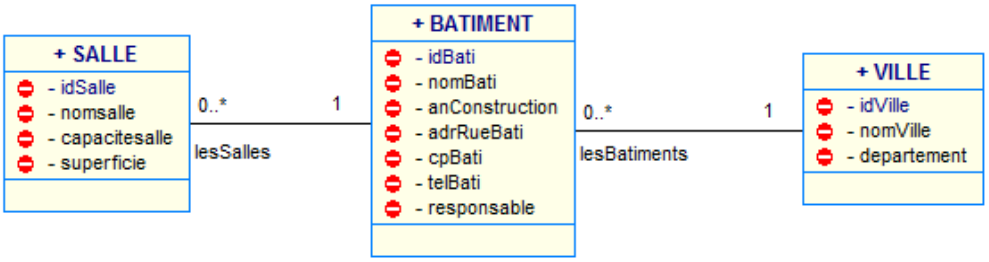
Objectifs :

- Découvrir l'ORM Entity Framework
- Créer sous Entity Designer Model un modèle objet et générer la base de données
- Procéder au mapping objet-relationnel avec l'ORM Entity Framework

1. Le modèle objet Entity Framework

1.1 C'est quoi un ORM ?

Si on avait demandé à WinDesign de produire le diagramme des classes de notre application MusicAtout, permettant la gestion des villes, des bâtiments et des salles, on aurait obtenu le diagramme suivant :



Le diagramme des classes est généré en vue de la programmation des objets, instanciés des classes.

Du point de vue de la programmation ce sont des objets que nous manipulons.

==> Avec C#, le diagramme des classes sera bien utile !

Est-il possible d'associer les tables de la base de données MusicAtout.mdf avec le diagramme des classes tel qu'on l'utilisera sous Visual Studio pour manipuler les données avec une application C# ? **Oui !**

C'est le principe de **l'ORM** : l'Object-Relational Mapping

> **L'ORM est un outil qui permet de générer une couche d'accès aux données des tables d'une base de données relationnelle, depuis un modèle objet et des classes décrites sous Visual Studio.**

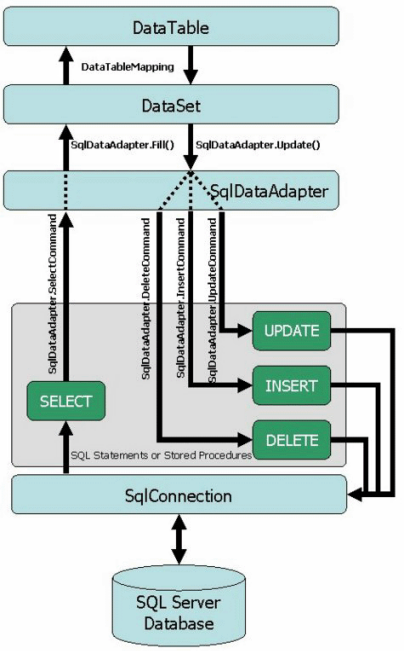
Des correspondances sont définies entre la structure de la base de données relationnelle et les objets instanciés et manipulés dans nos programmes C#.

La table VILLE **est mappée avec** **la classe VILLE**

Entity Framework est un ORM : il permet d'interfacer les données de la base aux objets manipulés par programme.

Un **mapping objet-relationnel** (**Object-Relational Mapping** ou **ORM**) est une technique de programmation qui crée l'illusion d'une base de données orientée objets à partir d'une base de données relationnelle en définissant des correspondances entre cette base de données et les objets du langage utilisé. On pourrait le désigner par « **correspondance entre monde objet et monde relationnel** »

1.2 L'ORM « Entity Framework »



Entity Framework est l'ORM du framework .Net de Microsoft. Il est totalement intégré à l'IDE Visual Studio. On peut donc l'utiliser avec le langage C#.

Entity Framework permet :

- de modéliser les données
- de gérer la base de données correspondante
- de générer un modèle objet à partir d'une base de données existante
- de gérer tous les accès à la base de données : lecture, écrire, ajout, suppression, ...

