

(ฉบับนักพัฒนา)

FastMCP

เรียนรู้พื้นฐาน สู่การใช้งานจริง

อนุชิต ชาญ

สำนักพิมพ์ก๊อปวาง

FastMCP ฉบับนักพัฒนา

เรียนรู้พื้นฐานสู่การใช้งานจริง

อนุชิต ชโลธร

คำนำสำนักพิมพ์

ในโลกที่ AI กำลังเปลี่ยนวิธีการทำงานทุกอุตสาหกรรม ความสามารถในการ “เชื่อมต่อ” โมเดลภาษาขนาดใหญ่ (LLMs) เข้ากับเครื่องมือและข้อมูลภายนอกอย่างปลอดภัยและน่าเชื่อถือ กลายเป็นทักษะที่แยกนักพัฒนาระดับธรรมดาออกจากมืออาชีพที่สร้างนวัตกรรมได้จริง

FastMCP ฉบับนักพัฒนา คือคู่มือที่พาคุณตรงเข้าสู่หัวใจของ Model Context Protocol (MCP) — โพรโตคอลที่เปรียบได้กับ “USB-C ของโลก AI” ทำให้ LLM สามารถทำงานร่วมกับระบบใดก็ได้ได้อย่างราบรื่นและควบคุมได้ หนังสือเล่มนี้จะไม่เพียงสอนวิธีใช้งาน FastMCP ซึ่งพร้อมสำหรับงานระดับโปรดัคชัน แต่ยังถ่ายทอดแนวคิด การออกแบบ และเทคนิคเชิงลึก ที่คุณสามารถนำไปต่อยอดได้ทันที

คุณจะได้เรียนรู้ตั้งแต่ศูนย์จนถึงขั้นสูง — สร้าง MCP Server แรกของคุณ เพิ่ม Tools และ Resources อย่างปลอดภัย ทดสอบ ปรับแต่ง และประกอบหลายเซิร์ฟเวอร์เข้าด้วยกัน เพื่อสร้างระบบ AI ที่ซับซ้อน พร้อมเชื่อมต่อกับ REST API อัดโน้ตผ่าน OpenAPI และ FastAPI

นี่ไม่ใช่แค่หนังสือเทคนิค แต่คือ “คู่มือสร้างโครงสร้างพื้นฐาน AI” ที่จะช่วยให้คุณ

- สร้างระบบได้เร็วขึ้น
- ลดความเสี่ยงด้านความปลอดภัย
- ขยายความสามารถของ LLM ได้อย่างไรขีดจำกัด

หากคุณจริงจังกับการพัฒนา AI ที่ทั้ง ทรงพลัง ยืดหยุ่น และปลอดภัย
หนังสือเล่มนี้คือเครื่องมือที่คุณไม่ควรพลาด

คำนำ

ในยุคที่ปัญญาประดิษฐ์และโมเดลภาษาขนาดใหญ่ (Large Language Models หรือ LLMs) กำลังเข้ามามีบทบาทอย่างลึกซึ้งในหลากหลายวงการ ความสามารถในการเชื่อมต่อ LLM กับเครื่องมือและข้อมูลภายนอกอย่างปลอดภัยและน่าเชื่อถือ กลายเป็นความท้าทายสำคัญที่นักพัฒนาทุกคนต้องเผชิญ

หนังสือ “*FastMCP ฉบับนักพัฒนา*” เล่มนี้ จะพาคุณไปรู้จักกับ FastMCP – เฟรมเวิร์กที่ออกแบบมาเพื่อช่วยจัดการความซับซ้อนของ Model Context Protocol (MCP) โปรโตคอลที่ทำหน้าที่เป็นสะพานเชื่อมระหว่างโลกของ AI ซึ่งเต็มไปด้วยความยืดหยุ่นและไม่แน่นอน กับโลกของซอฟต์แวร์ที่ต้องการความแม่นยำ ปลอดภัย และควบคุมได้

