/\*\*

\* @fileoverview ملف البيئة (Environment & Configuration) - v16.2

\* @module GAssistant.Config

\* @description

\* يحتوي على جميع الإعدادات العامة والثابتة للمساعد والمشروع،

\* منظمة ضمن مساحة اسم GAssistant.Config.

\* هذا الملف هو المرجع المركزي للإعدادات المشتركة في ai\_core.gs و Sidebar وغيرها.

\* تم التوسيع والتعليق الكامل لجميع الثوابت وتطبيق معايير Clear Namespacing,

\* Dynamic Access Control, و Secret Store Integration.

\*/

// تعريف الكائن العام GAssistant إذا لم يكن موجودًا بالفعل

var GAssistant = GAssistant || {};

// تعريف وحدة الإعدادات والتكوين ضمن GAssistant

GAssistant.Config = (function() {

'use strict';

// ⏱️ مدة صلاحية الكاش بالثواني (30 دقيقة)

const CACHE\_DURATION\_SECONDS = 1800;

/\*\*

\* @const {object} \_DEFAULT\_SETTINGS

\* @description

\* كائن يحتوي على جميع الإعدادات الافتراضية والثابتة للنظام.

\* يتم قراءة بعض هذه الإعدادات ديناميكيًا من ورقة 'التعليمات الرئيسية'.

\* تم تسميته بـ `\_DEFAULT\_SETTINGS` للإشارة إلى أنه خاص بالوحدة.

\*/

const \_DEFAULT\_SETTINGS = (function () {

return {

ENVIRONMENT: "production", // يمكن تغييره إلى 'staging' أو 'development' عند الاختبار

DEBUG\_MODE: true, // ✅ إذا true: يُسجّل التنبيهات والمفاتيح الناقصة

SYSTEM\_LANGUAGE: "ar", // اللغة الافتراضية (يمكن تعريب أو ترجمة الردود)

SYSTEM\_VERSION: "v16.2", // إصدار المساعد الذكي

// --- 🖥️ واجهة المستخدم ---

SIDEBAR\_FILE: 'Sidebar', // اسم ملف HTML للشريط الجانبي الرئيسي

DOCS\_SIDEBAR\_FILE: 'BuiltinDocsSidebar', // اسم ملف HTML لشريط توثيق الدوال المدمجة

// --- 🗂️ أوراق تحكم النظام ---

MASTER\_INSTRUCTIONS\_SHEET\_NAME: 'التعليمات الرئيسية', // ورقة الإعدادات والتعليمات

LOG\_SHEET\_NAME: 'سجل العمليات', // ورقة تسجيل كل الأوامر

STAGING\_SHEET\_NAME: '🚀 ورشة عمل المبرمج', // ورشة العمل البرمجية

// --- 📊 أوراق محاسبية رئيسية ---

REVENUE\_SHEET: "الإيرادات",

EXPENSE\_SHEET: "المصروفات",

ACCOUNTS\_SHEET: "دليل الحسابات",

CLIENTS\_SHEET: "العملاء",

VENDORS\_SHEET: "الموردين",

DAILY\_SUMMARY\_SHEET: "ملخص يومي",

// --- 🎨 تنسيقات مرئية ---

PRIMARY\_HEADER\_COLOR: "#d9ead3", // لون رؤوس الجداول

HIGHLIGHT\_COLOR: "#fff2cc", // لون التمييز للخلايا

ERROR\_COLOR: "#f4cccc", // لون الخطأ

SUCCESS\_COLOR: "#d9ead3", // لون النجاح

// --- 💰 إعدادات العملة والضرائب ---

DEFAULT\_CURRENCY: "SAR", // تم تحديثها لتكون SAR افتراضيًا كما في طلبك السابق

DEFAULT\_TAX\_RATE: 0.15,

DATE\_FORMAT: "yyyy-MM-dd",

currentUserRole: "viewer", // دور المستخدم الافتراضي (يمكن قراءته من الورقة)

// --- ➗ صيغ حسابية جاهزة ---

FORMULAS: {

SUM: (rangeA1) => `=SUM(${rangeA1})`,

TODAY: "=TODAY()",

INCOME\_STATEMENT: (sales, cost, expense) => `=SUM(${sales}) - SUM(${cost}) - SUM(${expense})`,

