

開發者的匠心

Poy Chang



什麼是匠心？

...

不是“降薪”噲



所謂的**匠心**，就是工匠們在製作物品時所
傾注的心血、專注與技藝，體現了工匠們
對技藝的熱愛，對品質的執著追求，以及
對細節精益求精的態度。

...



我喜歡解決問題

我是細節控

我想要變強

用 AI 產生程式能對付問題
但對付不了
想要深入探究某件事的心



Poy Chang

任職於全美百大私人企業，負責企業內部商業應用解決方案的設計與開發，專注於 Azure、.NET 等相關技術研究

- ✓ STUDY4 社群核心成員
- ✓ Microsoft MVP 微軟最有價值專家
- ✓ .NET Conf Taiwan 的一份子
- ✓ 著作《.NET Conf 總召真心話》
- ✓ 譯著《企業級軟體架構模式》
- ✓ 譯著《無瑕的程式碼 函數式設計篇》



問題是驅動學習的動力

開發者面對問題的方式與一般人不同：

1. 問題不是阻礙，而是挑戰的契機
2. 錯誤和不符合預期的訊息，都是學習的起點
 - 驅使我們去搜尋資料、嘗試新方法、拆解複雜的情境
3. 結果不是解決方案，是拓展的知識邊界



問題解決是從**初始狀態**出發，使用既有概念和策略，尋找通往目標狀態的路徑，在結束時**內化新概念**、**狀態改變**

案例

使用 Docker CLI 維運散落在多台伺服器的容器時，經常會要使用類似以下的指令來對遠端進行操作：

```
$ docker --host 192.168.64.5 ps  
$ docker --host 192.168.64.12 ps  
$ docker --host 192.168.64.43 ps
```

在慣用的 PowerShell 環境下，可以怎麼做讓維運更順暢？

案例

```
function Add-DynamicFunction {
```

PowerShell

```
    Param(
```

```
        [Parameter(Mandatory = $true, Position = 0, HelpMessage = "Function Name")] [string]$FuncName,
```

```
        [Parameter(Mandatory = $true, Position = 1, HelpMessage = "Function Action")] [string]$FuncAction
```

```
    )
```

```
    Set-Variable -name Func -value "function global:$($FuncName)() { $($FuncAction) }"
```

```
    Invoke-Expression $Func
```

```
    # 執行此函式時，相當於在全域建立一個如下的函式，供使用者呼叫
```

```
    # function global:FuncName() { <# Do Function Action #> }
```

```
}
```


案例

```
$SERVER_LIST = @(
    [PSCustomObject]@{ Name = "docker1"; Host = "192.168.64.5"; },
    [PSCustomObject]@{ Name = "docker2"; Host = "192.168.64.12"; },
    [PSCustomObject]@{ Name = "docker3"; Host = "192.168.64.43"; });
```

```
$SERVER_LIST.ForEach({
    Add-DynamicFunction `
        -FuncName $_.Name `
        -FuncAction "docker -H $($_.Host) $(' $args ')"
})
```

啟動 PowerShell 時載入此段程式，即可使用以下方式操作，甚至搭配管線指令做後續動作

```
# docker1 ps | grep some-webapp
```

```
# docker2 ps | grep some-webapi
```

PowerShell

對於開發者而言，**匠心**經常出現在
當我們**極度**想要...

純熟的使用

工具

輕鬆的解決

問題

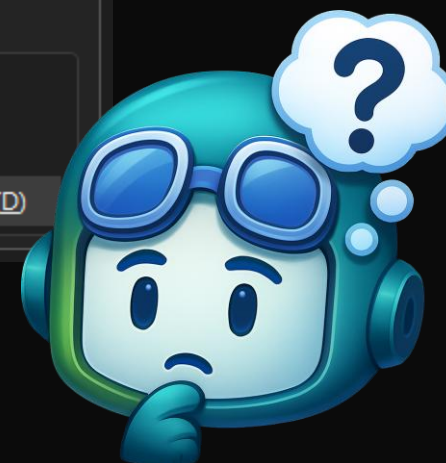
充分掌控的

方案

學習是成為匠者的過程

開發者的學習可以這麼做：

1. 過去，看文件、前人的程式碼，學習用程式產出解決方案
2. 後來，用 SQ 學習如何寫出好一點的程式
3. 現在，可以用 AI 精進寫出來的程式
4. 精進的過程可能是 AI 代勞，我們從 AI 的勞務中學習



GitHub Copilot 聊天

效能分析工具

這段程式碼怎麼寫可以效能與可讀性兼備？
> 1 個參考

分析工具
> 1 個參考

以下是兼顧效能與可讀性的建議，照你目前的用途由易到難：

- 保持現狀（可讀性佳且已相當快）
 - 你現在用 `JsonDocument.Parse(str)` 後再 `JsonSerializer.Serialize(doc.RootElement, options)` 是走 `JsonElement` 的快路徑，已足夠高效且簡潔。
 - `JsonSerializerOptions` 以 `static readonly` 快取很好。
- 需要再擠性能時才做的小改
 - 用 `Utf8JsonWriter + doc.RootElement.WriteTo(writer)` 輸出：
 - 優點：少一層 `JsonSerializer` 的泛型與管線，對大字串略少配置與少量 CPU。
 - 缺點：程式碼稍冗長；要自己建 `ArrayBufferWriter<byte>` 和 `JsonWriterOptions`，最後 `Encoding.UTF8.GetString(...)`。
 - 新增可重用的 `overload`，避免重複剖析：
 - `string ToIndented(this JsonElement element) / string ToMinify(this JsonElement element)`
 - 若呼叫端已經有 `JsonDocument/JsonElement`，可直接格式化，省一次 `Parse`。

