/\*\*

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

\* @file workshop.gs – v12.9

\* @module WORKSHOP\_TOOLS

\* @description

\* أدوات ذكية لورشة عمل المبرمج. تقوم بتحليل الأكواد المقترحة

\* وتقترح الملف الأنسب لوضعها فيه، مما يسهل عملية تطوير وصيانة المشروع.

\* - تستخدم خوارزمية أوزان لتحديد أولوية اقتراح الملف.

\* - تدعم أنماطًا قابلة للتكوين ديناميكيًا من الإعدادات.

\* - تسجل عمليات الاقتراح للمراجعة.

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

\*/

var WORKSHOP\_TOOLS = (function(ENV, DIALOGUE\_TOOLS, SETTINGS\_PROVIDER, HELPERS) {

'use strict';

/\*\*

\* 🧠 [مساعدة] يحلل محتوى الكود ويقترح اسم الملف الأنسب لوضعه فيه.

\* يستخدم نظام أوزان لتحديد الأولوية.

\* @private

\* @param {string} functionCode - الكود البرمجي للدالة المقترحة.

\* @returns {string} - اسم الملف المقترح (مثال: 'tools\_sheets.gs').

\*/

function suggestTargetFile(functionCode) {

HELPERS.logOperation('WORKSHOP\_TOOLS', 'suggestTargetFile:start', { codeLength: functionCode.length });

const code = (functionCode || '').toLowerCase();

const cfg = SETTINGS\_PROVIDER.getSettings();

// قائمة الأنماط والكلمات المفتاحية المرتبطة بكل ملف، مع أوزان.

// هذه القائمة يتم جلبها الآن ديناميكيًا من SETTINGS\_PROVIDER.

const patterns = cfg.TOOL\_FILE\_SUGGESTION\_PATTERNS || [];

let bestMatch = { file: 'tools\_general.gs', score: 0 }; // افتراضي

const scores = {};

patterns.forEach(pattern => {

scores[pattern.file] = scores[pattern.file] || 0;

pattern.keywords.forEach(keyword => {

if (code.includes(keyword.toLowerCase())) {

scores[pattern.file] += pattern.weight;

}

});

});

for (const file in scores) {

if (scores[file] > bestMatch.score) {

bestMatch = { file: file, score: scores[file] };

}

}

HELPERS.logOperation('WORKSHOP\_TOOLS', 'suggestTargetFile:end', { suggestedFile: bestMatch.file, score: bestMatch.score });

return bestMatch.file;

}

/\*\*

\* @function proposeAndStageCode

\* @description

\* يحلل الكود المقترح ويقترح ملفًا مناسبًا له، ثم يجهز الكود في ورقة عمل للمراجعة.

\* @param {string} functionCode - الكود البرمجي للدالة المقترحة.

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

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

\*/

function proposeAndStageCode(functionCode, functionName) {

HELPERS.logOperation('WORKSHOP\_TOOLS', 'proposeAndStageCode:start', { functionName, codeLength: functionCode.length });

try {

if (!functionCode || !functionName) {

throw new Error("يجب توفير كود الدالة واسمها.");

}

const suggestedFile = suggestTargetFile(functionCode);

const stagingSheetName = ENV.getSetting('STAGING\_SHEET\_NAME') || 'Developer Workshop';

const ss = SpreadsheetApp.getActiveSpreadsheet();

let sheet = ss.getSheetByName(stagingSheetName);

if (!sheet) {

sheet = ss.insertSheet(stagingSheetName);

sheet.appendRow(['نوع العنصر', 'القيمة / الوصف']);

HELPERS.formatAsTable(sheet.getDataRange().getA1Notation(), sheet); // تنسيق كجدول

}

// مسح المحتوى القديم في ورقة التجهيز (اختياري، يمكن تعديله للإضافة بدلاً من المسح)

if (sheet.getLastRow() > 1) {

sheet.getRange(2, 1, sheet.getLastRow() - 1, sheet.getLastColumn()).clearContent();

