Module 7: Accéder à une base de données

# **Contenu:**

|  |  |
| --- | --- |
|  | [Présentation du module](https://www.skillpipe.com/api/2.1/content/urn:uuid:21ae3267-8517-4025-bf9a-2dd83635ed12@2020-12-11T20:16:12Z/OPS/html/20483C07.html#P1) |
| **Leçon 1:** | [Création et utilisation de modèles de données d'entité](https://www.skillpipe.com/api/2.1/content/urn:uuid:21ae3267-8517-4025-bf9a-2dd83635ed12@2020-12-11T20:16:12Z/OPS/html/20483C07.html#P2) |
| **Leçon 2:** | [Interroger des données à l'aide de LINQ](https://www.skillpipe.com/api/2.1/content/urn:uuid:21ae3267-8517-4025-bf9a-2dd83635ed12@2020-12-11T20:16:12Z/OPS/html/20483C07.html#P3) |
| **Laboratoire:** | [Récupération et modification des données de note](https://www.skillpipe.com/api/2.1/content/urn:uuid:21ae3267-8517-4025-bf9a-2dd83635ed12@2020-12-11T20:16:12Z/OPS/html/20483C07.html#P4) |
|  | [Revue du module et points à retenir](https://www.skillpipe.com/api/2.1/content/urn:uuid:21ae3267-8517-4025-bf9a-2dd83635ed12@2020-12-11T20:16:12Z/OPS/html/20483C07.html#P5) |

# **Présentation du module**

De nombreuses applications nécessitent un accès aux données stockées dans une base de données. Microsoft® Visual Studio® 2017 et Microsoft .NET Framework fournissent des outils et des fonctionnalités que vous pouvez utiliser pour accéder, interroger et mettre à jour facilement les données.

Dans ce module, vous apprendrez à créer et à utiliser des modèles de données d'entité (EDM) et à interroger de nombreux types de données à l'aide de la requête intégrée au langage (LINQ).

### **Objectifs**

Après avoir terminé ce module, vous serez en mesure de:

|  |  |
| --- | --- |
| • | Créez, utilisez et personnalisez un EDM. |
| • | Interrogez les données à l'aide de LINQ. |

# Leçon 1: Création et utilisation de modèles de données d'entité

Les applications d'accès aux données ont traditionnellement été fastidieuses à développer. Ils contiennent souvent des requêtes écrites sous forme de chaînes de texte qui ne peuvent pas être vérifiées de type ou de syntaxe au moment de la compilation, et les résultats sont renvoyés sous forme d'enregistrements de données non typés. ADO.NET Entity Framework résout ces problèmes et simplifie le processus de développement d'applications d'accès aux données à l'aide d'EDM.

Dans cette leçon, vous apprendrez à utiliser les outils de données d'entité ADO.NET pour créer des EDM, à personnaliser les classes générées par les outils et à accéder aux entités du modèle généré.

## **Objectifs de la leçon**

Après avoir terminé cette leçon, vous serez en mesure de:

|  |  |
| --- | --- |
| • | Décrivez ADO.NET Entity Framework. |
| • | Utilisez les outils de modèle de données d'entité ADO.NET. |
| • | Personnalisez les classes générées. |
| • | Lisez et modifiez les données à l'aide de Entity Framework. |

## **Introduction à ADO.NET Entity Framework**

Historiquement, lorsque vous écrivez du code pour accéder aux données stockées dans une base de données, vous devez comprendre la structure des données dans la base de données et la manière dont elles sont liées. Souvent, il est stocké de manière normalisée, où les tables ne correspondent pas logiquement aux objets réels qu'elles représentent. ADO.NET Entity Framework vous permet de développer des applications qui ciblent un modèle conceptuel au lieu de la structure de base de données normalisée dans la couche de stockage.

