Документация по LLMClient

Как работает

LLMClient реализует ключевой паттерн **Metabot Agent System (MAS)** — фреймворк для построения мультиагентных систем через композицию простых, понятных блоков кода.

MAS следует принципу "**сложное через простое**":

- **Декларативность**: Сложные AI-операции описываются простыми, читаемыми выражениями
- **Модульность**: Каждый flow — независимая единица работы с собственным контекстом
- **Прозрачность**: Вся логика работы агента видна в коде скрипта/плагина
- **Композиция**: Сложные мультиагентные сценарии собираются из простых блоков

Методы класса LLMClient

new LLMClient(sessionName, agentName = "", apiFormat = "")

Создаёт новый экземпляр клиента для работы с LLM.

Параметры

Имя	Тип	Описание
sessionName	String	Уникальное имя сессии
agentName	String	Имя агента (опционально)
apiFormat	String	Формат API (опционально)

Возвращает
LLMClient — экземпляр клиента.

---

setPromptTable(tableName, agentName)

Устанавливает таблицу промптов и имя агента.

Параметры

Имя	Тип	Описание
tableName	String	Имя таблицы промптов
agentName	String	Имя агента

---

addUserQuery(content)

Сохраняет пользовательский запрос для текущей сессии.

Параметры

Имя	Тип	Описание
content	String	Пользовательский запрос

---

getUserQuery()

Получает последний пользовательский запрос.

Параметры
Нет

Возвращает
String — последний пользовательский запрос.

---

addPrompt(role, content)

Добавляет промпт с указанной ролью.

Параметры

Имя	Тип	Описание
role	String	Роль: 'system', 'user', 'assistant'
content	String/Array	Текст или массив строк

---

addRawPrompts(prompts)

Добавляет массив промптов без обработки.

Параметры

Имя	Тип	Описание
prompts	Array	Массив объектов {role, content}

---

addSystemPrompt(content)

Добавляет системный промпт.

Параметры

Имя	Тип	Описание
content	String	Системный промпт

---

addUserPrompt(content)

Добавляет пользовательский промпт.

Параметры

Имя	Тип	Описание
content	String	Пользовательский промпт

---

addAssistantPrompt(content)

Добавляет промпт ассистента.

Параметры

Имя	Тип	Описание
content	String	Промпт ассистента

---

addHistoryToPrompts(historyOverride = null)

Добавляет историю сообщений к промптам.

Параметры

Имя	Тип	Описание
historyOverride	Array/Null	Массив сообщений или null

---

clearPrompts()

Очищает все промпты текущей сессии.

Параметры
Нет

---

setHistoryMaxLength(lmax)

Устанавливает максимальную длину истории сообщений.

Параметры

Имя	Тип	Описание
lmax	Number	Максимальная длина истории

---

setHistory(messages)

Сохраняет историю сообщений.

Параметры

Имя	Тип	Описание
messages	Array	Массив объектов {role, content}

---

getHistory()

Получает историю сообщений.

Параметры
Нет

Возвращает
Array — история сообщений.

---

addToHistory(message)

Добавляет сообщение в историю.

Параметры

Имя	Тип	Описание
message	Object	Объект {role, content}

---

disableHistory()

Отключает сохранение истории сообщений.

Параметры
Нет

---

clearHistory()

Очищает историю сообщений.

Параметры
Нет

---

getHistoryStr()

Получает историю сообщений в виде строки.

Параметры
Нет

Возвращает
String — история сообщений.

---

setModel(model)

Устанавливает модель для генерации ответа.

Параметры

Имя	Тип	Описание
model	String	Название модели

---

setProvider(providerName)

Устанавливает провайдера LLM.

Параметры

Имя	Тип	Описание
providerName	String	Название провайдера

---

getProvider()

Получает текущего провайдера.

Параметры
Нет

Возвращает
String — название провайдера.

---

setApiFormat(apiFormat)

Устанавливает формат API.

