MCP server deep dive with tools

Overview

In this lesson, we took a deep dive into the **MCP server** and explored how it connects with an **MCP client** and interacts with various tools. Here's a rundown of what we covered:

- **MCP Client to Server Connection**: We looked at how an MCP client connects one-on-one with an MCP server.
- **Tool Integration**: We discussed how the server has access to various tools, such as **A1**, **A2**, and **A3**.

Tool Definition in an MCP Server

When we define a tool in an MCP server, it's crucial to specify the following parameters:

- **Name**: This is essentially the title of the tool or function, like "get weather."
- **Description**: This details what the tool does, including the expected arguments.
- **Input Schema**: A dictionary defining the arguments a tool or function accepts, e.g., `location string`.

These parameters ensure the MCP client and host can effectively decide whether to call a specific tool.

Static and Implicit Typing

- **Static Typing**: Parameters can be set explicitly when the tool is created.
- **Implicit Typing**:
  - The function name can be used as the name parameter.
  - The function's doc string serves as the description, making it dynamic and context-aware, which is quite handy!

Function Logic and Usage

For each tool in the MCP server, we focus on designing the function logic. The purpose of these functions doesn't matter much—whether they access an API or perform local computations, what's important is:

- **Execution**: The tool carries out its defined task.
- **Parameter Communication**: Tools communicate their name, description, and input schema, allowing clients and hosts to understand when and how to interact with them.

Extras: Prompts and Resources

Although not the central focus of this session, it's worth noting:

- An MCP server can also manage **prompts** and **resources**, which expand its capabilities beyond tools.

This is just a high-level overview, and we'll have multiple opportunities to build and refine these server components as we go along. Stay tuned for more practical insights!