# Spring AI
# &
# MCP

# Przemek Skwiercz

- A programer & cyclist
- One of the founders and organizers of the Bydgoszcz JUG
- I’ve been developing software for over 20 years, so I guess I’m a senior now (at least age-wise 😄)
- In my free time, I’m a fan of bike trips, craft beer, old punk rock, and theatre (not necessarily in that order)

BYDGOSZCZ JUG

# Agenda

# Agenda

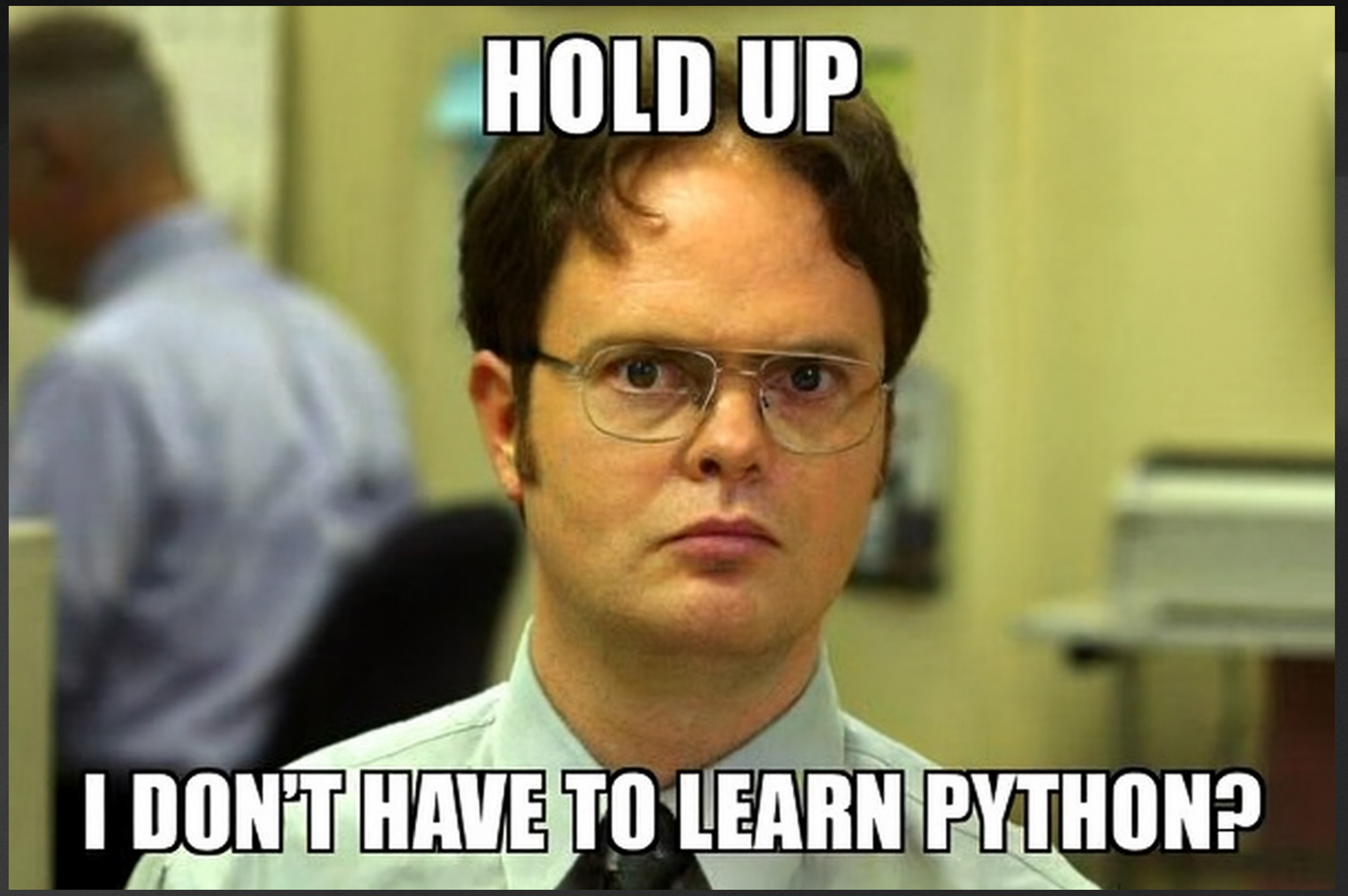
- Why JAVA?
- LLM Limitations
- Spring AI Framework
- MCP - Model Context Protocol

AI Agents in Java?

# Why JAVA

# Python?

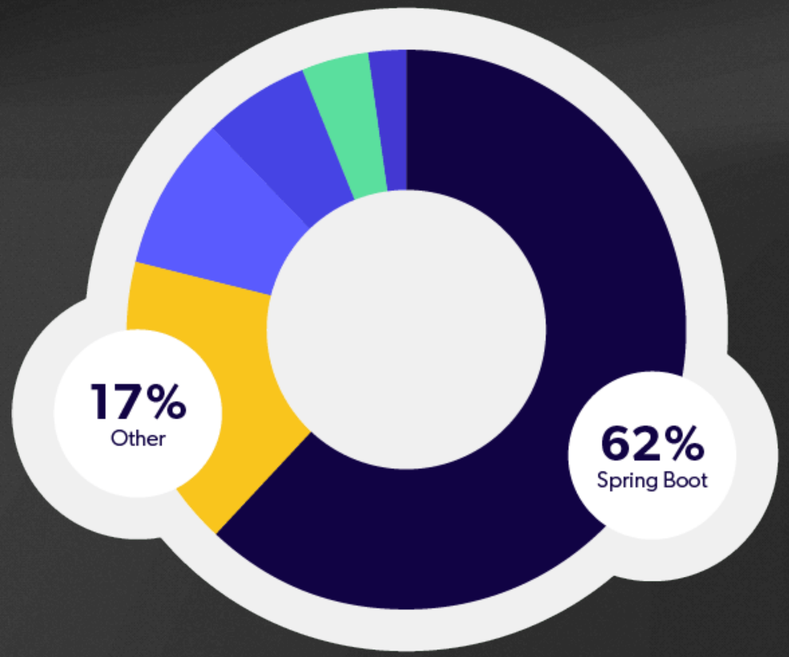
## Nope, not for me

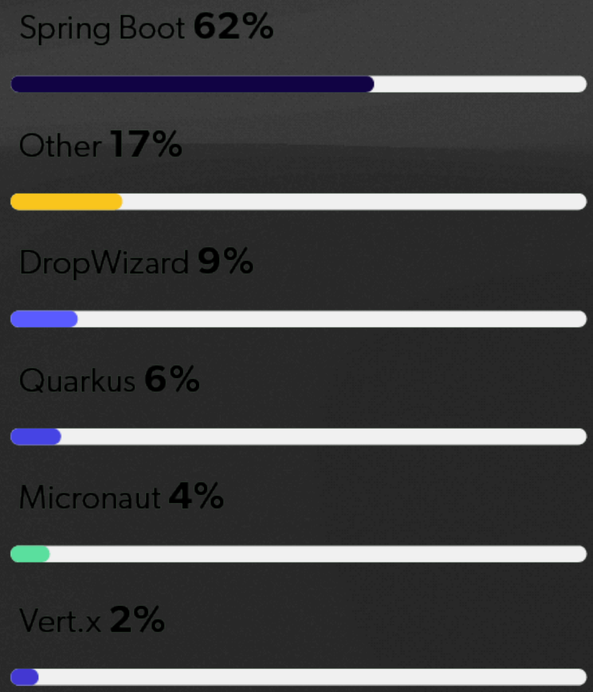



*Source: Chat Gpt*

# Why Spring AI?

- Spring Boot is a highly popular Java framework, particularly for developing microservices and enterprise applications.




