

# ADO.NET

COGNITIC - MORRE THIERRY ©2020



1

- Introduction
- ADO.Net et .Net 5
- Configurer « SQL Server Express LocalDB »
- Projet base de données SQL Serveur
- Les espaces de noms
- Notion de connexions et de commandes
- Sélection de données
- Insertion, modification et suppression de données
- Cas particulier : utilisation du mot clé « output »
- L'injection SQL
- Les requêtes paramétrées
- Appeler une procédure stockée
- Gestion des transactions
- Exercice
- Design Pattern : Abstract Factory
- Abstract Factory et ADO.Net
- Exercice
- Références

## C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

2

2

# INTRODUCTION

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

3

3

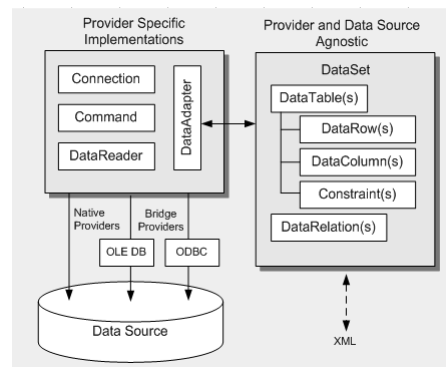
## INTRODUCTION

ADO.NET est un ensemble de classes qui exposent les services d'accès aux données pour les programmeurs utilisant le Framework .NET.

ADO.NET propose un large ensemble de composants pour la création d'applications distribuées avec partage de données.

Partie intégrante du .NET Framework, il permet d'accéder à des données relationnelles ou XML.

Durant ce cours, nous viserons plus le côté d'accès base de données relationnelles.



COGNITIC - MORRE THIERRY ©2020

4

4

# ADO.NET ET .NET 5

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

5

5

## CRÉATION D'UN PROJET

.NET – A unified platform



COGNITIC - MORRE THIERRY ©2020

6

Les Frameworks .Net Core et .Net 5 se reposent sur un système de librairies dites « standard ».

Pour rappel, ces librairies sont disponibles sur « Nuget » au travers de l'outil de gestion de packages.

De ce fait, si nous utilisons ces Frameworks, nous devons installer le package « System.Data.SqlClient » si nous voulons interagir avec SQL Server.

6

# CONFIGURER « SQL SERVER EXPRESS LOCALDB »

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

7

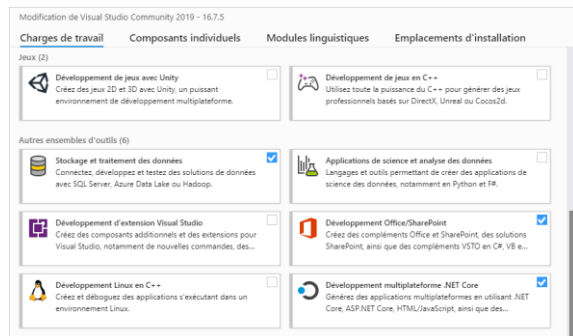
7

## CONFIGURER « SQL SERVER EXPRESS LOCALDB »

Lors de l'installation de Visual Studio, si nous cochons « Stockage et traitement des données », ce dernier installe un « Sql Server Express » que nous pourrions utiliser pour notre développement ainsi que des types de projets permettant de créer des bases de données pour « SQL Server ».

« SQL Server Express » est une version allégée de « SQL Server » dont on a que le moteur de bases de données, et dont chaque base de données est limitée en taille à 10 Go.

Il n'est donc pas nécessaire de posséder une licence ou d'installer un « SQL Server », dont l'installation peut paraître compliquée pour les non initiés, pour créer et de manipuler nos bases de données.



COGNITIC - MORRE THIERRY ©2020

8

8

# CONFIGURER « SQL SERVER EXPRESS LOCALDB »

Une instance équivaut à une installation de SQL Server, chaque instance possède ses propres bases de données et sa propre sécurité.

Afin de nous permettre créer une instance de « Sql Server Express » sur notre machine, nous allons utiliser la « Developer Powershell for VS 20xx ».