MCP เปรียบเสมือน “USB-C สำหรับ AI” ที่ทำให้ LLM ทุกตัวสามารถเชื่อมต่อกับเครื่องมือหรือข้อมูลภายนอกได้อย่างราบรื่น ผ่านอินเทอร์เฟซที่มีโครงสร้างชัดเจนและเปิดเผย ช่วยให้ระบบสามารถค้นพบฟังก์ชันใหม่ๆ ระหว่างรันไทม์ ป้องกันการรันโค้ดโดยไม่ได้รับอนุญาต และสามารถประกอบรวมเซิร์ฟเวอร์หลายตัวเข้าด้วยกัน เพื่อสร้างแอปพลิเคชันที่ซับซ้อนและทรงพลังได้อย่างมั่นใจ

FastMCP เป็นไลบรารี Python ที่ช่วยให้นักพัฒนาสามารถสร้าง MCP Server ได้อย่างรวดเร็ว พร้อมเครื่องมือและโครงสร้างพื้นฐานที่ออกแบบมาอย่างดี เพื่อให้คุณสามารถมุ่งเน้นไปที่การพัฒนาฟังก์ชันหลักของแอปพลิเคชันได้อย่างเต็มที่ โดยเนื้อหาในหนังสือเล่มนี้อิงจาก FastMCP เวอร์ชัน 2.0 ซึ่งครบถ้วนด้วยฟีเจอร์ที่พร้อมสำหรับการใช้งานจริงในระดับโปรดักชัน

คุณจะได้เรียนรู้ทั้งแนวคิดและขั้นตอนการใช้งาน ตั้งแต่การสร้างเซิร์ฟเวอร์ FastMCP เบื้องต้น การเพิ่มเครื่องมือ (Tools) และทรัพยากร (Resources) การทดสอบเซิร์ฟเวอร์ การจัดการด้านความปลอดภัย การใช้งานไคลเอนต์ขั้นสูง การประกอบเซิร์ฟเวอร์หลายตัวเข้าด้วยกัน ไปจนถึงการเชื่อมต่อ REST API ผ่าน OpenAPI และ FastAPI สำหรับให้ LLM เข้าถึงได้โดยอัตโนมัติ

เราหวังเป็นอย่างยิ่งว่า “FastMCP ฉบับนักพัฒนา” เล่มนี้ จะเป็นกุญแจดอกสำคัญที่ช่วยเปิดประตูสู่โลกใหม่ของการพัฒนา AI ที่ทั้งทรงพลัง ยืดหยุ่น และปลอดภัย พร้อมเสริมศักยภาพให้คุณสร้างนวัตกรรมได้อย่างมั่นใจในยุคแห่ง AI

สารบัญ

แนะนำ FastMCP และ Model Context Protocol (MCP)	1
MCP คืออะไร	1
MCP เปรียบเทียบกับ REST API	2
LLM เรียกใช้ MCP อย่างไร	2
องค์ประกอบของ MCP Server	3
Tool	3
Resource	4
Prompt	5
FastMCP คืออะไร	7
ทำไมต้องใช้ FastMCP	7
โค้ดตัวอย่าง MCP Server	7
ความสำคัญของ FastMCP ในยุคของ LLM Application	8
เริ่มต้นใช้งาน FastMCP	10
การติดตั้ง FastMCP	10
ตรวจสอบการติดตั้ง	10
อัปเดตจาก FastMCP เวอร์ชัน 1.0	11
การสร้าง FastMCP Server เบื้องต้น	12
การเพิ่ม Tool	12

