



Dotnet France
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

Premiers pas avec le Framework Entity

Version 1.2



James RAVAILLE

<http://blogs.dotnet-france.com/jamesr>

Sommaire

1	Présentation de l'application à développer	3
1.1	Création de la base de données	3
1.1.1	Le modèle conceptuel de données	3
1.1.2	Le modèle logique de données	3
1.2	Alimentation de la base de données	5
1.3	Architecture de l'application	5
1.4	Création des projets	6
2	Création du composant d'accès aux données.....	7
2.1	Création du composant Entity Data Model.....	7
2.2	Création de la couche d'accès aux données.....	13
2.2.1	Création de la classe de gestion du contexte de données	14
2.2.2	Extension « globale » des entités	16
2.2.3	Extension des classes d'entité	17
3	Affichage et gestion des données	20
3.1	Création du formulaire <i>FrmDetailStagiaire</i>	20
3.1.1	Design du formulaire	20
3.1.2	Code-behind du formulaire	21
3.2	Création du formulaire <i>FrmGestionListeStagiaires</i>	23
3.2.1	Design du formulaire	23
3.2.2	Code-behind du formulaire	25
4	Exécution de l'application	33
5	Conclusion	34

1 Présentation de l'application à développer

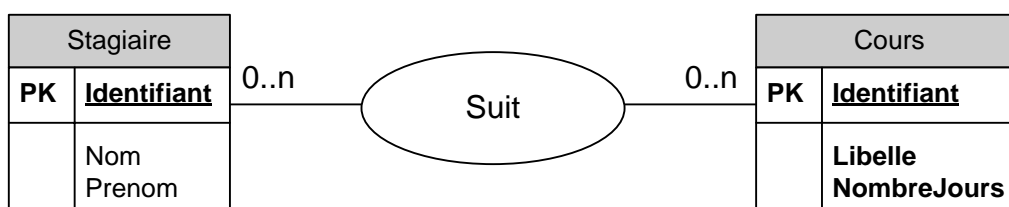
Voici la description d'une application que nous allons développer tout au long de ce cours. L'application aura pour but de consulter et de gérer une liste de stagiaires, et des cours qu'ils peuvent suivre.

1.1 Création de la base de données

1.1.1 Le modèle conceptuel de données

Le modèle conceptuel des données (aussi appelé MCD) permet de représenter de façon formelle, sous forme d'un schéma, les données qui seront utilisées par une application. Ce schéma est composé d'entités, reliées entre elles par des relations d'association.

Voici le modèle conceptuel de données de notre application :



Ce modèle décrit les entités Stagiaire et Cours. Il met aussi en évidence :

- Qu'un stagiaire peut suivre un ou plusieurs cours.
- Qu'un cours peut être suivi par aucun, un ou plusieurs stagiaires.

1.1.2 Le modèle logique de données

Nous allons maintenant créer le modèle logique de données. La relation n-aire bilatérale entre les tables Stagiaire et Cours entraîne la création d'une table supplémentaire, que nous appelons Stagiaire2Cours. Cette table agrège les champs constituant la clé primaire des tables Cours et Stagiaire. Le modèle logique de données obtenu est donc le suivant :



Pour créer ce schéma de base de données, lancer SQL Server Management Studio (interface d'administration de SQL Server). Créez une base de données nommée DotnetFrance, puis dans une fenêtre de requête, exécutez le jeu d'instructions Transact-SQL suivant :



```

-- SQL

USE DotnetFrance

CREATE TABLE [dbo].[Stagiaire] (
    [Identifiant] [int] IDENTITY(1,1) NOT NULL,
    [Nom] [varchar] (50) NOT NULL,
    [Prenom] [varchar] (50) NOT NULL,
    CONSTRAINT [PK_Stagiaire] PRIMARY KEY CLUSTERED
    (
        [Identifiant] ASC
    ) WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
    CONSTRAINT [UQ__Stagiaire__7F60ED59] UNIQUE NONCLUSTERED
    (
        [Identifiant] ASC
    ) WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Cours] (
    [Identifiant] [int] IDENTITY(1,1) NOT NULL,
    [Libelle] [varchar] (50) NOT NULL,
    [NombreJours] [int] NOT NULL,
    CONSTRAINT [PK_Cours] PRIMARY KEY CLUSTERED
    (
        [Identifiant] ASC
    ) WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY],
    CONSTRAINT [UQ__Cours__023D5A04] UNIQUE NONCLUSTERED
    (
        [Identifiant] ASC
    ) WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Stagiaire2Cours] (
    [IdStagiaire] [int] NOT NULL,
    [IdCours] [int] NOT NULL,
    CONSTRAINT [PK_Stagiaire2Cours] PRIMARY KEY CLUSTERED
    (
        [IdStagiaire] ASC,
        [IdCours] ASC
    ) WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Stagiaire2Cours] WITH CHECK ADD CONSTRAINT
[FK_Stagiaire2Cours_Cours] FOREIGN KEY([IdCours])
REFERENCES [dbo].[Cours] ([Identifiant])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Stagiaire2Cours] WITH CHECK ADD CONSTRAINT
[FK_Stagiaire2Cours_Stagiaire] FOREIGN KEY([IdStagiaire])
REFERENCES [dbo].[Stagiaire] ([Identifiant])
ON UPDATE CASCADE
ON DELETE CASCADE
GO

```

Voici une brève description des tables :

- **Stagiaire** : contient toutes les informations relatives aux stagiaires.
- **Cours** : contient toutes les informations relatives aux cours. Les stagiaires doivent suivre des cours afin de pouvoir travailler sur un projet d'un client.
- **Stagiaire2Cours** : permet de lier les stagiaires à un cours. Un stagiaire peut assister à plusieurs cours, et un cours peut concerner plusieurs stagiaires.

1.2 Alimentation de la base de données

Voici un jeu d'instructions Transact-SQL, permettant d'alimenter les tables en données :

```
-- SQL

-- Alimentation de la table des stagiaires.
INSERT INTO Stagiaire (Nom, Prenom) VALUES ('DEROUX', 'Alain')
INSERT INTO Stagiaire (Nom, Prenom) VALUES ('RAVILLE', 'James')
INSERT INTO Stagiaire (Nom, Prenom) VALUES ('SIRON', 'Karl')
INSERT INTO Stagiaire (Nom, Prenom) VALUES ('EMATO', 'Julie')

-- Alimentation de la table des stagiaires.
INSERT INTO Cours (Libelle, NombreJours) VALUES ('SQL Server -
Administration de serveurs', 5)
INSERT INTO Cours (Libelle, NombreJours) VALUES ('XHTML / CSS', 3)
INSERT INTO Cours (Libelle, NombreJours) VALUES ('C#', 5)
INSERT INTO Cours (Libelle, NombreJours) VALUES ('ASP .NET 3.5', 5)
INSERT INTO Cours (Libelle, NombreJours) VALUES ('ASP .NET AJAX', 3)

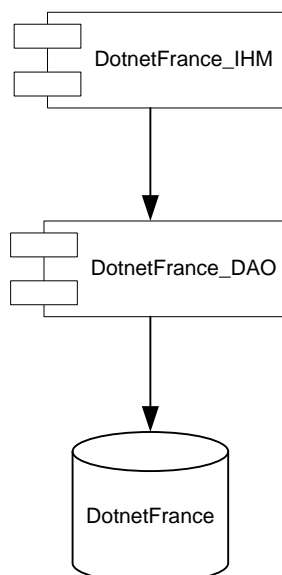
-- Affectation des cours aux stagiaires.
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (1, 1)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (1, 3)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (2, 2)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (2, 1)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (2, 3)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (2, 4)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (3, 1)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (3, 4)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (3, 5)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (4, 4)
INSERT INTO Stagiaire2Cours (IdStagiaire, IdCours) VALUES (4, 5)
```

1.3 Architecture de l'application

L'application est composée de deux projets :

- **DotnetFrance_IHM** : projet permettant d'afficher les données de la base de données DotnetFrance, et d'interagir avec les utilisateurs pour gérer ces données.
- **DotnetFrance_DAO** : projet d'accès et de gestion des données contenues dans la base de données DotnetFrance.

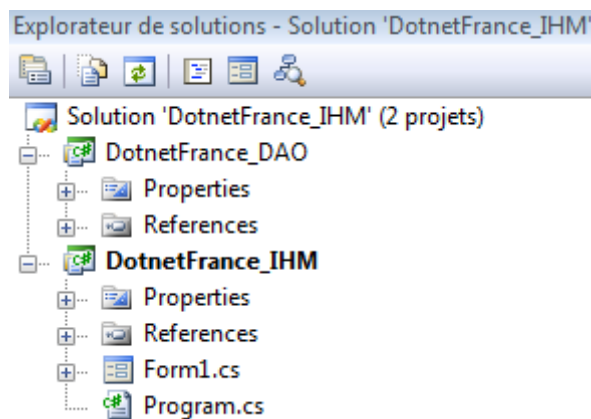
Voici le schéma de notre application :



1.4 Création des projets

Dans Visual Studio 2008, créer deux projets :

- Un projet de type Bibliothèque de classes, nommé *DotnetFrance_DAO* (Data Access Object).
- Un projet de type Windows Forms appelé *DotnetFrance_IHM*.

