TP – Un cas d'école dans l'industrie

Nicolas DAGNAS

Durée du TP : 10h

Objectif du TP

- Ce TP a pour objectif la création d'une application de gestion d'un catalogue de fournitures de bureau par l'intermédiaire d'une base de données de type SQLite.

- Il va vous être demandé trois fonctionnalités :
  - Intégrer un fichier « csv » pour alimenter la base de données.
  - Permettre la gestion de la base de données via des écrans clairs et explicites.
  - Permettre l'export de toutes les données dans un fichier « csv » au même format que le fichier « csv » fournis pour la première demande.

- Une série de contraintes seront explicitées au fur et à mesure de l'avancé de votre projet. Ces contraintes sont là pour vous donner un aperçu de ce que l'on trouve dans le monde professionnel. Le respect de ses contraintes représenteront une part importante de votre note finale, suivez-les rigoureusement.

TP – Les règles de codage

1. Votre code devra être commenté et documenté.
2. Les noms des classes, structures, attributs, méthodes et variables devront suivre le même schéma, première lettre de chaque mot en majuscule) comme présenté ci-après (Ex: OnLoad, ObjetLeft, etc…). Les variables trop courtes de type « i » ou « Id » sont proscrites.
3. Espacez vos lignes de codes (ni trop, ni trop peu) pour l'éclaircir.

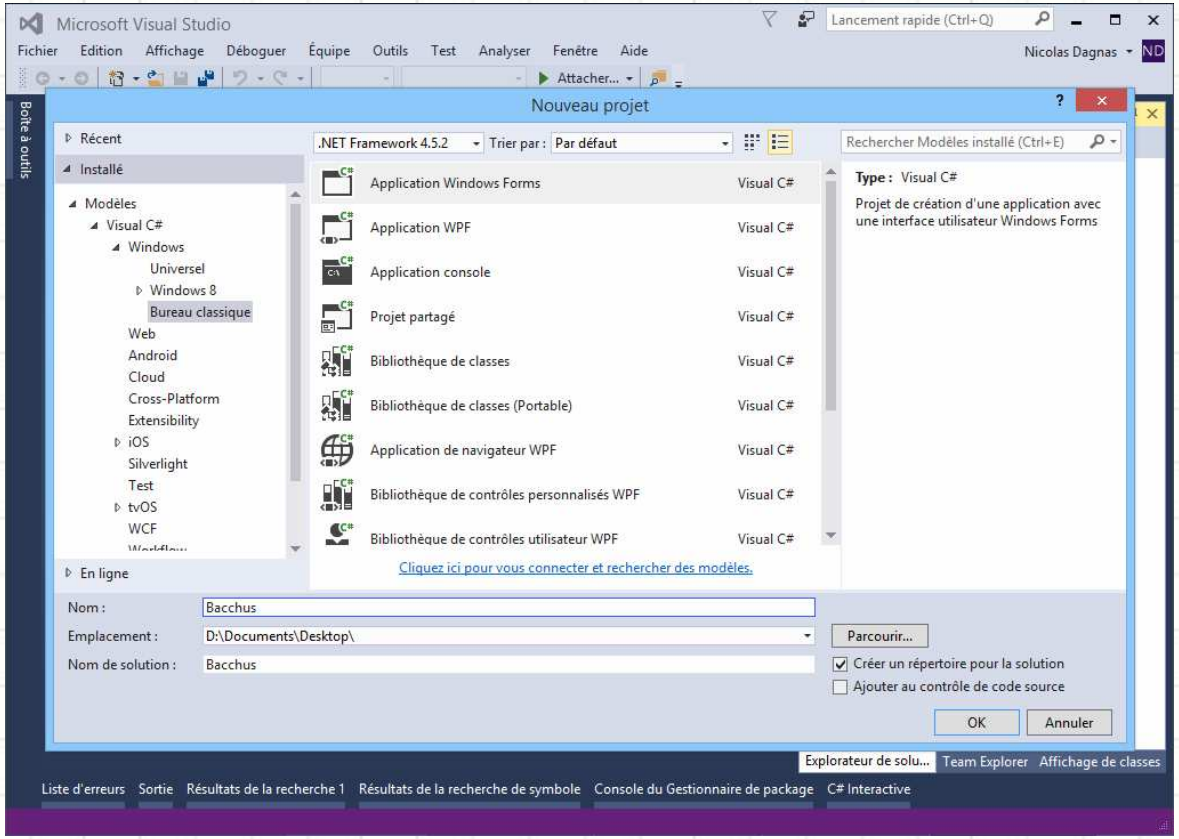
```
/// <summary>
/// Déclenche l'événement Load.
/// </summary>
/// <param name="Args"><b>EventArgs</b> qui contient les données de l'événement.</param>
protected override void OnLoad ( EventArgs Args )
    {
    base.OnLoad ( Args );

    // Récupère la valeur de la propriété d'application 'Left'

    object SettingLeft = Bacchus.Properties.Settings.Default["Left"];

    // Si on a une valeur, on l'applique

    if ( SettingLeft != null && SettingLeft is int )
        this.Left = (int)SettingLeft;

    //...
    }
```

