Конфигурация

Конфигурационный плагин бизнеса (snippet `Business.AgentsParams.BSPbConfig`) используется для централизованного управления параметрами агентов в системе MAS. Он определяет настройки для различных сценариев работы агентов, включая маршрутизацию, выбор моделей, параметры генерации, обработку ошибок и интеграцию с внешними инструментами.

Структура файла

Файл представляет собой JavaScript-объект, где каждая ветка соответствует определённому агенту или сценарию. Основные разделы:

- **common** — общие параметры агента (имя, таблица промптов, максимальная длина истории, сценарий выхода и др.)
- **detectRoute** — параметры для маршрутизации запросов (провайдер, модель, промпт, инструменты маршрутизации, обработчик ошибок)
- **detectIntent** — параметры для определения намерения пользователя (провайдер, модель, промпт, обработчик ошибок, база знаний)
- **userReply** — параметры для формирования ответа пользователю (провайдер, модель, история, промпты, обработчик ошибок)
- **execSQL** — параметры для SQL-агентов (провайдер, модель, промпт, обработчик ошибок)

Пример структуры

```
{
  common: {
    title: "Основной Flow с маршрутизатором",
    agentName: "bsp",
    promptTable: "gpt_prompts",
    userQueryAttibName: "user_query",
    historyMaxLength: 4,
    useRoute: 0,
    exitScript: "MainFlow_FollowUp",
    MBQuery_fallback: {
      script_code: "MBQuery_TimeOut",
```

```
      timeout: 180
    }
  },
  detectRoute: {
    provider: "OpenAI",
    model: "gpt-5-mini",
    modelParams: { temperature: 1 },
    prompt: "$route_prompt",
    routerTools: [ ... ],
    errorScript: "RAG_ErrorFallback"
  },
  detectIntent: {
    provider: "OpenAI",
    apiFormat: "OpenAI",
    model: "gpt-5-mini",
    modelParams: { temperature: 1 },
    prompt: "$rag_intent_prompt",
    errorScript: "RAG_ErrorFallback",
    kbName: "defKnowBase",
    kbDomain: "tech_domain"
  },
  userReply: {
    provider: "OpenAI",
    apiFormat: "OpenAI",
    model: "gpt-5-mini",
    modelParams: { temperature: 1 },
    errorScript: "RAG_ErrorFallback",
    useHistory: 1,
    addUserQuery: 1,
    sendBotAnwser: 1,
    systemPrompts: {
      start: [["$start_prompt", "$rag_prompt"]],
      final: ["$final_prompt"]
    }
  }
}
```

Как работает подстановка

1. **Загрузка snippet**: Конфигурация подгружается через snippet с именем `Business.AgentsParams.BSPbConfig`.
2. **Выбор активного агента**: В коде определяется активный агент через переменную `activeAgent` (например, `bsp`, `CompanyInfo`, `Table1`).
3. **Инициализация agentCFG**: В зависимости от значения `activeAgent` выбирается соответствующая ветка конфигурации и присваивается переменной `agentCFG`.
4. **Передача параметров**: agentCFG используется для инициализации LLMClient, настройки промптов, истории, провайдера, модели и других параметров.
5. **Маршрутизация и обработка**: Ветка `detectRoute` определяет инструменты маршрутизации, которые используются для выбора сценария обработки запроса пользователя.
6. **Обработка ошибок**: Для каждого сценария можно задать обработчик ошибок (`errorScript`) и fallback-скрипты.
7. **Интеграция с базой знаний**: Ветка `detectIntent` может содержать параметры для подключения к базе знаний (`kbName`, `kbDomain`).

Структура кода

```
let aAgent = lead.getAttr("activeAgent")
let agentCFG
if (aAgent === "bsp") {
  agentCFG = { ... } // параметры для bsp
} else if (aAgent === "CompanyInfo") {
  agentCFG = { ... } // параметры для CompanyInfo
} else if (aAgent === "Table1") {
  agentCFG = { ... } // параметры для Table1
} else {
  bot.sendMessage(`BSPbConfig:snippet - неизвестный activeAgent:  ${aAgent}`)
  bot.stop()
}
```

Куда подставляются параметры

- **LLMClient**: параметры модели, провайдера, истории, промптов и сценариев подставляются при создании и настройке клиента.
- **Маршрутизатор**: инструменты из `routerTools` используются для выбора сценария обработки.
- **Обработка ошибок**: скрипты из `errorScript` и `MBQuery_fallback` используются для fallback-логики.

- **Промпты**: значения из `systemPrompts` подставляются в соответствующие методы LLMClient для формирования запроса.
- **База знаний**: параметры `kbName` и `kbDomain` используются для интеграции с внешними источниками знаний.

Рекомендации

- Все параметры должны быть явно определены для каждого сценария, чтобы избежать ошибок при маршрутизации и генерации ответа.
- Для расширения функционала добавляйте новые ветки конфигурации с нужными параметрами.

FAQ

Вопрос: Как добавить нового агента?
Ответ: Добавьте новую ветку в объект конфигурации с нужными параметрами и обработчиками.

Вопрос: Как изменить модель или провайдера?
Ответ: Измените значения `model` и `provider` в нужной ветке конфигурации.

Вопрос: Как задать fallback-скрипт?
Ответ: Укажите параметры в `MBQuery_fallback` или `errorScript` для нужного сценария.

Вопрос: Как интегрировать базу знаний?
Ответ: Добавьте параметры `kbName` и `kbDomain` в ветку сценария, где требуется интеграция.

Версия #1
Павел Борисов создал 2025-11-06 11:42:57 UTC
Павел Борисов обновил 2025-11-12 14:03:47 UTC