前言

模型情境協定（MCP）是一項革命性技術，為開發者提供了強大的工具與標準化介面，使其能更高效地建構和管理基於大型語言模型的應用。本書目的在於提供 MCP 領域的權威指引，協助讀者快速掌握 MCP 開發技巧。

本書採用循序漸進的結構設計，從 MCP 基礎知識到進階應用，逐步引導讀者深入理解 MCP 的工作原理與實現方法。無論是初次接觸 MCP 的初學者，還是希望提升技能的資深開發者，皆能從本書中獲得實用且深入的學習指導。

本書各章圍繞特定主題展開，從 MCP 的基本概念到實際應用案例，讀者可依自身需求與興趣選擇性的閱讀相關內容。本書涵蓋大量應用範例與詳細實作步驟，透過學習本書，讀者不僅可以掌握 MCP 的核心技術，還能在實際業務場景中靈活運用，打造更智慧、更高效的 AI 應用。

本書共分 8 章，每章內容大致如下：

- 第 1 章介紹 MCP 的定義、核心特點與優勢、技術架構、發展歷程及應用場景。
- 第 2 章講解 MCP 的核心架構、資源、提示、工具、取樣與根目錄等基礎知識。
- 第 3 章介紹 MCP SDK 的發展歷程、核心價值、多語言生態及快速入門方法。
- 第 4 章圍繞 Claude 桌面應用配置 MCP 伺服器，包括基礎配置、MCP 伺服器配置範例及常見問題排除。

- 第 5 章針對 MCP 伺服器開發，介紹相關基礎知識，並以天氣預報 MCP 伺服器為例講解開發流程。
- 第 6 章深入探討 MCP Inspector 工具的使用方法、核心功能及最佳實務。
- 第 7 章聚焦於 MCP 生態系統，包括主機應用、領域應用、開發者工具與服務，以及 MCP 廣場。
- 第 8 章分享 MCP 在高效軟體開發與創意內容生成中的應用實務。

本書目的與價值

本書目的在於提供全面性、系統性的 MCP 學習資源，協助讀者快速掌握 MCP 這一強大工具的使用方法。具體而言，本書目標為：

- 提供 MCP 技術的完整概述，包括其架構、組件與工作原理。
- 詳細介紹 MCP SDK 的使用方法，協助讀者快速上手。
- 探討 MCP 在各種應用場景中的實際應用，提供可複製的解決方案。
- 分享 MCP 開發的最佳實務與最佳化策略，協助讀者建構高品質的應用。

透過閱讀本書，讀者將獲得紮實的 MCP 技術基礎，掌握靈活運用這些知識解決實際問題的能力，進而在 AI 應用開發領域持續創新。

目標讀者

本書適合以下讀者群體：

- **軟體開發者**：希望利用 MCP 建構智慧應用或參與 MCP 開發與改善的開發人員，包括前端、後端、全端工程師及開源貢獻者。

- **AI 工程師**：需要深入了解如何有效整合與最佳化大型語言模型（LLM）應用的技術人員。
- **產品經理**：需要全面評估 MCP 技術潛力與應用場景的產品負責人。
- **技術愛好者**：渴望掌握尖端 AI 應用開發技術的自學者。
- **教育工作者**：在 AI 與軟體開發領域從事教學與研究工作的相關人員。
- **學生**：希望拓展 AI 應用開發能力的在校學生。

如何使用本書

為了獲得最佳學習效果，建議讀者依以下方式使用本書：

- **基礎學習**：若你是 MCP 初學者，建議從第 1 章開始依序閱讀，以建構完整的技術知識體系。
- **針對性學習**：若已有一定基礎，可直接閱讀感興趣的章節。
- **實務結合**：閱讀過程中，建議跟隨書中範例進行實際操作，以加深理解並強化技能。
- **參考使用**：在實際開發過程中，讀者可直接跳到相關章節查閱所需資訊。