Cette console, nous donne accès à toute une série d'outils dont l'exécutable « SqlLocalDB.exe » qui va nous permettre de gérer nos instances avec la commande structurée comme suit :

SqlLocalDB.exe <action> <instance>

Ensuite une fois créée, nous pouvons utiliser l'« explorateur d'objets SQL Server de Visual Studio » ou « Sql Server Management Studio » pour nous connecter et gérer nos bases de données.

Action	Description
Create	Crée une nouvelle instance de SQL Server Express.
Delete	Supprime l'instance de SQL Server Express
Start	Démarre l'instance de SQL Server Express
Stop	Arrête l'instance de SQL Server Express
Info	Fournit des informations sur l'instance de SQL Server Express

Télécharger SQL Server Management Studio

COGNITIC - MORRE THIERRY ©2020

9

9

# CONFIGURER « SQL SERVER EXPRESS LOCALDB »

```
PS D:\Briareos\Source> SqlLocalDB.exe create MonInstance
Instance de base de données locale « MonInstance » créée avec la version 13.1.4001.0.
PS D:\Briareos\Source>
PS D:\Briareos\Source> SqlLocalDB.exe info MonInstance
Nom : MonInstance
Version : 13.1.4001.0
Nom partagé :
Propriétaire : AM-BRIAREOS\Brian
Création automatique : Non
État : Arrêté
Dernière heure de début : 20/10/2020 09:47:13
Nom de canal de l'instance :
PS D:\Briareos\Source> SqlLocalDB.exe start MonInstance
Instance de base de données locale « MonInstance » démarrée.
PS D:\Briareos\Source>
```

COGNITIC - MORRE THIERRY ©2020

```
PS D:\Briareos\Source> SqlLocalDB.exe stop MonInstance
Instance de base de données locale « MonInstance » arrêtée.
PS D:\Briareos\Source>
PS D:\Briareos\Source> SqlLocalDB.exe delete MonInstance
Instance de base de données locale « MonInstance » supprimée.
PS D:\Briareos\Source>
```

10

10

# PROJET BASE DE DONNÉES SQL SERVEUR

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

11

11

- Création d'un projet
- Configuration
- Créer des tables
- Créer une vue
- Créer une procédure stockée
- Créer un script post-déploiement
- Déployer sa base de données
- Importer une base de données existante
- Exercice

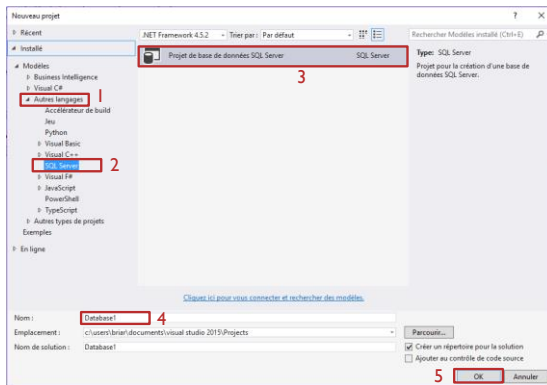
## PROJET BASE DE DONNÉES SQL SERVEUR

COGNITIC - MORRE THIERRY ©2020

12

12

## CRÉATION D'UN PROJET



Tout d'abord, sélectionnons le type de projet :

1. Autres langages
2. SqlServer
3. Projet de base de données SQL Serveur
4. Nommons le
5. Validons

COGNITIC - MORRE THIERRY ©2020

13

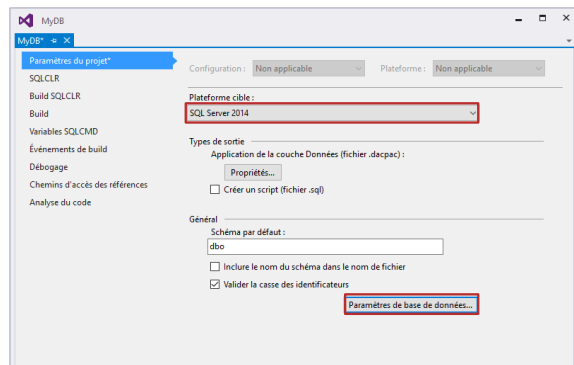
13

## CONFIGURATION : CHOIX DE LA VERSION DE SQL SERVEUR

Pour configurer notre projet et choisir la version du SQL serveur sur laquelle nous allons travailler, cliquons droit sur le projet puis Propriétés.