การทดสอบ Server ด้วย FastMCP Client	13
การรัน Server	15
การเรียกใช้ผ่านไฟล์ Client	17
การรัน Server ด้วย FastMCP CLI	17
การสร้าง Tools ใน FastMCP Server	20
Tools คืออะไร?	20
การกำหนด Tools ด้วย @mcp.tool	21
การกำหนดชื่อ คำอธิบาย และ Tag	22
Tools แบบ Asynchronous (Async) และ Synchronous (Sync)	24
การกำหนดชนิดข้อมูลและข้อมูลเมต้าของพารามิเตอร์	26
ชนิดข้อมูลพื้นฐาน (Built-in Types)	27
ชนิดข้อมูลสำหรับวันที่และเวลา	27
ชนิดข้อมูล Collection	28
ชนิดข้อมูล Union และ Optional	28
ชนิดข้อมูล Constrained	28
Literals	28
Enums	29
ข้อมูลไบนารี	30
Paths และ UUIDs	31
pathlib.Path	31

uuid.UUID	32
Pydantic Models	33
Pydantic Fields	34
การจัดการ Argument	36
Parameters ที่จำเป็น (Required) และทางเลือก (Optional)	36
การยกเว้น Argument จาก Schema ของ Tool (Excluding Arguments)	36
การส่งคืนข้อมูล	37
Content Blocks	37
ข้อมูลแบบมีโครงสร้าง	38
กฎของ Automatic Structured Content	38
การส่งออกข้อมูลแบบมีโครงสร้าง	39
ส่งคืนข้อมูลเป็น ToolResult	39
การจัดการข้อผิดพลาด	40
การซ่อนรายละเอียดข้อผิดพลาด	41
การเพิ่ม Metadata ให้กับ Tools	41
การแจ้งเตือนเมื่อมีการเปลี่ยนแปลง	43
การจัดการ Tools ที่ซ้ำกัน	44
การเรียกใช้ข้อมูลใน MCP Context	45
การลบ Tools แบบไดนามิก	48

การสร้าง Resources ใน FastMCP Server	50
Resources คืออะไร?	50
การกำหนด Resources ด้วย @mcp.resource	51
รูปแบบข้อมูลที่ส่งคืน	52
สร้าง Resource แบบไดนามิก	53
การจัดการข้อผิดพลาด	54
การแจ้งเตือนเมื่อมีการเปลี่ยนแปลง	54
การใช้งาน Resource จากฝั่ง Client	54
รองรับ Async Resource	55
การใช้ Resource Classes	55
การจัดการ Resources ซ้ำกัน	56
การสร้าง Prompts ใน FastMCP Server	58
Prompts คืออะไร?	58
การกำหนด Prompts ด้วย @mcp.prompt	59
Argument Types และ Automatic Serialization	62
Return Values ของ Prompt	63
Required vs. Optional	64
Async Prompts	65
การเข้าถึง MCP Context	66
การแจ้งเตือนสำหรับการเปลี่ยนแปลง Prompt	67
การจัดการ Prompt ที่มีชื่อซ้ำกัน	68

การใช้งาน Tool ใน FastMCP Client	71
การค้นหาเครื่องมือ	71
การกรองด้วยแท็ก	72
การรันเครื่องมือ	72
รันแบบพื้นฐาน	72
รันแบบกำหนดตัวเลือกขั้นสูง	73
การจัดการผลลัพธ์	74
คุณสมบัติหลักของ CallToolResult	74
ตัวอย่างการเข้าถึงข้อมูลเชิงโครงสร้าง	74
กรณีไม่มีข้อมูลโครงสร้าง	75
การคืนค่าชนิดพื้นฐาน	75
การจัดการข้อผิดพลาด	76
แบบใช้ Exception	76
แบบตรวจสอบเอง	76
การเข้าถึงโปรโตคอล MCP แบบดิบ	77
การส่งอาร์กิวเมนต์	77
การใช้งาน Resource ใน FastMCP Client	79
ประเภทของ Resource	79
การดึงรายการ Resource	79
Static Resources	79
Resource Templates	80