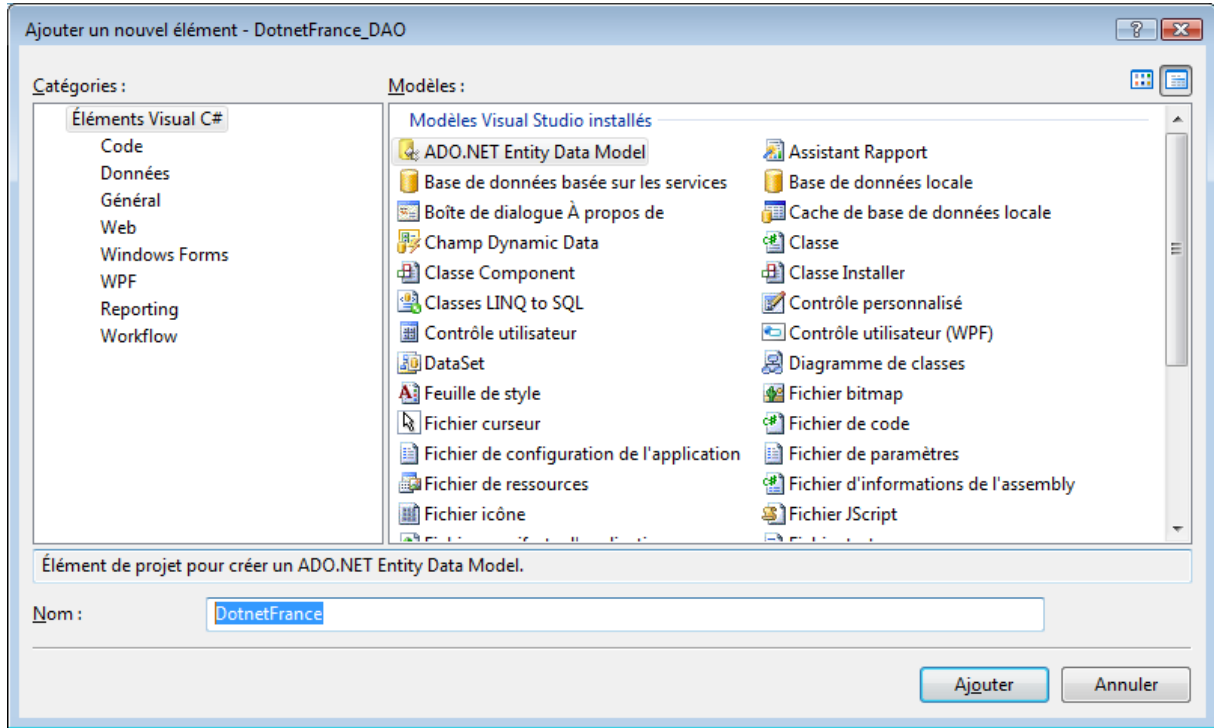


Dans le projet *DotnetFrance_IHM*, ajouter une référence vers le projet *DotnetFrance_DAO*.

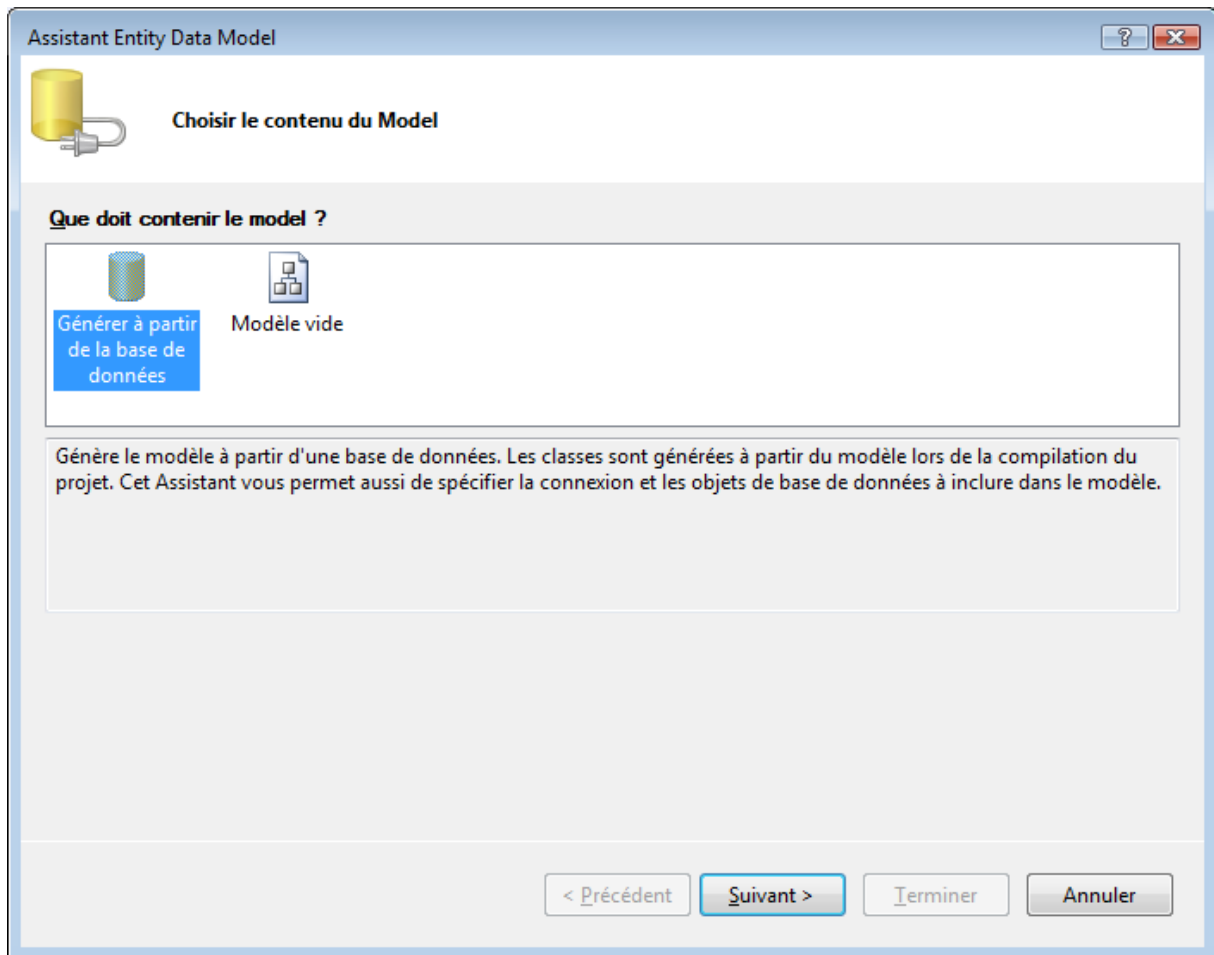
2 Création du composant d'accès aux données

