Document d'aide pour la série de vidéos « EF Core Full stack MVVM »

Lien vers la liste

Ce document sert d'appuie pour les éléments importants à retenir de la série de vidéos mis en lien précédemment.

L'application qui est développé dans cette série est une application de transaction d'action à la bourse qui utilise l'architecture MVVM jumelé à Entity Framework Core.

L'objectif de l'écoute de cette série est de vous familiariser avec les différents concepts vus dans plusieurs cours, mais appliqué dans une application réelle.

Table des matières

Vidéo 1 – Introduction au domaine et installation

Sujet : Introduction au domaine et l'installation d'Entity Framework Core

- Diagramme de classes du domaine
- Liste des interfaces qui seront utilisées
- @3:07 : Utilisez Net Core 5.0
- @10:05 : Utilisez Net Core 5.0
- @10:45 : Prenez la dernière version soit 5.0
- @13:50 : Utilisez une base de données de type SQLite. Cela va faciliter le partage de l'application, car il suffira d'avoir un fichier SQLite dans le dossier du projet.
    - Installez l'extension **SQLite Compact Toolbox** via « Extension → Gérer les extensions… » et recherchez l'outil
    - Cet outil permet de visualiser le contenu d'un fichier SQLite
    - Créez une nouvelle BD SQLite nommée **SimpleTrader.db** (Voir Annexe – Créer un BD SQLite) dans le dossier du projet des classes de domaine
- @15:05 : Utilisez Net Core 5.0
- @15:25 : Outils → Nuget Package Manager → Package Manager Console
- @16:25 : **OnModelCreating** permet de configurer les entités indépendamment pour des cas d'exception. Par exemple, par défaut EF nécessite un champ ID. Dans le cas de *Stock* où il n'y a pas de nécessité clé **ID** et dont l'objet ne se retrouve que dans la classe *AssetTransaction*, on veut indiquer qu'*AssetTransaction* ne possède qu'un seul *Stock*. La relation est qu'un *Stock* peut être dans plusieurs transactions, mais que la transaction n'implique qu'un seul *Stock*.
- @18:50 : Utilisez **SQLite Compact Toolbox** pour visualiser le contenu de la BD

Résumé

Mot clé	**Description**
Classe du domaine	Classe en lien avec le domaine d'application du logiciel
DbContext	Gère les interactions entre la base de données et Entity Framework.
DbSet	Sert à indiquer quels sont les entités qui sont emmagasinées dans la BD.
Chaîne de connexion	String qui sert à indiquer les informations de connexion à la source de données tels que le mot de passe, l'endroit de la BD, le format, etc.
PM Console	CLI pour effectuer des commandes de migration et de mise à jour de la structure de la BD selon les modèles définis dans le DBContext
add-migration	Commande pour créer le code de migration pour la BD
update-database	Commande pour modifier la BD avec le code de migration

Vidéo 02 - Refactorisation

Sujet : Refactorisation du projet en utilisant une **Fabrique** (*Factory*) pour créer le DBContext du Projet

- Ne pas oublier d'adapter le code pour SQLite
- @16:00 : EntityEntry<T> est une classe fournit par EF qui permet de suivre le changement d'un objet en particulier.

Résumé

Élément clé	Description
Type générique avec contrainte	Il est possible d'utiliser des contraintes pour les types génériques avec le mot clé **where** (où). Exemple : `IDataService<T> where T : DomainObject`
Héritage pour avoir des champs communs	Utiliser une super classe (héritage) pour avoir des champs communs.
Thread safe	Concept indiquant qu'une ressource mémoire peut être accédée par plusieurs threads au même moment.
DbContext n'est pas *Thread Safe*	Si plusieurs threads tentent d'effectuer des opérations sur le DbContext, il y a des chances d'avoir des exceptions liées aux threads. L'utilisation d'une fabrique dans le cas du projet permet de régler ce problème.
EntityEntry<T>	**EntityEntry<T>** est une classe fournit par EF qui permet de suivre le changement d'un objet en particulier. La propriété **Entity** représente l'objet qui est suivi.