Un élément très important du concept est la **chaîne de connexion** qui permet d'associer le modèle objets à la base qui contient les données.

2. Le modèle objet « MusicAtout »

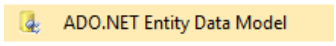
On lance donc maintenant Visual Studio 2010.
On va maintenant **créer** un **nouvel objet** qui permettra de connecter l'application C# à la base de données MusicAtout.mdf.

2.1 Création du modèle objet avec EDM : « Entity Designer Model »

On crée une nouvelle application en mode formulaires de nom « **MusicAtoutV01** »

Attention à bien sélectionner le dossier pour votre application !

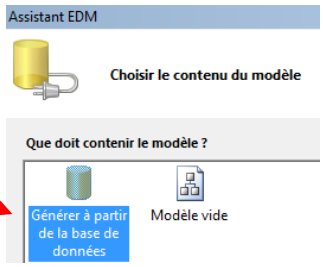
Ensuite **Ajouter** un **nouveau composant** de type **ADO.Net Entity Data Model**



Le nommer **mdlMusicAtout.Edmx**

ADO.Net, comme Access Data Object, est la nouvelle bibliothèque logicielle d'accès aux données fournie en standard dans le Framework .Net. C'est un ensemble de classes, de structures, de types gérant l'accès à des sources de données. La connexion à une source de données s'effectue par le biais d'un fournisseur d'accès.

EDMx comme : Entity Model Designer qui permet de modéliser les objets



On choisit "**Générer à partir d'une base de données**", c'est à dire qu'on veut que la connexion s'établisse entre l'application et **musicAtout.mdf**

Une fenêtre s'ouvre qui permet de créer la chaîne de connexion.

Nouvelle connexion

Source de données **Microsoft SQL Server**

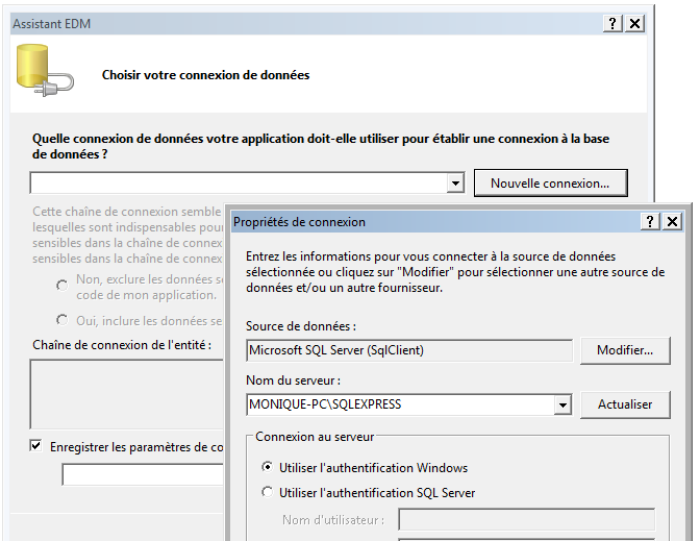
Nom du serveur **SQLEXPRESS**

Authentification **Windows**

Sélectionner la base de données **MUSICATOUT**

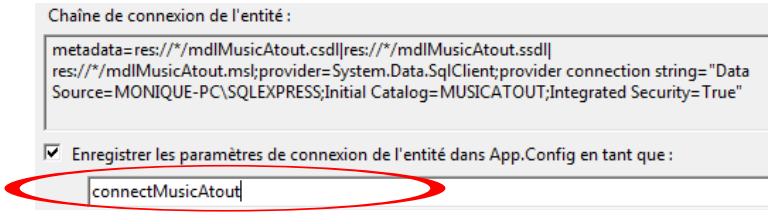
Tester la connexion elle doit avoir **réussi**.

OK pour valider.

