



DelphiDay

italian conference

MCP

2025 - ultima frontiera



PAOLO ROSSI
WINTech ITALIA - CTO
SENCHA & EMB. MVP



 blog.paolorossi.net

 paolo@paolorossi.net

 twitter.com/awebguy

 github.com/paolo-rossi

 linkedin.com/in/paolo-rossi-pc



19 Novembre 2025
Padova





GITHUB PROJECTS



github.com/paolo-rossi



Delphi JWT

JSON Web Token Library



WiRL

REST Library for Delphi



Linux Daemon

Real Linux daemons



Delphi Neon

JSON Serialization Library



OpenAPI-Delphi

OpenAPI 3.0 Library



NATS Delphi

NATS Client Library for Delphi



AGENDA



1. MCP: an introduction
2. Why does it matter
3. Introducing MCPConnect
4. The idea behind MCPConnect
5. How to MCPConnect



MCP

1



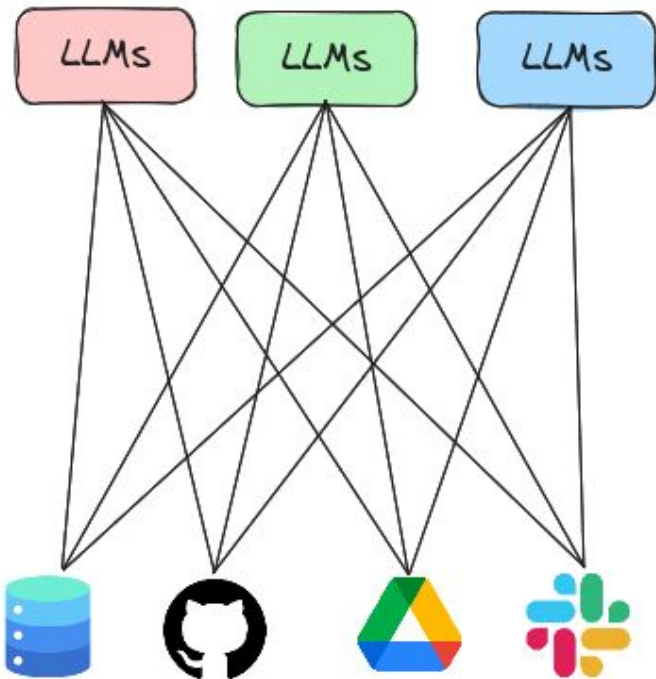
Model Context Protocol

- Open-source standard for connecting AI applications to external systems
- Using MCP, LLM can connect to
 - data sources (e.g. local files, databases)
 - tools (e.g. search engines, calculators)
 - workflows (e.g. specialized prompts)
- This enables LLM to access key information and perform tasks

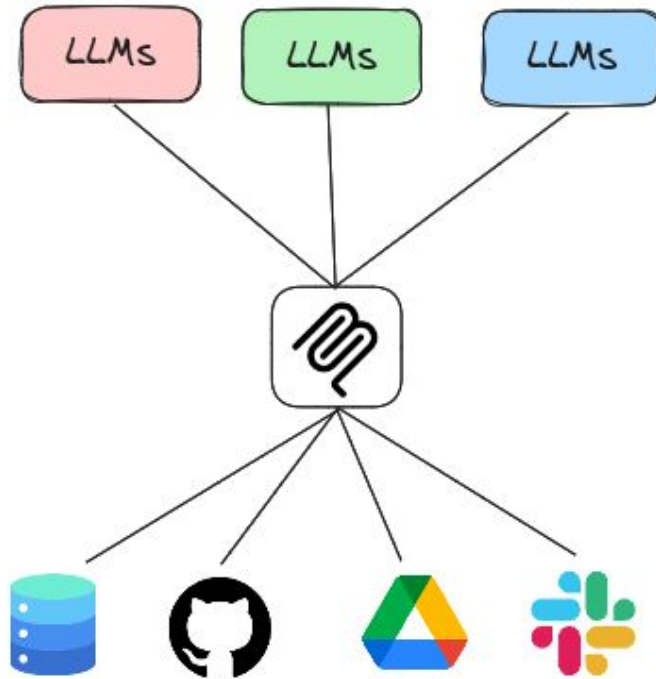


Why MCP?