Vidéo 3 - Préparer le projet WPF

Sujet : Préparer le projet WPF

Lien

- @2:15 : Dans le App.cs.xaml, on retire le startup URI pour pouvoir éventuellement effectuer de l'injection de dépendance.
- @17:05 : Il fait une injection de dépendance.
- @21:30 : Pour accéder à un type public, on doit importer le *namespace*, ensuite on peut accéder au type via `{x:Static ns:type}`
- @23:30 : Il est possible que dans le constructeur de MainView qu'il y a déjà une ligne dans laquelle on configure le MainViewModel au contexte de données. Vous pouvez supprimer celle-ci si c'est le cas.
-

Résumé

Élément clé	Description
StartupURI dans App.xaml	StartupURI indique la fenêtre de démarrage par défaut. Toutefois, s'il est utilisé, il ne sera pas possible d'effectuer des manipulations dans la fenêtre avant son affichage.
ResourceDictionary	Un ResourceDictionary permet de partager des ressources tel que la couleur, des formats, etc. à travers les différentes entités de l'application.
Fusion de dictionnaire	À l'instar du CSS, il est possible d'ajouter plusieurs fichiers de ressource dans l'application. On peut utiliser la balise `<ResourceDictionary.MergedDictionaries>` pour fusionner des ressources dans une application.

Vidéo 4 – Appels API

Sujet : Appels API et chargement asynchrone de ViewModel

Lien

Mise à jour importante

- Créer un compte sur ce site pour obtenir une clé API. Ce site permet de récupérer la valeur des actions.
    - Pour récupérer la clé API.
- **Important**! Vidéo pour utiliser « Financial Modeling Prep ». N'écouter que la première partie.
- Le Endpoint a été mis à jour sur l'API. L'url équivalent est maintenant celui-ci : https://financialmodelingprep.com/api/v3/quote/^DJI?apikey=VOTRE_CLE
- Dans **MajorIndex**, la propriété **Changes** doit être **Change**
- De plus, le code est maintenant celui-ci pour convertir le JSON reçu

```
List<MajorIndex> majorIndices =
JsonConvert.DeserializeObject<List<MajorIndex>>(jsonResponse);

return majorIndices[0];
```

Résumé

Élément clé	Description
ContinueWith	Permet d'ajouter du code asynchrone qui continue après que la tâche s'est exécutée.

Vidéo 5 – Ajout de styles sur les contrôles

Sujet : Ajout de styles sur les contrôles

Lien

Résumé

Élément clé	Description
Style	On peut utiliser plusieurs styles sur les contrôles.
Événement	On peut configurer les événements dans le XAML à l'aide, entre autres, du concept de StoryBoard.

Vidéo 6 – Ajout de cartes pour l'affichage

Sujet : Ajout de cartes pour l'affichage des indices boursiers

Lien

Résumé

Élément clé	Description
WPF - StringFormat	Dans un contrôle ayant du texte, on peut utiliser l'attribut StringFormat pour indiquer le formatage. Par exemple pour des valeurs monétaires : `StringFormat={}{0:c}`
WPF - FallbackValue	S'il n'y a aucune valeur, l'interface peut se rabattre sur la valeur inscrite à FallbackValue

Convertisseur	Un converter est une classe qui implémente l'interface IValeurConverter.
	Cela permet de convertir n'importe quel objet vers une autre objet.
	Pour intégrer un converter dans un contrôle, il faudra utiliser les attribut ConverterXY dans le Binding.
	
	Exemple
	`{Binding CurrentViewModel,`
	`Mode=OneWay,`
	`Converter={StaticResource EqualValueToParameterConverter},`
	`ConverterParameter={x:Type vm:PortfolioViewModel}}`

Vidéo 7 – Obtenir le prix d'une action en particulier

Sujet : Obtenir le prix d'une action via l'appel API

Lien

Documentation : https://financialmodelingprep.com/developer/docs#Stock-Price

Élément clé	Description
Exception personnalisée	Il est possible de créer une exception personnalisée en créant une nouvelle classe qui hérite d'Exception

Vidéo 8 – IBuyStockService, AccountDataService

Sujet : Création d'un service d'achat d'action et refactorisation.

Lien

Élément clé	Description

EF Navigation Property	Une propriété d'une entité qui fait référence à une autre entité.
	
