Out Of Process - Windows Forms Designer

ITLightON

**Auteurs :**

Jérôme Perciot

**Résumé :**

Ce document décrit les échanges nécessaires pour la création du projet « OopWindowsFormsDesigner ».

**Visas :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Approuvé par** | **Date** | **Nom** | **Signature** |
| *JPT* | 06/05/2018 | PERCIOT Jérôme | *DigJPT#0001* |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Vérifié par** | **Date** | **Nom** | **Signature** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Historiques des éditions :**

|  |  |  |
| --- | --- | --- |
| **Édition** | **Date** | **Liste des paragraphes ajoutés ou modifiés** |
| 0\_1 | 08/07/2020 | Création du document |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table des matières

[2 Historique 5](#_Toc44054655)

[3 Concepts 5](#_Toc44054656)

[3.1 Fichier « UiKit » 5](#_Toc44054657)

[3.2 Fichier « UiKitProject » 5](#_Toc44054658)

[3.3 Base de données 6](#_Toc44054659)

[3.4 Architecture 6](#_Toc44054660)

[3.4.1 Assembly « OopDesigner » 6](#_Toc44054661)

[3.4.2 Assembly « ToolKit » 6](#_Toc44054662)

[3.5 Description de l’interface utilisateur 6](#_Toc44054663)

[3.5.1 Ecran principal 6](#_Toc44054664)

[Figure 1 - Exemple de l'écran principal 7](#_Toc44054651)

# Historique

En 2020, j’ai pris la décision de développer mon propre système de design de composants / contrôles Windows Forms mais qui ne soit pas intégré à Visual Studio.

Le designer de Visual étant abandonné et pauvre depuis trop longtemps, mon système doit être exécuté en mode « Out-of-process » pour pouvoir être exécuté dans un environnement autonome.

Le but de cet outil est de prendre la place du designer de Visual Studio et de générer son contenu en utilisant les composants prévus à cet effet, et dans lequel vous pourrez décider de ce que vous voulez inscrire dans le designer de Visual Studio.

Un mode de compilation sera également proposé pour vous permettre de compiler le projet et de générer une assembly de test (sous réserve de faisabilité technique).

Un des nombreux avantages de cette solution, est que les applications Windows Forms pourront être portées sur .NET Core sans se préoccuper de cibler le .NET Framework pour profiter du design automatique.

Les commentaires doivent être saisis sur chaque fonction / classe publique en anglais.

L’application est conçue pour fonctionner avec « **DevExpress** ».

# Concepts

## Fichier « UiKit »

Un nouveau type de fichier sera créé automatiquement par le système (stocké en dehors de la solution ou du projet sélectionné). Le type de ce fichier est un fichier « **Json** », sa lecture et son écriture seront donc très rapides.

## Fichier « UiKitProject »

Ce fichier décrit l’ensemble des fichiers « **UiKit** » qui ont été générés, et leur mode de fonctionnement en sortie.

Ce sont ces fichiers qui génèrent ensuite les «. **Designer.cs** » correspondants.

Un exemple de fichier UiKitProject :

{

Version : 1.0,

Date : 2020-02-04T13 :58 :13Z01,

Solution :

{

RootPath : « c:\temp\myproject.uikitproject

}

}

## Base de données

Une base de données interne sera créée pour stocker les fichiers qui serviront à votre solution et votre projet.

Le moteur de la base utilisée sera « **Realm** ». **ATTENTION**, cette base de données est assez sensible notamment au niveau des accès Inter-Threads.

## Architecture

### Assembly « OopDesigner »

Cette bibliothèque C# permet de gérer l’ossature de votre projet (« \***. UiKitProject** »).

Cette librairie devra proposer les fonctionnalités suivantes :

* Lecture/écriture des fichiers Json de configuration. L’utilisation de **Newtonsoft.Json** sera préconisé.
* Interrogation des composants pouvant être ajoutés au designer interne. L’Api permet d’enregistrer si cela est nécessaire les composants d’autres parties, une fonction d’enregistrement sera alors prévue à cet effet.
* Les composantes de type « **Designer** » seront purement et simplement ignorées.

