项目新增工具调用框架及天气查询工具功能说明文档

文档用途

本文档旨在详细介绍项目新增的**工具调用框架**及核心功能，包括工具注册机制、统一调度能力、配置集中管理系统，以及具体的天气查询工具实现，帮助开发人员快速理解功能架构、使用方式及扩展方法。

目录

一、功能概述

本次新增功能围绕“工具调用框架”展开，核心目标是实现多类型工具的标准化管理、灵活调用与高效扩展，具体包括：

- 建立**工具注册机制**：集中管理所有可用工具的元信息（功能描述、参数要求等），支持新增工具快速接入；
- 提供**统一调度能力**：封装工具调用流程封装，屏蔽不同工具的调用差异，支持异步调用与异常处理；
- 实现**配置集中管理**：整合服务器、大模型、工具API等配置项，支持环境变量与文件配置，适配多场景部署；
- 落地**天气查询工具**：作为示例工具，支持心知天气、和风天气多源数据获取，无API时自动返回模拟数据，验证框架可行性。

该框架可直接支撑上层应用提供天气、新闻、股票等工具的调用能力，且支持快速工具（如翻译、地图）的快速扩展。

二、核心功能模块说明

1. 工具注册表 - 统一管理可用工具

功能定位

作为工具的“登记中心”，记录所有可用工具的关键信息，是调度工具识别、调用校验的基础，确保工具接入标准化。

核心实现

- **工具元信息管理**：通过 `TOOLS_REGISTRY` 字典存储工具信息，每个工具包含：
  - `function`：工具实际执行函数（如天气查询的 `get_weather`）；
  - `description`：功能描述（用于上层应用展示，如“天气查询工具，支持7天预报”）；
  - `required_params`：必传参数列表（如天气工具的 `location`）；
  - `optional_params`：可选参数列表（如天气工具的 `days`）。
- **已注册工具列表**：

工具名称	功能描述	必传参数	可选参数
weather	天气查询（支持7天预报）	`location`	`days`
news	新闻检索	`query`	`limit`、`category`
stock	股票数据查询	`symbol`	`days`
calculate	数值计算	`expression`	`variables`
document	文档生成	`template`、`content`	`data`、`format`

关键作用

- 标准化工具接入标准，新增工具只需按格式注册即可被调度器识别；
- 明确参数约束，为调用前的参数校验提供依据，减少无效调用。

2. 工具调度器 - 统一调用与流程控制

功能定位

作为工具调用的“协调中心”，封装工具调用逻辑，提供统一接口，处理调用过程中的异常与异步兼容，降低上层应用使用成本。

核心实现

- **ToolScheduler类**：
  - 初始化时加载注册表中的工具信息，存储于 `self.tools`；
  - `call_tool`（异步方法）：核心调用逻辑，步骤如下：
    - 校验工具是否存在（不存在则返回错误信息）；
    - 提取工具执行函数，通过 `asyncio.to_thread` 在后台线程执行（避免阻塞事件循环）；

- 捕获取消操作、运行时错误等异常，返回标准化结果（含 `success`、`error`、`data` 字段）。
- `get_available_tools`：返回所有工具的元信息（名称、描述、参数要求），方便上层应用展示可用工具列表。

关键特性

- **异步友好**：通过线程池执行同步工具函数，适配异步框架（如FastAPI）；
- **结果标准化**：所有工具调用结果格式统一，简化上层解析逻辑；
- **错误隔离**：单个工具调用失败不影响其他工具，提升系统稳定性。

3. 配置管理系统 - 集中管控环境与参数

功能定位

集中管理项目所有配置项，支持多环境（开发/生产）配置切换，避免硬编码，提升配置灵活性与安全性。

核心实现

- **Settings类**（基于Pydantic）：
  - **服务器配置**：绑定地址（`HOST`）、端口（`PORT`）、调试模式（`DEBUG`）；
  - **大模型配置**：API密钥（`LLM_API_KEY`）、接口地址（`LLM_BASE_URL`）、默认模型（`LLM_MODEL`）；
  - **工具API配置**：天气工具（心知天气 `WEATHER_API_UID` / `SECRET`、和风天气 `WEATHER_API_KEY`）、新闻（`NEWS_API_KEY`）、股票（`STOCK_API_KEY`）；
  - **工具调用控制**：超时时间（`TOOL_TIMEOUT`）、最大调用步骤（`MAX_TOOL_STEPS`）。
- **配置加载逻辑**：
  - 支持从 `.env` 文件加载配置，敏感信息（如API密钥）无需硬编码；
  - 优先级：环境变量 > `.env` 文件 > 默认值，适配不同部署场景；
  - 类型校验：自动校验配置项类型（如 `PORT` 必须为整数），减少运行时错误。