ADO.NET Entity Framework fournit les éléments suivants: The ADO.NET Entity Framework provides the following:

|  |  |
| --- | --- |
| • | EDM. Il s'agit de modèles que vous pouvez utiliser pour mapper des tables de base de données et des requêtes sur des objets .NET Framework. |
| • | Langage de requête structuré d'entité (SQL). Il s'agit d'un langage de requête indépendant du stockage qui vous permet d'interroger et de manipuler des constructions EDM. |
| • | Services d'objets. Il s'agit de services qui vous permettent de travailler avec les objets CLR (Common Language Runtime) dans un modèle conceptuel.  Ces composants vous permettent de: |
| • | Écrivez du code sur un modèle conceptuel qui inclut des types prenant en charge l'héritage et les relations. |
| • | Mettez à jour les applications pour cibler un modèle de stockage différent sans réécrire ou redistribuer tout votre code d'accès aux données. |
| • | Écrivez un code standard qui ne dépend pas du système de stockage de données. |
| • | Écrivez un code d'accès aux données qui prend en charge la vérification de type et la vérification de syntaxe au moment de la compilation. |

Le modèle conceptuel avec lequel vous travaillez dans Entity Framework décrit la sémantique de la vue métier des données. Il définit les entités et les relations dans un sens commercial et est mappé au modèle logique des données sous-jacentes dans la source de données. Par exemple, dans une application de ressources humaines, les entités peuvent inclure des employés, des emplois et des succursales. Une entité est une description des éléments et de leurs propriétés, et ils sont liés par des relations, comme un employé étant lié à une succursale particulière.

**Lecture supplémentaire:**Pour plus d'informations sur ADO.NET Entity Framework, reportez-vous à la page ADO.NET Entity Framework à l'adresse<http://go.microsoft.com/fwlink/?LinkID=267806>.

## **Utilisation des outils de modèle de données d'entité ADO.NET**

Visual Studio 2017 fournit les outils de modèle de données d'entité que vous pouvez utiliser pour créer et mettre à jour des EDM dans vos applications. Il prend en charge à la fois la conception de base de données et la conception de code d'abord:

|  |  |
| --- | --- |
| • | Conception basée sur la base de données. Dans la conception basée sur la base de données, vous concevez et créez votre base de données avant de générer votre modèle. Ceci est couramment utilisé lorsque vous développez des applications sur une source de données existante; cependant, cela peut limiter la flexibilité de l'application à long terme. |
| • | Conception axée sur le code. Dans la conception code-first, vous concevez les entités de votre application, puis créez la structure de base de données autour de ces entités. Les développeurs préfèrent cette méthode car elle vous permet de concevoir votre application autour des fonctionnalités métier dont vous avez besoin. Cependant, en réalité, vous devez souvent travailler avec une source de données existante. |

**Utilisation des outils de modèle de données d'entité**

Visual Studio 2017 fournit les outils de modèle de données d'entité ADO.NET, qui incluent le concepteur de modèle de données d'entité pour créer graphiquement et mettre en relation des entités dans un modèle et trois assistants pour travailler avec des modèles et des sources de données. Le tableau suivant décrit les assistants.

| **magicien** | **La description** |
| --- | --- |
| Assistant de modèle de données d'entité | Vous permet de générer un nouveau modèle conceptuel à partir d'une source de données existante à l'aide de la méthode de conception basée sur la base de données. |
| Assistant de mise à jour du modèle | Vous permet de mettre à jour un modèle conceptuel existant avec les modifications apportées à la source de données sur laquelle il est basé. |
| Générer l'assistant de base de données | Vous permet de générer une base de données à partir d'un modèle conceptuel que vous avez conçu dans Entity Data Model Designer à l'aide de la méthode de conception code-first. |

Lorsque vous créez un modèle à l'aide de l'Assistant Modèle de données d'entité, le modèle s'ouvre dans le volet Concepteur, affichant les entités que l'Assistant a générées et les relations entre elles. Vous pouvez utiliser ce volet pour ajouter, modifier et supprimer des entités et des relations.

Par défaut, lorsque vous créez un modèle à partir d'une base de données, Entity Designer génère automatiquement les mappages de la source de données vers le modèle conceptuel. Vous pouvez afficher, modifier et supprimer ces mappages dans le volet Détails du mappage.

**Lecture supplémentaire:**Pour plus d'informations sur les outils de modèle de données d'entité, reportez-vous à la page Outils de modèle de données d'entité ADO.NET à l'adresse<http://go.microsoft.com/fwlink/?LinkID=267807>.