我可以先用 BenchmarkDotNet 幫你比較兩種實作（現行 `Serialize(JsonElement)` vs `Utf8JsonWriter + WriteTo`），再據數據給出最優解。你若同意，我會照下面步驟進行：

1. 建立基準測試：準備小、中、大三種 JSON 輸入，實測兩種寫法（編排/壓縮各一）。
2. 執行基準測試並回報配置數與時間。
3. 根據結果給出是否值得改成 `Utf8JsonWriter` 的結論與最小變更範例。

要我現在建立並執行基準測試嗎？

+

📄 使用中文件 ✕

詢問 Copilot 或使用 @workspace

📄

⚡ 代理程式 ▾

GitHub Copilot 聊天 方案總管 Git 變更

Tweaking tips



持續探索的三層挑戰

「時間花在哪裡，成就就在哪裡」

- 這句話的重點看似是時間，但真正重要的是後面三個字在哪裡
- 三層挑戰就是去尋找在哪裡，而在哪裡也許就是匠心所在

概念式
學習

往下挖
三層

可能性
探索

案例

LLM 能讀懂人說的話，也懂得說人看得懂的話，但是應用程式卻無法根據 LLM 的回答，接續往下執行，因為程式需要：

- 動作指令要**明確**
- 資料型別要**正確**
- 結果要**能夠預期**

自然語言的回應太自然了，那些 MCP 的功能呼叫是怎麼做到的？

Assistant > Hi there! How can I help you today?

User > echo "my word"

Request Body:

```
{"tools":[{"function":{"description":"Echoes the message back to the client.,"name":"McpTools-echo","strict":false,"parameters":{"type":"object","required":["message"],"properties":{"message":{"type":"string"}}}},{"type":"function"},{"function":{"description":"Echoes in reverses the message.,"name":"McpTools-reveres_echo","strict":false,"parameters":{"type":"object","required":["message"],"properties":{"message":{"type":"string"}}}},{"type":"function"}],"messages":[{"role":"system","content":"你是一位 Model Context Protocol 工具助理，會根據使用者輸入決定是否要使用 tool 來回答問題。"}, {"role":"user","content":"hi"}, {"role":"assistant","content":"Hi there! How can I help you today?"}, {"role":"user","content":"echo \"my word\""}], "model":"BTIA-Agent-gpt-5", "tool_choice":"auto"}
```

