.NET Talks 直播分享 2026.04.22

Microsoft Agent Framework .NET v1.2.0：从 Agent Demo 到工程化骨架

这次 1.0 的意义，**不是功能变多**，而是**统一抽象、稳定边界、具备生产级工程能力**。

将过去分散在 Semantic Kernel、AutoGen 的能力，收敛为一套统一的开发底座。

Agents Workflows Skills

S

讲者：[圣杰 | Microsoft MVP | .NET + AI 系列课程讲师]
面向 .NET 开发者与技术管理者

Agents

处理开放式/对话式任务，支持 LLM + 工具/MCP，具备中间件拦截与状态管理

LLM 工具 MCP

Workflows

确定性流程编排，支持顺序、并发、handoff，具备 checkpoint 与可恢复机制

图模型 并发 可恢复

Skills

能力封装与复用，支持渐进式披露，可按需加载、可审计

模块化 可复用 可审计

圣杰

微软最有价值专家

讲师介绍

- 10年+ 一线 .NET 研发经验
- 资深 架构师
- Microsoft Ignite 特邀讲师
- .NET Conf China 特邀讲师
- 「向 AI 而行」公众号主理人

群聊：.NET+AI 社区群 2

该二维码7天内(4月29日前)有效，重新进入将更新

Hermes Agent 养成指南

向 AI 而行 ›

阅读1435

本课程是带你把 Hermes 调教为懂你的私人助手。内容涵盖安装与配置、模型/工具集选择与安全边界、Skills 构建与迭代、终端与多平台使用、Cron 定时自动化、以及学习/工作/内容创作等高频场景的工作流落地。

你得到的不是功能说明书，而是我自己在用的那套：从 0 到 1 搭建、从 1 到 N 复用、从单点到系统的实践路径。加入学习，一起在 AI 时代向阳生长。

微信：fatiaobot

等63人付费 ›

已发表9个 计划发表30个 每周更新　正序

9. 让 Hermes 7×24 待命：飞书对接详细指南
昨天 可试读84%

8. 从微信收藏到知识库自动收录并自检：一套 llm-wiki 自动化工作流搭建实录 | Hermes+Obsidian+LLM-Wiki
3天前 可试读99%

7. 基于 Karpathy 大神 llm-wiki 打造会进化的知识库 | Hermes+Obsidian+LLM-Wiki | 快来解救困在“数字冷宫”的颜如玉
4天前 可试读71%

面向 AI 编程
.NET 10 + AI | 智能体开发进阶

.NET+AI 智能体开发进阶

.NET 开发者专属的 AI Agent 实战课程

课程大纲

0 开篇：.NET+AI 基础
环境搭建与快速入门

1 M.E.AI
.NET 平台的 AI 底座

2 M.E.VD
RAG 检索增强生成基础

3 MCP 协议
大模型的外挂商店

4 智能体 Agent
开发实战与企业落地

5 A2A 协议
智能体协作通信

6 工作流 Workflow
多智能体编排

7 AGUI 协议
用户智能体交互协议

8 Agent Skills
可复用的智能体技能封装

9 C# 实现简版 Claude Code
终端 AI 编程助手实战

10 案例解读
真实企业 AI Agent 拆解分析

.NET 10　MAF　OpenAI/DeepSeek/Qwen/etc.　Polyglot Notebook

永久有效

.NET+AI | 智能体开发进阶 | .NET 开发者专属的 AI Agent 开发进阶课程

.NET AI 生态发展时间线

从 Semantic Kernel 到 Microsoft Agent Framework 的演进之路

2023年12月

发布 Semantic Kernel (SK) v1.0 正式版

提供了首个**生产就绪的企业级AI应用编排SDK**，标志着 .NET AI 生态进入稳定阶段。

随着MAF的发布，SK进入维护阶段，官方建议开发者逐步迁移至MAF。

2024年10月

推出 Microsoft.Extensions.AI (MEAI)

奠定 .NET AI 生态的基石，通过统一抽象层（如 IChatClient），极大简化了AI服务的集成与切换。

2025年10月

推出 Microsoft Agent Framework (MAF)