2.1 Création du composant Entity Data Model

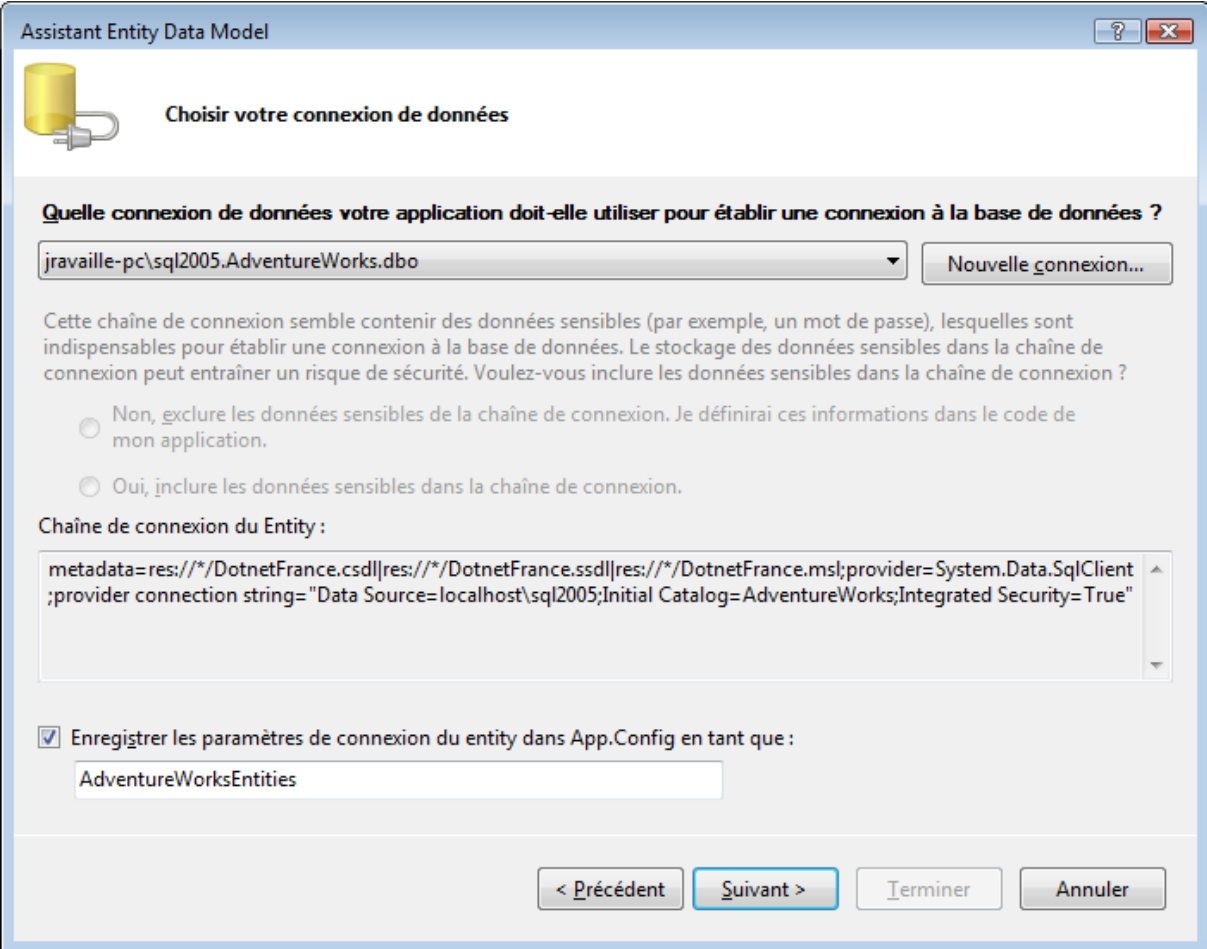
Dans le projet *DotnetFrance_DAO*, ajoutons un composant de type ADO.NET Entity Data Model, nommé *DotnetFrance* :



Après avoir validé, la fenêtre suivante apparaît :



Nous allons créer notre composant d'accès et gestion de données à partir de notre base de données DotnetFrance. Nous sélectionnons alors l'item « Générer à partir de la base de données », et cliquons sur le bouton « Suivant ». La fenêtre suivante apparaît :



Assistant Entity Data Model

Choisir votre connexion de données

Quelle chaîne de connexion votre application doit-elle utiliser pour établir une connexion à la base de données ?

jravaille-pc\sql2005.AdventureWorks.dbo Nouvelle connexion...

Cette chaîne de connexion semble contenir des données sensibles (par exemple, un mot de passe), lesquelles sont indispensables pour établir une connexion à la base de données. Le stockage des données sensibles dans la chaîne de connexion peut entraîner un risque de sécurité. Voulez-vous inclure les données sensibles dans la chaîne de connexion ?

Non, exclure les données sensibles de la chaîne de connexion. Je définirai ces informations dans le code de mon application.

Oui, inclure les données sensibles dans la chaîne de connexion.

Chaîne de connexion du Entity :

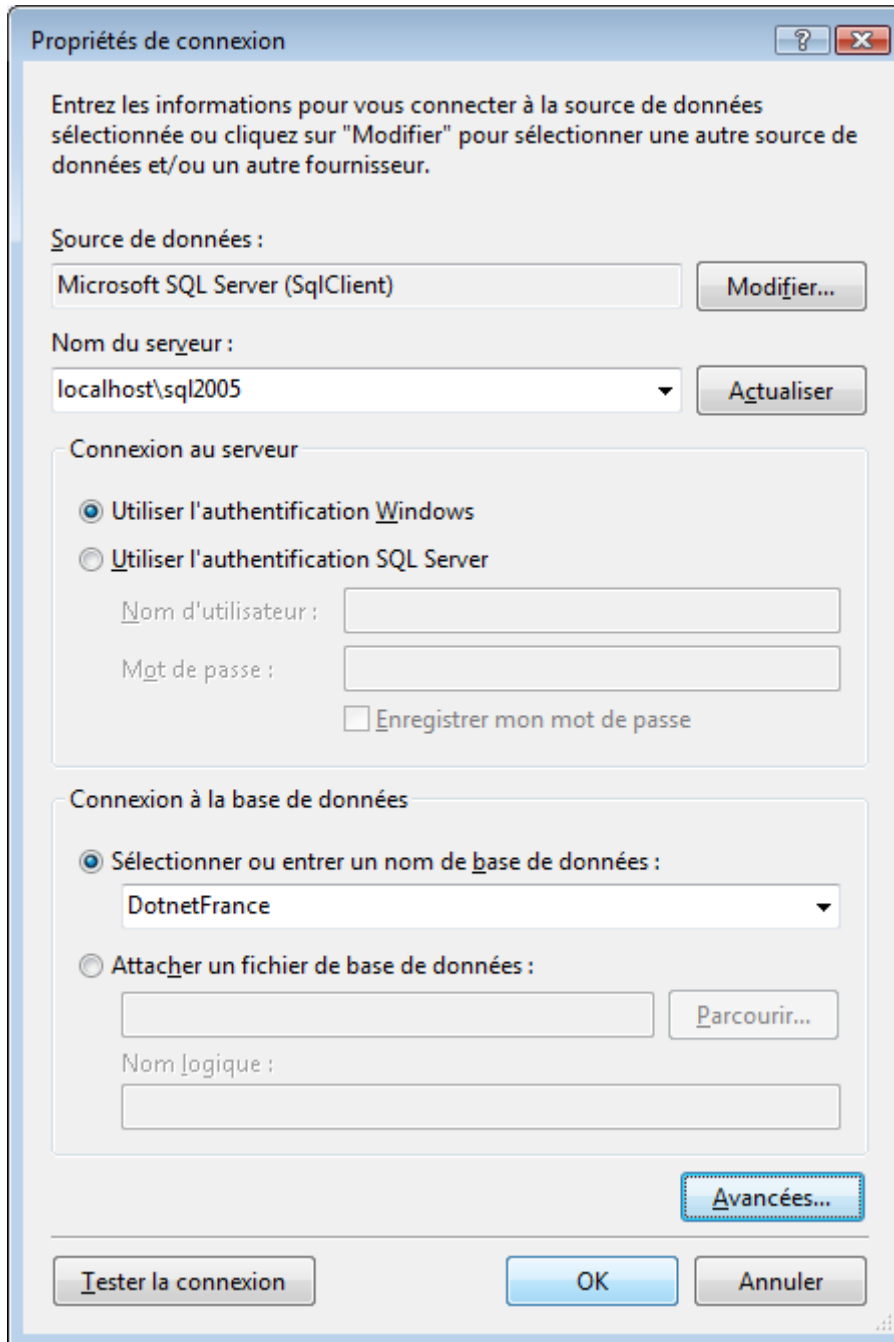
```
metadata=res://*/DotnetFrance.csdl|res://*/DotnetFrance.ssdl|res://*/DotnetFrance.msl;provider=System.Data.SqlClient;provider connection string="Data Source=localhost\sql2005;Initial Catalog=AdventureWorks;Integrated Security=True"
```

Enregistrer les paramètres de connexion du entity dans App.Config en tant que :

AdventureWorksEntities

< Précédent Suivant > Terminer Annuler

Dans cette fenêtre, choisir une connexion pointant vers notre base de données, ou alors créer une nouvelle connexion en cliquant sur le bouton « Nouvelle connexion... ». Dans le second cas, la fenêtre suivante apparaît :



Propriétés de connexion

Entrez les informations pour vous connecter à la source de données sélectionnée ou cliquez sur "Modifier" pour sélectionner une autre source de données et/ou un autre fournisseur.

Source de données :
Microsoft SQL Server (SqlClient) [Modifier...]

Nom du serveur :
localhost\\sql2005 [Actualiser]

Connexion au serveur

Utiliser l'authentification Windows
 Utiliser l'authentification SQL Server

Nom d'utilisateur : []
Mot de passe : []
 Enregistrer mon mot de passe

Connexion à la base de données


Sélectionner ou entrer un nom de base de données :
DotnetFrance []
 Attacher un fichier de base de données :
[] [Parcourir...]
Nom logique : []

[Avancées...]