}

const startRow = sheet.getLastRow() + 1; // الصف الذي ستبدأ منه الإضافة

sheet.getRange(startRow, 1).setValue("🚀 اقتراح كود جديد").setFontWeight('bold');

sheet.getRange(startRow + 1, 1).setValue("اسم الدالة المقترحة:").setFontWeight('bold');

sheet.getRange(startRow + 1, 2).setValue(functionName);

sheet.getRange(startRow + 2, 1).setValue("الملف المقترح:").setFontWeight('bold');

sheet.getRange(startRow + 2, 2).setValue(suggestedFile);

sheet.getRange(startRow + 3, 1).setValue("الكود المقترح:").setFontWeight('bold');

sheet.getRange(startRow + 3, 2).setValue(functionCode).setWrap(true);

const steps = [

`1. راجع الكود المقترح في الخلية B${startRow + 3}.`,

`2. انسخ الكود والصقه في ملف ${suggestedFile}.gs في محرر Apps Script.`,

`3. اختبر الدالة الجديدة للتأكد من أنها تعمل بشكل صحيح.`,

`4. قم بحذف هذا الاقتراح من ورقة "${stagingSheetName}" بعد الانتهاء.`

].join('\n');

sheet.getRange(startRow + 5, 1).setValue("✅ خطوات التفعيل:").setFontWeight('bold');

sheet.getRange(startRow + 5, 2).setValue(steps).setWrap(true).setVerticalAlignment("top");

sheet.getRange(startRow, 1, 6, 2).setBorder(true, true, true, true, true, true, '#cccccc', SpreadsheetApp.BorderStyle.SOLID);

sheet.autoResizeRows(startRow, 6); // ضبط ارتفاع الصفوف تلقائيا

ss.setActiveSheet(sheet);

HELPERS.logOperation('WORKSHOP\_TOOLS', 'proposeAndStageCode:success', { functionName, suggestedFile });

return DIALOGUE\_TOOLS.showMessage("success", `تم تجهيز ورشة العمل. يرجى مراجعة ورقة "${stagingSheetName}".`);

} catch (e) {

HELPERS.logOperation('WORKSHOP\_TOOLS', 'proposeAndStageCode:error', { error: e.message, functionName });

return DIALOGUE\_TOOLS.showError(`فشل في تجهيز ورشة العمل: ${e.message}`);

}

}

/\*\*

\* 🔌 واجهة مساعدة للشريط الجانبي.

\* تُستخدم لاستدعاء suggestTargetFile من واجهة المستخدم (HTML).

\* @param {string} functionCode - كود الدالة المقترحة من المستخدم.

\* @returns {{suggestedFile: string}} - كائن يحتوي على اسم الملف المقترح.

\*/

function suggestFileFromSidebar(functionCode) {

HELPERS.logOperation('WORKSHOP\_TOOLS', 'suggestFileFromSidebar:start', { codeLength: functionCode.length });

try {

const file = suggestTargetFile(functionCode);

HELPERS.logOperation('WORKSHOP\_TOOLS', 'suggestFileFromSidebar:success', { suggestedFile: file });

return { suggestedFile: file };

} catch (e) {

HELPERS.logOperation('WORKSHOP\_TOOLS', 'suggestFileFromSidebar:error', { error: e.message });

DIALOGUE\_TOOLS.showError(`فشل في اقتراح الملف: ${e.message}`);

return { suggestedFile: 'N/A', error: e.message };

}

}

return {

proposeAndStageCode,

suggestFileFromSidebar // تصدير الدالة للاستخدام من الشريط الجانبي

};

})(GAssistant.Config, GAssistant.AI.Dialogue, GAssistant.Config, GAssistant.Helpers); // تم تغيير ENV و SETTINGS\_PROVIDER إلى GAssistant.Config