*Source: https://www.jrebel.com/resources/java-developer-productivity-report-2021*

# Why JAVA?

## When to choose Java for AI Agents

- **Performance** is critical, and low-latency responses are required
- **Scalability** is a priority, especially in systems with high concurrency demands
- **Integration** with existing Java-based enterprise systems is necessary
- **Maintainability and long-term project support** are important considerations
- **Security**, **Monitoring**

# LLM Limitations

# LLM Limitations

- **Non-Deterministic**
- **Hallucination**
- **Domain Gaps**
- **Stale data** (today's temperature, stock price)
- **Bias and Safety** (toxic language, stereotypes)
- **Privacy Leak** (proprietary/PII text)
- **Cost and Latency**
- **Security**

- **Prompt Stuffing**
- **RAG**
- **Tool/Function Calling**
- **Output Converters**
- **Prompt Guarding**
- **Evaluation**
- **MCP**

# Spring AI

# Spring AI

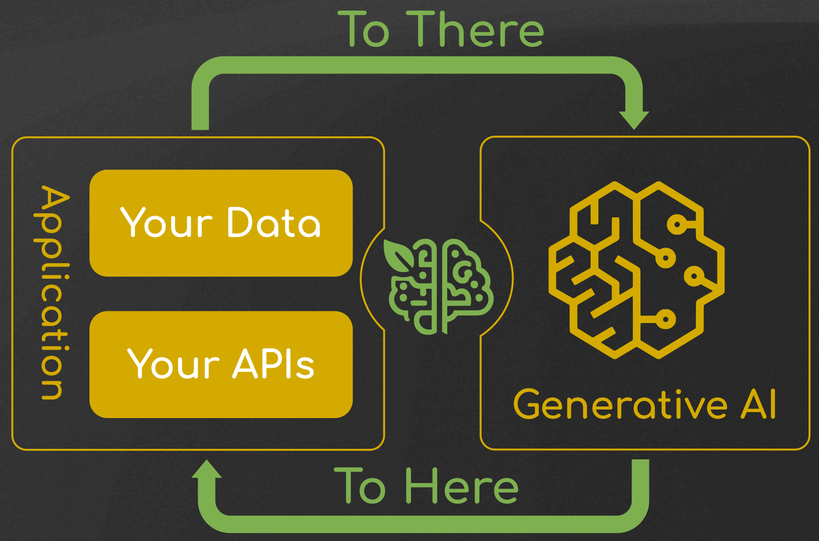
## Alternatives for Spring AI

- **LangChain4J** (Python)
- **Semantic Kernel** (Python and C#)

# Spring AI

## What is it used for?

- **Spring AI** is designed to **integrate AI models with Spring applications** and **create AI Agents**
- It allows us to connect our data and API with AI models




*Source: https://spring.io/projects/spring-ai*

# Spring AI
## History

In **mid-2023**

Spring revealed that it was working on a module that makes it easier to create AI applications using the Spring framework

**01.2024**

Version 0.8.0 was released (first useful release)

**12.2024**

Add Model Context Protocol

**02.2025**

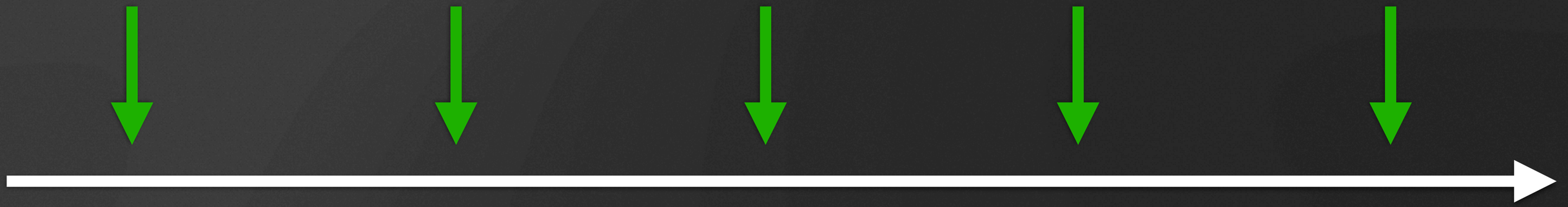
@Tool annotation

Chain Workflow (AI Agents)

**20.05.2025**

**1.0.0 GA**

**Production Ready**



# Spring AI

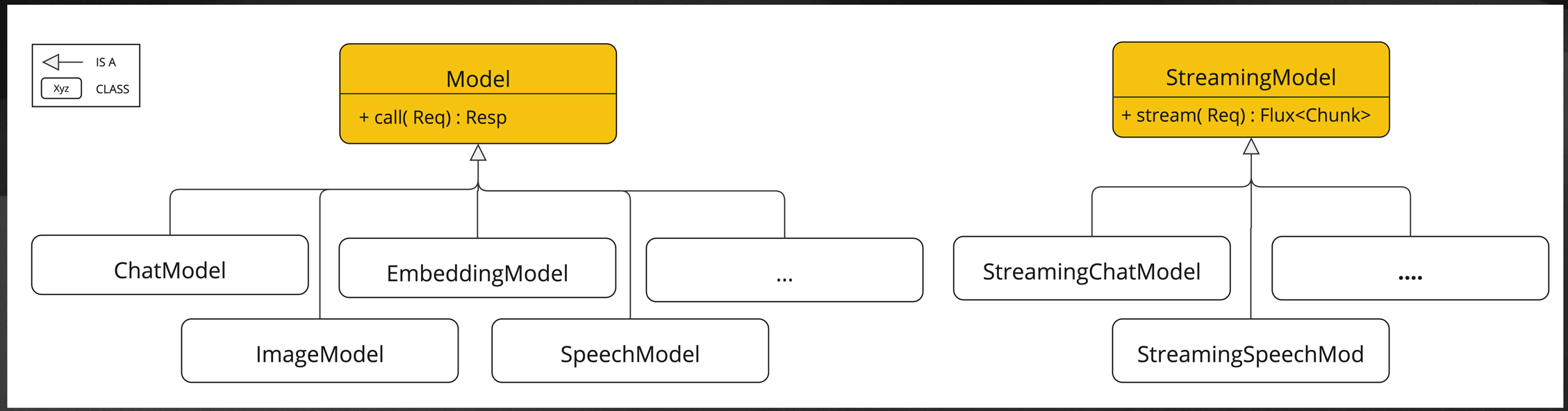
## Spring AI Models

- **Remote / Cloud-Hosted** (*OpenAI, AWS SageMaker, etc.*)
- **Local / On-premisses** (*Ollama*)

# Spring AI
## Supported LLMs

- OpenAI Chat Completion (streaming, multi-modality & function-calling support)
- Microsoft Azure Open AI Chat Completion (streaming & function-calling support)
- Ollama Chat Completion (streaming, multi-modality & function-calling support)
- Hugging Face Chat Completion (no streaming support)
- Google Vertex AI PaLM2 Chat Completion (no streaming support)
- Google Vertex AI Gemini Chat Completion (streaming, multi-modality & function-calling support)
- Amazon Bedrock (Cohere, Llama, Titan, Anthropic, Jurassic2)
- Mistral AI Chat Completion (streaming & function-calling support)
- Anthropic Chat Completion (streaming & function-calling support)
- watsonx.AI and many others…

