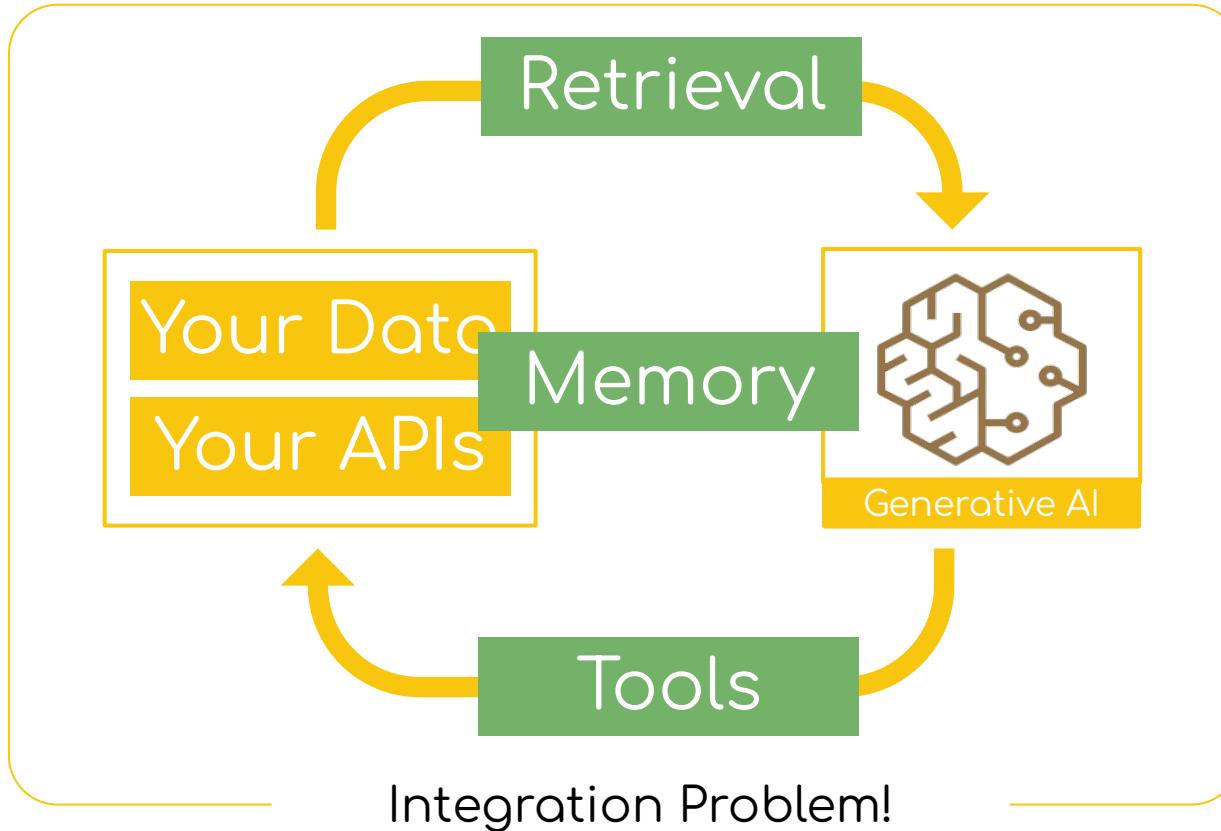


AI Integration with Spring AI and Model Context Protocol

Christian Tzolov
Spring AI & MCP Java SDK
 @christtzolov ,  @tzolov

Applying Generative AI



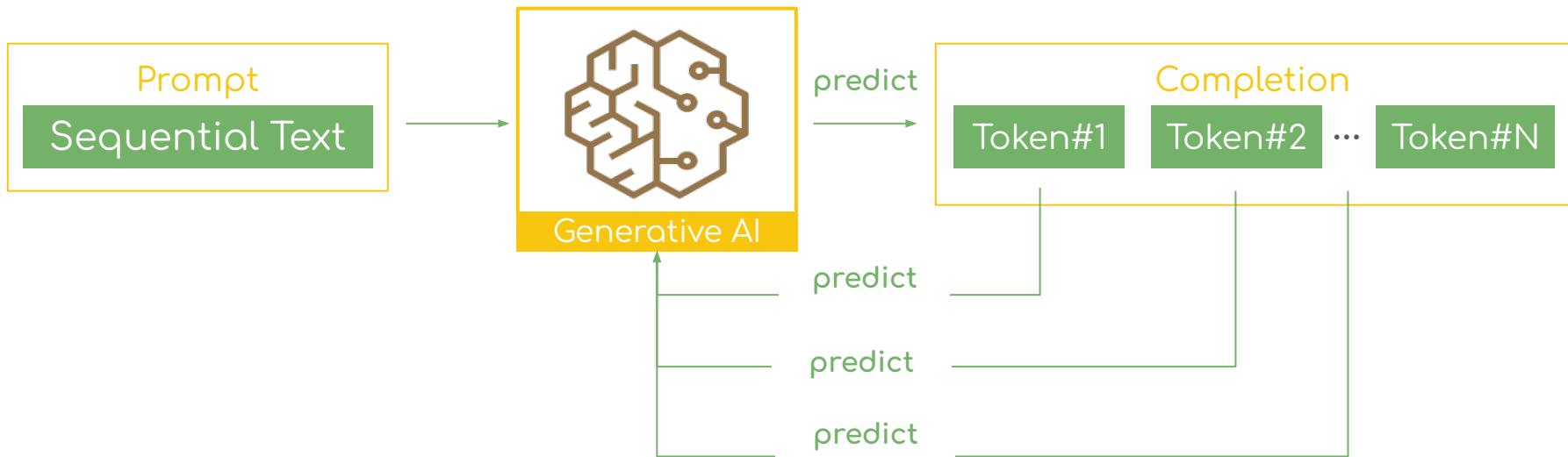


Spring AI

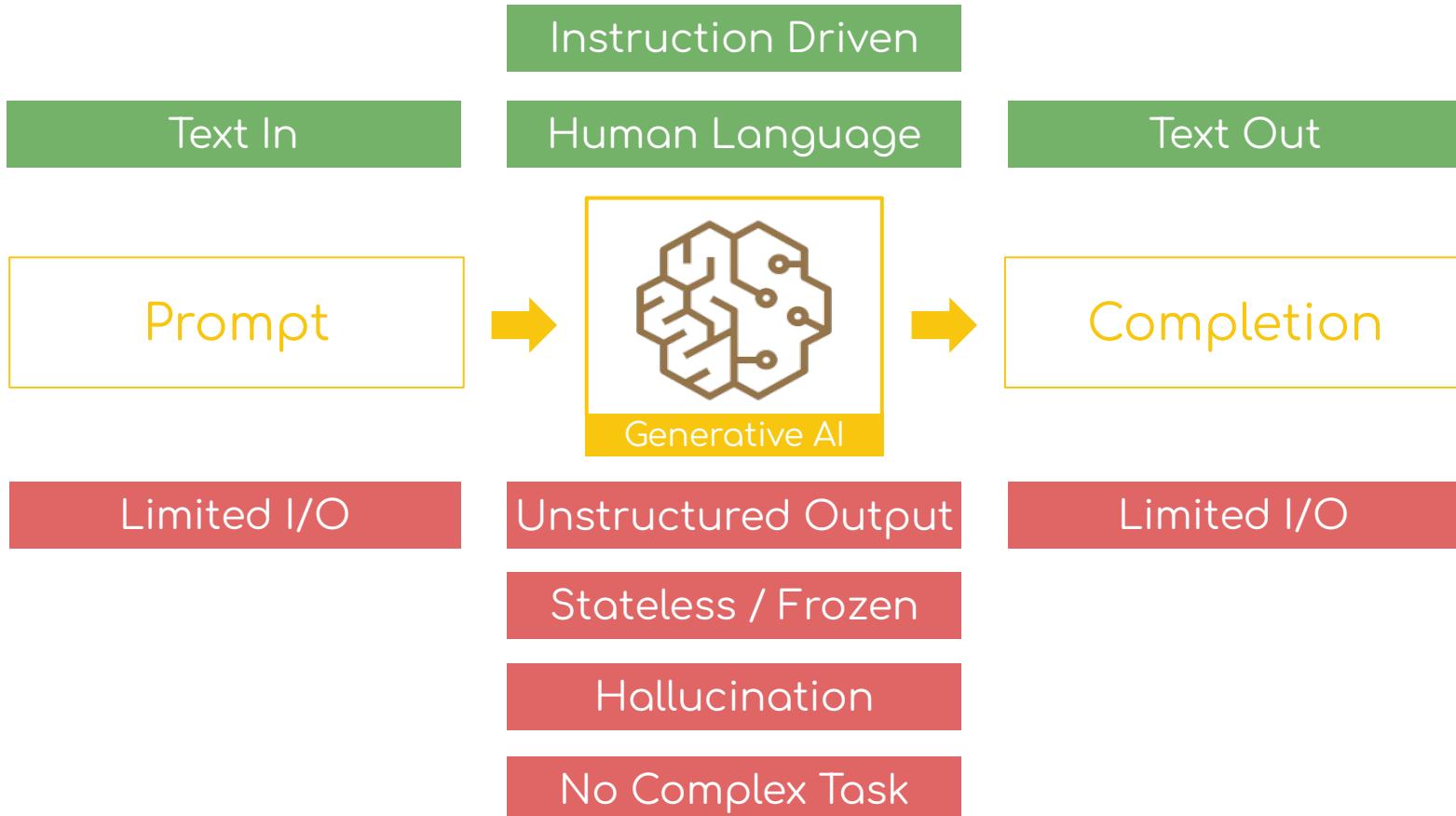
SpringAI.java

Provider	Multimodality	Tools/Functions	Streaming	Retry	Observability	Built-in JSON	Local	OpenAI API Compatible
Anthropic Claude	text, pdf, image	✓	✓	✓	✓	✗	✗	✗
Azure OpenAI	text, image	✓	✓	✓	✓	✓	✗	✓
DeepSeek (OpenAI-proxy)	text	✗	✓	✓	✓	✓	✓	✓
Google VertexAI Gemini	text, pdf, image, audio, video	✓	✓	✓	✓	✓	✗	✓
Groq (OpenAI-proxy)	text, image	✓	✓	✓	✓	✗	✗	✓
HuggingFace	text	✗	✗	✗	✗	✗	✗	✗
Mistral AI	text, image	✓	✓	✓	✓	✓	✗	✓
Minimax	text	✓	✓	✓	✓	✗	✗	✗
Moonshot AI	text	✗	✓	✓	✓	✓	✗	✗
NVIDIA (OpenAI-proxy)	text, image	✓	✓	✓	✓	✗	✗	✓
OCI GenAI/Cohere	text	✗	✗	✗	✗	✗	✗	✗
Ollama	text, image	✓	✓	✓	✓	✓	✓	✓
OpenAI	In: text, image, audio Out: text, audio	✓	✓	✓	✓	✓	✗	✓
