/\*\*

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

\* @file tools\_sheets.gs – v15.16 (محرك إدارة جداول Google Sheets)

\* @module TOOLS\_SHEETS

\* @description

\* مكتبة شاملة لتلاعب جداول Google Sheets:

\* • CRUD أساسية (قراءة/كتابة/مسح/حذف/إدراج صفوف/معادلات)

\* • فرز وفلترة الأعمدة

\* • بحث واستبدال جماعي

\* • تحليل وتلوين القيم الشاذة

\* • استخراج روابط من أعمدة

\* • تظليل الخلايا الفارغة

\* • إرسال تنبيه بعد الحذف (بريد)

\* • إنشاء جداول جديدة من القوالب مع تأكيد المستخدم

\* • حماية نطاقات وأعمدة وفق الأدوار أو بشكل عام

\* • دمج، تكرار، ودمج خلايا/أوراق

\* • [جديد] أدوات مساعدة: ملخص الصفحة، تحويل رقم العمود إلى حرف، تنسيق جدول بصري

\* • [جديد] ضبط عرض الأعمدة تلقائيًا

\* • [جديد] إعادة تسمية الأوراق

\* • [جديد] حماية النطاقات

\* • [جديد] تلخيص إحصائيات الورقة

\* • [جديد] جلب وصف الورقة بصيغة JSON

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

\*/

var TOOLS\_SHEETS = (function(HELPERS, AI\_TOOLS\_DIALOGUE, SETTINGS\_PROVIDER, ENV) {

'use strict';

/\*\*

\* @function \_getSheet

\* @description دالة مساعدة داخلية لجلب كائن الورقة بالاسم.

\* @param {string} name - اسم الورقة.

\* @param {Array<string>} [headers] - رؤوس الأعمدة إذا كانت الورقة جديدة.

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

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

\*/

function \_getSheet(name, headers) {

return HELPERS.getSheet(name, headers); // استخدام دالة المساعدة العامة

}

/\*\*

\* @function writeToCell

\* @description تكتب قيمة إلى خلية محددة.

\* @param {string} sheetName - اسم الورقة.

\* @param {string} cellA1 - مرجع الخلية (مثال: "A1").

\* @param {any} value - القيمة المراد كتابتها.

\* @returns {object} - كائن رسالة نجاح.

\*/

function writeToCell(sheetName, cellA1, value) {

HELPERS.logOperation('TOOLS\_SHEETS', 'writeToCell:start', { sheetName, cellA1, value });

try {

const sheet = \_getSheet(sheetName);

sheet.getRange(cellA1).setValue(value);

HELPERS.logOperation('TOOLS\_SHEETS', 'writeToCell:success', { sheetName, cellA1 });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم كتابة القيمة في الخلية ${cellA1} في ورقة "${sheetName}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'writeToCell:error', { sheetName, cellA1, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في الكتابة', `❌ فشل الكتابة إلى الخلية ${cellA1}: ${e.message}`);

}

}

/\*\*

\* @function clearRange

\* @description تمسح محتويات نطاق محدد من الخلايا.

\* @param {string} sheetName - اسم الورقة.

\* @param {string} rangeA1 - النطاق المراد مسحه (مثال: "A1:B10").

\* @returns {object} - كائن رسالة نجاح.

\*/

function clearRange(sheetName, rangeA1) {

HELPERS.logOperation('TOOLS\_SHEETS', 'clearRange:start', { sheetName, rangeA1 });

try {

const sheet = \_getSheet(sheetName);

sheet.getRange(rangeA1).clearContent();

HELPERS.logOperation('TOOLS\_SHEETS', 'clearRange:success', { sheetName, rangeA1 });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم مسح النطاق ${rangeA1} في ورقة "${sheetName}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'clearRange:error', { sheetName, rangeA1, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في المسح', `❌ فشل مسح النطاق ${rangeA1}: ${e.message}`);

}

}

/\*\*

\* @function insertRowAt

\* @description تُدرج صفًا جديدًا في فهرس محدد.

\* @param {string} sheetName - اسم الورقة.

\* @param {number} rowIndex - فهرس الصف الذي سيتم إدراج الصف الجديد قبله (يبدأ من 1).

\* @returns {object} - كائن رسالة نجاح.

