Create a simple MCP Server 1

In this lesson, we embarked on a journey to create our very own **MCP server**. Think of it as the "hello world" of MCP servers, providing us with a foundational understanding of how these servers operate!

Prerequisites

Before diving into the server creation, we ensured a few things were in place:

- **Dependencies**: Make sure you have all dependencies installed, like **Node**, **Python**, and especially **UV**. Check these using your terminal.
- **IDE Preference**: You can code in your choice of IDE, but we used **VS Code**.
- **Folder Setup**: Avoid using a synced folder (like OneDrive or Dropbox) for your code to prevent annoying errors.

Environment Setup

Here's what we did to set up our environment:

- **Project Initialization**: Start a new project with `uv init`.
- **Virtual Environment**: Create a virtual environment using `uv venv`; this isolates our project dependencies.
- **Activate Environment**: Enter the virtual environment to ensure all dependencies are properly managed.
- **Install Dependencies**: Use `uv add` to install the necessary package like `MCP CLI`.

Creating the Server

We then started getting into the actual coding:

- **New File Creation**: We created a new Python file, `weather.py`, for our code execution.
- **Library Imports**: Imported necessary libraries, primarily `Fast MCP` and `Server`, choosing `Fast MCP` for its ease and abstraction.

MCP Object and Tool Creation

Here's how we structured our server:

- **MCP Object**: Created a new MCP object with `Fast MCP` and gave it a significant name.
- **Function Definition**: Defined a function `getWeather` that takes a location and always returns "the weather is hot and dry"—our simple example for a tool.
- **Decorator Use**: Used `@MCP.tool` to elevate our function to a tool, making it accessible via the MCP framework.

Running the Server

The final steps included:

- **Server Execution**: Used a Python script convention to run our server using `mcp.run`.

- **Integration**: Prepared to add it to Anthropic and future MCP clients, which we'll explore in upcoming lessons!

Final Thoughts

- **Easy vs. Challenging**: Depending on your experience, this might have felt easy or confusing. If you found it easy, feel free to speed through the content. If it was challenging, don't hesitate to revisit the video or reach out to the community with questions.
- **Deep Dive Ahead**: We'll cover more intricate details in a deep dive section coming soon.

That's it for this lesson! We laid the groundwork for MCP server creation, setting the stage for more complex implementations.