Dans cette fenêtre, la « plateforme cible » détermine sur quelle version de SQL Server nous allons travailler.

En cliquant ensuite sur « Paramètres de la base de données... », nous pourrions paramétrer les options de la base de données.

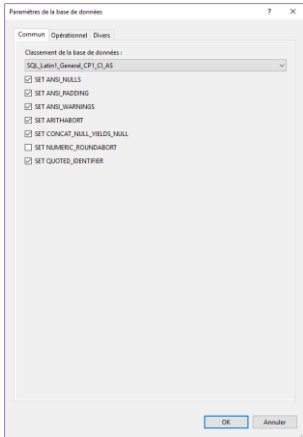


COGNITIC - MORRE THIERRY ©2020

14

14

# CONFIGURATION : PARAMÉTRER LA BASE DE DONNÉES



COGNITIC - MORRE THIERRY ©2020

Le premier onglet permet de choisir la collation de la base de données ainsi que de configurer quelques options de la base de données.

- [SET ANSI\\_NULLS](#)
- [SET ANSI\\_PADDING](#)
- [SET ANSI\\_WARNINGS](#)
- [SET ARITHABORT](#)
- [SET CONCAT\\_NULL\\_YIELDS\\_NULL](#)
- [SET NUMERIC\\_ROUNDABORT](#)
- [SET QUOTED\\_IDENTIFIER](#)

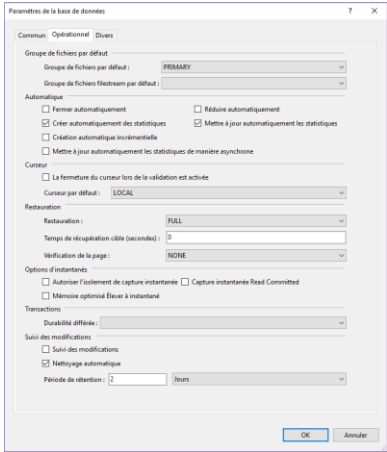
Pour plus d'informations, cliquez sur les liens

15

15

# CONFIGURATION : PARAMÉTRER LA BASE DE DONNÉES

Dans ce deuxième onglet, nous pouvons configurer les options de la base de données telles que le groupe de fichier par défaut ou le mode de restauration pour ne citer que les principales.



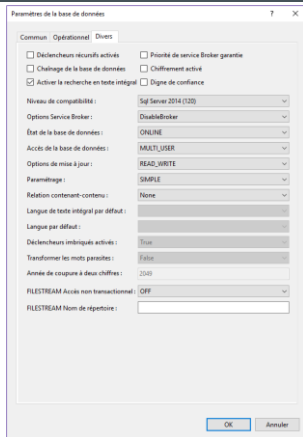
COGNITIC - MORRE THIERRY ©2020

16

16



## CONFIGURATION : PARAMÉTRER LA BASE DE DONNÉES



Enfin dans ce dernier onglet nous pourrons spécifier des options plus avancées comme les triggers récursifs, le niveau de compatibilité, le « Filestream », etc.

COGNITIC - MORRE THIERRY ©2020

17

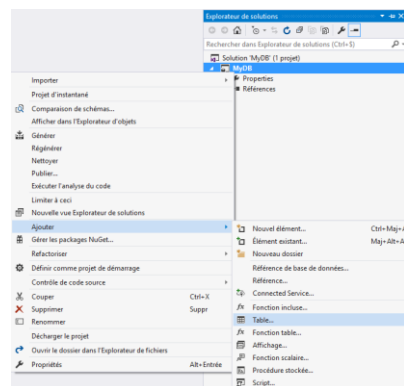
17

## CRÉER UNE TABLE

Commençons à créer nos tables.