\*/

function insertRowAt(sheetName, rowIndex) {

HELPERS.logOperation('TOOLS\_SHEETS', 'insertRowAt:start', { sheetName, rowIndex });

try {

const sheet = \_getSheet(sheetName);

sheet.insertRows(rowIndex);

HELPERS.logOperation('TOOLS\_SHEETS', 'insertRowAt:success', { sheetName, rowIndex });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم إدراج صف جديد في الصف ${rowIndex} في ورقة "${sheetName}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'insertRowAt:error', { sheetName, rowIndex, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في الإدراج', `❌ فشل إدراج صف في الصف ${rowIndex}: ${e.message}`);

}

}

/\*\*

\* @function deleteRow

\* @description تحذف صفًا محددًا.

\* @param {string} sheetName - اسم الورقة.

\* @param {number} rowIndex - فهرس الصف المراد حذفه (يبدأ من 1).

\* @returns {object} - كائن رسالة نجاح أو خطأ.

\*/

function deleteRow(sheetName, rowIndex) {

HELPERS.logOperation('TOOLS\_SHEETS', 'deleteRow:start', { sheetName, rowIndex });

try {

const sheet = \_getSheet(sheetName);

if (rowIndex < 1 || rowIndex > sheet.getLastRow()) {

throw new Error(`فهرس الصف ${rowIndex} خارج النطاق الصالح (1-${sheet.getLastRow()}).`);

}

sheet.deleteRows(rowIndex);

HELPERS.logOperation('TOOLS\_SHEETS', 'deleteRow:success', { sheetName, rowIndex });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم حذف الصف ${rowIndex} من ورقة "${sheetName}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'deleteRow:error', { sheetName, rowIndex, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في الحذف', `❌ فشل حذف الصف ${rowIndex}: ${e.message}`);

}

}

/\*\*

\* @function filterByValue

\* @description تقوم بتصفية البيانات في عمود معين بناءً على قيمة محددة.

\* @param {string} sheetName - اسم الورقة.

\* @param {number} columnIndex - فهرس العمود المراد تصفيته (يبدأ من 1).

\* @param {any} value - القيمة المراد التصفية بها.

\* @returns {object} - كائن رسالة نجاح.

\*/

function filterByValue(sheetName, columnIndex, value) {

HELPERS.logOperation('TOOLS\_SHEETS', 'filterByValue:start', { sheetName, columnIndex, value });

try {

const sheet = \_getSheet(sheetName);

const range = sheet.getDataRange();

const filter = SpreadsheetApp.newFilter(range);

const criteria = SpreadsheetApp.newFilterCriteria()

.whenTextEqualTo(String(value))

.build();

filter.setColumnFilterCriteria(columnIndex, criteria);

HELPERS.logOperation('TOOLS\_SHEETS', 'filterByValue:success', { sheetName, columnIndex, value });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم تطبيق التصفية في ورقة "${sheetName}" على العمود ${columnIndex} بالقيمة "${value}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'filterByValue:error', { sheetName, columnIndex, value, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في التصفية', `❌ فشل تطبيق التصفية: ${e.message}`);

}

}

/\*\*

\* @function sortColumn

\* @description تقوم بفرز عمود محدد تصاعديًا أو تنازليًا.

\* @param {string} sheetName - اسم الورقة.

\* @param {number} columnIndex - فهرس العمود المراد فرزه (يبدأ من 1).

\* @param {boolean} [ascending=true] - true للفرز التصاعدي، false للفرز التنازلي.

\* @returns {object} - كائن رسالة نجاح.

\*/

function sortColumn(sheetName, columnIndex, ascending = true) {

HELPERS.logOperation('TOOLS\_SHEETS', 'sortColumn:start', { sheetName, columnIndex, ascending });

try {

const sheet = \_getSheet(sheetName);

const range = sheet.getDataRange();

range.sort({ column: columnIndex, ascending: ascending });

HELPERS.logOperation('TOOLS\_SHEETS', 'sortColumn:success', { sheetName, columnIndex, ascending });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم فرز العمود ${columnIndex} في ورقة "${sheetName}" ${ascending ? 'تصاعديًا' : 'تنازليًا'}.`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'sortColumn:error', { sheetName, columnIndex, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في الفرز', `❌ فشل فرز العمود ${columnIndex}: ${e.message}`);

}

}

/\*\*

\* @function extractUrls

\* @description تستخرج جميع الروابط (URLs) من عمود محدد.

\* @param {string} sheetName - اسم الورقة.

\* @param {number} columnIndex - فهرس العمود المراد استخراج الروابط منه (يبدأ من 1).