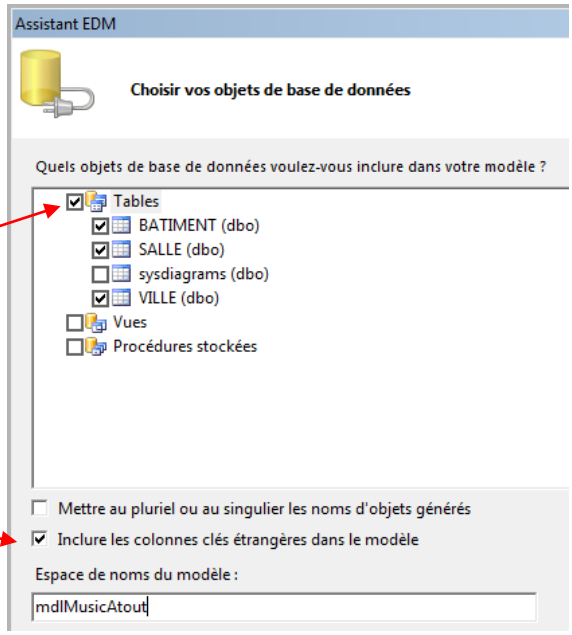


Une nouvelle fenêtre s'ouvre

Attention à bien renommer la **classe « chaine de connexion »** que l'on vient de créer : **connectMusicAtout** .



Choisir les objets que l'on veut utiliser pour la gestion des villes, des salles et des bâtiments

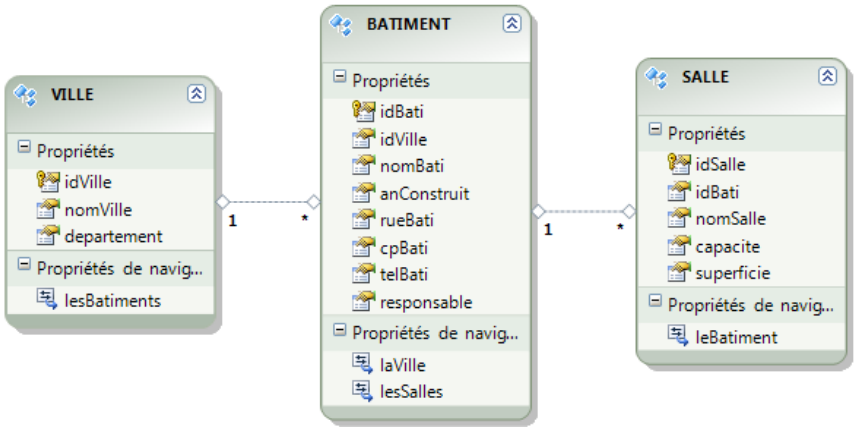


Renommer le modèle EDMX : **mdlMusicAtout**

Le modèle objet final devra ressembler à celui-ci :

Observons ce diagramme des classes.

Les propriétés de navigation ont été renommées pour simplifier la lecture du diagramme et le traitement des données.



2.3 La chaîne de connexion

Observons et comprenons les paramètres de la **chaîne de connexion** qui permet de relier le modèle objet à la base de données SQLServer permettant de manipuler les données depuis C# sans être tributaire des contraintes liées au SGBD.

Les paramètres de la chaîne de connexion sont mémorisés dans le fichier **App.Config** dont on peut lire le contenu en mode texte avec l'éditeur XML (clic droit ouvrir avec)

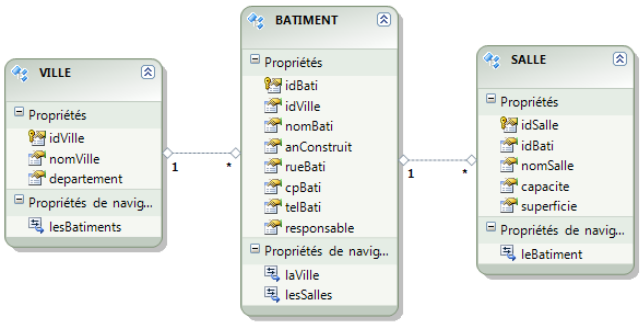
App.Config | mdlMusicAtout.edmx* | Form1.cs [Design]