[Tester la connexion] [OK] [Annuler]

Dans cette fenêtre, saisir le nom de votre instance SQL Server, sur laquelle votre base de données est hébergée. Puis choisissez votre mode d'authentification, ainsi que la base de données. Cliquer sur le bouton « Tester la connexion », afin de vérifier si toutes ces informations sont correctes. Cela est nécessaire, car elles seront utilisées pour créer la chaîne de connexion. Une fois cette fenêtre validée, nous revenons à la fenêtre suivante :

Assistant Entity Data Model

 Choisir votre connexion de données

Quelle connexion de données votre application doit-elle utiliser pour établir une connexion à la base de données ?

Cette chaîne de connexion semble contenir des données sensibles (par exemple, un mot de passe), lesquelles sont indispensables pour établir une connexion à la base de données. Le stockage des données sensibles dans la chaîne de connexion peut entraîner un risque de sécurité. Voulez-vous inclure les données sensibles dans la chaîne de connexion ?

Non, exclure les données sensibles de la chaîne de connexion. Je définirai ces informations dans le code de mon application.

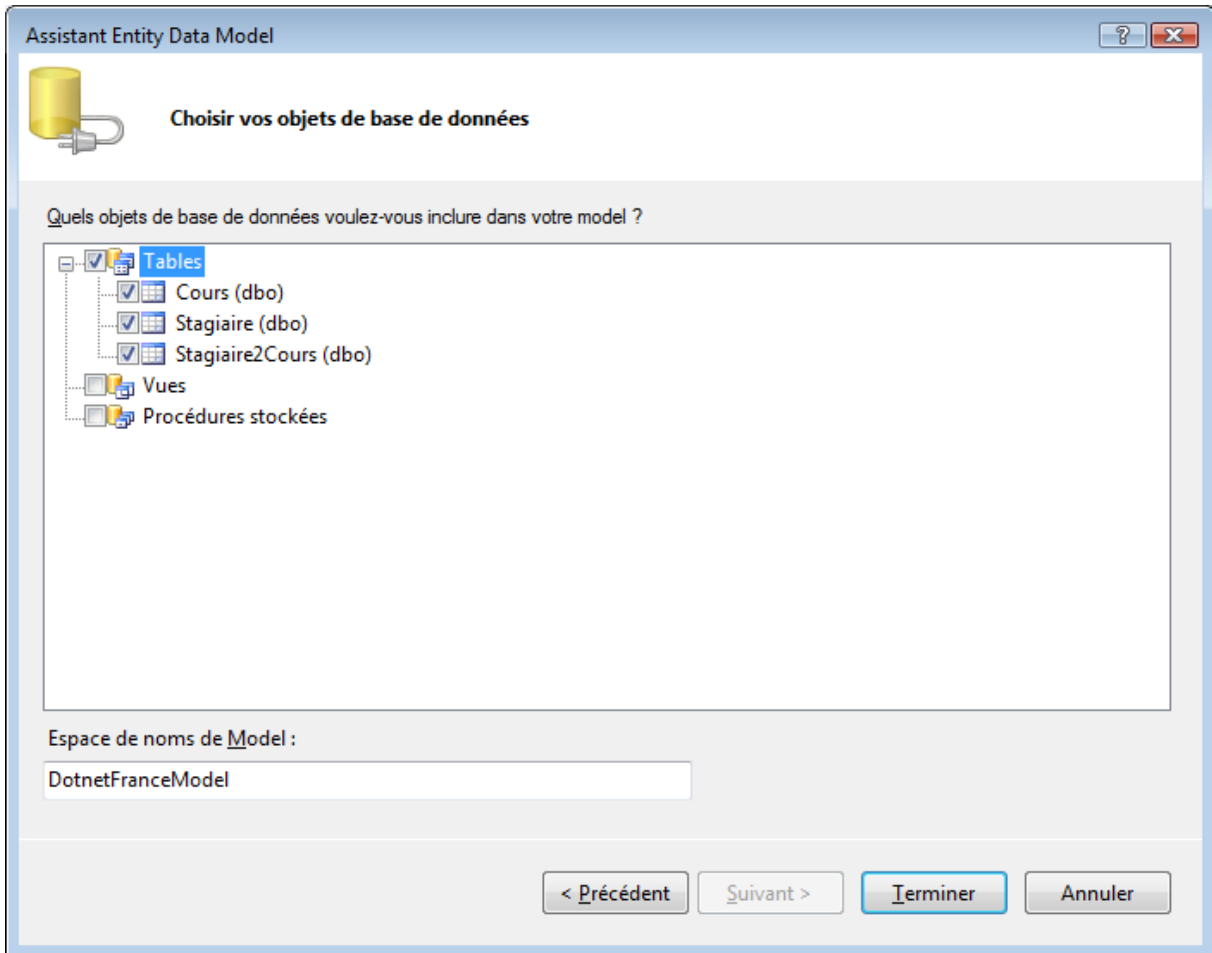
Oui, inclure les données sensibles dans la chaîne de connexion.

Chaîne de connexion du Entity :

```
metadata=res://*/DotnetFranceEntities.csdl|res://*/DotnetFranceEntities.ssdl|res://*/DotnetFranceEntities.msl;provider=System.Data.SqlClient;provider connection string="Data Source=localhost\sql2005;Initial Catalog=DotnetFrance;Integrated Security=True"
```

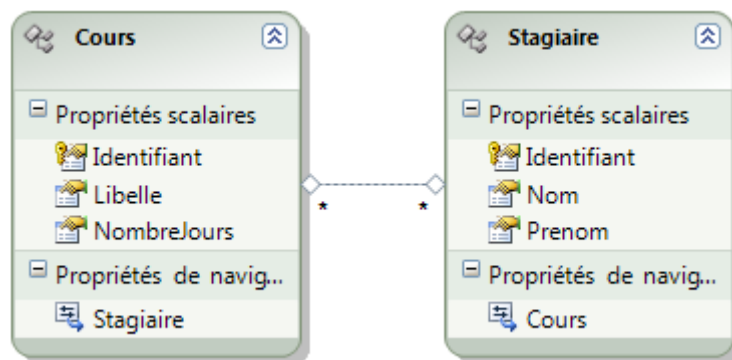
Enregistrer les paramètres de connexion du entity dans App.Config en tant que :

Cliquer alors sur le bouton « Suivant ». La fenêtre suivante apparaît :



Choisissons les tables vues et procédures stockées, que nous allons utiliser dans notre modèle d'entités. Dans notre exemple, nous allons gérer les données contenues dans les tables *Cours*, *Stagiaires* et *Stagiaire2Cours*, que nous sélectionnons. Cliquons ensuite sur le bouton « Terminer ».

Le model suivant apparaît :



Il est possible à tout moment de recharger le modèle à partir de la base de données source, en affichant le menu contextuel et en cliquant sur l'item « Mettre à jour le modèle à partir de la base de données ... ».

Vous pouvez observer que la « table de liaison » *Stagiaire2Cours*, présente dans le modèle logique de données de la base de données, n'est pas présente. En effet, le Framework Entity se base sur une vision conceptuelle de la base de données.

L'implémentation de ces classes est contenue dans le fichier *DotnetFrance.Designer.cs* ou *DotnetFranceEntities.vb*. En ouvrant ce fichier, on peut remarquer :

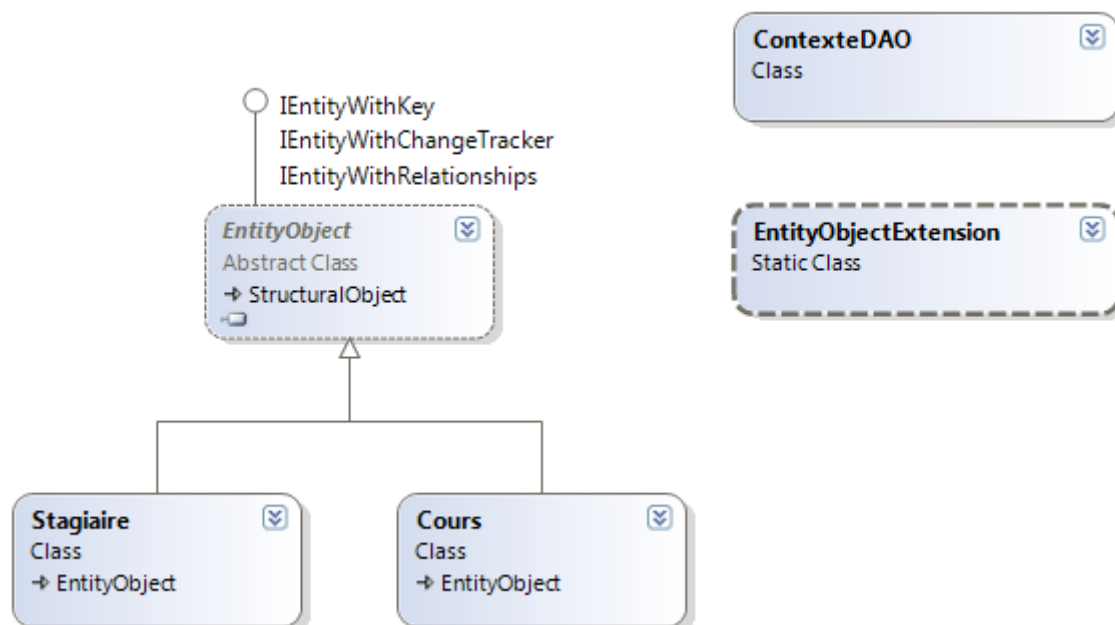
- Que toutes les classes d'entité dérivent de la classe *System.Data.Objects.DataClasses.EntityObject*.
- Qu'une classe supplémentaire est générée. Elle porte le même nom que le composant *Entity Data Model* précédemment créé, suffixé par « Entities ». Cette classe joue un rôle essentiel dans l'accès et la gestion des entités, car elle permet :
 - o De définir un ensemble de membres communs.
 - o De charger les entités dans un contexte de données.
 - o D'effectuer un suivi des modifications effectuées sur les entités.
 - o ...

