企业级场景中大模型应用开发工具的选型分析

杨 赛[1]
YANG Sai

摘 要 2025年，大模型技术已从概念走向量产，企业关注点不再是“能否用上大模型”，而是聚焦“如何快速、安全、低成本地实现落地”。当前低代码平台、云原生框架与模块化引擎三条技术路线并行，决策者亟需一份兼顾交付速度、治理成熟度与性能天花板的选型指南。文章通过对比Dify、Spring AI Alibaba与LangChain4j在全链路开发中的真实表现——从拖拽式模板、单JAR云原生到模块组件自由组合，逐一拆解三款工具在开发、治理、性能、观测及扩展维度的优缺点，旨在为中小型企业快速验证、中大型在线业务部署及超高并发场景深度定制三类需求，提供清晰的技术选型路径，帮助企业在有限资源内做出最优技术决策。

关键词 大模型开发工具；Java生态；Dify；Spring AI Alibaba；LangChain4j

doi：10.3969/j.issn.1672-9528.2025.12.017

0 引言

如今，人工智能已从概念验证阶段迈入深度应用时代。随着算力突破、算法革新与数据生态的协同发展，大模型技术快速应用到医疗、教育、制造等核心领域。这些拥有千亿级参数的智能系统，不仅能够通过自然语言交互完成复杂任务，更展现出解决现实问题的实际能力[1]。

在面向企业级的应用场景中，Java作为一种广泛采用的编程语言，凭借其稳定性和丰富的生态系统，成为了开发复杂业务逻辑和服务端应用的首选语言，目前Java语言仍是企业开发语言的首选，这一现状为AI大模型的落地提供了广阔场景。近年来，随着AI技术的不断成熟及硬件成本的有效降低，AI大模型在企业级应用中的普及成为必然趋势[2]。因此，构建适用于Java生态系统的AI框架，满足企业级开发的需求，已成为亟待解决的关键问题。

然而，现有大模型开发工具的研究与实践仍存在多方面局限，难以支撑Java生态企业的科学选型：

一是研究聚焦国外非Java生态工具，Java专属工具分析存在空白。当前学界对大模型开发工具的讨论多围绕Python生态的LangChain、LlamaIndex展开，极少涉及Java生态工具，这种“重Python、轻Java”的研究现状，导致Java技术型企

1. 浪潮软件股份有限公司 山东泰安 260000

[12] ZHU Y C, ZHUANG F Z, WANG D Q. Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources[C]//AAAI'19/IAAI'19/EAAI'19: Proceedings of the 33rd AAAI Conference on Artificial Intelligence and 31st Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence. Palo Alto: AAAI Press, 2019: 5989-5996.

[13] XU R J, CHEN Z L, ZUO W M, et al. Deep cocktail network: multi-source unsupervised domain adaptation with category shift[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Piscataway: IEEE, 2018: 3964-3973.

【作者简介】

李林（1992—），男，黑龙江佳木斯人，硕士研究生，研究方向：深度学习、迁移学习。

王闯（1995—），男，安徽滁州人，硕士研究生，研究方向：迁移学习、模式识别。

杜文博（2002—），男，河南驻马店人，硕士研究生，研究方向：无线网络、机器学习。

俞璐（1973—），女，吉林长春人，博士，副教授，研究方向：多媒体信息处理、模式识别、图像处理等。

李宁（1991—），男，江苏连云港人，硕士研究生，研究方向：智能化训练。

（收稿日期：2025-07-03 修回日期：2025-12-16）

业难以获取针对性参考。

二是缺乏定量对比维度，选型依据多为定性描述。现有文献对工具的评价多停留在“功能有无”的表层分析，未提供可量化的性能、成本指标，导致企业选型时缺乏数据支撑。

三是忽视工程化治理维度，仅关注功能与性能的局部分析。大模型工具的企业级落地不仅需“能开发”，更需“能运维、可治理”，但现有研究多聚焦“功能实现”（如工具调用、链式流程），对服务治理（如限流、熔断）、运维成本（如部署复杂度、故障排查效率）的分析严重不足，导致部分工具虽在功能测试中表现优异，但因无法融入企业现有 Java 治理体系（如 Nacos 配置中心、Sentinel 限流），最终落地失败。