整合了 AutoGen 的研究优势与 Semantic Kernel 的生产级能力，旨在打造统一的智能体开发框架。

2025年11月

发布 .NET 10

一个**深度优化的"AI Ready"平台**，从性能、硬件支持到工具链，为AI应用提供全方位支撑。

2026年4月

发布 Microsoft Agent Framework (MAF) 1.0 正式版

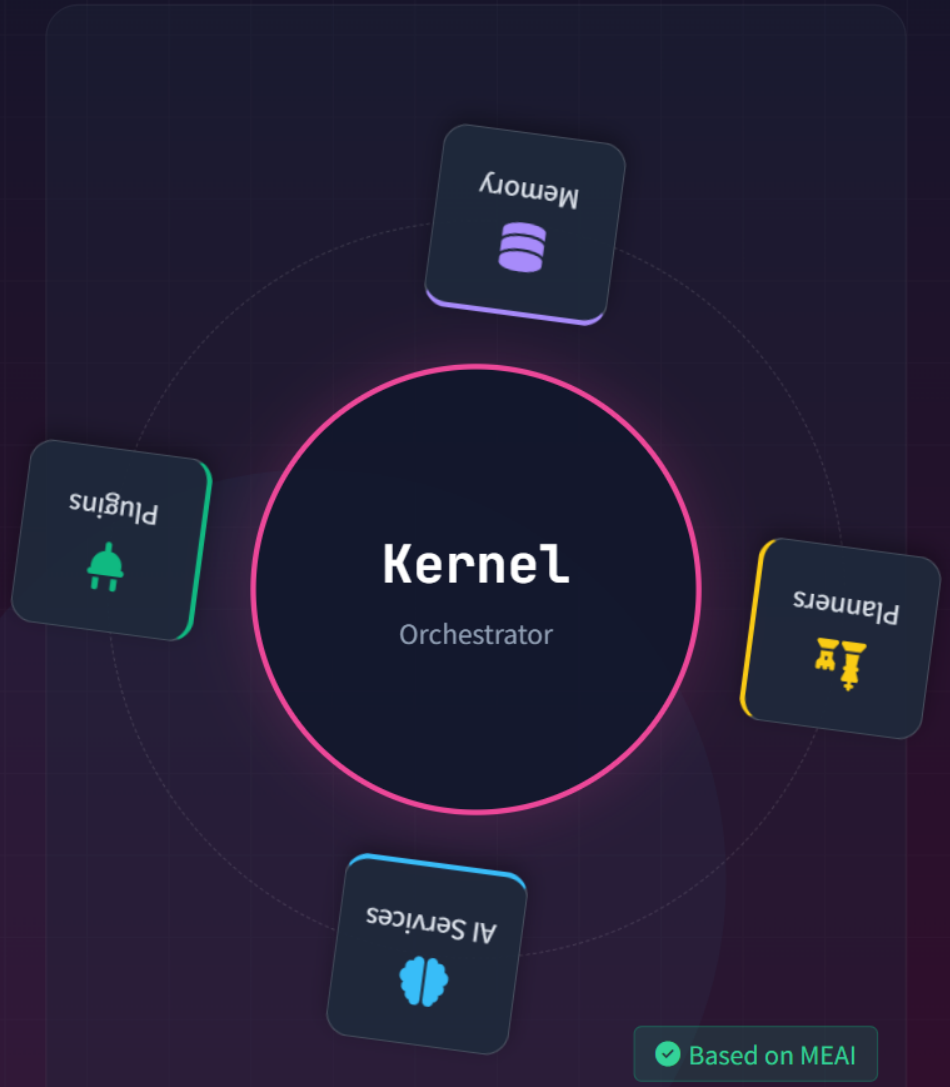
首个**生产就绪的企业级智能体开发框架**正式亮相，统一了多智能体编排、工具调用与 A2A 通信能力，标志着 .NET AI 生态全面迈入智能体时代。

跟上 .NET + AI 生态发展，抓住智能体开发新机遇

Semantic Kernel

SK 1.68+

AI 编排框架：融合大模型与现有代码，构建智能 Agent 的中枢



Agent 框架

内置 **ChatCompletionAgent**，支持创建能够自主规划、调用工具和执行任务的智能代理。

自动规划 (Planning)

能够将用户复杂的自然语言指令拆解为多个步骤，并自动选择合适的插件函数执行。

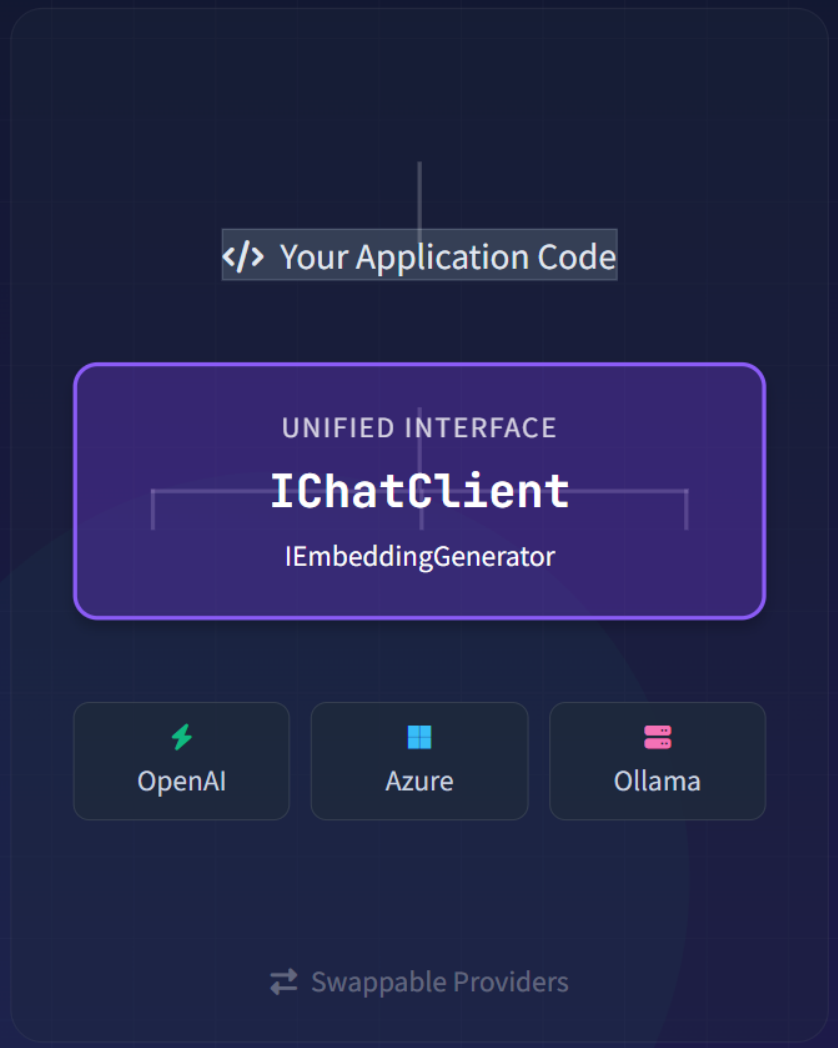
AgentExample.cs

```
// 1. 创建 Kernel 并注册插件
var kernel = Kernel.CreateBuilder()
    .AddOpenAIChatCompletion(...)
    .AddFromTypeCodeReviewPlugin() // 注册本地代码技能
    .Build();

// 2. 创建智能 Agent
var agent = new ChatCompletionAgent
{
    Name = "CodeReviewer",
    Instructions = "你是一个严谨的代码审查专家，请使用 ReviewPlugin 检查代码。",
    Kernel = kernel
};

// 3. 执行 Agent 任务 (自动规划与调用)
var response = await agent.InvokeAsync("请审查项目中的 UserAuth.cs 文件并修复漏洞");
```