2.2 Création de la couche d'accès aux données

La construction de la couche d'accès aux données est importante. Elle doit centraliser toutes les opérations :

- Création et gestion du contexte de données.
- De chargement de données et de collections de données :
 - o Chargement de la liste des stagiaires.
 - o Chargement de la liste des cours des stagiaires.
- Ajout d'un stagiaire.
- Modifications des données des objets :
 - o Modification du nom et du prénom d'un stagiaire.
- Suppression des données :
 - o Suppression d'un stagiaire, et de ses inscriptions aux cours.

Voici le diagramme de classes de notre composant *DotnetFrance_DAO* :



2.2.1 Création de la classe de gestion du contexte de données

L'accès aux données proposé par le Framework Entity est basé sur un contexte de données. Nous allons donc créer une classe nommée *ContexteDAO*, dont l'implémentation est la suivante :

- Déclarons un contexte de données, utilisée par l'application *DotnetFrance_IHM*.
- Initialisons le contexte de données. Il ne doit être créé qu'une seule fois, à partir de la chaîne de connexion contenue dans le fichier de configuration de l'application *DotnetFance_IHM*. Pour ce faire, on effectuera donc cette action dans le constructeur statique de la classe.
- Créons une méthode permettant de persister dans la base de données, toutes les actions sur les objets du contexte de données : ajouts, modifications et suppressions de d'entité.

Voici le code de cette classe :

```
// C#  
  
public partial class ContexteDAO  
{  
    private static DotnetFranceEntities _ContexteDonnees;  
    internal static DotnetFranceEntities ContexteDonnees  
    {  
        get { return ContexteDAO._ContexteDonnees; }  
        set { ContexteDAO._ContexteDonnees = value; }  
    }  
  
    static ContexteDAO()  
    {  
        ContexteDAO.ContexteDonnees = new  
DotnetFranceEntities(ConfigurationManager.ConnectionStrings["CS_DotnetFra  
nce"].ConnectionString);  
    }  
  
    public static void Enregistrer()  
    {  
        ContexteDAO.ContexteDonnees.SaveChanges();  
    }  
}
```

```
' VB .NET  
  
Partial Public Class ContexteDAO  
    Private Shared _ContexteDonnees As DotnetFranceEntities  
    Friend Shared Property ContexteDonnees() As DotnetFranceEntities  
        Get  
            Return ContexteDAO._ContexteDonnees  
        End Get  
        Set (ByVal value As DotnetFranceEntities)  
            ContexteDAO._ContexteDonnees = value  
        End Set  
    End Property  
  
    Shared Sub New()  
        ContexteDAO.ContexteDonnees = New  
DotnetFranceEntities(ConfigurationManager.ConnectionStrings("CS_DotnetFra  
nce").ConnectionString)  
    End Sub  
  
    Public Shared Sub Enregistrer()  
        ContexteDAO.ContexteDonnees.SaveChanges()  
    End Sub  
End Class
```

Pour utiliser la classe *ConfigurationManager*, il est nécessaire de référencer l'assembly *System.Configuration.dll* du Framework .NET.

Aussi, nous n'oublions pas d'ajouter ultérieurement dans le fichier de configuration de l'application *DotnetFrance_IHM*. Cette chaîne de connexion est la suivante :

```

metadata=res://*/DotnetFrance.csdl|res://*/DotnetFrance.ssdl|res://*/DotnetFrance.msl;provider=System.Data.SqlClient;provider connection string="Data Source=localhost\sql2005;Initial Catalog=DotnetFrance;Integrated Security=True;MultipleActiveResultSets=True"

```

Elle contient des informations sur le mappage et les métadonnées de l'Entity Data Model. Elle présente les caractéristiques suivantes :

Mot clé	Description
Metadata	Permet de spécifier la liste des fichiers séparés par un caractère « pipe » (), contenant les informations de métadonnées et de mappage (csdl, ssdl et msl).
Provider Connection String	Permet de spécifier la chaîne de connexion. Elle est exprimée à l'aide de paires mot clé/valeur, valides pour le fournisseur de données. Par exemple pour SQL Server : <ul style="list-style-type: none"> - Data Source : nom de l'instance SQL Server contenant la base de données. - Initial Catalog : nom de la base de données. - Pour l'authentification : <ul style="list-style-type: none"> o Integrated Security=True : utilisation de l'authentification Windows o User id="nom utilisateur SQL" et password="mot de passe" : utilisation de l'authentification SQL Server

2.2.2 Extension « globale » des entités

Toutes les classes d'entité dérivent de la classe *System.Data.Objects.DataClasses.EntityObject*. Ainsi, en étendant cette classe via des méthodes d'extension, ces méthodes peuvent être appliquées à toutes les instances des classes dérivant de cette même classe. Les méthodes d'extension que nous créons, correspondent aux actions suivantes :

- Ajouter une entité.
- Supprimer une entité.
- Annuler les modifications effectuées sur une entité, en relisant les données dans la base de données.

Voici le code de ces méthodes d'extension :

```
// C#

public static class EntityObjectExtension
{
    public static void Ajouter(this EntityObject aObject)
    {
        ContexteDAO.ContexteDonnees.AddObject(aObject.GetType().Name,
aObject);
        ContexteDAO.Enregistrer();
    }

    public static void Supprimer(this EntityObject aObject)
    {
        ContexteDAO.ContexteDonnees.DeleteObject(aObject);
        ContexteDAO.Enregistrer();
    }

    public static void Rafraichir(this EntityObject aObject)
    {
        ContexteDAO.ContexteDonnees.Refresh(System.Data.Objects.RefreshMode.Store
Wins, aObject);
    }
}
```

```
' VB .NET

Public Module EntityObjectExtension
    <Extension()> _
    Public Sub Ajouter(ByVal aObject As EntityObject)
        ContexteDAO.ContexteDonnees.AddObject(aObject.GetType().Name,
aObject)
        ContexteDAO.Enregistrer()
    End Sub

    <Extension()> _
    Public Sub Supprimer(ByVal aObject As EntityObject)
        ContexteDAO.ContexteDonnees.DeleteObject(aObject)
        ContexteDAO.Enregistrer()
    End Sub

    <Extension()> _
    Public Sub Rafraichir(ByVal aObject As EntityObject)
        ContexteDAO.ContexteDonnees.Refresh(System.Data.Objects.RefreshMode.Store
Wins, aObject)
    End Sub
End Module
```

2.2.3 Extension des classes d'entité

Etendons les classes d'entité *Stagiaire*, au travers du mécanisme des classes partielles. La classe *Stagiaire* propose :

- Un accesseur en lecture seule nommé *NombreCours*, permettant de connaître le nombre de cours auxquels le stagiaire est inscrit.

- Un accesseur en lecture seule nommé *NombreJoursDeCours*, permettant de connaître le nombre de jours total de formation du stagiaire.
- Une méthode statique nommée *GetListeInstances*, permettant de charger dans le contexte de données et d'obtenir, les entités sur les stagiaires et leurs cours.

```
// C#  
  
public partial class Stagiaire  
{  
    public int NombreCours  
    {  
        get  
        {  
            return this.Cours.Count;  
        }  
    }  
  
    public int NombreJoursDeCours  
    {  
        get  
        {  
            return (from oCours in this.Cours  
                    select oCours.NombreJours).Sum();  
        }  
    }  
  
    public static List<Stagiaire> GetListeInstances()  
    {  
        return ContexteDAO.ContexteDonnees.Stagiaire  
            .Include("Cours")  
            .ToList();  
    }  
}
```

```
' VB .NET  
  
Public Class Stagiaire  
    Public ReadOnly Property NombreCours() As Integer  
        Get  
            Return Me.Cours.Count  
        End Get  
    End Property  
  
    Public ReadOnly Property NombreJoursDeCours() As Integer  
        Get  
            Return (From oCours In Me.Cours _  
                    Select oCours.NombreJours).Sum()  
        End Get  
    End Property  
  
    Public Shared Function GetListeInstances() As List(Of Stagiaire)  
        Return ContexteDAO.ContexteDonnees.Stagiaire _  
            .Include("Cours") _  
            .ToList()  
    End Function  
End Class
```

Dans la méthode *GetListeInstances*, vous remarquerez l'utilisation de la méthode *Include*, afin de charger dans le contexte de données les cours des stagiaires.

