4 Fonctionnalités Phase III (juin 2020)

Lisez l'énoncé attentivement jusqu'à la fin

4.1 Mise en situation et objectifs

L'objectif de cette dernière partie de laboratoire est de mettre en évidence dans le cadre d'une application WPF, quelques techniques non encore mises en place durant les précédents laboratoires.

Avec un peu de recul, vous vous êtes certainement rendu compte que vous n'aviez pas toujours emprunté le meilleur chemin pour construire votre application. C'est la raison pour laquelle, une nouvelle application est créée à partir de rien pour cette dernière partie de laboratoire. Celle-ci devrait normalement profiter de votre expérience récente dans le domaine.

Pour cette dernière application, nous allons faire appel à votre créativité. Vous avez certainement un hobby, un sport, un centre d'intérêt, une thématique pour laquelle vous pourriez créer cette application. Les seuls éléments qui sont imposés sont des éléments techniques qui doivent apparaître dans celle-ci. La thématique abordée doit être autre que celle abordée lors des précédentes phases du laboratoire et validée par le professeur de laboratoire qui pourrait refuser une thématique abordée déjà par un autre étudiant.

4.2 Les composants de l'application

4.2.1 Les données

Trois classes de données de base doivent être créées. Celles-ci ont des relations entre elles (référence de l'une dans l'autre, héritage, …)

Elles doivent contenir différentes sortes de données :

- string,
- nombre,
- date,
- image, dont on mémorise le chemin d'accès dans la classe sous la forme d'une string

Des collections de données doivent être gérées.

D'autres classes doivent être créées pour les besoins d'une architecture applicative, orientée objet plus performante.

4.2.2 Persistance des données

Les données doivent être enregistrées de façon XML ou binaire.

Nous préférons le format XML pour ses nombreux avantages : lisibilité, puissance d'expression, portabilité, etc… Néanmoins, nous sommes conscients que certains types de relation ne peuvent être écrits au format XML dans ce cas nous acceptons que vous passiez au format binaire.

4.2.3 Interface utilisateur

Les interfaces utilisateur sont créées en WPF. Elles utilisent au mieux les notions de layout et de databinding (et donc de DataContext). Elles doivent être le plus ergonomiques, intuitives, agréables à utiliser et redimensionnables possible. Nous accordons plus d'importance à la qualité et qu'à la beauté des interfaces.

La fenêtre principale manipule les données. Celles-ci peuvent être créées, sélectionnées pour en voir les détails, modifiées ou supprimées.

Pour la création et la modification des données, cela se fera au moins une fois via un formulaire, au moins une fois via une DataGrid.

Elle comporte au moins une DataGrid personnalisée ou une TreeView personnalisée.

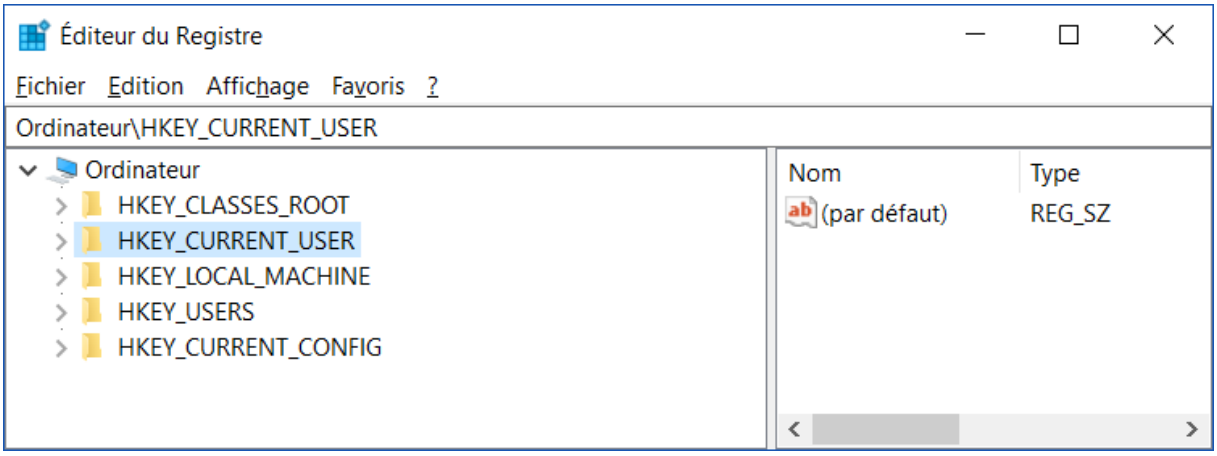
Elle comporte au moins un dataconverter.