PROFIT\_MARGIN: (income, sales) => `=${income}/${sales}`

},

// --- 🧩 قوالب رؤوس الأعمدة الجاهزة ---

TEMPLATES: {

SALES: { SHEET\_NAME: "الإيرادات", HEADERS: ["التاريخ", "الرمز", "المنتج", "الكمية", "سعر الوحدة", "الإجمالي", "القناة", "ملاحظات"] },

EXPENSES: { SHEET\_NAME: "المصروفات", HEADERS: ["التاريخ", "الوصف", "المبلغ", "التصنيف", "المورد", "الفاتورة"] },

CLIENTS: { SHEET\_NAME: "العملاء", HEADERS: ["معرف العميل", "الاسم", "رقم التواصل", "البريد الإلكتروني", "المدينة", "تاريخ الانضمام"] },

VENDORS: { SHEET\_NAME: "الموردين", HEADERS: ["معرف المورد", "اسم الشركة", "رقم التواصل", "مسؤول الاتصال", "مجال العمل"] }

},

// --- 🔒 حماية النطاقات المهمة ---

PROTECTED\_RANGES: {

INCOME\_TOTALS: "الإيرادات!G:G",

EXPENSE\_TOTALS: "المصروفات!C:C"

},

// --- ⚠️ إعدادات أمنية مستقبلية ---

MAX\_ENTRIES\_PER\_DAY: 200,

ALLOWED\_USERS: [], // قائمة الإيميلات المسموح لها استخدام النظام

// تعريف الأدوار المسموح بها ومستخدميها (يمكن قراءتها من PropertiesService أو ورقة)

ALLOWED\_ROLES: {

'admin': ['you@example.com', 'owner@example.com'], // استبدل بالإيميلات الحقيقية

'editor': ['editor@example.com'],

'viewer': [] // أي مستخدم يمكنه العرض

},

// --- 🔑 مفاتيح API (أسماء المفاتيح في PropertiesService) ---

API\_KEYS\_NAMES: { // تم تغيير الاسم لتجنب الالتباس مع المفاتيح الفعلية

GEMINI: "GEMINI\_API\_KEY",

CHATGPT: "CHATGPT\_API\_KEY"

},

// --- 🤖 النماذج المدعومة من Gemini ---

ALLOWED\_MODELS: ["gemini-1.5-pro", "gemini-1.5-flash", "gemini-1.5-flash-lite-preview-06-17"],

DEFAULT\_MODEL: "gemini-1.5-flash",

MODEL\_FOR\_SIMPLE\_TASKS: "gemini-1.0-pro",

// --- ⚙️ إعدادات توليد النموذج ---

GENERATION\_CONFIG: {

temperature: 0.7,

topP: 0.95,

topK: 40,

maxOutputTokens: 8192,

},

// --- 🛡️ إعدادات الأمان للنموذج ---

SAFETY\_SETTINGS: [

{ category: 'HARM\_CATEGORY\_HARASSMENT', threshold: 'BLOCK\_MEDIUM\_AND\_ABOVE' },

{ category: 'HARM\_CATEGORY\_HATE\_SPEECH', threshold: 'BLOCK\_MEDIUM\_AND\_ABOVE' },

],

// --- 🧠 إعدادات التفكير (Thinking) ---

THINKING\_CONFIG: {

thinkingBudget: 10000, // ميزانية التفكير بالمللي ثانية

includeThoughts: true, // هل يجب تضمين الأفكار في الاستجابة؟

},

SAMPLE\_DATA\_ROW\_COUNT: 5,

// --- 🧩 أنماط ملفات الأدوات (للتوثيق الديناميكي) ---

TOOL\_FILE\_PATTERNS: [

{ file: "tools\_sheets.gs", pattern: "TOOLS\_SHEETS" },

{ file: "tools\_accounting.gs", pattern: "TOOLS\_ACCOUNTING" },

{ file: "tools\_general.gs", pattern: "TOOLS\_GENERAL" },

]

};

})();

// ---------------------------------------------------------------------------

// دوال مساعدة داخلية (Private Helpers) - لا يمكن الوصول إليها مباشرة من الخارج

// ---------------------------------------------------------------------------

/\*\*

\* @function \_getAndCacheActiveSettings

\* @description

\* 🧠 يقرأ الإعدادات النشطة للنظام. يحاول جلبها من الكاش أولاً.

\* إذا لم تكن في الكاش، يقرأ الإعدادات الافتراضية ثم يحاول تحديثها

\* من ورقة 'التعليمات الرئيسية' في جداول البيانات.

\* يقوم بتخزين الإعدادات في الكاش للمدة المحددة.