Microsoft.Extensions.AI

MEAI

统一抽象层：像 ILogger 一样解耦 AI 供应商，构建可移植应用



标准接口

使用 **IChatClient** 编写业务逻辑，代码不再依赖特定的 SDK (如 OpenAI SDK)，实现真正的供应商解耦。

中间件管道

支持链式组装功能，如日志记录、自动重试、函数调用等，逻辑清晰且易于维护。

Program.cs

```
// 1. 创建基础客户端 (如 Azure OpenAI)
IChatClient baseClient = new AzureOpenAIClient(...)
    .AsChatClient("gpt-4o");

// 2. 使用 Builder 组装中间件管道
IChatClient client = new ChatClientBuilder(baseClient)
    .UseLogging()           // 自动记录对话日志
    .UseFunctionInvocation() // 自动处理函数调用
    .UseRetry(options)      // 失败自动重试
    .Build();

// 3. 业务调用 (完全无感知底层是 Azure 还是 Ollama)
var response = await client.GetResponseAsync("Hello AI!");
```

MEAI 是构建上层框架 (如 Semantic Kernel, MAF) 的基石。

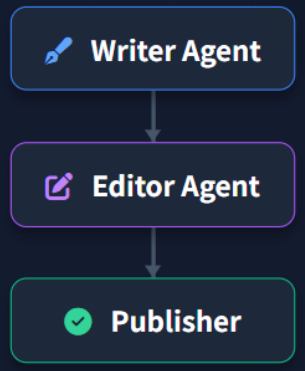
Microsoft Agent Framework MAF

面向生产的 Agent 框架：统一标准，强大编排，极简开发

工作流模式 (Patterns)

- Sequential (顺序)
- Concurrent (并发)
- Handoff (交接)
- GroupChat (群聊)

EXAMPLE: SEQUENTIAL



- DI 集成
- Minimal API
- OpenTelemetry
- DevUI

Workflow.cs

```
// 1. 定义两个 Agent
AIAgent writer = new ChatClientAgent(chatClient, new() {
    Name = "Writer", Instructions = "Write engaging stories."
});

AIAgent editor = new ChatClientAgent(chatClient, new() {
    Name = "Editor", Instructions = "Fix grammar and enhance plot."
});

// 2. 编排为顺序工作流 (Sequential)
// 结果会自动从 Writer 传递给 Editor
Workflow workflow = AgentWorkflowBuilder.BuildSequential(writer, editor);

// 3. 执行工作流
AIAgent workflowAgent = await workflow.AsAgentAsync();
var response = await workflowAgent.RunAsync("Write a story about AI.");
```

MAF 建立在 **MEAI** (抽象层) + **Semantic Kernel** (编排) + **AutoGen** (多代理) 之上，提供统一开发体验。

2025年11月正式发布

.NET 10 重磅发布

性能狂飙 · 云原生进化 · AI 生态融合，开启即时运行新时代

性能持续优化

JIT 编译器深度改进，更低的内存占用，更高的运行时吞吐量，延续性能之王的神话。

容器支持增强

专为云原生设计，镜像体积进一步缩减，启动速度大幅提升，与 Kubernetes 集成更丝滑。

原生 AOT 进化

支持更多反射场景和库，编译速度提升。让应用像 Go/Rust 一样轻量高效，无依赖独立运行。

New

直接运行 .cs 文件

`dotnet run app.cs` 像脚本语言一样直接执行单文件，无需项目文件，极简开发体验。

New

dnx 工具链登场

.NET 版的 npx/uvx。无需安装即可运行 NuGet 工具，支持隔离执行环境，工具使用更安全便捷。

AI Ready

NuGet 分发 MCP

将 AI 工具（MCP Server）打包为 NuGet 包分发，用户一键安装即可获得 Agent 能力，生态大融合。

Microsoft Agent Framework

Agent 开发正式进入工程化时代

✔2026年4月 正式发布 V1.0

整合 **Semantic Kernel** 与 **AutoGen** 研究优势，收敛为面向生产的统一 Agent 开发底座。从"可以演示"走向"可以落地、可持续演进"。

WHY IT MATTERS 01

边界清晰 · 抽象稳定

三层统一抽象：Agents 处理开放任务，Workflows 控制确定流程，Skills 封装领域能力

WHY IT MATTERS 02

工程级 Runtime 基础设施

状态管理、中间件、可观测、Checkpointing、Human-in-the-loop，全面就绪

WHY IT MATTERS 03

.NET 的 AI 时代优势

类型系统、企业集成在 Agent 时代凸显；.NET 开发者迎来最佳入场时机

技术栈全景 · MAF TECHNOLOGY STACK

🔷 **M.E.AI** 统一 AI 抽象层	🗄️ **M.E.VectorData** RAG / 向量检索	🔌 **MCP** 模型上下文协议	🤝 **A2A Protocol** 多智能体通信
🖥️ **AG-UI** Agent UI 交互协议	🤖 **Agents** 语义任务执行	🔀 **Workflows** Graph 流程编排	🧩 **Agent Skills** 可复用能力封装
📊 **OpenTelemetry** 可观测 / 追踪	⚡ **.NET 10** AI-Ready 运行时		

