Polytool: 一种通用智能体框架

1. 设计理念与目标

Polytool 的设计核心是“技术普惠与专业深度的统一”，即通过低代码开发范式、插件化架构设计与 增强 RAG 技术三大支柱，力求构建兼顾易用性与扩展性的 AI 应用开发体系。既需要降低开发人员的使用与开发门槛与时间开销，又需要为开发者保留深度定制空间，同时通过标准化与模块化设计支撑企业级复杂场景下的需求。

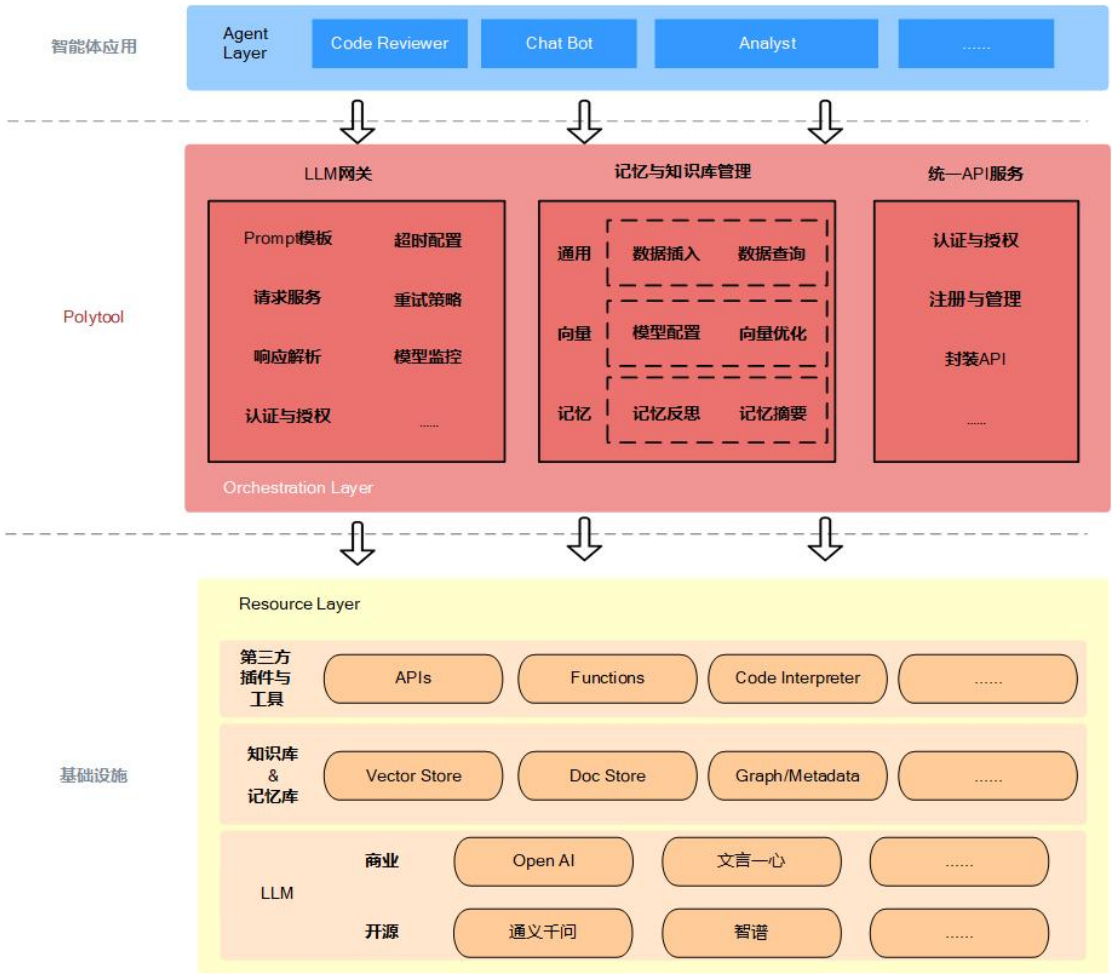
在**低代码开发**方面，平台的核心目标在于简化开发流程，使开发者无需深入底层代码，即可轻松协调多种**基础资源**实现来完成一个完整的、用例级别的**应用服务**。甚至于，可以尝试进一步的提供直观的可视化界面，通过拖拽和配置即可完成 AI 应用开发。