# Spring AI
## Model Abstraction




*Source: https://spring.io/projects/spring-ai*

# Spring AI
## Simple app

```
spring.application.name=spring-ai-ud
server.port=8080
spring.ai.openai.api-key=${OPEN_AI_KEY}
spring.ai.openai.chat.options.model=gpt-4o
```

```
@Service  no usages  1 inheritor  pskwiercz *
public class OpenAiServiceExample {

    private final ChatClient chatClient;  2 usages

    public OpenAiServiceExample(ChatClient.Builder chatClient) {  new *
        this.chatClient = chatClient.build();
    }

    Rename usages
    public String ask(String question) {  new *

        return chatClient
                .prompt() ChatClientRequest
                .user(question)
                .call() CallResponseSpec
                .content();

    }
}
```

```
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.3.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.pskwiercz</groupId>
    <artifactId>spring-ai-ud</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>spring-ai-ud</name>
    <description>spring-ai-ud</description>
    <properties>
        <java.version>21</java.version>
        <spring-ai.version>1.0.0-M1</spring-ai.version>
    </properties>
    <dependencies> Add Starters...
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.ai</groupId>
            <artifactId>spring-ai-openai-spring-boot-starter</artifactId>
        </dependency>
    </dependencies>
```

# RAG

## Prompts

**Prompts** - are the inputs that guide an AI model to generate specific outputs

**Prompt Types:**

SYSTEM

USER

ASSISTANT

TOOL

# SPRING AI

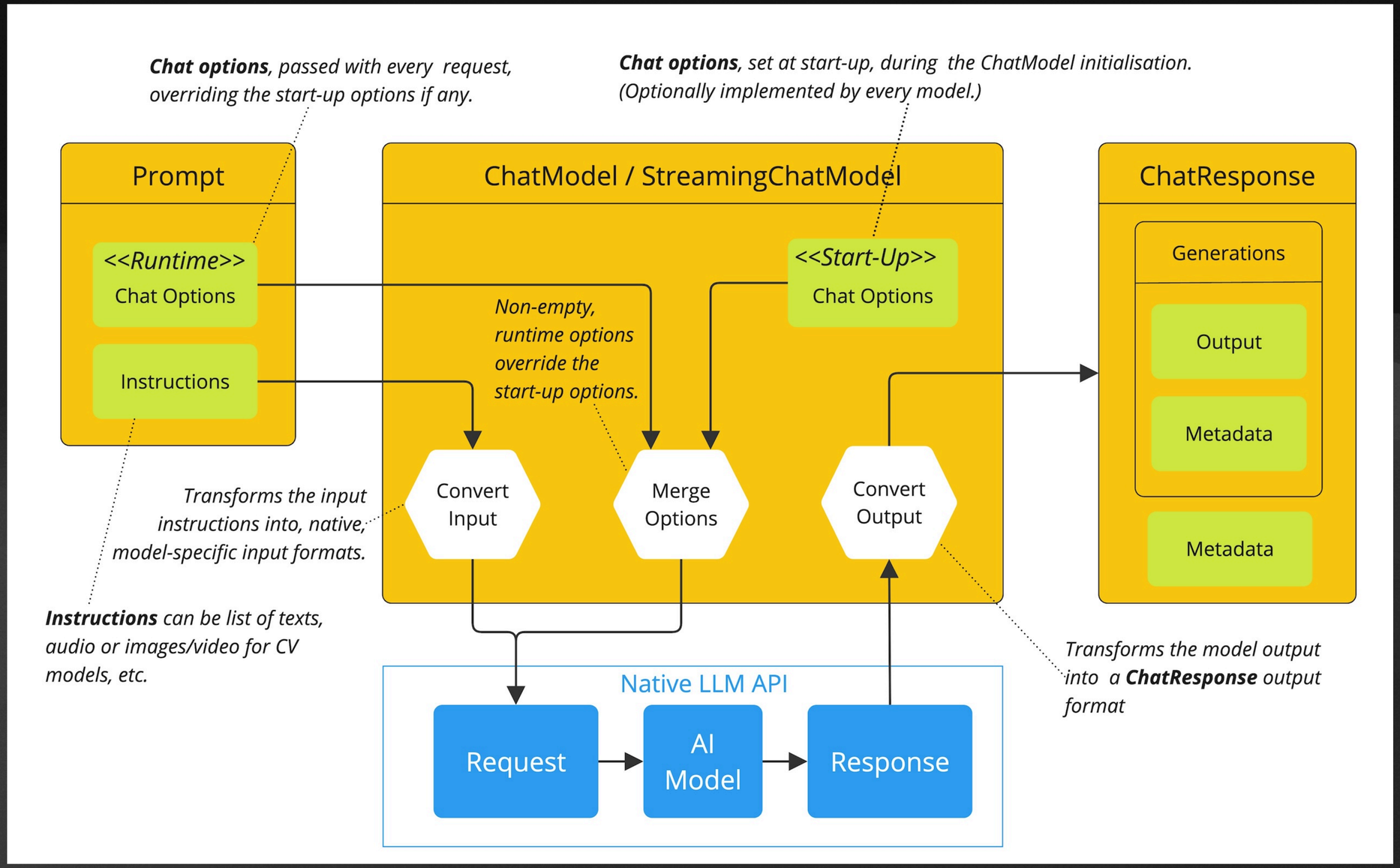
## Prompts

```java
public class Prompt implements ModelRequest<List<Message>> {

    private final List<Message> messages;

    private ChatOptions chatOptions;
}
```

```java
public interface ChatOptions extends ModelOptions {

    String getModel();
    Float getFrequencyPenalty();
    Integer getMaxTokens();
    Float getPresencePenalty();
    List<String> getStopSequences();
    Float getTemperature();
    Integer getTopK();
    Float getTopP();
    ChatOptions copy();

}
```

*Source: https://spring.io/projects/spring-ai*

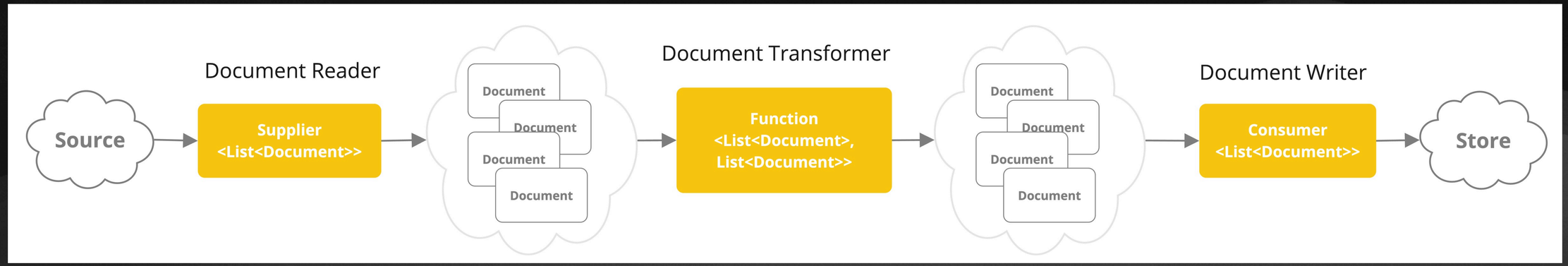
# Spring AI
## Chat Model




Source: https://spring.io/projects/spring-ai

# Spring AI

## ETL - Extract-Transform-Load




- Text
- JSON
- Markdown
- PDF page/chapter
- Apache Tika (doc, html, ppt…)

- TextSplitter
- TokenSplitter
- KeywordMetadataEnricher
- SummaryMetadataEnricher

