/\*\*

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

\* @file ai\_dispatcher.gs – v16.0

\* @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, GAssistant.Security.PrivacyEnforcer, GAssistant.AI.Memory

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

\*/

var GAssistant = GAssistant || {};

GAssistant.AI = GAssistant.AI || {};

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

GAssistant.AI.Dispatcher = (function(GA\_Tools\_Catalog, GA\_AI\_Dialogue, GA\_Helpers, GA\_Security\_PrivacyEnforcer, GA\_AI\_Memory) { // إضافة GA\_Security\_PrivacyEnforcer و GA\_AI\_Memory

'use strict';

/\*\*

\* @function preProcessIntent

\* @description

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

\* يتعرف على أوامر فورية قابلة للتنفيذ فورًا.

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

\* @returns {object} - كائن الحالة المحدث، مع `halt` إذا تم تنفيذ إجراء فوري.

\*/

function preProcessIntent(state) {

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

const lowerPrompt = state.userPrompt.toLowerCase().trim();

// أوامر فورية لا تتطلب استدعاء النموذج

if (lowerPrompt === 'مسح الذاكرة' || lowerPrompt === 'clear memory') {

GA\_AI\_Memory.clearSessionContext();

state.final = GA\_AI\_Dialogue.showMessage("success", "✅ تم مسح ذاكرة الجلسة.");

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'preProcessIntent:clear\_memory', {});

return state;

}

// يمكن إضافة المزيد من الأوامر الفورية هنا

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'preProcessIntent:no\_immediate\_action', {});

return state;

}

/\*\*

\* @function postProcessResult

\* @description

\* يحلل رد النموذج (نص، استدعاء دالة، إخراج مهيكل) وينفذ الإجراء المناسب.

\* @param {object} state - كائن الحالة الحالي بعد استدعاء النموذج.

\* @returns {object} - كائن الحالة المحدث، مع `final` و `halt` إذا تم الانتهاء.

\*/

function postProcessResult(state) {

if (state.halt) return state;

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

const response = state.response;

let lastToolResult = null;

// 1. معالجة النص العادي

if (response.type === 'text' && response.text) {

state.final = GA\_AI\_Dialogue.showMessage('info', response.text);

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:text\_response', { length: response.text.length });

return state;

}

// 2. معالجة الاستجابة المهيكلة (Structured Output)

if (response.type === 'structured\_output' && response.data) {

// هنا يمكن تحليل الـ JSON وعرضه بشكل مناسب، ربما كجدول

state.final = GA\_AI\_Dialogue.showSummaryTable(response.data, 'نتائج مهيكلة من النموذج');

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:structured\_output', { data: response.data });

return state;

}

// 3. معالجة استدعاءات الدوال (Function Calls)

if (response.type === 'function\_call' && response.functionCalls && response.functionCalls.length > 0) {

const callsToExecute = response.functionCalls;

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

for (const call of callsToExecute) {

const functionName = call.name;

const args = call.args;

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

if (!tool) {

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:tool\_not\_found', { functionName });

state.final = GA\_AI\_Dialogue.showError(`❌ الأداة "${functionName}" غير موجودة في الكتالوج.`);

state.halt = true;

return state;

}

// 🔐 ضمان الخصوصية والأمان: استدعاء PrivacyEnforcer قبل التنفيذ

const privacyCheckResult = GA\_Security\_PrivacyEnforcer.enforcePrivacyForToolCall(functionName, args, state);

if (privacyCheckResult.type === 'error') {

state.final = privacyCheckResult; // عرض رسالة الخطأ من PrivacyEnforcer

state.halt = true;

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:privacy\_check\_failed', { functionName, error: privacyCheckResult.text });

return state;

}

// 🤖 استخدامه في HITL & Function Refinement:

// إذا كانت الأداة تتطلب تأكيدًا بشريًا (Human-in-the-Loop)

if (tool.requiresConfirmation) {

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

`هل أنت متأكد من تنفيذ "${tool.displayName || functionName}"؟`,

`confirm\_${functionName}`,

{ actionType: functionName, ...args } // تمرير الحمولة لتنفيذها لاحقًا

);

state.halt = true; // إيقاف Pipeline بانتظار تأكيد المستخدم

GA\_Helpers.logOperation('GAssistant.AI.Dispatcher', 'postProcessResult:confirmation\_required', { tool: functionName });

return state;

}

// تنفيذ الأداة

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

// إذا كانت نتيجة الأداة تتطلب إيقاف الـ pipeline (مثل خطأ، تأكيد مطلوب، اقتراح تبديل نموذج)

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: functionName, 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: preProcessIntent,

postProcessResult: postProcessResult

};

})(GAssistant.Tools.Catalog, GAssistant.AI.Dialogue, GAssistant.Helpers, GAssistant.Security.PrivacyEnforcer, GAssistant.AI.Memory);