在**插件化架构**方面，通过统一接口规范实现模型与工具的“即插即用”。例如，在模型方面，支持 GPT、LLaMA 等主流大语言模型无缝接入；在第三方工具方面，可集成 API 调用，乃至支持 IoT 设备控制等数字与物理世界能力，扩展 AI 的行动边界。

而通过**增强 RAG** 技术，则解决了传统检索增强生成技术的落地痛点。不仅能够实现外部知识的检索，更能提供“上下文感知”与“记忆增强”等功能。允许 AI 应用开发过程中通过动态、智能地为其装配最相关、最丰富的上下文信息，突破模型在知识时效性、私有数据和专业深度上的限制。让 LLM 智能体能够像一位经验丰富的领域专家一样，不仅拥有广博的知识（LLM 本身），还能随时调用专属的“记忆库”和“知识库”，进行深度、连续且专业的思考和分析。

2. 总体架构

如图所示，Polytool 采用模块化设计，架构分为三个核心层次，自底向上分别是：资源层（Resource Layer）、管理层（Orchestration Layer）和智能体层（Agent Layer）。其中，资源层提供基础 AI 能力、数据资源以及第三方插件与工具；资源管理层提供统一资源抽象与调度，其关键子模块包括为 LLM 服务、知识与记忆数据库以及插件系统进行的 API 构建；而智能体层负责具体的应用构建与编排。在接下来的章节中，将对不同层次的具体功能进行进一步介绍，并主要对资源管理层的不同子模块进行详细介绍。



https://www.processon.com/v/68c8faab71eae511dca77fd9?cid=68c8d01a8e2d82685409321f

3. 分层详解

3.1 资源层 (Resource Layer)

此层是 Polytool 的“世界知识”和“执行能力”的最终来源，将所有外部资源抽象为三大类，即大模型、记忆与知识库以及第三方插件与工具库。下表中，对这三类资源的功能描述以及泛化后的通用形态进行了进一步的描述。

组件	功能描述	泛化后的通用形态
大模型 (Large models)	提供通用智能能力，包括语言理解、内容生成、逻辑推理和多任务处理等。	①通用任务处理器 (General Task Processor):通过提示词工程适应各种任务（如文案创作、代码生成、数据分析），无需针对每项任务重新训练模型。 ②多模态融合中枢 (Multimodal Fusion Hub):理解和生成文本、图像、音频、视频等多模态内容，实现跨模态信息的统一处理和转换。 ③智能决策引擎 (Intelligent Decision Engine):通过 Function Calling 等技术调用第三方工具和

		API，实现“模型决策+工具执行”的闭环（如查询数据库后生成报告）。 ④逻辑规划中枢(Logic Planning Hub):对复杂任务进行分解、排序和策略制定,接收抽象目标，通过逻辑推理生成可执行的步骤序列，并能在执行过程中根据反馈进行动态调整。
记忆与知识库 (Memory and Knowledge Base)	存储智能体所需的领域知识、历史记录和结构化信息。	①向量存储 (Vector Store):存储任何文本、文档以及历史对话记录的嵌入向量，用于语义检索。 ②图数据库/元数据库 (Graph/Metadata Store):存储实体、关系、属性（如代码函数调用关系、产品目录结构、人物关系网）。 ③文档存储 (Doc Store):存储原始文档、图片、音视频等非结构化数据。
第三方插件与工具库 (Third-party Plugins and Tool Base)	为智能体提供执行具体操作、获取实时信息的能力。	①第三方插件:插件是一个工具集，一个插件内可以包含一个或多个工具（如 Jira, Salesforce, GitHub, 天气 API），使用这些插件，可以帮助大模型拓展智能体能力边界。 ②函数工具 (Functions):封装可执行代码（如计算器、数据格式化工具、代码执行器等）。

3.2 管理层 (Orchestration Layer)

3.2.1 LLM 网关

LLM 网关是一种典型的中间件，负责为外部客户端提供一致、稳定、安全的入口，同时屏蔽后端众多 AI 模型、插件和服务的复杂性，充当 “服务中介”。

它接收来自用户应用程序\智能体的请求，对请求进行处理或批量操作后，发送至用户选择的 AI 服务提供商。当大模型完成处理后，网关会收集响应数据，必要时进行进一步处理，最终将结果返回给用户。

其核心功能包括管理 Prompt，将 Prompt 提示词和 AI 模型组合成自定义的 AI API; 提供统一 API，简化不同 LLM 服务提供商的 API 调用差异; 进行智能解析，过滤大语言模型的自然语言请求和响应中的关键信息; 增强安全性和合规性，通过集中认证、访问控制和数据处理等措施，帮助维护法规合规性; 以及提供多维度的 LLM 流量监控，使 LLM 的成本可视化等。

