PR #1981 完整报告

THUDM/slime

[agent] extract Adapter class

合并时间：2026-05-29 20:41

原文链接：http://prhub.com.cn/THUDM/slime/pull/1981

执行摘要

- 一句话：抽取 Adapter 基类，封装会话生命周期管理
- 推荐动作：该 PR 值得核心开发者精读，尤其是 BaseAdapter 的设计模式和类型安全 AppKey 的使用。外部用户若已使用 agent 适配器应尽快迁移。

功能与动机

消除 anthropic.py 与 openai.py 之间重复的模块级状态变量和函数，提供统一的面向接口的会话管理方式，降低新增适配器的门槛并提高可维护性。

实现拆解

1. 定义 BaseAdapter 与应用键（slime/agent/adapters/common.py）：新增 BaseAdapter 类，封装 store、inflight、closed 等状态和 open_session / shutdown_session / finish_session（抽象）方法；定义 ADAPTER_KEY、TOKENIZER_KEY 等类型安全的 web.AppKey，替换原有的字符串键。
2. 改造 AnthropicAdapter（slime/agent/adapters/anthropic.py）：将原有的 _Store、_inflight、_closed 模块变量移除，新类 AnthropicAdapter 继承 BaseAdapter，__init__ 调用基类构造后添加路由，finish_session 实现子代理和 final 段的合并。
3. 改造 OpenAIAdapter（slime/agent/adapters/openai.py）：类似地迁移为类，finish_session 仅处理最终段。
4. 更新模块导出（slime/agent/adapters/__init__.py）：导出 AnthropicAdapter、OpenAIAdapter、BaseAdapter。
5. 更新示例（examples/coding_agent_rl/generate.py）：使用 AnthropicAdapter 实例替代模块函数调用，调用 adapter.open_session / adapter.finish_session。
6. 更新测试（tests/test_agent_adapters.py、tests/test_agent_sdk_adapters.py）：适配新类语法，使用 adapter.app 创建 TestServer。
7. 同步文档（docs/*/get_started/agent.md 等）：更新为新的类使用方式。

关键文件：

- slime/agent/adapters/common.py（模块 核心；类别 source；类型 core-logic；符号 BaseAdapter, init, open_session, shutdown_session）：新增 BaseAdapter 基类和应用键，是所有适配器共用的生命周期管理核心。

- slime/agent/adapters/anthropic.py（模块 适配器；类别 source；类型 core-logic；符号 AnthropicAdapter, init, finish_session）：AnthropicAdapter 继承 BaseAdapter，实现会话结束时子代理和最终段的合并逻辑。
- slime/agent/adapters/openai.py（模块 适配器；类别 source；类型 core-logic；符号 OpenAIAdapter, init, finish_session）：OpenAIAdapter 继承 BaseAdapter，实现会话最终段的合并。
- examples/coding_agent_rl/generate.py（模块 示例；类别 source；类型 dependency-wiring）：示例代码迁移到新 API，使用 AnthropicAdapter 实例替代模块函数调用。

关键符号：BaseAdapter.init, BaseAdapter.open_session, BaseAdapter.shutdown_session, BaseAdapter.finish_session, AnthropicAdapter.init, AnthropicAdapter.finish_session, OpenAIAdapter.init, OpenAIAdapter.finish_session

关键源码片段

slime/agent/adapters/common.py

新增 BaseAdapter 基类和应用键，是所有适配器共用的生命周期管理核心。

```
class BaseAdapter:
    """Base HTTP adapter with per-instance session lifecycle state."""
    session_cls: type

    def __init__(self, *, tokenizer, sglang_url, tool_parser=None, reasoning_parser=None) ->
    None:
        # 每个实例拥有独立的会话存储、inflight 任务集和关闭标记，
        # 替代原来模块级的 _Store、_inflight、_closed 变量。
        self.store: dict[str, Any] = {}
        self.inflight: dict[str, set[asyncio.Task]] = {}
        self.closed: set[str] = set()
        self.app = web.Application(client_max_size=64 * 1024 * 1024)
        # 使用类型安全的 AppKey 代替字符串 key，IDE 可获得更好的补全和类型检查。
        self.app[ADAPTER_KEY] = self
        self.app[TOKENIZER_KEY] = tokenizer
        self.app[SGLANG_URL_KEY] = sglang_url.rstrip("/") if isinstance(sglang_url, str) else
        sglang_url
        self.app[TOOL_PARSER_KEY] = tool_parser
        self.app[REASONING_PARSER_KEY] = reasoning_parser

    def open_session(self, sid: str, *, sampling_defaults: dict | None = None, max_context_
    tokens: int = 0) -> None:
        # 委托给公共的 register_session 函数，但通过实例的 store 和 session_cls 完成注册。
        register_session(self.store, sid, self.session_cls, sampling_defaults=sampling_defaults,
        max_context_tokens=max_context_tokens)

    async def shutdown_session(self, sid: str, *, wait_timeout: float = 5.0) -> None:
        # 关闭会话时先取消所有 in-flight 任务，然后标记 closed。
        await shutdown_session_tasks(sid, self.closed, self.inflight, wait_timeout=wait_timeout)
```

```
    async def finish_session(self, sid: str, *, wait_timeout: float = 5.0) -> list[TokenSegment]:
        # 抽象方法，子类需实现如何从 store 中提取并合并 TokenSegment。
        raise NotImplementedError
```