without MCP



with MCP



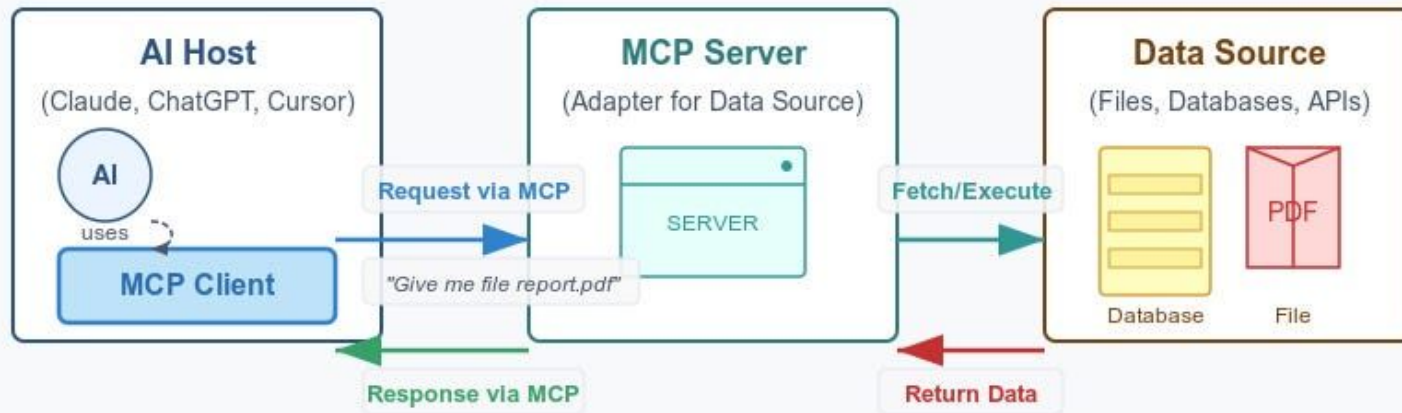


MCP architecture

- MCP Host
 - The AI application that coordinates and manages one or multiple MCP clients
- MCP Client
 - A component that maintains a connection to an MCP server and obtains context from an MCP server for the MCP host to use
- **MCP Server**
 - A program that provides context to MCP clients



Model Context Protocol (MCP) Architecture



Model Context Protocol (MCP) Flow

The MCP Client translates AI requests into the standardized protocol format, communicates with MCP Servers, which then interact with external Data Sources.



Layers

- Data layer
 - Defines the **JSON-RPC** based protocol for client-server communication, including lifecycle management, and core primitives, such as **tools**, **resources**, **prompts** and notifications.
- Transport layer
 - Defines the communication mechanisms and channels that enable data exchange between clients and servers, including transport-specific **connection** establishment, **message** framing, and **authorization**



Data Layer

- The data layer implements a JSON-RPC 2.0 exchange protocol
- Defines the message structure and semantics and includes:
 - **Lifecycle management:** connection initialization, capability negotiation, and connection termination between clients and servers
 - **Server features:** enables servers to provide functionality including tools, resources, and prompts
 - **Client features:** enables servers to ask the client to sample from the host LLM, and log messages to the client
 - **Utility features:** supports additional capabilities like notifications



Basic JSON-RPC example



```
{  
  "jsonrpc": "2.0",  
  "method": "subtract",  
  "params": [42, 23],  
  "id": 1  
}
```



tools/call example

```
{
  "jsonrpc": "2.0",
  "id": 3,
  "method": "tools/call",
  "params": {
    "name": "weather_current",
    "arguments": {
      "location": "San Francisco",
      "units": "imperial"
    }
  }
}
```

