprise Java is ready for GenAI
- Choose frameworks based on your experience and use case
- Spring AI gets you productive fast, and LangChain4j gives flexibility in framework choice
- Keep an eye on evolving standards like MCP, or frameworks like Semantic Kernel

Follow us
@sKriemhild
@salmto

Thank You

https://github.com/timosalm/ai-recipe-finder

Microsoft & vmware by Broadcom