四是无场景化选型模型，无法匹配企业差异化需求。不同规模、不同业务场景的企业对工具的需求存在显著差异：中小型企业更关注“开发速度”，大型企业更重视“并发稳定性”，金融、工业等领域则对“实时性”要求极高。但现有研究大多数仅提出“通用选型建议”，未区分“10 人团队的内部工具”“日均百万调用的用户端应用”“毫秒级响应的风控场景”等差异，这种“一刀切”的结论，导致企业难以找到贴合自身需求的工具。

针对上述研究空白与企业痛点，本文以 Java 生态企业级大模型应用开发需求为核心，在研究设计方面有三个先进性：

（1）构建 Java 生态核心工具的定量对比体系。区别于现有研究的定性描述，本文以 Dify、Spring AI Alibaba、LangChain4j 为研究对象，引入多项量化指标，结合官方技术公告、社区压测报告等验证数据，形成可直接复用的对比数据集；

（2）建立“功能 - 性能 - 治理”三维分析框架。突破现有研究“重功能、轻治理”的局限，将“服务治理（限流、熔断、链路追踪）”“运维成本（部署复杂度、升级效率）”纳入分析维度，填补 Java 生态工具工程化能力研究的空白；

（3）提出场景化选型指南。针对“中小型快速验证”“中大型高并发业务”“超高实时性深度定制”三类典型场景，结合工具特性与企业需求的匹配度，形成“规模 - 场景 - 工具”的对应关系，解决现有研究“无差异化建议”的问题，为 Java 生态企业提供“按需选型”的实操依据。

1 工具介绍

Dify 推出于 2023 年 5 月，是一个开源的低代码无代码 AI 应用开发平台，由 Langgenius 团队开发，遵循 Apache 2.0 许可证。其核心理念是让用户通过直观的可视化界面，定位是 AI 应用的快速开发与管理平台，它填补了传统开发与业务需求之间的鸿沟定义和构建专属的 AI 应用，无需深入的编码经验 [3]。Dify 结合了后端即服务（backend-as-a-service, Baa）和大语言模型运维（LLMOps）的理念，提供了一个端到端的开发环境，涵盖数据管理、模型调用、工作流设计、插件集成和应用发布。它允许将模型能力进行封装，并通过 API 发布，可以完美适配各种语言应用。

Spring AI 是 Spring 官方于 2024 年 10 月推出的开源框架，无缝集成了 Spring Boot、Spring Cloud 等组件，借助 Spring Boot 的自动配置功能以及 Spring Cloud 的分布式系统解决方案 [4]，使得开发者可以在熟悉的 Spring 开发环境中轻松引入 AI 功能，极大降低了 Java 开发者使用 AI 技术的门槛，无需额外学习复杂的 AI 框架或工具，只需遵循 Spring 的开发规范，即可快速构建出具有智能交互能力的应用系统。而 Spring AI Alibaba 则是阿里基于 Spring AI 的企业级扩展，更加适配国内生态环境，适配通义千问等多种国产大模型，优化中文场景的 Prompt 工程与检索增强生成（RAG）能力。

LangChain4j 是 LangChain 生态的 Java 实现，于 2024 年 7 月发布了 1.0.0-alpha 版本，其专注于构建模块化、可扩展的大语言模型（LLM）应用。核心设计理念为链式编程（Chaining），通过标准化接口集成 OpenAI、Anthropic 等 20+ 主流模型，并内置 RAG 工具、记忆管理、Agent 框架等模块 [5]。开发者可灵活组合功能链（如知识检索 + 文本生成），快速实现智能问答、文档分析等场景，最新版本也强化了流式响应支持与国产模型适配。