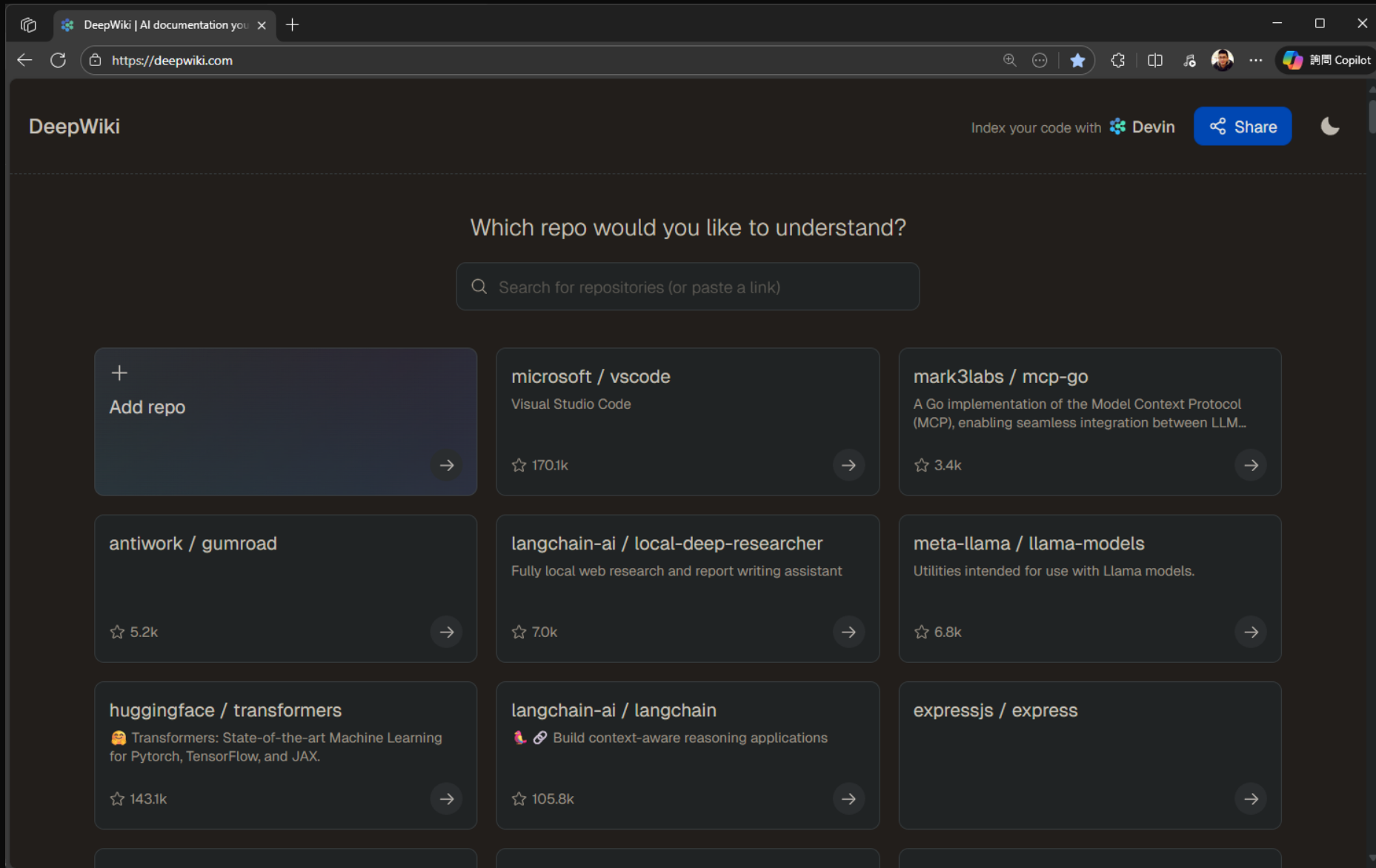
Response Body:

```
{"choices":[{"content_filter_results":{},"finish_reason":"tool_calls","index":0,"logprobs":null,"message":{"annotations":[],"content":null,"refusal":null,"role":"assistant","tool_calls":[{"function":{"arguments":{"message":"my word"},"name":"McpTools-echo"},"id":"call_03cpwGcS5Tqx3oeA7a0gLPqA","type":"function"}]}],"created":1758872032,"id":"chatcmpl-CJxFgWfIfUThIsn04GazdpQC01kQo","model":"gpt-5-2025-08-07","object":"chat.completion","prompt_filter_results":[{"prompt_index":0,"content_filter_results":{"hate":{"filtered":false,"severity":"safe"},"self_harm":{"filtered":false,"severity":"safe"},"sexual":{"filtered":false,"severity":"safe"},"violence":{"filtered":false,"severity":"safe"}}}], "system_fingerprint":null,"usage":{"completion_tokens":415,"completion_tokens_details":{"accepted_prediction_tokens":0,"audio_tokens":0,"reasoning_tokens":384,"rejected_prediction_tokens":0},"prompt_tokens":215,"prompt_tokens_details":{"audio_tokens":0,"cached_tokens":0},"total_tokens":630}}
```


如何看深一點？

How to understand deeper?

...



Search | DeepWiki

https://deepwiki.com/search/sdkjson-schema_e852d9e2-c79e-49d9-be17-98ffd9649c64

Index your code with Devin

Share

← return to modelcontextprotocol/csharp-sdk

此SDK是如何將函數的參數型別轉成JSON Schema ?

Fast

Searched across modelcontextprotocol/csharp-sdk

Go deeper

您詢問的是此 C# SDK 如何將函數的參數型別轉換成 JSON Schema。這個轉換過程主要發生在 `AIFunctionFactory.Create` 方法中，該方法會使用 `Microsoft.Extensions.AI` 框架來自動生成 JSON Schema。

JSON Schema 生成流程

此 SDK 使用 `AIFunction.JsonSchema` 屬性來獲取輸入參數的 JSON

modelcontextprotocol/csharp-sdk src/ModelContextProtocol.Core/Server/AIFunctionMcpServerTool.cs

```
77     Description = options?.Description,
78     MarshalResult = static (result, _, cancellationToken) => new ValueTask<object?>(result),
79     SerializerOptions = options?.SerializerOptions ?? McpJsonUtilities.DefaultOptions,
80     JsonSchemaCreateOptions = options?.SchemaCreateOptions,
81     ConfigureParameterBinding = pi =>
82     {
83         if (RequestServiceProvider<CallToolRequestParams>.IsAugmentedWith(pi.ParameterType) ||
84             (options?.Services?.GetService<IServiceProviderIsService>() is { } ispis &&
85              ispis.IsService(pi.ParameterType)))
86         {
87             return new()
88             {
89                 ExcludeFromSchema = true,
90                 BindParameter = (pi, args) =>
91                     args.Services?.GetService(pi.ParameterType) ??
92                     (pi.HasDefaultValue ? null :
93                      throw new ArgumentException("No service of the requested type was found.")),
94             };
95         }
96     }
97     if (pi.GetCustomAttribute<FromKeyedServicesAttribute>() is { } keyedAttr)
98     {
99         return new()
100         {
101             ExcludeFromSchema = true,
102             BindParameter = (pi, args) =>
103                 (args?.Services as IKeyedServiceProvider)?.GetKeyedService(pi.ParameterType, keyedAttr) ??
104                 (pi.HasDefaultValue ? null :
105                  throw new ArgumentException("No service of the requested type was found.")),
106         };
107     }
108     return default;
109 }
```

