

Lab 021: Build an MCP Server in C#

Level: L200 Path: MCP Time: ~45 min ? Cost: Free (local + Ollama)

What You'll Learn

Create an MCP server using the official ModelContextProtocol .NET SDK Expose tools, resources, and prompts from C# Test the server with MCP Inspector Connect it to GitHub Copilot Agent Mode via mcp.json

Introduction

Python is great for rapid MCP prototyping, but .NET is common in enterprise environments. The official ModelContextProtocol NuGet package makes building MCP servers in C# first-class.

Prerequisites

.NET 8 SDK or later ? free Lab 012: What is MCP? recommended Node.js (for MCP Inspector) ? free

? Supporting Files

note "Download these files before starting the lab" Save all files to a lab-021/ folder in your working directory.

File | Description | Download | BrokenMcpServer.cs | Bug-fix exercise (3 bugs + self-tests)

| ? Download |

Lab Exercise

Step 1: Create the project

[diagram or code sample]

Step 2: Build the MCP server

Replace Program.cs with:

[diagram or code sample]

Create ProductTools.cs:

[diagram or code sample]

Step 3: Run and test with MCP Inspector

Terminal 1 ? build the server: [diagram or code sample]

Test with MCP Inspector: [diagram or code sample]

In Inspector, click Tools and test search_products with query "camping". You should see the tent returned.

Step 4: Add a Resource

Resources expose read-only data (files, database views, etc.). Add to ProductTools.cs:

[diagram or code sample]

Update Program.cs to register resources: [diagram or code sample]

Step 5: Connect to GitHub Copilot

Add to .vscode/mcp.json in your workspace:

[diagram or code sample]

Enable Agent Mode in VS Code, then ask: `""What camping products are in stock?""`

Key Differences vs Python SDK

| Python | C# | Decorator | @mcp.tool() | [McpServerTool] | Description | docstring |

[Description("...")] | Resources | @mcp.resource() | [McpServerResource(...)] | Transport | mcp.run(transport="stdio") | .WithStdioServerTransport() | DI container | ? |

Microsoft.Extensions.Hosting |

? Bug-Fix Exercise: Fix the Broken MCP Server

This lab includes a deliberately broken C# MCP server file. Your challenge: find and fix 3 bugs.

[diagram or code sample]

Setup: [diagram or code sample]

The 3 bugs:

| Tool | Symptom | Type | 1 | list_categories | NullReferenceException on startup | Null initialization | 2 | search_products | Always returns empty list [] | Logic inversion (!) | 3 | get_product_details | Returns "not found" for lowercase IDs | Case-sensitive comparison |

Verify your fixes: After fixing all 3 bugs, connect with the MCP Inspector and run:

list_categories() ? should return ["Backpacks", "Sleeping Bags", "Tents"]

search_products(keyword: "tent") ? should return P001, P002, P003

get_product_details(productId: "p001") ? should return TrailBlazer Tent 2P details

? Knowledge Check

question "Q1 (Run the Lab): After fixing all 3 bugs and calling list_categories(), what does the tool return? List the categories in the order they appear in the output."

Fix the bugs, start the server, connect with MCP Inspector, and call list_categories().

??? success "? Reveal Answer" ["Backpacks", "Sleeping Bags", "Tents"]

The categories are returned in alphabetical order because the original code uses a sorted List. Bug #1 (categories = null) caused a NullReferenceException before returning anything ? fixing it reveals the properly sorted list.

question "Q2 (Run the Lab): After fixing bug #3 (the case-sensitive comparison bug), what StringComparison value replaces StringComparison.Ordinal in the fix?"

Read the bug #3 description carefully, then look at the fix you applied in ?

BrokenMcpServer.cs.

??? success "? Reveal Answer" StringComparison.OrdinalIgnoreCase

The original code used StringComparison.Ordinal which is case-sensitive, so get_product_details("p001") failed because the stored IDs are uppercase ("P001"). Replacing it with StringComparison.OrdinalIgnoreCase makes ID lookups work regardless of the case the client sends.

question "Q3 (Multiple Choice): Bug #2 in search_products caused it to always return an empty list. What was the root cause?"

- A) The keyword parameter was null - B) The Contains() call was inverted with ! ? it filtered OUT matches instead of keeping them - C) The product list was not initialized - D) The search was case-sensitive and no products matched

??? success "? Reveal Answer" Correct: B ? Logic inversion

The code had !product.Name.Contains(keyword) ? the ! negated the condition, so products that DID contain the keyword were excluded, and products that did NOT contain the keyword were returned. With an empty results list, there were no non-matching products either.

Removing the ! fixes the logic.

Next Steps

Deploy this server to the cloud: ? Lab 028 ? Deploy MCP to Azure Container Apps Python

version of MCP server: ? Lab 020 ? MCP Server in Python

Phase 3 modernization

warning "Verified 2026-10-27" Target MCP specification 2026-07-28. Prefer stateless server design, capability discovery, current Tier 1 SDKs, and explicit multi-round-trip state. Add Registry and authorization only where the trust model requires them.

[diagram or code sample]

Model Context Protocol specification MCP authorization MCP Registry

Supporting kit

The site build generates stable PDF and ZIP downloads. The ZIP contains the lab guide and every supporting file declared for this lab.