Perplexity (OpenAI-proxy)	text	✗	✓	✓	✓	✗	✗	✓
QianFan	text	✗	✓	✓	✓	✗	✗	✗
ZhiPu AI	text	✓	✓	✓	✓	✗	✗	✗
Watsonx.AI	text	✗	✓	✗	✗	✗	✗	✗
Amazon Bedrock Converse	text, image, video, docs (pdf, html, md, docx ...)	✓	✓	✓	✓	✗	✗	✗

LLM as a System - Prediction Engine

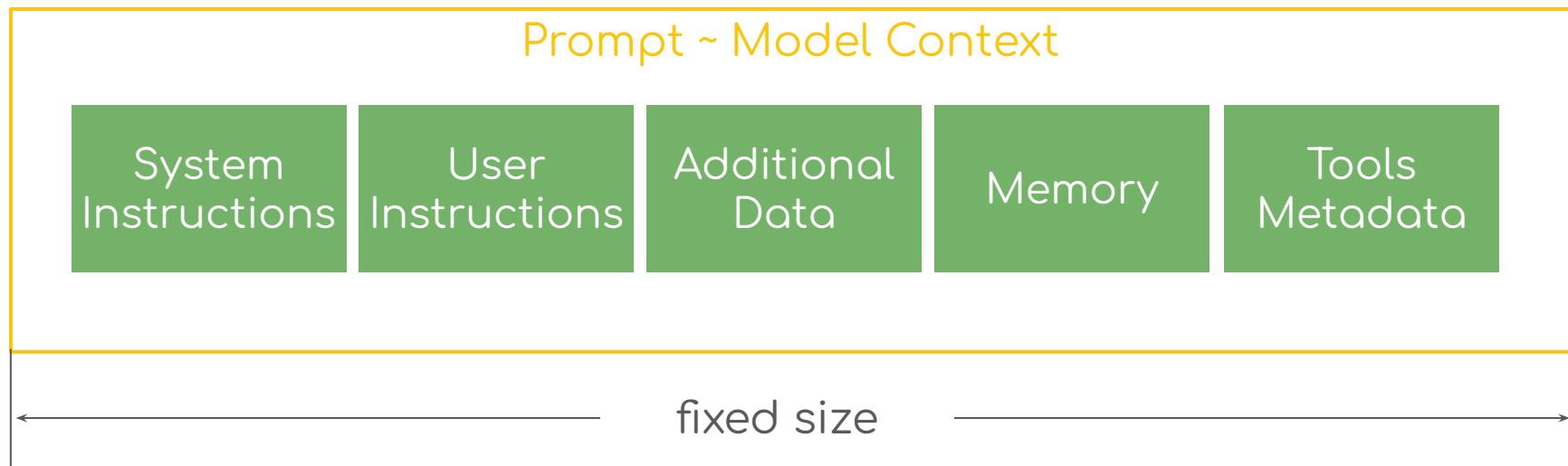


LLM Single-turn Model & Constraints



Prompt ~ Model Context

AI Models are only as good as the **Model Context** provided to them



LLM Prompt Augmentation Strategies

- System Instructions
 - Short-term Chat Memory
 - Long-term Chat Memory
- Conversation Memory
- Prompt Stuffing
 - Retrieval Augmented Generation (RAG)
- Additional Context
- Structured Output
 - Tool Calling
- Tool Metadata

Agentic System

“Leverages an **AI model** to **interact with** its **environment** in order to **solve** a **user-defined task**”

Combines **Planning** & **Actions** to fulfill the tasks



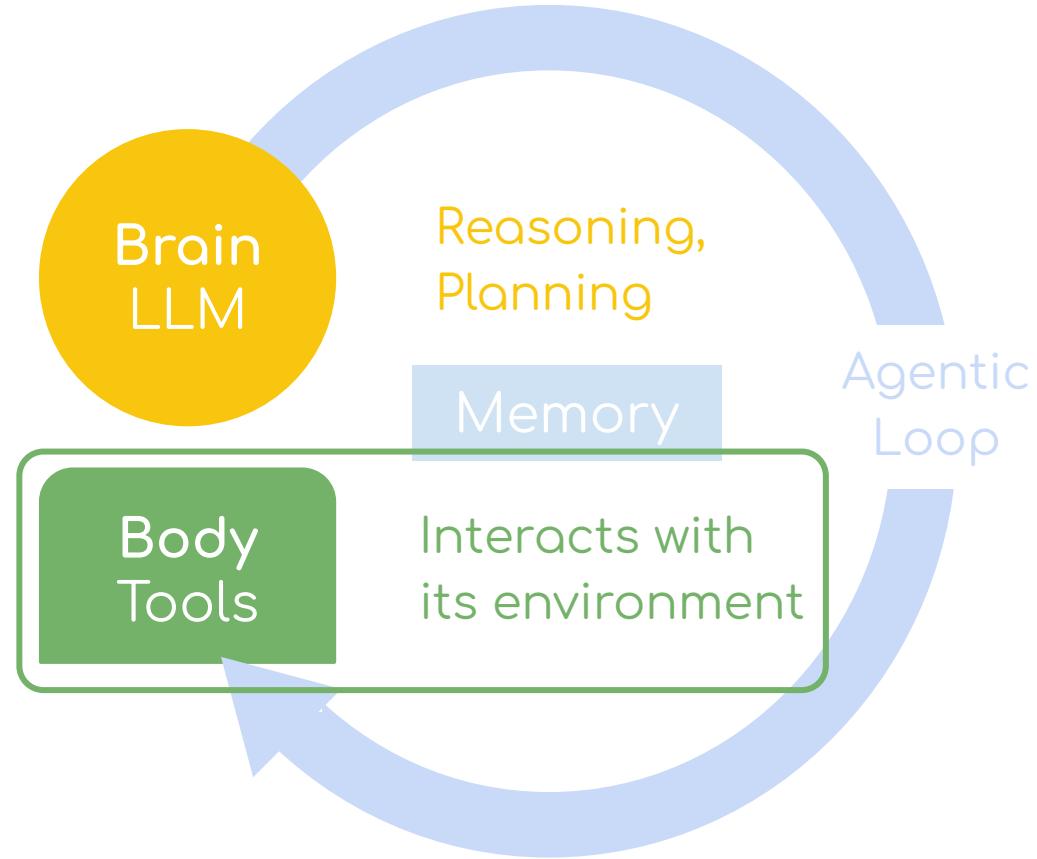
Brain (LLM)