Notes

此轉換過程完全依賴 `Microsoft.Extensions.AI` 框架的 `AIFunction` 實作來生成 JSON Schema。SDK 主要負責配置參數綁定策略，決定哪些參數應該包含在 schema 中，以及提供自訂 schema 生成的選項。對於資源和提示，也使用相同的機制進行參數 schema 生成。

Wiki pages you might want to explore:

- [Server Prompts and Resources \(modelcontextprotocol/csharp-sdk\)](#)
- [AI Function Integration \(modelcontextprotocol/csharp-sdk\)](#)

Copy Response

Copy thread

Share

Ask a follow-up question

Deep Research ☐

→

```
89     ExcludeFromSchema = true,
90     BindParameter = (pi, args) =>
91         args.Services?.GetService(pi.ParameterType) ??
92         (pi.HasDefaultValue ? null :
93          throw new ArgumentException("No service of the requested type was found.")),
94     };
95 }
96
97 if (pi.GetCustomAttribute<FromKeyedServicesAttribute>() is { } keyedAttr)
98 {
99     return new()
100     {
101         ExcludeFromSchema = true,
102         BindParameter = (pi, args) =>
103             (args?.Services as IKeyedServiceProvider)?.GetKeyedService(pi.ParameterType, keyedAttr) ??
104             (pi.HasDefaultValue ? null :
105              throw new ArgumentException("No service of the requested type was found.")),
106     };
107 }
108 return default;
109 }
```

DeepWiki

Index your code with [Devin](#) [Share](#)

← return to modelcontextprotocol/csharp-sdk

此SDK是如何將函數的參數型別轉成JSON Schema ?

Fast

Searched across modelcontextprotocol/csharp-sdk [Go deeper](#)

您詢問的是此 C# SDK 如何將函數的參數型別轉換成 JSON Schema。這個轉換過程主要發生在 `AIFunctionFactory.Create` 方法中，該方法會使用 `Microsoft.Extensions.AI` 框架來生成 Schema。

JSON Schema 的可選參數

此 SDK 使用

```
modelcontextprotocol/csharp-sdk src/ModelContextProtocol.Core/Server/AIFunctionMcpServerTool.cs
77 Description = options?.Description,
78 MarshalResult = static (result, _, cancellationToken) => new ValueTask<object?>(result),
79 SerializerOptions = options?.SerializerOptions ?? McpJsonUtilities.DefaultOptions,
80 JsonSchemaCreateOptions = options?.SchemaCreateOptions,
81 ConfigureParameterBinding = pi =>
82 {
83     if (RequestServiceProvider<CallToolRequestParams>.IsAugmentedWith(pi.ParameterType) ||
84         (options?.Services?.GetService<IServiceProviderIsService>() is { } ispis &&
85             ispis.IsService(pi.ParameterType)))
86     {
87         return new()
88         {
89             ExcludeFromSchema = true,
90             BindParameter = (pi, args) =>
91                 args.Services?.GetService(pi.ParameterType) ??
92                 (pi.HasDefaultValue ? null :
93                     throw new ArgumentException("No service of the requested type was found.")),
94             };
95     }
96     if (pi.GetCustomAttribute<FromKeyedServicesAttribute>() is { } keyedAttr)
97     {
98         return new()
99         {
100             ExcludeFromSchema = true,
101             BindParameter = (pi, args) =>
102                 (args?.Services as IKeyedServiceProvider)?.GetKeyedService(pi.ParameterType, keyedAttr) ??
103                 (pi.HasDefaultValue ? null :
104                     throw new ArgumentException("No service of the requested type was found.")),
105             };
106     }
107     return default;
108 }
```

Not found

此轉完全依賴 `Microsoft.Extensions.AI` 框架的 `AIFunctionFactory.Create` 方法。該方法會生成 JSON Schema。SDK 主要負責配置參數綁定策略，方法中，以及提供自訂 schema 生成的選項。該方法會使用 `Microsoft.Extensions.AI` 框架來生成 Schema。

Wiki page: [AI Function Factory \(modelcontextprotocol/csharp-sdk\)](#)

- [Server Function \(modelcontextprotocol/csharp-sdk\)](#)
- [AI Function Factory \(modelcontextprotocol/csharp-sdk\)](#)

What want to explore:

[Copy Response](#) [Copy thread](#) [Share](#)

Ask a follow-up question

Deep Research ☐

→

DeepWiki

Index your code with Devin Share

← return to dotnet/extensions

我想要了解 Microsoft.Extensions.AI 這個命名空間下，能將型別轉成 Json Schema 的機制

Fast

Searched across dotnet/extensions Go deeper

Answer

您詢問的是 Microsoft.Extensions.AI 命名空間中將 .NET 型別轉換為 JSON Schema 的機制。這個功能主要由 AIJsonUtilities 類別提供，特別是其中的 CreateJsonSchema 和 CreateFunctionJsonSchema 方法。

核心機制

主要的 JSON Schema 生成功能位於 AIJsonUtilities 類別中：
`AIJsonUtilities.Schema.Create.cs:171-190`