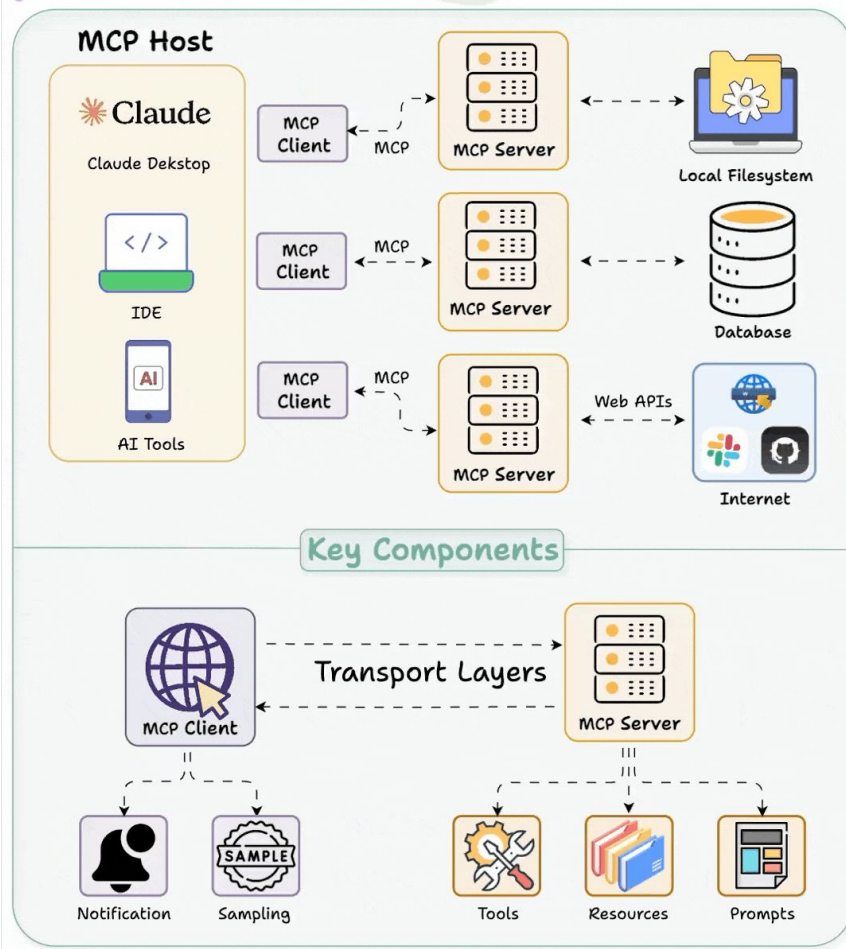


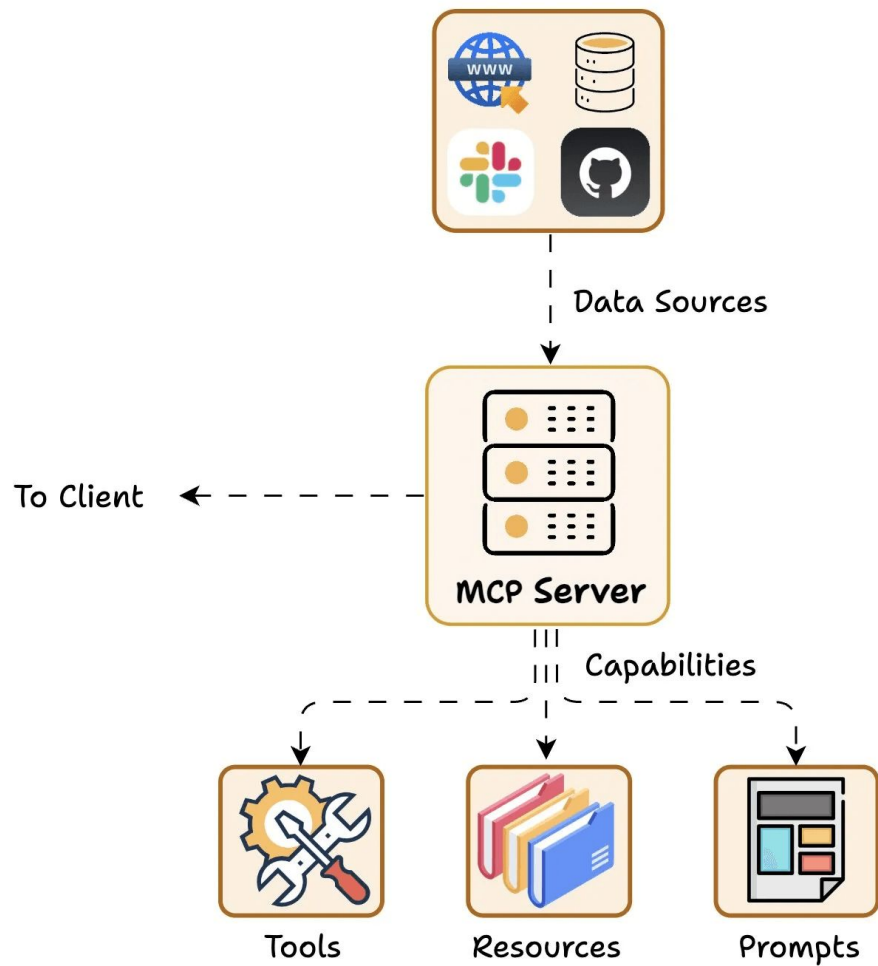
Ttransport Layer

- Manages communication channels and authentication between clients and servers MCP supports two transport mechanisms:
 - **Stdio transport:** Uses standard input/output streams for direct process communication between local processes on the same machine, providing optimal performance with no network overhead
 - **Streamable HTTP transport:** Uses HTTP POST for client-to-server messages with optional Server-Sent Events (**SSE**) for streaming capabilities



Model Context Protocol (MCP), clearly explained







Primitives (servers)

- The most important concept within MCP
 - They define what clients and servers can offer each other
- MCP defines three core primitives for **servers**:
 - **Tools**: Executable functions that AI applications can invoke to perform actions (e.g., file operations, API calls, database queries)
 - **Resources**: Data sources that provide contextual information to AI applications (e.g., file contents, database records, API responses)
 - **Prompts**: Reusable templates that help structure interactions with language models (e.g., system prompts, few-shot examples)



Primitives (clients)

- Allow MCP server authors to build richer interactions
 - **Sampling:** Allows servers to request language model completions from the client's AI application. This is useful when servers' authors want access to a language model, but want to stay model independent and not include a language model SDK in their MCP server. They can use the sampling/complete method to request a language model completion from the client's AI application



Primitives (clients)

- Allow MCP server authors to build richer interactions
 - **Elicitation:** Allows servers to request additional information from users. This is useful when servers' authors want to get more information from the user, or ask for confirmation of an action. They can use the elicitation/request method to request additional information from the user.
 - **Logging:** Enables servers to send log messages to clients for debugging and monitoring purposes.



NOTIFICATIONS