Les composants utilisés sont adaptés à la donnée manipulée. Par exemple dans le cas de la classe DateTime qui représente une date (et une heure si nécessaire), les composants WPF de type calendrier (Calendar) ou sélectionneur de date peuvent servir (DatePicker). Pour info, DateTime.Now est une propriété static de DateTime qui retourne un objet qui donne la date et l'heure à l'instant présent.

Elle doit faire apparaître une fenêtre secondaire modale dans laquelle il est possible de régler au moins 2 paramètres de l'application. Parmi ceux-ci, on trouve la localisation du chemin d'accès au dossier dans lequel les fichiers sont enregistrés. Ces paramètres sont enregistrés dans « l'éditeur de registre » (Registry - voir §4.2.4).

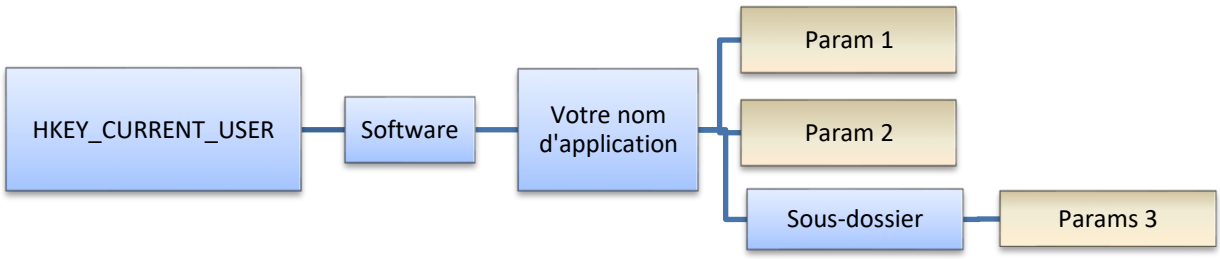
4.2.4 Données de la registry

L'éditeur de registre (registry en anglais, exécutable « regedit.exe ») est l'application Windows qui permet de régler les paramètres d'installation de votre machine et de vos applications. Dans l'invite de commande, lancer « regedit » pour la faire apparaître.



Les paramètres peuvent être modifiés par programmation. Voici la structure que nous conseillons de mettre en place ici.

La registry (classe Registry) permet de mémoriser plusieurs informations dans l'arborescence présentée ci-dessous.



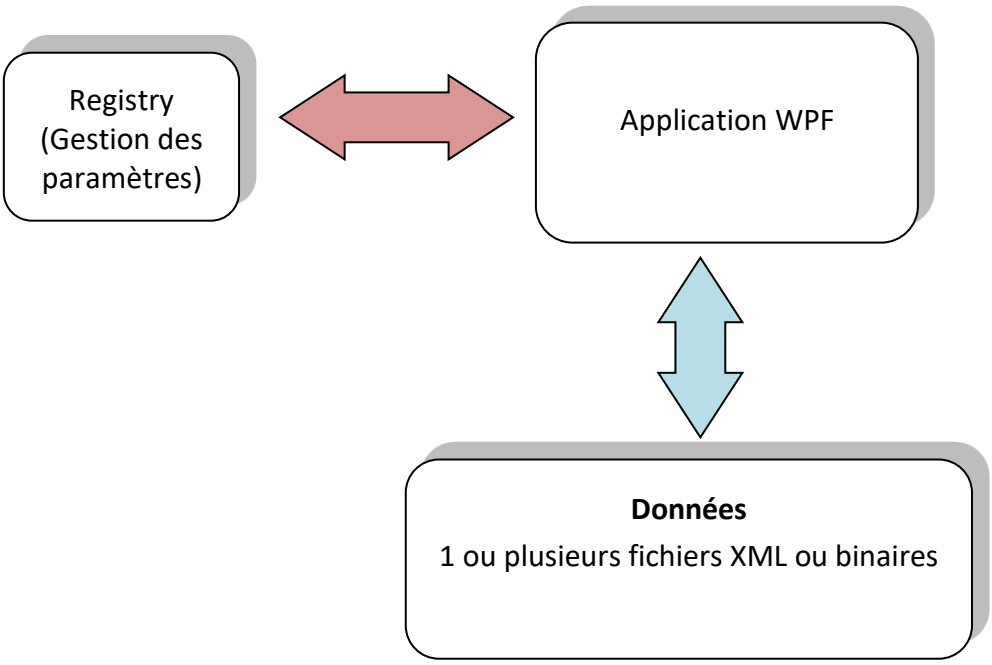
Vous avez la maîtrise de votre hiérarchie de paramètres à partir du nom de votre application.