Dans notre exemple, étendre la classe *Cours* n'apporte aucune plus-value.

3 Affichage et gestion des données

Dans le projet *DotnetFrance_IHM*, nous allons créer des formulaires d'affichage et de gestion des données contenues dans la base de données DotnetFrance, au travers du composant *DotnetFrance_DAO*.

Le projet *DotnetFrance_IHM* est composé de deux formulaires :

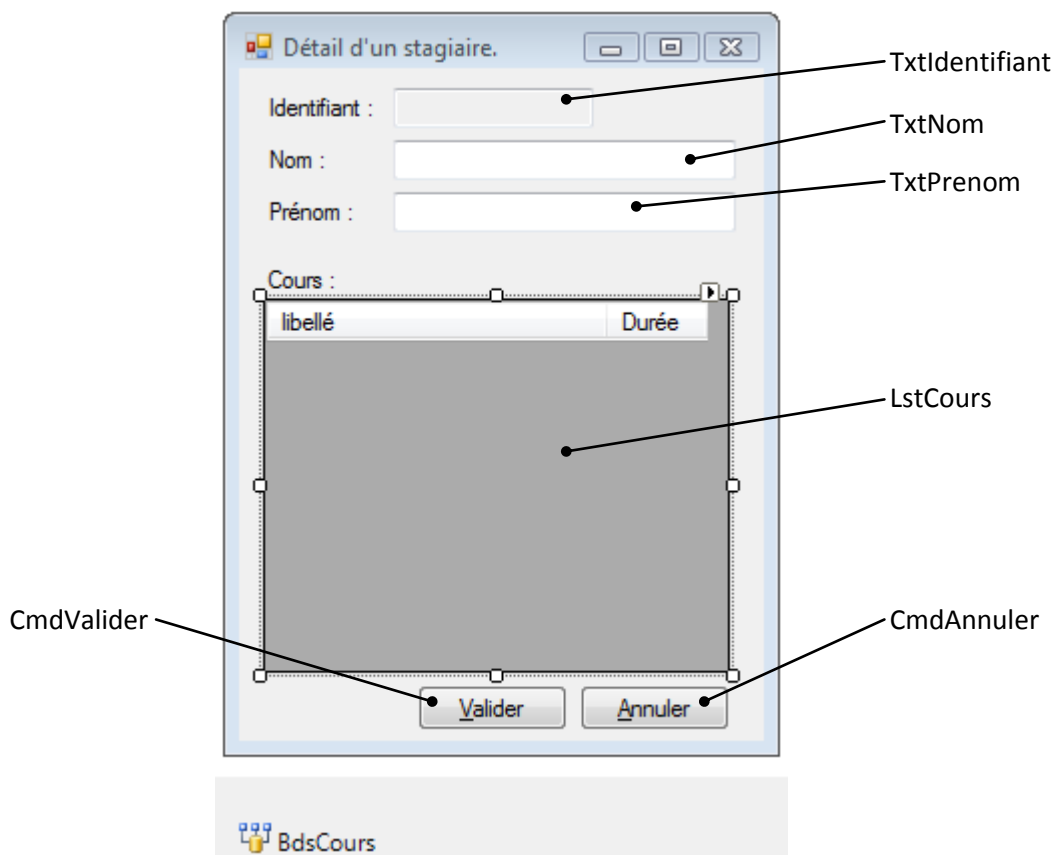
- Un formulaire nommé *FrmGestionListeStagiaires*. Il permet de consulter et gérer la liste des stagiaires.
- Un formulaire nommé *FrmDetailStagiaire*, qui permet de consulter / modifier les informations relatives à un stagiaire. Ce formulaire sera aussi utilisé pour ajouter un stagiaire.

Afin de pouvoir utiliser les classes d'entité dans le projet *DotnetFrance_IHM*, il est nécessaire d'ajouter une référence vers l'assembly *System.Data.Entity.dll* du Framework .NET.

3.1 Création du formulaire *FrmDetailStagiaire*

3.1.1 Design du formulaire

Voici une présentation de ce formulaire :



Voici quelques propriétés des contrôles de ce formulaire :

Contrôles	Propriétés	Valeurs
TxtIdentifiant		
	ReadOnly	True
LstCours		
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	Columns	Ajout de deux colonnes : <ul style="list-style-type: none"> - Libelle : liée à la propriété <i>Libelle</i> - Durée : liée à la propriété <i>NombreJours</i>
	DataSource	BdsCours
	MultiSelect	False
	ReadOnly	True
	RowHeadersVisible	false
	SelectionMode	FullRowSelect

3.1.2 Code-behind du formulaire

Ce formulaire va « gérer » les informations concernant les stagiaires. Nous allons donc :

- Créer un attribut de type *Stagiaire*, avec son accesseur.
- Initialiser cet attribut dans le constructeur.
- Implémenter l'évènement *Load*, de manière à utiliser le databinding pour :
 - o Afficher l'identifiant. Dans le cas d'un ajout d'un stagiaire, les caractères « -- » seront affichés. En effet, la colonne *Identifiant* de la table *Stagiaire* étant une colonne identité, l'identifiant du stagiaire sera déterminé par SQL Server lors de l'ajout du stagiaire.
 - o Afficher le nom, prénom, et la liste des cours auxquels le stagiaire est inscrit.
- Implémenter l'évènement *Click* sur le bouton *CmdValider*, afin d'enregistrer que l'utilisateur confirme l'ajout / modification du stagiaire.
- Implémenter l'évènement *Click* sur le bouton *CmdAnnuler*, afin d'enregistrer que l'utilisateur annule l'ajout / modification du stagiaire.

Voici le code behind de ce formulaire :



```
// C#

public partial class FrmDetailStagiaire : Form
{
    private Stagiaire _oStagiaire;
    public Stagiaire oStagiaire
    {
        get { return _oStagiaire; }
        set { _oStagiaire = value; }
    }

    public FrmDetailStagiaire(Stagiaire aStagiaire)
    {
        InitializeComponent();

        // Initialisation des attributs.
        this.oStagiaire = aStagiaire;
    }

    private void CmdValider_Click(object sender, EventArgs e)
    {
        this.DialogResult = DialogResult.OK;
    }

    private void CmdAnnuler_Click(object sender, EventArgs e)
    {
        this.DialogResult = DialogResult.Cancel;
    }

    private void FrmDetailStagiaire_Load(object sender, EventArgs e)
    {
        if (oStagiaire.EntityKey == null)
        {
            TxtIdentifiant.Text = "--";
        }
        else
        {
            TxtIdentifiant.DataBindings.Add("Text", this.oStagiaire,
"Identifiant");
        }

        TxtNom.DataBindings.Add("Text", this.oStagiaire, "Nom");
        TxtPrenom.DataBindings.Add("Text", this.oStagiaire, "Prenom");

        BdsCours.DataSource = this.oStagiaire.Cours;
    }
}
```



```
' VB .NET

Public Class FrmDetailStagiaire

    Private _oStagiaire As Stagiaire
    Private Property oStagiaire() As Stagiaire
        Get
            Return Me._oStagiaire
        End Get
        Set(ByVal value As Stagiaire)
            Me._oStagiaire = value
        End Set
    End Property

    Public Sub New(ByVal aStagiaire As Stagiaire)

        ' Cet appel est requis par le Concepteur Windows Form.
        InitializeComponent()

        ' Ajoutez une initialisation quelconque après l'appel
        InitializeComponent().
        Me.oStagiaire = aStagiaire
    End Sub

    Private Sub FrmDetailStagiaire_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load
        If (oStagiaire.EntityKey Is Nothing) Then
            TxtIdentifiant.Text = "--"
        Else
            TxtIdentifiant.DataBindings.Add("Text", Me.oStagiaire,
            "Identifiant")
        End If

        TxtNom.DataBindings.Add("Text", Me.oStagiaire, "Nom")
        TxtPrenom.DataBindings.Add("Text", Me.oStagiaire, "Prenom")

        BdsCours.DataSource = Me.oStagiaire.Cours
    End Sub

    Private Sub CmdValider_Click(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles CmdValider.Click
        Me.DialogResult = DialogResult.OK
    End Sub

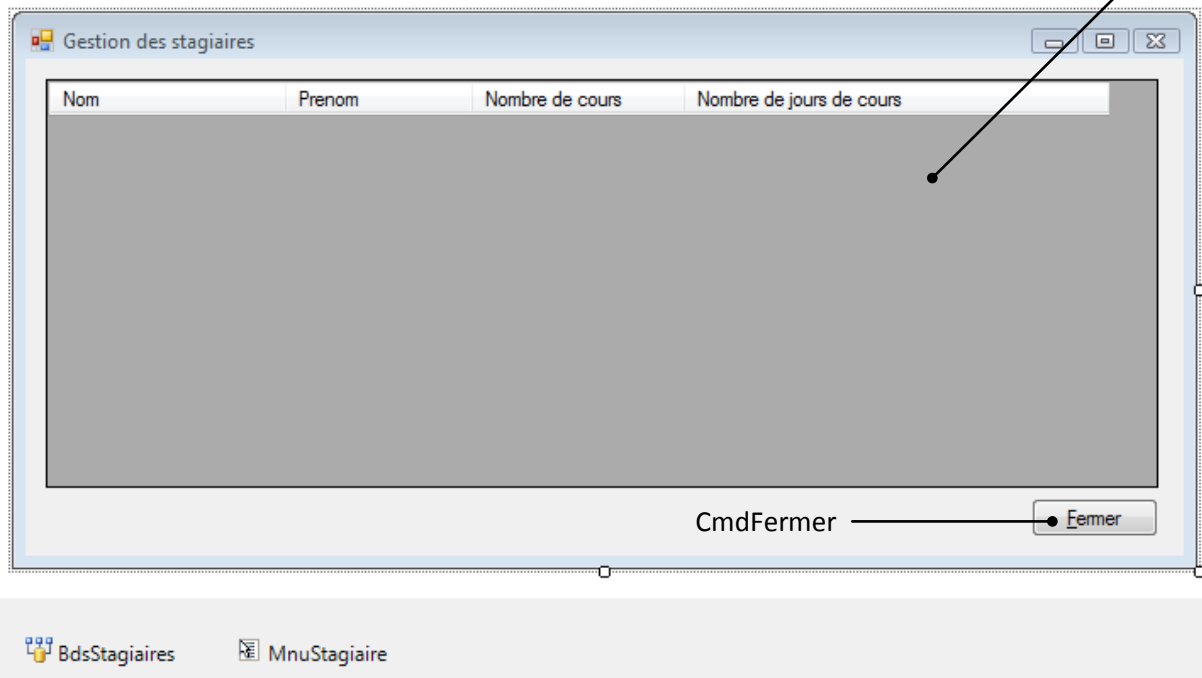
    Private Sub CmdAnnuler_Click(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles CmdAnnuler.Click
        Me.DialogResult = DialogResult.Cancel
    End Sub

End Class
```