- The protocol supports real-time notifications to enable dynamic updates between servers and clients
- For example, when a server's available tools change—such as when new functionality becomes available or existing tools are modified—the server can send tool update notifications to inform connected clients about these changes
- Notifications are sent as JSON-RPC 2.0 notification messages (without expecting a response) and enable MCP servers to provide real-time updates to connected clients



What can it do for you?

- LLMs (through agents) can (general)
 - Access your Google Calendar and Notion, acting as a more personalized AI assistant.
 - Generate an entire web app using a Figma design.
 - Connect to multiple databases across an organization, empowering users to analyze data using chat.
 - Create 3D designs on Blender and print them out using a 3D printer.



What can it do for you?

- LLMs (through agents) can (enterprise)
 - Access your files
 - Access information on databases
 - Access in-memory data (applications)
 - Run code in your system
 - ...



MCP vs API (REST)

Feature	MCP	REST APIs
State Management	Stateful: maintains context across interactions	Stateless: each request is independent
Connection Type	Persistent, bidirectional connections	One-way request/response
Communication Style	JSON-RPC based with ongoing sessions	HTTP-based with discrete requests
Context Handling	Context is intrinsic to the protocol	Context must be manually managed
Tool Discovery	Runtime discovery of available tools	Design-time integration (OpenAPI)
Integration Approach	Runtime integration with dynamic capabilities	Design-time integration (code changes)



Why does MCP matter??

- MCP can have a range of benefits:
 - **Developers:** MCP reduces development time and complexity when building, or integrating with, an AI application or agent.
 - **AI applications or agents:** MCP provides access to an ecosystem of data sources, tools and apps which will enhance capabilities and improve the end-user experience.
 - **End-users:** MCP results in more capable AI applications or agents which can access your data and take actions on your behalf when necessary.



