MCP2

We are asking AI to use different tools from different MCP servers. If we ask AI to create a new file, it will do it but if I ask AI to push to GitHub it wont be able to do it. If I ask AI to pick up Jira ticket, it wont be able to do it because they are all on different servers. You are not asking IDE to do it, you are basically asking LLM to do it. How AI will have access to all these services? All these services should provide some way to interact with. LLM basically asks Host (Machine where our code is running) to execute the tool. If you ask AI to talk to GitHub, it will not allow AI, GitHub will allow only the Host to talk to GitHub. I have to find a way to talk to GitHub from my machine. That's where we can use MCP, which is a standard. Say we have 10 tools to talk to and these 10 tools have to talk to each other as well. Imagine how much code we got to write then. If GitHub wants to create their own tool, they can follow MCP standards. We can have an MCP client and say which MCP server you want to connect with "@modelcontextprotocol/server-filesystem". GitHub, Jira all these services will create MCP servers. What you create in your software is the MCP client, which will connect to the server.
MCP (Model Context Protocol) is an open-source standard for connecting AI applications to external systems. External systems is GitHub or Jira. MCP server can be a local server or remote server.

You cannot connect your Host directly to MCP server, what you need in between is MCP client. That means we need a connection between MCP client and MCP server that's where layers come into picture. Transport layer: whenever two machines are connected, for that we need a Transport layer. We cannot use HTTP to connect to the local server, but yes to connect to the remote server, we can use HTTP there. MCP has done some modifications to the Transport layer: if it is local, we are going to use *stdio* and if it is remote, we are going to use *Streamable HTTP transport*. Say for the NewsTool, we are invoking their API using RestTemplate, that means we got to write some code in our machine. "@modelcontextprotocol/server-filesystem" server-filesystem has some tools or methods() executing somewhere else or on a server. How do you call a method() from your machine to the remote server? Thats where we have the concept, *RMI () and RPC (Remote Procedure Call)*. Say you have two different softwares running on the same machine, can you call the function of one software from other software? If it is the same machine, then we have something called as "Native calls". Lets say now the Softwares are on different machines, in that case, we can use something called as "RPC", which has the power to call the function of another computer from one computer. SSE (Server-Side Events) is basically one-sided communication Server to Client. Streamable HTTP supports on both sides.

RPC (Remote Procedure Call)
Machine 1 (function1) ==> Machine 2 (function2). How will you ask Machine 1 to call a function of Machine 2? How M2 knows which function to invoke and what are the parameters? In that case, we got to send some data, JSON. They built something called as "JSON RPC". It's RPC the way you communicate is through JSON.