การกรอง Resource ด้วย Tags	81
การอ่าน Resource	81
Static Resources	81
Resource Templates	82
ประเภทของเนื้อหา	82
การใช้งานกับ Multi-Server Clients	83
การเข้าถึงข้อมูล MCP Protocol แบบดิบ	84
รูปแบบ URI ที่พบบ่อย	84
การใช้งาน Prompts ใน FastMCP Client	86
การดูรายการ Prompts ที่มีอยู่	86
การกรอง Prompts ด้วย Tags (ตั้งแต่เวอร์ชัน 2.11.0)	87
การเรียกใช้งานพื้นฐาน	88
Prompt พร้อม Arguments	88
การแปลง Arguments อัตโนมัติ (ตั้งแต่เวอร์ชัน 2.9.0)	89
การทำงานกับผลลัพธ์ Prompt	90
การเข้าถึงโปรโตคอล MCP โดยตรง	91
การใช้งานกับ Multi-Server Clients	91
รูปแบบการใช้งาน Prompt ที่พบบ่อย	92
System Messages	92
Conversation Templates	92
การตรวจสอบสิทธิ์	95

ความท้าทายของการตรวจสอบสิทธิ์ใน MCP	95
ทำความเข้าใจระดับความรับผิดชอบในการตรวจสอบสิทธิ์	96
การตรวจสอบด้วย Token	97
การใช้ผู้ให้บริการยืนยันตัวตนภายนอก	99
การใช้งาน OAuth เต็มรูปแบบ	100
การกำหนดค่าการตรวจสอบสิทธิ์ฝั่ง Server	101
การกำหนดค่าแบบ Programmatic	102
การกำหนดค่าด้วย Environment Variables	102
การตรวจสอบสิทธิ์ฝั่ง Client	103
Bearer Token Authentication	103
OAuth Authentication	105
การเลือกใช้งานการตรวจสอบสิทธิ์ที่เหมาะสม	108
การใช้งาน FastMCP Client ขั้นสูง	110
สถาปัตยกรรม Client-Transport	110
Client ที่ใช้ไฟล์ Configuration-Based	112
จัดการวงจรชีวิตการเชื่อมต่อ	114
การทำงานกับผลลัพธ์ของ Tools และ Resources	116
CallToolResult	116
ส่งคืนผลลัพธ์ประเภทข้อมูลดั้งเดิม	117
Callback Handlers	120

คุณสมบัติขั้นสูงและแนวทางในการนำเซิร์ฟเวอร์ MCP ไปใช้	127
งาน	
การควบคุมพฤติกรรมของเซิร์ฟเวอร์	127
การตั้งค่าส่วนประกอบที่ซ้ำกัน	127
การลบ Tools แบบไดนามิก	129
การรวมเซิร์ฟเวอร์ MCP	130
การทำ Proxy MCP Server	131
สร้าง MCP Server อัตโนมัติจาก OpenAPI	132
สร้าง MCP Server อัตโนมัติจาก FastAPI	137
การตั้งค่า Server	137
การตั้งค่าเฉพาะ Server	138
การตั้งค่าเซิร์ฟเวอร์แบบ Global	139
การกำหนดค่าเซิร์ฟเวอร์เฉพาะ Transport	141
การ Serialization Tool แบบกำหนดเอง	143
การใช้งานร่วมกับไคลเอนต์ MCP	146
การใช้งานร่วมกับไคลเอนต์ MCP	146
มาตรฐาน MCP JSON Configuration	146
โครงสร้างไฟล์การตั้งค่า	147
การตั้งค่าไคลเอนต์ MCP ด้วย FastMCP CLI	148
การใช้งานพื้นฐาน	149
การตั้งชื่อเซิร์ฟเวอร์	151