- File
- Vector DB

*Source: https://spring.io/projects/spring-ai*

# Spring AI

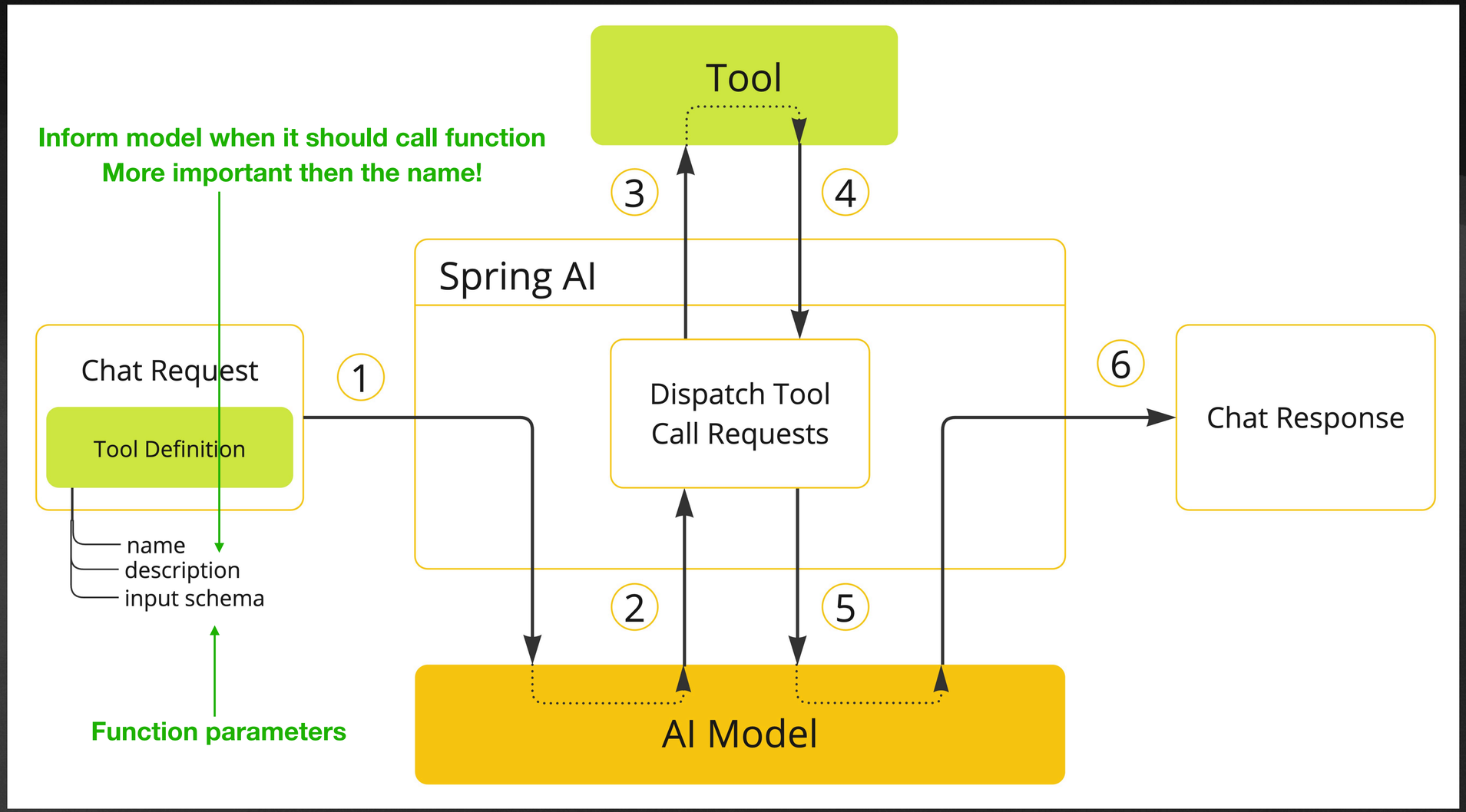
## Structured Output

### Response conversion types

- to List
- to custom Object
- to List of Objects
- To Map of Objects

*Source: https://spring.io/projects/spring-ai*

# Spring AI
## Tool Calling




Source: https://spring.io/projects/spring-ai

# Spring AI

## Tool Calling (~~Function Calling~~)

- **Method Based Tools** - Java method annotated with **@Tool**, use regular Spring Beans and methods
- **Function/Callback Based Tools** - instances of **ToolCalback** that allow register arbitrary functions, suppliers, consumers, or bi-functions as “tools” for the model to invoke

# Spring AI
## Advisors

- **Advisors** - something like *interceptors* or *filters* in SpringBoot

- **MessageChatMemoryAdvisor** - retrieves memory and adds it as a collection of messages to the prompt (*conversation history*)
- **PromptChatMemoryAdvisor** - retrieves memory and incorporates it into the prompt's system text
- **VectorStoreChatMemoryAdvisor** - retrieves memory from a VectorStore and adds it into the prompt's system (*large datasets*)

# Spring AI

## Advisors

- **QuestionAnswerAdvisor** - uses a vector store to provide question-answering capabilities, implementing the RAG pattern
- **RetrievalAugmentationAdvisor** - advisor that implements common Retrieval Augmented Generation (RAG)

- **ReReadingAdvisor** - reasoning advisor (implements a *re-reading strategy* for LLM reasoning, dubbed RE2)

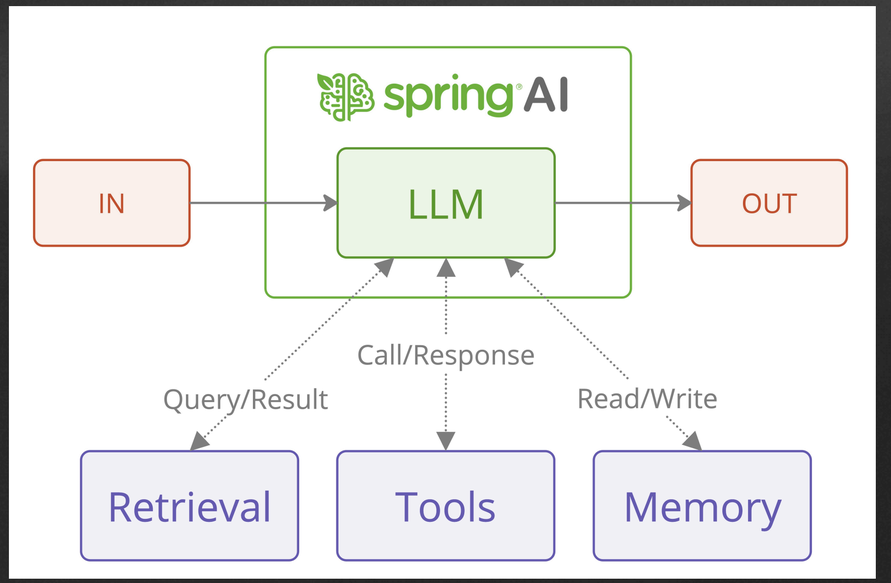
- **SafeGuardAdvisor** - simple advisor designed to prevent the model from generating harmful or inappropriate content

# Spring AI
## Evaluation

- The Spring AI interface for evaluating responses is **Evaluator**. There are two implementations at this moment:
  - **RelevancyEvaluator** - which uses the AI model for evaluation
  - **FactCheckingEvaluator** - helps detect and reduce hallucinations in AI outputs by verifying if a given statement (*claim*) is logically supported by the provided context (*document*).

