# MCP: using Java and Quarkus to bridge LLMs with your applications and data

Horacio Gonzalez

2025-01-30

@ Lost In Brittany

clever cloud

SVQJUG
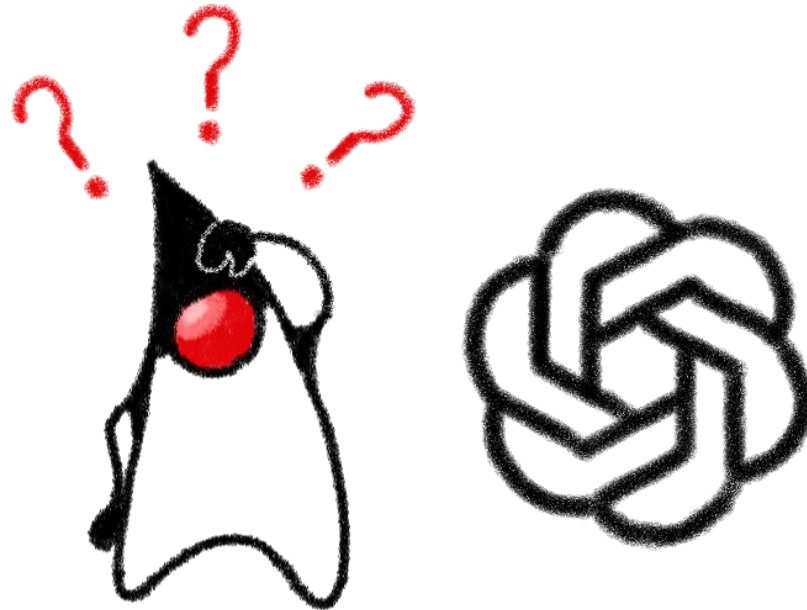
# Summary

clever cloud

@LostInBrittany

# Introduction

**LLMs are changing software development—
how can Java developers take full advantage?**

# Horacio Gonzalez

## @LostInBrittany
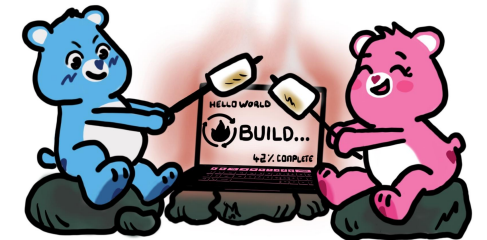
Spaniard Lost in Brittany

Head of DevRel

clever cloud

Finist Devs

clever cloud
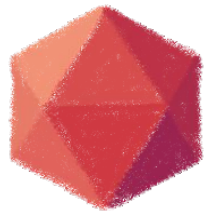
@LostInBrittany

# Clever Cloud



clever cloud

From Code to Product

Our mission: give more **speed** to your **teams**
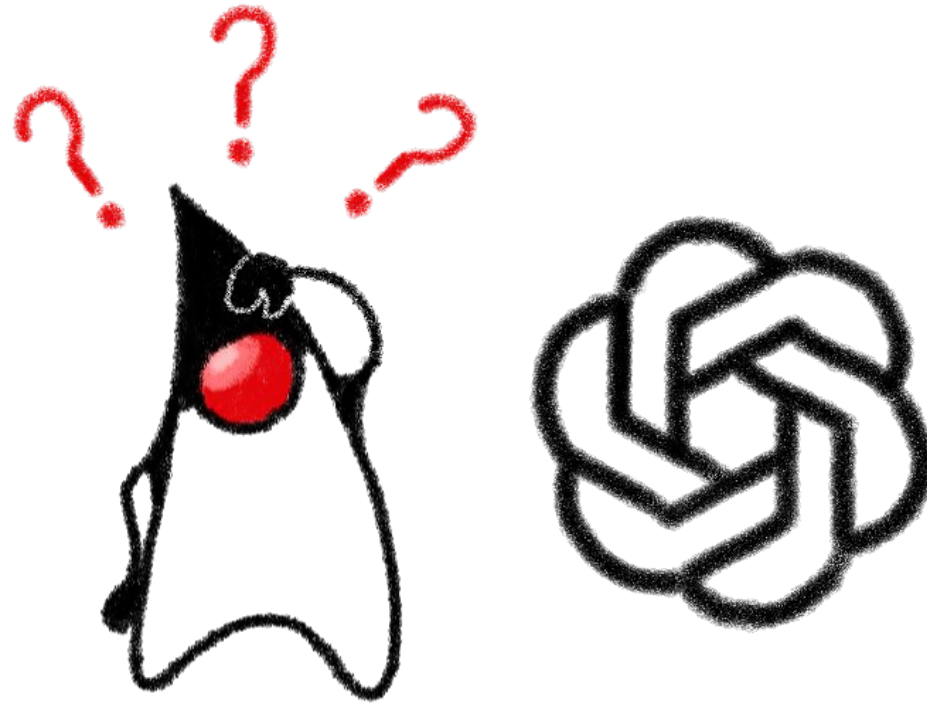and better **quality** to your **projects**