对此，在下表中，针对该 **LLM 网关**功能描述以及泛化后的通用形态进行了进一步的说明。

组件	功能描述
Prompt 管理模块	允许对每个 LLM API 的 Prompt 进行独立管理，将 Prompt 提示词和 AI 模型组合成自定义的 AI API，甚至可以在上述基础上进一步发布到 API 门户供订阅，还能设置审核流程。

统一 API 层\请求服务	作为应用与 LLM 交互的唯一入口，封装不同 LLM（如 GPT、文心一言、Llama 系列等）的 API 协议差异，提供标准化的请求 / 响应格式（如统一的参数命名、返回结构）让应用无需修改底层代码即可切换不同 LLM 服务。
响应解析模块	基于规则引擎与轻量 NLP 能力，自动解析请求中的结构化信息、关键实体（如调用模型类型、参数、功能标识）及响应中的核心结果，支持自定义解析规则，为后续流量分发与结果处理提供基础。
认证与授权模块	通过集中认证、访问控制、数据脱敏等措施，增强安全性和合规性，保护用户数据，确保 AI 系统在合规和安全标准内运行。
超时配置与重试策略模块	支持按 LLM 类型、请求类型（如流式 / 非流式）自定义超时阈值（如短文本请求 3 秒、长文档处理 30 秒）；内置阶梯式重试策略，可配置重试次数、间隔时长及触发条件（如网络超时、非业务错误码），并支持失败后的降级处理（如切换备用 LLM）。
监控与日志模块	实时采集全链路指标，包括请求量、成功率、响应耗时、Token 消耗、错误码分布等，通过可视化仪表盘展示；记录每一次请求的详细日志（含调用方、请求参数、Prompt 内容、响应结果、耗时等），支持日志检索、导出与审计，满足故障排查与合规追溯需求。

3.2.2 记忆与知识库管理

在记忆与知识库管理相关的系统或模块中，往往需要封装包括连接管理和基础 CRUD 操作在内基本 API 功能。此外，不同类型数据与数据库各有其不同的重要作用以及独特的功能需要被单独封装。以及以向量数据库为例，往往需要使用先行使用专门的嵌入模型将自然语言映射到一个高维向量空间（例如 384、768 或 1024 维），才能够进一步的进行存储并管理（插入、检索、删除等）。而与之相对的，针对智能体而言，记忆与知识又需要不同的管理方式，相对于知识库而言，智能体在管理并优化存储记忆的过程中，又进一步提出了一系列新的需求，这包括反思（提取更高层次的见解）、摘要（压缩文本同时突出关键点）、遗忘（选择性丢弃信息）、截断（在 token 限制内格式化）和判断（评估信息的存储优先级）等。

3.2.2.1 通用子模块

如前文所示，作为一个设计良好的数据库管理\封装层，需要封装包括**连接管理**和基**础 CRUD 操作**在内基本 API 功能。下表中，该类 API 的功能描述以及泛化后的通用形态进行了进一步的说明。

组件	功能描述	泛化后的通用形态

数据库连接管理 (Database Connection Management)	负责智能体与各类数据库的底层连接建立、维护、认证授权及高效交互调度，保障数据读写操作的稳定性、安全性与性能，同时屏蔽不同数据库的协议差异，为上层应用提供统一的数据访问接口。	①统一连接接口层 (Unified Connection Interface Layer): 封装不同数据库的原生驱动与通信协议，提供标准化的**连接创建、断开**。 ②认证与权限代理 (Authentication & Permission Proxy): 集中管理数据库访问凭证（账号、密码、密钥等），支持多种认证方式，并基于预设策略校验访问权限，过滤非法数据操作请求，保障数据安全。 ③连接池管理器 (Connection Pool Manager): 预创建并缓存一定数量的数据库连接，实现连接的复用与动态扩容 / 缩容，避免频繁创建 / 销毁连接带来的性能开销，同时监控连接状态（空闲、活跃、异常）并进行自动回收与故障重试。
基础 CRUD 操作 (Basic CRUD Operations)	作为数据交互的核心基础能力，封装对目标数据源（数据库、存储系统等）中数据的创建（Create）、读取（Read）、更新（Update）、删除（Delete）四类核心操作，实现数据的全生命周期基础管理，为上层业务逻辑提供直接、可靠的数据增删改查支撑。	①结构化数据 CRUD 引擎 (Structured Data CRUD Engine): 针对关系型数据库（如 MySQL、PostgreSQL）、结构化表格等场景，支持基于表结构、字段映射的精准操作，可关联主键 / 外键实现关联数据的 CRUD，适配 SQL 等结构化查询语言。 ②非结构化数据 CRUD 管理器 (Unstructured Data CRUD Manager): 面向文档、图片、音视频等非结构化数据，提供基于唯一标识（如文件 ID、哈希值）的创建（上传）、读取（下载 / 预览）、更新（覆盖 / 版本迭代）、删除（软删除 / 物理删除）操作，支持关联元数据的同步管理。