# Spring AI
## Agent Workflows

- Easy implementation of **LLM Workflow Patterns**
  - Chain Workflow
  - Parallelisation Workflow
  - Routing Workflow
  - Orchestrator-Workers
  - Evaluator-Optimizer




*Source: https://spring.io/blog/2025/01/21/spring-ai-agentic-patterns*

# SPRING AI

## String Templates

river-prompt.st

```
What is the longest river in {country}? Response in JSON with the name of the river in a property named
Do not wrap the json codes in JSON
```

OpenAiService.java

```java

@Service  2 usages  pskwiercz *
public class OpenAiService {

    private final ChatClient chatClient;  2 usages

    @Value("classpath:templates/river-prompt.st")
    private Resource riverPrompt;

sk-ant-api03-oigg9Z8ml9ZbWqO8Qa0_x_NLF7R5TAxOfal_Y9egDetchXki-LGV6cP6F2b76P0mUGUzRMEvsqEqfSmA0mIiA-af3f-wAA
    public OpenAiService(ChatClient.Builder chatClient) { this.chatClient = chatClient.build(); }

    public Answer getRiver(GetRiverRequest getRiverRequest) {  1 usage  new *

        Prompt prompt = new PromptTemplate(riverPrompt)
                .create(Map.of( k1: "country", getRiverRequest.country()));

        String answer = chatClient
                .prompt(prompt) ChatClientPromptRequest
                .call() CallPromptResponseSpec
                .content();

        return new Answer(answer);
    }
}
```

# Spring AI - RAG

## Supported Vector DB

- **SimpleVectorStore** - A simple implementation of persistent vector storage, good for educational purposes.
- **PgVector Store**
- **Azure Vector Search**
- **Apache Cassandra**
- **Elasticsearch Vector Store**
- **GemFire Vector Store**
- **Milvus Vector Store**
- **MongoDB Atlas Vector Store**
- **Neo4j Vector Store**
- **OpenSearch Vector Store**
- **Oracle Vector Store**
- **Pinecone Vector Store**
- **Qdrant Vector Store**
- **Redis Vector Store**
- **SAP Hana Vector Store**

# SPRING AI - RAG

## Vector DB Configuration

- Vector DB are used to integrate customer data with AI models.
- When a user query is to be sent to the AI model, a set of similar documents is first retrieved. These documents then serve as the context for the user's question and are sent to the AI model, along with the user's query.

```java
public interface VectorStore {

    void add(List<Document> documents);

    Optional<Boolean> delete(List<String> idList);

    List<Document> similaritySearch(String query);

    List<Document> similaritySearch(SearchRequest request);
}
```

*Source: https://spring.io/projects/spring-ai*

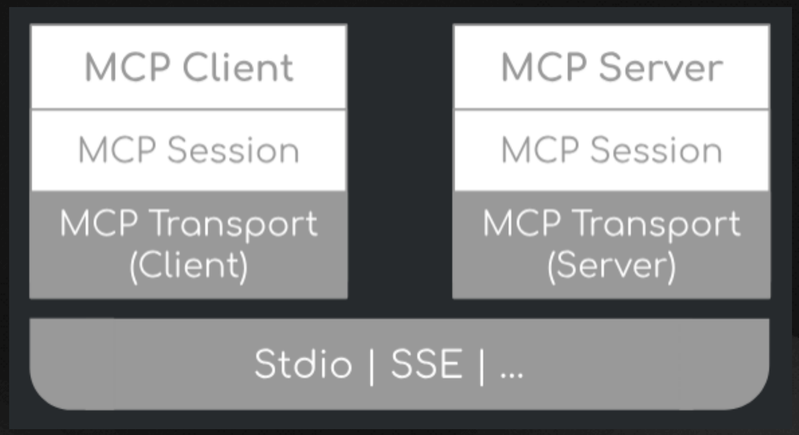
# SPRING AI

## Vector DB configuration

```
spring.application.name=JugSpringAiApp
server.port=9999

# OpenAI props
spring.ai.openai.chat.options.model=gpt-4-turbo
spring.ai.openai.embedding.options.model=text-embedding-ada-002
spring.ai.openai.api-key=${OPENAI_API_KEY}

# PGVector props
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=postgres

spring.ai.vectorstore.pgvector.initialize-schema=true
spring.ai.vectorstore.pgvector.index-type=HNSW
spring.ai.vectorstore.pgvector.distance-type=COSINE_DISTANCE
spring.ai.vectorstore.pgvector.dimensions=1536
spring.ai.vectorstore.pgvector.schema-validation=true
spring.ai.vectorstore.pgvector.remove-existing-vector-store-table=true
spring.ai.vectorstore.pgvector.schema-name=public
spring.ai.vectorstore.pgvector.table-name=vector_store
```

# Model Context Protocol

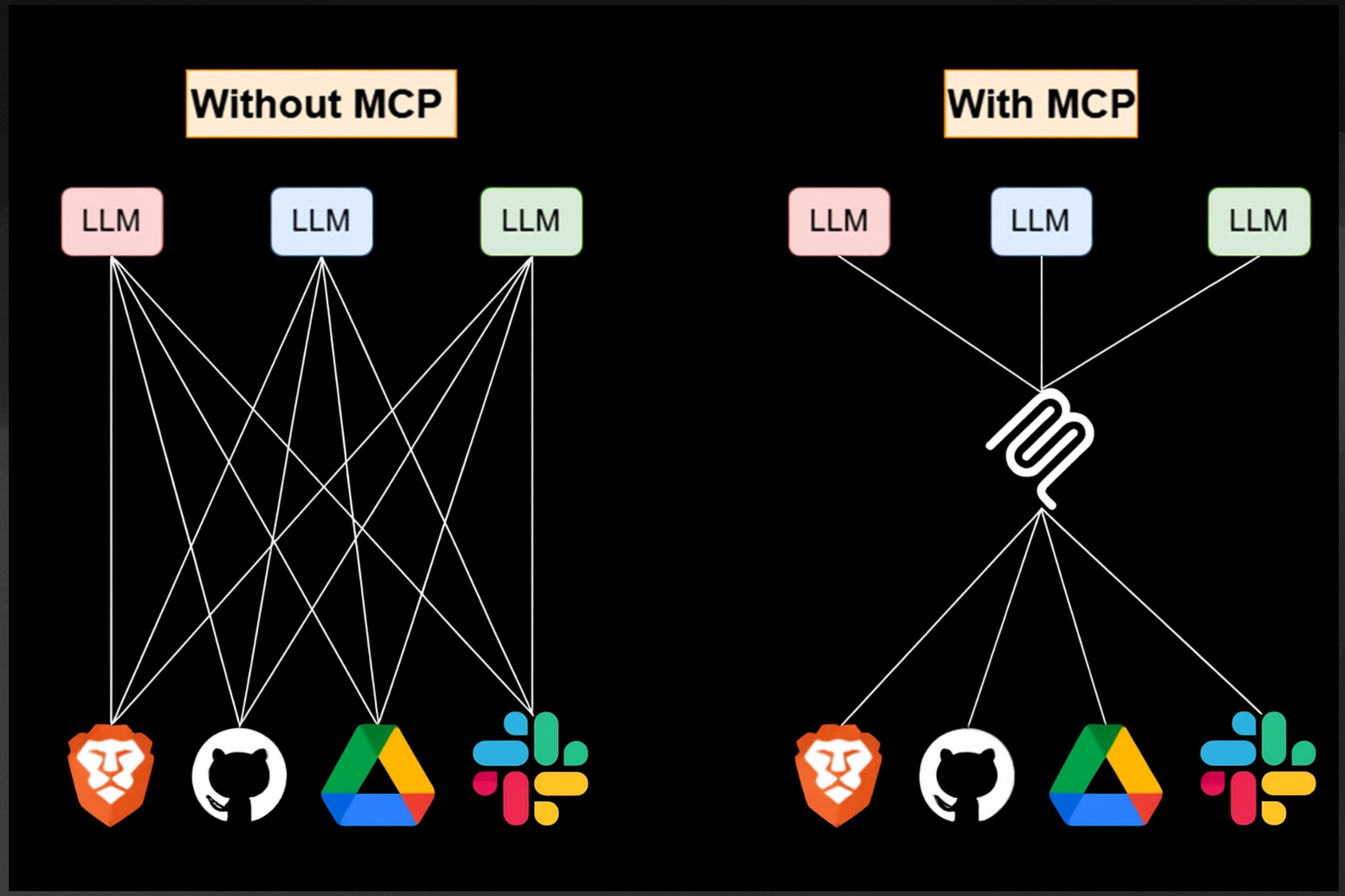
# Spring AI MCP
## Introduction




- The Model Context Protocol (**MCP**) is a standardized protocol that enables AI models to interact with external tools and resources in a structured way
- MCP supports multiple transport mechanisms to provide flexibility across different environments (Spring AI MCP support Standard I/O (stdio) and Server-Sent Events (SSE))
- Spring AI MCP extends the MCP Java SDK with Spring Boot integration, providing both client and server starters
- **MCP is language agnostic!**

