Create a simple MCP Server 2

Overview

In this lesson, we embarked on deploying an MCP server and connecting it with an MCP client or host like Claude. Here’s how we navigated through the process:

- **Deploying the MCP Server**: We've taken our pre-built MCP server and aimed to ensure its availability in various clients.
- **Connecting with MCP Client/Host**: Whether it's Claude, VS Code, or others, we highlighted how these can be linked.

Initial Deployment

- **Running the Server**: We began by launching our server using commands like `uv run weather.py` or `python weather.py`.
- **Understanding Server Behavior**: Realized that the server needs an MCP client to interact with, otherwise, it remains idle.

Testing with MCP Inspector

- **Inspector Introduction**: The MCP inspector is critical for testing servers.
- **Activating the Inspector**:
  - Accessed using `mcp dev weather.py`.
  - This simulated a client connection, verifying the server’s response.
- **Connection Confirmation**: Upon connecting, we could see available tools like `get weather`.

Engaging with Tools

- **Tool Interaction**:
  - Used inspector to list tools and run them.
  - Successfully tested fetching weather details for a location like California.

Preparation for Claude

- **Configuration File Setup**: MCP servers connect via a `server.json` file.
- **Updating Commands**:
  - Adjusted command from `npx` to `uv run weather.py`.
  - Ensured correct directory paths and syntax, especially on Windows, using escaped backslashes.

Handling Directory Paths

- **Absolute Path Requirement**: Needed to specify the directory where `weather.py` is held.
- **Windows Specific Instructions**: Adapted folder paths using double backward slashes to escape JSON characters.

Finalizing and Testing in Claude

- **Restarting Claude**: Modified settings required us to restart Claude.

- **Verification**:
    - Confirmed by seeing `get weather` in Claude’s tools list.
    - Successfully executed a query to check weather in California.

Troubleshooting Tips

- **Common Issues**: Highlighted potential pitfalls and provided insights on resolving these.
- **Further Testing**: Advised leveraging the MCP inspector for problem-solving before engaging real clients.

By the end of this lesson, we effectively bridged our MCP server with a client, ensuring seamless interaction and tool usage. Our journey included understanding the deployment intricacies, testing methodologies, and client configuration nuances. Let's keep practicing and troubleshooting to master these deployments!