Handles reasoning and planning
Decides which Actions to take

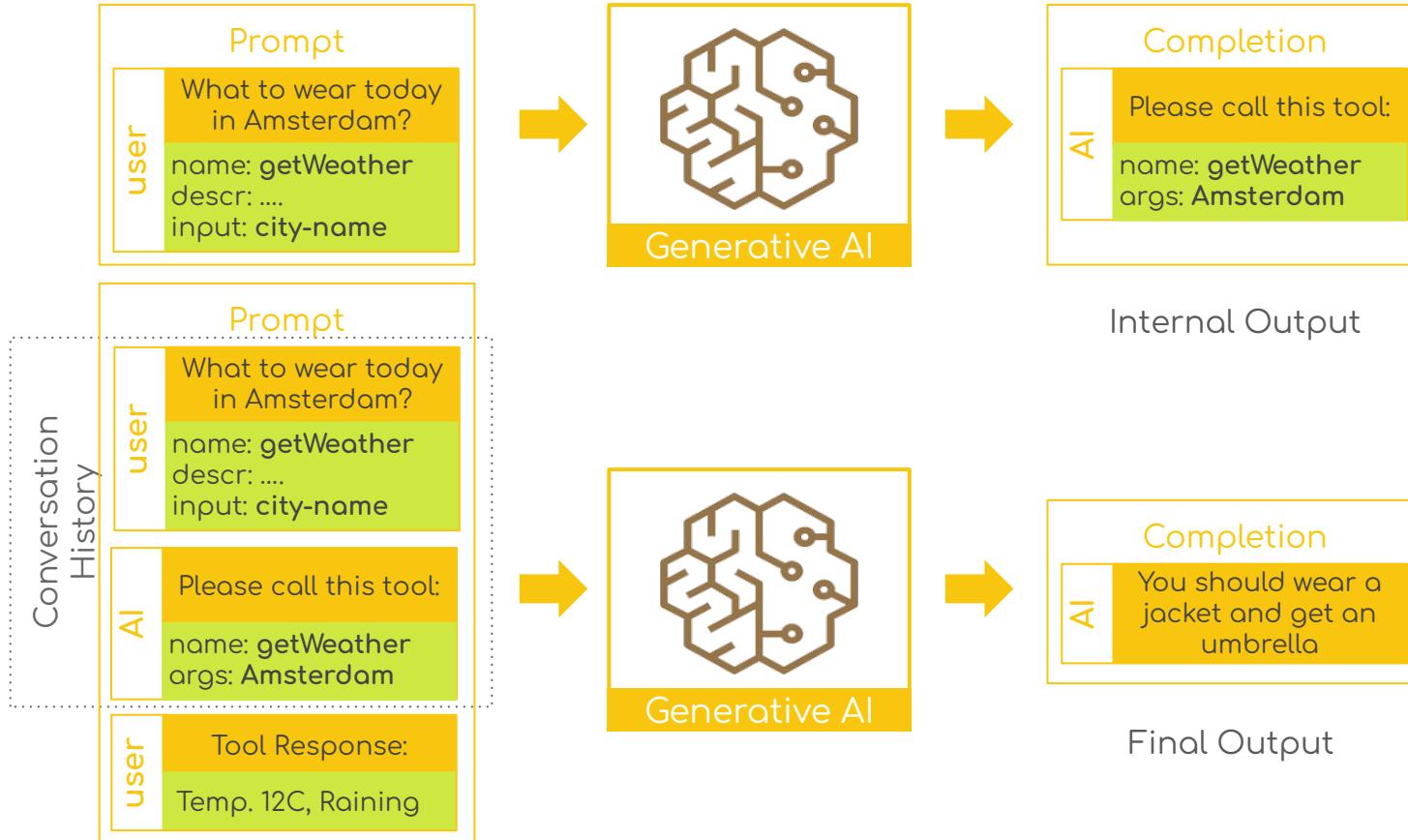


Body (Tools)

What Agent is equipped to do
Interact with its environment



Tool Calling



Tool Calling - Spring AI Demo

```
@Tool(description = "Get the temperature (in celsius) for a specific location")
public WeatherResponse getTemperature(@ToolParam(description = "The location latitude") double latitude,
    @ToolParam(description = "The location longitude") double longitude,
    @ToolParam(description = "The city name") String city) {

    WeatherResponse response = restClient
        .get()
        .uri("https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&current=tempera
            latitude, longitude)
        .retrieve()
        .body(WeatherResponse.class);

    logger.info("Check temparature for {}. Lat: {}, Lon: {}. Temp: {}", city, latitude, longitude,
        response.current);

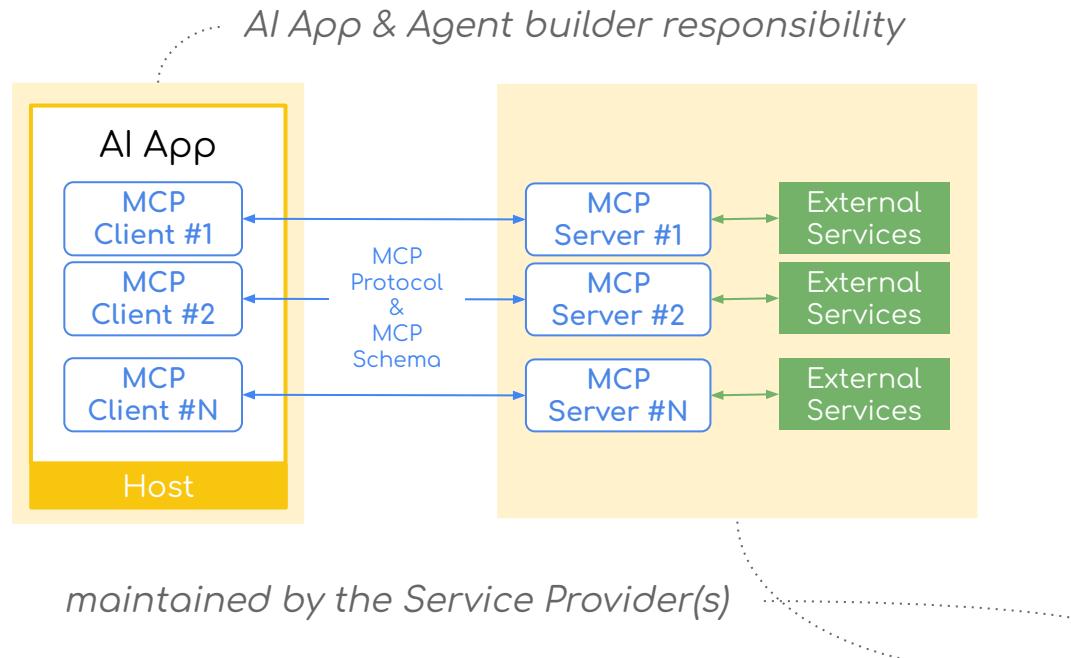
    return response;
}
```

```
WeatherTools myTools = ...
var output = chatClient.prompt()
    .tools(myTools)
    .user("What to wear today in Amsterdam and in Barcelona?")
    .call()
    .content();
```

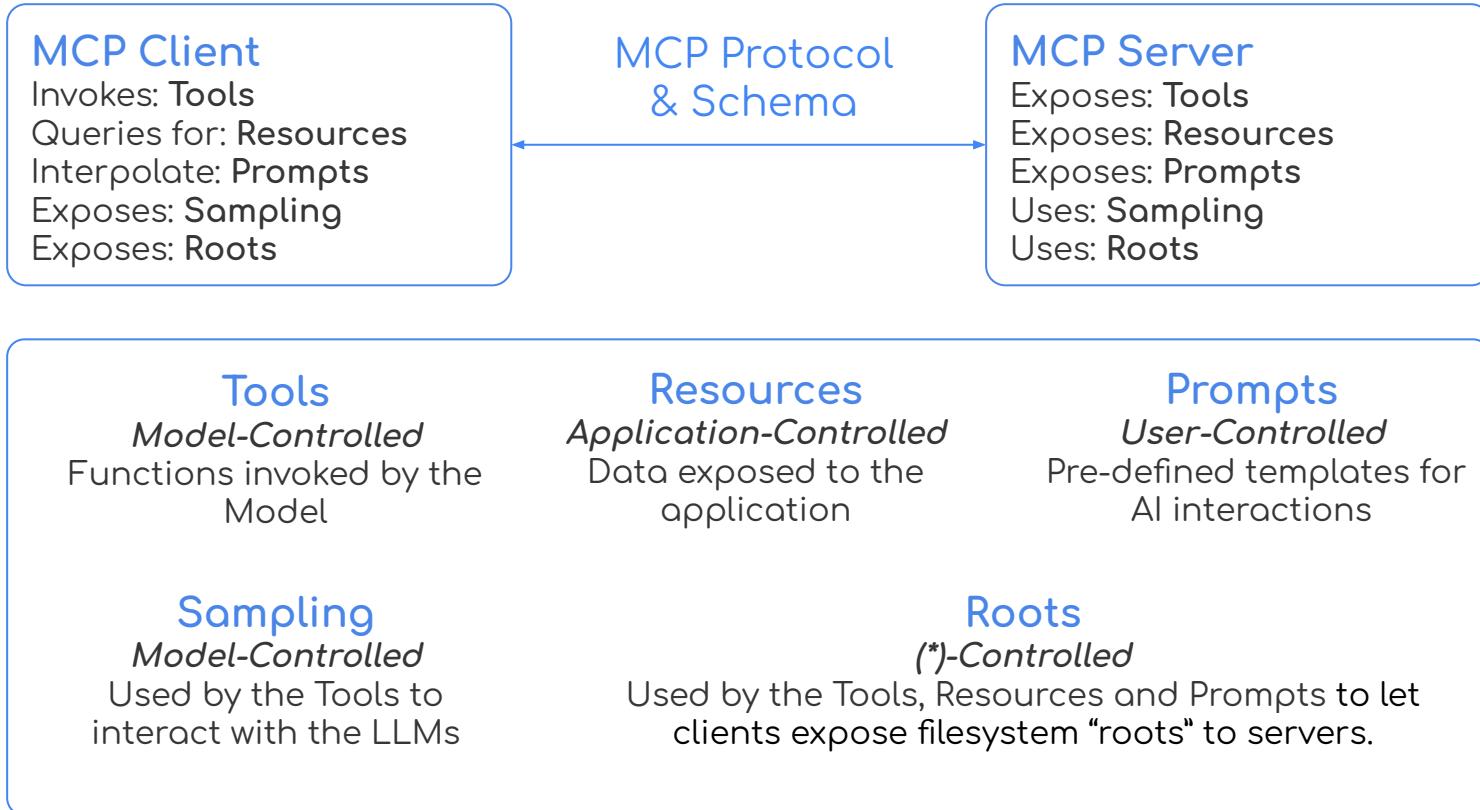
Wouldn't it be nice ...