การเพิ่ม Dependencies	152
การตั้งค่า Environment Variables	153
การระบุเวอร์ชัน Python และไลบรารีโครงการ	153
การเลือก Server ที่ต้องการใช้	153
การคัดลอกคอนฟิกไปยังคลิปปอร์ด	154
การเรียกใช้ Server พื้นฐาน	154
การเรียกใช้ Server แบบ Production	155
การกำหนดค่าขั้นสูง	155
การใช้งานแบบ Pipeline	157
การใช้ FastMCP ร่วมกับระบบ API ที่มีอยู่	159
OpenAPI	159
FastAPI	162
Starlette / ASGI	169
การตั้งค่าเซิร์ฟเวอร์แบบ Remote HTTP	177
การเลือกวิธีการตั้งค่า	177
การตั้งค่าแบบ Direct HTTP Server	178
การปรับใช้แบบ ASGI Application	179
การตั้งค่าเซิร์ฟเวอร์	181
การตั้งค่าใน Production	182
การโฮสต์เซิร์ฟเวอร์ของคุณ	184
ตัวอย่างการสร้าง Task Management MCP Server	185

แนวคิดเบื้องต้น	185
Features	185
Tools ที่ Server เปิดให้ Client เรียกใช้งาน	186
Resources ที่ Server เปิดให้ Client เรียกใช้งาน	186
เริ่มต้นโปรเจกต์	187
โครงสร้างของ Task Model	187
TaskManager Class	188
การเชื่อมต่อกับ FastMCP	189
การสร้าง MCP Instance	189
การสร้าง Tools	189
การสร้าง Resources	190
การรันเซิร์ฟเวอร์	190
ตัวอย่างการใช้งาน Tools	190
โค้ดตัวอย่างเต็ม	191
การเชื่อมต่อกับ Claude Desktop ด้วย FastMCP CLI	206
ติดตั้ง FastMCP CLI	207
ลงทะเบียน MCP Server กับ Claude Desktop	207
เริ่มใช้งานใน Claude Desktop	207
ติดตั้งสำหรับ Cursor / VS Code	208
ตัวอย่างการสร้าง Weather Forecast MCP Server	210
สถาปัตยกรรม	211

Tools – ฟังก์ชันที่ Server เปิดให้ Client เรียกใช้งาน	211
Resources – ที่ Server เปิดให้ Client เรียกใช้งาน	211
Flow การทำงานของ Server	212
เริ่มต้นโปรเจกต์	212
การเตรียมระบบและติดตั้ง	213
โค้ดตัวอย่าง	213
การเชื่อมต่อกับ Claude Desktop ด้วย FastMCP CLI	238
ติดตั้ง FastMCP CLI	239
ลงทะเบียน MCP Server กับ Claude Desktop	239
เริ่มใช้งานใน Claude Desktop	239
ติดตั้งสำหรับ Cursor / VS Code	240
ตัวอย่างการใช้งาน FastMCP ร่วมกับ Agent	242
ทดลองใช้งาน	243
ดูสภาพอากาศปัจจุบัน	243
พยากรณ์อากาศล่วงหน้า 2 วัน	243
เปรียบเทียบสภาพอากาศระหว่าง 2 เมือง	244
ดาวนโหลดซอร์สโค้ด	247

แนะนำ FastMCP และ Model Context Protocol (MCP)

MCP คืออะไร

Model Context Protocol (MCP) คือมาตรฐานการสื่อสารแบบเปิดที่ออกแบบมาสำหรับการทำให้โมเดลภาษา (LLMs) สามารถโต้ตอบกับโลกภายนอกได้อย่างปลอดภัย มีโครงสร้าง และสามารถทำซ้ำได้ (reproducible) โดยไม่ต้องอาศัยโค้ดเฉพาะกิจ (ad hoc integration) อีกต่อไป

การมีโปรโตคอลกลางเช่น MCP ทำให้นักพัฒนาสามารถสร้างระบบที่

- โมเดลภาษา “เข้าใจ” ได้ว่ามีเครื่องมือหรือข้อมูลอะไรให้ใช้งาน
- สามารถจำกัดขอบเขตการทำงานของโมเดลให้อยู่ภายใต้กรอบที่ควบคุมได้
- ทำงานร่วมกับเครื่องมืออื่นๆ ได้แบบ plug-and-play

