

lab

April 3, 2026

1 Lab 13 — Building Your Own MCP Server

Build an MCP server with tools and resources using FastMCP.

Exercises: 1. Minimal server with an echo tool 2. Add note management tools 3. Add a notes:// resource

```
[ ]: # Setup
# In a real scenario you'd use: from mcp.server.fastmcp import FastMCP
# For this notebook, we simulate the decorators so you can develop and test
# tool logic without running a server process.

from dotenv import load_dotenv
load_dotenv()

# Simulated FastMCP for notebook use
class NotebookMCP:
    def __init__(self, name):
        self.name = name
        self._tools = {}
        self._resources = {}

    def tool(self):
        def decorator(func):
            self._tools[func.__name__] = func
            return func
        return decorator

    def resource(self, uri_pattern):
        def decorator(func):
            self._resources[uri_pattern] = func
            return func
        return decorator

mcp = NotebookMCP("lab-server")
print(f"Notebook MCP server '{mcp.name}' ready.")
```

1.1 Exercise 1 — Minimal Server with Echo Tool

Create the simplest possible MCP tool: echo.

```
[ ]: # TODO: Register an echo tool using @mcp.tool()
# It should take message: str and return "Echo: {message}"

raise NotImplementedError("Exercise 1: implement echo tool")

[ ]: # Test Exercise 1
print(echo(message="Hello MCP!"))
# Expected: "Echo: Hello MCP!"
```

1.2 Exercise 2 — Add Notes Tools

Add add_note and list_notes tools to the server.

```
[ ]: # TODO: Create a notes dict for storage
# TODO: Implement add_note(title: str, content: str) -> str
# TODO: Implement list_notes() -> str

raise NotImplementedError("Exercise 2: implement notes tools")

[ ]: # Test Exercise 2
print(add_note(title="Python", content="A versatile programming language"))
print(add_note(title="MCP", content="Model Context Protocol for LLM tools"))
print(add_note(title="FastMCP", content="High-level framework for MCP servers"))
print()
print(list_notes())
```

1.3 Exercise 3 — Add Notes Resource

Add notes://all and notes://{title} resources.

```
[ ]: # TODO: Implement get_all_notes() -> str (for notes://all)
# TODO: Implement get_note_by_title(title: str) -> str (for notes://{title})

raise NotImplementedError("Exercise 3: implement notes resources")

[ ]: # Test Exercise 3
print("=== All Notes ===")
print(get_all_notes())
print()
print("=== Single Note ===")
print(get_note_by_title(title="Python"))
print()
```

```
print("=== Missing Note ===")  
print(get_note_by_title(title="Nonexistent"))
```