clever cloud
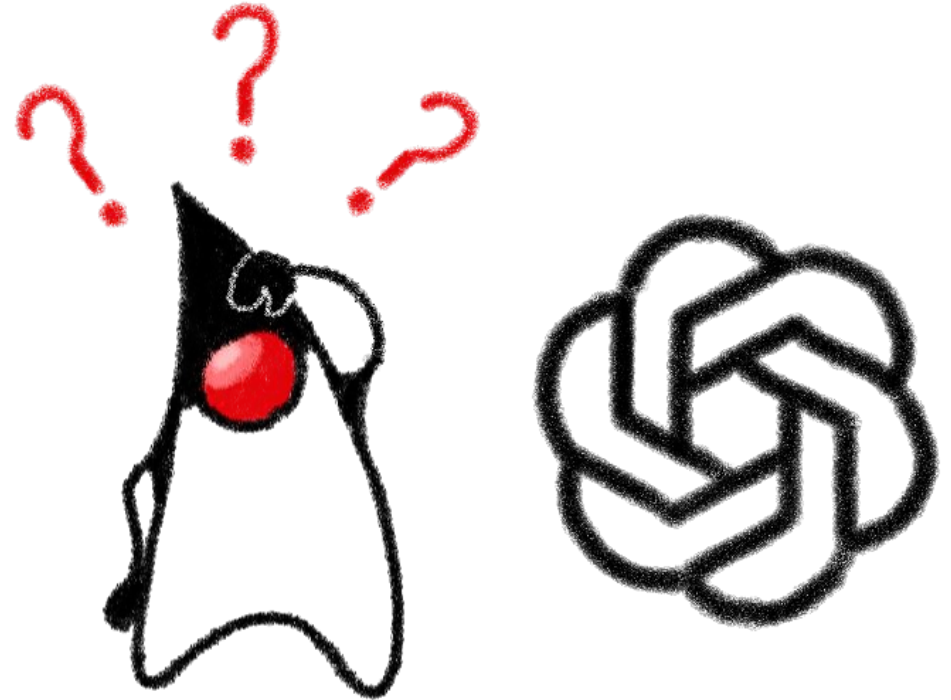
@LostInBrittany

# Why are we talking about this?

LLMs are changing development,
but most Java apps don't fully leverage them

# How do you use LLMs for your dev job?

1. Who here has already used LLM?

2. Who here has already used LLM professionally?

3. Who here has already used LLM with code?

4. Who here has already used LLMs in a Java app?

# How LLMs are changing dev jobs

## The 70% problem: Hard truths about AI-assisted coding

A field guide and why we need to rethink our expectations

**ADDY OSMANI**
DEC 04, 2024

♡ 800     💬 44     🔁 121     Share

After spending the last few years embedded in AI-assisted development, I've noticed a fascinating pattern. While engineers report being dramatically more productive with AI, the actual software we use daily doesn't seem like it's getting noticeably better. What's going on here?
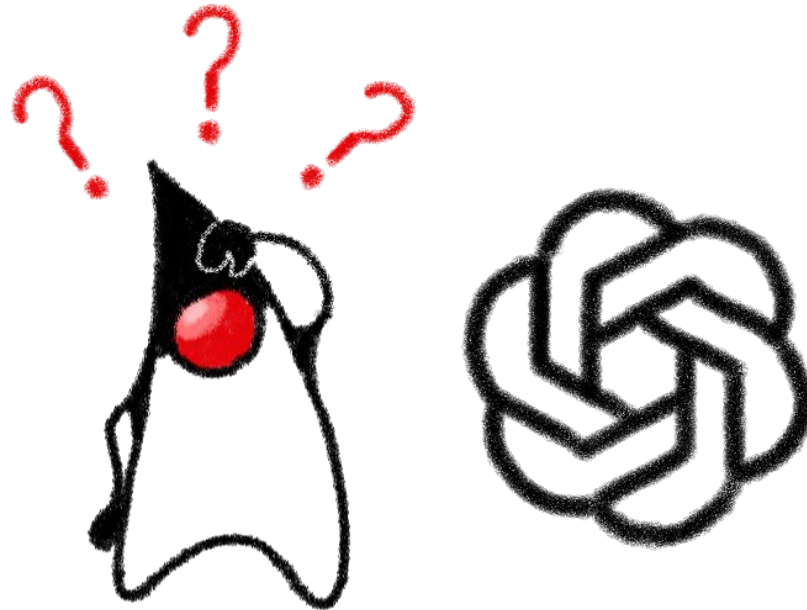
I think I know why, and the answer reveals some fundamental truths about software development that we need to reckon with. Let me share what I've learned.

clever cloud

@LostInBrittany

# Understanding LLMs
# & Their Landscape

**Closed-source, open-source, local—
choosing the right LLM for your Java application.**

# LLMs come in different flavors

**Not all LLMs are created equal**

They have different trade-offs
in capabilities, accessibility, and control

Choosing the right one depends on
your use case, security needs, and infrastructure.

clever cloud

@LostInBrittany

# Closed-source LLMs (Cloud-based APIs)

📌 **Examples**

