/\*\*

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

\* @file tools\_dialogue.gs – v15.7

\* @module AI\_TOOLS\_DIALOGUE

\* @description

\* أدوات التفاعل والذاكرة والسياق للمساعد الذكي G-Assistant.

\* تشمل:

\* - توحيد هيكل الردود (response format)

\* - دعم سياق الجلسة مع صلاحية مؤقتة (expiresAt)

\* - توصيات ذكية مثل suggestModelSwitch و suggestNextActions

\* - تسجيل كل دالة في سجل الاستخدام (عبر HELPERS.logOperation)

\* تعتمد على: ENV, AI\_MEMORY, HELPERS, AI\_CORE

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

\*/

var AI\_TOOLS\_DIALOGUE = (function(ENV, AI\_MEMORY, HELPERS, AI\_CORE) {

'use strict';

const SETTINGS = ENV.SETTINGS;

// ========== 1. رسائل التفاعل ==========

/\*\*

\* @function showMessage

\* @description

\* تنشئ كائن رسالة موحد لعرضه في واجهة المستخدم.

\* @param {string} level - مستوى الرسالة (success, warning, error, info).

\* @param {string} text - نص الرسالة.

\* @returns {object} - كائن الرسالة.

\*/

function showMessage(level, text) {

const emojiMap = { error: "🚨", warning: "⚠️", success: "✅", info: "ℹ️" };

return { type: "message", level, text: `${emojiMap[level] || '💬'} ${text}` };

}

/\*\*

\* @function showError

\* @description

\* تنشئ كائن رسالة خطأ موحد.

\* @param {string} message - رسالة الخطأ.

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

\*/

function showError(message) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'showError', { message: message });

return showMessage('error', message);

}

/\*\*

\* @function askUserConfirmation

\* @description

\* تطلب تأكيدًا من المستخدم قبل تنفيذ إجراء حساس.

\* @param {string} message - رسالة التأكيد.

\* @param {string} [confirmationType='normal'] - نوع التأكيد (مثال: 'delete', 'critical').

\* @param {object} [payload={}] - بيانات إضافية يمكن تمريرها مع التأكيد.

\* @returns {object} - كائن يطلب التأكيد من الواجهة الأمامية.

\*/

function askUserConfirmation(message, confirmationType = 'normal', payload = {}) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'askUserConfirmation', { message: message, type: confirmationType });

return { type: 'confirmation\_required', message: message, confirmationType: confirmationType, payload: payload };

}

/\*\*

\* @function showSummaryTable

\* @description

\* تعرض بيانات في شكل جدول ملخص.

\* @param {Array<Object>} data - مصفوفة من الكائنات لتمثيل الصفوف.

\* @param {string} title - عنوان الجدول.

\* @returns {object} - كائن يحتوي على بيانات الجدول للعرض.

\*/

function showSummaryTable(data, title = 'ملخص البيانات') {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'showSummaryTable', { title: title, rowCount: data.length });

return { type: 'summary\_table', title: title, data: data };

}

// ========== 2. إدارة سياق الجلسة ==========

/\*\*

\* @function setSessionContext

\* @description

\* تحفظ قيمة في سياق الجلسة.

\* @param {string} key - المفتاح لتخزين القيمة.

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

\* @param {number} [expiryMinutes] - مدة صلاحية القيمة بالدقائق.

\*/

function setSessionContext(key, value, expiryMinutes) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'setSessionContext', { key: key, expiryMinutes: expiryMinutes });

AI\_MEMORY.setSessionContext(key, value, expiryMinutes);

}

/\*\*

\* @function getSessionContext

\* @description

\* تسترجع قيمة من سياق الجلسة.

\* @param {string} key - المفتاح لاسترجاع القيمة.

\* @returns {any|null} - القيمة المخزنة أو null إذا لم تكن موجودة أو انتهت صلاحيتها.

\*/

function getSessionContext(key) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'getSessionContext', { key: key });

return AI\_MEMORY.getSessionContext(key);

}

/\*\*

\* @function clearSessionContext

\* @description

\* تمسح قيمة أو كل سياق الجلسة.

\* @param {string} [key] - المفتاح للمسح، إذا لم يحدد، سيتم مسح كل السياق.

\*/

function clearSessionContext(key) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'clearSessionContext', { key: key || 'all' });

AI\_MEMORY.clearSessionContext(key);

}

/\*\*

\* @function summarizeSession

\* @description

\* تلخص حالة الذاكرة قصيرة المدى (سياق الجلسة).