- To be able to use Tools written in other languages?
- To have Spring AI Tools used in non-Java envs?

MCP Components



MCP Deep-Dive

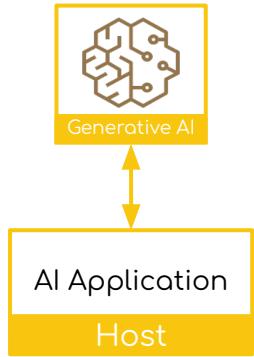


What We'll Cover?

- Introduction
 - LLM as a System
 - Agentic AI System
 - Model Context Protocol
- Unified way to connect AI applications and Agents to different data sources, tools, ...

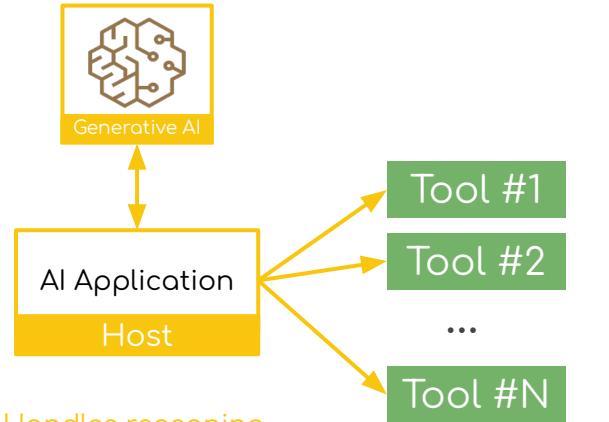
The Road ...

1 LLM



Text-In /
Text-Out

2 LLM + Tool Calling



Handles reasoning
and planning.
Decides which
Actions to take

Agent

Brain
LLM

Body
Tools

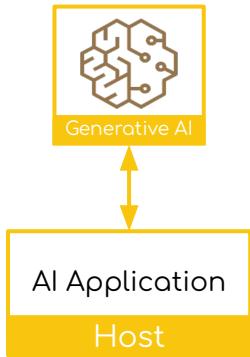
Equipment to
Interact with the
environment

Wouldn't it be nice ...

- To be able to use Tools written in other languages?
- To have Spring AI Tools used in non-Java envs?

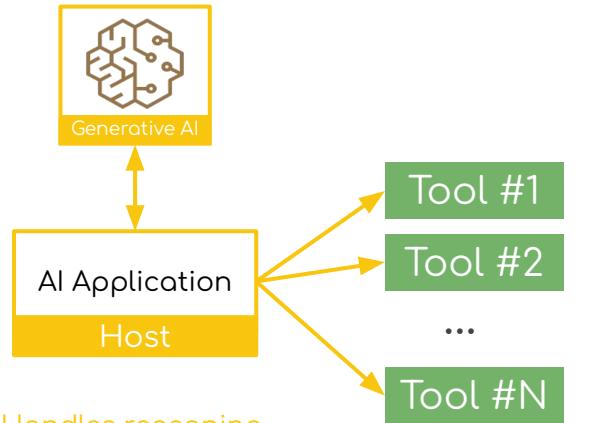
The Road to Model Context Protocol (MCP)

1 LLM

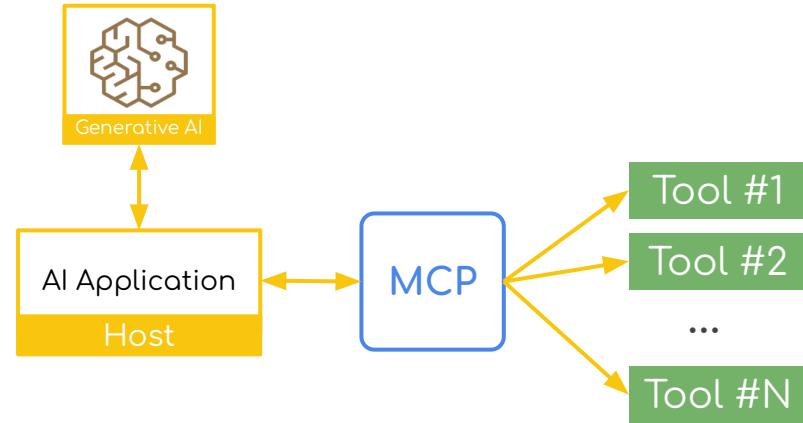


Text-In /
Text-Out

2 LLM + Tool Calling

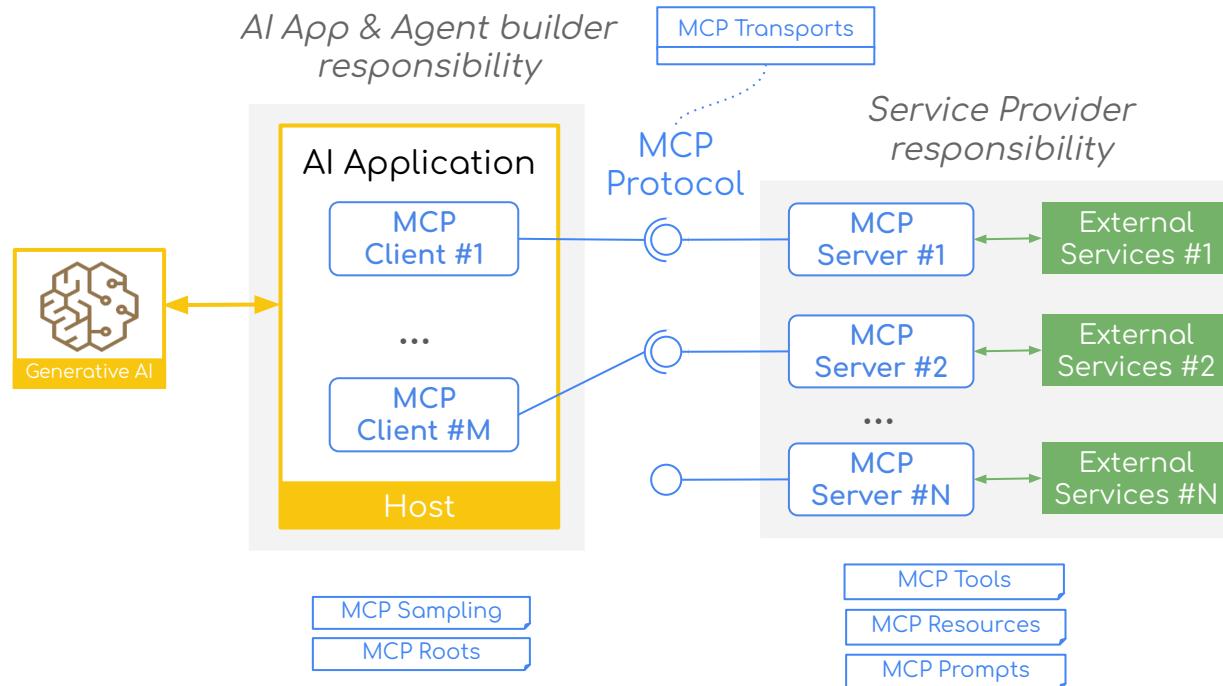


3 LLM + MCP



Unified way to connect AI applications to different data sources, tools, ...

MCP Components



MCP Client & Server Ecosystem

MCP Hub

Model Context Protocol

A protocol for extending AI capabilities through local servers

Model Context Protocol Universal Standard Semantic Understanding Context Aware
Secure Connection

313 Live

Search by name...