- OpenAI (ChatGPT), Anthropic (Claude), Google (Gemini), Microsoft (Copilot)

✅ **Advantages**:

- Powerful and well-trained (best models available)
- Easy to use via APIs
- Regularly updated & improved

❌ **Challenges**:

- Black box (you don't control how they work)
- Expensive (API calls can add up quickly)
- Data privacy concerns (sending requests to external servers)

💡 **When to use?**

- If you need the most advanced models and don't mind API costs or external dependencies.

clever cloud

@LostInBrittany

# Open-source LLMs (Self- or cloud-hosted)

📌 **Examples**

- Meta's Llama 3, Mistral, Google's Gemma, Alibaba's Qwen

✅ **Advantages**:

- Greater control (you know exactly how the model works)
- Can be fine-tuned for specific needs
- No external API costs

❌ **Challenges**:

- Requires more setup (you have to run the model yourself)
- May not be as powerful as the latest closed models
- Needs infrastructure (e.g., GPUs for hosting)

💡 **When to use?**

- If you need control over the model & lower costs but are okay with slightly weaker performance

clever cloud

@LostInBrittany

# Local models (on your machine or server)

📌 **Examples**

- Ollama, GGUF-based models (e.g., Llama, Mistral, Mixtral)

✅ **Advantages**:

- Works offline (great for security-sensitive applications)
- No API costs (completely free to use once set up)
- Low latency (responses are instant if hardware is good)

❌ **Challenges**:

- Limited by your hardware (needs a strong CPU/GPU)
- Not always as capable as cloud-hosted models
- Setup complexity (installing and optimizing models)