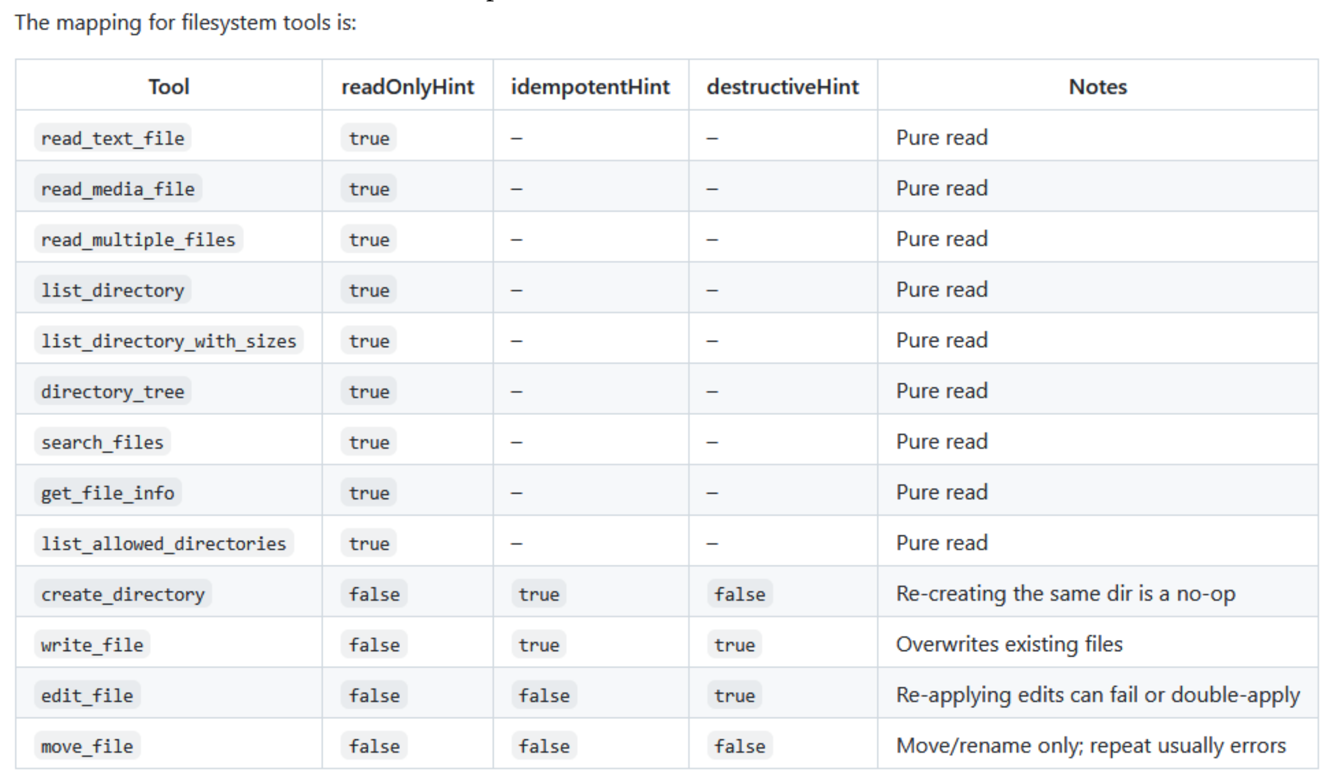
Example of JSON RPC:
this particular request goes to the other machines
{"jsonrpc": "2.0", "method": "subtract", "params": {"subtrahend": 23, "minuend": 42}, "id": 3}
we are calling a method called as "subtract" with parameters {"subtrahend": 23, "minuend": 42}. Every request will have an id. Now this request goes to M2. SpringAI or Langchain will take care of all the requests. This is what we are sending to a Server. Server will process it and give you a response.
Example response:
{"jsonrpc": "2.0", "result": 19, "id": 3}. it returns an ID as well, id:3, which matches id of the request. There are some error codes also. What if there is no function called as "subtract"?
https://www.jsonrpc.org/specification

code	message	meaning
-32700	Parse error	Invalid JSON was received by the server. An error occurred on the server while parsing the JSON text.
-32600	Invalid Request	The JSON sent is not a valid Request object.
-32601	Method not found	The method does not exist / is not available.
-32602	Invalid params	Invalid method parameter(s).
-32603	Internal error	Internal JSON-RPC error.
-32000 to -32099	Server error	Reserved for implementation-defined server-errors.

When we asked AI to create a File or Folder, it was calling function of MCP server: server-filesystem. "@modelcontextprotocol/server-filesystem"

https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem

All these servers are built with the help of tools



The mapping for filesystem tools is:

Tool	readOnlyHint	idempotentHint	destructiveHint	Notes
`read_text_file`	`true`	–	–	Pure read
`read_media_file`	`true`	–	–	Pure read
`read_multiple_files`	`true`	–	–	Pure read
`list_directory`	`true`	–	–	Pure read
`list_directory_with_sizes`	`true`	–	–	Pure read
`directory_tree`	`true`	–	–	Pure read
`search_files`	`true`	–	–	Pure read
`get_file_info`	`true`	–	–	Pure read
`list_allowed_directories`	`true`	–	–	Pure read
`create_directory`	`false`	`true`	`false`	Re-creating the same dir is a no-op
`write_file`	`false`	`true`	`true`	Overwrites existing files
`edit_file`	`false`	`false`	`true`	Re-applying edits can fail or double-apply
`move_file`	`false`	`false`	`false`	Move/rename only; repeat usually errors

Whenever you something, you are executing a tool. In the end, you are calling some function, passing some parameters to do the job for us.

Concept of Lifecycle:
When you send a request, it goes from Client to Server.