\* @returns {object} - كائن رسالة نجاح مع قائمة الروابط أو خطأ.

\*/

function extractUrls(sheetName, columnIndex) {

HELPERS.logOperation('TOOLS\_SHEETS', 'extractUrls:start', { sheetName, columnIndex });

try {

const sheet = \_getSheet(sheetName);

const data = sheet.getDataRange().getValues();

const urls = [];

const urlRegex = /(https?:\/\/[^\s]+)/g; // regex بسيط للروابط

for (let i = 0; i < data.length; i++) {

const cellValue = String(data[i][columnIndex - 1]); // -1 لأن الفهارس تبدأ من 0

let match;

while ((match = urlRegex.exec(cellValue)) !== null) {

urls.push(match[0]);

}

}

HELPERS.logOperation('TOOLS\_SHEETS', 'extractUrls:success', { sheetName, columnIndex, urlCount: urls.length });

if (urls.length > 0) {

return AI\_TOOLS\_DIALOGUE.showSummaryTable(urls.map(url => ({ URL: url })), `الروابط المستخرجة من العمود ${columnIndex}`);

} else {

return AI\_TOOLS\_DIALOGUE.showInfo('لا توجد روابط', `ℹ️ لم يتم العثور على أي روابط في العمود ${columnIndex} في ورقة "${sheetName}".`);

}

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'extractUrls:error', { sheetName, columnIndex, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في استخراج الروابط', `❌ فشل استخراج الروابط من العمود ${columnIndex}: ${e.message}`);

}

}

/\*\*

\* @function highlightEmptyCells

\* @description تظلل الخلايا الفارغة في نطاق محدد.

\* @param {string} sheetName - اسم الورقة.

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

\* @param {string} [color='#FF0000'] - لون التظليل (افتراضي: أحمر).

\* @returns {object} - كائن رسالة نجاح.

\*/

function highlightEmptyCells(sheetName, rangeA1, color = '#FF0000') {

HELPERS.logOperation('TOOLS\_SHEETS', 'highlightEmptyCells:start', { sheetName, rangeA1, color });

try {

const sheet = \_getSheet(sheetName);

const range = sheet.getRange(rangeA1);

const values = range.getValues();

let highlightedCount = 0;

for (let i = 0; i < values.length; i++) {

for (let j = 0; j < values[i].length; j++) {

if (values[i][j] === null || String(values[i][j]).trim() === '') {

sheet.getRange(range.getRow() + i, range.getColumn() + j).setBackground(color);

highlightedCount++;

}

}

}

HELPERS.logOperation('TOOLS\_SHEETS', 'highlightEmptyCells:success', { sheetName, rangeA1, highlightedCount });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم تظليل ${highlightedCount} خلية فارغة في النطاق ${rangeA1} في ورقة "${sheetName}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'highlightEmptyCells:error', { sheetName, rangeA1, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في التظليل', `❌ فشل تظليل الخلايا الفارغة: ${e.message}`);

}

}

/\*\*

\* @function autoSizeColumns

\* @description تضبط عرض الأعمدة تلقائيًا في نطاق محدد.

\* @param {string} sheetName - اسم الورقة.

\* @param {string} [rangeA1] - النطاق الذي سيتم ضبط أعمدته (مثال: "A:Z"). إذا لم يحدد، يتم ضبط جميع الأعمدة المستخدمة.

\* @returns {object} - كائن رسالة نجاح.

\*/

function autoSizeColumns(sheetName, rangeA1) {

HELPERS.logOperation('TOOLS\_SHEETS', 'autoSizeColumns:start', { sheetName, rangeA1 });

try {

const sheet = \_getSheet(sheetName);

if (rangeA1) {

const range = sheet.getRange(rangeA1);

sheet.autoResizeColumns(range.getColumn(), range.getNumColumns());

} else {

sheet.autoResizeColumns(1, sheet.getLastColumn());

}

HELPERS.logOperation('TOOLS\_SHEETS', 'autoSizeColumns:success', { sheetName, rangeA1 });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم ضبط عرض الأعمدة تلقائيًا في ورقة "${sheetName}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'autoSizeColumns:error', { sheetName, rangeA1, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في ضبط العرض', `❌ فشل ضبط عرض الأعمدة: ${e.message}`);

}

}

/\*\*

\* @function renameSheet

\* @description تعيد تسمية ورقة عمل.

\* @param {string} oldName - الاسم الحالي للورقة.