关键作用

- 配置集中化，便于环境切换与运维管理；
- 类型安全校验，提前暴露配置错误；
- 可扩展性，新增工具的API配置可直接在 `Settings` 类中扩展。

4. 天气查询工具 - 多源数据与降级策略实现

功能定位

功能定位

作为框架的示例工具，实现天气查询功能，支持多数据源获取与自动降级，验证工具调用框架的可行性与稳定性。

核心逻辑

参数处理：

- 校验必传参数 `location`（城市名称），缺失返回错误；
- 处理可选参数 `days`（查询天数），默认7天，最大限制7天。

多源数据获取与降级：

- **优先使用心知天气API**（需配置 `WEATHER_API_UID` 和 `SECRET`）：
  - 调用接口获取数据，解析为标准化格式（含日期、天气、温度、湿度等）；
  - 处理温度单位转换（如 `"15°C"` 转整数15）。
- **降级使用和风天气API**（心知不可用时，需配置 `WEATHER_API_KEY`）：
  - 支持两种认证方式（请求标头/参数），失败时自动切换；
  - 先通过城市搜索接口获取城市ID（提升准确性），失败则直接用城市名称查询。
- **最终降级为模拟数据**（所有API不可用时）：
  - 基于城市名称和日期生成确定性模拟数据（相同输入返回相同结果）；
  - 模拟不同城市基础温度（如哈尔滨-18°C、广州18°C），增强真实性。

异常处理与结果返回：

- 捕获网络错误、API认证失败、数据格式错误等异常，触发降级策略；
- 返回标准化结果，包含执行状态（`success`）、数据（`data`）、错误信息（`error`）及元数据（耗时、数据源等）。

关键特性

- **多源冗余**：通过多API降级确保功能可用性，避免单一依赖失效；
- **格式统一**：屏蔽不同API的返回差异，提供标准化数据结构；
- **开发友好**：无API密钥时自动返回模拟数据，便于本地调试。

三、功能模块间关系

依赖关系：

- 调度器依赖工具注册表获取工具元信息，实现工具查找与调用；
- 天气查询工具依赖配置管理系统获取API密钥等配置；
- 所有工具的调用均通过调度器执行，工具自身无需关注调用流程。

数据流向：

- 上层应用 → 调度器 `get_available_tools()` → 获取可用工具列表；
- 上层应用 → 调度器 `call_tool(tool_name, params)` → 目标工具函数（如 `get_weather`）→ 返回标准化结果。

四、使用说明

新增工具步骤：

- 步骤1：实现工具函数（如新闻检索 `search_news`），接收 `params` 参数并返回标准化结果；
- 步骤2：在工具注册表中注册工具（填写 `function`、 `description`、参数列表）；
- 步骤3（可选）：在配置管理系统中添加工具所需API密钥等配置。

调用工具示例：

```
from app.tools.scheduler import ToolScheduler

# 初始化调度器
scheduler = ToolScheduler()

# 获取所有可用工具
available_tools = scheduler.get_available_tools()
print("可用工具：", available_tools)

# 调用天气工具查询北京3天天气
result = await scheduler.call_tool(
    tool_name="weather",
    parameters={"location": "北京", "days": 3}
)
print("调用结果：", result)
```

配置说明

- 新建 `.env` 文件配置敏感信息（如 **WEATHER_API_KEY=your_key**）；
- 调整 **TOOL_TIMEOUT**（工具超时）、**MAX_TOOL_STEPS**（最大调用步数）等参数控制工具行为。

五、总结

本次新增的工具调用框架通过“注册表 + 调度器 + 配置中心 + 示例工具”的组合，实现了工具的标准化管理与灵活调用：

- 工具注册表解决了工具集中管理与接入标准化问题；
- 工具调度器屏蔽了调用细节，提供统一接口与异常处理；
- 配置管理系统支持多环境适配，提升配置灵活性；
- 天气查询工具验证了框架的可行性，展示了多源数据与降级策略的最佳实践。

该框架可直接作为项目工具调用的基础组件，支持快速扩展翻译、地图等更多工具，为上层应用提供

该框架可直接作为项目工具调用的基础组件，支持快速扩展翻译、地图等更多工具，为上层应用提供稳定、可扩展的功能支撑。