"1.0 不是终点，而是主航道的开始。SK + AutoGen 在此汇合，.NET 开发者迎来构建 Agent 系统的最佳时机。"

向 AI 而行

.NET + AI Agent

协议 SDK 全面支持

.NET 生态全面拥抱 AI 互操作标准

Production Ready

MCP SDK

Model Context Protocol

```
dotnet new mcpserver
```

一键创建 MCP Server，通过 Attribute 轻松暴露工具，支持 **NuGet 分发**。

C#
```
public class MyTools {
```

C#
```
[McpServerTool]
[Description("Get random number")]
public int GetRandom(int min, int max)  {
return Random.Shared.Next(min, max); }
}
```

A2A SDK

Agent-to-Agent

```
dotnet add package A2A
```

定义 Agent 间通信标准，包含 **Agent Card** 发现机制与任务投递。

C#
```
// Server-side registration
var manager = new TaskManager();
app.MapA2A(manager, "/echo");
```

C#
```
// Client-side invocation
var client = new A2AClient(agentUrl);
await client.SendMessageAsync(new Message
{ Role = MessageRole.User, Content =
"Collaborate!" });
```

AG-UI SDK

Agent UI Protocol

```
Microsoft.Agents.AI.AGUI
```

流式 UI 交互协议，支持实时进度反馈与 **Human-in-the-loop** 确认。

C#
```
// Server: Map Agent to Endpoint
app.MapAGUI("/", agent);
```

C#
```
// Client: Stream UI Updates await
foreach (var update in
client.RunStreamingAsync(messages))
{   // Render UI updates in real-time
Console.Write(update.Text); }
```

.NET + AI 深度融合

从底层抽象到生产级 Agent 的完整工具链

APPLICATION LAYER

Microsoft Agent Framework

面向生产的 Agent 框架。支持复杂工作流编排、状态管理、多 Agent 协作 (Handoff/GroupChat)。

ORCHESTRATION LAYER

Semantic Kernel (SK)

核心编排引擎。将 LLM 的推理能力与现有代码 (Plugins) 连接，处理 Planning 和 Memory。

ABSTRACTION LAYER

Microsoft.Extensions.AI

统一抽象层 (IChatClient)。解耦具体模型供应商 (OpenAI/Ollama)，提供标准化的中间件支持。

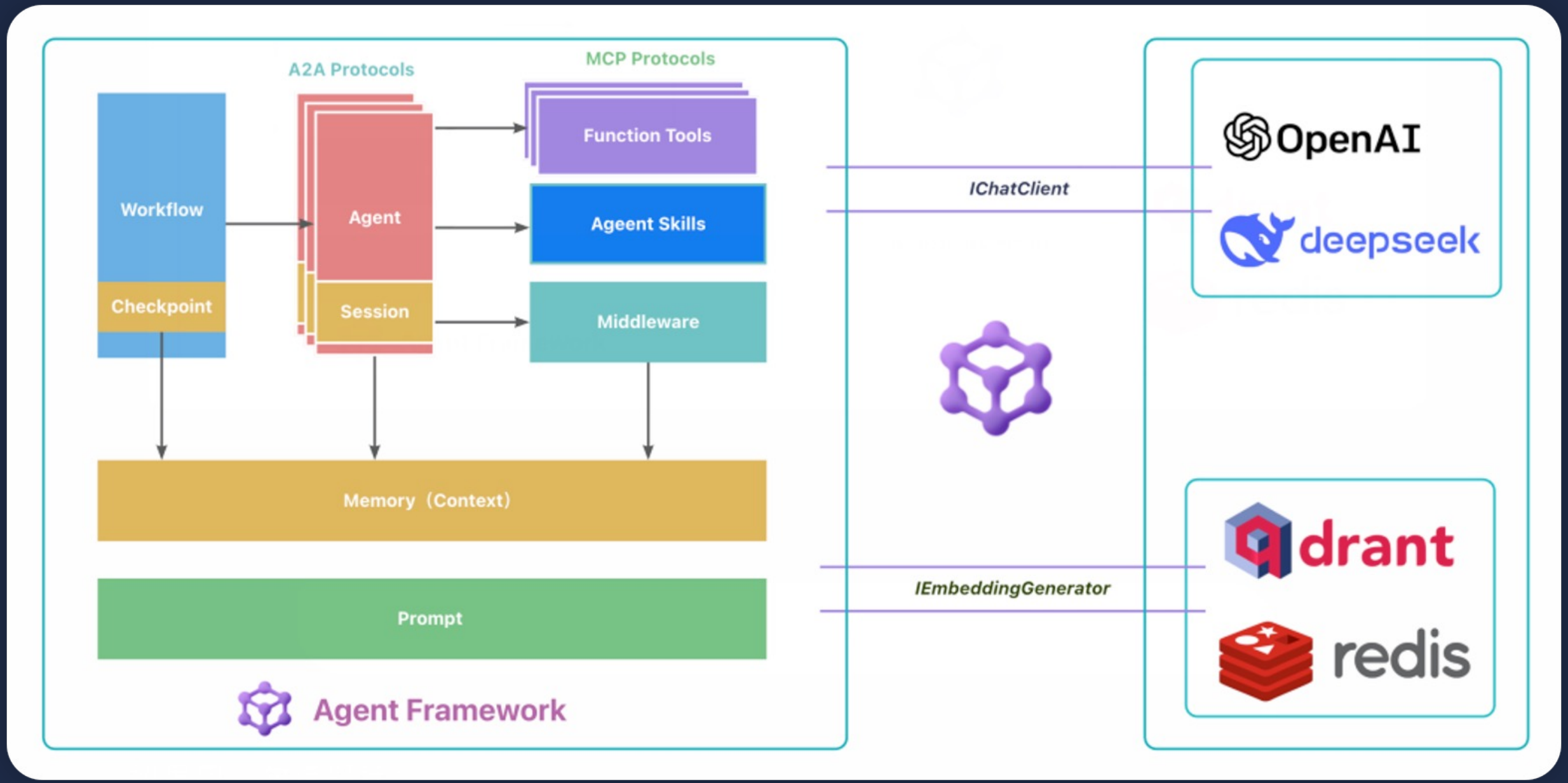
全面支持协议互操作

贯穿整个技术栈的标准协议，确保生态兼容性。

MCP　A2A　AG-UI　Skills

为何重要?

