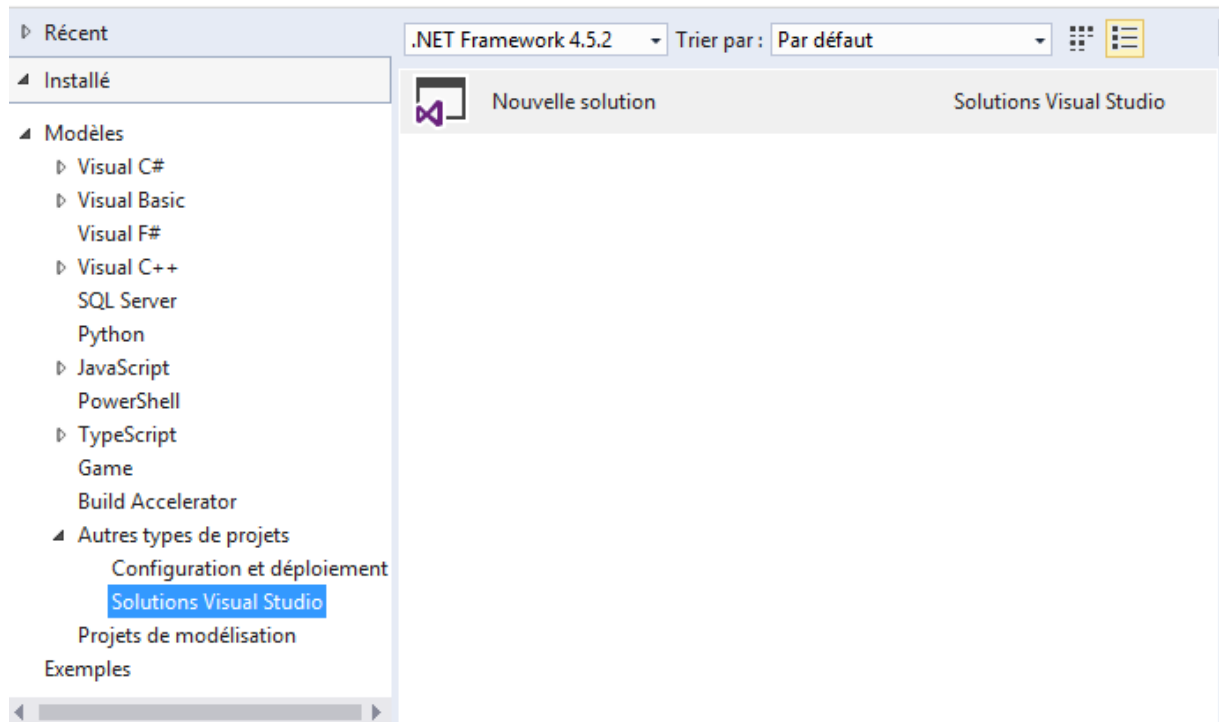


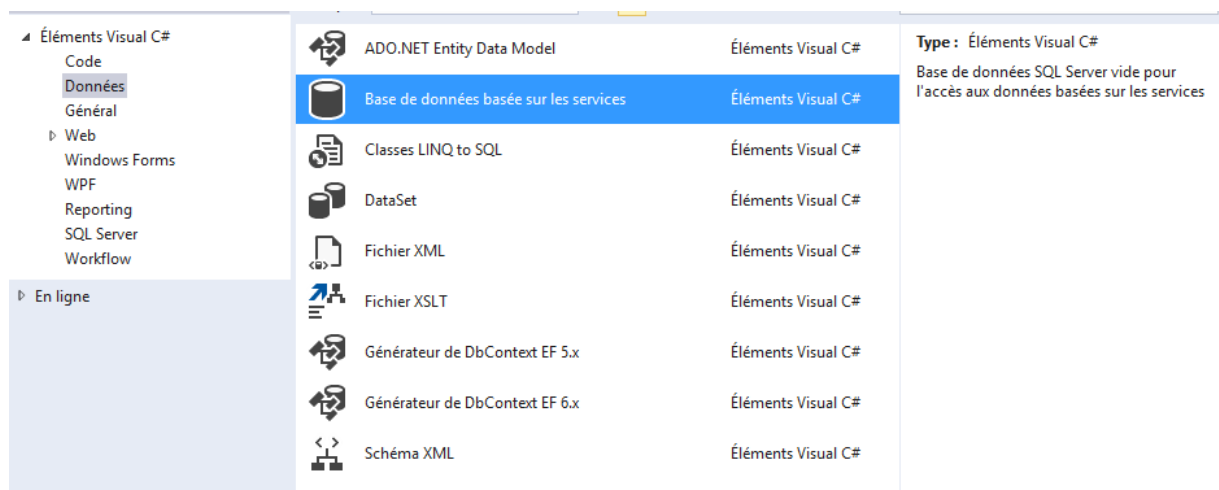
TP : 1er Jalon

J0 : Création de la solution

- 1) Créer un projet de type solution sous Visual Studio. Le nommage de ce projet devra être le suivant : NomProjet.NomBinome1.NomBinome2



- 2) Ajouter les projets liés au jalon 1 (vous ajouterez le reste des projets au fur et à mesure) c'est-à-dire au minimum une bibliothèque de classe et un projet de test unitaire (il est conseillé d'ajouter un projet console pour commencer)
- 3) Ajouter la base de données au format MDF dans votre bibliothèque de classe.