\* @returns {object} - كائن يحتوي على ملخص الجلسة.

\*/

function summarizeSession() {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'summarizeSession', {});

const ctx = AI\_MEMORY.loadSessionContext();

const keys = Object.keys(ctx).filter(k => k !== 'history' && k !== 'lastPrompt' && k !== 'lastModel'); // استبعاد المفاتيح الداخلية

if (keys.length === 0) {

return { type: "session\_summary", content: "\*\*ذاكرة الجلسة فارغة.\*\*" };

}

const lines = keys.map(k => {

const entry = ctx[k];

const expiry = entry.expiresAt ? ` (تنتهي في: ${new Date(entry.expiresAt).toLocaleTimeString()})` : '';

return `- \*\*${k}\*\*${expiry}: \`${(String(entry.value)).substring(0, 100)}\``;

});

return { type: "session\_summary", content: `\*\*ملخص ذاكرة الجلسة:\*\*\\n${lines.join("\\n")}` };

}

// ========== 3. اقتراحات واسترجاع ==========

/\*\*

\* @function suggestModelSwitch

\* @description

\* تقترح على المستخدم التبديل إلى نموذج AI مختلف.

\* @param {string} modelName - اسم النموذج المقترح.

\* @param {string} message - رسالة توضيحية للاقتراح.

\* @returns {object} - كائن يطلب من الواجهة الأمامية اقتراح تبديل النموذج.

\*/

function suggestModelSwitch(modelName, message) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'suggestModelSwitch', { model: modelName });

return { type: 'model\_switch\_suggestion', model: modelName, message: message };

}

/\*\*

\* @function suggestNextActions

\* @description

\* تقترح مجموعة من الإجراءات التالية المحتملة للمستخدم.

\* @param {string} jsonSuggestions - سلسلة JSON تحتوي على مصفوفة من كائنات الاقتراحات ({ text: string, action: string }).

\* @returns {object} - كائن يحتوي على قائمة بالاقتراحات.

\*/

function suggestNextActions(jsonSuggestions) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'suggestNextActions:start', {});

try {

const arr = JSON.parse(jsonSuggestions);

if (!Array.isArray(arr)) throw new Error("الاقتراحات يجب أن تكون مصفوفة.");

return {

type: 'suggestions',

content: "🧭 ما الخطوة التالية؟",

suggestions: arr

};

} catch (e) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'suggestNextActions:error', { error: e.message });

return showError(`فشل تحليل الاقتراحات: ${e.message}`);

}

}

/\*\*

\* @function repeatLastPrompt

\* @description

\* يعيد تنفيذ آخر أمر تم إرساله إلى المساعد.

\* @returns {object} - كائن الرد النهائي من AI\_CORE.ask.

\*/

function repeatLastPrompt() {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'repeatLastPrompt:start ', {}); // ✅ تحسين: سجل بداية العملية

const ctx = AI\_MEMORY.loadSessionContext();

const prompt = ctx.lastPrompt;

const model = ctx.lastModel || SETTINGS.DEFAULT\_MODEL; // التأكد من أن SETTINGS.DEFAULT\_MODEL موجودة في ENV.SETTINGS

if (!prompt) {

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'repeatLastPrompt:no\_prompt ', {}); // ✅ تحسين: سجل عدم وجود أمر سابق

return showMessage("warning ", "⏪ لا يوجد أمر سابق محفوظ.");

}

// استدعاء AI\_CORE.ask لإعادة تشغيل الـ pipeline

HELPERS.logOperation('AI\_TOOLS\_DIALOGUE', 'repeatLastPrompt:calling\_ai\_core', { prompt, model }); // ✅ تحسين: سجل استدعاء AI\_CORE

return AI\_CORE.ask(prompt, model);

}

// ========== 📦 التصدير ==========\

return {

showMessage: showMessage,

showError: showError,

askUserConfirmation: askUserConfirmation,

showSummaryTable: showSummaryTable,

setSessionContext: setSessionContext,

getSessionContext: getSessionContext,

clearSessionContext: clearSessionContext,

summarizeSession: summarizeSession,

suggestModelSwitch: suggestModelSwitch,

suggestNextActions: suggestNextActions,

repeatLastPrompt: repeatLastPrompt

};

})(

ENV,

AI\_MEMORY,

HELPERS,

AI\_CORE

);