這個方法可以為任何 .NET 型別生成對應的 JSON Schema，支援描述、預設值等參數。

函數 Schema 生成

對於方法參數的 Schema 生成，使用 `CreateFunctionJsonSchema` 方法：
`AIJsonUtilities.Schema.Create.cs:75-90`

這個方法會分析方法的參數，為每個參數生成 Schema。

配置選項

Schema 生成行為可以透過 `AIJsonSchemaCreateOptions` 進行配置：

Ask a follow-up question

Deep Research ☐

→

dotnet/extensions

src/Libraries/Microsoft.Extensions.AI.Abstractions/Utilities/AIJsonUtilities.Schema.Crea...

```
72  /// <param name="inferenceOptions">The options controlling schema creation.</param>
73  /// <returns>A JSON schema document encoded as a <see cref="JsonElement"/>.</returns>
74  /// <exception cref="ArgumentNullException"><paramref name="method"/> is <see langword="null"/>.</exception>
75  public static JsonElement CreateFunctionJsonSchema(
76      MethodBase method,
77      string? title = null,
78      string? description = null,
79      JsonSerializerOptions? serializerOptions = null,
80      AIJsonSchemaCreateOptions? inferenceOptions = null)
81  {
82      _ = Throw.IfNull(method);
83
84      serializerOptions ??= DefaultOptions;
85      inferenceOptions ??= AIJsonSchemaCreateOptions.Default;
86      title ??= method.Name;
87      description ??= method.GetCustomAttribute<DescriptionAttribute>().Description;
88
89      JsonObject parameterSchemas = new();
90      JsonArray? requiredProperties = null;
91      foreach (ParameterInfo parameter in method.GetParameters())
92      {
93          if (string.IsNullOrEmpty(parameter.Name))
94
168  /// <param name="serializerOptions">The options used to extract the schema from the specified type.</param>
169  /// <param name="inferenceOptions">The options controlling schema creation.</param>
170  /// <returns>A <see cref="JsonElement"/> representing the schema.</returns>
171  public static JsonElement CreateJsonSchema(
172      Type? type,
173      string? description = null,
174      bool hasDefaultValue = false,
```

https://deepwiki.com/search/-microsoftextensionsai-json-sc_6a9d8c98-bab9-42ba-8e68-c26fbd1309c9

DeepWiki

Index your code with [Devin](#) [Share](#)

← return to dotnet/extensions across dotnet

我想要了解 Microsoft.Extensions.AI.Abstractions 命名空間
下，能將型別轉換成 JSON Schema 的機制。

Fast Searched

Answer

您詢問的是 Microsoft.Extensions.AI.Abstractions 命名空間下的 CreateJsonSchema 和 CreateFunctionSchema 的機制。這個功能主要由 AIJsonUtilities.Schema.Create 方法實現。

核心機制

主要的方法 AIJsonUtilities.Schema.Create 負責生成 JSON Schema。這個方法會分析方法的參數，為每個參數生成 Schema。對於方法參數的 Schema 生成，使用 AIJsonSchema.Create 方法。

AIJsonUtilities.Schema.Create.cs:75-90

這個方法會分析方法的參數，為每個參數生成 Schema。

配置選項

Schema 生成行為可以透過 AIJsonSchemaCreateOptions 進行配置：

```
72 /// <param name="inferenceOptions">The options controlling schema creation.</param>
73 /// <returns>A JSON schema document encoded as a <see cref="JsonElement"/>.</returns>
74 /// <exception cref="ArgumentNullException"><paramref name="method"/> is <see langword="null"/>.</exception>
75 public static JsonElement CreateFunctionJsonSchema(
76     MethodBase method,
77     string? title = null,
78     string? description = null,
79     JsonSerializerOptions? serializerOptions = null,
80     AIJsonSchemaCreateOptions? inferenceOptions = null)
81 {
82     _ = Throw.IfNull(method);
83
84     serializerOptions ??= DefaultOptions;
85     inferenceOptions ??= AIJsonSchemaCreateOptions.Default;
86     title ??= method.Name;
87     description ??= method.GetCustomAttribute<DescriptionAttribute>().Description;
88
89     JsonObject parameterSchemas = new();
90     JsonArray? requiredProperties = null;
91     foreach (ParameterInfo parameter in method.GetParameters())
92     {
93         if (string.IsNullOrEmpty(parameter.Name))
94         {
95             continue;
96         }
97         JsonElement schema = AIJsonSchema.Create(parameter.ParameterType, serializerOptions, inferenceOptions);
98         parameterSchemas.Add(parameter.Name, schema);
99         if (schema.IsObject)
100         {
101             requiredProperties ??= new JsonArray();
102             requiredProperties.Add(parameter.Name);
103         }
104     }
105
106     JsonElement jsonSchema = new JsonObject
107     {
108         { "title", title },
109         { "description", description },
110         { "parameters", parameterSchemas },
111         { "required", requiredProperties }
112     };
113     return jsonSchema;
114 }
```