## **Démonstration: création d'un modèle de données d'entité**

Dans cette démonstration, vous utiliserez l'assistant de données d'entité pour générer un EDM pour une base de données existante.

### **Démonstration étapes**

Vous trouverez les étapes dans la section «Démonstration: création d'un modèle de données d'entité» sur la page suivante:<https://github.com/MicrosoftLearning/20483-Programming-in-C-Sharp/blob/master/Instructions/20483C_MOD07_DEMO.md>.

## **Personnalisation des classes générées**

Lorsque vous utilisez l'Assistant Modèle de données d'entité pour créer un modèle, il génère automatiquement des classes qui exposent les entités du modèle à votre code d'application. Ces classes contiennent des propriétés qui permettent d'accéder aux propriétés des entités.

L'exemple de code suivant montre une classe Employee générée par l'Assistant Modèle de données d'entité.

**Classes générées par l'assistant**

espace de noms FourthCoffee.Employees

{

en utilisant le système;

using System.Collections.Generic;

Employé de classe partielle publique

{

public int EmployeeID {get; ensemble; }

chaîne publique FirstName {get; ensemble; }

chaîne publique LastName {get; ensemble; }

public Nullable <System.DateTime> DateOfBirth {get; ensemble; }

public Nullable <int> Branch {get; ensemble; }

public Nullable <int> JobTitle {obtenir; ensemble; }

public virtuel Branch Branch1 {get; ensemble; }

public virtuel JobTitle JobTitle1 {get; ensemble; }

}

}

Vous pouvez trouver que vous souhaitez ajouter une logique métier personnalisée aux classes d'entité; toutefois, si à tout moment dans le futur vous exécutez l'Assistant de modèle de mise à jour, les classes seront régénérées et votre code sera écrasé. Cependant, les classes générées sont définies comme des classes partielles.

Le mot-clé partial permet de diviser les définitions de certaines conceptions en plusieurs parties, sur plusieurs fichiers source. Les classes, structures et interfaces peuvent toutes être partielles. Lors de l'utilisation du modificateur partial, toutes les définitions de ce type doivent contenir le mot-clé partial, et toutes doivent exister dans le même espace de noms au sein du même assembly. Les types partiels ne peuvent pas être répartis sur plusieurs fichiers .dll et .exe. Lorsque le compilateur rencontre une définition partielle, il recherche dans l'assembly d'autres du même type et combine tout leur contenu en un seul type. Un type peut être divisé en autant ou aussi peu de définitions partielles que vous le souhaitez. Une seule définition partielle est tout à fait valable.

L'assistant de modèle de données d'entité produira toujours ses classes comme partielles; par conséquent, vous pouvez les étendre pour ajouter des fonctionnalités personnalisées aux classes.

Par exemple, si vous avez une propriété de date de naissance dans votre modèle, vous pouvez écrire une méthode GetAge dans une classe partielle pour activer un calcul d'exécution de l'âge de l'employé.

L'exemple de code suivant montre comment vous pouvez ajouter une logique métier à une classe générée à l'aide d'une classe partielle.

**Ajout de logique métier dans une classe partielle**

Employé de classe partielle publique

{

public int GetAge ()

{

DateHeure DOB = (DateHeure) \_DateOfBirth;

TimeSpan difference = DateTime.Now.Subtract (DOB);

int ageInYears = (int) (difference.Days / 365.25);

return ageInYears;

}

}

**Lecture supplémentaire:**Pour plus d'informations sur les classes partielles, reportez-vous aux classes et méthodes partielles (Guide de programmation C #) à l'adresse<http://go.microsoft.com/fwlink/?LinkID=267808>.

## **Lecture et modification de données à l'aide de Entity Framework**

Les fichiers de code générés automatiquement pour un modèle contiennent également une classe partielle qui hérite de la classe System.Data.Entity.DbContext. La classe DbContext fournit des fonctionnalités pour interroger et travailler avec des données d'entité en tant qu'objets. Il contient un constructeur par défaut qui initialise la classe à l'aide de la chaîne de connexion que l'Assistant génère dans le fichier de configuration de l'application. Cela définit la connexion de données et la définition du modèle à utiliser. La classe DbContext contient également une propriété DbSet qui expose une classe DbSet (TEntity) pour chaque entité de votre modèle. La classe DbSet (TEntity) représente un jeu d'entités typé que vous pouvez utiliser pour lire, créer, mettre à jour et supprimer des données.

