// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// --- START OF FILE: 02\_utils.gs ---

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

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

\* @file 02\_utils.gs

\* @module GAssistant.Utils

\* @description

\* وحدة الأدوات المساعدة العامة والموحدة مع نظام حقن التبعيات وفحص البيئة.

\* هذه الوحدة هي "صندوق الأدوات السويسري" للمشروع.

\*

\* التحسينات المضافة:

\* 1. \*\*قياس زمن التنفيذ:\*\* `executeSafely` الآن تقيس وتُسجل زمن تنفيذ كل دالة.

\* 2. \*\*تسجيل مختصر:\*\* يتم قص البيانات الطويلة قبل تسجيلها في الشيت لمنع الأخطاء.

\* 3. \*\*فحص صلاحيات الورقة:\*\* `getSheet` الآن تتحقق إذا كانت الورقة محمية أو مخفية.

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

\*/

var GAssistant = GAssistant || {};

GAssistant.Utils = (() => {

'use strict';

function \_getConfig() {

if (GAssistant.Config && typeof GAssistant.Config.getAll === 'function') {

return GAssistant.Config.getAll();

}

console.error("Utils: Config module is not available.");

return { DEBUG\_MODE: true, OPERATION\_LOG\_SHEET: "ErrorLog", MAX\_LOG\_ENTRIES: 500 };

}

function log(message, data) {

console.log(`[INFO] ${message}`, data ? JSON.stringify(data) : '');

if (\_getConfig().DEBUG\_MODE) \_logToSheet('INFO', message, data);

}

function warn(message, data) {

console.warn(`[WARN] ${message}`, data ? JSON.stringify(data) : '');

if (\_getConfig().DEBUG\_MODE) \_logToSheet('WARN', message, data);

}

function error(message, err) {

const errorDetails = (err instanceof Error) ? { message: err.message, stack: err.stack } : err;

console.error(`[ERROR] ${message}`, JSON.stringify(errorDetails));

\_logToSheet('ERROR', message, errorDetails);

}

function \_logToSheet(level, message, data) {

try {

const config = \_getConfig();

const sheet = getSheet(config.OPERATION\_LOG\_SHEET, ["Timestamp", "Level", "Message", "Data"]);

if (!sheet) return;

// ✅ \*\*تحسين مضاف\*\*: قص البيانات الطويلة لمنع تجاوز حدود الخلية

const compactData = data ? JSON.stringify(data).substring(0, 1000) : "";

sheet.appendRow([new Date(), level, message, compactData]);

const maxEntries = config.MAX\_LOG\_ENTRIES || 500;

if (sheet.getLastRow() > maxEntries) {

sheet.deleteRows(2, sheet.getLastRow() - maxEntries);

}

} catch (e) { console.error(`Failed to write to log sheet: ${e.message}`); }

}

function getSheet(name, headers) {

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

error('getSheet failed: Invalid sheet name provided.');

return null;

}

const ss = SpreadsheetApp.getActiveSpreadsheet();

let sheet = ss.getSheetByName(name);

// ✅ \*\*تحسين مضاف\*\*: فحص صلاحيات الورقة

if (sheet && (sheet.isSheetHidden() || sheet.isProtected())) {

warn(`getSheet: Cannot operate on sheet "${name}" because it is hidden or protected.`);

// لا نرجع خطأ، بل null، لكي تتمكن الدالة المستدعية من التعامل مع الحالة

return null;

}

if (!sheet) {

sheet = ss.insertSheet(name);

if (Array.isArray(headers) && headers.length > 0) {

const headerRange = sheet.getRange(1, 1, 1, headers.length);

const headerColor = GAssistant.Config?.get('PRIMARY\_HEADER\_COLOR') || "#4a5568";

headerRange.setFontWeight('bold').setBackground(headerColor).setFontColor("#ffffff");

}

}

return sheet;

}

function executeSafely(func, args = [], contextMessage = '') {

// ✅ \*\*تحسين مضاف\*\*: قياس زمن التنفيذ

const startTime = Date.now();

try {

const result = func(...args);

const duration = Date.now() - startTime;

log(`Execution Time for '${func.name || contextMessage}'`, { duration\_ms: duration });

return result;

} catch (e) {

const duration = Date.now() - startTime;

const errorMessage = `Execution failed for ${func.name || 'anonymous function'}. ${contextMessage}`.trim();

error(errorMessage, e);

// تسجيل بيانات الأداء حتى في حالة الفشل

log(`Execution Time for FAILED '${func.name || contextMessage}'`, { duration\_ms: duration });

const Dialogue = GAssistant.UI?.Dialogue;

return Dialogue ? Dialogue.createError(`حدث خطأ: ${e.message}`) : { type: 'error', text: e.message };

}

}

function getFunctionByPath(functionPath) {

return functionPath.split('.').reduce((acc, part) => acc && acc[part], GAssistant);

}

const Injector = {

get: (...moduleNames) => {

const resolved = {};

for (const name of moduleNames) {

const mod = getFunctionByPath(`GAssistant.${name}`);

if (!mod) {

const errorMessage = `CRITICAL DEPENDENCY ERROR: Module GAssistant.${name} is not loaded or does not exist.`;

error(errorMessage);

throw new Error(errorMessage);

}

const key = name.includes('.') ? name.substring(name.lastIndexOf('.') + 1) : name;

resolved[key] = mod;

}

return resolved;

}

};

return {

log, warn, error, getSheet, executeSafely, getFunctionByPath, Injector

};

})();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// --- END OF FILE: 02\_utils.gs ---

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*