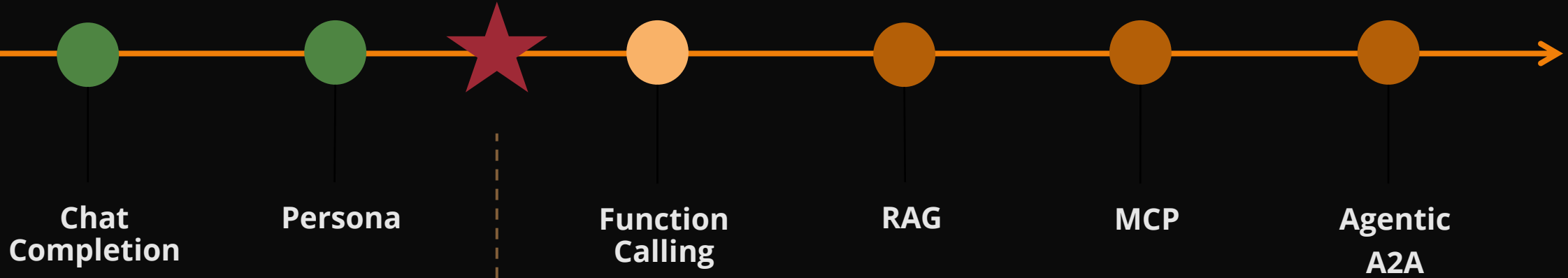
168 /// <param name="serializerOptions">The options used to extract the schema from the specified type.</param>
169 /// <param name="inferenceOptions">The options controlling schema creation.</param>
170 /// <returns>A <see cref="JsonElement"/> representing the schema.</returns>
171 public static JsonElement CreateJsonSchema(
172 Type? type,
173 string? description = null,
174 bool hasDefaultValue = false,

Ask a follow-up question

Deep Research ☐

→

JSON Mode



LLM 驚訝了全世界
知識工作者開始緊張
擔心自己被 AI 取代

可程式化驚艷了開發界
程式開發者開始緊張
開始割自己的命...