2 工具对比

2.1 功能开发

Dify 在功能快速落地表现最为突出，其作为一款低代码 AI 开发平台，用户无需编写任何代码，仅需拖拽即可完成整个 AI 功能的开发。模板中心内置多种模版，拖拽即可生成调用 API，可视化工作流程和丰富的组件，也极大加速了工作流程，内置 RAG 全链路（文档解析 → 向量化 → 检索 → 生成），高效易用，功能从开发到完成整套流程以小时计算，在企业级应用快速开发落地中表现良好。然而其作为低代码应用，一些缺点也是不可避免的，在应用颗粒度方法，只能通过参数进行控制，无法做到算法插拔调整，同时其 Agent 只能有向无环调用，复杂多 Agent 协作尚不支持，另外，在复杂业务流中仍需 Python 代码处理 LLM 输出结果，而中小企业非技术人员普遍缺乏 Python 基础，导致落地门槛升高，最后，Dify 应用在敏感信息的权限控制、数据安全等方面缺乏有效措施，可能会对企业信息安全造成难以预料的损失。

Spring AI Alibaba 完全基于 Spring 生态，只需遵循 Spring 的开发规范，即可和开发传统应用一样快速构建出具有智能交互能力的应用系统。与 Dify 等低代码开发平台相

比，它主张完全由开发者掌控，对任何功能的扩展基本上都不存在障碍，具备极高的灵活度。在开发模式上，延续了 Spring 注解 + 自动配置的风格，RAG、Tool、向量库为 Starter，一行注解即可注入，通过 Bean 暴露关键扩展点，Splitter、Retriever、PromptTemplate，均可被 @Bean 覆盖，颗粒度≈“类级别”。通过 pom 文件注入相关依赖，Java 团队可以将其无缝接入现有 MVC 与治理体系，可以沿用已有的权限控制，数据安全等功能。在功能快速落地方法，Spring AI Alibaba 与 Dify 还有一定差距，主要原因为开发人员需要熟悉框架中的相关 API，且编写代码需要一定时间，另外 Spring AI 的部分新功能（多智能体协作）案例和文档较少，出现问题后排查难度较高。

LangChain4j 是 LangChain 生态的 Java 实现，与 Spring AI Alibaba 类似，他也主张由开发者对功能的完全掌控，所有功能代码由开发人员编写。不同的是，LangChain4j 不再强依赖于 Spring 生态，而且更加灵活，颗粒度更细，它的所有能力均以 Interface 的形式暴露，如 EmbeddingStore、Retriever、Memory、PromptTemplate 等，将这些能力作为可插拔组件，允许开发者自由组合，也可以添加循环、并行、快照、多模型路由。另外，其通过统一 API 接口也实现了与其他语言和 LLM 服务间的无缝对接，拥有了多语言项目的协作能力。也正是由于它高度自由的特性，使得开发团队需要自行选择插件，需要编写大量代码，且不断调优尝试，学习曲线陡峭，另外，LangChain4j 由开源社区主导开发，功能全面但是缺少大厂官方背书，稳定性方面可能存在隐患。

2.2 性能指标

Dify 作为低代码快速开发平台，其架构设计更侧重易用性而非极致性能。Dify 共用流程引擎，在工作流执行过程中需要频繁更新状态、存储观测数据，这种设计虽然加快了开发效率，但是也降低了性能，链路长、IO 频繁，导致 CPU 占用过高，在企业级场景中难以承受高并发请求 [6]。

此外，Dify 将大部分功能设计为插件形式，方便了用户，但是带来了组件间的频繁通信，延迟了响应速度。Dify 的底层开发语言也是其性能短板之一，与 Java、Go 等语言相比，Python 在 CPU 密集型任务中的表现并不突出。根据 Dify 官方技术公告和阿里云压测报告显示，在 4C8G 单实例下，峰值 RPS≈10，p99 延迟 ≈60 s，成功率 <10%，必要情况下需要进行集群部署。

Spring AI Alibaba 和 LangChain4j 使用 Java 语言编写，性能基础上比 Python 更加优越。Spring AI Alibaba 基于 Spring 生态的线程池和负载均衡机制，观测带来的性能损耗仅 5%，并且使用 Reactor+Netty 全链路异步来提高性能，开发者也可以更精细的调控应用资源、优化 JVM 参数。