微软不再只是提供单一的 SDK，而是构建了**分层解耦**的完整生态。开发者既可以使用 MEAI 进行底层模型调用，也可以用 SK 编写插件，最终通过 MAF 构建复杂的企业级智能体应用。


A2A Protocols
MCP Protocols
Workflow
Checkpoint
Agent
Session
Function Tools
Ageent Skills
Middleware
Memory（Context）
Prompt
Agent Framework
IChatClient
IEmbeddingGenerator
OpenAI
deepseek
drant
redis

为什么这次 1.0 值得关注

Production-ready 发布，标志着 Agent 开发进入工程化时代

这次 1.0 的价值，**不在功能变多**，而在**边界更清晰、抽象更稳定、工程信号更强**。
从"可以演示"走向"可以落地、可以扩展、可以长期维护"。

Production-ready

稳定 API + 长期支持承诺，.NET/Python 同步 GA，企业级多智能体编排

Stable APIs　LTS

统一开发底座

整合 Semantic Kernel + AutoGen 能力，单一框架替代分散体系

SK　AutoGen

Agents + Workflows

明确骨架：开放式 Agent + 确定性 Workflow，支持并发/分支

图模型　并发

企业级能力

Session 状态管理、中间件拦截、Telemetry、Type Safety

可观测　类型安全

互操作性

A2A/MCP 协议支持，多模型提供商（Azure OpenAI/OpenAI/Anthropic）

A2A　MCP

职业信号

竞争力从 Prompt 技巧转向状态治理、流程控制、工具约束

工程化　系统化

Sources: Microsoft Agent Framework 1.0 公告, Microsoft Learn Overview, Workflows docs

Agent Framework 1.0　Microsoft Learn　DevBlogs

微软到底在统一什么

Agents + Workflows + Skills：三层抽象架构解析

微软不是在堆功能，而是在**统一 Agent 抽象、流程抽象与能力封装抽象**，让复杂系统具备清晰的边界、稳定的演进路径和工程化的治理手段。

Agents

开放式任务处理

LLM + 工具/MCP，处理开放式/对话式任务，支持多轮会话与状态管理

LLM | Tools | MCP | Sessions

智能决策能力

具备自主工具调用、规划推理、中间件拦截与上下文管理

规划 | 推理 | 拦截

状态持久化

支持会话状态管理、上下文记忆、多轮对话恢复机制

State | Memory | Recovery

Workflows

图模型编排

Graph-based orchestration，支持顺序、并发、分支、循环等复杂流程

图模型 | 并发 | 分支

可控执行

Checkpointing、Human-in-the-loop、可恢复机制、类型安全路由

Checkpoint | HITL | Recovery

可观测性

Events、Telemetry、日志追踪，支持调试与性能监控

Events | Telemetry

Skills

能力封装

SKILL.md + scripts + references + assets，模块化能力复用

SKILL.md | Scripts | Assets

渐进式披露

Advertise → Load → Read → Run，按需加载，上下文管理

渐进式 | 按需加载

可审计治理

可复用、可审计、可按需加载，支持权限控制与版本管理

可审计 | 版本管理

Session State | Context Providers | Middleware | Telemetry | A2A | MCP | Model Providers

边界清晰 → 抽象稳定 → 增量演进

建立可持续演进的 Agent 系统架构

Agents vs Workflows：什么时候用谁

Agent 负责弹性，Workflow 负责边界；两者不是替代关系，而是分工关系。

Agents (开放式/对话式任务)

适用场景

步骤未知、需要语义理解与规划、单轮/多轮会话。

控制方式

LLM 动态决策，依靠 System Prompt 与 Tools/MCP 驱动。

典型任务

探索式产出（总结、生成、改写）、信息提取与整理。

风险边界

执行路径不确定，难以实现严格的审计与流程回放。

VS

Workflows (确定性流程)

适用场景

需要显式步骤与顺序控制、跨系统/Agent 长时运行编排。

控制方式

图模型 (Graph-based)、条件路由、强类型约束。

典型任务

多步审批审批、复杂数据管道、需要 Checkpoint 的长流程。

风险边界

过度设计可能导致灵活性丧失，需与 Agent 配合使用。

架构选型规则

1. 若能写成确定性函数
   → 优先编写代码函数
2. 若步骤未知/需探索规划
   → 优先使用 Agent
3. 若关键节点受控(审计/HITL)
   → 用 Workflow 包裹 Agent/函数

Sources: Microsoft Learn - Agent Framework Overview / Workflows docs

Workflow：把不确定任务转为可控流程

Workflow 的价值，不是替代 Agent，而是把不可预测的执行过程放进可治理的骨架。

工程价值： 显式控制 | 可追踪 | 可恢复 | 可协作 | 可演进

Sources: Microsoft Learn - Workflows

Workflow 的工程能力

5 个关键内建机制

真正让 Workflow 进入生产环境的，不是能画图，而是这些工程机制是否内建。

Type-safe Routing

强类型消息与模式校验，有效降低运行时错误。

降低错误

Checkpointing

保存执行状态，支持长流程的暂停、恢复与回放。

支持恢复

Human-in-the-loop

支持在关键节点无缝插入审批、确认等人工干预环节。

人工干预

Events & Telemetry

内置丰富事件流与遥测追踪，让系统运行完全可观测。

可观测性

Recovery

内置重试与补偿策略机制，轻松处理各类异常状况。

异常处理

核心术语 (Core Concepts)