# Spring AI MCP

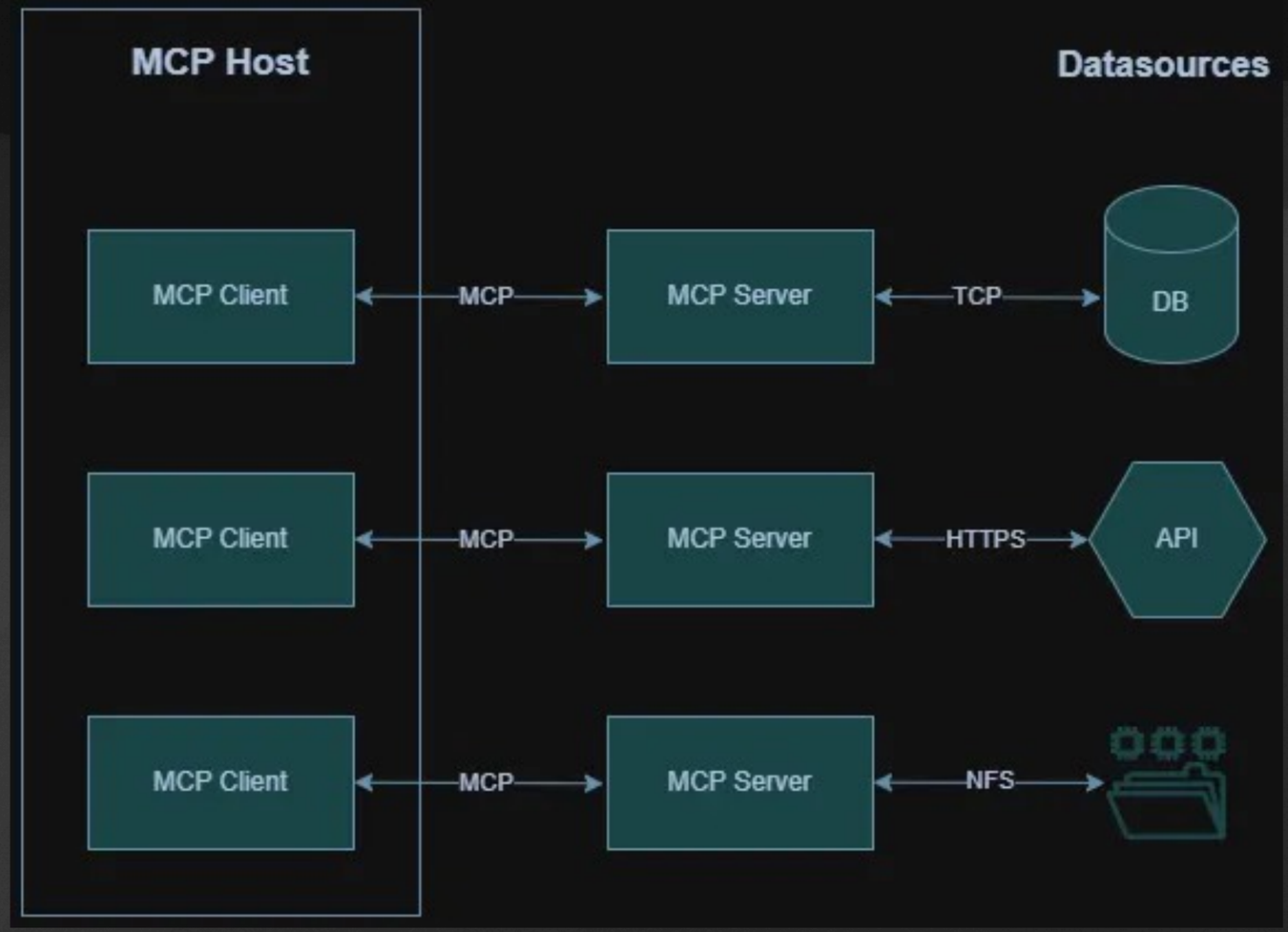
## Why?




*Source: https://www.fdintelligence.co.uk/an-introduction-to-model-context-protocol/*

# Spring AI MCP

## MCP Protocol

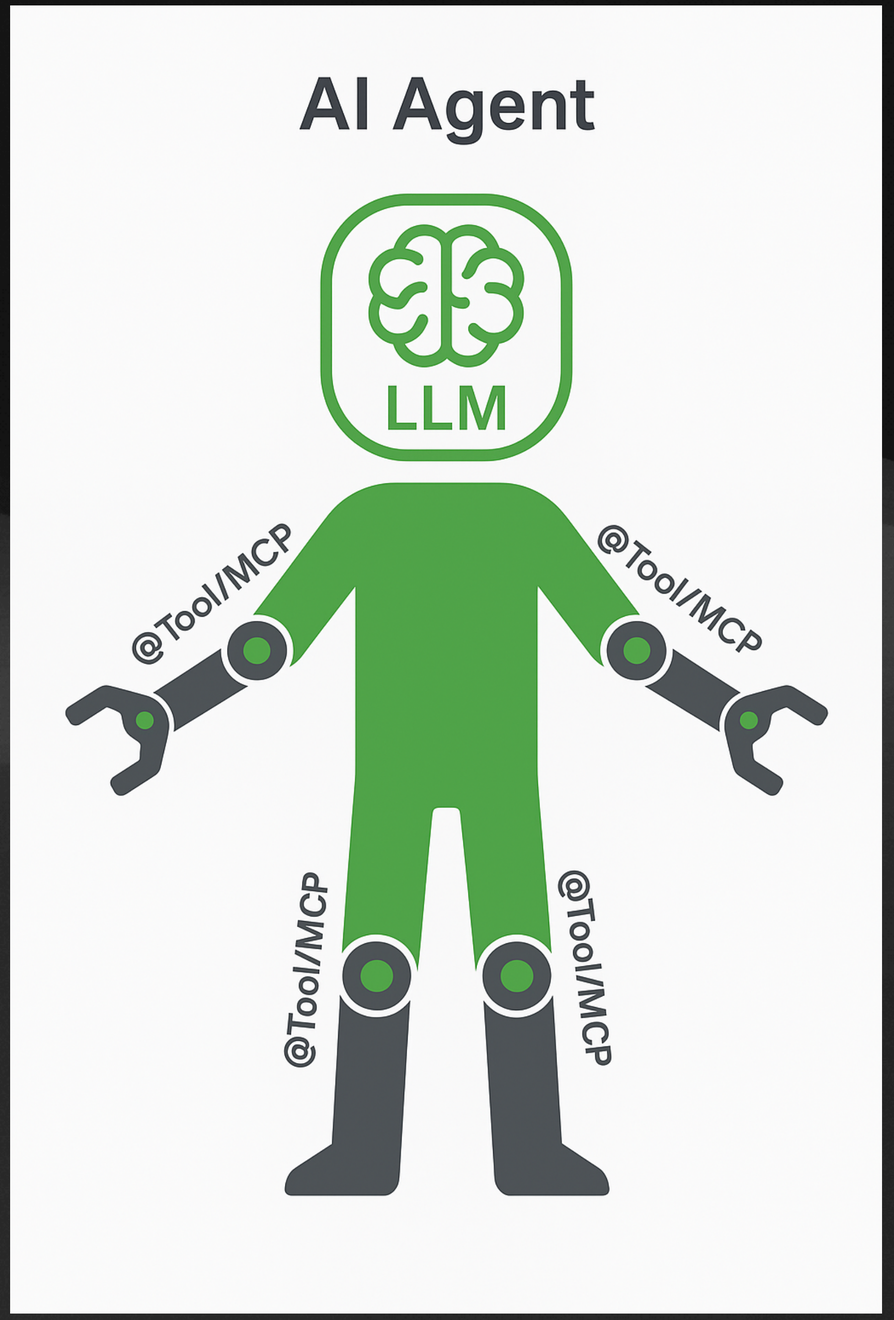



*Source: https://medium.com/@dperez_/model-context-protocol-mcp-a-standard-for-connecting-llms-to-business-contexts-df38274b5651*