3.2 Création du formulaire *FrmGestionListeStagiaires*

3.2.1 Design du formulaire

Voici une présentation de ce formulaire :



Voici quelques propriétés des contrôles de ce formulaire :

Contrôles	Propriétés	Valeurs
LstStagiaires		
	AllowUserToAddRows	False
	AllowUserToDeleteRows	False
	Columns	Ajout de quatre colonnes : - Nom : liée à la propriété <i>Nom</i> - Prénom : liée à la propriété <i>Prenom</i> - Nombre de cours : <i>NombreCours</i> - Nombre de jours de cours : <i>NombreJoursDeCours</i>
	DataSource	BdsStagiaires
	MultiSelect	False
	ReadOnly	True
	RowHeadersVisible	False
	SelectionMode	FullRowSelect

Le contrôle de type *BindingSource* nommé *BdsStagiaire*, permet de simplifier la mise en œuvre du databinding dans notre formulaire. Nous l'utiliserons pour :

- Afficher la liste des stagiaires et des informations connexes.
- Ajouter / modifier / supprimer un stagiaire dans/de la liste.
- Obtenir à tout moment un objet métier (entité, objet fortement typé) correspondant au stagiaire sélectionné dans la liste.

Le contrôle de type *ContextMenuStrip* nommé *MnuStagiaire*, permet d'afficher un menu contextuel sur un stagiaire sélectionné dans la liste. Les items de ce menu permettent d'ajouter, modifier et supprimer un stagiaire.

3.2.2 Code-behind du formulaire

3.2.2.1 Pour obtenir l'entité du stagiaire courante

Nous allons créer un accesseur en lecture seule, permettant d'obtenir l'entité du stagiaire sélectionné dans la liste :

```
// C#  
  
private Stagiaire StagiaireCourant  
{  
    get  
    {  
        return (Stagiaire)BdsStagiaires.Current;  
    }  
}
```

```
' VB .NET  
  
Private ReadOnly Property StagiaireCourant() As Stagiaire  
    Get  
        Return CType(BdsStagiaires.Current, Stagiaire)  
    End Get  
End Property
```

3.2.2.2 Lors du chargement du formulaire

Lors du chargement du formulaire, obtenir et afficher la liste des stagiaires. Pour ce faire, nous faisons appel à la méthode statique *GetListeInstances* de la classe *Stagiaire* de notre modèle d'entités :

```
// C#  
  
private void FrmListeStagiaires_Load(object sender, EventArgs e)  
{  
    try  
    {  
        // Affichage de la liste des stagiaires.  
        BdsStagiaires.DataSource = Stagiaire.GetListeInstances();  
    }  
    catch (Exception aEx)  
    {  
        MessageBox.Show(aEx.Message);  
    }  
}
```

```
' VB .NET

Private Sub FrmGestionListeStagiaires_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
    Try
        ' Affichage de la liste des stagiaires.
        BdsStagiaires.DataSource = Stagiaire.GetListeInstances()
    Catch aEx As Exception
        MessageBox.Show(aEx.Message)
    End Try
End Sub
```

3.2.2.3 Pour fermer le formulaire

Voici l'implémentation de l'évènement Click sur le bouton intitulé Fermer :

```
// C#

private void CmdFermer_Click(object sender, EventArgs e)
{
    // Fermeture du formulaire.
    this.Close();
}
```

```
' VB .NET

Private Sub CmdFermer_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CmdFermer.Click
    ' Fermeture du formulaire.
    Me.Close()
End Sub
```

3.2.2.4 Gestion de l'affichage du menu

Le menu contextuel sur un stagiaire doit être affiché, uniquement un stagiaire est sélectionné dans la liste. Voici l'implémentation de l'évènement *Opening* sur le menu *MnuStagiaire* :

```
// C#

private void MnuStagiaire_Opening(object sender, CancelEventArgs e)
{
    // Si aucun stagiaire n'est sélectionné, alors on n'affiche pas le
    menu contextuel.
    e.Cancel = this.StagiaireCourant == null;
}
```

```
' VB .NET

Private Sub MnuStagiaire_Opening(ByVal sender As System.Object, ByVal e
As System.ComponentModel.CancelEventArgs) Handles MnuStagiaire.Opening
    ' Si aucun stagiaire n'est sélectionné, alors on n'affiche pas le
    menu contextuel.
    e.Cancel = Me.StagiaireCourant Is Nothing
End Sub
```

3.2.2.5 Ajouter un stagiaire

Pour ajouter un stagiaire, nous créons une instance de classe *Stagiaire* de notre modèle d'entités. Puis, nous gérons les données contenues dans cet objet, via une instance du formulaire *FrmDetailStagiaire*. Après la fermeture cette fenêtre, si l'utilisateur a cliqué sur le bouton *Valider*, alors le stagiaire est enregistré dans la base de données.

```
// C#

private void ajouterToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Variables locales.
    Stagiaire oStagiaire;
    FrmDetailStagiaire oForm;
    DialogResult oResult;

    try
    {
        // Initialisation.
        oStagiaire = new Stagiaire();

        // Création du formulaire de détail sur un stagiaire.
        oForm = new FrmDetailStagiaire(oStagiaire);

        // Affichage.
        oResult = oForm.ShowDialog();

        if (oResult == DialogResult.OK)
        {
            // Ajout du stagiaire en base de données.
            oStagiaire.Ajouter();

            // Affichage du stagiaire à l'écran.
            BdsStagiaires.Add(oStagiaire);
        }
    }
    catch (Exception aEx)
    {
        MessageBox.Show(aEx.Message);
    }
}
```

```
' VB .NET

Private Sub ajouterToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ajouterToolStripMenuItem.Click
    ' Variables locales.
    Dim oStagiaire As Stagiaire
    Dim oForm As FrmDetailStagiaire
    Dim oResult As DialogResult

    Try
        ' Initialisation.
        oStagiaire = New Stagiaire()

        ' Création du formulaire de détail sur un stagiaire.
        oForm = New FrmDetailStagiaire(oStagiaire)

        ' Affichage.
        oResult = oForm.ShowDialog()

        If (oResult = DialogResult.OK) Then
            ' Ajout du stagiaire en base de données.
            oStagiaire.Ajouter()

            ' Affichage du stagiaire à l'écran.
            BdsStagiaires.Add(oStagiaire)
        End If
    Catch aEx As Exception
        MessageBox.Show(aEx.Message)
    End Try
End Sub
```