ลองนึกภาพว่าคุณกำลังพัฒนา LLM Agent ที่ต้องเชื่อมต่อกับระบบฐานข้อมูล, API ภายในองค์กร หรือฟังก์ชันเฉพาะของตนเอง การใช้ MCP ทำให้คุณสามารถประกาศความสามารถเหล่านี้ให้โมเดลเข้าถึงได้ผ่าน schema ที่เป็นทางการ

MCP เปรียบเทียบกับ REST API

แม้ว่า REST API จะถูกใช้ในงาน AI อย่างแพร่หลาย แต่การออกแบบ REST ไม่ได้คำนึงถึงบริบทของ LLM เป็นหลัก เช่น การอธิบายว่า endpoint นี้ใช้ทำอะไร, รับพารามิเตอร์อะไร และมีความหมายในโลกของผู้ใช้หรือโมเดลอย่างไร ในทางกลับกัน MCP มีการกำหนด schema และ metadata ที่ชัดเจนสำหรับทุกเครื่องมือและทรัพยากร ซึ่งทำให้โมเดลสามารถคิดและวางแผนได้ดีขึ้น

LLM เรียกใช้ MCP อย่างไร

เมื่อโมเดลภาษารับ prompt ที่บอกว่า “คุณสามารถใช้เครื่องมือเหล่านี้ได้” มันจะวางแผนการโต้ตอบกับเครื่องมือเหล่านั้นผ่านการสร้าง JSON-RPC requests เพื่อเรียก tool หรือ resource ที่เซิร์ฟเวอร์เปิดเผยไว้ ตัวอย่างการเรียกใช้งานจากโมเดล

```
{
  "method": "tool/send_email",
  "params": {
    "to": "user@example.com",
    "subject": "Reminder",
    "body": "Your report is due today."
  }
}
```

ฝั่งเซิร์ฟเวอร์จะตรวจสอบความถูกต้องของพารามิเตอร์และดำเนินการตามฟังก์ชันที่ได้กำหนดไว้ จากนั้นส่งผลลัพธ์กลับให้โมเดล

องค์ประกอบของ MCP Server

เซิร์ฟเวอร์ MCP เปิดเผยความสามารถของตนเองให้โลกภายนอกรับรู้ผ่านองค์ประกอบหลัก 3 ส่วน ได้แก่ Tools, Resources และ Prompts ซึ่งแต่ละส่วนมีหน้าที่แตกต่างกันอย่างชัดเจน เพื่อให้ LLM สามารถโต้ตอบกับระบบได้อย่างมีแบบแผนและคาดเดาได้ เราจะมาทำความรู้จักกับองค์ประกอบแต่ละส่วนกัน

Tool

Tool คือ องค์ประกอบหลักที่เปิดโอกาสให้ LLM สามารถโต้ตอบกับระบบภายนอก เรียกใช้โค้ด และเข้าถึงข้อมูลที่ไม่ได้อยู่ในชุดข้อมูลการฝึกของโมเดลได้โดยตรง ในระบบ FastMCP เครื่องมือเหล่านี้ถูกนิยามเป็น **ฟังก์ชันของ Python** ปกติ เปิดให้ LLM เรียกใช้งานผ่านโปรโตคอล MCP

การทำงานของ Tool ใน FastMCP

เมื่อ LLM ตัดสินใจเรียกใช้ Tool ใด Tool หนึ่ง

1. LLM จะส่งคำร้องขอพร้อมพารามิเตอร์ที่ตรงกับ schema ของ Tool นั้น

2. FastMCP จะตรวจสอบความถูกต้องของพารามิเตอร์กับ signature ของฟังก์ชัน
3. ฟังก์ชันของคุณจะถูกเรียกใช้ด้วยค่าที่ผ่านการตรวจสอบแล้ว
4. ผลลัพธ์ที่ได้จะถูกส่งกลับไปยัง LLM เพื่อนำไปใช้ในการสร้างคำตอบต่อไป

