LLMTracer - Плагин для трассировки запросов к LLM

Описание

LLMTracer - это модуль для трассировки запросов к языковым моделям (LLM) и внешним сервисам. Он позволяет записывать данные о запросах, ответах, ошибках и использовании токенов в таблицу `llm_tracer`.

Установка

Импортируйте модуль в ваш код:

```
const LLMTracer = require('Common.Platform.LLMTracer');
```

Основные возможности

- Создание сессий трассировки
- Запись информационных сообщений
- Запись ошибок
- Подсчет использованных токенов за указанный период

Быстрый старт

1. Создание сессии трассировки

```
// Создание сессии с параметрами по умолчанию
const session = LLMTracer.createSession();

// Создание сессии с указанием параметров
const session = LLMTracer.createSession({
  agent_name: "ChatGPT Assistant",
  provider: "OpenAI",
  model: "gpt-4"
});
```

Параметры:

- `options` (Object) - Параметры сессии
  - `business_id` (string|number) - ID бизнеса
  - `bot_id` (string|number) - ID бота
  - `lead_id` (string|number) - ID лида
  - `script_id` (string|number) - ID скрипта
  - `command_id` (string|number) - ID команды
  - `agent_name` (string) - Имя агента
  - `provider` (string) - Провайдер LLM
  - `model` (string) - Модель LLM

2. Прямая запись трассировки

```
// Прямая запись трассировки с указанием всех параметров
LLMTracer.recordTrace({
  outclient_name: "OpenAI API",
  step: 1,
  outclient_time_sec: 2.5,
  provider: "OpenAI",
  model: "gpt-4",
  type: "info",
  message: "Запрос выполнен",
  status: "success",
  input_tokens: 150,
  output_tokens: 50
});
```

Параметры:

- `traceData` (Object) - Данные для записи в таблицу
  - `outclient_name` (string) - Имя внешнего клиента
  - `step` (number|string) - Шаг выполнения
  - `outclient_time_sec` (number) - Время выполнения запроса в секундах
  - `provider` (string) - Провайдер LLM
  - `model` (string) - Модель LLM
  - `type` (string) - Тип записи (info, error)
  - `message` (string) - Сообщение
  - `record` (string) - Дополнительные данные в формате JSON
  - `status` (string|number) - Статус выполнения
  - `input_tokens` (number) - Количество входных токенов
  - `output_tokens` (number) - Количество выходных токенов

3. Запись информационных сообщений

```
// Запись простого сообщения
LLMTracer.info("Запрос к LLM выполнен успешно");

// Запись сообщения с дополнительными данными
LLMTracer.info("Запрос к LLM выполнен успешно", {
  outclient_name: "OpenAI API",
  input_tokens: 150,
  output_tokens: 50,
  outclient_time_sec: 2.5
});
```

Параметры:

- `message` (string) - Информационное сообщение
- `data` (Object) - Дополнительные данные для записи
  - `outclient_name` (string) - Имя внешнего клиента
  - `step` (number|string) - Шаг выполнения
  - `outclient_time_sec` (number) - Время выполнения запроса в секундах
  - `provider` (string) - Провайдер LLM
  - `model` (string) - Модель LLM
  - `status` (string|number) - Статус выполнения
  - `input_tokens` (number) - Количество входных токенов
  - `output_tokens` (number) - Количество выходных токенов

3. Запись ошибок

```
try {
  // Ваш код...
} catch (error) {
  LLMTracer.error("Ошибка при запросе к LLM", {
    error_message: error.message,
    outclient_name: "OpenAI API"
  });
}
```

4. Подсчет токенов

```
// Подсчет токенов за все время
const allTokens = LLMTracer.countTokens();
debug(`Всего использовано токенов: ${allTokens.total_tokens}`);

// Подсчет токенов за определенный период
const tokensForPeriod = LLMTracer.countTokens({
  from: "2023-01-01 00:00:00",
  to: "2023-01-31 23:59:59"
});

debug(`Входящие токены: ${tokensForPeriod.input_tokens}`);
debug(`Исходящие токены: ${tokensForPeriod.output_tokens}`);
debug(`Всего токенов: ${tokensForPeriod.total_tokens}`);
```

Параметры:

- `dateRange` (Object) - Объект с параметрами временного диапазона
  - `from` (string) - Начальная дата в формате "YYYY-MM-DD HH:MM:SS"
  - `to` (string) - Конечная дата в формате "YYYY-MM-DD HH:MM:SS"

Возвращает: Объект с количеством токенов

- `input_tokens` (number) - Количество входных токенов
- `output_tokens` (number) - Количество выходных токенов
- `total_tokens` (number) - Общее количество токенов

Структура таблицы llm_tracer

Модуль записывает данные в таблицу `llm_tracer` со следующими полями:

- `business_id` - ID бизнеса
- `lead_id` - ID лида
- `script_id` - ID скрипта
- `session_id` - ID сессии
- `agent_name` - Имя агента
- `outclient_name` - Имя внешнего клиента
- `step` - Шаг выполнения
- `outclient_time` - Время выполнения запроса в секундах
- `provider` - Провайдер LLM
- `model` - Модель LLM
- `type` - Тип записи (info, error)
- `message` - Сообщение
- `record` - Дополнительные данные в формате JSON
- `status` - Статус выполнения
- `input_tokens` - Количество входных токенов
- `output_tokens` - Количество выходных токенов
- `created_at` - Дата и время создания записи (автоматически)

Импортировать таблицу можно по ссылке:

https://stage.metabot.dev/business/export/1216/show?bots=&b_c=0&b_o=0&b_i=0&b_ls=0&b_r=0&b_s=0&b_li=0&b_t=0&b_br=0&b_aie=0&b_aee=0&b_sa=0&cts_llm_tracer=1&b_llm_tracer=1

Версия #2
Павел Борисов создал 2025-09-01 18:22:17 UTC
Павел Борисов обновил 2025-11-12 14:03:47 UTC