Personal MCP Server YouTube Watch Later MCP Server MCP Knowledge Graph

Dify MCP Server (TypeScript) COLUMBIA-MCP-SERVERS MCP Tool Builder

Personal MCP Server AI-assisted personal health tracking server 2 months ago	YouTube Watch Later MCP Server Access YouTube Watch Later playlist via MCP 2 months ago	MCP Knowledge Graph A server enabling persistent memory for Claude. 2 months ago
Dify MCP Server (TypeScript) A TypeScript MCP server for Dify workflows. 2 months ago	COLUMBIA-MCP-SERVERS Deployment infrastructure for Columbia's MCP servers 2 months ago	MCP Tool Builder A server for dynamically creating tools 2 months ago
...

MCP Client & Server Ecosystem

Running MCP-Based

Learn from Java code examples how to run MCP-based AI Agents in AWS, using Amazon Bedrock and AWS Lambda.

mcp bedrock ai-agents java

James Ward
Amazon Employee

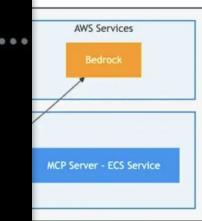
Published Apr 1, 2025 | Last Modified

Like (4) Comments

Satya Nadella  CEO
@satyanadella

GitHub Copilot Agent Mode & MCP support is rolling out to all VS Code users:

which will then call our MCP scheduling service. Here's what it looks like:



There are many exciting things happening in the AI space, and one of them is that it is becoming easier to integrate AI into our everyday lives. At the heart of this are AI Agents which can be used to perform various tasks.

← Post

Sam Altman  @sama

people love MCP and we're excited to announce that we'll be supporting it for our Gemini models and SDK. Look forward to developing it further with the MCP team and others in the industry

Demis Hassabis  @demishassabis

First we will need an SSE-transport based MCP server running in a VPC. As mentioned earlier there are some requirements for the MCP server to work correctly.

a basic Java / Spring AI

server with MCP support.

Following



Demis Hassabis  @demishassabis

Co-Founder & CEO @GoogleDeepMind
- working on AGI. Revolutionising drug discovery @IsomorphicLabs. Trying to understand the nature of reality. Nobel Laureate.

7:02 PM · Mar 26, 2025 · 1.7M Views

487

1.3K

10K

1.8K



```
17
18 @Service
19 class DogAdoptionAppointmentScheduler {
20
21     @Tool(description = "schedule an appointment to adopt a dog" +
22         " at the Pooch Palace dog adoption agency")
```

MCP Official Java SDK

modelcontextprotocol.io/sdk/java/mcp-overview

Model Context Protocol Documentation SDKs Features

Search... ⌘K

Python SDK TypeScript SDK Java SDK Kotlin SDK C# SDK Specification

Java Overview MCP Client MCP Server

Protocol Tool discovery Resources Roots list

Code Issues 22 Pull requests 12 Zenhub Discussions Actions Security

java-sdk Public Edit Pins Watch 19 Fork 144 Starred 750

Christian Tzolov Dariusz Jędrzejczyk

Failed to start process with... 79ec5b5 · 4 days ago

Add CI GitHub Action and rename s... 2 weeks ago

initial 4 months ago

Next development version 2 weeks ago

feat(webflux): Add configurable SS... last week

refactor: remove deprecated 0.7.0 c... last week

fix(tests): Failed to start process wit... 4 days ago

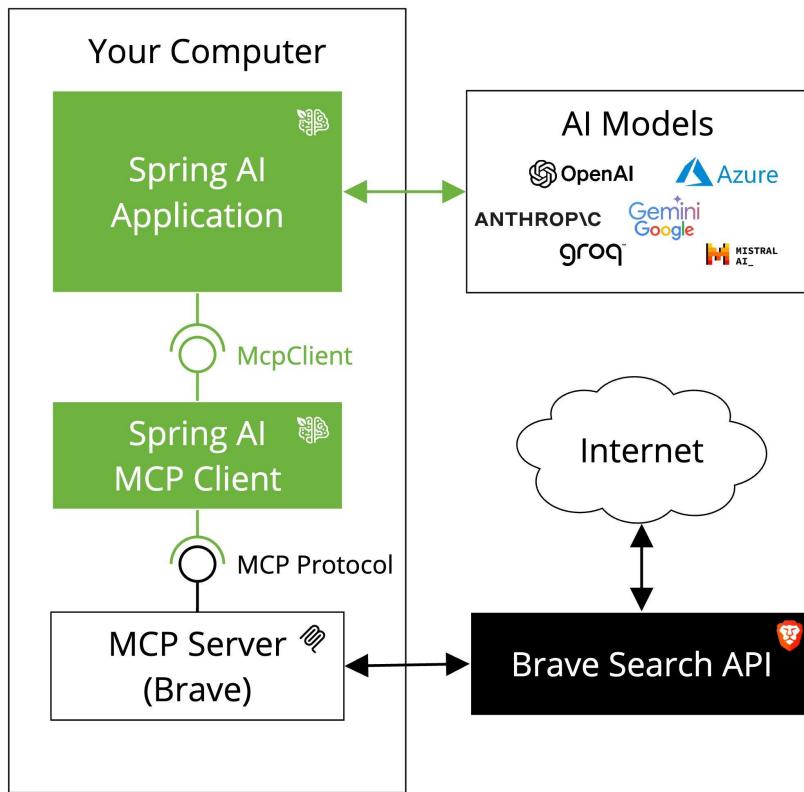
About

The official Java SDK for Model Context Protocol servers and clients. Maintained in collaboration with Spring AI

modelcontextprotocol.io/sdk/java/mcp-overview

Readme MIT license Code of conduct Security policy Activity Custom properties

MCP Spring AI - WebSearch & Filesystem Tools



The screenshot shows a code editor with two tabs: "mcp-servers-config.json" and "Application.java".

mcp-servers-config.json:

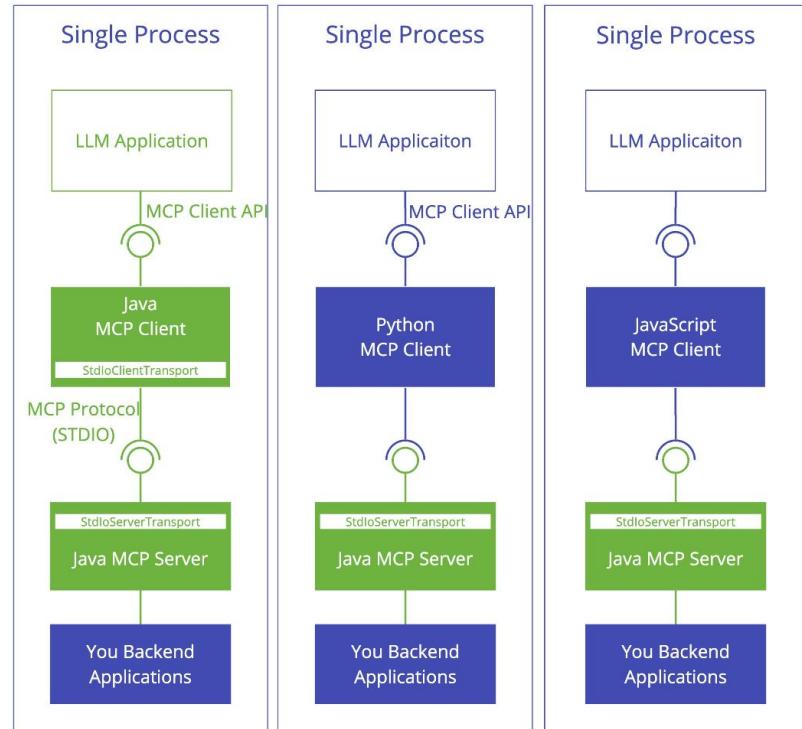
```
{  
  "mcpServers": {  
    "brave-search": {  
      "command": "npx",  
      "args": [  
        "-y",  
        "@modelcontextprotocol/server-brave-search"  
      ],  
      "env": {}  
    },  
    "filesystem": {  
      "command": "npx",  
      "args": [  
        "-y",  
        "@modelcontextprotocol/server-filesystem",  
        "/Users/christiantzolov/Desktop/tmp"  
      ]  
    }  
  }  
}
```

Application.java:

```
@Bean  
public CommandLineRunner chatbot(ChatClient.Builder chatClientBuilder, ToolCallbackProvider tools)  
  
  return args -> {  
    var output = chatClientBuilder.build().prompt()  
    .system("You are useful assistant and can perform web searches Brave's search API to reply  
    .user("Create a summary about the Talent Arena conference and save it as markdown talent-a  
    .tools(tools)  
    .call()  
    .content();  
  };  
}
```

