/\*\*

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

\* @module GAssistant.Helpers – نسخة محدثة v17.0

\* @description

\* وحدة الأدوات المساعدة العامة تشمل:

\* - تنفيذ الأدوات بأمان (Failsafe Execution)

\* - تسجيل العمليات الموسّع

\* - إدارة أوراق Google وتنسيقاتها

\* - التعامل مع المعاملات، المفاتيح، والتنسيق

\* تعتمد على: GAssistant.Config, GAssistant.Tools.Catalog, GAssistant.AI.Dialogue, GAssistant.AI.Memory

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

\*/

var GAssistant = GAssistant || {};

// حقن التبعيات: يجب تمرير الوحدات المعتمدة هنا

GAssistant.Helpers = (function(GA\_Config, GA\_Tools\_Catalog, GA\_AI\_Dialogue, GA\_AI\_Memory) {

'use strict';

// ---------------------------------------------------------------------------

// دوال مساعدة داخلية (Private Helpers) - لا يمكن الوصول إليها مباشرة من الخارج

// ---------------------------------------------------------------------------

/\*\*

\* @function \_logOperation

\* @description

\* 🧠 تسجيل موسّع للعمليات.

\* يضيف زمن التنفيذ وسياق إضافي للتتبع.

\* @param {string} prompt - وصف موجز للعملية أو الطلب.

\* @param {string} action - الإجراء الذي تم تنفيذه (مثال: 'start\_analysis', 'execution\_error').

\* @param {object} summary - كائن يحتوي على تفاصيل إضافية للعملية.

\* @returns {void}

\*/

function \_logOperation(prompt, action, summary) {

try {

// GA\_AI\_Memory.logOperation هي الدالة المسؤولة عن التسجيل الفعلي

// يجب أن تتولى GA\_AI\_Memory إضافة الطابع الزمني وتفاصيل السياق الأخرى.

GA\_AI\_Memory.logOperation(prompt, action, summary);

} catch (e) {

console.error(`GAssistant.Helpers.\_logOperation failed: ${e.message}`);

}

}

/\*\*

\* @function \_toolExecutionFailsafe

\* @description

\* 🛡️ تنفيذ آمن لأية دالة في النظام.

\* يتحقق من وجود الأداة وصلاحيتها قبل التنفيذ، ويعالج الأخطاء.

\* @param {string} funcName - اسم الدالة المراد تنفيذها.

\* @param {Array<any>} args - مصفوفة المعاملات التي ستمرر للدالة.

\* @returns {object} - كائن نتيجة موحد { type: 'success'|'error', content: string|object }.

\*/

function \_toolExecutionFailsafe(funcName, args) {

try {

// التحقق من وجود الدالة في فهرس الأدوات المتاحة للتنفيذ

if (!GA\_Tools\_Catalog.AVAILABLE\_TOOLS.hasOwnProperty(funcName)) {

throw new Error(`الأداة "${funcName}" غير معرفة أو غير متاحة للتنفيذ المباشر.`);

}

// التحقق مما إذا كانت الدالة داخلية ويجب عدم تنفيذها مباشرة

// (يفترض أن الأدوات الداخلية تبدأ بـ '\_internal\_')

if (funcName.startsWith('\_internal\_')) {

throw new Error(`دالة "${funcName}" داخلية ولا يمكن تنفيذها مباشرة عبر Failsafe.`);

}

let result;

try {

// استدعاء الدالة الفعلية باستخدام apply لتمرير المعاملات كمصفوفة

result = GA\_Tools\_Catalog.AVAILABLE\_TOOLS[funcName].apply(null, args);

} catch (innerError) {

// التقاط الأخطاء الداخلية التي تحدث أثناء تنفيذ الدالة نفسها

throw new Error(`خطأ داخلي أثناء تنفيذ "${funcName}": ${innerError.message}`);

}

// تسجيل نجاح العملية

\_logOperation('toolExecutionFailsafe', 'execution\_success', { toolName: funcName, resultPreview: String(result).slice(0, 50) });

// التأكد من أن النتيجة هي كائن موحد، وإلا يتم تغليفها

return (typeof result === 'object' && result !== null && result.hasOwnProperty('type') && result.hasOwnProperty('content'))

? result

: { type: 'success', content: String(result) };

} catch (e) {

// تسجيل الخطأ

\_logOperation('toolExecutionFailsafe', 'execution\_error', { toolName: funcName, error: e.message, stack: e.stack });

// عرض رسالة الخطأ للمستخدم عبر واجهة المستخدم

GA\_AI\_Dialogue.showError(`حدث خطأ أثناء تنفيذ "${funcName}": ${e.message}`);

// إرجاع كائن خطأ موحد

return { type: 'error', content: `فشل تنفيذ الأداة: ${e.message}` };

}

}

/\*\*

\* @function \_normalizeFunctionArgs

\* @description

\* 🔄 تحويل المعاملات إلى مصفوفة، بغض النظر عن صيغتها الأصلية (كائن أو مصفوفة).