\* @returns {object} - كائن الإعدادات النشطة.

\*/

function \_getAndCacheActiveSettings() {

const cache = CacheService.getScriptCache();

const cached = cache.get("ACTIVE\_SETTINGS");

if (cached) return JSON.parse(cached);

// نبدأ بنسخة عميقة من الإعدادات الافتراضية لتجنب التعديل المباشر

const settings = JSON.parse(JSON.stringify(\_DEFAULT\_SETTINGS));

try {

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(settings.MASTER\_INSTRUCTIONS\_SHEET\_NAME);

if (sheet) {

// قراءة نطاق أوسع لضمان التقاط جميع الإعدادات المحتملة

const range = sheet.getRange("A1:B20").getValues(); // توسيع النطاق لـ B20

range.forEach(([key, value]) => {

const k = key?.toString().trim().toLowerCase();

if (!k || value === undefined || value === '') return; // تجاهل المفاتيح الفارغة أو القيم الفارغة

// تحديث الإعدادات بناءً على المفتاح

switch (k) {

case 'default\_model':

if (settings.ALLOWED\_MODELS.includes(value)) settings.DEFAULT\_MODEL = value;

break;

case 'system\_language':

settings.SYSTEM\_LANGUAGE = value;

break;

case 'default\_currency':

settings.DEFAULT\_CURRENCY = value;

break;

case 'user\_role':

settings.currentUserRole = value;

break;

case 'debug\_mode':

settings.DEBUG\_MODE = (String(value).toLowerCase() === 'true');

break;

case 'system\_version':

settings.SYSTEM\_VERSION = value;

break;

// يمكن إضافة المزيد من المفاتيح هنا لتحديث إعدادات أخرى من الورقة

}

});

}

} catch (e) {

if (settings.DEBUG\_MODE) Logger.log("⚠️ فشل في قراءة الإعدادات من الورقة: " + e.message);

}

// تخزين الإعدادات المحدثة في الكاش

cache.put("ACTIVE\_SETTINGS", JSON.stringify(settings), CACHE\_DURATION\_SECONDS);

cache.put("ACTIVE\_SETTINGS\_LAST\_UPDATED", new Date().toISOString(), CACHE\_DURATION\_SECONDS);

return settings;

}

/\*\*

\* @function \_getLastSettingsUpdateTime

\* @description

\* ⏱️ يُعيد وقت آخر تحديث للإعدادات من الكاش.

\* @returns {string|null} - سلسلة نصية بتاريخ ووقت التحديث بتنسيق ISO، أو null إذا لم يتم التحديث بعد.

\*/

function \_getLastSettingsUpdateTime() {

return CacheService.getScriptCache().get("ACTIVE\_SETTINGS\_LAST\_UPDATED");

}

/\*\*

\* @function \_saveSettingsToDrive

\* @description

\* 💾 يحفظ نسخة احتياطية من الإعدادات النشطة إلى Google Drive كملف JSON.

\* @param {string} [fileName="settings\_backup.json"] - اسم الملف المراد حفظ الإعدادات فيه.

\* @returns {void}

\*/

function \_saveSettingsToDrive(fileName = "settings\_backup.json") {

const settings = \_getAndCacheActiveSettings();

const folderId = PropertiesService.getScriptProperties().getProperty("SETTINGS\_BACKUP\_FOLDER\_ID");

const folder = folderId ? DriveApp.getFolderById(folderId) : DriveApp.getRootFolder();

try {

const file = folder.createFile(fileName, JSON.stringify(settings, null, 2), MimeType.PLAIN\_TEXT);

Logger.log(`✅ إعدادات محفوظة في Drive: ${file.getUrl()}`);

} catch (e) {

if (settings.DEBUG\_MODE) Logger.log("❌ فشل حفظ الإعدادات إلى Drive: " + e.message);

}

}

/\*\*

\* @function \_sendSlackAlert

\* @description

\* 🔔 يرسل تنبيهًا إلى Slack باستخدام Webhook URL.

\* @param {string} message - الرسالة المراد إرسالها.