ประโยชน์ของ Tool

การใช้ Tool ทำให้ LLM สามารถ

- ดึงข้อมูลจากฐานข้อมูล
- เรียกใช้ API ภายนอก
- คำนวณหรือประมวลผลข้อมูล
- เข้าถึงไฟล์หรือระบบอื่นๆ

ซึ่งช่วย **ขยายขีดความสามารถของ LLM** ให้สามารถตอบสนองต่อคำถามหรือคำสั่งได้อย่างชาญฉลาดและแม่นยำยิ่งขึ้น โดยไม่จำกัดอยู่เพียงข้อมูลในชุดการฝึกเดิมนั้น

Resource

Resource คือข้อมูลหรือไฟล์ที่ไคลเอนต์ของ MCP สามารถเรียกดูได้ในลักษณะ *read-only* (อ่านอย่างเดียว) โดยไม่สามารถแก้ไขหรือเปลี่ยนแปลงได้ และ Resource Template คือการขยายความ

สามารถนี้ให้สามารถรับพารามิเตอร์ผ่าน URI และสร้างเนื้อหาแบบไดนามิกตามค่าพารามิเตอร์นั้น

Resource ทำงานอย่างไร

เมื่อไคลเอนต์ร้องขอ URI ของ resource

1. FastMCP จะค้นหานิยามของ resource ที่ตรงกับ URI นั้น
2. ถ้าเป็น resource แบบไดนามิก (เช่น ถูกนิยามโดยฟังก์ชัน) ฟังก์ชันนั้นจะถูกเรียกใช้งาน
3. ระบบจะคืนค่าข้อมูลให้แก่ไคลเอนต์ เช่น ข้อความ, JSON หรือ ข้อมูลไบนารี

วิธีนี้ ทำให้ LLM หรือแอปพลิเคชันสามารถเข้าถึงข้อมูลที่จำเป็นต่อบริบทของการสนทนาได้ เช่น

- ไฟล์ในระบบ
- ข้อมูลจากฐานข้อมูล
- การตั้งค่าหรือคอนฟิก
- ข้อมูลที่ถูกสร้างขึ้นแบบไดนามิกตามคำร้องขอ

Resources ช่วยให้การเชื่อมต่อข้อมูลระหว่างโมเดลภาษาและระบบภายนอกเกิดขึ้นอย่างปลอดภัยและควบคุมได้

Prompt

Prompt คือเทมเพลตข้อความที่สามารถนำกลับมาใช้ซ้ำได้ ซึ่งช่วยให้ LLM สร้างข้อความตอบกลับที่มีโครงสร้าง ชัดเจน และตรง วัตถุประสงค์มากขึ้น

การทำงานของ Prompt ใน FastMCP

เมื่อไคลเอนต์ร้องขอ Prompt ใด ๆ

1. FastMCP จะค้นหานิยามของ Prompt ที่ตรงกับคำร้องขอ
2. ถ้ามีพารามิเตอร์ ระบบจะตรวจสอบความถูกต้องกับ signature ของฟังก์ชัน
3. ฟังก์ชันของคุณจะถูกเรียกใช้ด้วยค่าพารามิเตอร์ที่ผ่านการตรวจสอบแล้ว
4. ข้อความที่ถูกสร้างขึ้นจะถูกส่งกลับไปยัง LLM เพื่อใช้เป็นแนวทางในการสร้างคำตอบ

ประโยชน์ของ Prompt

- สร้าง **ข้อความต้นแบบ** ที่มีโครงสร้างแน่นอน
- ลดความซ้ำซ้อนของข้อความในหลายบริบท
- ช่วยให้ LLM **ตอบกลับได้อย่างสม่ำเสมอและตรงประเด็น** ไม่ว่าจะเรียกใช้จากไคลเอนต์ไหนหรือบริบทใดก็ตาม

