

# Building AI-Driven Applications

## With Spring AI

Sandra Ahlgrimm  
Senior Cloud Advocate, Microsoft

Timo Salm  
Master Solutions Architect, Broadcom

September 2025

# Who We Are



**Sandra Ahlgrimm**

Senior Cloud Advocate  
Microsoft

X (Twitter): [@sKriemhild](#)

Bluesky: [@sandraahlgrimm.bsky.social](#)

LinkedIn: <https://linkedin.com/in/sandraahlgrimm>



**Timo Salm**

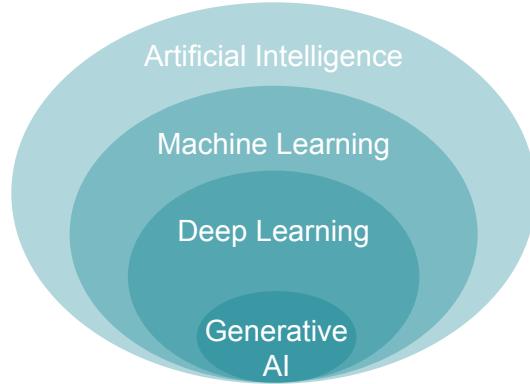
Master Solutions Architect  
Broadcom

X (Twitter): [@salmto](#)

Bluesky: [@timosalm.bsky.social](#)

LinkedIn: <https://linkedin.com/in/timosalm>

# (Generative) AI Fundamentals

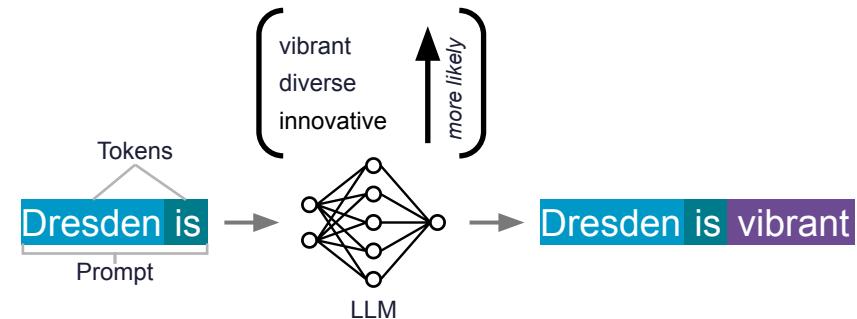


**Artificial Intelligence:** Machines are capable of performing cognitive functions typically associated with human minds

**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



# Why (Java) AI Frameworks?

```
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":"Provide a recipe that includes Cheese"}
        ]}
        ""))
    .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*

# 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

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

# Official Spring Projects

[spring.io/projects](https://spring.io/projects)



Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

3.5.4 + 10 versions



Spring Framework

Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.

6.2.9 + 7 versions



Spring Data

Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.

2025.0.2 + 5 versions



Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.

2025.0.0 + 10 versions



Spring Security

Protects your application with comprehensive and extensible authentication and authorization support.

6.5.2 + 12 versions



Spring for GraphQL

Spring for GraphQL provides support for Spring applications built on GraphQL Java.

1.4.1 + 8 versions



Supports the well-known Enterprise Integration Patterns through lightweight messaging and declarative adapters.

6.5.1 + 11 versions



Spring Modulith

Spring Modulith allows developers to build well-structured Spring Boot applications and guides developers in finding and working with application modules driven by the domain.

1.4.2 + 7 versions



Spring AI

Spring AI is an application framework for AI engineering. At its core, Spring AI addresses the fundamental challenge of AI integration: Connecting your enterprise Data and APIs with the AI Models.

1.0.1 + 1 version

# Spring AI

Provides the key ingredients for creating (Generative) AI applications!

Aligns with Spring ecosystem design principles:

- Component abstractions and default implementations
- Portability across AI Provider APIs, Vector databases and more
- Auto Configuration and Starters with Spring Boot

## Additional features

- Support for several techniques to adapt models to your needs like “Function Calling” or “Retrieval Augmented Generation”
- Multimodality
- Structured outputs (mapping of AI model output to POJOs)
- Observability
- AI model evaluation testing support
- Builds upon other projects in the Spring ecosystem

# Demo

## Recipe Finder Sample & Getting Started with Spring AI

