Secteur Tertiaire Informatique

Filière « Etude et développement »

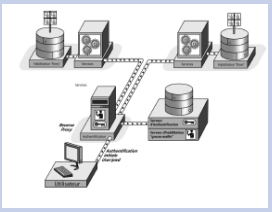
Construire une application organisée en couches en mettant en œuvre des Frameworks

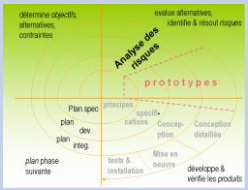
Mettre en œuvre un Framework de persistance
Manipuler les objets

Apprentissage

Mise en pratique

Evaluation





Version	Date	Auteur(s)	Action(s)
1.0	20/01/20	Vincent Bost	Création du document

TABLE DES MATIERES

Préambule

Ce document propose une série d'exercices autour de la mise en place d'un Framework de persistance.

Pour réaliser ces exercices, il vous faut disposer :

- De l'environnement de développement intégré Visual Studio version 2019 ou ultérieure
- D'Entity Framework Core 3.1 ou ultérieure.

Avant d'aborder les exercices proposés dans ce document vous devez au préalable avoir réalisé ceux relatifs à la mise en place du mappage objet / relationnel.

Objectifs

A l'issue de la réalisation de ces exercices, vous serez capable de mettre en place les mécanismes visant à assurer la persistance des entités de votre domaine.

Assurer la persistance des objets au sein de la base de données nécessite de mettre en œuvre deux scénarios de gestion de la persistance :

- Le mode connecté
- Le mode déconnecté

Méthodologie

Vous avez assimilé lors de la précédente phase d'apprentissage la notion d'ORM et avez pu élaborer le mappage de vos types d'entités aux tables de la base de données.

Nous allons aborder maintenant la manipulation des objets issus de la couche de persistance.

A partir du résultat obtenu lors de la précédente étape d'apprentissage, vous réaliserez les exercices proposés à l'issue de la lecture de chaque chapitre comportant une mise en pratique des techniques exposées.

Vous pourrez ainsi, par la pratique, pas à pas, atteindre les objectifs de cette unité d'apprentissage consacrée à la mise en œuvre de la persistance des entités métier au sein d'un SGDBR.

Confrontez vos travaux à ceux de vos pairs et essayez de résoudre vos difficultés par l'échange avec ceux-ci ou votre formateur avant de prendre connaissance de la solution lorsque celle-ci est mise à votre disposition.

1. GESTION DE LA PERSISTANCE EN MODE CONNECTE

1.1 PREPAREZ LE CONTEXTE DE DONNEES

Créer une base de données AFPA_2020.

Exécutez le script SQL mis à votre disposition pour

Générez les modèles à partir de la base de données.

Le modèle d'entités ayant été généré à partir d'une base de données existante, le contexte de données expose les jeux d'entités qui correspondent aux tables de la base. Il doit toutefois être amendé pour exposer les jeux d'entités des types spécialisés issus de mécanismes d'héritage. Dans EF Core, seul le modèle TPH (Table par Hiérachie) est implémenté. Il faut disposer dans la table d'une colonne qui agit comme discriminant de type. Ici, deux tables :

- ProduitFormation avec TypeFormation / discriminant
- Personne avec CatPersonne

1. Complétez le complément de la définition du contexte d'entités (classe partielle AFPA_ORMEntities)
2. Complétez les définitions existantes pour obtenir un mécanisme permettant d'extraire les entités :
    a. Personnes
    b. Stagiaires
    c. Collaborateurs AFPA
    d. Produits Formation Qualifiante
    e. Produits Formation Continue

1.2 VERIFIEZ LA CONFIGURATION DU CONTEXTE DE DONNEES

Il est essentiel de pouvoir analyser le code SQL généré à partir des expressions de requête, notamment pour identifier les causes des problèmes de performances.

Créer une méthode d'analyse des requêtes comme suggéré dans le point 2.4 de votre support de cours.

Vérifiez que votre contexte de données soit configuré correctement en réalisant les requêtes suivantes avec l'opérateur de conversion ToList :

1. Liste des stagiaires
2. Liste des collaborateurs AFPA
3. Liste des personnes
4. Liste des produits de formation qualifiante
5. Liste des produits de formation continue
6. Liste des produits de formation

Prenez le temps d'analyser avec attention les requêtes SQL générées.

1.3 GEREZ LES ENTITES EN MODE CONNECTE

Vous allez ajouter de nouvelles entités dans la base de données.

Nous bénéficierons ici d'un suivi automatique des modifications.

Décorez vos classes modèles pour prendre en charge le contrôle de validité des données.

Voir le dernier chapitre de ce document 2.4 Compléter la définition de nos entités

Créez une méthode statique par exercice afin de faciliter les phases de mise au point et de correction.