\* @param {string} newName - الاسم الجديد للورقة.

\* @returns {object} - كائن رسالة نجاح.

\*/

function renameSheet(oldName, newName) {

HELPERS.logOperation('TOOLS\_SHEETS', 'renameSheet:start', { oldName, newName });

try {

const sheet = \_getSheet(oldName);

sheet.setName(newName);

HELPERS.logOperation('TOOLS\_SHEETS', 'renameSheet:success', { oldName, newName });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم إعادة تسمية الورقة من "${oldName}" إلى "${newName}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'renameSheet:error', { oldName, newName, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في إعادة التسمية', `❌ فشل إعادة تسمية الورقة: ${e.message}`);

}

}

/\*\*

\* @function protectRange

\* @description تحمي نطاقًا محددًا من الخلايا، مع إمكانية تحديد المحررين.

\* @param {string} sheetName - اسم الورقة.

\* @param {string} rangeA1 - النطاق المراد حمايته (مثال: "A1:B10").

\* @param {Array<string>} [editorsEmails] - مصفوفة من عناوين البريد الإلكتروني للمستخدمين المسموح لهم بالتحرير.

\* @returns {object} - كائن رسالة نجاح.

\*/

function protectRange(sheetName, rangeA1, editorsEmails) {

HELPERS.logOperation('TOOLS\_SHEETS', 'protectRange:start', { sheetName, rangeA1, editorsEmails });

try {

const sheet = \_getSheet(sheetName);

const protection = sheet.getRange(rangeA1).protect();

// إزالة جميع المحررين الحاليين ثم إضافة الجدد

protection.removeEditors(protection.getEditors());

if (editorsEmails && editorsEmails.length > 0) {

protection.addEditors(editorsEmails);

} else {

// إذا لم يتم تحديد محررين، يتم حماية النطاق بحيث لا يمكن لأحد التعديل عليه سوى المالك

protection.setDomainEdit(false);

}

HELPERS.logOperation('TOOLS\_SHEETS', 'protectRange:success', { sheetName, rangeA1, editorsEmails });

return AI\_TOOLS\_DIALOGUE.showMessage('نجاح', `✅ تم حماية النطاق ${rangeA1} في ورقة "${sheetName}".`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'protectRange:error', { sheetName, rangeA1, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في الحماية', `❌ فشل حماية النطاق: ${e.message}`);

}

}

/\*\*

\* @function summarizeSheetStats

\* @description تلخص الإحصائيات الأساسية لورقة العمل (عدد الصفوف، الأعمدة، الخلايا الممتلئة).

\* @param {string} sheetName - اسم الورقة.

\* @returns {object} - كائن رسالة نجاح مع ملخص الإحصائيات.

\*/

function summarizeSheetStats(sheetName) {

HELPERS.logOperation('TOOLS\_SHEETS', 'summarizeSheetStats:start', { sheetName });

try {

const sheet = \_getSheet(sheetName);

const lastRow = sheet.getLastRow();

const lastColumn = sheet.getLastColumn();

const dataRange = sheet.getDataRange();

const numCells = dataRange.getNumRows() \* dataRange.getNumColumns();

const nonBlankCells = dataRange.getValues().flat().filter(String).length;

const summary = {

sheetName: sheet.getName(),

lastRow: lastRow,

lastColumn: lastColumn,

totalCells: numCells,

nonBlankCells: nonBlankCells,

blankCells: numCells - nonBlankCells

};

HELPERS.logOperation('TOOLS\_SHEETS', 'summarizeSheetStats:success', summary);

return AI\_TOOLS\_DIALOGUE.showSummaryTable(summary, `إحصائيات ورقة "${sheetName}"`);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'summarizeSheetStats:error', { sheetName, error: e.message });

return AI\_TOOLS\_DIALOGUE.showError('خطأ في الإحصائيات', `❌ فشل تلخيص إحصائيات الورقة: ${e.message}`);

}

}

/\*\*

\* @function generateSheetAnalysisPrompt

\* @description

\* 📝 [Markdown] توليد وصف مختصر للورقة الحالية بصيغة Markdown.

\* هذا هو التجسيد الجديد لدالة getSheetInsightForPrompt.

\* @param {string} [sheetName] - اسم الورقة (اختياري).

\* @returns {string} - ملخص نصي منسق بصيغة Markdown.