💡 **When to use?**

- If you need privacy and control, and you have the hardware to run an LLM efficiently

clever cloud

@LostInBrittany

# Choosing the Right Model for your Apps

- Cloud APIs
  - Great for rapid development, but costly and not always secure
- Self-hosted open models
  - Best balance for long-term control and scalability
- Local models
  - Best for privacy-sensitive applications
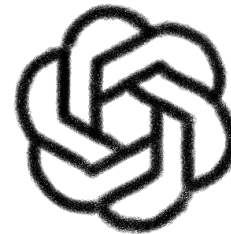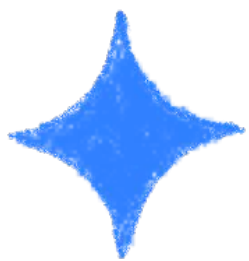
clever cloud

@LostInBrittany

# Making Java Applications LLM-Aware

## LangChain4j simplifies LLM integration–
## let's see how it works!



clever cloud

@LostInBrittany

# Two ways to integrate LLMs with Java

1. Java applications using LLMs
   - Using LLMs as assistants, API consumers, or reasoning engines
   - Easier and works well for code generation, chatbots, and AI assistants

2. LLMs using Java applications
   - Exposing Java functions, APIs, and databases to LLMs for tool execution
   - More powerful but requires tool calling and execution control.

clever cloud

@LostInBrittany

# What is LangChain4j?

LangChain4j is a Java library that simplifies LLM integration

It's the Java equivalent of LangChain (Python/JS)

✅ Prompt management – Structured input for LLMs.

✅ Memory – Keeping conversational state.

✅ Agents – Allowing LLMs to decide which tools to use.

✅ Tool calling – Exposing functions to LLMs.

clever cloud

@LostInBrittany

# LangChain4j Hello Sevilla JUG

```
HelloLangChain4j01.java

///usr/bin/env jbang "$0" "$@" ; exit $?
//DEPS dev.LangChain4j:LangChain4j-open-ai:1.0.0-beta1
import dev.LangChain4j.model.openai.OpenAiChatModel;
import static dev.LangChain4j.model.openai.OpenAiChatModelName.GPT_4_O_MINI;


public class LangChain4jHelloMadridJug {
    public static void main(String... args) {
        String apiKey = System.getenv("OPENAI_API_KEY");
        OpenAiChatModel model = OpenAiChatModel.builder()
            .apiKey(apiKey).modelName(GPT_4_O_MINI).build();
        String response = model.chat(
            "I am doing a demo at Sevilla JUG. Can you introduce yourself and say hello?");
         System.out.println(response);
    }
}
```

With Jbang, of course

clever cloud

@LostInBrittany

# We want to see it working!



**https://github.com/LostInBrittany/langchain4j-for-tools-and-mcp-demo/**
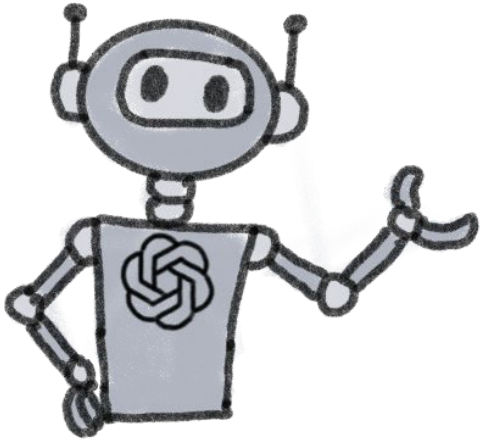
# What are tools in LLMs?