3.2.2.2 向量数据库专有子模块

如前文所示，向量数据库往往需要使用先行使用专门的嵌入模型将自然语言映射到一个高维向量空间（例如 384、768 或 1024 维），才能够进一步的进行存储并管理（插入、检索、删除等）。此外，针对向量数据这一不同的记忆存储方式，又可以进一步衍生出不同的记忆检索以及利用方法。对此，在下表中，针对该类 API 的功能描述以及泛化后的通用形态进行了进一步的说明。

组件	功能描述	泛化后的通用形态

向量嵌入 (Vector Embedding)	将文本、图像、音频等离散的符号或非结构化数据，通过深度学习模型映射到低维\高维连续的向量空间，并使其在该空间中的几何关系（如距离、方向）能够表示原始数据之间的语义或特征相似性。	① **嵌入模型 (Embedding Model)**: 这是核心功能形态，指任何能够产生嵌入向量的算法或神经网络（如 Word2Vec、BERT、CLIP、OpenAI 的 text-embedding-ada-002）。它定义了从数据到向量空间的映射规则。 ② **特征提取与嵌入 (Feature Extractors & Embedding):** 这是其更广泛的数学抽象，是语义向量的生成过程，也泛指任何将原始数据转换为更具代表性、更紧凑的特征向量的技术与实现。
相似度计算 (Similarity Computation)	核心功能是量化两个或多个数据对象（如向量）之间的关联程度或相近程度，为检索、去重、排序任务提供核心判断依据。	① **距离/相似度度量算法 (Distance/Similarity Metric Algorithms)**: 最基础的形态，提供一系列数学公式来定义"相似"或"相近"，如向量空间中的余弦相似度、欧氏距离等。 ② **近似最近邻搜索引擎 (Approximate Nearest Neighbor (ANN) Search Engine)**: 面向海量高维数据的高性能检索形态。它通过**牺牲少量精确度来换取极快的检索速度**，是使向量检索在大规模应用中可行的关键技术（如基于 HNSW、LSH 等算法）。
重排序 (Re-ranking)	对初步检索得到的候选结果列表进行精细化处理和重新排序，通过引入更多元、更复杂的特征或规则，从而提升最终结果列表的相关性、多样性或符合特定业务目标的程度。	**① 重排序模型 (Re-ranker Model):** 最典型的形态，通常是一个机器学习模型（如 BERT Cross-Encoder、LLM 模型）。它接收"查询结果"和"排序目标"作为输入，进行更深度的交互计算，输出一个更精确的相关性分数，用于重新排序。 ② **多目标排序策略 (Multi-objective Ranking Strategy)**: 一种更复杂的决策形态。它不再仅仅考虑"相关性"，而是综合平衡多个目标（如相关性、时效性、重要性、用户偏好）来定义一个更优的全局排序方案。

3.2.2.3 记忆管理专有子模块

如前文所示，随着对 LLM 内存理解的不断深入，研究者开始探索知识的显式操作方法，推动内存管理从隐式表示向工具化接口演进。所存储记忆需要能够进行自我管理、优化以及更新迭代。而基于管理并优化存储记忆的一系列研究，又可以进一步衍生出一系列的常见核心操作，这包括编码（将文本信息转换为 latent 空间嵌入，可以参考上一节中向量嵌入的功能介绍）、反思（提取更高层次的见解）、摘要（压缩文本同时突出关键点）、遗忘（选择性丢弃信息）、截断（在 token 限制内格式化）和判断（评估信息的存储优先级）等。对此，在下表中，针对该类 API 的功能描述以及泛化后的通用形态进行了进一步的说明。