L'exemple de code suivant montre la classe du modèle FourthCoffeeEntities.

**Classe FourthCoffeeEntities**

classe partielle publique FourthCoffeeEntities: DbContext

{

public FourthCoffeeEntities (): base ("nom = FourthCoffeeEntities")

{

}

public DbSet <Branch> Branches {get; ensemble; }

public DbSet <Employee> Employees {get; ensemble; }

public DbSet <JobTitle> JobTitles {get; ensemble; }

}

Pour utiliser le jeu d'entités typé, vous créez une instance de la classe DbContext, puis accédez aux propriétés à l'aide de la notation par points standard.

L'exemple de code suivant montre comment lire et mettre à jour des données à l'aide de la classe DbSet (TEntity).

**Lecture des données**

FourthCoffeeEntities DBContext = nouveau FourthCoffeeEntities ();

// Imprime une liste d'employés.

foreach (FourthCoffee.Employees.Employee emp dans DBContext.Employees)

{

Console.WriteLine ("{0} {1}", emp.FirstName, emp.LastName);

}

La classe DbSet (TEntity) implémente l'interface IEnumerable qui fournit un certain nombre de méthodes d'extension qui vous permettent de localiser facilement des données spécifiques dans la source. Par exemple, la méthode d'extension First localise la première correspondance pour la condition spécifiée, telle qu'un nom de famille de Prescott.

L'exemple de code suivant montre comment utiliser la méthode d'extension First pour localiser un employé, puis comment mettre à jour les données à l'aide de la notation par points standard.

**Localisation et modification des données**

// Mettre à jour l'employé avec un nom de famille «Prescott».

var emp = DBContext.Employees.First (e => e.LastName == "Prescott");

si (emp! = nul)

{

emp.LastName = "Forsyth";

}

**Lecture supplémentaire:**Pour plus d'informations sur la classe DbSet (Entity), reportez-vous à la page Classe DbSet (Entity) à l'adresse

<https://aka.ms/moc-20483c-m7-pg1>.

Pour plus d'informations sur les méthodes énumérables, reportez-vous à la page Méthodes énumérables à l'adresse<https://aka.ms/moc-20483c-m7-pg2>.

Après avoir modifié les données dans le modèle, vous devez explicitement appliquer ces modifications aux données de la source de données. Vous pouvez le faire en appelant la méthode SaveChanges de l'objet ObjectContext.

L'exemple de code suivant montre comment utiliser la méthode SaveChanges.

**Modifications persistantes de la base de données**

DBContext.SaveChanges ();

## **Démonstration: lecture et modification de données dans un EDM**

Dans cette démonstration, vous utiliserez la classe ObjectSet (TEntity) pour lire et modifier des données dans un EDM.

### **Démonstration étapes**

Vous trouverez les étapes dans la démonstration: lecture et modification de données dans une section EDM sur la page suivante:<https://github.com/MicrosoftLearning/20483-Programming-in-C-Sharp/blob/master/Instructions/20483C_MOD07_DEMO.md>.

# Leçon 2: Interroger des données à l'aide de LINQ

Au lieu d'utiliser Entity Framework pour interroger des données, vous pouvez utiliser LINQ. Cela prend également en charge la vérification de la syntaxe et la vérification de type au moment de la compilation et utilise également Microsoft IntelliSense® dans Visual Studio. LINQ définit une plage d'opérateurs de requête standard qui vous permettent de récupérer exactement les données dont vous avez besoin de manière déclarative.

Dans cette leçon, vous apprendrez à interroger des données et à utiliser des méthodes anonymes et à forcer l'exécution de requête pour remplacer le comportement d'exécution de requête différée par défaut.