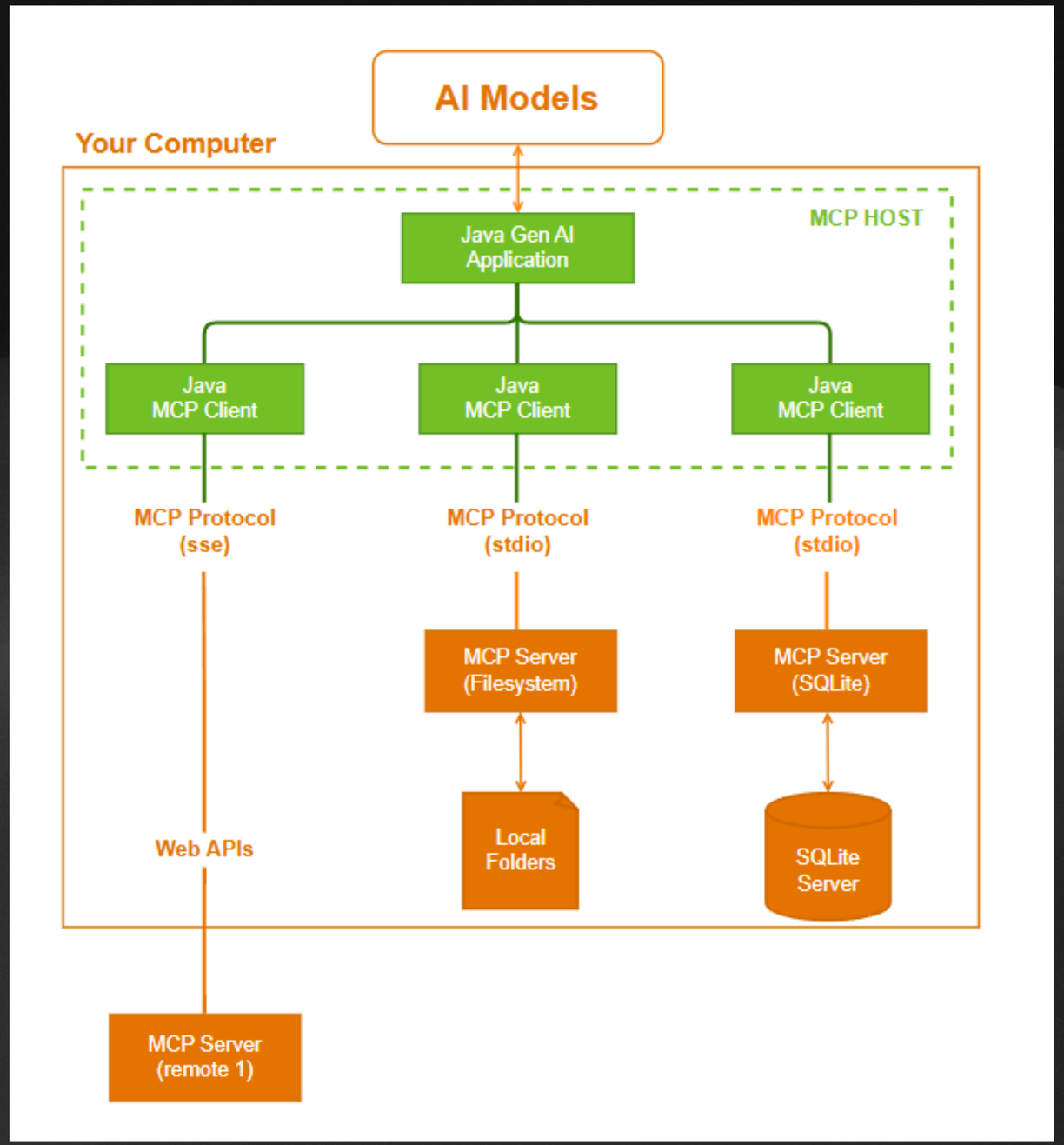
# Spring AI MCP

## AI Agent

- **LLM** -> Brain/Think
- **@Tool/MCP/A2A** -> Hands/Work



# Spring AI MCP
## Client



# Spring AI MCP

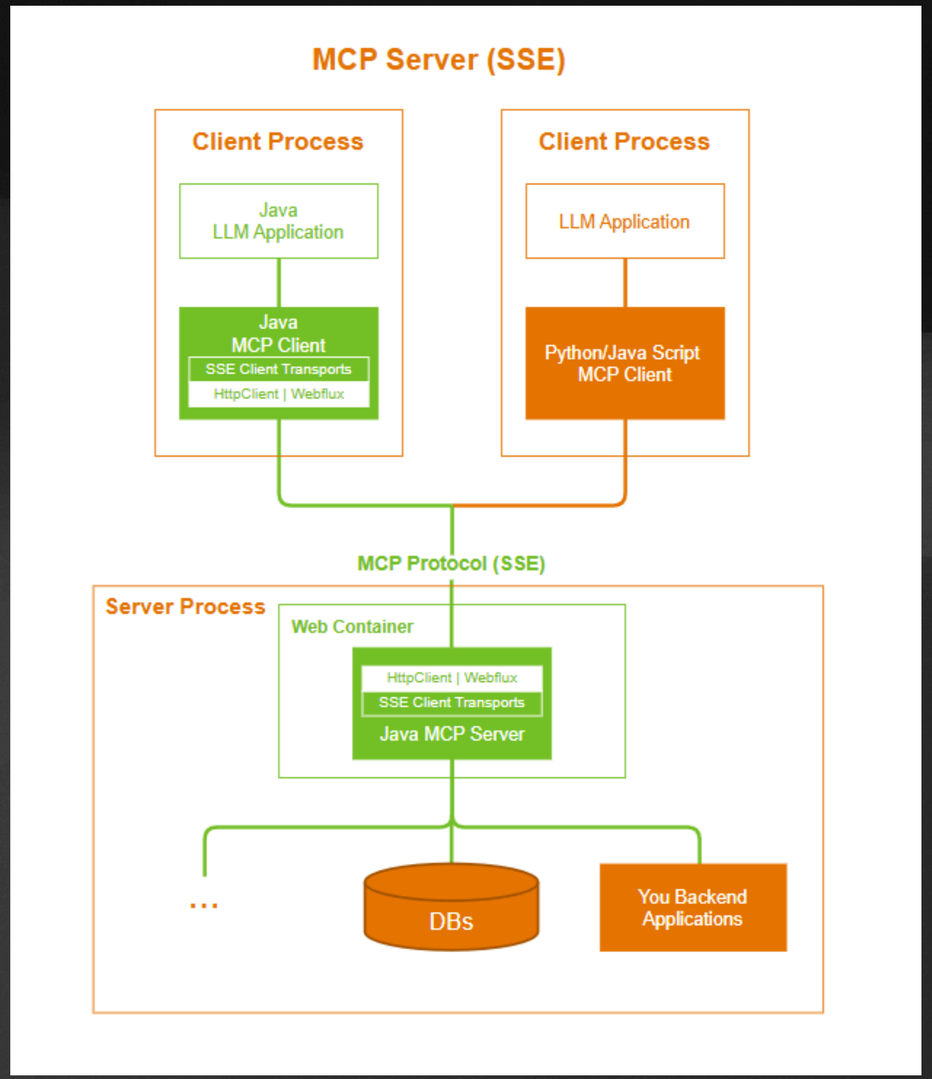
## Client Configuration

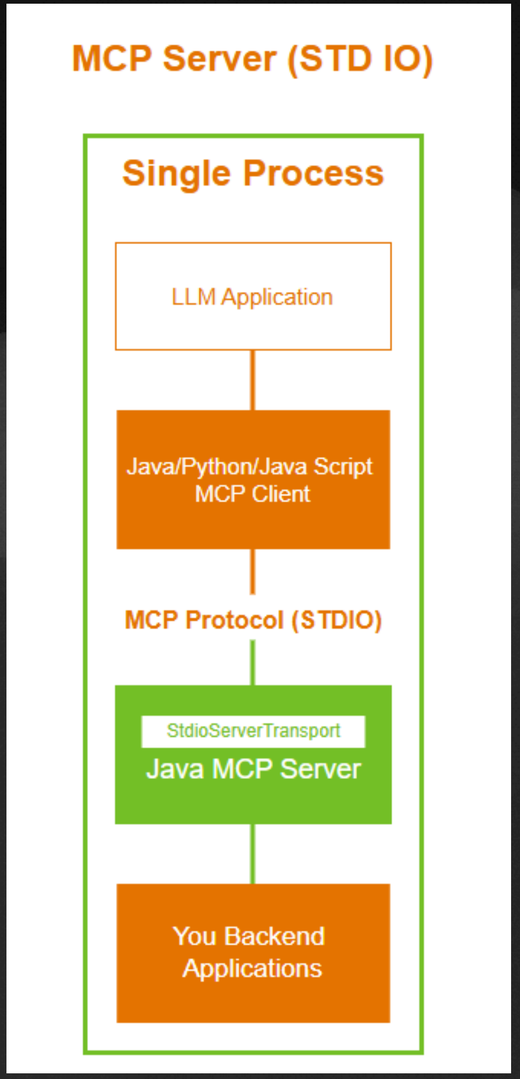
```yaml
spring:
  ai:
    mcp:
      client:
        sse:
          connections:
            task-mcp-server:
              url: http://task-server:8060
```

```java
public TaskController(ChatClient.Builder chatClientBuilder,
                      ToolCallbackProvider tools) {
    this.chatClient = chatClientBuilder
            .defaultToolCallbacks(tools)
            .build();
}
```

# Spring AI MCP

## Server




*Source: https://spring.io/projects/spring-ai*

# Spring AI MCP
## MCP Utilities

- Tool callbacks utilities