Définissez l'instance de votre contexte de données dans la portée de la méthode pour éviter des effets de bord indésirables. Traitez des erreurs de validation.

Introduisez quelques données ne passant pas les contrôles de validité mis en place au niveau de l'entité Stagiaire par exemple. Vous constaterez aussi que les règles définies au niveau de votre couche modèle sont prises en compte dans les mécanismes de validation et que les entités en erreur ne feront pas l'objet de sauvegarde en base de données.

1.3.1 Ajouts

Le contexte de données est conservé tout au long du processus d'ajout. Faites un seul appel à la méthode de sauvegarde de l'état de vos entités.

1. Ajoutez un nouveau produit de formation CDA. Voir le code sur feuille de présence.
2. Ajoutez une nouvelle offre de formation associée au produit de formation CDA. Reprenez les infos définies sur votre feuille de présence. Cette offre de formation est organisée dans l'établissement de Brive.
3. Ajoutez les références de votre formateur à l'offre.
4. Ajoutez quelques stagiaires à cette offre de formation, de préférence parmi ceux qui sont inscrits dans votre parcours afin d'avoir un jeu d'essai réaliste. Ces stagiaires démarreront la formation au premier jour de celle-ci.

Vous noterez l'intérêt de pouvoir utiliser alternativement, en fonction du contexte, l'identifiant de la propriété de navigation (IdProduitFormation) ou la référence de l'objet (Etablissement, Collaborateur,..).

Sauvegardez l'ensemble.

1.3.2 Modifications

Vous allez ici mettre à jour les informations nouvellement ajoutées.

1. Une erreur s'est glissée lors de l'ajout de l'offre de formation : la date de démarrage de la formation est erronée. La formation doit démarrer le jour suivant.
   Cette modification doit être aussi reportée sur l'entité d'association entre stagiaire et offre de formation.
2. Vous devez modifier la date de naissance du premier stagiaire : son jour de naissance est erroné. Il est né le premier jour du mois.
3. Vous devrez enfin ajouter un nouveau stagiaire qui a bénéficié d'une réduction de parcours et démarrera donc sa formation 2 semaines après le démarrage de l'offre.

Vous devez charger l'ensemble des éléments à prendre en considération pour modification en une seule fois.
L'ensemble des modifications doit être considéré comme une seule transaction.

2. GESTION DE LA PERSISTANCE EN MODE DECONNECTE

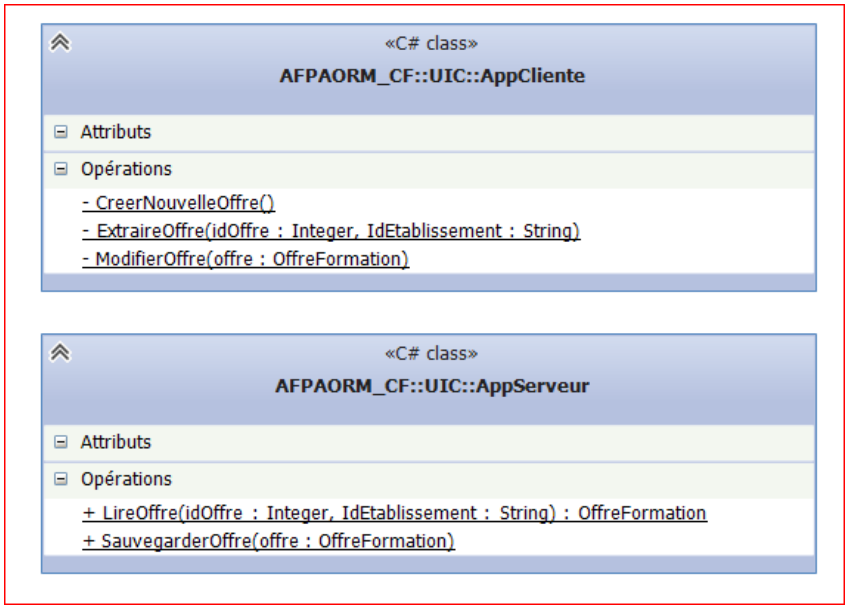
Vous allez aborder ici un scénario plus complexe mais de nature à satisfaire les exigences d'une application en couches où les objets métiers peuvent être manipulés par des services, applications distantes ou déconnectées. Je vous propose ici de reconstituer, au travers d'une simple application console, des contraintes environnementales similaires.

Vous devez avoir pris connaissance des techniques de gestion de la persistance en mode déconnecté de votre support de couts.

Le principe : les données sont extraites à partir de la base de données.
Ces entités sont ensuite manipulées et modifiées par l'application sans pour autant être attachées au contexte de données.