## **Objectifs de la leçon**

Après avoir terminé cette leçon, vous serez en mesure de:

|  |  |
| --- | --- |
| • | Interrogez les données. |
| • | Interrogez des données à l'aide de types anonymes. |
| • | Forcer l'exécution de la requête. |

## **Interroger des données**

Vous pouvez utiliser LINQ pour interroger des données à partir d'un large éventail de sources de données, y compris les collections .NET Framework, les bases de données Microsoft SQL Server®, les ensembles de données ADO.NET et les documents XML. En fait, vous pouvez l'utiliser pour interroger n'importe quelle source de données qui implémente l'interface IEnumerable.

La syntaxe de toutes les requêtes LINQ a la même base, comme suit:

**de**<noms de variables> dans <source de données>

**sélectionner**<noms de variables>

Cependant, vous pouvez personnaliser cette syntaxe de plusieurs manières pour récupérer exactement les données dont vous avez besoin dans le format souhaité. Les exemples de code suivants utilisent tous LINQ to Entities pour interroger des données dans un EDM; cependant, la syntaxe de la requête elle-même ne change pas si vous utilisez un autre type de source de données.

**Sélection des données**

L'exemple de code suivant montre comment utiliser une clause de sélection simple pour renvoyer toutes les données dans une seule entité.

**Utilisation d'une clause de sélection**

IQueryable <Employee> emps = from e dans DBContext.Employees

sélectionnez e;

Le type de données renvoyé par la requête est un <Employee> IQueryable, vous permettant de parcourir les données renvoyées.

**Filtrage des données par ligne**

L'exemple de code suivant montre comment utiliser le mot clé where pour filtrer les données renvoyées par ligne afin de ne contenir que les employés dont le nom de famille est Prescott.

**Utilisation d'une clause where**

string \_LastName = "Prescott";

IQueryable <Employee> emps = from e dans DBContext.Employees

où e.LastName == \_LastName

sélectionnez e;

**Filtrage des données par colonne**

L'exemple de code suivant montre comment déclarer un nouveau type dans lequel stocker un sous-ensemble de colonnes renvoyé par la requête; dans ce cas, uniquement les propriétés FirstName et LastName de l'entité Employee.

**Utilisation d'une nouvelle classe pour renvoyer un sous-ensemble de colonnes**

classe privée FullName

{

chaîne publique Prénom {get; ensemble; }

chaîne publique Nom de famille {get; ensemble; }

}

private void FilteringDataByColumn ()

{

IQueryable <FullName> names = from e dans DBContext.Employees

sélectionnez un nouveau nom complet {Forename = e.FirstName, Name = e.LastName};

}

**Travailler avec les résultats**

Pour travailler ensuite avec les données renvoyées par l'une de ces requêtes, vous utilisez la notation par points pour accéder aux propriétés des membres du type IQueryable <>, comme le montre l'exemple de code suivant.

**Accéder aux données renvoyées**

foreach (var emp dans emps)

{

Console.WriteLine ("{0} {1}", emp.FirstName, emp.LastName);

}

## **Démonstration: interroger des données**

Dans cette démonstration, vous utiliserez LINQ to Entities pour interroger des données.

### **Démonstration étapes**

Vous trouverez les étapes dans la section Démonstration: Interroger des données sur la page suivante:<https://github.com/MicrosoftLearning/20483-Programming-in-C-Sharp/blob/master/Instructions/20483C_MOD07_DEMO.md>.

## **Interroger des données à l'aide de types anonymes**

Dans les exemples de la rubrique et de la démonstration précédentes, les données de retour étaient toujours stockées dans une variable IQueryable <Type> fortement typée; cependant, dans le scénario de filtrage par colonne, il est nécessaire de définir le type contenant un sous-ensemble de colonnes avant de définir la requête. Bien que ce soit une manière parfaitement valable de travailler, il peut devenir fastidieux de définir explicitement plusieurs classes.

Vous pouvez utiliser des types anonymes pour stocker les données renvoyées en déclarant le type de retour en tant que variable locale implicitement typée, une var, et en utilisant le mot clé new dans la clause select pour créer l'instance du type.

**Filtrage des données par colonne**

L'exemple de code suivant montre comment utiliser le type de données var et le mot clé new dans la clause select pour filtrer les données renvoyées par colonne.