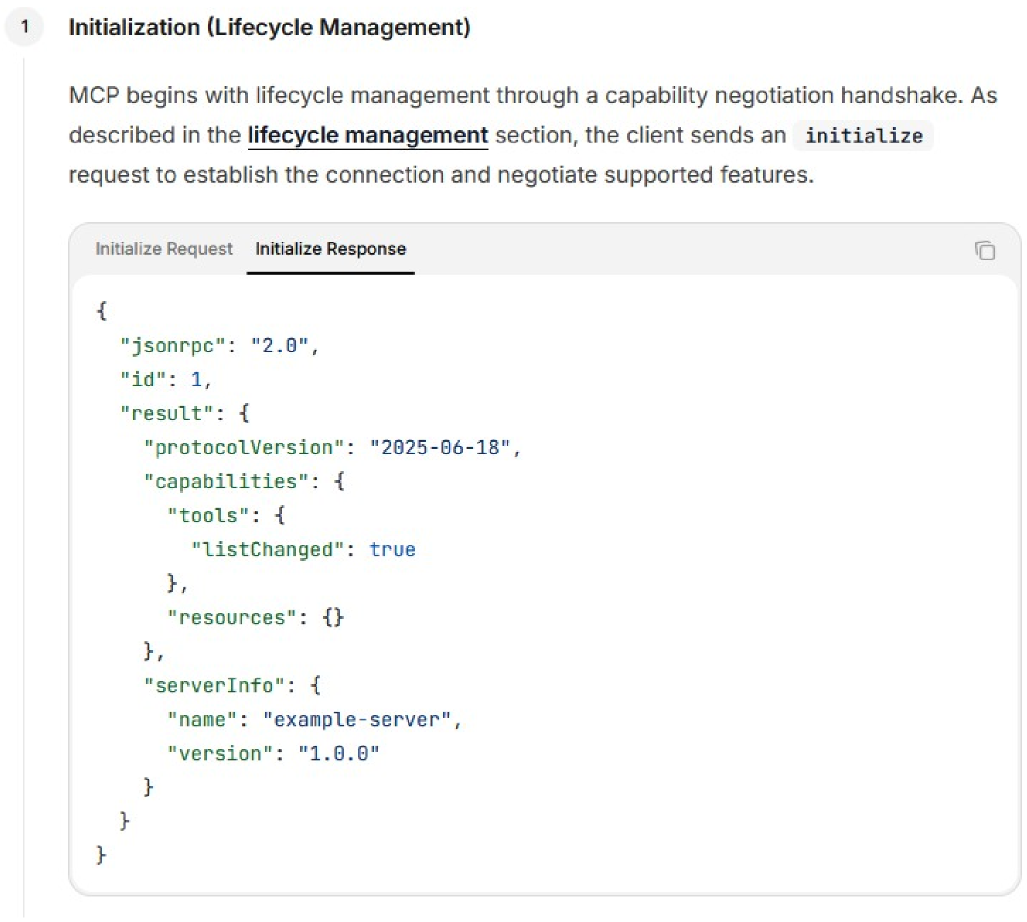
https://modelcontextprotocol.io/docs/learn/architecture

Initialize Request

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "initialize",
  "params": {
    "protocolVersion": "2025-06-18",
    "capabilities": {
      "elicitation": {}
    },
    "clientInfo": {
      "name": "example-client",
      "version": "1.0.0"
    }
  }
}
```

For Initialize Request, MCP server will respond

In the



1 **Initialization (Lifecycle Management)**

MCP begins with lifecycle management through a capability negotiation handshake. As described in the **lifecycle management** section, the client sends an `initialize` request to establish the connection and negotiate supported features.

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "protocolVersion": "2025-06-18",
    "capabilities": {
      "tools": {
        "listChanged": true
      },
      "resources": {}
    },
    "serverInfo": {
      "name": "example-server",
      "version": "1.0.0"
    }
  }
}
```