本書中的程式碼範例與案例研究均經過精心設計，目的在於展示 MCP 的實際應用案例，讀者可依此進行實作練習，並根據自身需求進行適當調整。

Chapter

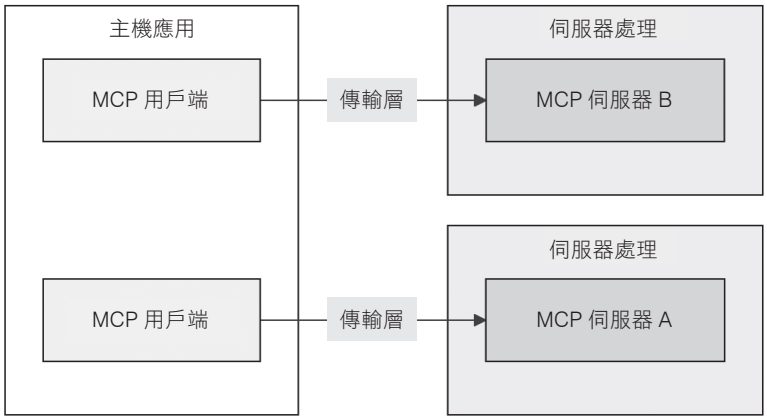
2

MCP 的基礎概念

透過第 1 章的學習，相信讀者已對 MCP 有了初步認識，了解其定義、特點、發展歷程及應用場景。本章將帶領讀者更深入探索 MCP 的基礎概念，包括其核心架構、資源、提示、工具、取樣、根目錄。這些概念構成 MCP 的基礎框架，理解它們對於高效率使用 MCP 實現應用開發至關重要。

2.1 MCP 核心架構

MCP 依照用戶端 - 伺服器架構，其中主機應用可連結多個伺服器，如圖 2-1 所示。架構中的每個組件在 LLM 與外部資源互動過程中扮演特定角色。



▲ 圖 2-1

2.1.1 核心組件

MCP 建立在靈活、可擴展的架構之上，目的在於實現 LLM 應用與整合間的無縫通訊。該架構中的元素主要分為 3 個核心角色，即 MCP 主機、MCP 伺服器與 MCP 用戶端。

- **MCP 主機**：使用 MCP 連結各種資源的 LLM 應用（如 Claude 桌面應用、Cursor、Windsurf 整合開發環境）。
- **MCP 伺服器**：基於 MCP 揭示特定功能的輕量級程式。
- **MCP 用戶端**：主機內部的必要組件，負責與單一 MCP 伺服器維持一對一的連結。

伺服器以特殊的 MIME 類型（text/event-stream）回應請求，每條訊息由一對換行符號分隔，以文字格式發送事件流程。SSE 特別適用於需要即時更新的應用場景。在 HTTP SSE 機制下，MCP 用戶端透過 HTTP POST 請求向 MCP 伺服器發送訊息，伺服器則透過 SSE 訊息通知用戶端。

2.1.2 連結的生命週期

每個 MCP 連結依照以下生命週期，分別為初始化、訊息交換與終止。

初始化

MCP 的初始化階段是用戶端與伺服器之間的第一次互動，包含 3 個關鍵步驟：

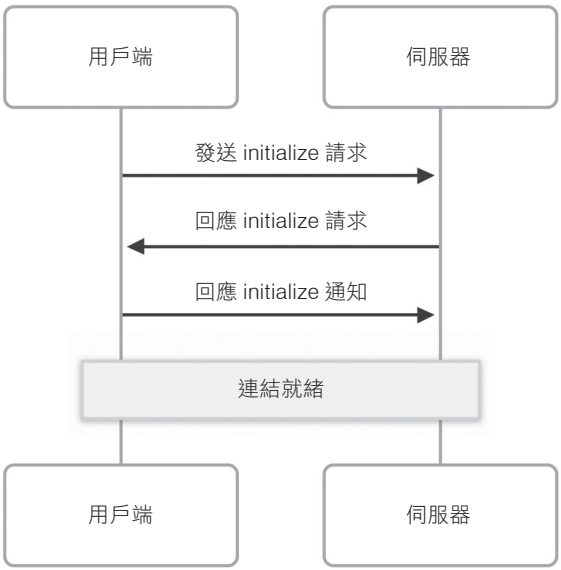
STEP 01 用戶端必須首先發送 initialize 請求，包含其支援的協定版本、用戶端能力與實現資訊。