  - ToolCallBack - adapts MCP tools to Spring AI's @Tool interface
  - ToolCallBackProvider - queries connected MCP servers to retrieve available tool definitions
- McpToolUtils - Converting Spring AI tool callbacks to MCP tool specification
- **MCP Inspector** - UI tool from Anthropic

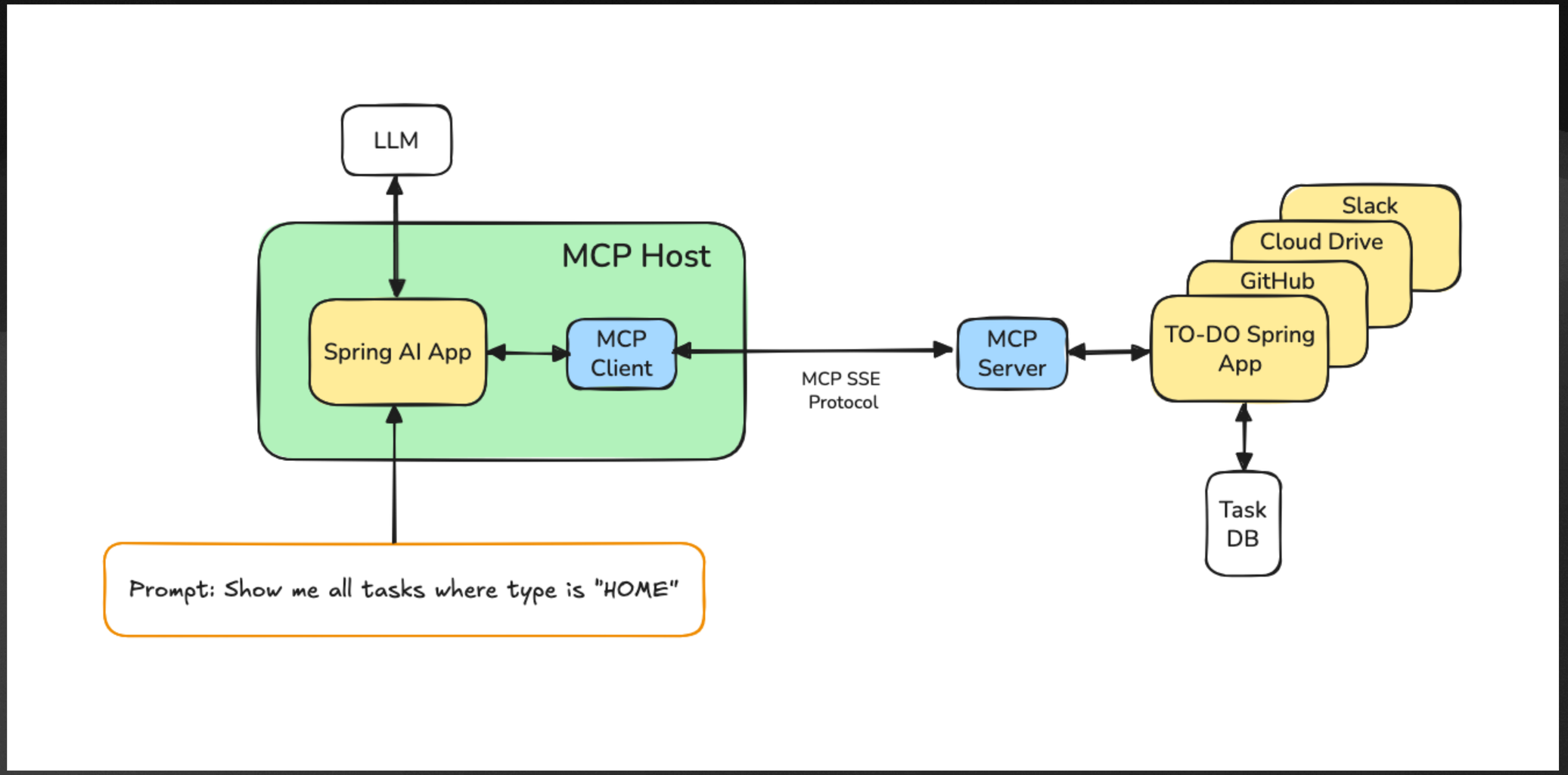
# Spring AI MCP

## Security

- **Security** had been a major issue for the **MCP**, so **OAuth2** support was added to the specification ( March 2025)
- **Spring implementation**
  - Add Spring Security & Spring Authorization Server to Application
  - Configure OAuth2 client (application.properties)
  - SecurityFilterChain - authentication and token validation

# Spring AI MCP

## MCP Example



# Q&A

Spring AI

MCP Client

MCP Server