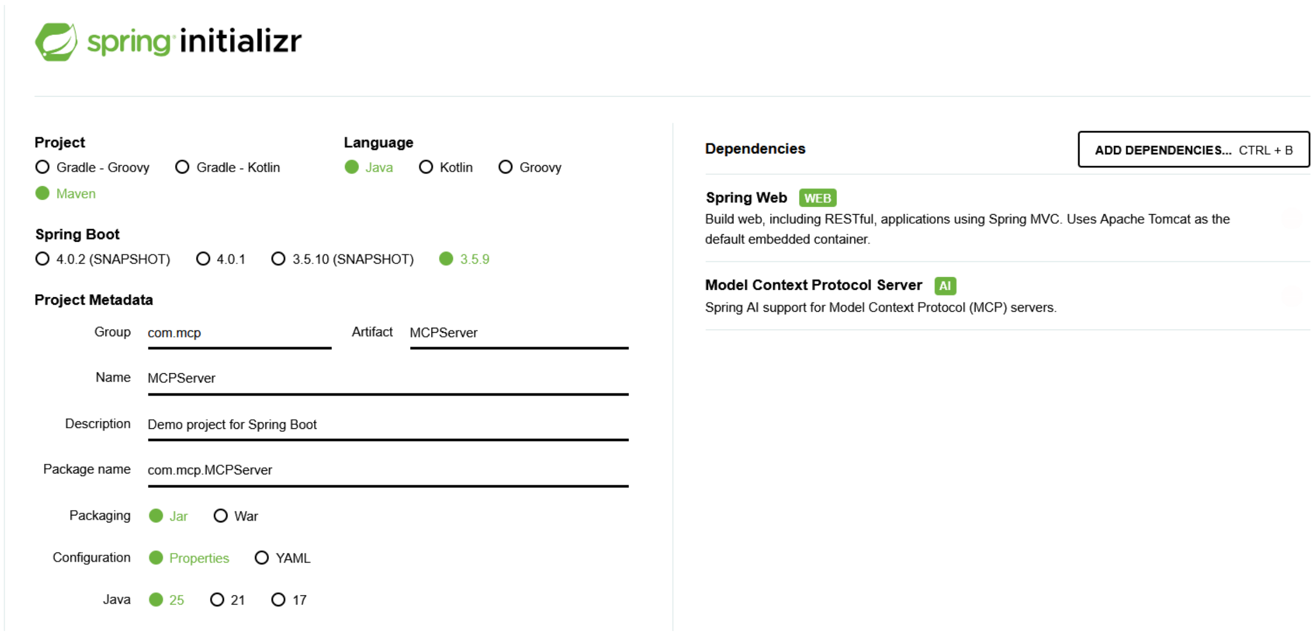
response, it specifies what are the capabilities of MCP server. Three capabilities: 1. Tools, 2. Resources like static content data, PDF, 3. Prompts to create a new repo.

Why do we need Prompts in MCP server?

Say you are talking to GitHub server, what you expect is, when you write a prompt: create a new repository on GitHub. While you provide a prompt, sometimes maybe you are not providing a correct prompt to GitHub server. For incorrect prompts, it can hallucinate or do something wrong. So, GitHub MCP server apart from tools, or resources, it will also provide pre-defined prompts. To perform next operations on GitHub server, your MCP client will use those prompts to send further requests so LLM will not mess up.

Capability Discovery: The capabilities object allows each party to declare what features they support, including which primitives they can handle (*tools, resources, prompts*) and whether they support features like *notifications*. This enables efficient communication by avoiding unsupported operations. Tools, Resources and Prompts are called as Primitives. Why do we need Notifications? Say you got a few resources on some server, you have fetched them. After sometime, you have some updates on the resource. In this case, you can subscribe to a particular resource. Everytime there is a change in the resource, it will give you some Notifications.

<u>Build a new MCP Server</u> (with some tools). We don't need to add OpenAI dependency because we don't need LLM to find DateTime or News etc



Project
Gradle - Groovy Gradle - Kotlin [✓ Maven]

Language
[✓ Java] Kotlin Groovy

Spring Boot
4.0.2 (SNAPSHOT) 4.0.1 3.5.10 (SNAPSHOT) [✓ 3.5.9]

Project Metadata

Group	com.mcp
Artifact	MCPServer
Name	MCPServer
Description	Demo project for Spring Boot
Package name	com.mcp.MCPServer
Packaging	[✓ Jar] War
Configuration	[✓ Properties] YAML
Java	[✓ 25] 21 17