slime/agent/adapters/anthropic.py

AnthropicAdapter 继承 BaseAdapter，实现会话结束时子代理和最终段的合并逻辑。

```
class AnthropicAdapter(BaseAdapter):
    """Anthropic Messages-compatible HTTP adapter with session lifecycle helpers."""
    session_cls = Session

    def __init__(self, *, tokenizer, sglang_url, tool_parser=None, reasoning_parser=None) ->
    None:
        super().__init__(
            tokenizer=tokenizer,
            sglang_url=sglang_url,
            tool_parser=tool_parser,
            reasoning_parser=reasoning_parser,
        )
        # 注册 Anthropic Messages API 特有的路由
        self.app.router.add_post("/v1/messages", _handle_request)
        self.app.router.add_post("/v1/messages/count_tokens", _count_tokens)
        self.app.router.add_get("/healthz", _ok)
        self.app.router.add_get("/v1/models", _ok)

    async def finish_session(self, sid: str, *, wait_timeout: float = 5.0) -> list[TokenSegment]:
        # 先关闭 inflight 任务并标记 closed
        await self.shutdown_session(sid, wait_timeout=wait_timeout)
        s = self.store.pop(sid, None)
        if s is None:
            return []
        # 如果存在活跃子代理且其 turns 不为空，先作为 subagent 段追加
        if s.active_sub is not None and s.active_sub.turns:
            s.segments.append(make_turn_segment(s.active_sub.turns, kind="subagent"))
        # 主链的 turn 作为 final 段
        if s.main.turns:
            s.segments.append(make_turn_segment(s.main.turns, kind="final"))
        return merge_turn_segments(s.segments, max_context_tokens=s.max_context_tokens)
```

slime/agent/adapters/openai.py

OpenAIAdapter 继承 BaseAdapter，实现会话最终段的合并。

```
class OpenAIAdapter(BaseAdapter):
    """OpenAI-compatible HTTP adapter with session lifecycle helpers."""
    session_cls = Session

    def __init__(self, *, tokenizer, sglang_url, tool_parser=None, reasoning_parser=None) ->
    None:
        super().__init__(
```

```
            tokenizer=tokenizer,
            sglang_url=sglang_url,
            tool_parser=tool_parser,
            reasoning_parser=reasoning_parser,
        )
        # 注册 OpenAI 兼容的聊天和响应端点
        self.app.router.add_post("/v1/chat/completions", _handle_chat_completions)
        self.app.router.add_post("/v1/responses", _handle_responses)
        self.app.router.add_get("/healthz", _ok)
        self.app.router.add_get("/v1/models", _ok)

    async def finish_session(self, sid: str, *, wait_timeout: float = 5.0) -> list[TokenSegment]:
        # 先关闭 inflight 任务并标记 closed
        await self.shutdown_session(sid, wait_timeout=wait_timeout)
        s = self.store.pop(sid, None)
        if s is None:
            return []
        # OpenAI 适配器不处理子代理，仅将主链 turn 作为 final 段
        if s.main.turns:
            s.segments.append(make_turn_segment(s.main.turns, kind="final"))
        return merge_turn_segments(s.segments, max_context_tokens=s.max_context_tokens)
```

评论区精华

- 暂无高价值评论线程

风险与影响

- 风险：
  1. API 兼容性风险：原有的 pop_session_split、start 等模块级函数被移除，任何直接调用它们的代码（如未迁移的测试或示例）将失败。
  2. 状态管理迁移：原本模块变量（_inflight、_closed）变为实例属性，确保同一进程内多个适配器实例不会互相干扰，但若原有代码依赖全局单例则需注意。
  3. 测试覆盖：测试文件已同步更新，但关键路径（如子代理 segment 拆分）的覆盖率需确认。 - 影响：对使用 agent 适配器的开发者：需从调用函数改为实例化类，并调用相应方法。对系统：无运行时性能影响，但代码结构更清晰。对团队：降低了后续新增适配器（如 Gemini）的重复劳动。 - 风险标记：API 迁移风险，旧函数移除，测试覆盖需确认

关联脉络

- PR #1979 [agent] Add openai and anthropic adapters: 该 PR 引入了最初的适配器模块，本 PR 将这些模块中的功能抽取为类，是前者的重构演进。
- PR #1960 Extract more util code from coding_agent_rl example: 类似地抽取通用工具到核心模块，与本 PR 的抽取思想一致，属于同一重构脉络。
- PR #1963 Fix trajectory merging logic: 涉及 trajectory 合并逻辑的修复，本 PR 中的 finish_session 依赖 merge_turn_segments，因此关联。