Il s'agit donc, à l'aide d'une nouvelle classe que l'on pourrait appeler MyAppParamManager de proposer une gestion complète d'accès aux paramètres de l'application.

- Cette classe offre des propriétés qui permettent d'accéder directement aux données de la Registry ou qui passent pas l'utilisation de variables membre et leurs propriétés associées initialisées à partir de méthodes telles que LoadRegistryParameter() ou SaveRegistryParameter(). Les deux méthodes ont leurs avantages et leurs inconvénients.
- *Ressources* : Voir « Etude C# et prog .NET –Part 6 V1.0.pdf » sur Moodle.

4.3 **Architecture**

Vous veillerez à découper votre solution en plusieurs projets en positionnant les différents fichiers dans le projet qui parait le plus pertinent.



4.4 **Méthodologie**

4.4.1 Choix de la thématique et classes

Identifiez la thématique sur laquelle vous voulez travailler. Identifiez les classes demandées, les relations entre elles, définissez les variables membres.

Ce projet est un excellent entrainement pour mettre en avant, ou développer, vos compétences techniques, mais aussi communicatives. Il va de soi qu'un tel projet demande un échange régulier avec votre professeur pendant les heures de laboratoire afin de faire évoluer correctement votre projet. Nous pensons vous avoir transmis les outils nécessaires pour mettre sur pied une application WPF *digne* et *complète* qui permet la gestion d'un domaine qui *vous* passionne. Voyez ce projet comme une opportunité de nous montrer, à nous, ou un futur maitre de stage, ce dont vous êtes déjà capable. Nous attendons de vous une démarche proactive.

Pour faire valider votre projet par votre professeur de laboratoire, déposez votre brève description de celui-ci, description des classes (ou diagramme de classes) dans le dossier GitHub associé au projet. Le lien GitHub pour ce nouveau projet sera accessible durant la journée du 30 mars 2020.

Méthodologie du dépôt de votre document :

1. Créer votre Solution Visual Studio contenant votre projet WPF
2. Ajouter à ce projet votre fichier portant l'extension .md (permet une bonne mise en page sur GitHub) ou .txt.
3. Dans Visual Studio, effectuez un « Add To Source Control » → choisir Git et le nouveau lien pour la remise de votre projet.
4. **Prévenir votre professeur de laboratoire que vous souhaitez faire valider votre projet.**
5. Conseils :
    a. Ne déposez pas le fichier seul sur le repository sans y mettre en même temps votre Solution Visual Studio. En effet, vous seriez obligés alors de fusionner les deux, ce qui est possible, mais plus complexe pour les débutant GitHub que vous pourriez être.
    b. Choisissez un nom de Solution « général » pour ne pas ressentir le besoin de la renommer si le projet que vous aviez choisi au départ ne pouvait être validé tel quel.

4.4.2 Architecture - UI

Identifiez les différentes parties sur lesquelles vous allez travailler.

- Posez-vous comme première question « de quoi ai-je **besoin** pour *gérer* mes XXX ».
- À partir de ces **besoins**, formulez des **fonctionnalités**. « Dans mon application je veux pouvoir *enregistrer* mes XXX ».
- À partir de ces fonctionnalités, imaginer les fenêtres et éléments graphiques dont vous avez besoin pour permettre ces fonctionnalités.
- Identifiez les librairies (donc les nouveaux projets) à créer qui sont assez indépendantes des interfaces utilisateur et qui pourraient être réutilisées dans d'autres applications manipulant les mêmes objets,

L'objectif est ici de découper votre projet en différentes parties isolées et performantes qui interagissent entre elles au service de l'application à développer.

Voici quelques suggestions concernant l'enregistrement des données :

Vous avez peut-être plusieurs classes qui doivent pouvoir s'enregistrer. Elles répondent donc toutes au même « pattern ». Comme vous le savez, en C#, cela peut se faire au travers de l'utilisation d'une interface commune qui pourrait être **IPersistent**, nouvelle interface qui identifie les objets qui peuvent s'enregistrer ou se charger parce qu'elle propose deux méthodes Save(…) et Load (…) qui reçoivent toutes deux le nom du fichier à charger ou à enregistrer.

D'autre part, tous les fichiers à enregistrer sont des fichiers au format XML ou binaire. Il est donc utile de créer également une classe générique contenant des méthodes static qui est capable de sérialiser des données d'un certain type.

4.5 Ressources

- Voir Moodle et forum du laboratoire pour infos et remarques supplémentaires
- Voir projets existants présentés durant le cours théorique sur GitHub https://github.com/HEPL-Moitroux-Programmation-CSharp-Base.