Elles ont ensuite été ajoutées à un nouveau contexte de données pour permettre leur sauvegarde en base de données. Lors de l'ajout au contexte de données, il convient de préciser l'état de l'entité pour déterminer quelle opération exécuter sur le SGBDR.

Vous allez ici créer une nouvelle offre de formation avec de nouveaux stagiaires
Modifier une offre existante.
Testez différents scénarios.
Il est important de conserver ici la notion d'unité de travail et de s'assurer que l'ensemble des opérations soient traitées comme une seule transaction.
Je vous suggère de créer deux classes :



2.1 FAITES EVOLUER VOS MODELES D'ENTITE POCO

Vous devez disposer dans chaque entité d'une propriété qui permettra de déterminer l'état de l'entité présente dans le graphe afin de déterminer l'opération SQL à exécuter. Consultez l'exemple du chapitre 4-4-2-3 page 20 de votre support de cours.

Déclarez un type énuméré EntityPocoState au niveau de l'assembly contenant vos modèles dont les valeurs seront celles du type énuméré EntityState d'Entity Framework. En procédant de la sorte, nous pouvons obtenir une correspondance directe entre les deux attributs.

Implémenter une propriété **Etat** dans chaque entité susceptible de faire l'objet d'une sauvegarde en base de données. Les entités dérivées obtiendront cette propriété du type de base. Afin d'avoir un code plus rigoureux et mieux structuré, recourez à une interface.

2.2 FAITES EVOLUER VOTRE CONTEXTE DE DONNEES

Pas de suivi automatique des modifications dans ce contexte. Modifiez les options du contexte de données pour éviter le suivi des modifications.

Afin de vous assurer que ces choix s'appliquent systématiquement pour tout usage du contexte de données, configurez-les dans le constructeur.

Faites évoluer le mappage de vos entités. La propriété Etat ne doit pas persister. Sans cette information, le système chercherait à sauvegarder l'info et vous obtiendriez une erreur liée à une différence de schéma entre Db et Entité.

2.3 PROGRAMMEZ VOS COMPOSANTS D'INTERFACE

Développez tout d'abord les méthodes du scénario d'Ajout.

Développez ensuite les méthodes du scénario de Modification.

2.4 COMPLETER LA DEFINITION DE NOS ENTITES

Lorsque les modèles sont générés à partir de la base de données, il est utile d'observer quelques principes pour préserver la définition de nos classes métier (modèle) de toute destruction non intentionnelle.

Les classes d'entité étant générées à partir du modèle, il faut éviter de modifier le code source des fichiers générés : toute modification apportée manuellement à ce code sera perdue lors d'une régénération du modèle.

Les amendements relatifs aux classes métier devront donc être implémentés dans des fichiers sources différents aux travers de :

- Classes partielles pour les attributs non mappés et les méthodes.
- Classes de métadonnées pour compléter la définition des attributs.

Nous devons disposer des composants de la bibliothèque System.ComponentModel.DataAnnotations.

Exemple pour une classe Stagiaire.

Implémentation de mécanismes permettant de déterminer si deux objets stagiaires représentent le même stagiaire : surcharge de Equals et GetHashCode (possibilité d'implémenter l'interface IEqualityComparer).

Implémentation d'une propriété Age en lecture seule

Implémentation d'une règle de validité de la valeur de l'attribut Matricule

```
[MetadataType(typeof(StagiaireMetaData))]
7 références
public partial class Stagiaire
{
    0 références
    public int Age
    {
        get
        {
            if (this.DateNaissanceStagiaire == DateTime.MinValue) { return 0; }
            if (DateTime.Now.DayOfYear >= this.DateNaissanceStagiaire.DayOfYear)
                return DateTime.Now.Year - this.DateNaissanceStagiaire.Year + 1;
            else
                return DateTime.Now.Year - this.DateNaissanceStagiaire.Year;
        }
    }
    0 références
    public override bool Equals(object obj)
    {
        Stagiaire stagiaire = obj as Stagiaire;
        return (stagiaire == null ? false : stagiaire.MatriculeStagiaire == this.MatriculeStagiaire);
    }
    - références
    public override int GetHashCode()
    {
        return (this.MatriculeStagiaire == null ? 0 : this.MatriculeStagiaire.GetHashCode());
    }
}
```

Classe de métadonnées

```
public sealed class StagiaireMetaData
{
    [Display(Name = "Matricule stagiaire")]
    [StringLength(08, ErrorMessage = "Longueur invalide")]
    [Required(ErrorMessage = "Le matricule est requis")]
    [RegularExpression("^[0-9]{8}$", ErrorMessage = "Ne doit comporter que des chiffres")]
    public string MatriculeStagiaire;
}
```

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Vincent Bost - formateur

Date de mise à jour : 20/01/20