MCP Server - Overview

Wednesday, 21 May 2025 11:17 PM

MCP servers are simply wrappers or interfaces that federate access to various systems and tools, making them accessible to AI applications in a standardized way.

And they do this by exposing functionality through three primary interfaces.

1. The first one is tools, so model controlled functions that the AI can invoke when needed.
    - Get weather, get forecast get alerts. And they all boil down to functions.

   So the AI system and here we saw cursor and we saw cloud desktop or any other application it can decide when to use those tools based on the context.
2. The second thing that MCP servers can expose are resources. And resources are application controlled data that is exposed to the AI system.
    - So it can be static data like PDF documents, like text files, images, JSON, whatever format youwant.
    - But it can also be dynamic. So you simply tell how to get those resources dynamically. And they don't have to be static.
3. Last functionality that MCP servers expose can be prompts.
    - prompts are user controlled templates for common interactions, and they are predefined templates that the users can invoke.
    - This usually helps to standardize rice complex interactions.

Two types:

- In local
- In cloud

- Architecture::
    - The model context protocol supposed to standardize how applications provide context to llms
        - So context can be additional information to the prompt.
        - A context can be maybe which tool to invoke.
        - And a context can even be the prompt itself.
    - Advantages:
        - We have a huge list of integrations and tools and data sources we can plug and play to our LMS.
        - the core of AI application development is the fact that we are not coupled to any LLM vendor or to any AI application builder.

          NOTE :: Langchain does these 2 in a different way
- **MCP Hosts**:
    - for example Cloud desktop an IDE like cursor or windsurf or any other specialized AI application

- **MCP Server**:
    - The servers are the component that will expose those resources, those tools, those prompts.
    - And they will be the proxy, the gateway however you want to call it that will expose that functionality.
    - And in order to expose that functionality. It needs to have certain methods and certain functions.
    - For example **least prompts, get prompt list tools, call tool list resource templates, and progress notification**
    - The point here is once we implement an MCP server, then we can basically connect it to any MCP host that is supporting the protocol.
    - So this means we can write functionality once and then plug it to, for example, many, many MCP hosts.

- **Mcp client**:
    - It sits inside MCP Hosts and it's going to be able to interact and to talk with the MCP servers and they're going to do this via the MCP protocol.
    - 1 to 1 connection between an MCP client and an MCP server. So you can't have an MCP client that talks to multiple MCP servers.
    - MCP hosts, if you want to connect them with multiple MCP servers, then we'll need to have multiple MCP clients inside them.
- **Composability**:
    - any application or agent can be both an MCP client and server. And this really enables us to have this kind of multi-layered agentic application that allows us to have specialized agents that focus on particular tasks.

- **Registry and discovery**:
    - Similar to Maven there will be a central registry API for discovering MCP servers.
    - you can list your MCP server that you implement and let other people use it.
    - we have verification of official servers to prevent from attackers

- MCP Inspectors:
    - It is an interactive dev tool designed for testing and debugging MCP servers and it allows developers to inspect and interact with the MCP server without requiring any installation.
    - it can be run locally from npx, And it has a bunch of very useful features that are going to help us.
        - npx @modalcontextprotocol/inspector <command>
    - Uses:
        - we can connect into an MCP server.
        - We can check out the resources tab, which will list all the available resources and show the metadata and allow the content inspection, which is very convenient.
        - We have a section and a tab on prompts which will display the prompt. Templates, show the prompt arguments, and it will even allow us to enable testing with custom inputs to the prompt templates, and
        - it also has a tools tab which will list all the available tools, all their schemas, and will even allow us to test those tools with custom inputs.
        - We also have a notifications pane which will present logs and notifications from the server.
- **LLM.txt**:
    - llm.txt is a machine-readable markdown file placed in a website's root directory that helps LLMs and AI agents (like ChatGPT or Gemini) better understand and extract content by summarizing key pages and their purposes.
    - Benefits include improved AI accuracy, enhanced content discoverability, and better contextual understanding of the website's structure.

        - Types:
            - LLM.txt
            - LLM-full.txt

    llm.txt:
        - **Purpose**: Concise project overview for AI assistants
        - **Content**: Essential information only (project description, key files, basic structure)
        - **When to use**:
            - Quick AI queries about your project
            - When context window is limited
            - For general project understanding
            - Initial AI conversations about codebase
        - **Size**: Typically under 1-2KB
        - **Focus**: High-level overview, core concepts

    llm_full.txt:
        - **Purpose**: Comprehensive project documentation for AI
        - **Content**: Detailed information (full API docs, examples, configuration details, troubleshooting)
        - **When to use**:
            - Complex development tasks
            - Detailed code generation/refactoring
            - When AI needs complete context
            - Advanced debugging or architecture discussions
        - **Size**: Can be several KB to MB

- **Focus**: Complete technical details, implementation specifics