Pour ce faire :

- Cliques droit sur le projet (ou sur un répertoire du projet)
- Ajouter
- Table

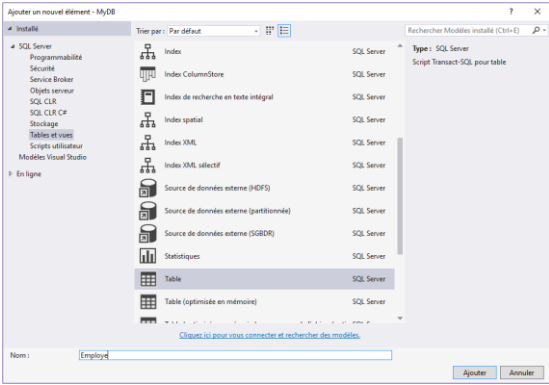


COGNITIC - MORRE THIERRY ©2020

18

18

# CRÉER UNE TABLE



Nommons la et validons.

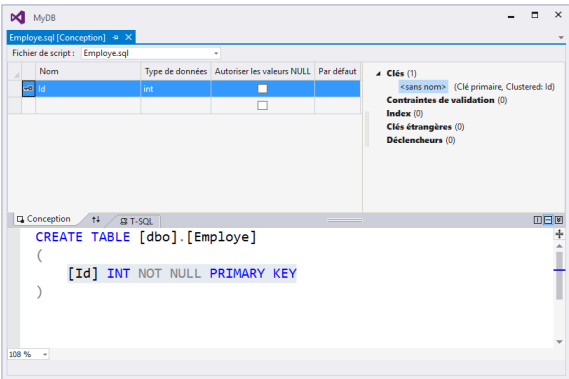
COGNITIC - MORRE THIERRY ©2020

19

19

# CRÉER UNE TABLE

Une fois la table créée, nous pouvons ajouter nos champs soit via l'interface graphique, soit en modifiant le T-SQL en dessous.

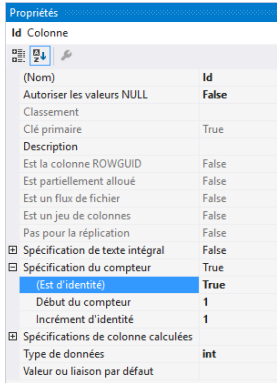


COGNITIC - MORRE THIERRY ©2020

20

20

# CRÉER UNE TABLE : AJOUTER UNE COLONNE AUTO INCRÉMENTÉE



Pour spécifier un champs auto-incrémenté, il nous faudra aller dans la partie propriété (F4) en ayant sélectionné le bon champs.

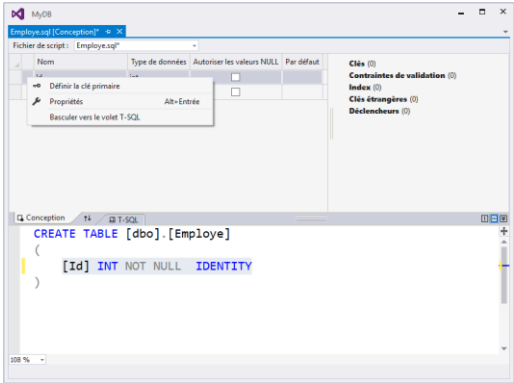
Ensuite, dans la section « Spécification du compteur » Passé l'état de la propriété « Est d'identité » à « True ».

Ou nous spécifions « Identity » dans la partie T-SQL.

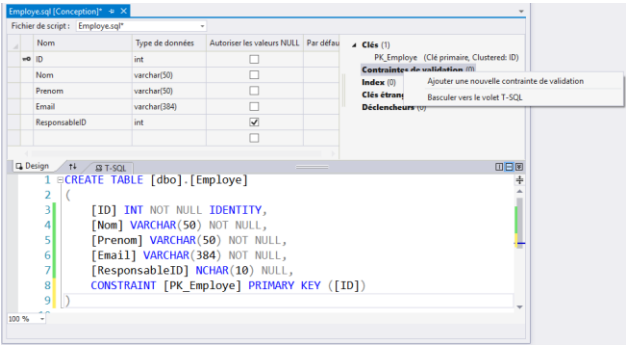
# CRÉER UNE TABLE : AJOUTER UNE CLÉ PRIMAIRE

Pour spécifier une clé primaire, rien de plus simple. Soit on choisi le ou les champs dans la partie graphique, on clique droit et on choisi « Définir la clé primaire ».