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

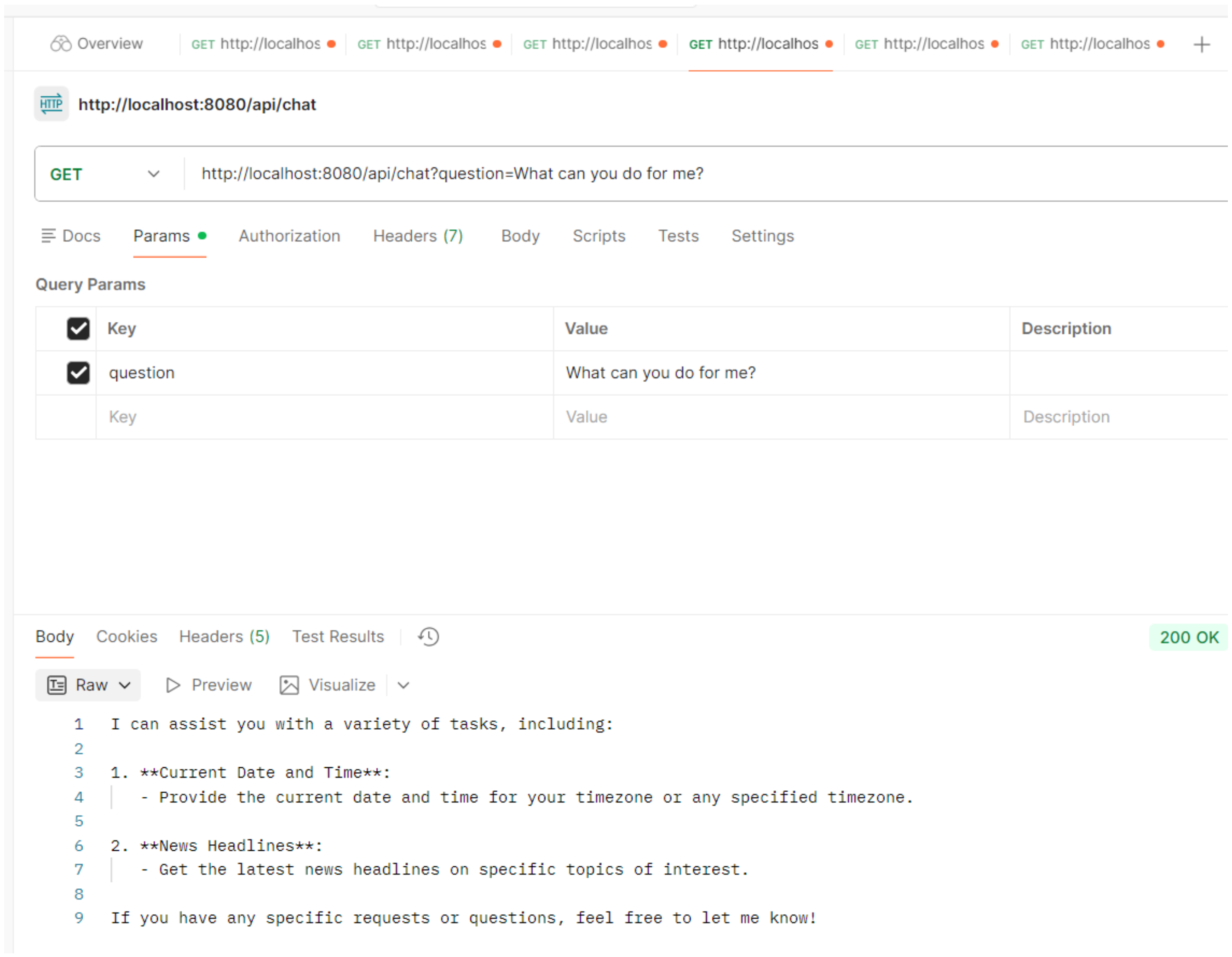
Model Context Protocol Server AI
Spring AI support for Model Context Protocol (MCP) servers.

We are converting our tools into MCP servers

LLM knows about MCP server or what are the tools available only through ToolCallbackProvider toolCallBackProvider

In MCP_Client: we got to inform about the MCP_Server. In this case, MCP_Server even though it is running on the local machine. We cant use stdio since it is running on a separate Tomcat server. It is not considered local.

spring.ai.mcp.client.sse.connections.spring-ai-mcp.url=http://localhost:8282



http://localhost:8080/api/chat

GET http://localhost:8080/api/chat?question=What can you do for me?

Query Params

	Key	Value	Description
✓	question	What can you do for me?	
	Key	Value	Description

```
I can assist you with a variety of tasks, including:

1. **Current Date and Time**:
   - Provide the current date and time for your timezone or any specified timezone.

2. **News Headlines**:
   - Get the latest news headlines on specific topics of interest.

If you have any specific requests or questions, feel free to let me know!
```

Now we are seeing new capabilities
I can assist you with a variety of tasks, including:

1. **Current Date and Time**:
   - Provide the current date and time for your timezone or any specified timezone.

2. **News Headlines**:
   - Get the latest news headlines on specific topics of interest.

If you have any specific requests or questions, feel free to let me know!