\* @returns {void}

\*/

function \_sendSlackAlert(message) {

const url = PropertiesService.getScriptProperties().getProperty("SLACK\_WEBHOOK\_URL");

if (!url) {

Logger.log("⚠️ SLACK\_WEBHOOK\_URL غير موجود في خصائص المشروع. لا يمكن إرسال تنبيه Slack.");

return;

}

try {

UrlFetchApp.fetch(url, {

method: "POST",

contentType: "application/json",

payload: JSON.stringify({ text: message })

});

} catch (e) {

Logger.log("❌ فشل إرسال تنبيه Slack: " + e.message);

}

}

// ---------------------------------------------------------------------------

// الواجهة العامة للوحدة (Public Interface)

// ---------------------------------------------------------------------------

return {

/\*\*

\* @function getSetting

\* @description

\* ⚙️ يُعيد قيمة إعداد محدد من الإعدادات النشطة.

\* @param {string} key - مفتاح الإعداد.

\* @returns {\*} - قيمة الإعداد أو null إذا لم يكن موجودًا.

\*/

getSetting: function(key) {

return \_getAndCacheActiveSettings()[key] ?? null;

},

/\*\*

\* @function setSetting

\* @description

\* ✏️ يُعين قيمة لإعداد محدد في خصائص المشروع (Script Properties)

\* ويقوم بمسح الكاش لضمان تحديث الإعدادات عند القراءة التالية.

\* هذا يُستخدم لتغيير الإعدادات التي يجب أن تستمر عبر الجلسات.

\* @param {string} key - مفتاح الإعداد.

\* @param {\*} value - القيمة المراد تعيينها.

\* @returns {void}

\*/

setSetting: function(key, value) {

PropertiesService.getScriptProperties().setProperty(key, String(value));

// مسح الكاش لإعادة تحميل الإعدادات عند الطلب التالي

CacheService.getScriptCache().remove("ACTIVE\_SETTINGS");

CacheService.getScriptCache().remove("ACTIVE\_SETTINGS\_LAST\_UPDATED");

},

/\*\*

\* @function getAllSettings

\* @description

\* ⚙️ يُعيد جميع الإعدادات النشطة (من الكاش أو الورقة).

\* @returns {object} - كائن الإعدادات النشطة.

\*/

getAllSettings: function() {

return \_getAndCacheActiveSettings();

},

/\*\*

\* @function checkAccess

\* @description

\* 🔒 يتحقق من صلاحية وصول المستخدم بناءً على دوره أو بريده الإلكتروني.

\* @param {string} role - الدور المطلوب للوصول (مثال: 'admin', 'editor', 'viewer').

\* @returns {boolean} - صحيح إذا كان المستخدم لديه الصلاحية، خطأ بخلاف ذلك.

\*/

checkAccess: function(role) {

const userEmail = Session.getActiveUser().getEmail();

const settings = \_getAndCacheActiveSettings();

const allowedRolesMap = settings.ALLOWED\_ROLES;

// إذا كان الدور المطلوب هو 'viewer'، فكل المستخدمين مسموح لهم

if (role === 'viewer') return true;

// التحقق من قائمة الإيميلات المسموح بها للدور المحدد

const allowedEmailsForRole = allowedRolesMap[role];

if (allowedEmailsForRole && allowedEmailsForRole.includes(userEmail)) {

return true;

}

// التحقق من قائمة المستخدمين المسموح لهم بشكل عام (إذا لم يكن الدور محددًا)

// هذا الجزء يمكن تعديله ليكون أكثر تحديدًا حسب متطلباتك

if (settings.ALLOWED\_USERS.length > 0 && settings.ALLOWED\_USERS.includes(userEmail)) {

return true;

}

return false;

},

/\*\*

\* @function getSecret

\* @description

\* 🔑 يجلب مفتاحًا سريًا (مثل مفتاح API) من خصائص المشروع بشكل آمن.

\* يتحقق من أن المفتاح المطلوب هو ضمن قائمة المفاتيح السرية المحددة.

\* @param {string} keyName - اسم المفتاح السري المراد جلبه (مثال: "GEMINI\_API\_KEY").

\* @returns {string|null} - قيمة المفتاح السري أو null إذا لم يكن موجودًا أو غير مصرح به.

\*/

getSecret: function(keyName) {

const settings = \_getAndCacheActiveSettings();

const secretKeys = Object.values(settings.API\_KEYS\_NAMES); // المفاتيح السرية هي قيم API\_KEYS\_NAMES

if (!secretKeys.includes(keyName)) {

if (settings.DEBUG\_MODE) Logger.log(`⚠️ محاولة الوصول إلى مفتاح غير سري أو غير معرف: ${keyName}`);

return null;

}

const key = PropertiesService.getScriptProperties().getProperty(keyName);

if (!key && settings.DEBUG\_MODE) {

Logger.log(`⚠️ المفتاح السري "${keyName}" غير موجود في خصائص المشروع!`);

\_sendSlackAlert(`🔐 مفقود المفتاح السري "${keyName}" في مشروع G-Assistant`);

}

return key || null;

},

/\*\*

\* @function setSecret

\* @description

\* 🔒 يُعين قيمة لمفتاح سري في خصائص المشروع بشكل آمن.