```
    * String representation.
    * - String that represents location, as 2 double values split with comma.
     */
    public Point parseLocation(String locationString) {
        Preconditions.checkNotNull(locationString, errorMessage: "Location String should not be null!");
        Preconditions.checkArgument(locationString.contains(","), errorMessage: "Location must be split by comma");
        locationString = locationString.trim();
        if (locationString.contains(".")) {
            locationString = locationString.replaceAll( regex: " ", replacement: ",");
        }
        if (locationString.contains(",.")) {
            locationString = locationString.replaceAll( regex: ", ", replacement: "," );
        }
        String[] location = locationString.split( regex: "," );
        Preconditions.checkArgument( location.length >= 2, errorMessage: "Location should const"
        double lat = Double.parseDouble(location[0]);
        double lon = Double.parseDouble(location[1]);
        return new Point(lat, lon);
    }
```

# 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

# Demo

## Tool Calling

# Adapting Foundation Models

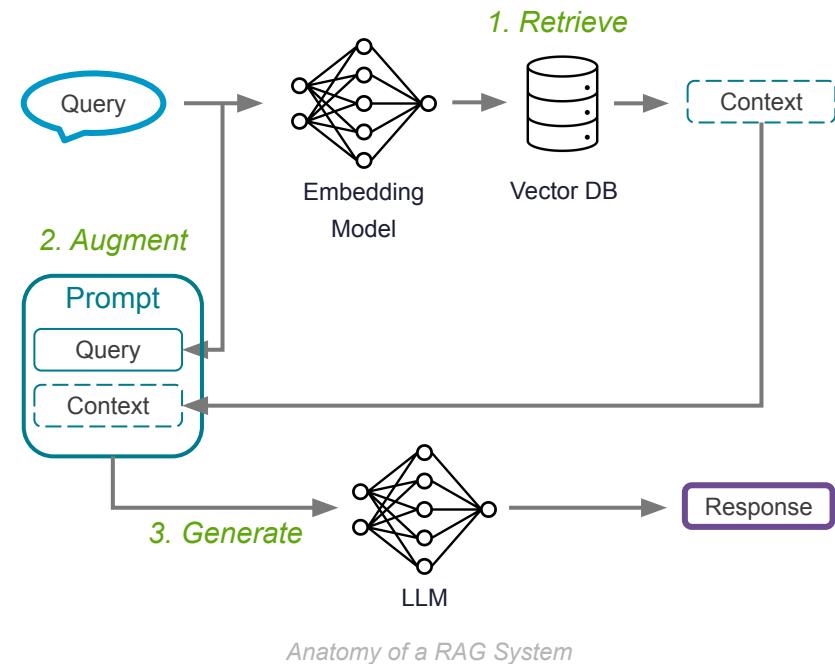