WHAT CAN IT DO FOR YOU??

?



AI con Delphi

21 Gennaio 2026

Parma + Remoto

Usare AI da Delphi

l'AI nell'IDE (quale IDE?)

I componenti di Delphi

Scrivere applicazioni "AI": cosa significa?

MCP server con Delphi

22 Gennaio 2026

Parma + Remoto

Perchè dobbiamo usare MCP

MCPConnect da zero

Integrazione con LLM (locali e remoti)

MCP + REST per non riscrivere tutto!

Esempi reali di utilizzo MCP

Partecipanti Delphi Day Padova - 20% sconto



MCPConnect

2



MCPConnect

- New OpenSource Library
 - <https://github.com/delphi-blocks/MCPConnect>
- MCP Servers (and Clients)
- JSON-RPC 2.0 Library
- Uses Neon
- Work in Progress...



Concepts

- Attributes to call class methods
- Automatic tools/list from multiple Delphi classes
- Easy Server configuration
- WebBroker/Indy integration



Project's structure

- **MCPConnect.JRPC.***
 - JSON-RPC library (independent library in the future)
- **MCPConnect.Configuration.***
 - Configuration classes for JRPC/MCP (Neon, Auth, etc...)
- **MCPConnect.MCP.***
 - MCP core classes
- **MCPConnect.Transport.***
 - Transport mechanisms for MCP servers



MCP Project

- WebBroker application
- On FormCreate
 - FServer.OnParseAuthentication := ParseAuthentication;
 - VHandled := True;
- On WebModule MCPConnect Configuration
 - Like WiRL configuration
 - Fluent style, plugin-based



MCP configuration

FJRPCServer

```
.Plugin.Configure<IAuthTokenConfig>
    .SetToken('my-secret-token')
    .ApplyConfig

.Plugin.Configure<IMCPConfig>
    .SetServerName('delphi-mcp-server')
    .SetServerVersion('2.0.0')
    .RegisterWriter(TMCPImageWriter)
    .RegisterWriter(TMCPPictureWriter)
    .RegisterWriter(TMCPStreamWriter)
    .RegisterWriter(TMCPStringListWriter)
    .SetToolClass(TDataModule1)
.ApplyConfig;
```



MCP Tools

- Attribute [MCPTool] on standard Delphi methods
- Attribute [MCPPParam] on function params
- **That's it, you have your MCPServer up and running!**



Demo:

Basic MCPConnect example

- ★ Server configuration
- ★ Delphi method to MCP tool
- ★ Call from Bruno



LLM integrations

- LLM integration through http(s)
 - stdio planned for the future
- Each LLM has a peculiar configuration
- What to use to test/use MCPConnect-based servers?
 - LLMStudio (local LLM)
 - Claude Desktop (remote LLM)
 - Gemini CLI (remote LLM)
 - ...



Basic LLM configuration

- LLM integration through http(s)
 - stdio planned for the future
- Each LLM has a peculiar configuration
- What to use to test/use MCPConnect-based servers?
 - LLMStudio
 - Claude Desktop
 - Gemini CLI
 - ...



MCP Servers (base) configuration file

```
{
  "mcpServers": {
    "delphi-mcp": {
      "type": "http",
      "url": "http://localhost:8080/mcp",
      "headers": {
        "Authorization": "Bearer my-secret-token"
      }
    },
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "<YOUR_TOKEN>"
      }
    }
  }
}
```



Demo:

LLM integration and use

- ★ Claude Desktop configuration
- ★ Basic interaction through prompts
- ★ Gemini CLI *



AI con Delphi

21 Gennaio 2026

Parma + Remoto

Usare AI da Delphi

l'AI nell'IDE (quale IDE?)

I componenti di Delphi

Scrivere applicazioni "AI": cosa significa?

MCP server con Delphi

22 Gennaio 2026

Parma + Remoto

Perchè dobbiamo usare MCP

MCPConnect da zero

Integrazione con LLM (locali e remoti)

MCP + REST per non riscrivere tutto!

Esempi reali di utilizzo MCP

Partecipanti Delphi Day Padova - 20% sconto



THANK YOU