Prompt ช่วยให้คุณสามารถออกแบบการสื่อสารกับ LLM อย่างมีระบบ และควบคุมทิศทางของข้อความตอบกลับได้ดีขึ้น

FastMCP คืออะไร

FastMCP คือเฟรมเวิร์กที่ช่วยให้คุณสามารถสร้าง MCP Server ได้อย่างสะดวกรวดเร็วใน Python โดยไม่ต้องจัดการกับความซับซ้อนของโปรโตคอล MCP โดยตรง มันช่วยลด “friction” ระหว่างนักพัฒนากับการนำ AI เข้ามาใช้งานจริง

ทำไมต้องใช้ FastMCP

- ช่วยให้คุณเขียนฟังก์ชันธรรมดาใน Python แล้ว “เปิดเผย” ให้เป็นเครื่องมือใน MCP ได้ทันที
- สร้าง schema และ metadata ให้โดยอัตโนมัติจาก type annotations และ docstrings
- จัดการ session, authentication และ JSON-RPC ให้เบื้องหลัง
- รองรับการประกอบ (composition) เซิร์ฟเวอร์หลายตัวเข้าด้วยกัน

โค้ดตัวอย่าง MCP Server


```

from fastmcp import FastMCP

mcp = FastMCP("Demo")

@mcp.tool
def add(a: int, b: int) -> int:
    """Add two numbers"""
    return a + b

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

```

ในตัวอย่างนี้ ฟังก์ชัน add จะกลายเป็น Tool ใน MCP server โดยไม่ต้องเขียนโค้ด schema หรือ JSON-RPC เองเลย

ความสำคัญของ FastMCP ในยุคของ LLM Application

ในโลกที่ LLM กลายเป็น platform runtime สำคัญ เช่น Copilot, ChatGPT plugins หรือ AI agents อย่าง AutoGPT, LangGraph, การมีมาตรฐานสำหรับ “การสื่อสารระหว่างโมเดลกับโลกภายนอก” เป็นเรื่องที่กำลังเป็นมาขึ้นเรื่อยๆ

FastMCP จึงเข้ามาตอบโจทย์นี้ด้วย

- การออกแบบที่เน้นนักพัฒนาเป็นศูนย์กลาง
- ระบบแบบ declarative สร้าง schema และการเชื่อมต่อให้แบบอัตโนมัติ
- สนับสนุนการ deploy ไปยัง production อย่างง่ายดาย
- ระบบที่รองรับการทดสอบ (test), การทำงานผ่านพร็อกซี (proxy), การประกอบเชิร์ฟเวอร์ (compose) และการเขียนเครื่องมือใหม่แบบไดนามิก (dynamic tool rewriting) อย่างครบวงจร

FastMCP ช่วยให้คุณ “เชื่อม LLM เข้ากับโลกจริง” โดยใช้โค้ด Python แบบธรรมดา

ในบทนี้ เราได้เรียนรู้ว่า

- MCP คือมาตรฐานกลางสำหรับเชื่อมต่อ LLM กับเครื่องมือและข้อมูล
- องค์ประกอบหลัก MCP Server ประกอบด้วย Tools, Resources และ Prompts
- FastMCP เป็นเฟรมเวิร์กที่ช่วยสร้าง MCP Server ได้อย่างรวดเร็วและปลอดภัย
- FastMCP 2.0 คือเวอร์ชันล่าสุดที่มีเครื่องมือครบถ้วน พร้อมสำหรับการใช้งานใน production

ในบทถัดไป เราจะพาคุณไปติดตั้ง FastMCP และเริ่มต้นเขียนเชิร์ฟเวอร์ MCP ตัวแรกของคุณ