\* يتحقق من أن المفتاح المطلوب هو ضمن قائمة المفاتيح السرية المحددة.

\* @param {string} keyName - اسم المفتاح السري المراد تعيينه (مثال: "GEMINI\_API\_KEY").

\* @param {string} value - القيمة المراد تعيينها للمفتاح السري.

\* @returns {void}

\*/

setSecret: function(keyName, value) {

const settings = \_getAndCacheActiveSettings();

const secretKeys = Object.values(settings.API\_KEYS\_NAMES); // المفاتيح السرية هي قيم API\_KEYS\_NAMES

if (!secretKeys.includes(keyName)) {

if (settings.DEBUG\_MODE) Logger.log(`⚠️ محاولة تعيين قيمة لمفتاح غير سري أو غير معرف: ${keyName}`);

return;

}

PropertiesService.getScriptProperties().setProperty(keyName, value);

// مسح الكاش بعد تعيين مفتاح سري لضمان تحديثه عند القراءة التالية

CacheService.getScriptCache().remove("ACTIVE\_SETTINGS");

CacheService.getScriptCache().remove("ACTIVE\_SETTINGS\_LAST\_UPDATED");

},

/\*\*

\* @function refreshSettings

\* @description

\* 🔄 دالة عامة لتحديث الكاش الخاص بالإعدادات يدويًا.

\* تُستخدم بواسطة onOpen() في ui.gs لضمان تحديث الإعدادات عند فتح المصنف.

\* @returns {void}

\*/

refreshSettings: function() {

CacheService.getScriptCache().remove("ACTIVE\_SETTINGS");

CacheService.getScriptCache().remove("ACTIVE\_SETTINGS\_LAST\_UPDATED");

\_getAndCacheActiveSettings(); // إعادة تحميل الإعدادات وتخزينها في الكاش

SpreadsheetApp.getUi().alert("✅ تم تحديث إعدادات البيئة بنجاح!");

},

/\*\*

\* @function syncSettings

\* @description

\* 🔄 يحفظ الإعدادات المحدثة إلى ورقة 'التعليمات الرئيسية' ويقوم بمزامنة الكاش.

\* يقوم أيضًا بحفظ نسخة احتياطية من الإعدادات إلى Google Drive.

\* @param {object} updated - كائن يحتوي على الإعدادات المراد تحديثها (key-value pairs).

\* @returns {void}

\*/

syncSettings: function(updated = {}) {

try {

const settings = \_getAndCacheActiveSettings(); // الحصول على الإعدادات الحالية

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(settings.MASTER\_INSTRUCTIONS\_SHEET\_NAME);

if (!sheet) {

if (settings.DEBUG\_MODE) Logger.log("❌ ورقة التعليمات الرئيسية غير موجودة. لا يمكن مزامنة الإعدادات.");

return;

}

// ربط المفاتيح بقيم الخلايا في ورقة التعليمات الرئيسية

const cellMap = {

'default\_model': 'B1',

'system\_language': 'B2',

'default\_currency': 'B3',

'user\_role': 'B4',

'debug\_mode': 'B5',

'system\_version': 'B6'

};

for (const [key, value] of Object.entries(updated)) {

const targetCell = cellMap[key.toLowerCase()];

if (targetCell) {

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

} else {

if (settings.DEBUG\_MODE) Logger.log(`⚠️ لا يوجد ربط خلية للمفتاح: ${key}`);

}

}

// مسح الكاش لإعادة تحميل الإعدادات بعد التحديث

CacheService.getScriptCache().remove("ACTIVE\_SETTINGS");

CacheService.getScriptCache().remove("ACTIVE\_SETTINGS\_LAST\_UPDATED");

\_getAndCacheActiveSettings(); // إعادة تحميل الإعدادات المحدثة إلى الكاش

// حفظ نسخة احتياطية بعد كل مزامنة ناجحة

\_saveSettingsToDrive(`settings\_backup\_${new Date().toISOString().replace(/:/g, '-')}.json`);

} catch (e) {

if (\_getAndCacheActiveSettings().DEBUG\_MODE) Logger.log("❌ فشل مزامنة الإعدادات إلى الورقة: " + e.message);

}

},

/\*\*

\* @function getCacheDurationLabel

\* @description

\* ⏱️ يعرض مدة الكاش الحالية بطريقة وصفية بناءً على لغة النظام.