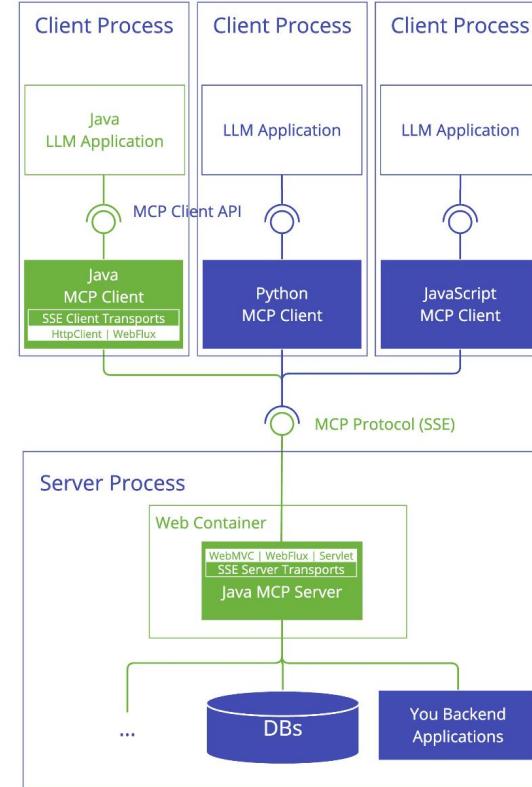
MCP Transports - STUDIO

- The client launches the MCP server as a subprocess
- JSON-RPC messages over (stdin/stdout)



MCP Transports - HTTP SSE

- Server is independent process
- Multiple client connections
- SSE endpoint - connection establishment & server to client communication
- HTTP POST endpoint - client to server communication



MCP Transports - Java SDK

Default MCP transports

- No external web frameworks required
- Stdio, client/server transports
- Java HttpClient-based SSE client
- Servlet-based SSE server

Optional Spring-based transports

- WebFlux SSE client and server transports
- WebMVC SSE server transport

MCP Java SDK - Initialization

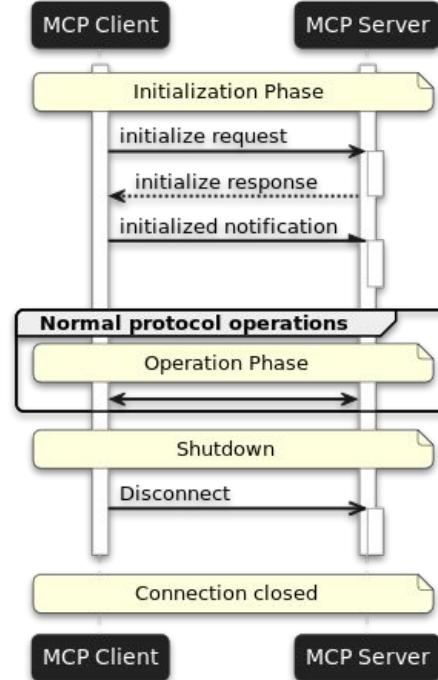
MCP Client

```
McpSyncClient client = McpClient.sync(transport)
    .requestTimeout(Duration.ofSeconds(10))
    .capabilities(ClientCapabilities.builder()
        .roots(true)      // Enable roots capability
        .sampling()       // Enable sampling capability
        .build())
    .sampling(request -> new CreateMessageResult(response))
    .build();

// Initialize connection
client.initialize();
```

MCP Server

```
// Create a server with custom configuration
McpSyncServer syncServer = McpServer.sync(transportProvider)
    .serverInfo("my-server", "1.0.0")
    .capabilities(ServerCapabilities.builder()
        .resources(true)      // Enable resource support
        .tools(true)          // Enable tool support
        .prompts(true)        // Enable prompt support
        .logging()            // Enable logging support
        .build())
    .build();
```



MCP Java SDK - Initialization -Spring AI MCP

```
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-starter-mcp-client</artifactId>
</dependency>
```

Alternatively, you can configure stdio connections using an external JSON file using the [Claude Desktop format](#):

```
spring:
  ai:
    mcp:
      client:
        stdio:
          servers-configuration: classpath:mcp-servers.json
```

The Claude Desktop format looks like this:

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "/Users/username/Desktop",
        "/Users/username/Downloads"
      ]
    }
  }
}
```

MCP Java SDK - Tools

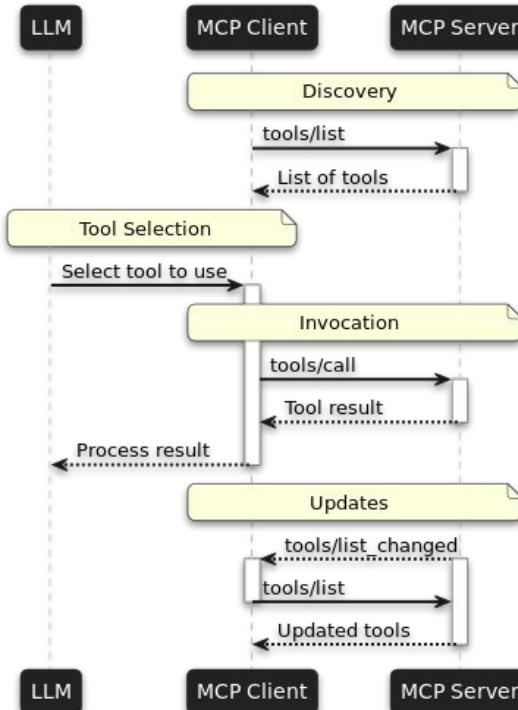
MCP Client

```
var tools = client.listTools();
tools.forEach(tool -> System.out.println(tool.getName()));

// Execute a tool with parameters
var result = client.callTool("calculator", Map.of(
    "operation", "add",
    "a", 1,
    "b", 2
```

MCP Server

```
var syncToolSpecification = new McpServerFeatures.SyncToolSpecification()
    .new Tool("calculator", "Basic calculator", schema),
    (exchange, arguments) -> {
        // Tool implementation
        return new CallToolResult(result, false);
}
```



MCP Java SDK - Tools - Spring AI MCP - Demo

```
public record WeatherResponse(Current current) {
    public record Current(LocalDateTime time, int interval, double temperature_2
    )
}

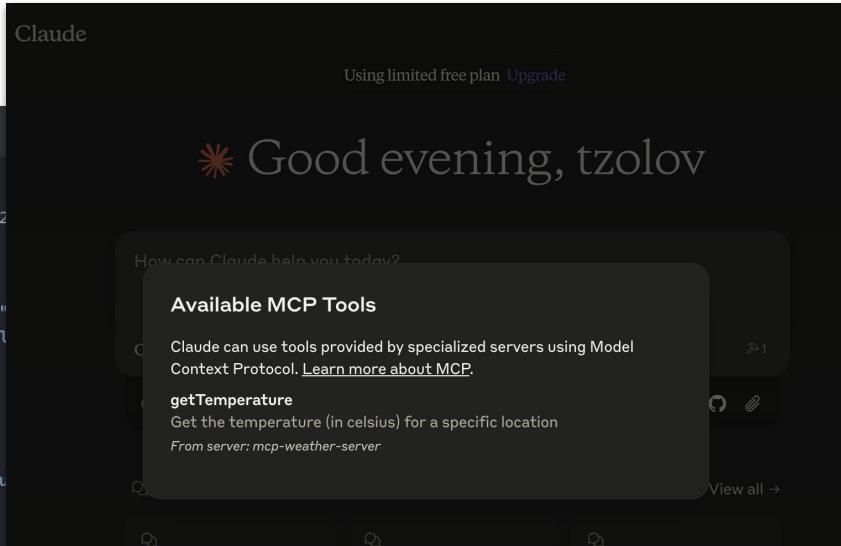
@Tool(description = "Get the temperature (in celsius) for a specific location"
public WeatherResponse getTemperature(@ToolParam(description = "The location l
    @ToolParam(description = "The location longitude") double longitude,
    @ToolParam(description = "The city name") String city) {

    WeatherResponse response = restClient
        .get()
        .uri("https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitu
            latitude, longitude)
        .retrieve()
        .body(WeatherResponse.class);

    logger.info("Check temparature for {} Lat: {} Lon: {} Temp: {}", city, latitude, longitude
        response.current);

    return response;
}
```

```
<dependency>
    <groupId>org.springframework.ai</groupId>
    <artifactId>spring-ai-starter-mcp-server</artifactId>
</dependency>
```



MCP Java SDK - Resources

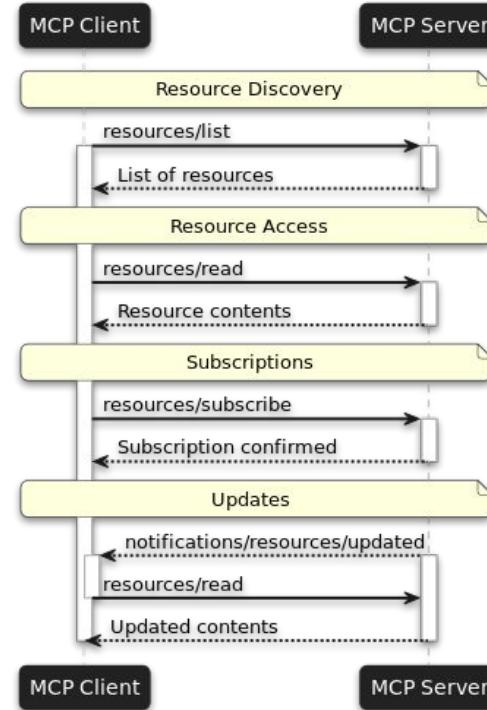
MCP Client