## Popular techniques

**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



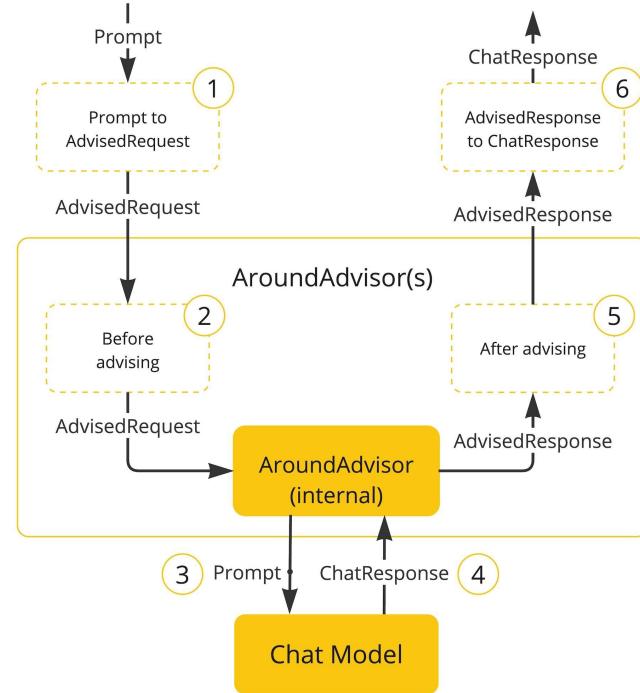
# Retrieval-Augmented Generation - Retrieval Advisors

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

**QuestionAnswerAdvisor** implements naive RAG

**RetrievalAugmentationAdvisor** and additional APIs to implement advanced RAG

More built-in examples are **MessageChatMemoryAdvisor** and **SafeGuardAdvisor**

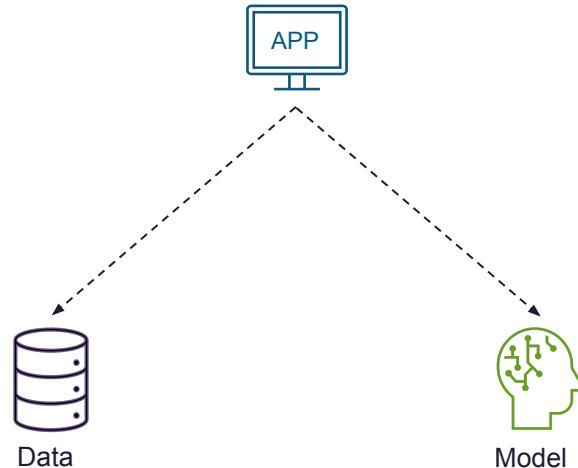


# Demo

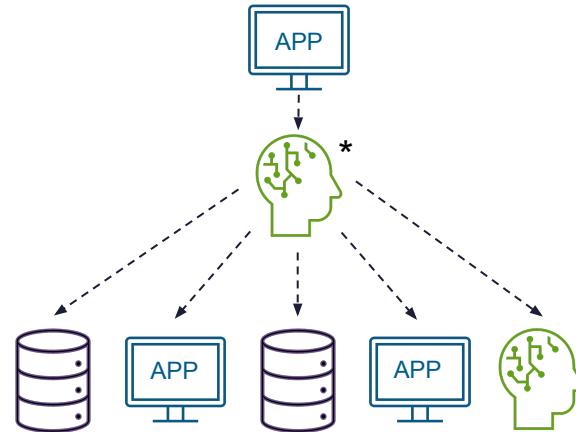
## RAG

# Agentic AI

Independent decision making and task execution, without constant human intervention



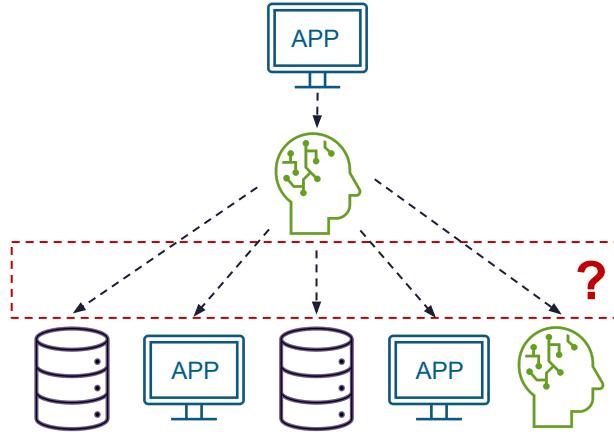
Intelligent Applications



Agentic Applications

\* Visualizes the autonomy and not component interactions

# Model Context Protocol (MCP)

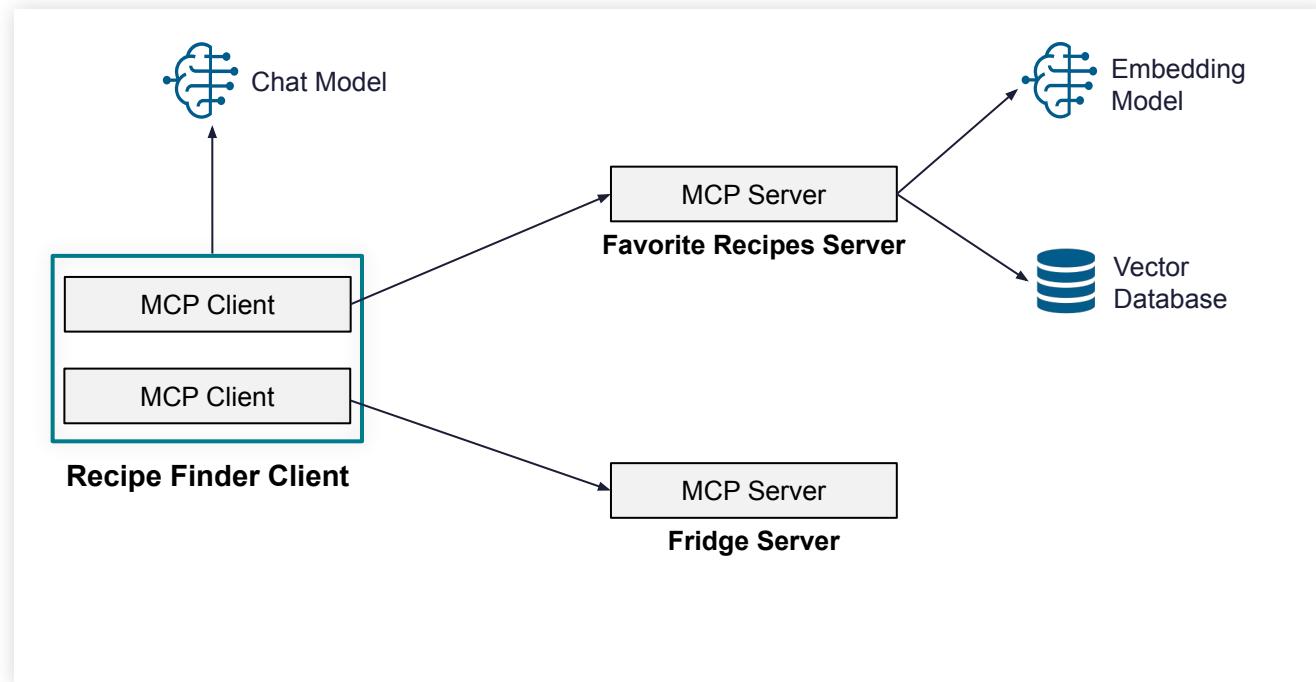


Agentic Applications

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

A **MCP Client** is responsible for establishing and managing connections with **MCP servers**, which provide tools and resources

# Recipe Finder MCP Architecture



# Demo

MCP



Follow us  
@sKriemhild  
@salmto

# Thank You



<https://github.com/timosalm/spring-ai-recipe-finder>