\* @returns {string} - وصف مدة الكاش.

\*/

getCacheDurationLabel: function() {

const minutes = CACHE\_DURATION\_SECONDS / 60;

const lang = \_getAndCacheActiveSettings().SYSTEM\_LANGUAGE;

return lang === "ar"

? `⏱️ الكاش فعال لمدة ${minutes} دقيقة`

: `⏱️ Cache is active for ${minutes} minutes`;

},

// -------------------------------------------------------------------------

// دوال الوصول السريع للإعدادات الشائعة (يمكن استبدالها بـ getSetting('KEY\_NAME'))

// -------------------------------------------------------------------------

getColors: function() {

const s = \_getAndCacheActiveSettings();

return {

header: s.PRIMARY\_HEADER\_COLOR,

highlight: s.HIGHLIGHT\_COLOR,

error: s.ERROR\_COLOR,

success: s.SUCCESS\_COLOR

};

},

getFormula: function(key, ...args) {

const s = \_getAndCacheActiveSettings();

const f = s.FORMULAS[key];

return typeof f === "function" ? f(...args) : f;

},

getTemplates: function() { return \_getAndCacheActiveSettings().TEMPLATES; },

getToolFilePatterns: function() { return \_getAndCacheActiveSettings().TOOL\_FILE\_PATTERNS; },

getGenerationConfig: function() { return \_getAndCacheActiveSettings().GENERATION\_CONFIG; },

getSafetySettings: function() { return \_getAndCacheActiveSettings().SAFETY\_SETTINGS; },

getThinkingConfig: function() { return \_getAndCacheActiveSettings().THINKING\_CONFIG; },

getAllowedModels: function() { return \_getAndCacheActiveSettings().ALLOWED\_MODELS; },

getMasterInstructionsSheetName: function() { return \_getAndCacheActiveSettings().MASTER\_INSTRUCTIONS\_SHEET\_NAME; },

getLogSheetName: function() { return \_getAndCacheActiveSettings().LOG\_SHEET\_NAME; },

getStagingSheetName: function() { return \_getAndCacheActiveSettings().STAGING\_SHEET\_NAME; },

getRevenueSheetName: function() { return \_getAndCacheActiveSettings().REVENUE\_SHEET; },

getExpenseSheetName: function() { return \_getAndCacheActiveSettings().EXPENSE\_SHEET; },

getAccountsSheetName: function() { return \_getAndCacheActiveSettings().ACCOUNTS\_SHEET; },

getClientsSheetName: function() { return \_getAndCacheActiveSettings().CLIENTS\_SHEET; },

getVendorsSheetName: function() { return \_getAndCacheActiveSettings().VENDORS\_SHEET; },

getDailySummarySheetName: function() { return \_getAndCacheActiveSettings().DAILY\_SUMMARY\_SHEET; },

getDateFormat: function() { return \_getAndCacheActiveSettings().DATE\_FORMAT; },

getDefaultTaxRate: function() { return \_getAndCacheActiveSettings().DEFAULT\_TAX\_RATE; },

getProtectedRanges: function() { return \_getAndCacheActiveSettings().PROTECTED\_RANGES; },

getMaxEntriesPerDay: function() { return \_getAndCacheActiveSettings().MAX\_ENTRIES\_PER\_DAY; },

getAllowedUsers: function() { return \_getAndCacheActiveSettings().ALLOWED\_USERS; },

getCurrentUserRole: function() { return \_getAndCacheActiveSettings().currentUserRole; },

getModelForSimpleTasks: function() { return \_getAndCacheActiveSettings().MODEL\_FOR\_SIMPLE\_TASKS; },

getCacheDurationSeconds: function() { return CACHE\_DURATION\_SECONDS; },

// تم تحديث getEnvironment و getCurrencyCode و getCacheTTL لاستخدام getSetting أو القيمة الثابتة

getEnvironment: function() { return \_getAndCacheActiveSettings().ENVIRONMENT; },

getCurrencyCode: function() { return \_getAndCacheActiveSettings().DEFAULT\_CURRENCY; },

getCacheTTL: function() { return CACHE\_DURATION\_SECONDS; }, // تستخدم الثابت مباشرة

};

})();