**Utilisation d'un type anonyme pour renvoyer un sous-ensemble de colonnes**

var names = from e dans DBContext.Employees

sélectionnez nouveau {e.FirstName, e.LastName};

Les types anonymes vous permettent d'effectuer des requêtes plus complexes dans LINQ.

**Regroupement des données**

L'exemple de code suivant montre comment utiliser une clause de groupe pour regrouper les employés renvoyés par leur ID d'intitulé de poste.

**Utilisation d'une clause de groupe**

var emps = from e dans DBContext.Employees

grouper e par e.JobTitle dans eGroup

sélectionnez nouveau {Job = eGroup.Key, Names = eGroup};

**Agrégation des données**

L'exemple de code suivant montre comment utiliser une clause de groupe avec une fonction d'agrégation pour compter le nombre d'employés avec chaque titre de poste.

**Utilisation d'une clause de groupe avec une fonction d'agrégation**

var emps = from e dans DBContext.Employees

grouper e par e.JobTitle dans eGroup

sélectionnez new {Job = eGroup.Key, CountOfEmployees = eGroup.Count ()};

**Naviguer dans les données**

L'exemple de code suivant montre comment utiliser les propriétés de navigation pour récupérer des données de l'entité Employees et de l'entité JobTitles associée.

**Utilisation de la notation par points pour parcourir les entités associées**

var emps = from e dans DBContext.Employees

sélectionner nouveau

{

FirstName = e.FirstName, LastName = e.LastName, Job = e.JobTitle1.Job

};

## **Démonstration: interroger des données à l'aide de types anonymes**

Dans cette démonstration, vous utiliserez LINQ to Entities pour interroger des données à l'aide de types anonymes.

### **Démonstration étapes**

Vous trouverez les étapes dans la démonstration: interroger des données à l'aide de types anonymes sur la page suivante: You will find the steps in the Demonstration: Querying Data by Using Anonymous Types section on the following page:<https://github.com/MicrosoftLearning/20483-Programming-in-C-Sharp/blob/master/Instructions/20483C_MOD07_DEMO.md>.

## **Forcer l'exécution de la requête**

Par défaut, lorsque vous définissez une requête LINQ qui renvoie une séquence de valeurs, elle n'est exécutée que lorsque vous essayez réellement d'utiliser certaines des données renvoyées. Cette fonctionnalité est connue sous le nom d'exécution de requête différée et garantit que vous pouvez créer une requête pour récupérer des données dans un scénario multi-utilisateur et savoir que chaque fois qu'elle est exécutée, vous recevrez les dernières informations.

Dans l'exemple de code suivant, la requête n'est réellement exécutée qu'au début du bloc foreach.

**Exécution de requête différée**

IQueryable <Employee> emps = from e dans DBContext.Employees

sélectionnez e;

foreach (var emp dans emps)

{

Console.WriteLine ("{0} {1}", emp.FirstName, emp.LastName);

}

Inversement, lorsque vous définissez une requête LINQ qui renvoie une valeur singleton, par exemple, une fonction Moyenne, Nombre ou Max, la requête est exécutée immédiatement. Ceci est appelé exécution immédiate de la requête et est nécessaire dans le scénario de résultat singleton car la requête doit produire une séquence pour calculer le résultat singleton.

Vous pouvez remplacer le comportement d'exécution de requête différée par défaut pour les requêtes qui ne produisent pas de résultat singleton en appelant l'une des méthodes suivantes sur la requête:

|  |  |
| --- | --- |
| • | **ToArray** |
| • | **ToDictionary** |
| • | **Lister** |

Dans l'exemple de code suivant, la requête est exécutée immédiatement après sa définition.

**Forcer l'exécution de la requête**

IList <Employee> emps = (de e dans DBContext.Employees

sélectionnez e;). ToList ()

foreach (var emp dans emps)

{

Console.WriteLine ("{0} {1}", emp.FirstName, emp.LastName);

}

## **Démonstration: Récupération et modification du laboratoire de données de notes**

Dans cette démonstration, vous découvrirez les tâches que vous effectuerez dans le laboratoire pour ce module.

