/\*\*

**\* Classe Assistant**

\*

\* Questa classe rappresenta le API di un assistente OpenAI.

\*/

class OpenAIAssistant {

/\*\*

\* La chiave API per l'accesso alle API di OpenAI.

\* @var string

\*/

private $api\_key;

/\*\*

\* L'ID dell'assistente.

\* @var string|null

\*/

private $assistant\_id;

/\*\*

\* L'URL di base per le chiamate API.

\* @var string

\*/

private $base\_url;

/\*\*

\* L'header di versione per le chiamate API.

\* @var string

\*/

private $version\_header;

/\*\*

\* Indica se sono presenti chiamate agli strumenti (tool) in corso.

\* @var bool

\*/

public $has\_tool\_calls = false;

/\*\*

\* L'ID della chiamata agli strumenti (tool) in corso.

\* @var string|null

\*/

public $tool\_call\_id = null;

/\*\*

\* Costruttore della classe OpenAIAssistant.

\*

\* @param string $api\_key La chiave API per l'accesso alle API di OpenAI.

\* @param string|null $assistant\_id L'ID dell'assistente (opzionale).

\* @param string $base\_url L'URL di base per le chiamate API (opzionale).

\* @param string $version\_header L'header di versione per le chiamate API (opzionale).

\*/

**public function \_\_construct(**

$api\_key,

$assistant\_id = null,

$base\_url = 'https://api.openai.com/v1',

$version\_header = 'OpenAI-Beta: assistants=v1'

)

{

$this->api\_key = $api\_key;

$this->assistant\_id = $assistant\_id;

$this->base\_url = $base\_url;

$this->version\_header = $version\_header;

}

/\*\*

\* Crea un nuovo assistente.

\*

\* @param string $name Il nome dell'assistente.

\* @param string $instructions Le istruzioni per l'assistente.

\* @param array $tools Gli strumenti (tools) da utilizzare.

\* @return string L'ID dell'assistente creato.

\* @throws \Exception Se non è possibile creare l'assistente.

\*/

public function create\_assistant($name, $instructions, $tools)

{

// Implementazione del metodo...

}

/\*\*

\* Modifica un assistente esistente.

\*

\* @param string $name Il nuovo nome dell'assistente.

\* @param string $instructions Le nuove istruzioni per l'assistente.

\* @param array $tools Gli strumenti (tools) aggiornati da utilizzare.

\* @return string L'ID dell'assistente modificato.

\* @throws \Exception Se non è possibile modificare l'assistente.

\*/

public function modify\_assistant($name, $instructions, $tools)

{

// Implementazione del metodo...

}

/\*\*

\* Ottiene la lista di tutti gli assistenti.

\*

\* @return array La lista di assistenti.

\*/

public function list\_assistants()

{

// Implementazione del metodo...

}

/\*\*

\* Crea un nuovo thread di conversazione.

\*

\* @param string $content Il contenuto del messaggio.

\* @param string $role Il ruolo del mittente (opzionale).

\* @return string L'ID del thread creato.

\* @throws \Exception Se non è possibile creare il thread.

\*/

public function create\_thread($content, $role = 'user')

{

// Implementazione del metodo...

}

/\*\*

\* Ottiene un thread di conversazione esistente.

\*

\* @param string $thread\_id L'ID del thread.

\* @return array Il thread di conversazione.

\* @throws \Exception Se non è possibile recuperare il thread.

\*/

**public function get\_thread($thread\_id)**

{

// Implementazione del metodo...

}

/\*\*

\* Aggiunge un messaggio a un thread di conversazione.

\*

\* @param string $thread\_id L'ID del thread.

\* @param string $content Il contenuto del messaggio.

\* @param string $role Il ruolo del mittente (opzionale).

\* @return string L'ID del messaggio aggiunto.

\* @throws \Exception Se non è possibile creare il messaggio.

\*/

public function add\_message($thread\_id, $content, $role = 'user')

{

// Implementazione del metodo...

}

/\*\*

\* Ottiene un messaggio all'interno di un thread di conversazione.

\*

\* @param string $thread\_id L'ID del thread.

\* @param string $message\_id L'ID del messaggio.

\* @return array Il messaggio.

\* @throws \Exception Se non è possibile recuperare il messaggio.

\*/

public function get\_message($thread\_id, $message\_id)

{

// Implementazione del metodo...

}

/\*\*

\* Ottiene la lista di tutti i messaggi in un thread di conversazione.

\*

\* @param string $thread\_id L'ID del thread.

\* @return array La lista di messaggi.

\*/

public function list\_thread\_messages($thread\_id)

{

// Implementazione del metodo...

}

/\*\*

\* Esegue un thread di conversazione.

\*

\* @param string $thread\_id L'ID del thread.

\* @return string|bool L'ID dell'esecuzione del thread se è necessaria un'azione, altrimenti false.

\*/

public function run\_thread($thread\_id)

{

// Implementazione del metodo...

}

/\*\*

\* Esegue gli strumenti richiesti da un thread di conversazione.

\*

\* @param string $thread\_id L'ID del thread.

\* @param string $execution\_id L'ID dell'esecuzione del thread.

\* @param object|null $optional\_object Un oggetto opzionale su cui chiamare i metodi degli strumenti.

\* @return array I risultati degli strumenti eseguiti.

\* @throws \Exception Se non è possibile eseguire gli strumenti.

\*/

public function execute\_tools($thread\_id, $execution\_id, $optional\_object = null)

{

// Implementazione del metodo...

}

/\*\*

\* Invia gli output degli strumenti al thread di conversazione.

\*

\* @param string $thread\_id L'ID del thread.

\* @param string $execution\_id L'ID dell'esecuzione del thread.

\* @param array $outputs Gli output degli strumenti.

\* @return string|bool L'ID dell'esecuzione del thread se è necessaria un'azione, altrimenti false.

\* @throws \Exception Se non è possibile inviare gli output degli strumenti.

\*/

public function submit\_tool\_outputs($thread\_id, $execution\_id, $outputs)

{

// Implementazione del metodo...

}

// Metodi privati

/\*\*

\* Esegue una richiesta GET alle API di OpenAI.

\*

\* @param string $route Il percorso della richiesta.

\* @return array La risposta della richiesta GET.

\*/

private function send\_get\_request($route)

{

// Implementazione del metodo...

}

/\*\*

\* Esegue una richiesta POST alle API di OpenAI.

\*

\* @param string $route Il percorso della richiesta.

\* @param mixed $payload Il payload della richiesta.

\* @return array La risposta della richiesta POST.

\*/

private function send\_post\_request($route, $payload = null)

{

// Implementazione del metodo...

}

/\*\*

\* Esegue una richiesta tramite cURL.

\*

\* @param resource $ch La risorsa cURL.

\* @return array La risposta della richiesta.

\* @throws \Exception Se la richiesta cURL fallisce.

\*/

private function execute\_request($ch)

{

// Implementazione del metodo...

}

// Altri metodi

/\*\*

\* Metodo aggiuntivo per scrivere un log delle chiamate agli strumenti.

\*

\* @param string $log\_entry L'entry del log.

\*/

private function write\_log($log\_entry)

{

// Implementazione del metodo...

}

}