\*/

function generateSheetAnalysisPrompt(sheetName) {

HELPERS.logOperation('TOOLS\_SHEETS', 'generateSheetAnalysisPrompt:start', { sheetName });

try {

const ss = SpreadsheetApp.getActiveSpreadsheet();

const sheet = sheetName ? ss.getSheetByName(sheetName) : ss.getActiveSheet();

if (!sheet) {

throw new Error(`لم يتم العثور على الورقة "${sheetName}".`);

}

const name = sheet.getName();

const data = sheet.getDataRange().getValues();

if (data.length < 1) return `📄 الورقة "${name}" فارغة.`;

const headers = data[0];

const rowCount = data.length - 1;

const colCount = headers.length;

let summary = `### 🧾 وصف الورقة "${name}"\n`;

summary += `- 📌 عدد الأعمدة: ${colCount}\n`;

summary += `- 📊 عدد الصفوف (بدون الرؤوس): ${rowCount}\n`;

summary += `- 🏷️ الأعمدة:\n`;

headers.forEach((h, i) => {

summary += ` - ${HELPERS.columnIndexToLetter(i + 1)}: ${h}\n`;

});

HELPERS.logOperation('TOOLS\_SHEETS', 'generateSheetAnalysisPrompt:success', { sheetName, summaryLength: summary.length });

return summary;

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'generateSheetAnalysisPrompt:error', { sheetName, error: e.message });

return `❌ فشل في توليد وصف Markdown: ${e.message}`;

}

}

/\*\*

\* @function getSheetInsightAsJson

\* @description

\* ✅ دالة جديدة: تُرجع وصفًا منظمًا للورقة الحالية بصيغة JSON.

\* مفيدة لتمرير بنية البيانات إلى أدوات أخرى أو للتحليل البرمجي.

\* @param {string} [sheetName] - اسم الورقة (اختياري).

\* @returns {string} - سلسلة JSON تحتوي على بنية الورقة وعينة بيانات.

\*/

function getSheetInsightAsJson(sheetName) {

HELPERS.logOperation('TOOLS\_SHEETS', 'getSheetInsightAsJson:start', { sheetName });

try {

const ss = SpreadsheetApp.getActiveSpreadsheet();

const sheet = sheetName ? ss.getSheetByName(sheetName) : ss.getActiveSheet();

if (!sheet) {

throw new Error(`لم يتم العثور على الورقة "${sheetName}".`);

}

const name = sheet.getName();

const data = sheet.getDataRange().getValues();

if (data.length < 1) {

HELPERS.logOperation('TOOLS\_SHEETS', 'getSheetInsightAsJson:empty\_sheet', { sheetName });

return JSON.stringify({ sheetName: name, message: "الورقة فارغة" }, null, 2);

}

const headers = data[0];

// استخدام إعداد من ENV لتحديد عدد صفوف العينة، أو قيمة افتراضية

const SAMPLE\_DATA\_ROW\_COUNT = ENV.getSetting('SAMPLE\_DATA\_ROW\_COUNT') || 5;

const sampleRows = data.slice(1, SAMPLE\_DATA\_ROW\_COUNT + 1); // نأخذ أول N صفوف فقط (N+1 بسبب الرؤوس)

const structure = {

sheetName: name,

columns: headers.map((h, i) => ({

name: h,

index: i + 1,

letter: HELPERS.columnIndexToLetter(i + 1),

// يمكن إضافة تحليل لنوع البيانات هنا إذا لزم الأمر

})),

sampleData: sampleRows

};

HELPERS.logOperation('TOOLS\_SHEETS', 'getSheetInsightAsJson:success', { sheetName, dataLength: JSON.stringify(structure).length });

return JSON.stringify(structure, null, 2);

} catch (e) {

HELPERS.logOperation('TOOLS\_SHEETS', 'getSheetInsightAsJson:error', { sheetName, error: e.message });

return `❌ فشل في توليد وصف JSON للورقة: ${e.message}`;

}

}

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

return {

writeToCell,

clearRange,

insertRowAt,

deleteRow,

filterByValue,

sortColumn,

extractUrls,

highlightEmptyCells,

autoSizeColumns,

renameSheet,

protectRange,

summarizeSheetStats,

generateSheetAnalysisPrompt, // تصدير الدالة المحدثة

getSheetInsightAsJson // ✅ تصدير الدالة الجديدة

};

})(GAssistant.Helpers, GAssistant.AI.Dialogue, GAssistant.Config, GAssistant.Config); // ✅ تم تحديث التبعيات