- **Executors**: 流程处理单元
- **Edges**: 连接与路由规则
- **Events**: 执行过程状态变更
- **Supersteps**: 状态执行生命周期

Sources: Microsoft Learn - Workflows / Agent Framework Overview

Agent Skills 核心解析

为什么它是 1.0 的最后一块拼图

没有 Skills，Agent 往往只是**调用工具的 Prompt 容器**；
有了 Skills，能力才真正成为**可复用资产**。

官方定义

包含指令、脚本、资源的便携包，遵循开放规范，Agent 按需调用。

便携包

解决的问题

防止 Prompt 无限膨胀，知识难以复用，以及黑盒执行导致的审计困难。

降本控险

核心价值

标准化领域知识，可复用、可审计、可按需加载，支持跨团队互操作。

能力资产化

架构关系

Skills 封装能力，Agents 负责使用与规划，Workflows 负责控制执行边界。

三层协同

典型场景

政策合规守则、业务流程手册、代码规范约束与标准分析模板。

领域知识沉淀

小结：Skills 解决的是能力沉淀与复用，不是流程控制。

Sources: Agent Skills Official Documentation / 社区专家分析文章

Progressive Disclosure：让上下文更轻、协作更高效

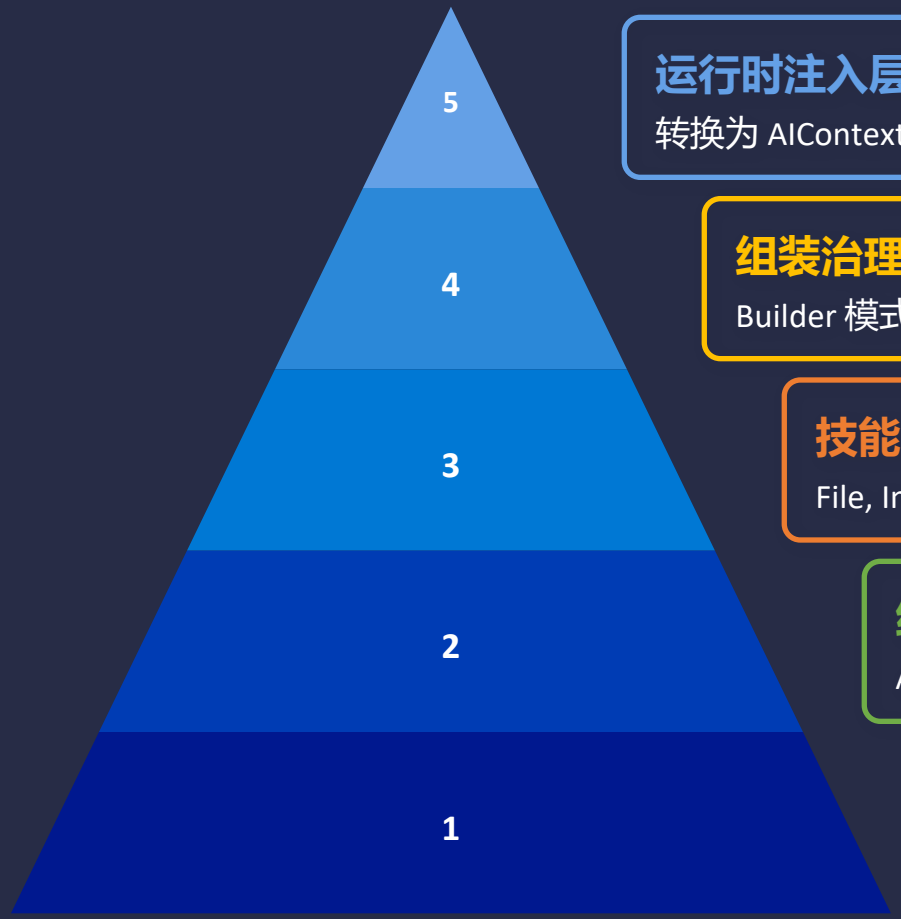
关键不是把所有知识都塞进上下文，而是只在需要时加载需要的那一部分。

实施收益： 更低 Token 成本 | 更清晰权限边界 | 更高协作效率 | 更强审计能力

Sources: Microsoft Learn - Agent Skills

MAF Agent Skills 五层架构

将静态的 Markdown 目录，升维为 Agent Runtime 中的可执行能力



运行时注入层：
转换为 AIContext 的 Instructions & Tool

组装治理层：
Builder 模式组合、过滤、去重、审批

技能来源层：
File, InMemory, Inline 等多种 Provider 挂载

统一模型层：
AgentSkill, Frontmatter, Resource, Script

协议与内容层：
对齐 agentskills.io 标准与目录规范

.NET 中 Skills 的创作方式与协作价值

三种 authoring 方式不是互斥关系，而是为了**匹配不同团队边界、发布节奏与治理要求**。

File-based

目录 + SKILL.md + scripts/refs

最佳场景

- 业务文档与流程模板沉淀
- 需要动态更新且免编译的能力
- 跨代码仓库共享的标准化资产

协作价值

非开发人员可维护，业务逻辑与代码解耦，支持热加载。

Class-based

C# 封装类 / NuGet 包分发

最佳场景

- 企业级核心系统能力（如 HR 系统）
- 跨团队强依赖的标准件
- 需要严格版本控制与依赖注入的模块

协作价值

版本化管理与分发，自动发现与挂载，利于统一审批治理。

Inline C#

应用程序内联代码定义

最佳场景

- 敏捷迭代阶段的快速原型验证
- 紧急需求的临时业务桥接
- 强依赖当前应用运行态内存上下文

协作价值

无需发布包即可补齐能力，待业务稳定后平滑抽出成包。

组合策略：同一个 Skills Provider 可以自由组合多源技能，实现团队独立发布与平台统一治理。

Sources: DevBlogs - Agent Skills in .NET: Three Ways to Author, One Provider to Run Them