NB : Le Framework ciblé est 4.0 ou 4.5

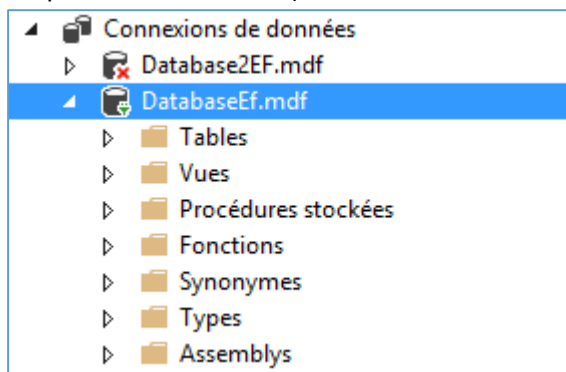
J1 : Création de la couche d'accès aux données, tests unitaires

Objectifs de ce jalon :

- Installer Entity Framework 6 dans les projets nécessaires.
- Créer un répertoire pour stocker vos entités. L'idéal est de séparer la base de données (fichier mdf), les entités et le mapping (pour ceux qui utiliseront Fluent).
- Créer les entités nécessaires conformément au modèle de données fourni par Monsieur X (page 2) pour créer votre modèle.
- Réaliser le mapping entre les entités et la base de données. Ce mapping pourra se faire de deux façons (vous choisirez la méthode qui vous convient le mieux) :
 - En utilisant les Data Annotation.
 - En utilisant l'API Fluent **(conseillé)**.
- Créer un contexte Entity Framework pour avoir accès à votre base de données.
- Tester unitairement chaque entité de votre modèle en réalisant les opérations suivantes :
 - Un ajout.
 - Une modification.
 - Une suppression.
 - Une liste.

Quelques conseils :

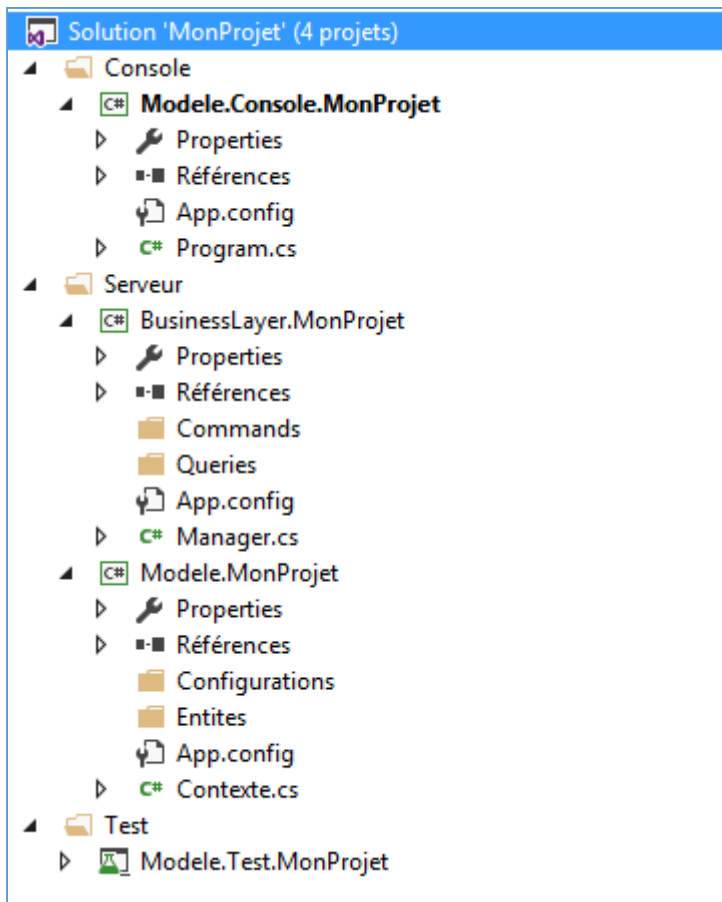
- L'installation d'EF est beaucoup plus facile en passant par le gestionnaire de package Nuget (voir le cours).
- Créer un projet de type Console pour tester au fur et à mesure votre mapping. Vous pouvez ajouter des données manuellement dans votre base de données pour vos premiers tests (via l'explorateur de serveur).



- Suivre le lien fourni en cours pour accéder à la documentation complète d'Entity Framework.
- Pour les tests unitaires, vous pouvez créer une seconde base (pour ne pas polluer la première qui vous servira tout au long de l'application).
- Des exemples utilisant Entity Framework sont disponibles dans le cours.
- Une classe par fichier .cs si possible.
- Pour ceux qui utilisent l'API Fluent, vous pouvez ajouter toutes vos classes de mapping (en une seule ligne) en utilisant les Assembly.

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);
    modelBuilder.HasDefaultSchema("dbo");
    modelBuilder.Configurations.AddFromAssembly(Assembly.GetExecutingAssembly());
}
```

Exemple d'architecture souhaitée :



NB : Veuillez à revoir les noms des projets pour que le tout soit cohérent.

Sorties du jalon :

- Un modèle commenté relié à une base de données.
- Le mapping doit permettre d'instancier un contexte Entity Framework valide.
- Une série de test unitaire que l'on peut exécuter à tout moment du projet. Les tests doivent bien évidemment être « success ».