\* @param {object|Array<any>} args - المعاملات التي تم استلامها.

\* @returns {Array<any>} - مصفوفة من المعاملات.

\*/

function \_normalizeFunctionArgs(args) {

// إذا كانت المعاملات كائنًا، يتم استخراج القيم في مصفوفة.

// إذا كانت مصفوفة بالفعل، يتم إرجاعها كما هي.

// إذا كانت فارغة أو ليست كائنًا/مصفوفة، يتم إرجاع مصفوفة فارغة.

return (Array.isArray(args)) ? args : ((args && typeof args === 'object') ? Object.values(args) : []);

}

/\*\*

\* @function \_getSheet

\* @description

\* 📄 جلب ورقة Google Sheet بالاسم المحدد. إذا لم تكن موجودة، يتم إنشاؤها

\* مع إضافة رؤوس الأعمدة إذا تم توفيرها.

\* @param {string} name - اسم الورقة.

\* @param {Array<string>} [headers] - مصفوفة من رؤوس الأعمدة لإضافتها إذا تم إنشاء ورقة جديدة.

\* @returns {GoogleAppsScript.Spreadsheet.Sheet} - كائن الورقة.

\* @throws {Error} إذا كان اسم الورقة غير صالح.

\*/

function \_getSheet(name, headers) {

if (typeof name !== 'string' || !name.trim()) {

throw new Error('اسم الورقة غير صالح.');

}

const ss = SpreadsheetApp.getActiveSpreadsheet();

let sheet = ss.getSheetByName(name);

if (!sheet) {

sheet = ss.insertSheet(name);

if (Array.isArray(headers) && headers.length) {

sheet.appendRow(headers);

}

}

return sheet;

}

/\*\*

\* @function \_formatAsTable

\* @description

\* 🎨 تنسيق نطاق محدد في ورقة كجدول احترافي مع حدود وتلوين صفوف.

\* @param {string} rangeA1 - النطاق المراد تنسيقه بصيغة A1 (مثال: "A1:Z100").

\* @param {GoogleAppsScript.Spreadsheet.Sheet} sheet - كائن الورقة المراد تنسيقها.

\* @returns {void}

\*/

function \_formatAsTable(rangeA1, sheet) {

try {

const range = sheet.getRange(rangeA1);

range.setBorder(true, true, true, true, true, true, true, true); // إضافة جميع الحدود

const header = range.offset(0, 0, 1, range.getNumColumns());

const colors = GA\_Config.getColors(); // جلب الألوان من GAssistant.Config

header.setBackground(colors.header || '#dfe6fd') // استخدام لون الرأس من الإعدادات

.setFontWeight('bold');

const body = range.offset(1, 0, range.getNumRows() - 1, range.getNumColumns());

if (body.getNumRows() > 0) {

const band = body.applyRowBanding(SpreadsheetApp.BandingTheme.LIGHT\_GREY);

// استخدام لون التمييز من الإعدادات، أو لون افتراضي

band.setSecondRowColor(colors.highlight || '#fce8e6');

}

} catch (e) {

console.error(`GAssistant.Helpers.\_formatAsTable failed: ${e.message}`);

// لا يتم عرض خطأ للمستخدم هنا، لأن هذه دالة مساعدة داخلية للتنسيق.

// يجب أن يتم التعامل مع الأخطاء على مستوى أعلى إذا كانت تؤثر على وظيفة مرئية.

}

}

// ---------------------------------------------------------------------------

// الواجهة العامة للوحدة (Public Interface) - الدوال المصدرة

// ---------------------------------------------------------------------------

return {

// تصدير الدوال الداخلية بأسماء عامة

logOperation: \_logOperation,

toolExecutionFailsafe: \_toolExecutionFailsafe,

normalizeFunctionArgs: \_normalizeFunctionArgs,

getSheet: \_getSheet,

formatAsTable: \_formatAsTable

// لا يتم تصدير getGeminiApiKey لأن الوصول إليها يتم الآن عبر GAssistant.Config.getSecret

};

})(

GAssistant.Config, // تم استخدام GAssistant.Config بدلاً من ENV

GAssistant.Tools.Catalog, // تم استخدام GAssistant.Tools.Catalog بدلاً من TOOLS\_CATALOG

GAssistant.AI.Dialogue, // تم استخدام GAssistant.AI.Dialogue بدلاً من AI\_TOOLS\_DIALOGUE

GAssistant.AI.Memory // تم استخدام GAssistant.AI.Memory بدلاً من AI\_MEMORY

);