/\*\*

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

\* @file ai\_memory.gs - v15.7 (محرك الذاكرة والسجل)

\* @module AI\_MEMORY

\* @description

\* إدارة "ذاكرة" المساعد بنوعيها:

\* - قصيرة المدى (Session Context): تُحفظ في DocumentProperties لتكون دائمة عبر الجلسات.

\* - طويلة المدى (Operations Log): تُحفظ في ورقة "سجل العمليات".

\* تعتمد على: ENV, HELPERS, AI\_CORE, AI\_TOOLS\_DIALOGUE

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

\*/

var AI\_MEMORY = (function(ENV, HELPERS, AI\_CORE, AI\_TOOLS\_DIALOGUE){ // ✅ حقن التبعيات الصحيحة: ENV, HELPERS, AI\_CORE, AI\_TOOLS\_DIALOGUE

'use strict';

const PROP\_SERVICE = PropertiesService.getDocumentProperties();

const CONTEXT\_KEY = 'AI\_SESSION\_CONTEXT';

// ✅ الوصول إلى SETTINGS مباشرة من ENV

const SETTINGS = ENV.SETTINGS; // تم تعريفها هنا للوصول السهل إليها داخل الوحدة

// ========== 1. إدارة الذاكرة قصيرة المدى (الجلسة) ==========

/\*\*

\* @function loadSessionContext

\* @description تحمل سياق الجلسة الكامل (بما في ذلك history و lastPrompt) من DocumentProperties.

\* @returns {object} - كائن السياق، أو كائن فارغ مع history فارغة إذا فشل التحميل أو لم يكن موجودًا.

\*/

function loadSessionContext() {

try {

const raw = PROP\_SERVICE.getProperty(CONTEXT\_KEY);

const context = raw ? JSON.parse(raw) : { history: [] };

HELPERS.logOperation('AI\_MEMORY', 'loadSessionContext:success', { contextKeys: Object.keys(context) });

return context;

} catch (e) {

HELPERS.logOperation('AI\_MEMORY', 'loadSessionContext:error', { error: e.message }); // ✅ استخدام HELPERS.logOperation

return { history: [] };

}

}

/\*\*

\* @function saveSessionContext

\* @description تحفظ كائن سياق الجلسة الكامل في DocumentProperties.

\* @param {object} ctx – كائن السياق المحدث.

\*/

function saveSessionContext(ctx) {

try {

PROP\_SERVICE.setProperty(CONTEXT\_KEY, JSON.stringify(ctx));

HELPERS.logOperation('AI\_MEMORY', 'saveSessionContext:success', { contextKeys: Object.keys(ctx) });

} catch (e) {

HELPERS.logOperation('AI\_MEMORY', 'saveSessionContext:error', { error: e.message }); // ✅ استخدام HELPERS.logOperation

}

}

/\*\*

\* @function clearSessionContext

\* @description تمسح ذاكرة الجلسة بالكامل (عبر حفظ كائن فارغ).

\* @returns {object} - كائن رسالة نجاح من AI\_TOOLS\_DIALOGUE.

\*/

function clearSessionContext() {

saveSessionContext({});

HELPERS.logOperation('AI\_MEMORY', 'clearSessionContext:executed', {}); // ✅ تسجيل العملية

return AI\_TOOLS\_DIALOGUE.showMessage("success", "✅ تم مسح ذاكرة الجلسة."); // ✅ استخدام AI\_TOOLS\_DIALOGUE

}

/\*\*

\* @function repeatLastPrompt

\* @description تعيد تنفيذ آخر طلب تم حفظه في الجلسة عن طريق استدعاء AI\_CORE.ask().

\* @returns {object} – نتيجة استدعاء AI\_CORE.ask() أو رسالة معلوماتية من AI\_TOOLS\_DIALOGUE إذا لم يكن هناك أمر سابق.

\*/

function repeatLastPrompt() {

HELPERS.logOperation('AI\_MEMORY', 'repeatLastPrompt:start', {}); // ✅ تسجيل بداية العملية

const ctx = loadSessionContext();

if (!ctx.lastPrompt) {

HELPERS.logOperation('AI\_MEMORY', 'repeatLastPrompt:no\_prompt', {}); // ✅ تسجيل عدم وجود أمر سابق

return AI\_TOOLS\_DIALOGUE.showMessage("warning", "⚠️ لا يوجد أمر سابق محفوظ في هذه الجلسة."); // ✅ استخدام AI\_TOOLS\_DIALOGUE

}

HELPERS.logOperation('AI\_MEMORY', 'repeatLastPrompt:calling\_ai\_core', { prompt: ctx.lastPrompt, model: ctx.lastModel }); // ✅ تسجيل الاستدعاء

return AI\_CORE.ask(ctx.lastPrompt, ctx.lastModel); // ✅ استخدام AI\_CORE

}

// ========== 2. إدارة الذاكرة طويلة المدى (سجل العمليات) ==========

// ❌ تم إزالة دالة logOperation من هنا.

// وفقًا لخارطة الطريق والمراجعات السابقة، HELPERS.logOperation هي الدالة الموحدة للتسجيل.

// يجب أن يتم استدعاء HELPERS.logOperation مباشرة من أي مكان يحتاج إلى تسجيل.

// --- تصدير الواجهة العامة للوحدة ---

return {

loadSessionContext,

saveSessionContext,

clearSessionContext,

repeatLastPrompt

// ❌ تم إزالة logOperation من التصدير

};

})(ENV, HELPERS, AI\_CORE, AI\_TOOLS\_DIALOGUE); // ✅ تمرير جميع التبعيات