3.2.2.6 Modifier un stagiaire

Pour modifier un stagiaire, nous récupérons une instance de classe *Stagiaire* via notre accesseur *StagiaireCourant*. Puis, nous gérons les données contenues dans cet objet, via une instance du formulaire *FrmDetailStagiaire*. Après la fermeture cette fenêtre, si l'utilisateur a cliqué sur le bouton *Valider*, alors les modifications effectuées sur le stagiaire sont enregistrées dans la base de données. Le cas échéant, ces modifications sont annulées.



```
// C#

private void modifierToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Variables locales.
    Stagiaire oStagiaire;
    FrmDetailStagiaire oForm;
    DialogResult oResult;

    try
    {
        // Initialisation.
        oStagiaire = this.StagiaireCourant;

        if (oStagiaire != null)
        {
            // Création du formulaire de détail sur un stagiaire.
            oForm = new FrmDetailStagiaire(oStagiaire);

            // Affichage.
            oResult = oForm.ShowDialog();

            if (oResult == DialogResult.OK)
            {
                // Enregistrement des modifications dans la base de
données.
                DotnetFranceEntities.Enregistrer();
            }
            else
            {
                // Annulation des modifications.
                oStagiaire.Rafraichir();
            }
        }
    }
    catch (Exception aEx)
    {
        MessageBox.Show(aEx.Message);
    }
}
```

```
' VB .NET

Private Sub modifierToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
modifierToolStripMenuItem.Click
    ' Variables locales.
    Dim oStagiaire As Stagiaire
    Dim oForm As FrmDetailStagiaire
    Dim oResult As DialogResult

    Try
        ' Initialisation.
        oStagiaire = Me.StagiaireCourant

        ' Création du formulaire de détail sur un stagiaire.
        oForm = New FrmDetailStagiaire(oStagiaire)

        ' Affichage.
        oResult = oForm.ShowDialog()

        If (oResult = DialogResult.OK) Then
            ' Enregistrement des modifications dans la base de données.
            ContexteDAO.Enregistrer()
        Else
            ' Annulation des modifications.
            oStagiaire.Rafraichir()
        End If
    Catch aEx As Exception
        MessageBox.Show(aEx.Message)
    End Try
End Sub
```

3.2.2.7 Supprimer un stagiaire

Pour supprimer un stagiaire, nous récupérons une instance de classe *Stagiaire* via notre accesseur *StagiaireCourant*. Une demande de confirmation est effectuée. Si l'utilisateur valide son choix de suppression, alors le stagiaire est supprimé de la base de données.

Notez que dans la base de données, toutes les relations d'association entre les tables de notre modèle acceptent la suppression en cascade. De ce fait, nous ne supprimons pas les inscriptions des stagiaires aux cours, avant de supprimer un stagiaire.



```
// C#

private void supprimerToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Variables locales.
    Stagiaire oStagiaire;
    DialogResult oResult;

    try
    {
        // Initialisation.
        oStagiaire = this.StagiaireCourant;

        if (oStagiaire != null)
        {
            // Demande de confirmation.
            oResult = MessageBox.Show("Etes-vous sûrs de vouloir
supprimer ce stagiaire ?", "Demande de confirmation",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);

            if (oResult == DialogResult.Yes)
            {
                // Suppression du stagiaire dans la base de données.
                oStagiaire.Supprimer();

                // Suppression du stagiaire dans la grille.
                BdsStagiaires.RemoveCurrent();
            }
        }
    }
    catch (Exception aEx)
    {
        MessageBox.Show(aEx.Message);
    }
}
```

```
' VB .NET

Private Sub supprimerToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
supprimerToolStripMenuItem.Click
    ' Variables locales.
    Dim oStagiaire As Stagiaire
    Dim oResult As DialogResult

    Try
        ' Initialisation.
        oStagiaire = Me.StagiaireCourant

        If (oStagiaire IsNot Nothing) Then
            ' Demande de confirmation.
            oResult = MessageBox.Show("Etes-vous sûrs de vouloir
supprimer ce stagiaire ?", "Demande de confirmation",
MessageBoxButtons.YesNo, MessageBoxIcon.Question)

            If (oResult = DialogResult.Yes) Then
                ' Suppression du stagiaire dans la base de données.
                oStagiaire.Supprimer()

                ' Suppression du stagiaire dans la grille.
                BdsStagiaires.RemoveCurrent()
            End If
        End If
    Catch aEx As Exception
        MessageBox.Show(aEx.Message)
    End Try
End Sub
```

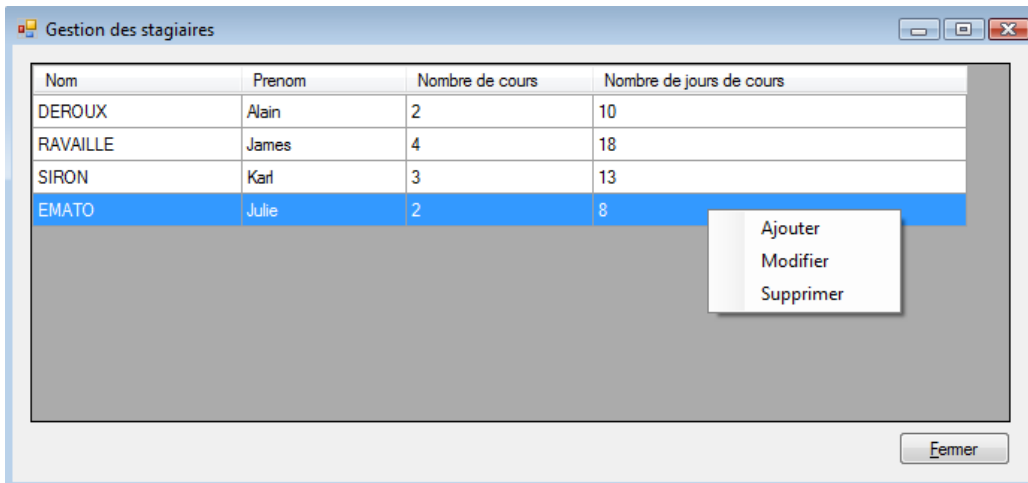
3.2.2.8 Paramétrage du fichier de configuration

Cette application doit définir la chaîne de connexion, qu'elle fournira au composant *DotnetFrance_DAO*. Pour ce faire, ajoutez un fichier de configuration dans cette application, dont le contenu est le suivant :

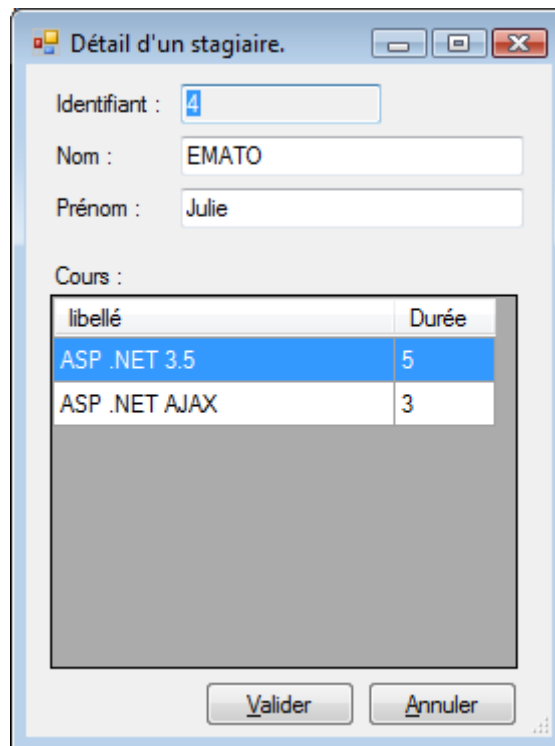
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="CS_DotnetFrance"
connectionString="metadata=res://*/DotnetFrance.csdl|res://*/DotnetFrance
.ssd|res://*/DotnetFrance.msl;provider=System.Data.SqlClient;provider
connection string=&quot;Data Source=localhost\sql2005;Initial
Catalog=DotnetFrance;Integrated
Security=True;MultipleActiveResultSets=True&quot;;"
providerName="System.Data.EntityClient" />
  </connectionStrings>
</configuration>
```

4 Exécution de l'application

Lors de l'exécution de l'application, la fenêtre de gestion des stagiaires apparaît :



Si l'utilisateur clique sur l'item Modifier du menu contextuel, la fenêtre suivante apparaît :



5 Conclusion

Tout au long de ce cours, nous avons vu comment accéder et gérer les données de notre base de données, grâce à un nouveau modèle de données appelé Entity Data Model (EDM). Nous avons pu mettre en évidence les caractéristiques du Framework Entity, et la rapidité de développement d'une simple couche d'accès aux données dans Visual Studio.