STEP 02 伺服器必須發送回應，提供自身的功能與實現資訊，可能會協商使用不同協定版本。

STEP 03 成功初始化後，用戶端必須發送 initialized 通知，表示已準備好開始正常操作。

此過程讓用戶端與伺服器能建立協定版本相容性、交換並協商各自功能，以及分享實現細節，如圖 2-2 所示。

在初始化階段，用戶端與伺服器完成主要完成版本與功能協商。在 initialize 請求中，用戶端必須發送其支援的協定版本，即用戶端支援的最新版本。如果伺服器支援請求的協定版本，它必須以相同的版本回應。否則，伺服器必須回應它支援的另一個協定版本，即伺服器支援的最新版本。如果用戶端不支援伺服器回應中的協定版本，它應該斷開連結。



▲ 圖 2-2

在初始化階段，用戶端與伺服器交換功能資訊，使雙方知曉對方所支援的功能。以下是一個用戶端發送的 initialize 請求範例：

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "initialize",
    "params": {
        "protocolVersion": "2025-03-26",
        "capabilities": {
            "roots": {
                "listChanged": true
            },
            "sampling": {}
        },
        "clientInfo": {
            "name": "DemoClient",
            "version": "1.0.0"
        }
    }
}
```

以下是伺服器 initialized 回應範例：

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": {
        "protocolVersion": "2025-03-26",
        "capabilities": {
            "logging": {},
            "prompts": {
                "listChanged": true
            },
            "resources": {
                "subscribe": true,
                "listChanged": true
            },
            "tools": {
                "listChanged": true
            }
        },
        "serverInfo": {
            "name": "DemoServer",
            "version": "1.0.0"
        }
    }
}
```

1. 版本協商（Version Negotiation）

在 initialize 請求中，用戶端必須發送其支援的協定版本，並且應當是用戶端支持的最新版本。若伺服器支援所請求的協定版本，則必須以相同的版本做出回應。否則，伺服器必須回應其支援的另一個協定版本，並且應當是伺服器支持的最新版本。若用戶端不支援伺服器回應中的協定版本，則應當斷開連接。

2. 能力協商（Capability Negotiation）

能力協商中的「能力」分為用戶端能力與伺服器能力。在能力協商階段，用戶端與伺服器分別公告自己所支援的選填特性。請參考表 2-1 瞭解目前 MCP 中包含的能力及其描述。

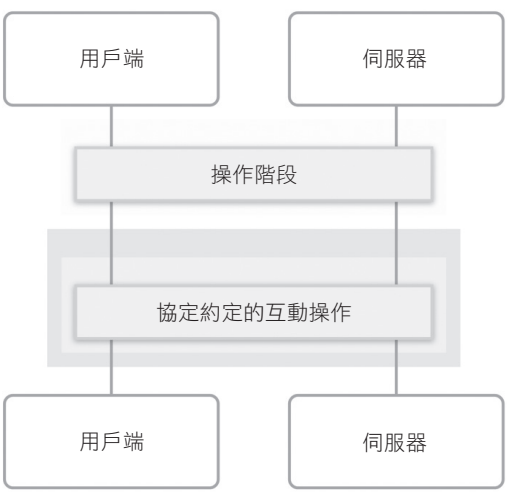
▼ 表 2-1 MCP 能力及其描述

分類	能力	描述
用戶端	根目錄（roots）	提供檔案系統根目錄的能力
	取樣（sampling）	支援 LLM 的取樣請求
	試驗性能力（experimental）	支援試驗性的非標準能力
伺服器	提示（prompts）	提供提示範本
	資源（resources）	提供資源
	工具（tools）	提供可呼叫工具
	日誌（logging）	發送結構化日誌訊息
	試驗性能力（experimental）	支援試驗性的非標準能力