動手玩玩看

再從非結構到結構

<https://github.com/poychang/HWDC-AI.JsonMode>



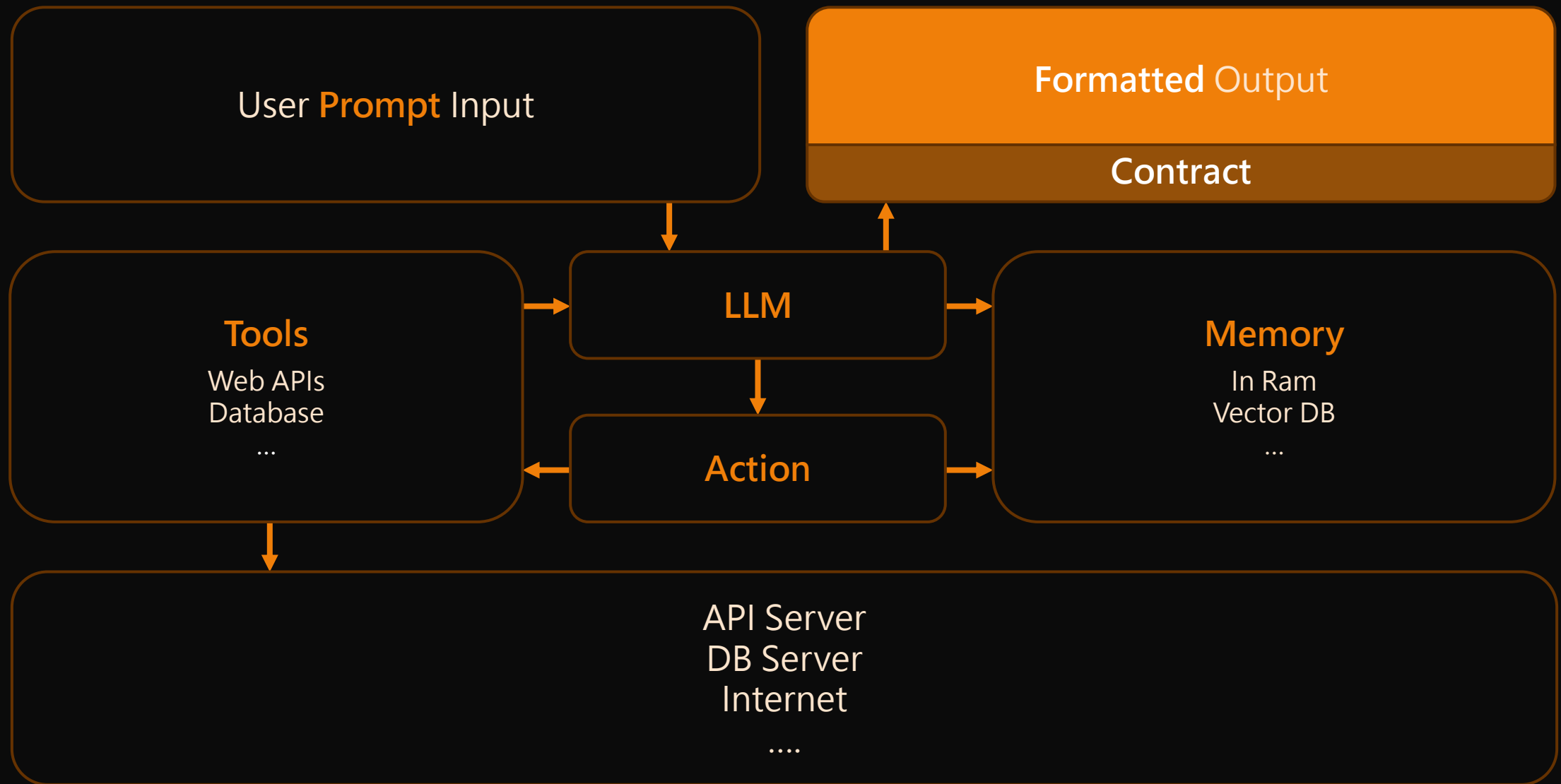
結構化輸出

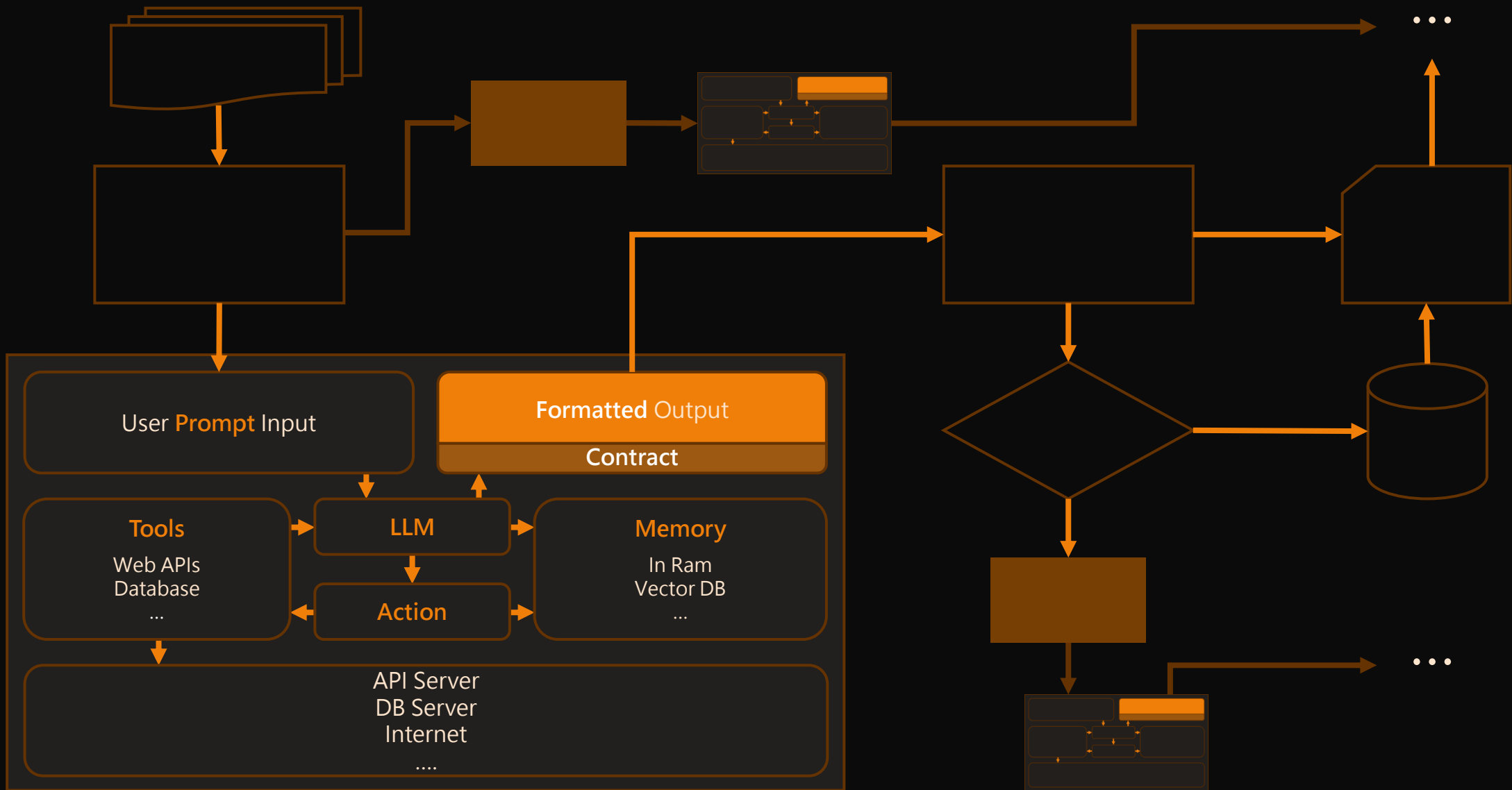
Structured Output

讓模型嚴格遵循你在推斷 API 呼叫中提供的 JSON 結構描述定義

- 支援的資料型別 String, Number, Boolean, Integer, Object, Array, Enum, anyOf
- 所有欄位都必須是必要項目
- 所有物件不能有未定義的屬性 additionalProperties: false
- 屬性數量最多 100 個
- 巢狀深度最多 5 層
- 特定型別有特定不支援的關鍵字