根据社区压测报告显示，最大 RPS 可达 150，是 Dify 的 10 倍，且 p99 延迟约为 18 s，成功率 99% 以上。而 LangChain4j 依赖于其轻量级设计，仅依赖 JAR，无容器、无 DB、无向量库强制捆绑，此外也可以参照官方示例配置异步 + 多模型负载均衡，实现吞吐量线性增长。LangChain4j-benchmark 数据显示，4 GB 内存情况下，100 并发内存占用率仅 50%，较 Spring AI Alibaba 降低 15%，配置了异步 +4×Ollama 路由模型后，RPS 可达 1 250，p99 延迟 3 s 以内，成功率 99% 以上。

2.3 服务治理与运维

Dify 的运维与治理注重自动化。单机部署时，只需执行“docker compose up -d”，即可一次运行 9 个容器完成启动，同时也提供生产级的 Helm Charts、Terraform（Azure/GCP）与 AWS CDK 脚本，支持裸机、K8s、公有云三种部署方式。Dify 内置 Nginx 反向代理，允许用 Higress 网关替代，提供限流、负载均衡策略。自带 LLMOps 仪表盘，可以查看调用次数、Token 消耗等内容，也可查看节点内的输入输出记录。同时，对于 Dify 功能的升级或者回滚功能只需导入对应的 DSL 文件即可，运维方便快捷。当然，Dify 开箱即用的便利也有一定代价，首先，资源占用过高，单机启动仍需 9 个组件；其次，服务治理只能在网关中进行，到节点级别，无法做到精确限流和分布式熔断；此外，其跨服务链路追踪需要手动拼接。

Spring AI Alibaba 基于 Spring 生态，运维工作与传统 Java 应用完全一致。单机部署直接 Jar 包启动，集群部署根据企业规范选取不同的策略，服务治理方面完全继承 Spring Cloud 全量能力，根据技术机构自由选取，此外，也提供了 AI 专属治理能力，包括 Token 级限流、模型 Fallback，与 RPC 治理策略统一。链路追踪方面，实现了跨服务 TraceId 自动透传，零代码入侵，排查问题更加方便准确。此外也可以通过 Nacos 配置实现模型的热切换而无需重启服务。对于部分高级功能，比如 GPU 负载均衡，模型级别的熔断，需要依赖阿里云 Maas，本地模型暂不具备此类能力。

LangChain4j 的服务治理也是极其自由，所有观测、限流、生命周期都支持自定义处理，纯 Jar 包升级只需要改 maven 版本。但这也正是他的弊端，灵活但需自建治理体系，这是个极大的工作量，远不如使用现有的成熟的企业级治理方案简单可靠，当然也可以将其融入 Spring Cloud 体系来获取现有的治理能力。日志观测、回调、熔断、状态快照等暂无官方产品支持，需要手动处理，这也会为开发团队带来一定的工作量。

3 选型总结

Dify 作为一款低代码 AI 应用开发平台，尽管在高并发处理、资源利用率等方面存在固有瓶颈，但凭借“低门槛、快落地”的核心优势，仍是中小型公司与中小型 AI 应用的首选方案。其可视化工作流编辑器、预制的模型调用插件（如 OpenAI、通义千问适配）及轻量化运维设计，能让缺乏深度 AI 技术团队的中小企业无需投入大量时间攻克技术细节，即可做到产品迅速落地，抢占市场。如顺丰速运内部 200+ 在线问答应用，使用 Dify 在 2 周内迅速落地投入使用 [6]。

而当业务规模扩大、需支撑高并发场景时，优先选择 Spring AI Alibaba 更为合适。其天然融入 Spring 生态，对于已采用 Spring Cloud、Nacos 等技术栈的企业，可实现无成本技术迁移，无需重构现有架构，配合内置的服务治理、全链路观测功能，既满足高并发下的性能需求，又保障系统稳定性，完美适配大型企业级应用的复杂诉求。如阿里云客服系统，在 B2B 商品咨询、订单查询方面使用 Spring AI Alibaba 开发，峰值可达 420QPS，满足“双十一购物节”等大促的智能问答流量需求。