訊息交換

初始化後，用戶端與伺服器可進行以下操作，如圖 2-3 所示。

- 發送請求並接收回應
- 發送通知
- 報告錯誤



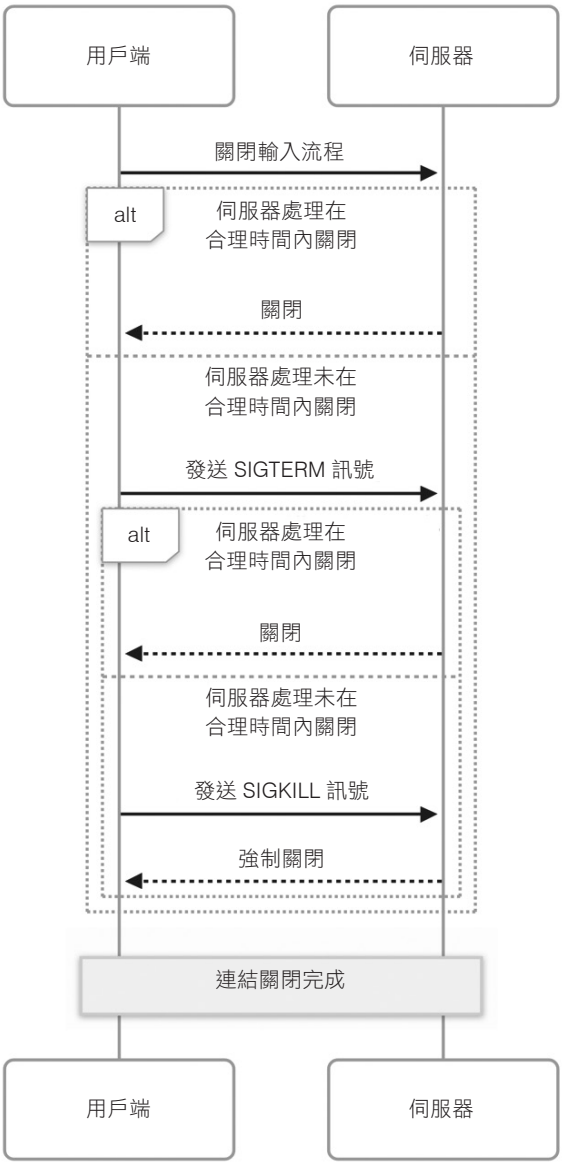
▲ 圖 2-3

終止

用戶端或伺服器皆可終止連結。目前 MCP 未定義明確的連結關閉訊息，用戶端與伺服器應採用底層傳輸協定相應的機制來通知連結關閉。

1. Stdio

用戶端應透過以下步驟關閉連結，如圖 2-4 所示。



▲ 圖 2-4

STEP 01 關閉子程序的輸入流程。

STEP 02 等待伺服器程序關閉，若伺服器在合理時間內未關閉，則發送 SIGTERM 信號。

STEP 03 若伺服器在接收 SIGTERM 信號後的合理時間內仍未關閉，則發送 SIGKILL 信號。

伺服器可透過關閉其對用戶端的輸出流程並退出來關閉連結。

2. HTTP

在以 HTTP 為傳輸協定的連結中，關閉 HTTP 連結意味著關閉 MCP 連結。任何一方皆可關閉 HTTP 連結，關閉連結意味著不再發送訊息或接收來自對端的訊息。此適用於用戶端與伺服器，雙方皆可根據需要選擇關閉連結，終止當前通訊。

2.1.3 錯誤處理

MCP 定義了標準錯誤碼與處理機制。伺服器特定的錯誤碼應大於 -32000。錯誤透過以下方式傳播：

- 傳輸層錯誤事件
- 協定層錯誤處理程式
- 請求的錯誤回應

以 TypeScript SDK 的原始程式碼為例，以下程式碼片段定義了錯誤碼。

```
export enum ErrorCode {
  // SDK error codes
  ConnectionClosed = -32000,
  RequestTimeout = -32001,
  // Standard JSON-RPC error codes
  ParseError = -32700,
```

```
  InvalidRequest = -32600,
  MethodNotFound = -32601,
  InvalidParams = -32602,
  InternalError = -32603,
}
```

2.2 資源

在人工智慧與大型語言模型的應用生態中，擴展 LLM 的知識範圍與互動能力至關重要。MCP 透過「資源」這一核心用語，為此挑戰提供了優秀的解決方案。資源（Resource）是 MCP 中的基礎建構模塊，允許伺服器向用戶端揭示各種資料與內容，這些內容可被讀取並用作 LLM 互動的情境。透過資源機制，LLM 能取得與處理即時、多樣的資訊，極大拓展其應用邊界。

