/\*\*

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

\* @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 - اسم العملية أو دالة التنفيذ.

\* @param {string} action - وصف مختصر للفعل المنفذ.

\* @param {any} summary - ملخص التنفيذ أو الكائن المرتبط.

\*/

function \_logOperation(prompt, action, summary) {

try {

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

} catch (e) {

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

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

}

}

/\*\*

\* @function \_toolExecutionFailsafe

\* @description

\* 🛡️ تنفيذ آمن لأي دالة عبر اسمها ومعاملاتها.

\* يعالج الأخطاء ويسجلها، ويعيد رسالة موحدة.

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

\* @param {object} args - كائن يحتوي على المعاملات كخصائص.

\* @returns {object} - كائن رسالة موحد (success/error).

\*/

function \_toolExecutionFailsafe(functionName, args) {

\_logOperation('GAssistant.Helpers.toolExecutionFailsafe', `Executing ${functionName}`, args);

try {

const toolMetadata = GA\_Tools\_Catalog.getToolMetadata(functionName);

if (!toolMetadata) {

throw new Error(`الأداة "${functionName}" غير مسجلة في الكتالوج.`);

}

// تحديد الوحدة التي تحتوي على الدالة

let targetModule;

switch (toolMetadata.category) {

case 'Sheets':

targetModule = TOOLS\_SHEETS;

break;

case 'Accounting':

targetModule = TOOLS\_ACCOUNTING;

break;

case 'General':

targetModule = TOOLS\_GENERAL;

break;

case 'Internal':

targetModule = TOOLS\_INTERNAL;

break;

case 'System':

targetModule = TOOLS\_SYSTEM;

break;

case 'Documentation':

targetModule = TOOLS\_DOC\_GEN; // افتراض وجود هذه الوحدة

break;

case 'Code Review':

targetModule = GAssistant.Tools.CodeReview; // استخدام الوحدة الجديدة

break;

case 'Workshop':

targetModule = WORKSHOP\_TOOLS; // استخدام وحدة ورشة العمل

break;

case 'Project Insights':

targetModule = GAssistant.Tools.ProjectInsights; // استخدام وحدة تحليلات المشروع

break;

default:

throw new Error(`تصنيف غير معروف للأداة: ${toolMetadata.category}`);

}

if (typeof targetModule[functionName] !== 'function') {

throw new Error(`الدالة "${functionName}" غير موجودة في الوحدة "${toolMetadata.category}".`);

}

// استدعاء الدالة باستخدام المعاملات

const result = targetModule[functionName](...Object.values(args));

\_logOperation('GAssistant.Helpers.toolExecutionFailsafe', `Completed ${functionName}`, result);

return result;

} catch (e) {

\_logOperation('GAssistant.Helpers.toolExecutionFailsafe', `Failed ${functionName}`, { error: e.message, stack: e.stack });

return GA\_AI\_Dialogue.showError(`🚨 فشل تنفيذ الأداة "${functionName}": ${e.message}`);

}

}

/\*\*

\* @function \_normalizeFunctionArgs

\* @description

\* يحول كائن المعاملات إلى مصفوفة بالترتيب الصحيح بناءً على تعريف الدالة.

\* (هذه الدالة قد لا تكون ضرورية إذا كانت دوال الأدوات تستقبل كائنًا واحدًا كمعامل).

\* @param {string} functionName - اسم الدالة.

\* @param {object} args - كائن المعاملات.

\* @returns {Array} - مصفوفة من المعاملات بالترتيب الصحيح.

\*/

function \_normalizeFunctionArgs(functionName, args) {

const toolMetadata = GA\_Tools\_Catalog.getToolMetadata(functionName);

if (!toolMetadata || !toolMetadata.parameters || !toolMetadata.parameters.properties) {

return Object.values(args); // إذا لم يكن هناك تعريف، أعد القيم كما هي

}

const orderedArgs = [];

for (const paramName in toolMetadata.parameters.properties) {

if (args.hasOwnProperty(paramName)) {

orderedArgs.push(args[paramName]);

} else {

// يمكن هنا التعامل مع المعاملات الاختيارية أو إعطاء قيمة افتراضية

orderedArgs.push(undefined);

}

}

return orderedArgs;

}

/\*\*

\* @function \_getSheet

\* @description

\* يجلب ورقة عمل بالاسم، أو ينشئها إذا لم تكن موجودة.

\* @param {string} name - اسم الورقة.

\* @param {string[]} [headers] - صف الرؤوس لإنشاء الورقة به إذا كانت جديدة.

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

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

\*/

function \_getSheet(name, headers) {

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

throw new Error('getSheet: اسم الورقة غير صالح أو فارغ.');

}

const ss = SpreadsheetApp.getActiveSpreadsheet();

let sh = ss.getSheetByName(name);

if (!sh) {

sh = ss.insertSheet(name);

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

sh.appendRow(headers);

}

}

return sh;

}

/\*\*

\* @function \_formatAsTable

\* @description

\* 🎨 تنسيق نطاق بصيغة جدول احترافي (رؤوس ملونة + صفوف متبادلة).

\* @param {string} rangeA1 – نطاق التنسيق بصيغة A1.

\* @param {GoogleAppsScript.Spreadsheet.Sheet} sheet – الورقة المعنية.

\*/

function \_formatAsTable(rangeA1, sheet) {

try {

const range = sheet.getRange(rangeA1);

range.setBorder(true, true, true, true, true, true);

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

const colors = GA\_Config.getSettings(); // جلب الإعدادات بما في ذلك الألوان

header.setBackground(colors.PRIMARY\_HEADER\_COLOR || '#dfe6fd')

.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,

GAssistant.AI.Dialogue,

GAssistant.AI.Memory

);