Soit on l'ajoute en T-SQL directement.



# CRÉER UNE TABLE : AJOUTER UNE CONTRAINTE DE VALIDATION



Pour les contraintes de validations (Check), dans la partie droite clique droit sur « Contraintes de validation » et « ajouter une nouvelle... »

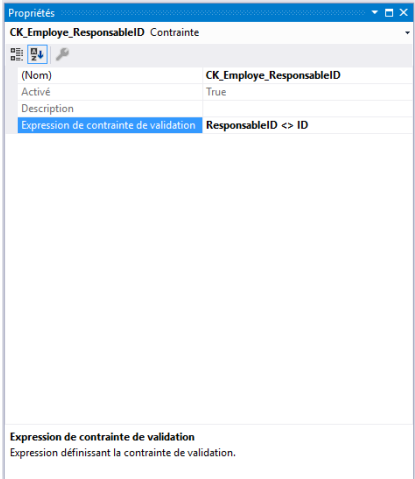
COGNITIC - MORRE THIERRY ©2020

23

23

# CRÉER UNE TABLE : AJOUTER UNE CONTRAINTE DE VALIDATION

Ensuite, nous pouvons la nommer et lui donner l'expression.  
Encore une fois nous pouvons également directement spécifier la contrainst dans la partie T-SQL

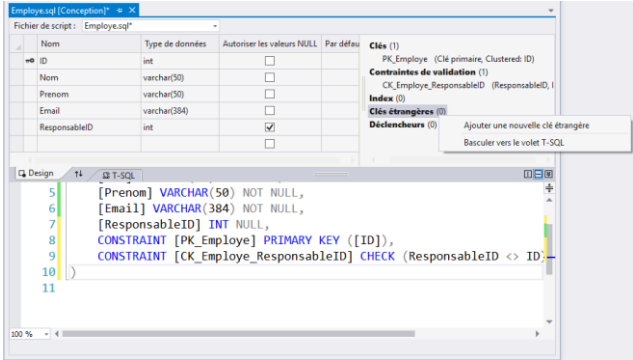


COGNITIC - MORRE THIERRY ©2020

24

24

# CRÉER UNE TABLE : AJOUTER UNE CLÉ ÉTRANGÈRE



Pour ajouter une clé étrangère, c'est le même principe que pour la contrainte de validations.

Clique droit sur « Clés étrangères » et « Ajouter une nouvelle... »

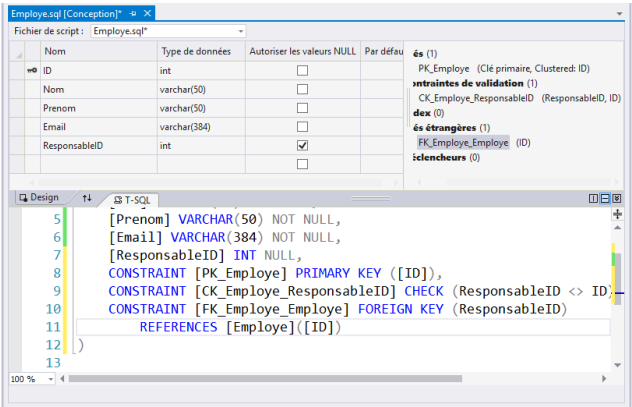
COGNITIC - MORRE THIERRY ©2020

25

25

# CRÉER UNE TABLE : AJOUTER UNE CLÉ ÉTRANGÈRE

À l'exception que nous devons spécifier la table et les champs ciblés dans la partie T-SQL obligatoirement.