2.2.1 資源概述

當思考 LLM 的限制時，一個關鍵挑戰是其對訓練資料外資訊的取得能力。資源機制正是為彌合這一鴻溝而設計。在 MCP 框架下，資源代表伺服器希望提供給用戶端的任何類型資料，其範圍廣泛，包括但不限於以下類型：

- **檔案內容**：從原始碼到設定檔，從文件到日誌。
- **資料庫紀錄**：結構化資料的動態取得。
- **API 回應**：來自外部服務的即時資訊。
- **即時系統資料**：作業系統狀態、效能指標等。
- **截圖與影像**：視覺化內容的取得。
- **日誌檔案**：系統與應用行為的紀錄。

這種靈活性使 MCP 幾乎能適應任何類型的資料需求，每個資源皆由唯一的 URI（統一資源識別符號）標識，內容可以是文字或二進位資料。這種設計為將各種資料來源與 LLM 整合提供了一個統一的介面。

支援資源的 MCP 伺服器必須宣告資源能力，資源能力支援兩種選填特性。

- **subscribe**：用於標識用戶端是否能訂閱資源變更通知。
- **listChanged**：用於標識伺服器是否會在資源清單變更時發送通知。

小提示

這兩個特性都是選填的，MCP 伺服器可以兩個都支援，也可以支援其一或者都不支援。

以下為 MCP 伺服器支援資源的 4 種情況。

兩個特性都不支援。

```
{
  "capabilities": {
    "resources": {}
  }
}
僅支援資源變更通知
{
  "capabilities": {
    "resources": {
      "subscribe": true
    }
  }
}
僅支援資源變更通知
{
  "capabilities": {
    "resources": {
      "listChanged": true
    }
  }
```

資源內容變更的工作流程如下：

- **訂閱**：用戶端利用資源 URI，透過 JSON-RPC 方法 resources/subscribe 訂閱特定資源內容的變更。
- **通知**：當資源內容變更時，伺服器發送 notifications/resources/updated 通知，告知用戶端資源已更新。
- **讀取**：用戶端收到通知後，可選擇透過 resources/read 方法讀取最新的資源資料。
- **取消訂閱**：當用戶端不再需要接收變更通知時，可透過 resources/unsubscribe 方法取消訂閱。

這種「訂閱 - 通知」模式讓用戶端能高效追蹤重要資源的變更，而無須頻繁輪詢所有資源。對於即時性要求高的應用場景，此機制尤為重要。

2.2.7 實現一個支援資源資料的 MCP 伺服器

理論知識需透過實務強化。以下透過一個 TypeScript 程式碼片段，展示一個支援資源資料的 MCP 伺服器的基本實現。此範例雖簡單，但包含資源功能的核心組件，有助於理解 MCP 資源機制的實際應用。

```
const server = new Server({
  name: "document-server",
  version: "1.0.0"
}, {
  capabilities: {
    resources: {}
  }
});
const resource_uri = "file:///project/src/docs/intro.md";
// 列出可用資源
server.setRequestHandler(ListResourcesRequestSchema, async () => {
  return {
    resources: [
      {
        uri: resource_uri,
        name: "Project document",
        mimeType: "text/markdown"
```

```
      }
    ]
  };
});
// 讀取資源內容
server.setRequestHandler(ReadResourceRequestSchema, async (request) => {
  const uri = request.params.uri;
  if (uri === resource_uri) {
    const markdownContent = await readLogFile();
    return {
      contents: [
        {
          uri,
          mimeType: "text/markdown",
          text: markdownContent
        }
      ]
    };
  }
  throw new Error("Resource not found");
});
```

此範例伺服器實現了兩個關鍵的請求處理器：

- **資源清單處理器**：實現 resources/list 方法，回應中列出一個 Markdown 文件資源。在實際應用中，此處理器可能回傳更複雜的資源清單。
- **資源讀取處理器**：實現 resources/read 方法，當請求的 URI 匹配伺服器已知的資源時，讀取並回傳該 Markdown 文件的內容。若資源不存在，則透過錯誤訊息通知用戶端。

