FastMCP vs. lowlevel server

Introduction

In this lesson, we dove into the key distinctions between the `FastMCP` class and the traditional `low level server` class when developing MCP servers. Here's a breakdown of what we covered:

Understanding MCP Server Development

- **Two Main Abstractions**:
  - We introduced the concept of using either `FastMCP` or the `low level server` (previously known just as `server`) when building MCP servers.
  - The **low level server** offers a more complex and detailed approach, similar to programming with C or C++.
  - **FastMCP** acts as a simpler, user-friendly abstraction, akin to Python's relationship to C/C++.

Why Choose FastMCP?

- **Simplicity**:
  - **FastMCP** abstracts much of the repetitive and detailed coding necessary when using the low level approach.
  - By wrapping the original server implementation, **FastMCP** minimizes boilerplate code and simplifies the process significantly.
- **Community and Support**:
  - **FastMCP** is actively developed and supported by the open-source community, making it the preferred choice for many developers.
  - It supports most, if not all, necessary features, making it a robust choice for MCP development.

Practical Differences

- **Comparison**:
  - With the **low level server**, we must:
    - Define function logic.
    - Set up tool calling capabilities.
    - Configure a tool library.
    - Start the server.
  - With **FastMCP**, we simply:
    - Instantiate an object and create functions.
    - Decorate the function with `@MCP.tool`, automating backend processes.
    - Let the tools, input schemas, and logic infer from the decorated function.

Conclusion

- **Recommendation**:

    - Always opt for **FastMCP** when possible, as it handles over 95% of MCP server tasks without the need for detailed low level configurations.
    - For those rare cases where **FastMCP** falls short, the option to revert to the low level server remains available.
- **Focus of the Course**:
    - We will primarily concentrate on **FastMCP**, for its ease and efficiency, throughout our course on building MCP servers and clients.

This approach ensures that we spend more time innovating and less time wrestling with complex code structures.