给 .NET 开发者的架构启示

从 Prompt 技巧转向系统演进与治理

未来的竞争力，**不再是提示词技巧**，而是把不确定性装进**可治理的软件结构**。

辨识边界：函数优先

能写成确定性函数的任务就不要上 Agent，避免引入不必要的复杂度与成本。

明确边界　降本增效

分流探索：Agent 负责未知

将需要语义理解、步骤规划和自主决策的探索性子任务交给 Agent 处理。

探索任务　动态规划

强化约束：Workflow 兜底

关键业务路径必须用 Workflow 包裹，确保执行顺序、审计点与合规要求。

关键路径　合规约束

积累资产：Skills 沉淀知识

用 Skills 将领域知识和流程经验标准化，实现跨团队的按需加载与复用。

知识沉淀　模块化

防御设计：面向故障构建

充分利用 Events、Checkpoint、HITL，构建具备可观测性和可恢复性的容错系统。

容错恢复　可观测

规范集成：类型安全优先

依托 .NET 强类型消息系统与 A2A/MCP 协议，实现稳定且易维护的多模型/工具集成。

类型安全　标准互通

→ **核心范式转变：从 Prompt Engineering 转向 State Governance（状态治理与工程管控）。**

Sources: Learn Overview, Workflows, Skills docs & Community Insights

Architecture　Engineering　.NET Ecosystem

建议的企业级 Agent 系统分层

企业级 Agent 系统不该直接把模型贴在界面上，而应该有**清晰的分层、编排与治理平面**。

体验层
(App / UI / 渠道)

- Copilot / Chat
- Web / API
- IDE / 插件

业务编排层
(Workflows)

- 图式流程 (Workflow)
- 条件路由
- 并发执行
- HITL
- Checkpoint

能力层
(Agents + Skills)

- Agents (规划与对话)
- Skills (封装领域资产)
- MCP 协议桥接

工具与数据层
(Tools & Data)

- 业务 API
- RAG / 向量库
- 数据库
- 云函数 (Functions)

治理与平台层
(Governance & Platform)