组件	功能描述	泛化后的通用形态
反思 (Reflection)	对 LLM 记忆库中的碎片化信息（如对话交互、推理过程、任务反馈等）进行深度梳理与关联分析，挖掘信息背后隐含的逻辑关联、经验规律、核心观点或潜在问题，将低层次的事实性信息提炼为具有概括性、指导性的更高层次见解，强化记忆的认知深度与决策支撑价值。	①逻辑关联洞察器 (Logical Association Insight Extractor)：通过分析记忆中信息的因果关系、时序脉络、矛盾点等逻辑要素，提炼隐含的逻辑规律或决策框架，适用于推理过程复盘、问题归因等场景。 ②经验规律提炼器 (Experience Rule Refiner)：针对多次相似任务的记忆记录，挖掘共性特征与差异化结果，总结可复用的经验准则或避坑要点，形成结构化的认知沉淀。 ③观点立场聚合器 (Viewpoint Position Aggregator)：对多轮对话或多源信息中的分散观点进行归类整合，识别核心立场与支撑依据，生成系统性的见解摘要，适配意见汇总、需求解读等场景。
摘要 (Summarization)	针对 LLM 记忆中冗长、繁杂的信息（如长对话历史、复杂文档片段、多轮推理记录），采用语义压缩与关键信息提取策略，在保留核心事实、核心观点与逻辑主线的前提下，缩减信息体积，生成简洁且重点突出的浓缩内容，提升记忆检索与上下文调用效率。	① 事实型摘要生成器 (Factual Summary Generator)：聚焦记忆中的客观事实（如时间、地点、数据、事件要素），剔除冗余修饰与重复表述，生成以关键事实为核心的结构化摘要，适用于数据记录、事件复盘等场景。 ② 观点型摘要提炼器 (Viewpoint Summary Extractor)：针对包含主观观点、论证过程的记忆内容，提取核心论点、支撑论据与结论，梳理逻辑链条，生成兼具观点完整性与简洁性的摘要，适配对话观点汇总、文档核心思想提炼等场景。 ③ 层级化摘要生成器 (Hierarchical Summary Creator)：支持按不同层次\粒度生成摘要（如故事章、节、册、全文摘要），可根据记忆用途与调用需求动态调整摘要详略程度，满足不同场景下的信息获取需求。
判断 (Judgment)	评估一段信息的价值、相关性、可靠性或紧迫性，并据此决定其**存储的优先级**、存储形式乃至保留时长，是记忆管理系统的决策核心。	① 相关性判断器 (Relevance Judger)：基于当前任务主题、用户历史需求与信息内容的语义相似度，评估信息的关联程度，按相关度高低界定优先级，高相关信息优先存储与检索。 ② 稀缺性判断器 (Scarcity Judger)：识别记忆中独特性高、获取难度大的信息（如罕见领域知识、专属用户资料），赋予其高存储优先级，避免稀缺信息被误清理。 ③ 影响度判断器 (Impact Judger)：分析信息对后续任务推进、决策质量提升或认知深化的潜在影响，通过量化评分界定优先级，高影响度信息纳入核心记忆层重点管理。 ④ 用户关注度判断器 (User Attention Judger)：结

		合用户交互行为（如重复提及、重点标注）识别高关注度信息，提升其存储优先级，适配个性化记忆管理场景。
遗忘 (Forgetting)	根据特定策略主动且选择性地丢弃或削弱记忆中的信息，旨在保护隐私、消除偏差、遵守法规（如"被遗忘权"）、以及**优化系统资源管理**。	① **记忆衰减与垃圾回收机制 (Memory Decay & Garbage Collection Mechanism)**: 模拟人脑的遗忘曲线，根据信息的新近度、访问频率、重要性得分等因素，为记忆单元分配一个强度值或存活时间（TTL），并定期清理低于阈值的内容，这是计算系统中的标准资源管理策略。 ② **认知偏置校正器 (Cognitive Bias Corrector)**: 识别并主动削弱或隔离那些可能导致模型产生不良输出（如事实错误、有害偏见、垃圾内容）的记忆内容，确保知识库的准确性和中立性。
截断 (Truncation)	在有限的上下文窗口容量内，**对信息进行裁剪和格式化，**以优先保留最相关、最重要的内容，确保核心指令和关键上下文能够被模型处理。	① **滑动窗口优先级队列 (Sliding Window Priority Queue)**: 维护一个固定大小的容器，当新元素加入时，最旧（FIFO）或优先级最低的元素被移出。它确保了系统始终在硬性约束（Token 限制）下运行。 ② **上下文感知过滤器 (Context-Aware Filter)**: 并非简单丢弃旧信息，而是根据当前对话的实时目标和焦点，动态地评估所有历史消息的相关性，并过滤掉无关的旁支话题，动态摘要信息生成、反思以及遗忘，保留与当前任务最连贯的上下文链。