### Assembly « ToolKit »

Les assemblies principales de l’application proviennent pour la plupart de ITLightON, donc sont propriétaires.

Le tableau ci-dessous recense la liste des assemblies intégrées dans l’application :

|  |  |
| --- | --- |
| Nom de l’assembly | Description |
| ITLightON.dll | Assembly principale gérant l’ensemble des échanges communs entre toutes les parties. |
| ITLightON.Components.Windows | Assembly contenant les composants Windows Forms applicable à l’application. |

### Assembly « Spécifiques »

Les assemblies spécifiques sont celles utilisées dans l’application « **OopWinformsDesigner** » pour sérialiser les informations des écrans/fenêtres applicatifs.

## Description de l’interface utilisateur

### Ecran principal

L’écran principal se compose d’un menu de type « **Ribbon** » vous permettant de sélectionner une solution Visual Studio.

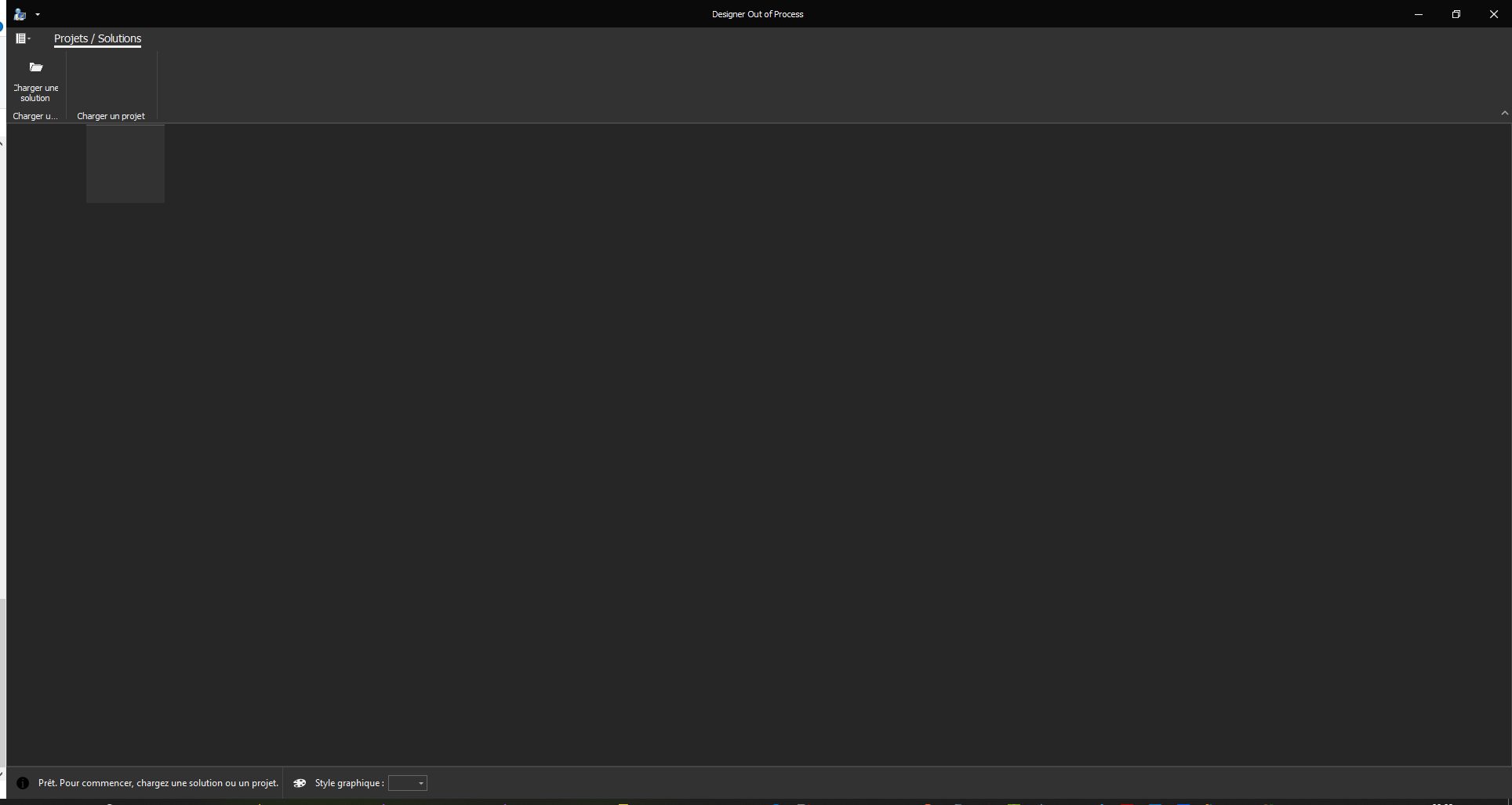


Figure 1 - Exemple de l'écran principal

#### Menu « Ruban »

Le menu principal de l’application permet d’ouvrir une solution (fichier propre à Microsoft Visual Studio). Seuls les fichiers « .sln » dont gérés, les anciennes versions « .dsw » ne sont pas prises en charge.

