// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// --- START OF FILE: 01\_config.gs ---

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

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

\* @file 01\_config.gs

\* @module GAssistant.Config

\* @description

\* الوحدة المركزية لإدارة جميع إعدادات المشروع.

\* تدمج الإعدادات الافتراضية مع الإعدادات المحفوظة، توفر Caching،

\* وتتضمن الآن وظائف للتحقق من الصحة وتحديث الإعدادات ديناميكيًا.

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

\*/

var GAssistant = GAssistant || {};

GAssistant.Config = (() => {

'use strict';

const CACHE = CacheService.getScriptCache();

const CACHE\_KEY = 'g\_assistant\_config\_v5'; // تحديث الإصدار لضمان مسح الكاش القديم

const CACHE\_DURATION = 1800; // 30 دقيقة بالثواني

// مصدر الحقيقة الواحد للإعدادات الافتراضية.

const DEFAULTS = {

// General Settings

DEBUG\_MODE: true,

DEFAULT\_LANGUAGE: "ar",

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

DEFAULT\_CURRENCY: "SAR",

// AI Core Settings

API\_KEY: "", // سيتم جلبه من PropertiesService

GEMINI\_PRO\_MODEL: "gemini-1.5-pro-latest",

GEMINI\_FLASH\_MODEL: "gemini-1.5-flash-latest",

API\_ENDPOINT: "https://generativelanguage.googleapis.com/v1beta/models/",

ALLOWED\_MODELS: ["gemini-1.5-pro-latest", "gemini-1.5-flash-latest", "gemini-1.0-pro"],

GENERATION\_CONFIG: {

temperature: 0.7,

topP: 0.95,

maxOutputTokens: 8192,

},

SAFETY\_SETTINGS: [

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

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

{ category: "HARM\_CATEGORY\_SEXUALLY\_EXPLICIT", threshold: "BLOCK\_MEDIUM\_AND\_ABOVE" },

{ category: "HARM\_CATEGORY\_DANGEROUS\_CONTENT", threshold: "BLOCK\_MEDIUM\_AND\_ABOVE" },

],

// System & Logging Settings

OPERATION\_LOG\_SHEET: "سجل العمليات",

DEVELOPMENT\_LOG\_SHEET: "سجل التطوير",

MAX\_LOG\_ENTRIES: 500,

// Tool-specific Settings

REVENUE\_SHEET: "الإيرادات",

EXPENSES\_SHEET: "المصروفات",

// UI Settings

ASSISTANT\_SIDEBAR\_FILE: "ui/AssistantSidebar",

DEVELOPER\_SIDEBAR\_FILE: "ui/DeveloperSidebar",

WEB\_APP\_FILE: "ui/WebApp",

};

let activeConfig = null;

/\*\*

\* يقوم بتحميل الإعدادات عن طريق دمج الإعدادات الافتراضية مع خصائص السكربت.

\* @private

\*/

function \_loadConfig() {

const props = PropertiesService.getScriptProperties().getProperties();

const mergedConfig = { ...DEFAULTS };

for (const key in props) {

if (DEFAULTS.hasOwnProperty(key)) {

try {

mergedConfig[key] = JSON.parse(props[key]);

} catch (e) {

mergedConfig[key] = props[key];

}

}

}

if (props.hasOwnProperty('DEBUG\_MODE')) {

const val = String(props.DEBUG\_MODE).toLowerCase();

mergedConfig.DEBUG\_MODE = val === 'true';

}

mergedConfig.API\_KEY = props.GEMINI\_API\_KEY || "";

mergedConfig.\_\_source = "ScriptProperties + Defaults";

console.log("Configuration loaded from PropertiesService.");

return mergedConfig;

}

/\*\*

\* يجلب كائن الإعدادات الكامل، من الكاش إذا كان متاحًا، أو يقوم بتحميله.

\* @param {boolean} [forceRefresh=false]

\*/

function getAll(forceRefresh = false) {

if (activeConfig && !forceRefresh) return activeConfig;

if (!forceRefresh) {

const cached = CACHE.get(CACHE\_KEY);

if (cached) {

try {

activeConfig = JSON.parse(cached);

activeConfig.\_\_source = "Cache";

return activeConfig;

} catch(e) { console.warn("Could not parse cached config. Reloading."); }

}

}

activeConfig = \_loadConfig();

CACHE.put(CACHE\_KEY, JSON.stringify(activeConfig), CACHE\_DURATION);

return activeConfig;

}

/\*\*

\* يجلب قيمة إعداد معين.

\* @param {string} key

\*/

function get(key) {

return getAll()[key];

}

/\*\*

\* يقوم بتعيين أو تحديث إعداد في ScriptProperties.

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

\* @param {any} value - القيمة الجديدة.

\*/

function set(key, value) {

try {

const props = PropertiesService.getScriptProperties();

const valueToStore = typeof value === 'object' ? JSON.stringify(value) : value;

props.setProperty(key, valueToStore);

activeConfig = null;

CACHE.remove(CACHE\_KEY);

console.log(`Configuration key "${key}" updated successfully.`);

return true;

} catch (e) {

console.error(`Failed to set configuration key "${key}".`, e);

return false;

}

}

/\*\*

\* دالة للتحقق من صحة الإعدادات الأساسية.

\* @returns {boolean}

\*/

function validate() {

const config = getAll(true);

const requiredKeys = ['API\_KEY', 'API\_ENDPOINT', 'GEMINI\_PRO\_MODEL'];

for (const key of requiredKeys) {

if (!config[key] || (String(config[key]).trim() === '')) {

console.error(`CONFIG VALIDATION FAILED: Required key "${key}" is missing or empty.`);

return false;

}

}

return true;

}

/\*\*

\* يولد وصفًا نصيًا لجميع الإعدادات الحالية.

\* @returns {string} - وصف منسق بصيغة Markdown.

\*/

function describeConfig() {

const cfg = getAll();

const description = Object.entries(cfg)

.map(([key, value]) => `- \*\*${key}\*\*: \`${JSON.stringify(value)}\` (Type: \`${typeof value}\`)`)

.join('\n');

return `### ⚙️ G-Assistant Current Configuration (Source: ${cfg.\_\_source})\n${description}`;

}

return {

get,

getAll,

set,

validate,

describeConfig

};

})();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// --- END OF FILE: 01\_config.gs ---

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*