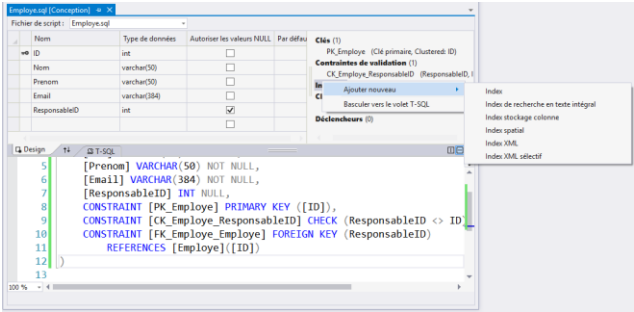


COGNITIC - MORRE THIERRY ©2020

26

26

# CRÉER UNE TABLE : AJOUTER UN INDEX



Pour ajouter un index sur une table, clique droit sur « Index » et « Ajouter un nouveau... » puis « Index »

COGNITIC - MORRE THIERRY ©2020

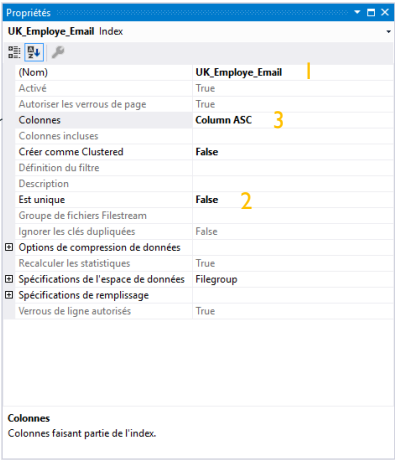
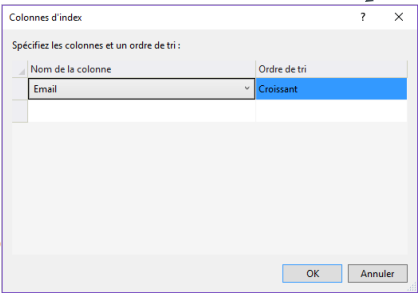
27

27

# CRÉER UNE TABLE : AJOUTER UN INDEX

Depuis la fenêtre de propriété, nous pourrons :

- 1. Spécifier son nom
- 2. Définir s'il est unique
- 3. Définir la ou les colonne(s) ciblée(s)

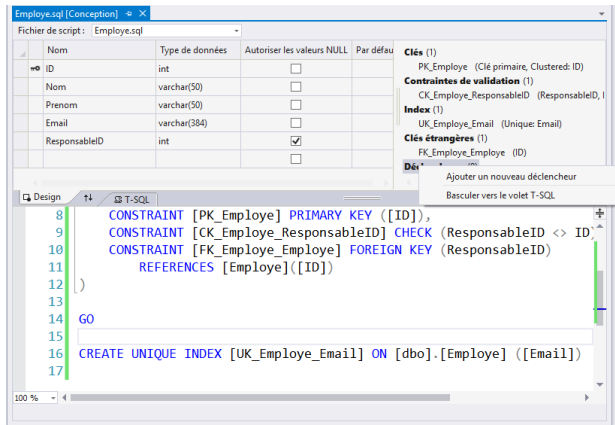


COGNITIC - MORRE THIERRY

28

28

## CRÉER UNE TABLE : AJOUTER UN DÉCLENCHEUR



COGNITIC - MORRE THIERRY ©2020

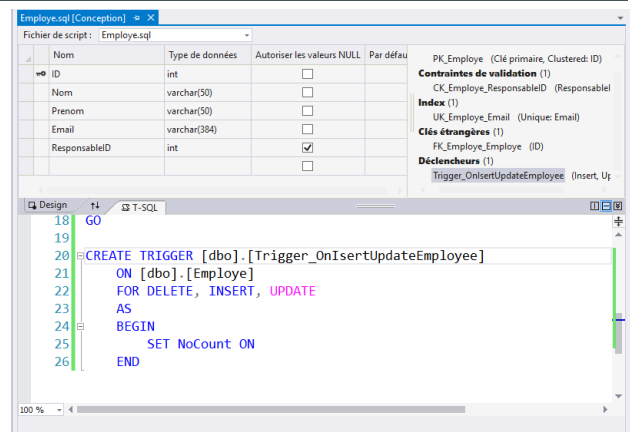
Pour ajouter un trigger sur une table, clique droit sur « Déclencheurs » et « Ajouter un nouveau... »

29

29

## CRÉER UNE TABLE : AJOUTER UN DÉCLENCHEUR

Spécifions le code T-SQL de ce dernier.