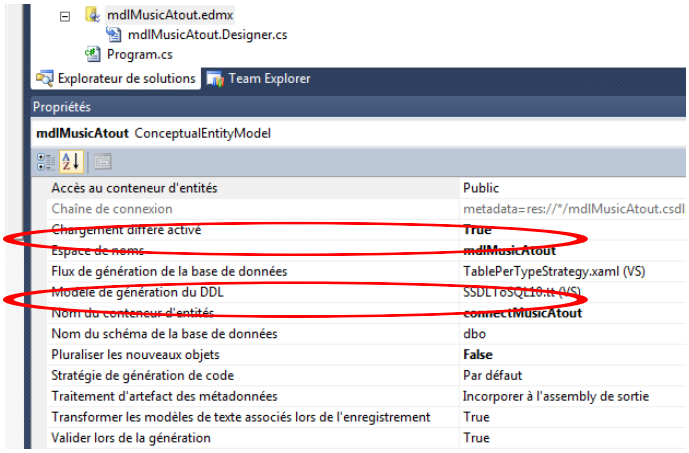
```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
    <add name="connectMusicAtout"
      connectionString="metadata=res://*/mdlMusicAtout.csdl|res://*/mdlMusicAtout.ssdl|res://*/mdlMusicAtout.msl;
        provider            =System.Data.SqlClient;
        provider connection string =&quot;
              data source         = MONIQUE-PC\SQLEXPRESS;
              initial catalog     = MUSICATOUT;
              integrated security = True;
              multipleactiveresultsets=True;
              App=EntityFramework&quot;"
      providerName="System.Data.EntityClient" />
  </connectionStrings>
</configuration>
```

2.4 Observons les propriétés du modèle

Dans la fenêtre du modèle, clic droit, **propriétés**
Vous devez régler deux propriétés :
Espace de noms = **mdlMusicAtout**
Nom du conteneur d'entités = **connectMusicAtout**

On choisit librement ces noms, mais ils vont s'avérer très utiles par la suite. Il est donc **très important de connaître les valeurs de ces deux propriétés**.





Il doit y avoir une **totale correspondance** entre les noms attribués à ces paramètres dans les propriétés du modèle edmx et les paramètres de la chaine de connexion

```
<configuration>
    <connectionStrings>
      <add name="connectMusicAtout"
connectionString="metadata=res://*/mdlMusicAtout.csdl|res://*/mdlMusicAtout.ssdl|res://*/mdl
MusicAtout.msl;
```

La moindre erreur de déclaration dans cette chaîne de connexion et on ne pourra pas mapper le modèle objet et le modèle relationnel, donc pas manipuler les données stockées dans la base depuis le programme C#

Notre dernière étape consiste à vérifier que le mapping modèle objet - modèle relationnel fonctionne et que C# peut accéder aux données de la base EDMRayonsProduits, les lire, voire, les mettre à jour. Mais ça ce sera la prochaine fois. On va maintenant compléter le fichier **Program.c#** qui va permettre de tester notre solution ORM Entity Framework.

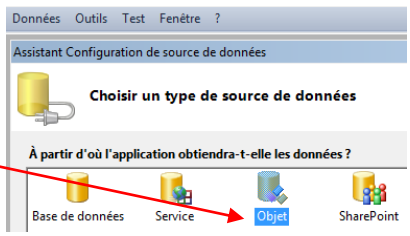
4. Manipuler les données ! En ... fin

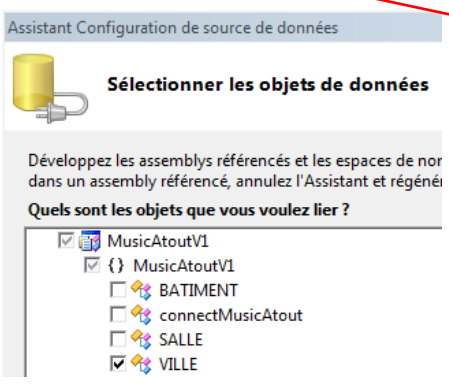
4.1 Première chose à faire ajouter une référence de bibliothèque système :