```
var resources = client.listResources();
resources.forEach(resource -> System.out.println(resource.get

// Retrieve resource content using a URI template
var content = client.getResource("file", Map.of(
    "path", "/path/to/file.txt"
));
```

MCP Server

```
var syncResourceSpecification = new McpServerFeatures.syncResourceSpecification()
    .with(new Resource("custom://resource", "name", "description", "mime-type"))
(exchange, request) -> {
    // Resource read implementation
    return new ReadResourceResult(contents);
}
);
```



MCP Java SDK - Prompts

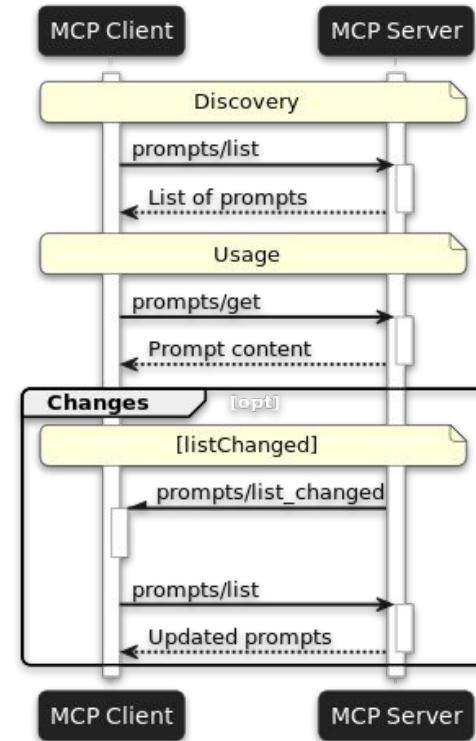
MCP Client

```
// List available prompt templates
var prompts = client.listPrompts();
prompts.forEach(prompt -> System.out.println(prompt.getName()));

// Execute a prompt template with parameters
var response = client.executePrompt("echo", Map.of(
    "text", "Hello, World!"
));
```

MCP Server

```
var syncPromptSpecification = new McpServerFeatures.syncPromptSpec
    new Prompt("greeting", "description", List.of(
        new PromptArgument("name", "description", true)
    )),
    (exchange, request) -> {
        // Prompt implementation
        return new GetPromptResult(description, messages);
    }
};
```



MCP Java SDK - Sampling

MCP Client

```
McpSyncServer server = McpServer.sync(transportProvider)
    .serverInfo("my-server", "1.0.0")
    .build();

// Define a tool that uses sampling
var calculatorTool = new McpServerFeatures.SyncToolSpecification(
    new Tool("ai-calculator", "Performs calculations using AI", schema),
    (exchange, arguments) -> {
        // Check if client supports sampling
        if (exchange.getClientCapabilities().sampling() == null) {
            return new CallToolResult("Client does not support AI capabilities");
        }

        // Create a sampling request
        McpSchema.CreateMessageRequest request = McpSchema.CreateMessageRequest
            .content(new McpSchema.TextContent("Calculate: " + arguments.get("query")))
            .modelPreferences(McpSchema.ModelPreferences.builder())
            .hints(List.of(
                McpSchema.ModelHint.of("claude-3-sonnet"),
                McpSchema.ModelHint.of("claude")
            ))
            .intelligencePriority(0.8) // Prioritize intelligence
            .speedPriority(0.5)       // Moderate speed importance
            .build();

        .systemPrompt("You are a helpful calculator assistant. Provide only numbers in the response." + System.lineSeparator())
        .maxTokens(100)
        .build();

        // Request sampling from the client
        McpSchema.CreateMessageResult result = exchange.createMessage(request);

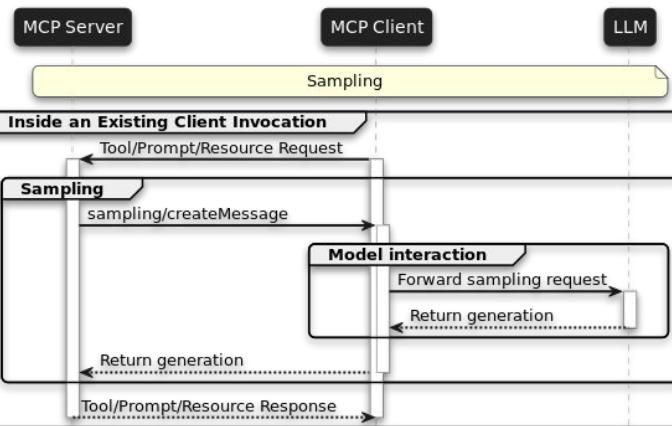
        // Process the result
        String answer = result.content().text();
        return new CallToolResult(answer, false);
    }
);

// Add the tool to the server
server.addTool(calculatorTool);
```

MCP Server

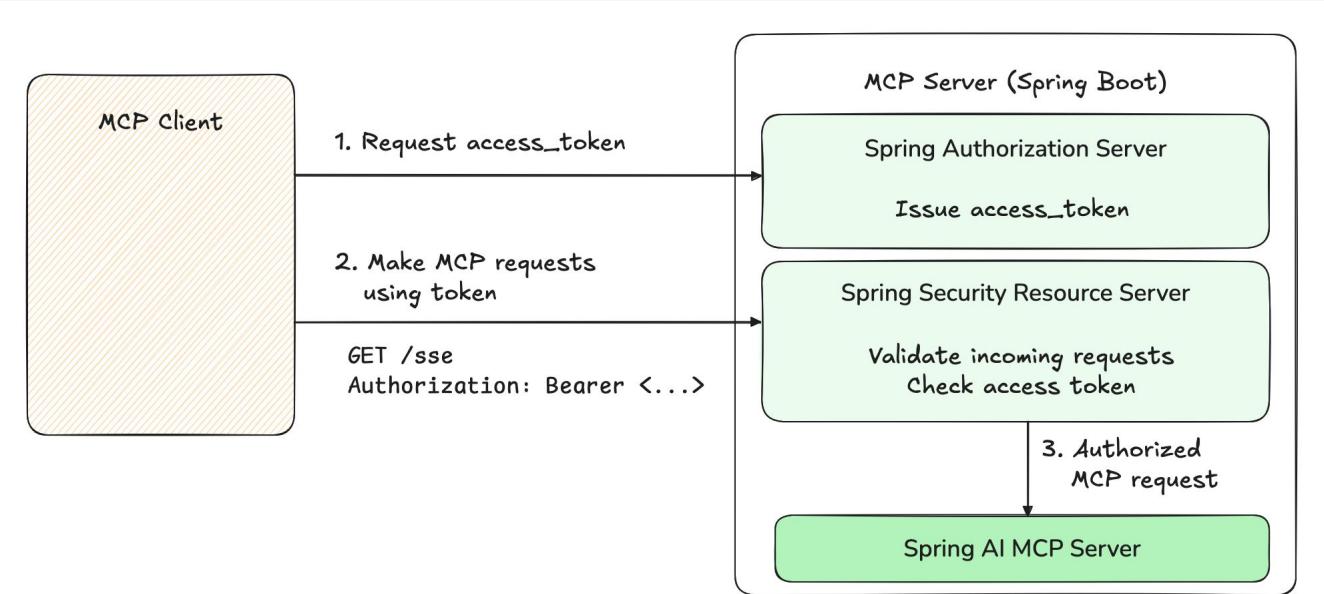
```
Function<CreateMessageRequest, CreateMessageResult> samplingHandler = () -> {
    // Sampling implementation that interfaces with LLM
    return new CreateMessageResult(response);
};

