# Spring AI MCP Utilities

**前置知识：**

此处为语雀内容卡片，点击链接查看：<https://www.yuque.com/yinhuidong/ai/wwvdablypzeturhb>

此处为语雀内容卡片，点击链接查看：<https://www.yuque.com/yinhuidong/ai/wwt0g8mh6qqt04lb>

**MCP 实用工具为将模型上下文协议（Model Context Protocol，MCP）与 Spring AI 应用程序集成提供基础支持。**

* 这些实用工具使 Spring AI 的工具系统与 MCP 服务器之间能够实现无缝通信，支持同步和异步操作。

* 它们通常用于以编程方式配置和管理 MCP 客户端和服务器的交互。

* 如果想要更简化、更快速的配置方式，可以考虑使用启动器（boot starters）。

## 1. ToolCallback Utility

### 1.1. Tool Callback Adapter

系统提供了适配层，可以让 MCP 工具无缝地以 Spring AI 认可的接口形式运行，同时支持同步调用和异步调用，增强了灵活性和兼容性。

* 同步：

McpSyncClient mcpClient = // obtain MCP client  
Tool mcpTool = // obtain MCP tool definition  
ToolCallback callback = new SyncMcpToolCallback(mcpClient, mcpTool);  
  
// Use the tool through Spring AI's interfaces  
ToolDefinition definition = callback.getToolDefinition();  
String result = callback.call("{\"param\": \"value\"}");

* 异步：

McpAsyncClient mcpClient = // obtain MCP client  
Tool mcpTool = // obtain MCP tool definition  
ToolCallback callback = new AsyncMcpToolCallback(mcpClient, mcpTool);  
  
// Use the tool through Spring AI's interfaces  
ToolDefinition definition = callback.getToolDefinition();  
String result = callback.call("{\"param\": \"value\"}");

### 1.2. Tool Callback Providers

系统能够自动检测（发现）MCP 客户端所暴露的工具，并将这些工具注册或提供出来，方便服务器或其他组件调用和管理。

* 同步：

McpSyncClient mcpClient = // obtain MCP client  
ToolCallbackProvider provider = new SyncMcpToolCallbackProvider(mcpClient);  
  
// Get all available tools  
ToolCallback[] tools = provider.getToolCallbacks();

* 异步：

McpAsyncClient mcpClient = // obtain MCP client  
ToolCallbackProvider provider = new AsyncMcpToolCallbackProvider(mcpClient);  
  
// Get all available tools  
ToolCallback[] tools = provider.getToolCallbacks();

对于多个客户端：

List<McpAsyncClient> clients = // obtain list of clients  
Flux<ToolCallback> callbacks = AsyncMcpToolCallbackProvider.asyncToolCallbacks(clients);

## 2. McpToolUtils

### 2.1. ToolCallbacks 到 ToolSpecifications

把我们在 Spring AI 里定义的工具回调，自动转成符合 MCP 协议要求的工具规格，这样这些工具就能被 MCP 服务器识别和调用，实现两者的兼容和联动。

* 同步：

List<ToolCallback> toolCallbacks = // obtain tool callbacks  
List<SyncToolSpecifications> syncToolSpecs = McpToolUtils.toSyncToolSpecifications(toolCallbacks);

* 异步：

List<ToolCallback> toolCallbacks = // obtain tool callbacks  
List<AsyncToolSpecification> asyncToolSpecifications = McpToolUtils.toAsyncToolSpecifications(toolCallbacks);

然后可以使用 McpServer.AsyncSpecification 注册工具规格：

McpServer.AsyncSpecification asyncSpec = ...  
asyncSpec.tools(asyncToolSpecifications);

### 2.2. MCP Clients 到 ToolCallbacks

从 MCP 客户端获取ToolCallback：

* 同步：

List<McpSyncClient> syncClients = // obtain sync clients  
List<ToolCallback> syncCallbacks = McpToolUtils.getToolCallbacksFromSyncClients(syncClients);

* 异步：

List<McpAsyncClient> asyncClients = // obtain async clients  
List<ToolCallback> asyncCallbacks = McpToolUtils.getToolCallbacksFromAsyncClients(asyncClients);

## 3. 原生镜像支持

**McpHints 类为 MCP（模型上下文协议）模式类提供了 GraalVM 原生镜像（native image）所需的提示信息（hints）。**

具体来说：

* 在使用 GraalVM 构建原生镜像时，某些类的反射操作需要提前声明（注册反射提示），否则运行时可能会出错。

* McpHints 类会自动帮我们注册 MCP 模式类所需的所有反射提示，确保这些类在原生镜像中能够正常使用反射功能。

* 这样我们就不需要手动配置这些反射信息，简化了构建流程。

代码地址：

* server：[spring-ai-test/mcp-server-utility at main · huidongyin/spring-ai-test](https://github.com/huidongyin/spring-ai-test/tree/main/mcp-server-utility)

* client：[spring-ai-test/mcp-client-utility at main · huidongyin/spring-ai-test](https://github.com/huidongyin/spring-ai-test/tree/main/mcp-client-utility)