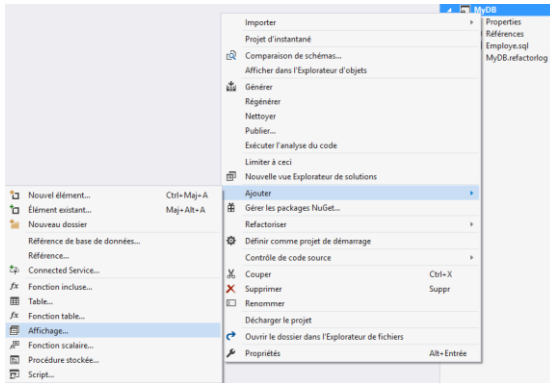


COGNITIC - MORRE THIERRY ©2020

30

30

## CRÉER UNE VUE



Pour ajouter une vue :

- Cliquons droit sur le projet (ou sur un répertoire du projet)
- Ajouter
- Affichage

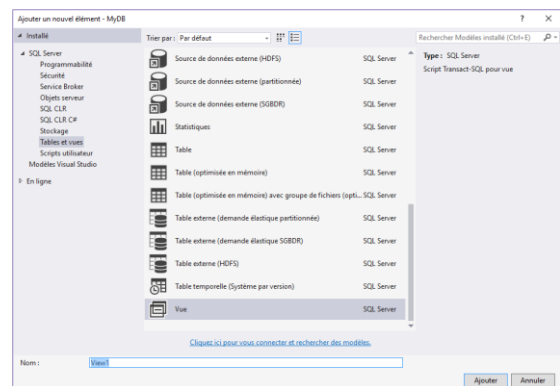
COGNITIC - MORRE THIERRY ©2020

31

31

## CRÉER UNE VUE

Nommons la et validons.



COGNITIC - MORRE THIERRY ©2020

32

32



## CRÉER UNE VUE

Enfin définissons la requête « Select » pour la générer la vue.

```
CREATE VIEW [dbo].[V_Employe]
AS SELECT e1.ID,
          e1.Nom,
          e1.Prenom,
          e1.Email,
          e1.ResponsableID,
          e2.Nom RespNom,
          e2.Prenom RespPrenom,
          e2.Email RespEmail
From [Employee] e1
join Employee e2 on e1.ResponsableID = e2.ID
```

COGNITIC - MORRE THIERRY ©2020

33

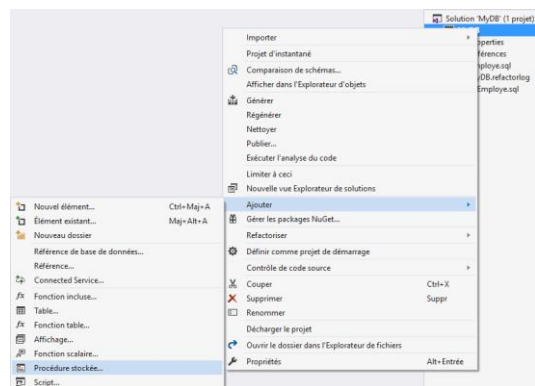
33

## CRÉER UNE PROCÉDURE STOCKÉE

Nous pourrions également créer des procédures stockées.

Pour ce faire :

- Cliquons droit sur le projet (ou sur un répertoire du projet)
- Ajouter
- Procédure stockée

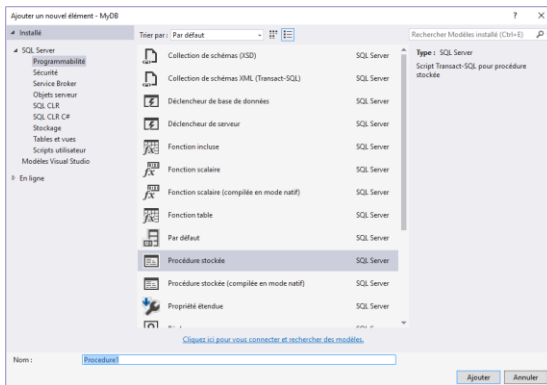


COGNITIC - MORRE THIERRY ©2020

34

34

## CRÉER UNE PROCÉDURE STOCKÉE



Nommons la et validons.