雖然這是一個簡化範例，但展示了 MCP 資源功能的基本框架。在實際應用中，伺服器可能支援更多功能，如資源模板、內容訂閱與變更通知等。

2.2.8 安全性如何保障

在實際運作場景中，MCP 伺服器發布的資源可能隨時變更，這些資源可能包含具隱私性、敏感性的資料。

資源機制使 MCP 具備強大功能，同時也引入一系列安全挑戰。MCP 伺服器發布的資源可能包含敏感資料，或提供對關鍵系統的存取權限。因此，在實現資源功能時，安全性必須是首要考量。

在發布資源時，應考慮以下安全因素，以確保資料的可存取性、安全性與隱私性：

- 驗證所有資源 URI。
- 部署適當、必要的存取控制。
- 清理檔案路徑以避免重複目錄。
- 謹慎處理二進位資料。
- 考慮對資源讀取進行限流。
- 稽核資源存取。
- 加密傳輸中的敏感資料。
- 驗證 MIME 類型。
- 為長時間執行的讀取進行超時控制。
- 適當的資源清理。

2.3 提示

提示（prompt）是 AI 時代最熱門的名詞。什麼是提示？這是在 AI 應用中使用的一種輸入形式，協助使用者與 DeepSeek、通義千問等 AI 大型模型進行有效溝通。透過提示，使用者可用自然語言表達需求，使 AI 理解並執行相應任務。

提示的形式可簡單（如日常問題或簡短指令），也可較為複雜（包含詳細任務描述或特定要求）。如同日常交談，使用者可透過提示向 AI 詢問資訊、請求協助，或進行持續對話。一個合適的提示通常包含具體需求與必要的情境資訊，這些資訊能幫助 AI 更好理解使用者的意圖。

提示也是 MCP 中的核心概念之一。MCP 定義的提示是可重用的模板機制，用於引導 LLM 有效與 MCP 伺服器互動。結構化的指令或情境有助於 LLM 理解所需任務或資訊。

小提示

本節介紹的提示，特指 MCP 中定義的提示模板機制。

2.3.1 提示概述

提示採用預定義模板形式，根據實際需求接收動態參數，使提示能靈活適應不同場景與需求。在執行時，模板可自動整合來自各種資源的情境資訊，確保 AI 模型能獲得更完整的背景資訊來處理任務。

這種模板化的提示設計支援多個互動環節的串聯，將複雜任務分解為一系列連貫步驟。透過預設的工作流程指引，引導 AI 模型按特定路徑完成任務，保證輸出結果的品質與一致性。

在實際應用中，提示模板可透過斜線命令等使用者友善的 UI 元素呈現，方便使用者快速呼叫與使用。這種設計既保持提示的強大功能，又提供簡單直覺的使用體驗。

與資源能力相同，支援提示的 MCP 伺服器必須宣告提示能力，提示能力支援唯一的特性 listChanged。範例如下：

```
{
  "capabilities": {
    "prompts": {
      "listChanged": true
    }
  }
}
```

在上述範例中，特性 listChanged 用於標識是否在可用提示清單發生變更時發送通知。

2.3.2 提示結構

每個提示結構如下：

```
{
  name: string;
  description?: string;
  arguments?: [
    {
      name: string;
      description?: string;
      required?: boolean;
    }
  ]
}
```

每個提示必須有一個唯一的名稱作為識別符號。提示可附帶一段描述以解釋其用途與功能，雖然並非必要，但強烈建議開發者提供描述，以幫助用戶端更好理解提示。提示還可包含參數清單，每個參數有特定的名稱與描述，參數可設為必填或選填。部分提示無須參數，因此提示參數清單也非必要。

以下是提示範例：

```
{
  "name": "translate",
  "description": "Asks the LLM to translate the given text to the expected language",
  "arguments": [
    {
      "name": "text",
      "description": "The text to translate",
      "required": true
    },
    {
      "name": "language",
      "description": "The expected language",
      "required": true
    }
  ]
}
```

支援取樣的 MCP 用戶端必須宣告取樣能力。取樣能力的宣告範例如下：

```
{
  "capabilities": {
    "sampling": {}
  }
}
```

2.5.1 取樣的工作原理

取樣功能使伺服器能向用戶端的 LLM 請求生成內容，然後使用這些生成內容執行更複雜的任務，其設計依照「安全優先、人機協同」原則。

特別重要的是，在安全與信任方面，MCP 規範強調人類參與循環（human in the loop），以便進行審查或拒絕取樣請求。用戶端應用應提供直覺的使用者介面，讓使用者能輕鬆審查請求，在發送前查看與編輯提示，並在結果回傳伺服器前進行審核。

