

Notes sur le module : *LInQ et Entity*.

Armel Pitelet

11 janvier 2022

Table des matières

1 LInQ	1
2 Entity Framework	2
3 Divers	3

Dans le folder Cours : ce qui a été fait et projeté, Exercice : ce que j'ai pris au passage.

1 LInQ

Mot clef : **LInQ**.

Language intergrated Query : Façon d'interroger une collection comme

- Set (liste qui garantie l'unicité les valeurs) : fonctionne via itérateur et pas indexeur. Très performant pour des petites collections, moins pour les grosses car il faut parcourir toute la collection.
- ICollection ou IList : la liste classique (ICollection est plus haut dans la chaine d'interface). IList : list avec accesseur numérique (indexeur) et des méthodes supplémentaire par rapport à ICollection.
- IDictionary : système clef-valeur. Clef unique. Plus rapide que l'accès via une liste (ex d'un dictionnaire de int, string) mais plus complexe à gérer. On s'en sert plutôt quand la clef est complexe (string ou objet). Utilisé avec des clefs int si la gestion de l'int n'est pas accessible (et donc pas connu).

Voir aussi pour la liste des méthodes LinQ : <https://docs.microsoft.com/fr-fr/dotnet/api/system.linq.enumerable?view=net-6.0>

Remarque : *Sur les types*.

Il faut connaître la correspondance de type en c# et SQL : **UserFakeRepositoryTest**. Le type doit être choisi en fonction de l'aplace que l'on veut allouer en mémoire (en octets).

Remarque : *sur text et ntext*.

Il vaut mieux éviter de s'en servir car ils ne sont pas limité en taille.

Pour la suite on va lister ce qu'il y a dans l'exercice LinqEntityFramework :

Remarque : *Fake repository*.

Donne des données pour faire du LinQ dessus (repository sera un objet qui renverra des données à partir d'une base).

Note :

Les méthodes Linq Prennent des fonctions anonymes en paramètre. => indique la déclaration d'une fonction. Ces fonction recoive comme paramètre l'élément courant (les contenues du conte-neurs)

Remarque : First et FirstOrDefault.

First renvoie l'élément et une erreur si il ne le trouve pas. FirstOrDefault renvoie l'élément ou null si il ne trouve pas

Remarque : Single et SingleOrDefault.

// Single renvoie l'élément et une erreur si il ne le trouve pas ou si il y en a plusieurs. SingleOr-Default renvoie l'élément ou null si il ne trouve pas ou bien une exception si il en trouve plusieurs

2 Entity Framework

ORM : Mapping Objet relationnel (*object-relational mapping*); Voir : https://fr.wikipedia.org/wiki/Mapping_objet-relationnel. Interface entre la ddb et le code. Donne accès à des listes en base de donnée sur lesquel en va pouvoir venir faire des requêtes.

Note :

Entity Framework ne gère pas les uint.

Dans poec.sql.repository (SqlDbContext). On a besoin des package Microsoft.EntityFrameworkCore et Microsoft.EntityFrameworkCore.SqlServer

Remarque : Lazy Loading Enabled.

permet de faire eleve.classe, ou classe.eleve, pour arriver à eleve.classe.eleve....A creuser...

Remarque : AutoDetectChangesEnabled.

Sauvegarde automatique des changement, peut être dangeureux car on peut enregistrer des choses que l'on ne veut pas. On préfère le apsser à false pour maitriser la donnée de bout en bout.

Remarque : DbSet.

Exécute une requete uniquement au moment ou l'on demande de récupérer un élément. Renvoie la plupart du temps une image de la base de donnée et ne fera la requete qu'au dernier moment. C'est une projection de la base de donnée.

Remarque : DbContext.

Objet qui fait le lien avec la base de donnée.

Remarque : Sur le mapping.

On peut le faire coté SqlDto (et un peu du DbContext) via des attribut décorateur ou bien direc-tement dans le DbContext (voir DbContext.cs). Il faut faire soit l'un soit l'autre.

Mot clef : SaveChanges.

Permet de valider la transaction [qu'on appelle commit](tant que l'on ne fait pas le save change, le DbContext n'applique pas les modification à la BDD) → Principe de transaction (possible aussi en SQL pure). En cas d'erreur le DbContext fait un roll back depuis le dernier SaveChanges. SaveChange retourne un int correspondant au nombre de ligne affectée par les changement. Cela permet de tes-ter la réussite ou non d'une opération Remove par exemple.

Remarque : *Sur l'autoincrément dans la bdd.*

Dans une BDD, l'autoincrément ne prend pas la dernière ligne mais le dernier élément inséré. Après une suppression la base garde quand même en mémoire l'id de l'élément supprimé. Un ajout après cela aura donc un incrément de +1 par rapport au dernier supprimé et non pas par rapport à la dernière ligne actuelle.

Remarque : *Sur les Add et Update.*

La bonne pratique est de renvoyer l'objet ajouter/modifier pour tester la nullité de l'objet ou non (et potentiellement vérifier son identité).

Note :

SqlContext.Where() retourne un IQueryable qui a le comportement donnée dans la remarque suivante.

Remarque : *Sur les performance.*

SqlContext.Set<UserSqlDto>().Where(), projette une requete mais ne l'exécute pas immédiatement. Il faut ensuite faire un .First, .List, ... pour que la requete soit exécuté. Cela permet de ne pas faire de requet inutile à chaque Set ou Where et de ne les exécuter qu'au besoin

3 Divers

Note :

Resharper (jetBrains) : analyse de code, changement dynamique, code helper,... Attention Payant.

Remarque : *Sur les région.*

Se déclare avec des #region #endregion (directive **prépocesseur**).

Remarque : *Sur les CRUD.*

Il faut essayer un maximum de faire les CRUD avec des types primitifs (pour rester le plus générique possible → notion de *scalabilité* du code).

Remarque : *Pattern AAA.*

Arrange, Action, Assert : utiliser pour du test. Evite de recopier du code plusieurs fois. Attention, On peut avoir autant d'arrange et assert que l'on veut mais une seule action (car test unitaire!!!).

Remarque : *Bonne pratique.*

Il vaut mieux renvoyer des interfaces si possible (IList au lieu de List).

Note :

On peut utiliser default pour forcer une valeur par défaut sur un des paramètres passés lors de l'appel d'une méthode, d'une instantiation ou autre.

Acronymes

DSP Nom simple. , *Glossaire* : [DSP](#)

Glossaire

DSP la description détaillé.