/\*\*

\* =============================================================================

\* @file ai\_memory.gs

\* @module GAssistant.AI.Memory

\* @description

\* وحدة إدارة "ذاكرة" المساعد بنوعيها:

\* - قصيرة المدى (Session Context): تُحفظ في CacheService لجلسة المستخدم.

\* - طويلة المدى (Operations Log): يتم تسجيلها عبر GAssistant.Utils.log.

\* =============================================================================

\*/

var GAssistant = GAssistant || {};

GAssistant.AI = GAssistant.AI || {};

GAssistant.AI.Memory = (() => {

'use strict';

const { log, error } = GAssistant.Utils;

const Config = GAssistant.Config;

const CACHE = CacheService.getUserCache(); // استخدام ذاكرة المستخدم للحفاظ على الخصوصية

const CONTEXT\_KEY = 'g\_assistant\_session';

const CACHE\_DURATION = 3600; // 1 hour

/\*\*

\* يحمل سياق الجلسة الكامل من ذاكرة التخزين المؤقت.

\* @returns {object} - كائن السياق، أو كائن افتراضي إذا فشل التحميل.

\*/

function \_loadSessionContext() {

try {

const raw = CACHE.get(CONTEXT\_KEY);

if (raw) {

return JSON.parse(raw);

}

} catch (e) {

error('AI.Memory.\_loadSessionContext: Failed to parse context from cache', e);

}

// Return a default structure if not found or on error

return { history: [] };

}

/\*\*

\* يحفظ سياق الجلسة الكامل في ذاكرة التخزين المؤقت.

\* @param {object} context - كائن السياق المراد حفظه.

\*/

function \_saveSessionContext(context) {

try {

CACHE.put(CONTEXT\_KEY, JSON.stringify(context), CACHE\_DURATION);

} catch (e) {

error('AI.Memory.\_saveSessionContext: Failed to save context to cache', e);

}

}

/\*\*

\* يمسح سياق الجلسة بالكامل.

\*/

function clearSessionContext() {

CACHE.remove(CONTEXT\_KEY);

log('AI.Memory.clearSessionContext: Session context cleared.');

GAssistant.UI.Dialogue.createInfo('✅ تم مسح ذاكرة الجلسة بنجاح.');

}

/\*\*

\* يضيف رسالة إلى سجل المحادثة في سياق الجلسة.

\* @param {string} role - دور المرسل ('user' أو 'model').

\* @param {string|object} content - محتوى الرسالة.

\*/

function addMessageToHistory(role, content) {

const ctx = \_loadSessionContext();

// تحويل الكائن إلى نص إذا كان الرد هو كائن رسالة

const messageContent = (typeof content === 'object' && content.text) ? content.text :

(typeof content === 'object') ? JSON.stringify(content) : content;

ctx.history.push({ role: role, parts: [{ text: messageContent }] });

// الحفاظ على حجم السجل ضمن الحد الأقصى

const maxHistory = Config.get('MAX\_HISTORY\_LENGTH');

if (ctx.history.length > maxHistory) {

ctx.history = ctx.history.slice(-maxHistory);

}

\_saveSessionContext(ctx);

log('AI.Memory.addMessageToHistory: Message added', { role, preview: messageContent.substring(0, 50) });

}

/\*\*

\* تُرجع سجل المحادثة الحالي.

\* @returns {Array<object>} - مصفوفة من كائنات الرسائل.

\*/

function getSessionHistory() {

return \_loadSessionContext().history;

}

return {

clearSessionContext,

addMessageToHistory,

getSessionHistory

};

})();