	Par exemple :
	• **Collection navigation entity** : une entité qui possède une collection d'une seconde entité.
	• **Reference navigation entity** : une entité qui possède une référence vers une seconde entité.
	
	Exemples concrets :
	• Account
	  ○ ICollection<AssetTransaction> AssetTransactions
	• AssetTransaction
	  ○ Account
	• Facture
	  ○ ICollection<LigneFacture> Lignes
	• LigneFacture
	  ○ Facture
	  ○ Produit

BD les jointures	Les jointures dans une base de données ralentissent les requêtes.
Non query	Terme utilisé pour désigner les requêtes qui ne retournent pas de valeur. Par exemple : Create, Update et Delete

Vidéo 9 – Configuration de l'injection de dépendance

Sujet : Implémentation d'un système d'injection de dépendance

Lien

Une des raisons pour l'implémentation du système d'injection de dépendance, c'est que le projet commence à avoir beaucoup de services à gérer.

Ce système nous permettra de récupérer les services nécessaires avec leur dépendance.

Ce système nous permettra d'avoir un seul endroit pour définir les dépendances.

Résumé

À la suite du visionnement de cette vidéo, on se rend vite compte que l'injection de dépendances permet de découpler beaucoup les dépendances entre les classes.

Élément clé	Description
IServiceProvider	Permet de générer les services nécessaires à l'application. La méthode AddSingleton permet d'instancier un seul service pour l'ensemble de l'application. La méthode AddScoped permet de créer une instance pour chaque service, mais ce sera la même instance pour le même service. La méthode AddTransient (pas dans la vidéo) permet de créer une nouvelle instance pour chaque client et chaque service. La méthode GetRequiredService<T> retourne le service requis et déclenche une erreur si le service est inexistant. La méthode GetService<T> retourne le service requis et retourne null si le service est inexistant.

Vidéo 10 – BuyViewModel et BuyView

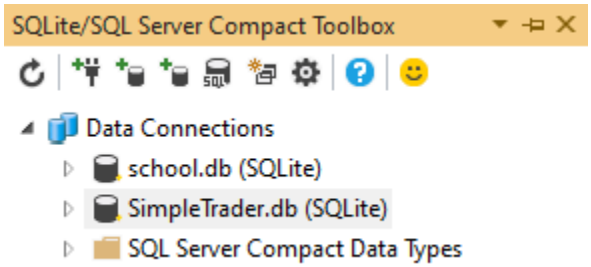
Sujet : Implémentation visuelle pour pouvoir acheter des actions

Lien

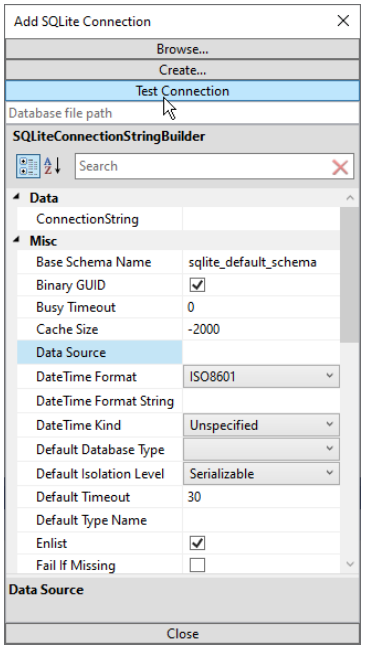
- @16:05 : La raison pour laquelle on n'aime pas faire des « async void » est qu'il ne sera pas possible de capturer l'erreur s'il y en a une.

Annexe – Créer un BD SQLite

Avec l'outil SQLite/SQL Server Compact Toolbox.



Cliquer sur le bouton « Add SQLite Connection » (3e bouton)



À partir de la nouvelle fenêtre, on peut récupérer une bd existante ou encore en créer une nouvelle.

Ensuite, on peut tester la connexion avec le bouton réservé à cet effet.

La chaîne de connexion pour une bd SQLite est simplement la propriété « Data Source » avec le chemin vers de fichier.

Exemple : `@"Data Source=Data\SimpleTrader.db"`

Important : Il faut s'assurer que le fichier soit copié dans le dossier de compilation.