Le rendu du TP

- Le TP sera à rendre au plus tard le lundi 3 juin 2019.
- En cas de TP en binôme, un seul rendu suffira, mais n'oubliez pas de préciser les noms et prénoms de chaque intervenant.
- Le rendu devra se faire à l'aide d'une archive contenant votre projet sans les dossiers « bin » et « obj » (avec ces dossiers, l'archive serait trop grosse) et sans les librairies « SQLite ».
- La propreté du code et la simplicité d'utilisation de vos fenêtres représentes la majeure partie de votre note. Donc inutile d'allez trop vite, soignez votre travail car vous pouvez avoir une meilleure note avec un travail incomplet mais propre qu'avec un travail bâclé même si fonctionnel.

TP – Nouveau projet

- Créez maintenant un nouveau projet de type « Application Windows Form » que vous nommerez « Bacchus ».
- Renommez le nom de la fenêtre principale « Form1 » en « FormMain » via un clic droit sur le nom de l'objet dans l'explorateur de solution.



TP – Fenêtre principale

- Ajoutez un menu « Fichier » à votre fenêtre principale.
  - Ajoutez à votre menu les sous-menus : « Actualiser », « Importer » et « Exporter ».
- Ajoutez un objet « StatusStrip ».
- Ajoutez un objet « TreeView » à votre fenêtre qui devra être « docké » à gauche.
- Ajoutez un objet « Splitter » qui va automatiquement se coller à droite de votre « TreeView ».
- Ajouter enfin un objet « ListView » qui devra remplir le reste de la fenêtre. L'objet devra être en mode « détails ».

TP – Sous-menu « Importer »

- Pour exploiter le fichier « Bacchus.SQLite » fournis, vous commencerez par récupérer le package « System.Data.SQLite » sur Nuget directement via Visual Studio.

- Ajoutez le fichier « Bacchus.SQLite » fourni à votre projet en le configurant pour qu'il soit copié dans le répertoire de sortie quand il est modifié.

- Maintenant que c'est fait, codez votre sous-menu « Importer » pour qu'il ouvre une fenêtre modale (centrée par rapport à la fenêtre principale) contenant :
  - Un bouton permettant de sélectionner un fichier « csv ».
  - Un champ contenant le nom du fichier sélectionné.
  - Un bouton permettant de lancer l'intégration en mode écrasement.
  - Un bouton permettant de lancer l'intégration en mode ajout.
  - Une barre de progression représentant l'intégration des données.

- Attention, lisez la suite avant de commencer le codage de l'intégration.

