ProfilManager : Le ProfilManager sert à stocker les données d'un profil utilisateur.

Explications propriétés du ProfilManager :

La propriété ListingTris contient tous les types de Tris applicables sur chaque Collection d'œuvre dans la vue.

La propriété ListOeuvres est un ConcurrentObservableSortedDictionary ayant pour but de stocker pour chaque Genre (key) l'ensemble des œuvres stockées dans une ObservableCollection<Oeuvre> (value) ayant ce genre.

La propriété ListingDates est un ConcurrentObservableSortedDictionary ayant pour but de stocker pour chaque Genre (key) l'ensemble des tris de dates pouvant être appliqués stockés dans une ConcurrentObservableSortedSet<string> (value).

La propriété ListingDatesParGenre est une propriété calculée qui se base sur la propriété ListingDates pour un genre sélectionné par l'utilisateur.

Explications Méthodes complexes du ProfilManager :

- void CkeckListDates(Genre genre, string year) :

Lors de la suppression d'une œuvre, cette méthode vérifie si, pour un genre de l'œuvre, une autre œuvre possède la même date de sortie de l'œuvre supprimée. SI aucune œuvre ne possède la même date, la date est supprimée de la collection ListingDates avec comme clef le genre de l'œuvre passé en paramètre.

- int AjouterOeuvre(Œuvre œuvre) :

Dans un premier temps la méthode ajoute l'œuvre à la collection ListingSerie qui représente l'ensemble des œuvres tous genres confondus. Pour chaque genre de l'œuvre, l'œuvre est ajouté à la collection ListOeuvres et ListingDates pour le genre correspondant si elle n'y est pas déjà présente. La méthode retourne 0 si l'œuvre a bien été ajoutée, 1 si l'œuvre est déjà présente dans les données du profil utilisateur et -1 si l'utilisateur n'a pas renseigné tous les champs obligatoires.

- void SupprimerOeuvre(Œuvre œuvre) :

Dans un premier temps la méthode supprime l'œuvre à la collection ListingSerie qui représente l'ensemble des œuvres tous genres confondus. Pour chaque genre de l'œuvre, l'œuvre est supprimée de la collection ListOeuvres. La vérification dans la collection ListingDates est faite via la méthode CkeckListDates.

- void ChangeGenreSélectionné(ConcurrentObservableSortedDictionary<Genre, ObservableCollection<Oeuvre>> données, string textGenre) :

Permets de changer le genre sélectionné en cas de suppression de ce même genre. Cette méthode sélectionne un nouveau genre sélectionné en se basant sur l'index du genre voulant être supprimé et étant lui-même sélectionner. Cela permet d'éviter des exceptions de mémoires.

- Void SupprimerGenre(string nom) :

Permets de supprimer un genre du dictionnaire ListOeuvres entrainant la suppression de la liste d'œuvres associée à ce genre. Attention cependant cela n'entraine pas forcément la suppression des données des œuvres associées :

Cas n*1 : l'œuvre associée au genre supprimé n'est pas affectée à d'autres genres, dans ce cas elle n'apparaît plus dans les données de l'utilisateur.

Cas n*2 : l'œuvre associée au genre supprimé est affectée à minimum un autre genre existant et est donc toujours présente dans les données de l'utilisateur pour le ou les genres encore existants.

Œuvre : Classe abstraite ayant pour attributs les points communs des différentes œuvres de l'application. Il n'y a que Serie qui hérite d'Œuvre pour l'instant mais dans des futures mises à jour il pourrait y avoir des nouveaux types d'œuvres dans le modèle par exemple une classe Film qui hériterait aussi d'Œuvre.

ŒuvreWatch : Une OeuvreWatch est construite à partir d'une œuvre et d'une DateTime. Pourquoi une DateTime ? car dans une future mise à jour La WatchList possèdera plusieurs listing d'OeuvresVisionnées. Chaque listing de la WatchList mettra en tout début les ŒuvresWatch dont la DateTime est la plus récente.

Watchlist : Permets de stocker une collection OeuvresWatch ainsi que les méthodes de gestion de cette même collection d'OeuvresWatch.

MainManager : Le MainManager sert à stocker les profils utilisateur.

La persistance est injectée via le constructeur de MainManager permettant de choisir le type de persistance désiré.

Persistance :

IPersistanceManager : Cette interface nous permet d'avoir un type commun pour les persistances qui dériveront donc de celle-ci pour ensuite avoir différents choix de persistance. Le choix de la persistance se fait donc dans notre vue (App.xaml.cs) lors de l'appel du constructeur de MainManager.

DataContractPers : Cette classe nous permet de faire de la persistance en XML en sérialisant et désérialisant une ObservableCollection<ProfilManager>.

DataContractPersJSON : Cette classe nous permet de faire de la persistance en JSON en sérialisant et désérialisant une ObservableCollection<ProfilManager>. Elle hérite de DataContractPers et réécrit la méthode SauvegardeDonnées.

Stub : Classe qui nous permet de faire une fausse persistance dans le cas où l'utilisateur ne possède pas le fichier utilisé pour le chargement de la persistance (selon la persistance choisie)

Type des variables possédant une collection non-dit explicitement sur le diagramme des classes :

ProfilManager :

ObservableCollection<Oeuvre> ListRecherche

ObservableCollection<Oeuvre> ListOeuvresParGenre (C'est une propriété calculée)

ObservableCollection<Oeuvre> ListFiltrée

LinkedList<Serie> ListingSerie

Œuvre :

HashSet<Genre> TagsGenres

List<Auteur> ListAuteur

Watchlist :

ConcurrentObservableSortedSet<OeuvreWatch> OeuvresVisionnees

MainManager :

ObservableCollection<ProfilManager> ListProfils