##### Ouvrir une solution

Ouvre une boîte de dialogue pour ouvrir un fichier de solution Visual Studio.

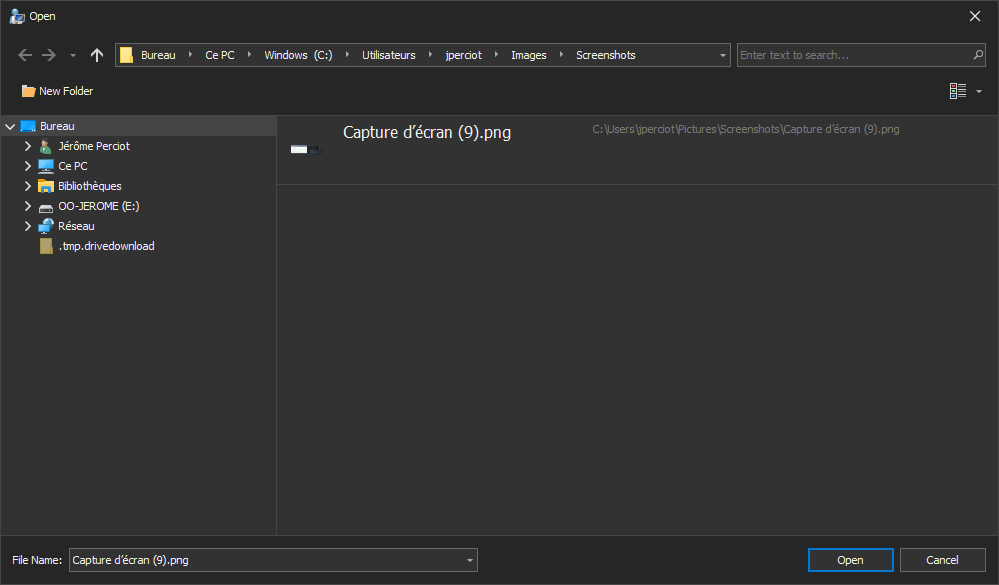


Figure - Chargement d'une solution

Une fois le fichier solution récupéré et chargé (opération qui peut prendre un peu de temps), l’interface est modifiée dynamiquement pour référencer tous les projets contenant des fenêtres Windows Forms (ayant un designer).

Charger une solution décharge celle existante, donc si une solution a déjà été chargée, prévoir un paramètre pour demander la confirmation de la fermeture de la solution avant de charger la nouvelle.

#### Barre d’état

La barre d’état propose un certain nombre d’éléments visuels qui aident à exploiter l’application et à améliorer l’expérience utilisateur.

##### Texte de statut

Il s’agit du texte qui s’affiche en bas à gauche, et indique le statut actif de l’application.

##### Thème graphique

Ce composant permet de sélectionner le thème graphique de l’application.