PRACTICAL TECH GUIDE

Building MCP Servers

From First Tool to Real-World Workflows

```
python -m exif_mcp_server.server
```

Building MCP Servers

A Practical Guide to Designing and Deploying MCP Services with Python

Blue J. Lion
Quiet Line Press (quietlinepress.com)

First Edition: May 2026

Published by Quiet Line Press (quietlinepress.com)

Table of Contents

About This Book

Book Positioning

This book is for developers, technical creators, AI builders, and curious learners who want to understand what an MCP server is, how to build one in Python, and how to move from a local prototype to a production-leaning service.

Instead of explaining MCP in the abstract, this guide uses an open-source sample project:

https://github.com/nextframedev/exif_mcp_server

The project is intentionally practical. It inspects and removes EXIF metadata from images. It works well as a teaching example because it:

1. has clear, useful tools

2. deals with real local files and safe defaults

3. shows how to structure a production-leaning MCP server without becoming overly complex

This book does not treat the EXIF server as the end goal. It treats it as a working example.

The real goal is to help you:

1. understand how MCP actually works

2. study a complete Python server from the inside

3. reuse the same patterns in your own MCP services

What You Will Learn

By the end of this guide, you should understand:

1. What MCP is and why MCP servers matter.

2. How capability and context work together across an MCP session.

3. How to build a minimal Python MCP server with tools, resources, and prompts.

4. How a local MCP server works.

5. How to install, run, and test the EXIF MCP server.

6. How to connect it to MCP clients such as Inspector and VS Code.

7. How the project is structured internally.

8. How to think about tool design, safety, output contracts, and boundaries.

9. How to move from `stdio` development to remote HTTP deployment decisions.

10. How to use this project as a reference when building your own MCP server.

Quick Start in 5 Minutes

If you want the shortest path to a working setup, follow this section first.

There are really two useful kinds of quick start:

1. the smallest possible MCP loop

2. the full EXIF project setup

If you want the fastest possible first success, start with the tiny example below. Then come back and run the full EXIF project.

Fastest First Success: A Tiny MCP Server

Create a file named `server.py` with:

```
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("minimal-server")


@mcp.tool()
def hello(name: str) -> dict[str, str]:
    return {"message": f"Hello {name}"}
```

```
if __name__ == "__main__":
    mcp.run()
```

Install the MCP SDK and launch Inspector:

python3 -m venv .venv
source .venv/bin/activate
pip install mcp
npx @modelcontextprotocol/inspector python server.py

`npx @modelcontextprotocol/inspector` launches the official MCP Inspector, a browser-based tool for testing MCP servers locally. Reference: https://modelcontextprotocol.io/docs/tools/inspector

Then call `hello` with a name such as `"MCP"` and confirm you get:

```
{
  "message": "Hello MCP"
}
```

That proves the full local loop:

1. define a tool

2. start a server

3. connect a client

4. call the tool

5. receive structured output

Full EXIF Project Quick Start

Prerequisites:

1. Python 3.11 or newer

2. Git

3. A terminal

Step 1. Clone the project

git clone https://github.com/nextframedev/exif_mcp_server.git
cd exif_mcp_server

Step 2. Create and activate a virtual environment

python3 -m venv .venv
source .venv/bin/activate

Step 3. Install the project

pip install -e '.[dev]'

Step 4. Run the tests

pytest

Step 5. Start the server

python -m exif_mcp_server.server

The terminal will look idle. That is normal. The server is waiting for an MCP client over `stdio`.

Step 6. Do a quick smoke test without a client

Open a second terminal in the same project and run:

.venv/bin/python -c "from exif_mcp_server.server import create_server; print(type(create_server()).__name__)"

If your virtual environment is already activated, plain `python` should work too.

Expected output:

FastMCP

`FastMCP` is the framework this project uses to build the server. It handles the protocol layer so the project code can focus on tool logic. You do not need to know its internals to study or use this project.

If you got that far, the project is installed correctly and ready for an MCP client.

Step 7. Do a first real MCP test

At this point, the next goal is simple: prove that a real MCP client can discover the server and call a tool.

Two good first options are:

1. MCP Inspector, if you want to inspect raw tools, resources, and prompts

2. VS Code, if you want to test the server inside an editor workflow

For a first live check, keep it simple:

1. launch MCP Inspector:

npx @modelcontextprotocol/inspector .venv/bin/python -m exif_mcp_server.server

2. let Inspector start the local `stdio` server for you

3. open the `Tools` tab

4. select `inspect_exif`

5. enter the required string parameter `image_path`: examples/sample_images/gps-exif.jpg

6. click `Run Tool`

7. confirm you get a structured JSON result back

That single call proves several things at once:

1. the client can start or connect to the server

2. the MCP handshake works

3. tool discovery works

4. the EXIF core logic works on a real file

5. the response shape is usable in a client

Appendix A: Key Commands

Clone:

git clone https://github.com/nextframedev/exif_mcp_server.git
cd exif_mcp_server

Set up:

python3 -m venv .venv
source .venv/bin/activate
pip install -e '.[dev]'

Test:

pytest

Lint:

ruff check .

Type-check:

mypy

Run server:

python -m exif_mcp_server.server

Run Streamable HTTP:

python -m exif_mcp_server.server --transport streamable-http

Appendix B: One-Sentence Summary

This book uses `exif-mcp-server` as a worked example for building, testing, designing, and beginning to deploy practical Python MCP services without losing sight of contracts, context, and operational safety.

If you want a deeper companion guide to the EXIF side of this project, check out the companion book by the same author, available on Amazon: `EXIF Data & Image Metadata: How to Analyze, Remove, and Protect Your Photos — A Practical Privacy & Security Guide (The Practical Tech Guide Series).`

The underlying open-source EXIF project is also available here: https://github.com/nextframedev/image-metadata-tool

Appendix C: Official References

These are the primary references worth bookmarking while reading or extending the project.

1. MCP home:

https://modelcontextprotocol.io/

2. MCP specification:

https://modelcontextprotocol.io/specification/

3. MCP architecture overview:

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

4. MCP Inspector:

https://modelcontextprotocol.io/docs/tools/inspector

5. Official MCP SDKs:

https://modelcontextprotocol.io/docs/sdk

6. MCP Registry overview:

https://modelcontextprotocol.io/registry/about

About the Author

Blue J. Lion has over 20+ years of experience in software development, with a focus on programming, data security, and privacy. He has worked across engineering and product environments, building practical solutions and tools.

Beyond software, he enjoys creating simple, thoughtful products-ranging from books and visual tools to creative projects that explore the intersection of technology and everyday life.

In his free time, he enjoys running, swimming, and working on new ideas.

Quiet Line Press

Author Portfolio