### **Démonstration étapes**

Vous trouverez les étapes dans la section Démonstration: Récupération et modification des données de notes sur la page suivante:<https://github.com/MicrosoftLearning/20483-Programming-in-C-Sharp/blob/master/Instructions/20483C_MOD07_DEMO.md>.

# **Atelier: Récupération et modification des données de note**

### **Scénario**

Vous avez été invité à mettre à niveau l'application prototype pour utiliser une base de données SQL Server existante. Vous commencez par travailler avec une base de données stockée sur votre ordinateur local et décidez d'utiliser l'assistant de modèle de données d'entité pour générer un EDM pour accéder aux données. Vous devrez mettre à jour le code d'accès aux données pour la section Notes de l'application, pour afficher les notes attribuées à un étudiant et pour permettre aux utilisateurs d'attribuer de nouvelles notes. Vous décidez également d'incorporer une logique de validation dans l'EDM pour vous assurer que les étudiants ne peuvent pas être affectés à une classe complète et que les données que les utilisateurs saisissent lorsqu'ils attribuent de nouvelles notes sont conformes aux valeurs requises.

### **Objectifs**

Après avoir terminé cet atelier, vous serez en mesure de:

|  |  |
| --- | --- |
| • | Créez un EDM à partir d'une base de données existante. |
| • | Mettez à jour les données à l'aide de .NET Entity Framework. |
| • | Étendez un EDM pour valider les données. |

##### **Configuration du laboratoire**

Durée estimée: 75 minutes

Vous trouverez les étapes de haut niveau sur la page suivante:<https://github.com/MicrosoftLearning/20483-Programming-in-C-Sharp/blob/master/Instructions/20483C_MOD07_LAB_MANUAL.md>.

Vous trouverez les étapes détaillées sur la page suivante:<https://github.com/MicrosoftLearning/20483-Programming-in-C-Sharp/blob/master/Instructions/20483C_MOD07_LAK.md>.

### **Exercice 1: Création d'un modèle de données d'entité à partir de la base de données de la School of Fine Arts**

##### **Scénario**

Dans cet exercice, vous allez utiliser l'assistant de modèle de données d'entité pour générer un EDM à partir de la base de données SQL Server SchoolGradesDB, puis passer en revue le modèle et le code que l'assistant génère.

**Résultat**: Une fois cet exercice terminé, l'application prototype doit inclure un EDM que vous pouvez utiliser pour accéder à la base de données de l'École des beaux-arts.

### **Exercice 2: Mettre à jour les données des élèves et des notes à l'aide de Entity Framework**

##### **Scénario**

Dans cet exercice, vous allez ajouter des fonctionnalités à l'application prototype pour afficher les notes d'un utilisateur. Les informations de note dans la base de données stockent l'ID du sujet pour une note, vous ajouterez donc du code à l'application pour le convertir en nom de sujet à des fins d'affichage. Vous allez également ajouter du code pour afficher la vue Ajouter une note à l'utilisateur, puis utiliser les informations que l'utilisateur saisit pour ajouter une note pour l'élève actuel. Enfin, vous exécuterez l'application et vérifierez que l'affichage des notes et la fonctionnalité d'ajout de notes fonctionnent comme prévu.

**Résultat**: Une fois cet exercice terminé, les utilisateurs verront les notes de l'élève actuel et ajouteront de nouvelles notes.

### **Exercice 3: Extension du modèle de données d'entité pour valider les données**

##### **Scénario**

Dans cet exercice, vous mettrez à jour l'application pour valider les données saisies par l'utilisateur.

Tout d'abord, vous allez ajouter du code pour vérifier si une classe est pleine avant d'inscrire un étudiant et lever une exception si c'est le cas. Ensuite, vous ajouterez un code de validation pour vérifier qu'un utilisateur entre une date valide et une note d'évaluation lors de l'ajout d'une note à un étudiant. Enfin, vous exécuterez l'application et vérifierez que la validation des données fonctionne comme prévu.

**Résultat**: Une fois cet exercice terminé, l'application lèvera et gérera les exceptions lorsque des données non valides sont saisies.

# **Revue du module et points à retenir**

Dans ce module, vous avez appris à créer et à utiliser des EDM et à interroger de nombreux types de données à l'aide de LINQ.

### **Question (s) de révision**