Using MCP Servers to make API calls

Overview

In this lesson, we explored the essential functionality of using **MCP servers** to make **API calls**. This may sound straightforward, but it's crucial because most MCP servers primarily function as **API wrappers**. Here's a breakdown of what we covered:

- **API Wrappers**: About 98% of MCP servers basically act as wrappers for API calls to different services.
- **Local Machine Execution**: When a user installs an MCP server, the API calls are executed from their local machine.

Key Concepts

- **API Calls from Local Machine**: With the **STDIO transport**, API calls are made from the local machine where the MCP server is installed. This is important for understanding how and where data is processed.
- **Streamable HTTP**: When using streamable HTTP, the server hosting the MCP server manages the API calls.

Step-by-Step Example

Creating a New MCP Server:

- We began by creating a new file called `crypto.py` to manage our cryptocurrency price API requests.
- Cloned existing server architecture and made necessary modifications.

Function Definition:

- Defined a function `get_cryptocurrency_price` that takes a cryptocurrency symbol (e.g., "Bitcoin") and returns its current price in USD.

API Integration:

- Utilized the CoinGecko API: `api.coingecko.com/v3/simple/price`.
- Parameters included:
  - `id`: the cryptocurrency symbol.
  - `vs_currency`: the currency in which to return the price, set as USD.

Error Handling:

- Added error handling to ensure the server returns a user-friendly error message rather than crashing.

Troubleshooting and Testing

- **Library Installation**: Installed the `requests` library to facilitate API calls.
- **MCP Development**: Tested functionality using `FMCP dev crypto.py` to ensure everything was working correctly.
- **Common Issues**:
  - Missing library: Ensured correct environment setup and installed any missing packages.

Deployment

- **Installation**: Deployed the server using `MCP install crypto`.
- **Verification**: Checked the installation within our development environment and confirmed successful setup.

Demonstration

- **API Requests**: Demonstrated API calls by querying prices for Bitcoin, Ethereum, and Litecoin.
- **Dynamic Comparisons**: Showcased the ability to compare prices between cryptocurrencies, utilizing the server’s reasoning to return meaningful results.

Additional Notes

- **Flexibility**: The same API call setup could be adapted for various services, including Google's document APIs or Microsoft’s Graph API.

Through this lesson, we've built a basic yet powerful MCP server for making API calls, providing a solid foundation for more advanced applications.