- Explorateur de solutions, **Ajouter une référence**, **.Net**, **System.Configuration**
- Ajouter une clause ***using System.Configuration*** en entête du programme Program.cs

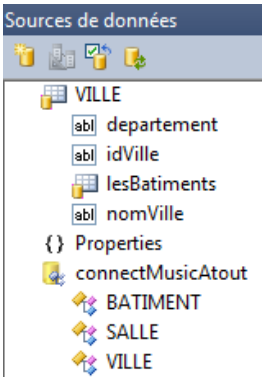
4.2 Gérer les sources de données associées au programme

- Barre de menus, **Données**, **Ajouter une nouvelle source de données**,
- **Choisir un type de source de données Objet**
- Sélectionner l'objet **VILLE**





Afficher les sources de données doit donner une fenêtre similaire à celle ci-contre :

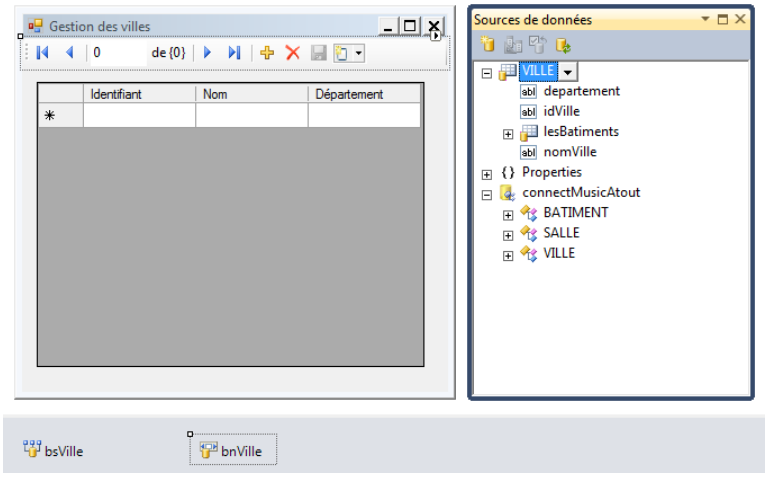


4.3 Créer un formulaire pour gérer les villes

On affiche les sources de données
On glisse l'objet **Ville** sur le formulaire
Les données « ville » sont associées au formulaire

Penser à donner un titre significatif au formulaire
Renommer les entête de colonnes (header)
Supprimer la propriété « lesBatiments »

On renomme les deux composants associés :
Binding Source = **bsVille**
Binding Navigator = **bnVille**
Data Gried View = **dgvVille**



On pourrait avoir quelque chose qui ressemble à cela

Il est temps de générer la solution, et de tenter une première exécution.
Alors ? Rien ne se passe ? Tiens donc …. Pourquoi ?

Corrigeons cela en connectant le binding source, via la chaîne de connexion, à la table de la BDD MusicAtout.mdf.

```
namespace MusicAtoutV1.Forms
{
   public partial class frmVilles : Form
   {
      private connectMusicAtout maConnexion;
      public  frmVilles()
      {
         InitializeComponent();
         maConnexion       =    new connectMusicAtout();

         /* le  bindingSource  bsVille est connecté à la table Ville de la BDD  via la chaîne de connexion */
         bsVille.DataSource = maConnexion.VILLE;
      }
   }
}
```

Un tout dernier point, ranger le formulaire dans un dossier **Forms** réservé à cet effet :
Dans le Programm.cs :

```
Application.Run(new Forms.frmVilles());
```

Votre travail pour finir ! Faire le même travail pour les bâtiments et les salles.