Tools allow LLMs to do **more than just generate text**. They can interact with APIs, databases, and execute functions.
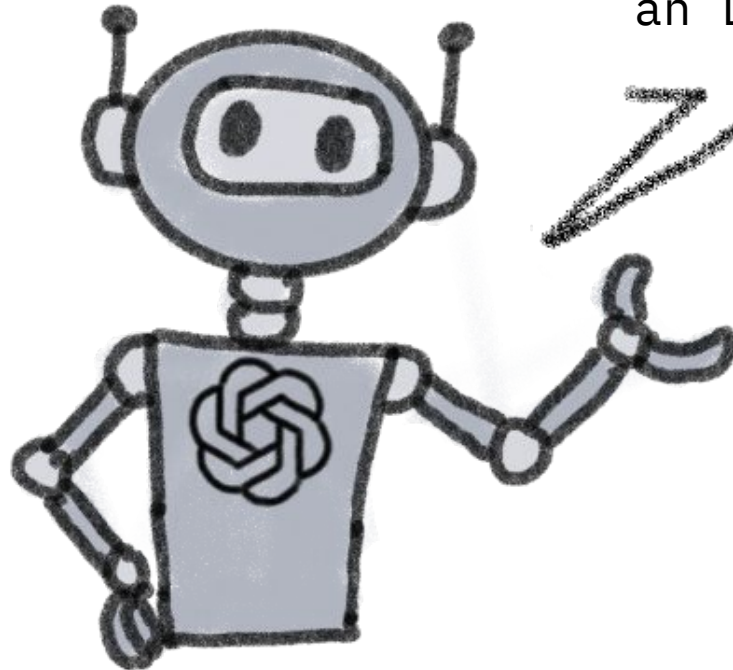
Instead of returning "I don't know," an LLM can call a tool **to fetch real data**.

clever cloud

@LostInBrittany

# An LLM without Tools can't answer this



What's the weather like in Sevilla today?

```
I'm unable to provide real-time
information or current weather updates.
```

# Example of how tool calling works

Weather API

Madrid (ES)

**1.8°C**

☀️

3.8 km/h wind

```
getWeather("Sevilla (ES)")
```

```
{"weather":"sunny",
 "temperature":"1.8ºC"}
```
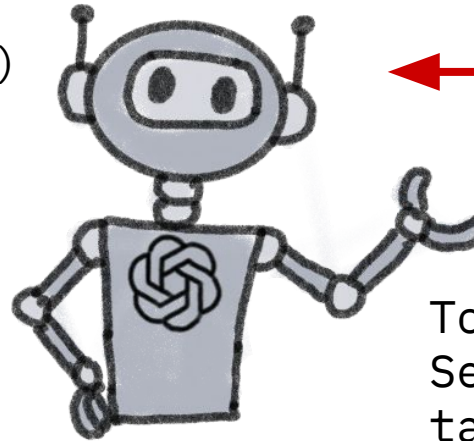
What's the weather like in Sevilla today?

Today it is sunny in Sevilla, but very cold, take a coat.

LLM recognizes it needs an external function and calls it.
It integrates the result into a natural-language response.

clever cloud

@LostInBrittany

# Why this matters?

- Moves LLMs from static text generation
  - dynamic system components
- Increases accuracy & real-world usability
- Allows developers to control what the LLM can access
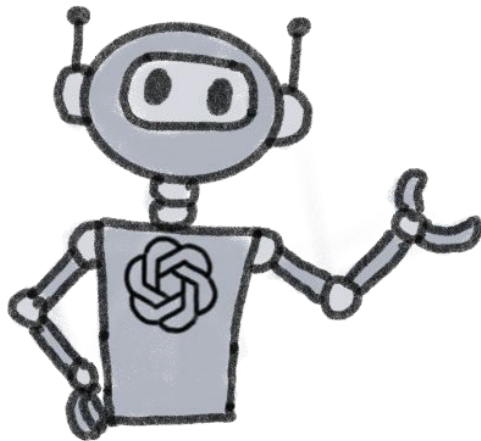
What's the weather like
in Madrid today?

```
Today it is sunny in Madrid,
but very cold, take a coat.
```

clever cloud

@LostInBrittany

# Using LangChain4j to Define LLM Tools

```java
// LyingWeatherTool.java

//DEPS dev.langchain4j:langchain4j:1.0.0-beta1


import dev.langchain4j.agent.tool.Tool;


public class LyingWeatherTool{
 @Tool("A tool to get the current weather in a city")
 public static String getWeather(String city) {
    return "The weather in " + city + " is sunny and hot.";
  }
}
```