Параметры

Имя	Тип	Описание
apiFormat	String	Формат API

---

setModelParams(params)

Устанавливает параметры модели.

Параметры

Имя	Тип	Описание
params	Object	Объект с параметрами модели

---

setTemperature(temp)

Устанавливает температуру генерации.

Параметры

Имя	Тип	Описание
temp	Number	Температура генерации

---

getParams()

Получает параметры модели.

Параметры
Нет

Возвращает
Object — параметры модели.

---

getMessages()

Получает промпты запроса.

Параметры
Нет

Возвращает
Array — промпты запроса.

---

deleteMessages()

Удаляет промпты запроса.

Параметры
Нет

---

prepareRequest(callbackScript = null)

Сохраняет промпты и задаёт callback-скрипт.

Параметры

Имя	Тип	Описание
callbackScript	String	Код скрипта для обработки ответа (опц.)

---

sendRequest()

Отправляет асинхронный запрос к LLM.

Параметры
Нет

Пример

```
const llm = new LLMClient("DetectIntent")
llm.setModel("gpt-3.5-turbo")
llm.setProvider("OpenAI")
llm.setPromptTable("gpt_prompts", "MyAgent")
llm.addSystemPrompt("$start")
llm.addUserPrompt(`Запрос пользователя: ${lead.getAttr("user_input")}`)
llm.prepareRequest("MyAgent:MyScript")
return llm.sendRequest()
```

---

handleResponse()

Обрабатывает ответ от LLM, сохраняет результат и историю.

Параметры
Нет

---

setErrorScript(scriptCode)

Назначает fallback-скрипт при ошибке.

Параметры

Имя	Тип	Описание
scriptCode	String	Код скрипта для обработки ошибки

---

getErrorScript()

Получает fallback-скрипт.

Параметры
Нет

Возвращает
String — код fallback-скрипта.

---

setFallbackConfig(scriptCode, timeout)

Настраивает резервный сценарий и таймаут.

Параметры

Имя	Тип	Описание
scriptCode	String	Код резервного скрипта
timeout	Number	Таймаут в секундах

---

scheduleFallback()

Запланировать отложенный запуск fallback-скрипта.

Параметры
Нет

---

unscheduleFallback()

Удалить отложенный запуск fallback-скрипта.

Параметры
Нет

---

getResponseResult()

Получает результат запроса (объект ответа провайдера).

Параметры
Нет

Возвращает
Object — объект ответа провайдера.

---

getProviderResult()

Получает "сырые" данные от провайдера.

Параметры
Нет

Возвращает
Object — исходные данные от провайдера.

---

getResponseText()

Извлекает текст ответа из результата.

Параметры
Нет

Возвращает
String — текст ответа.

---

extractAnswerForModel(content)

Извлекает ответ по формату модели.

Параметры

Имя	Тип	Описание
content	Object	Объект ответа провайдера

---

getModelContextSize(modelName = null)

Получает максимальный размер контекста модели.

Параметры

Имя	Тип	Описание
modelName	String	Название модели или null

Возвращает
Number — максимальный размер контекста.

---

getLastJSON(text = null)

Извлекает JSON из текста ответа.

Параметры

Имя	Тип	Описание
text	String	Текст ответа или null

Возвращает
Object — найденный JSON.

---

extractTextWithoutJSON(text = null)

Удаляет JSON из текста и возвращает чистый текст.

Параметры

Имя	Тип	Описание
text	String	Текст ответа или null

Возвращает
String — текст без JSON.

---

removeMarkdownHeaders(text = null)

Удаляет заголовки Markdown из текста.

Параметры

Имя	Тип	Описание
text	String	Текст ответа или null

Возвращает
String — текст без заголовков.

---

getAgentAnswer(agentName)

Получает ответ агента по имени.

Параметры

Имя	Тип	Описание
agentName	String	Имя агента

Возвращает
String — ответ агента.

---

getLastTokenUsage()

Получает статистику токенов последнего запроса.