TP – Sous-menu « Importer »

- Le résultat de l'intégration, nombre d'articles ajoutés, anomalies, devra être affiché.

- Comme on dit toujours, qui peut le plus, peut le moins. Donc pour l'utilisation des données, vous utiliserez exclusivement les objets « SQLiteConnection », « SQLiteCommand » et « SQLiteDataReader ». Vous n'utiliserez aucun schéma simplifiant l'utilisation des données, en gros vous ferez vos propres requêtes.

- La base de données vous étant fournis, à vous de l'analyser pour voir comment intégrer les données.

- Une astuce : ne faites pas une intégration « à l'arrache », soyez plus malin ;) Par exemple, la création et l'utilisation de structures reflétant les tables de la base de données pourrait être une bonne piste.

TP – Sous-menu « Exporter »

- Vous allez maintenant coder votre sous-menu « Exporter ». C'est bien entendu la même chose que l'intégration mais à l'envers.

- Cela devra se faire tout naturellement via une fenêtre modale (toujours centrée par rapport à la fenêtre principale).

- Attention, le format des données du fichier « csv » généré devra être rigoureusement le même que le format du fichier fournis, c'est-à-dire qu'on doit être capable d'intégrer ce fichier via le sous-menu « Intégrer » sans anomalie.

TP – Fenêtre principale (suite)

- Maintenant, faites apparaître dans votre objet « TreeView » les différents éléments :
  - Articles
  - Marques
  - Familles
  - Sous familles

- Un clic sur un de ces éléments doit charger votre objet « ListView » avec les données associées (Les noms des colonnes doivent correspondre aussi aux données).

- Dans chacun des cas, vous devrez gérer le trie des données à l'écran (exclusivement dans l'objet « TreeView », le but n'est pas de refaire une requête) via un clic sur les entêtes de colonne et gérer la notion de groupes d'éléments en reflétant le trie actuel.

TP – Interactions

- Les interactions suivantes avec l'objet « ListView » sont à coder :
  - La touche « Entrée » ou un double clic sur un élément ouvrira la fenêtre de modification de l'élément associé.
  - La touche F5 rechargera la liste des éléments tout comme le sous-menu « Actualiser ».
  - La touche « Supp » demandera la suppression de l'élément sélectionné.
  - Le clic droit devra ouvrir un menu surgissant permettant d'ajouter un élément, de modifier ou de supprimer l'élément sélectionné.

TP – Fenêtres Ajout/Modif.

- Les fenêtres d'ajout et de modification d'un élément devront toutes êtres modales et centrée par rapport à la fenêtre principale.

- Dans le cas de la fenêtre d'ajout/modification d'un article, vous utiliserez des objets « ComboBox » pour sélectionner les « marques », « familles » et/ou « sous-famille ».

- Vous contrôlerez aussi l'existence, le format, ou la taille de ce qui est sélectionné/saisie pour éviter toute erreur (on ne peut valider la fenêtre si un « ComboBox » est vide par exemple ou si une référence est trop grande.

- Toutes vos fenêtres devront respecter les règles basiques du design. Les champs doivent être alignés et la tabulation doit suivre un chemin logique.

- En gros, vous devez assister l'utilisateur finale pour qu'il n'ait pas à trop réfléchir lors de l'utilisation de vos fenêtre, imaginez comme si l'utilisateur n'y connaissait strictement rien.

TP – Compléments

- Enregistrez la position de votre fenêtre principale quand vous la fermez. Pour cela, utilisez « Settings » comme vous pouvez le voir dans le *slide* 3, le statut « Maximisé » ne doit pas être oublié.

- Puisque maintenant nous avons cette information, replacez votre fenêtre principale lors de son lancement à son dernier emplacement.

- Nous l'avons oublié, mais nous avons un objet « StatusStrip » sur notre fenêtre principale, et bien placez-y le nombre d'articles actuel dans notre base de données et mettez-le à jour quand cela vous semble logique de le faire ;)