具體而言，取樣流程依照以下步驟：

STEP 01 **發起請求**：伺服器向用戶端發送 sampling/createMessage 請求，包含所需的訊息內容與取樣參數。

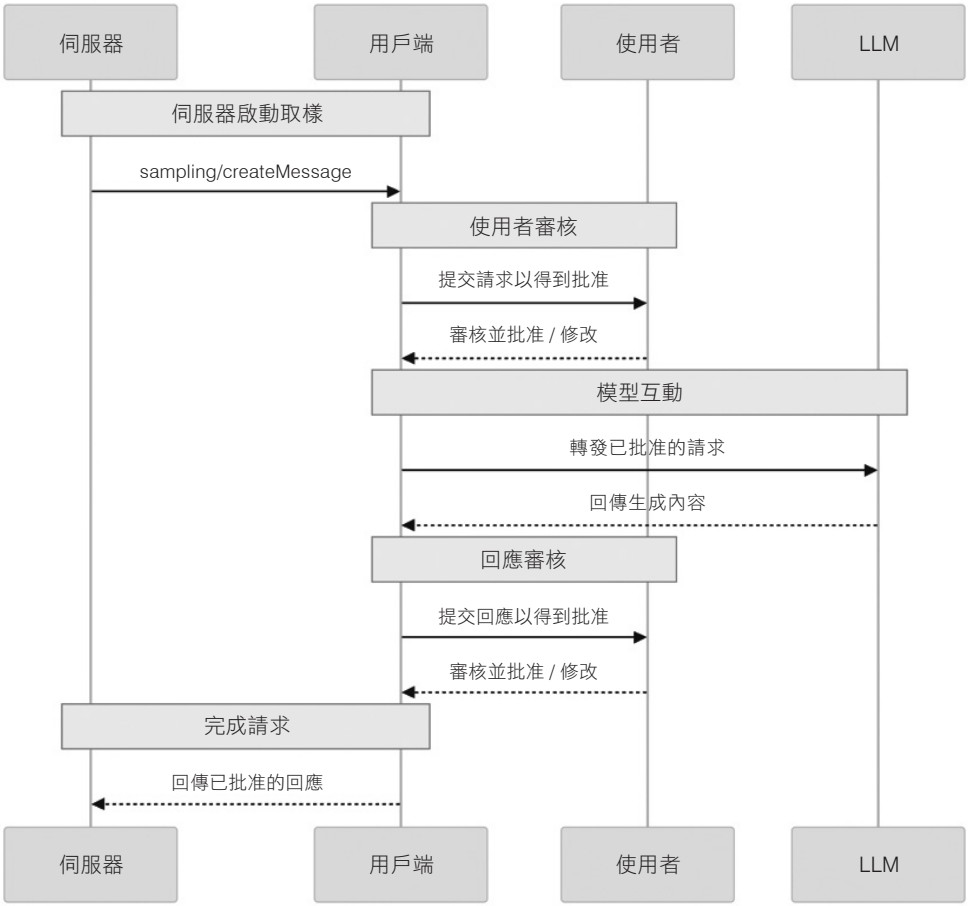
STEP 02 **使用者審核**：用戶端向使用者展示請求內容，以便審核或修改請求內容，確保安全性和隱私保護。

STEP 03 **LLM 生成內容**：用戶端將經使用者審批的請求轉發給 LLM，取得生成結果。此步驟即為真正的「取樣」過程，LLM 根據輸入生成相應輸出。

STEP 04 **回應審核**：用戶端向使用者展示生成結果，以便審核或修改內容，確保其符合安全與隱私要求。

STEP 05 **回傳結果**：用戶端將經使用者審批的結果回傳給伺服器，伺服器可基於這些結果執行後續操作。

取樣流程時序圖如圖 2-5 所示。



▲ 圖 2-5

這設計使伺服器能利用用戶端 LLM 的強大生成能力，同時透過使用者介入確保整個過程的安全性，用戶端保持對流程的控制，可以拒絕不安全或不適當的請求，保護使用者的隱私和安全。