Параметры
Нет

Возвращает
Object — статистика токенов.

---

setStatus(status)

Устанавливает статус сессии.

Параметры

Имя	Тип	Описание
status	String	Статус: 'IDLE', 'PREPARED', ...

---

getStatus()

Получает текущий статус.

Параметры
Нет

Возвращает
String — текущий статус.

---

clearStatus()

Очищает статус.

Параметры
Нет

---

nextFlow(scriptCode)

Переходит к следующему flow-скрипту.

Параметры

Имя	Тип	Описание
scriptCode	String	Код следующего flow-скрипта

---

sendResponse()

Отправляет ответ в лог админа.

Параметры
Нет

---

sendFormattedResponse(format = 'Markdown')

Отправляет ответ в заданном формате.

Параметры

Имя	Тип	Описание
format	String	Формат: 'Markdown', 'HTML', ...

---

finishFlow()

Завершает работу сессии.

Параметры
Нет

Примеры использования

```
const LLMClient = require("Common.MetabotAI.LLMClient")
const llm = new LLMClient("DetectIntent")
llm.setModel("gpt-3.5-turbo")
llm.setProvider("OpenAI")
llm.setPromptTable("gpt_prompts", "MyAgent")
llm.addSystemPrompt("$start")
llm.addUserPrompt(`Запрос пользователя: ${lead.getAttr("user_input")}`)
llm.prepareRequest("MyAgent:MyScript")
llm.setErrorScript("MyAgent:ErrorFlow")
return llm.sendRequest()
```

Жизненный цикл и статусы

Статус	Когда устанавливается	Описание
`IDLE`	`new LLMClient(...)`	Клиент создан, ничего не сделано.
`PREPARED`	`prepareRequest()`	Промпты собраны, запрос готов.
`WAITING`	`sendRequest()`	Запрос отправлен, ожидаем ответ.
`SUCCESS`	`handleResponse()` (200)	Ответ успешно получен.
`ERROR`	`handleResponse()` (≠200)	Ошибка при получении ответа.
`ERROR_HANDLED`	`handleResponse()` → fallback	Сработал обработчик ошибки.
`DONE`	вручную	Не используется напрямую.

Отладка и переменные

Переменная	Пример значения	Описание
`llm_<session>_status`	`WAITING`	Текущий статус запроса
`llm_<session>_provider`	`OpenAI`	Название LLM-провайдера
`llm_<session>_params`	`{ model: "gpt-3.5-turbo", temperature: 0.7 }`	Параметры модели
`llm_<session>_messages`	`[{"role": "system", ...}, ...]`	Все промпты запроса
`llm_<session>_history`	`[{"role": "user", ...}, ...]`	История переписки
`llm_<session>_callback_script`	`MyAgent:NextStep`	Скрипт для ответа
`llm_<session>_error_script`	`MyAgent:Fallback`	Скрипт для ошибки
`llm_<session>_raw_response`	`Вот ваш ответ...`	Текст ответа до форматирования
`llm_<session>_user_query`	`Сколько лет космосу?`	Последний пользовательский запрос

FAQ

Вопрос: Как добавить свой промпт?
Ответ: Используйте `addSystemPrompt`, `addUserPrompt` или `addAssistantPrompt`. Можно ссылаться на промпты из таблицы через `$name` или `@common_name`.

Вопрос: Как обработать ошибку?
Ответ: Назначьте обработчик через `setErrorScript(scriptCode)` или используйте резервный сценарий через `setFallbackConfig(scriptCode, timeout)`.

Вопрос: Как получить статистику токенов?
Ответ: Вызовите `getLastTokenUsage()` после получения ответа.

Вопрос: Как интегрировать с другими компонентами?
Ответ: Используйте методы перехода (`nextFlow`), отправки (`sendResponse`, `sendFormattedResponse`) и работы с историей.

Версия #3
Павел Борисов создал 2025-09-05 14:00:37 UTC
Павел Борисов обновил 2025-11-12 14:03:47 UTC