如果业务对实时性要求极高（如金融风控的毫秒级 AI 决策），需追求极致性能，LangChain4j 则是更优解。其模块化设计允许开发者自由替换核心组件，或配置虚拟线程减少并发调度开销，结合灵活的缓存策略与工具调用逻辑调优，可将响应时间压缩至毫秒级，精准匹配高实时性场景下的性能要求。乐天集团自 2023 年起采用 LangChain4j 搭建多智能体系统（Rakuten AI for Business）并应用于金融数据领域，利用虚拟线程优化高并发场景，32 000 名员工使用无阻塞。

本文通过构建“功能 - 性能 - 治理”三维框架完成工具选型分析，虽为 Java 生态大模型工具选型提供了实证参考，但受限于研究条件与技术边界，仍存在以下几方面局限需在后续研究中完善。

首先，研究对象覆盖范围存在一定局限性。本文以 Dify、Spring AI Alibaba、LangChain4j 三类主流工具为例展开对比，但 Java 生态中的大模型开发工具还包括云厂商专属解决方案（如百度文心千帆 Java SDK、华为云 ModelArts Java 客户端）及轻量级框架（如 Hutool-AI）等。受限于资源与精力，未将此类工具纳入对比体系。

其次，测试场景与环境的复杂度有待提升，未涉及企业级典型复杂场景如多模型混合部署（本地 Llama3 与云端 GPT-4 协同）、超大规模上下文处理（10 万 token 以上长文档分析）及极端流量冲击（每秒 1 000+ 并发请求）。这种简化的测试环境可能导致对工具在高复杂度场景下的稳定性、资源调度效率等关键指标的评估不够充分。

最后，研究结论的时效性存在动态变化可能。大模型工具处于快速迭代期，本文测试基于 2025 年 9 月的工具稳定版本，但工具的核心功能（如 Dify 的多 Agent 协作能力）、性能优化（如 LangChain4j 的批处理效率）可能随版本更新发生变化，导致部分结论的适用周期受限。

上述局限未从根本上影响本文“场景适配”的核心结论，但为后续研究指明了方向：可进一步扩展工具样本库，构建更多维度的评价体系，并在集群环境下开展长期追踪测试，以提升选型建议的全面性与时效性。

参考文献：

[1] 包义明，林阳，屠要峰 . 人工智能技术与应用前沿 [J]. 中兴通讯技术 ,2025,31(4):64-74.

[2] 赵美展，焦春奇 .Java 技术在人工智能领域的应用探索 [C]// 广西网络安全和信息化联合会 . 第九届工程技术管理与数字化转型学术交流会论文集 . 哈尔滨 : 哈尔滨信息工程学院 , 2025:196-197.

[3] Langgenius 团队 . Dify：一站式大模型应用开发平台（v1.8.0）[EB/OL]. 2025[2025-09-20].https://github.com/langgenius/dify.

[4] Alibaba Cloud 技术团队 .SpringAI Alibaba：基于 Spring 生态的大模型开发框架 [EB/OL].(2025-06-22)[2025-09-20]. https://github.com/alibaba/spring-ai-alibaba.

[5] LangChain4j 社区 .LangChain4j：Java 生态大模型工具链（v2.1.0）[EB/OL]. (2025-07-05)[2025-09-20]. https://github.com/langchain4j/langchain4j.

[6] 陈迪豪 . 改造 Dify 实现生产可用的 AI Agent 应用落地 [EB/OL].(2024-11-09)[2025-11-22].https://www.aidd.vip/resources/upload/a7844a45d8ab55e/1731296775246/ 陈 迪豪 - 改造 Dify 实现生产可用的 AIAgent 应用落地 .pdf.

【作者简介】

杨赛（1995—），男，山东聊城人，本科，主任工程师，研究方向：Java 生态应用开发。

（收稿日期：2025-09-24 修回日期：2025-12-09）