- Identity / 权限
- Telemetry / 日志
- 模型与 Key 管理
- 成本 / Quota
- Compliance

↔ 横向互操作与集成生态：A2A | MCP | AG-UI | Azure Functions | M365 | DevOps

Sources: Microsoft Learn - Overview, Workflows, Skills & Integration Docs

团队落地路径与协作方式

从单体 Demo 到企业级平台的演进步骤

真正可持续的 Agent 系统，不是一次性做大，而是沿着**可治理的路径逐步演进**。

⚡ **总结策略：先闭环，再复用； 先治理关键路径，再平台化演进。**

Sources: 结合官方架构建议与团队落地实践总结

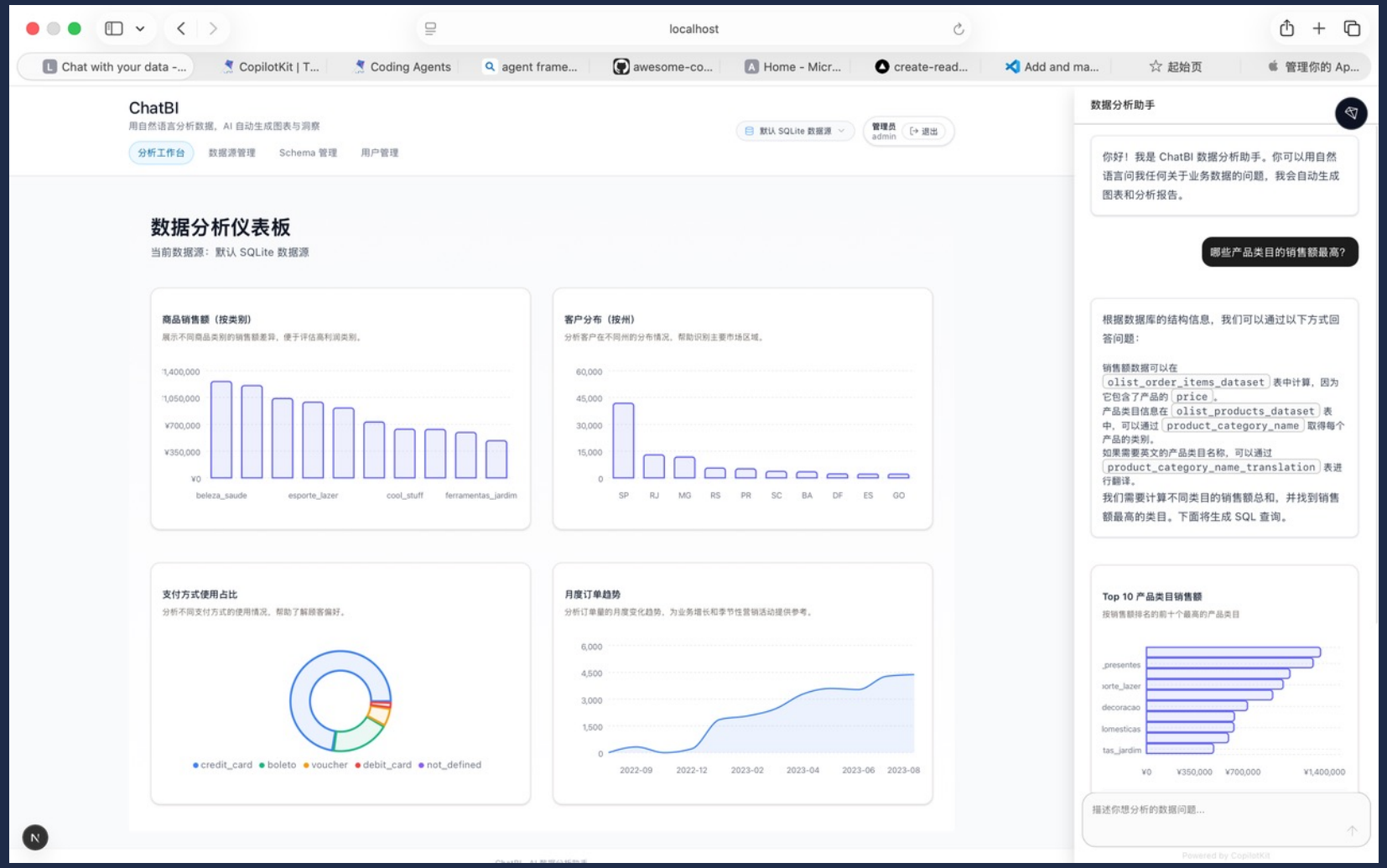
Sources / References

官方资料 (Microsoft Learn & DevBlogs)

1. **Microsoft Agent Framework 1.0 发布公告**
   devblogs.microsoft.com/agent-framework/microsoft-agent-framework-version-1-0/
2. **Agent Framework Overview**
   learn.microsoft.com/en-us/agent-framework/overview/
3. **Workflows Documentation**
   learn.microsoft.com/en-us/agent-framework/workflows/
4. **Agent Skills Specification**
   learn.microsoft.com/en-us/agent-framework/agents/skills
5. **Semantic Kernel and Microsoft Agent Framework**
   devblogs.microsoft.com/agent-framework/semantic-kernel-and-microsoft-agent-framework/
6. **Agent Skills in .NET: Three Ways to Author, One Provider to Run Them**
   devblogs.microsoft.com/agent-framework/agent-skills-in-net-three-ways-to-author...

Key References: Agent Framework v1.0 Blog | Workflows Docs | Agent Skills Docs | MAF GitHub

DEMO


localhost
Chat with your data -...
CopilotKit | T...
Coding Agents
agent frame...
awesome-co...
Home - Micr...
create-read...
Add and ma...
起始页
管理你的 Ap...
ChatBI
用自然语言分析数据，AI 自动生成图表与洞察
分析工作台
数据源管理
Schema 管理
用户管理
默认 SQLite 数据源
管理员
admin
退出
数据分析仪表板
当前数据源：默认 SQLite 数据源
商品销售额（按类别）
展示不同商品类别的销售额差异，便于评估高利润类别。
¥1,400,000
¥1,050,000
¥700,000
¥350,000
¥0
beleza_saude
esporte_lazer
cool_stuff
ferramentas_jardim
客户分布（按州）
分析客户在不同州的分布情况，帮助识别主要市场区域。
60,000
45,000
30,000
15,000
0
SP
RJ
MG
RS
PR
SC
BA
DF
ES
GO
支付方式使用占比
分析不同支付方式的使用情况，帮助了解顾客偏好。
credit_card
boleto
voucher
debit_card
not_defined
月度订单趋势
分析订单量的月度变化趋势，为业务增长和季节性营销活动提供参考。
6,000
4,500
3,000
1,500
0
2022-09
2022-12
2023-02
2023-04
2023-06
2023-08
数据分析助手
你好！我是 ChatBI 数据分析助手。你可以用自然语言问我任何关于业务数据的问题，我会自动生成图表和分析报告。
哪些产品类目的销售额最高?
根据数据库的结构信息，我们可以通过以下方式回答问题：
销售额数据可以在 olist_order_items_dataset 表中计算，因为它包含了产品的 price。
产品类目信息在 olist_products_dataset 表中，可以通过 product_category_name 取得每个产品的类别。
如果需要英文的产品类目名称，可以通过 product_category_name_translation 表进行翻译。
我们需要计算不同类目的销售额总和，并找到销售额最高的类目。下面将生成 SQL 查询。
Top 10 产品类目销售额
按销售额排名的前十个最高的产品类目
_presentes
orte_lazer
decoracao
lomesticas
tas_jardim
¥0
¥350,000
¥700,000
¥1,400,000
描述你想分析的数据问题...

Q & A

群聊：.NET+AI 社区群 2

该二维码7天内(4月29日前)有效，重新进入将更新

MAF Features going to GA

- Single Agent and service connectors
- Middleware Hooks & Connectors
- Agent Memory & Context Providers
- Multi-Agent Orchestration
- Agent Workflows
- Declarative Agents & Workflows (YAML)
- Hosting responses, A2A* & MCP
- Local, OpenAI, Anthropic, Amazon, Ollama)
- Migration Assistants (AutoGen & Semantic Kernel)

MAF Features still in Preview

- DevUI — interactive local agent debugger
- Foundry Hosted Agent Integration
- Foundry Tools, Memory & Observability Integration
- AG-UI / CopilotKit / ChatKit integration
- Skills — reusable domain capability packages
- GitHub Copilot CLI & Claude Code SDK integration
- Agent Harness (local shell/coding/messaging agent loop)