JSON Mode 保證產生有效的 JSON，但無法確保嚴格遵循預期的結構，強烈建議使用結構化輸出來處理需求





Enterprise App Flow

解決方案路徑圖

如何導入 AI 系統，來幫助組織提升人才技能、提高生產力，並改善產品與服務的品質？

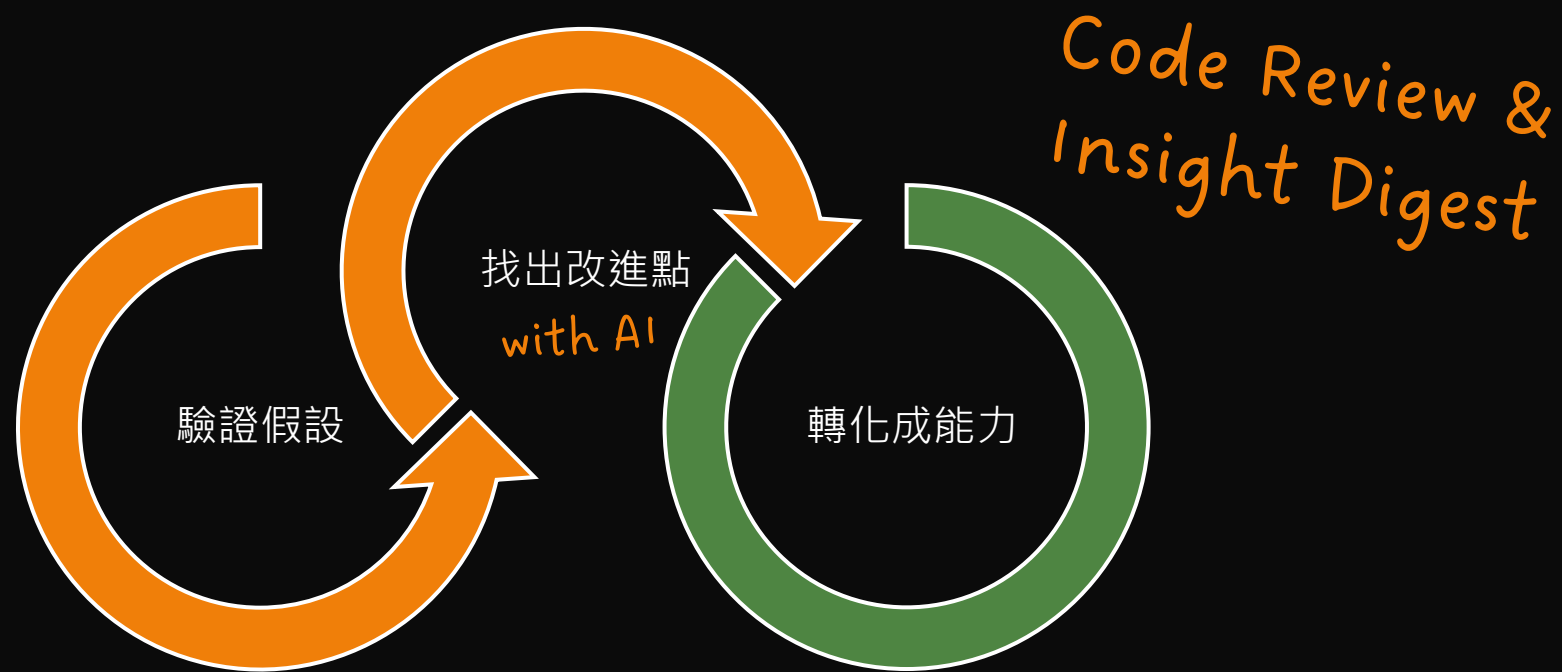


復盤

圍棋術語，本意為對弈者下完棋後，重新將對弈過程「擺」一遍，檢討、分析對弈過程，看看哪裡可以有不同甚至更好的下法

不是檢討，也不是總結

是從親身經歷中學習、反思與改進的過程



卡特爾的智力理論

Cattell's Theory of Intelligence

*Make good use of AI
if you're over 30*

流體智力

Fluid
Intelligence

- ✓ 快速思考和靈活推理以解決新問題的當前能力
- ✓ 對學習新事物的開放態度
- ✓ 隨著年齡的增長而減少

結晶智力

Crystallized
Intelligence

- ✓ 通過先前學習獲得的技能和知識的能力
- ✓ 涉及回憶具體事實
- ✓ 隨著年齡的增長而增加

結語

開發者的**匠心** == 精進的過程，是一條必須持續的路

在 AI 大行其道的時代，匠心不僅不會被取代，反而更顯珍貴

AI 時代，正是匠者**變強的最佳時代**

唯有持續精進、批判性思維、複雜溝通的能力
才能在新時代中發揮最大影響力



Vibe Coding 或 SDD (Spec Driven Development) 或許會成為未來系統開發的一種範式

- 如果成真，那麼各位將是未來最有價值的人
- 再也不會有人像各位這樣，這麼懂得寫程式了
- 如同現在懂得寫 Cobol 的人那樣，稀缺又富含價值



thanks for your
attention!