## LangChain4j gives as a Tool framework

clever cloud

@LostInBrittany

# Using LangChain4j to Define LLM Tools

LyingWeatherTool.java

```java
[...]
    Assistant assistant = AiServices.builder(Assistant.class)
        .chatLanguageModel(model).chatMemory(chatMemory)
        .tools(new LyingWeatherTool()).build();
    System.out.println("-------------------------------------");
    String question = "What will the weather be like in Madrid tomorrow?";
    String response = assistant.chat(question);
    System.out.println(response);
    System.out.println("-------------------------------------");
[...]
```

And an AI Service abstraction

clever cloud

@LostInBrittany

# Let's see the code!



https://github.com/LostInBrittany/langchain4j-for-tools-and-mcp-demo/

# Model Context Protocol (MCP): The missing link

## MCP bridges LLMs with your applications, enabling controlled, real-world interactions

@LostInBrittany

# Why Do We Need MCP?



Function calling is powerful, why do I need another concept?

LLM function calling is useful, but lacks structure

clever cloud

@LostInBrittany

# Why Do We Need MCP?

## Problem

- LLMs **don't automatically know** what functions exist.

- **No standard way** to expose an application's capabilities.

- **Hard to control** security and execution flow.

## Solution: MCP

- MCP **defines a standard way** to describe and expose functionalities.
- **Applications stay in control** over what LLMs can do.

Function calling is powerful, why do I need another concept?

clever cloud

@LostInBrittany

# Model Context Protocol



De facto standard for exposing system capabilities to LLMs

https://modelcontextprotocol.io/

# How MCP works

- Applications define an MCP manifest (structured JSON).

- The manifest describes available functions, input/output formats, and security policies.

- LLMs can discover and request function execution safely.

Weather
MCP Server

**Madrid (ES)**

**1.8°C**

☀️

3.8 km/h wind

clever cloud

@LostInBrittany
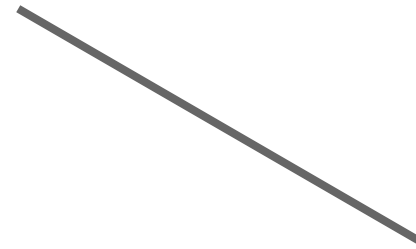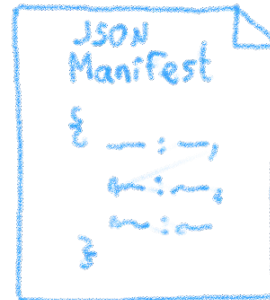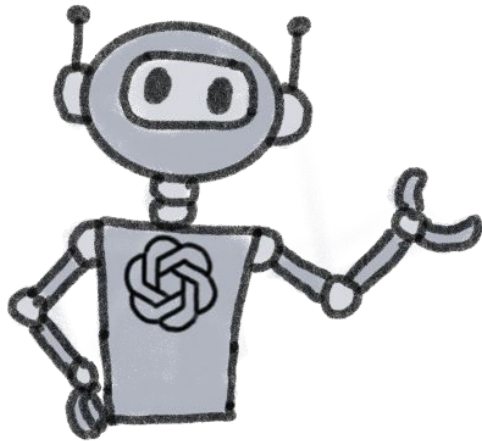
# MCP is provider-agnostic

Works with any LLM provider

Ensures standardized function exposure across platforms

clever cloud

@LostInBrittany

# Understanding the MCP Manifest

```
mcp-manifest.json

{ "functions": [
   {
     "name": "getWeather",
     "description": "Fetches the current weather for a city.",
      "parameters": {
         "city": { "type": "string", "required": true },
         "countryCode": { "type": "string", "required": true }
      }
   }
 ]}
```

- Lists available tools
- Describes expected inputs/outputs
- Defines execution policies

clever cloud

@LostInBrittany

# That's all, folks!

## Thank you all!