/\*\*

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

\* @file ai\_dispatcher.gs – v15.9 (محرك التوجيه الذكي)

\* @module GAssistant.AI.Dispatcher

\* @description

\* مسؤول عن تحليل النية وتوجيه الطلبات. يعمل كشرطي مرور:

\* - preProcessIntent: لاكتشاف أوامر فورية قابلة للتنفيذ فورًا.

\* - postProcessResult: لتحليل رد النموذج وتنفيذ الأدوات أو إرجاع الرد.

\* الاعتمادات: GAssistant.Tools.Catalog, GAssistant.AI.Dialogue, GAssistant.Helpers

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

\*/

var GAssistant = GAssistant || {};

GAssistant.AI = GAssistant.AI || {}; // التأكد من وجود مساحة الاسم AI

// حقن التبعيات: يجب تمرير الوحدات المعتمدة هنا بأسماء مساحات الأسماء الكاملة

GAssistant.AI.Dispatcher = (function(GA\_Tools\_Catalog, GA\_AI\_Dialogue, GA\_Helpers) {

'use strict';

/\*\*

\* @function preProcessIntent

\* @description

\* يحلل النية من نص المستخدم مباشرة قبل إرسالها للنموذج.

\* يتعرف على أوامر فورية قابلة للتنفيذ فورًا (مثل حذف صف، مسح نطاق).

\* @param {object} state - كائن الحالة الحالي للـ pipeline.

\* @param {boolean} [simulateOnly=false] – إذا true، لا ينفذ بل يختبر فقط.

\* @returns {object} – state المحدّث (مع halt=true إن تم التعامل مع الأمر).

\*/

function preProcessIntent(state, simulateOnly = false) {

const prompt = state.prompt.trim().toLowerCase();

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'preProcessIntent:start', { prompt, simulateOnly });

// أمر: حذف صف

const deleteMatch = prompt.match(/(?:احذف|امسح)\s+(?:ال)?صف\s+(\d+)/);

if (deleteMatch) {

state.halt = true; // نوقف الـ pipeline لأننا سنتعامل مع الأمر هنا

if (!simulateOnly) {

const row = parseInt(deleteMatch[1], 10);

state.final = GA\_AI\_Dialogue.askUserConfirmation(

`🗑️ هل أنت متأكد من حذف الصف ${row}؟`,

'confirm\_delete\_row', // نوع تأكيد مخصص

{ rowToDelete: row } // حمولة بيانات إضافية

);

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'preProcessIntent:delete\_row\_confirmation', { row });

} else {

// في وضع المحاكاة، نجهز استجابة افتراضية

state.final = GA\_AI\_Dialogue.showMessage('info', `✅ سيتم طلب تأكيد حذف الصف ${parseInt(deleteMatch[1], 10)}.`);

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'preProcessIntent:delete\_row\_simulated', { row: parseInt(deleteMatch[1], 10) });

}

return state; // نرجع الحالة بعد التعامل مع الأمر

}

// يمكن إضافة المزيد من الأوامر المباشرة هنا (مثلاً: "مسح النطاق A1:B10")

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'preProcessIntent:end', { halt: state.halt });

return state;

}

/\*\*

\* @function postProcessResult

\* @description

\* يُحلل رد النموذج (Gemini)، وإذا تضمن FunctionCall، ينفذ الأدوات.

\* خلاف ذلك يعيد الرد كنصي فقط أو يُرجع خطأ موحد.

\* يدعم الآن الاستجابات المهيكلة (structuredOutput).

\* @param {object} state - كائن الحالة الحالي للـ pipeline.

\* @returns {object} – الحالة المحدثة (مع halt=true بعد المعالجة).

\*/

function postProcessResult(state) {

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:start', {});

if (!state.geminiResponse) {

state.halt = true;

state.final = GA\_AI\_Dialogue.showError('❌ لا يوجد رد من نموذج Gemini بعد الاتصال.');

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:end', { halt: true, finalType: state.final.type, reason: 'no\_gemini\_response' });

return state;

}

const gemini = state.geminiResponse;

const part = gemini.payload?.candidates?.[0]?.content?.parts?.[0];

if (!part) {

state.halt = true;

state.final = GA\_AI\_Dialogue.showError('❌ لم يتم تلقي رد صالح أو أجزاء محتوى من النموذج.');

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:end', { halt: true, finalType: state.final.type, reason: 'no\_valid\_part' });

return state;

}

// استجابة نصية

if (part.text) {

state.final = {

type: 'final\_response',

content: part.text.trim(),

action: 'رد نصي مباشر'

};

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:end', { halt: true, finalType: state.final.type, action: state.final.action });

return state;

}

// ✅ دعم الاستجابة المهيكلة (structuredOutput)

if (part.struct) {

state.final = {

type: 'final\_response',

content: JSON.stringify(part.struct, null, 2), // تحويل الكائن المهيكل إلى سلسلة JSON منسقة

action: 'استجابة مهيكلة'

};

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:end', { halt: true, finalType: state.final.type, action: state.final.action });

return state;

}

// استدعاء أدوات

if (part.functionCall) {

// FunctionCall يمكن أن يكون كائنًا واحدًا أو مصفوفة من الكائنات

const callsToExecute = Array.isArray(part.functionCall) ? part.functionCall : [part.functionCall];

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:function\_call\_detected', { count: callsToExecute.length });

let lastToolResult = null;

for (const call of callsToExecute) {

const name = call.name;

const args = GA\_Helpers.normalizeFunctionArgs(call.args);

const meta = GA\_Tools\_Catalog.getToolMetadata?.(name) || {}; // استخدام GA\_Tools\_Catalog

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'executeTool', { name, args, meta });

// استخدام GA\_Helpers.toolExecutionFailsafe

const result = GA\_Helpers.toolExecutionFailsafe(name, args);

// إذا كانت نتيجة الأداة تتطلب إيقاف الـ pipeline (مثل تأكيد المستخدم أو خطأ)

if (result && (result.type === 'error' || result.type === 'confirmation\_required' || result.type === 'model\_switch\_suggestion')) {

state.final = result;

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:tool\_halted\_pipeline', { tool: name, resultType: result.type });

return state;

}

lastToolResult = result;

}

state.final = lastToolResult || GA\_AI\_Dialogue.showMessage('success', `✅ تم تنفيذ ${callsToExecute.length} أداة بنجاح.`);

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:end', { halt: true, finalType: state.final.type, action: `Function Call (${callsToExecute.map(t => t.name).join(', ')})` });

return state;

}

// حالة غير معروفة: إذا لم يكن هناك نص ولا استدعاء دالة ولا استجابة مهيكلة

state.final = GA\_AI\_Dialogue.showError('❌ لا يمكن تفسير رد النموذج: لا يوجد نص، استدعاء أداة، أو استجابة مهيكلة.');

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:end', { halt: true, finalType: state.final.type, reason: 'uninterpretable\_response' });

return state;

}

return {

preProcessIntent,

postProcessResult

};

})(

GAssistant.Tools.Catalog, // حقن GAssistant.Tools.Catalog

GAssistant.AI.Dialogue, // حقن GAssistant.AI.Dialogue

GAssistant.Helpers // حقن GAssistant.Helpers

);