Langchain vs MCP

Friday, 11 July 2025 7:06 PM

Similarities

Aspect	LangChain	MCP Server
Core Concept	Tools (external functions)	Tools (external functions)
Function Structure	Arguments, return values, descriptions	Arguments, return values, descriptions
Tool Definition Requirements	Function arguments, when to call, return values	Function arguments, when to call, return values
Description Importance	Critical for LLM to decide which tool to use	Critical for LLM to decide which tool to use
Collection Concept	Toolkits (collection of pre-built tools)	Servers (collection of tools)
Prompt Injection	Injects tool descriptions into LLM prompt	Injects tool descriptions into LLM prompt
Purpose	Enable AI models to call external functions	Enable AI models to call external functions
Developer-Written	Tools written by developers externally	Tools written by developers externally

Differences

Aspect	LangChain	MCP Server
Scope	Primarily tools (functions)	Tools + Resources + Prompts
Resource Types	Functions only	Functions, documents, PDFs, images, API calls
Integration Target	Binds directly to LLM	Binds to AI applications (Cursor, Windsurf, Claude)
Binding Method	bind_tools() method	Via MCP client to application
Architecture	Direct: LangChain → LLM	Layered: MCP Server → MCP Client → AI App → LLM
Abstraction Layers	Minimal abstraction	Multiple abstraction layers
Integration Level	Direct LLM integration	Application-level integration
Generalization	Focused approach	More generalized approach
Communication Flow	Direct tool binding	Server-client communication protocol

there is an official integration available! LangChain provides the **langchain-mcp-adapters** library that enables seamless integration between LangChain and MCP servers.