// Create client with sampling support
var client = McpClient.sync(transport)
    .capabilities(ClientCapabilities.builder()
        .sampling()
        .build())
    .sampling(samplingHandler)
    .build();
```



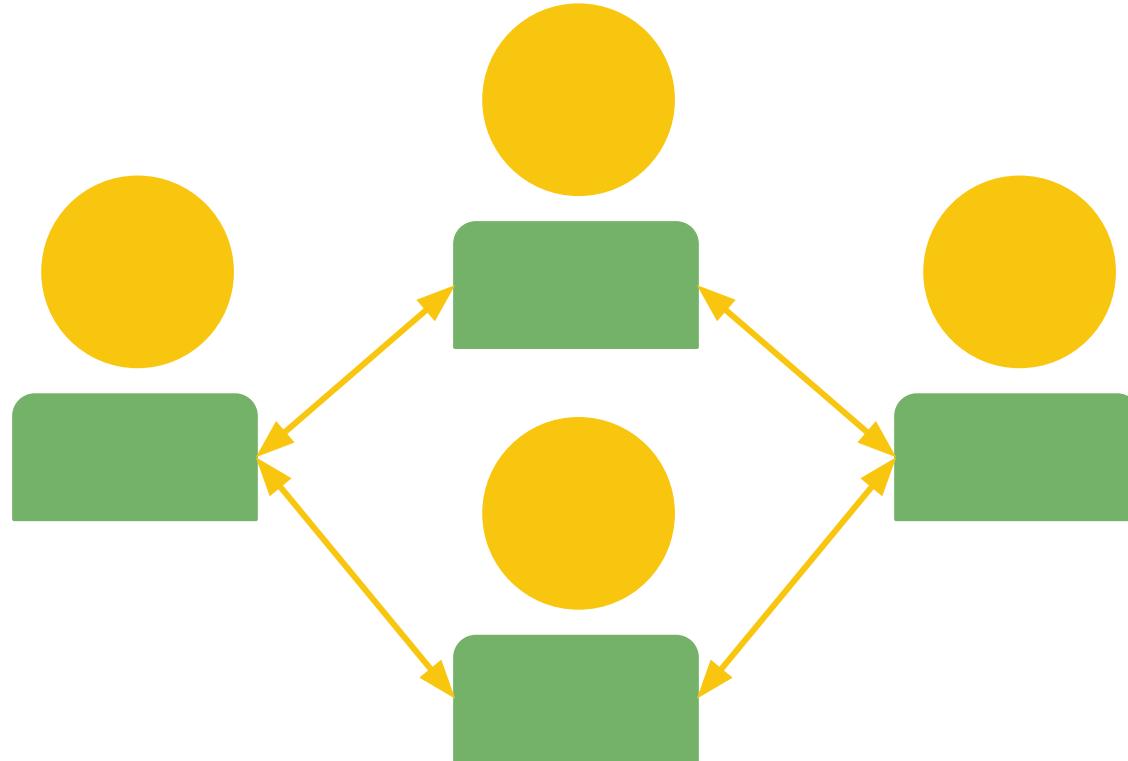
MCP Authorization

- New Features
 - API
 - Configuration
 - Error Handling
 - Session Management
 - Federation



- Spring
<https://github.com/spring-projects/spring-ai-examples/tree/main/model-context-protocol/weather/starter-webmvc-oauth2-server>

Agentic Systems



Agentic Systems Types

Engineering at Anthropic

- Various types of agents



Building effective agents

Published Dec 19, 2024

We've worked with dozens of teams building LLM agents across industries. Consistently, the most successful implementations use simple, composable patterns rather than complex frameworks.

accomplishing the tasks - e.g
autonomous

Retrieval

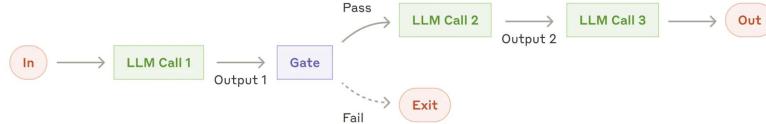
Tools

Memory

Agentic Workflows

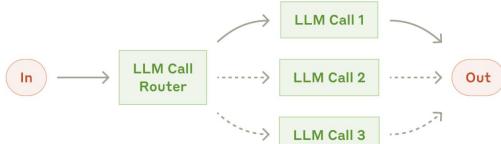
Chain Workflow

Break complex tasks down into simpler, more manageable steps



Routing Workflow

Complex tasks with different input types, handled by specialized processes. An LLM analyzes the input content and routes it to the specialized handler.



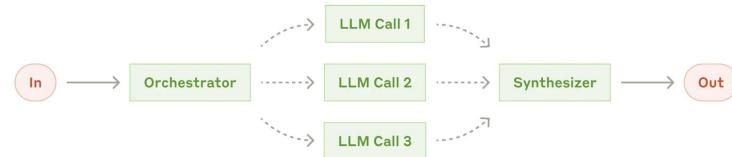
Parallelization Workflow

Work simultaneously on tasks and aggregate outputs



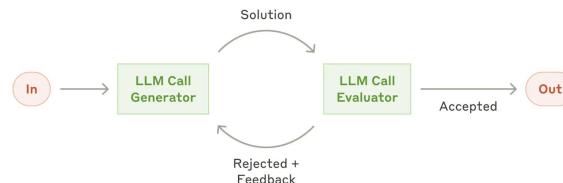
Orchestration - Worker Workflow

Central LLM orchestrates task decomposition. Specialized workers handle specific subtasks



Evaluator - Optimizer Workflow

Dual-LLM process - one LLM generates responses while another provides evaluation and feedback in an iterative loop



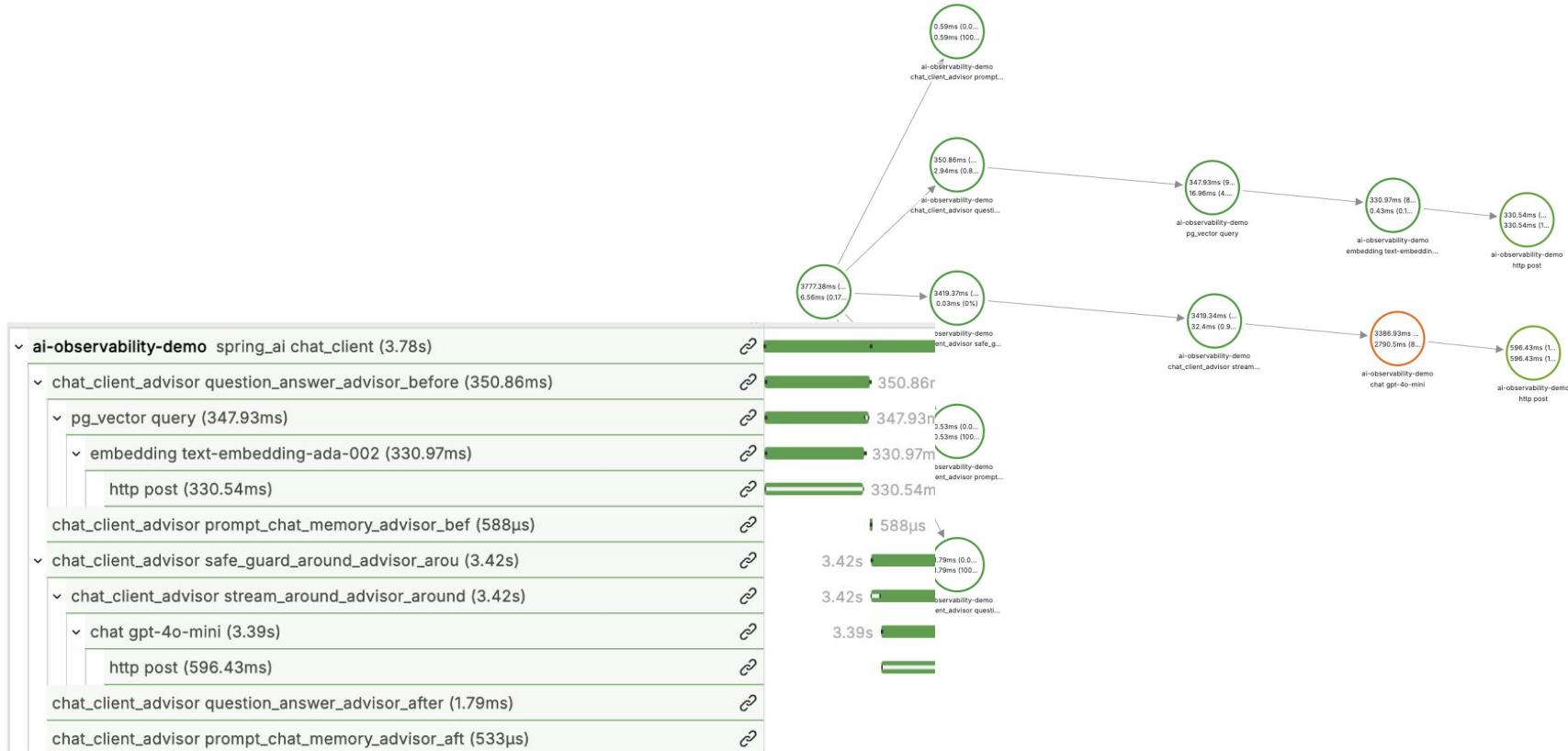
Building effective agents: <https://www.anthropic.com/engineering/building-effective-agents>

Building Effective Agents with Spring AI (Part 1): <https://spring.io/blog/2025/01/21/spring-ai-agentic-patterns>

Github Repo: <https://github.com/spring-projects/spring-ai-examples/tree/main/agentic-patterns>



Spring AI Observability



Resources



Docs

- <https://docs.spring.io/spring-ai/reference/index.html>
- <https://docs.spring.io/spring-ai/reference/api/mcp/mcp-overview.html>
- <https://modelcontextprotocol.io/sdk/java/mcp-overview>
- <https://spring.io/blog/2025/01/21/spring-ai-agentic-patterns>

Demos

- <https://github.com/spring-projects/spring-ai-examples>
- <https://github.com/tzolov/talent-arena-demo>

Spring AI (overview) Talk - 2024

- <https://www.youtube.com/watch?v=vuhMti8B5H0>

Thank You

Christian Tzolov  Spring AI
tweeter: @christtzolov , linkedin: @tzolov