3.2.3 统一 API 服务

作为智能体（Agent）与外部世界交互的**中枢神经系统**，该网关的核心功能是**为 AI 智能体提供一套标准化、安全、可靠且高性能的机制，以调用各种异构的外部功能、API 和服务（统称为"工具"）**。它将智能体的"工具调用"意图转化为具体的 API 请求，并返回结构化的结果，从而极大地扩展了智能体的能力边界，使其能够执行诸如查询天气、执行代码、操作数据库、发送邮件等无数现实世界任务。

该网关的泛化形态是一个**工具调用即服务（Tool-Calling-as-a-Service, TCaaS）平台**，其核心模块组成如下：

组件	功能描述
工具注册与管理中心 (Tool Registry & Management)	这是网关的"元数据库"，是所有可用工具的**清单和说明书**。**泛化形态：**一个存储所有工具元数据的目录系统（如使用 `SQL/NoSQL` 数据库）。每个工具的定义包括： • **名称、描述：** 供 LLM 理解工具用途。 • **参数模式：** 严格的 JSON Schema，定义输入参数的名称、类型、是否必需、描述等（例如，`{"city": {"type": "string", "description": "...}}`）。

组件	功能描述
	• **端点信息：** 实际调用的 API URL 和方法（GET/POST）。 • **认证方式：** 该工具所需的特定认证（如 API Key、OAuth 2.0）。 • **分类与标签：** 便于对工具进行检索和分组。
统一 API 层 (Unified API Layer)	为智能体提供**唯一且一致的调用接口**。无论后端工具千差万别，智能体都通过同一个 API 端点（例如 `POST /v1/tool/call`）发起请求。**泛化形态：**一个标准的 RESTful 或 HTTP/2 API，请求体和响应体格式固定。请求体包含工具名和参数，响应体统一返回 `success`、`data`、`error` 等字段。这使得智能体无需关心底层集成细节。
认证与授权中间件 (AuthN & AuthZ Middleware)	**对工具本身所需凭证的管理：** 网关集中管理所有第三方工具的 API 密钥或 OAuth Token。当网关将请求转发给第三方服务时，自动在请求头（Header）或参数中注入相应的凭证。**智能体本身永远接触不到这些敏感信息**，极大增强了安全性。

3.3 智能体层 (Agent Layer)

此层由基于 Polytool 框架构建的具体应用智能体构成。它们利用下层提供的强大上下文构建能力，专注于特定领域的任务逻辑。

比如 Code Reviewer 智能体:它接收 Code Diff，Polytool 为其装配相关代码、历史评审、最佳实践等上下文，然后由它生成评审意见。

客户支持智能体 (Support Bot):接收用户问题，Polytool 为其装配产品文档、历史工单、解决方案库等上下文，然后由它生成回复或创建工单。

数据分析智能体 (Analyst Agent):接收自然语言查询（如“上月销售额趋势”），Polytool 为其装配数据库 Schema、相关报表 API，由它生成 SQL 或调用 API 并解读结果。

内部知识库问答智能体 (QA Agent):接收员工问题，Polytool 为其检索公司内部文档、手册、会议纪要，由它生成摘要和回答。

4. 核心优势

1.领域无关 (Domain-Agnostic):通过资源层的抽象和管理层的通用流程，Polytool 可以快速适配到任何需要 LLM+知识+工具的领域。

2.关注点分离 (Separation of Concerns):**应用开发者（智能体层）只需关心领域逻辑，无需处理复杂的上下文构建、检索和工具调用细节**。基础设施开发者则专注于优化管理层和资源层的效率与能力。

3.持续学习与记忆 (Continuous Learning & Memory):向量存储和图数据库的引入，使智能体拥有了长期、结构化、可检索的记忆，而不仅仅是短暂的会话记忆。

4.可观测性与可控性 (Observability & Control):所有检索到的上下文来源都可以被记录

和审查，这增加了 AI 决策的透明度和可信度，也便于人类纠正和干预。

5. 总结

Polytool（不器）将“资源管理”和“上下文构建”能力产品化、通用化，形成了一个强大的智能体基础架构。它不再是单一功能的“器”，而是一个能够支撑各种高级 AI 应用的“道”，真正体现了“君子不器”的思想，成为一个灵活、强大且适应性广泛的多用途智能体平台（Polytool）。