COGNITIC - MORRE THIERRY ©2020

35

35

## CRÉER UNE PROCÉDURE STOCKÉE

Spécifions les codes T-SQL

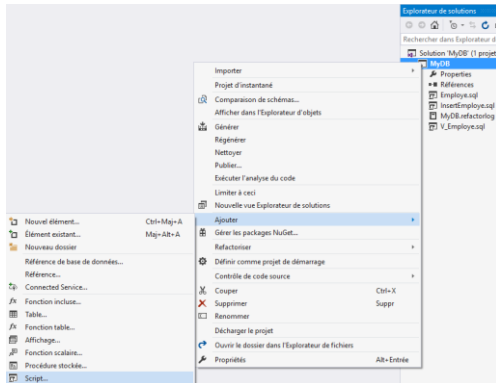
```
CREATE PROCEDURE [dbo].[InsertEmploye]
    @Nom varchar(50),
    @Prenom varchar(50),
    @Email varchar(384),
    @ResponsableID int
AS
Begin
    Set NoCount on;
    Insert into Employe (Nom, Prenom, Email, ResponsableID)
    Values (@Nom, @Prenom, @Email, @ResponsableID);
End
```

COGNITIC - MORRE THIERRY ©2020

36

36

## CRÉER UN SCRIPT POST-DÉPLOIEMENT



COGNITIC - MORRE THIERRY ©2020

37

Enfin, nous pourrions déclarer un script post-déploiement.

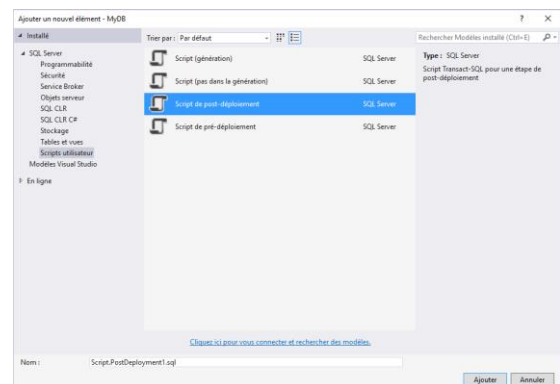
Comme son nom l'indique il sera exécuté après le déploiement de la base de données et pourra donc servir par exemple pour insérer des données, créer des logins et des utilisateurs, gérer les droits, ...

- Cliquons droit sur le projet (ou sur un répertoire du projet)
- Ajouter
- Script

37

## CRÉER UN SCRIPT POST-DÉPLOIEMENT

Choisissons « Script de post-déploiement », nommons le et validons.



COGNITIC - MORRE THIERRY ©2020

38

38

# CRÉER UN SCRIPT POST-DÉPLOIEMENT

```
Set Identity_Insert Employe On;

INSERT INTO Employe (ID, Prenom, Nom, Email, ResponsableID)
VALUES
(1, 'Ken', 'Sánchez', 'ken.sanchez@mycompany.com', NULL),
(273, 'Brian', 'Welcker', 'brian.welcker@mycompany.com', 1),
(274, 'Stephen', 'Jiang', 'stephen.jiang@mycompany.com', 273),
(275, 'Michael', 'Blythe', 'michael.blythe@mycompany.com', 274),
(276, 'Linda', 'Mitchell', 'linda.mitchell@mycompany.com', 274),
(285, 'Syed', 'Abbas', 'syed.abbas@mycompany.com', 273),
(286, 'Lynn', 'Tsoflias', 'lynn.tsoflias@mycompany.com', 285),
(16, 'David', 'Bradley', 'david.bradley@mycompany.com', 273),
(23, 'Mary', 'Gibson', 'mary.gibson@mycompany.com', 16);

Set Identity_Insert Employe Off;
```

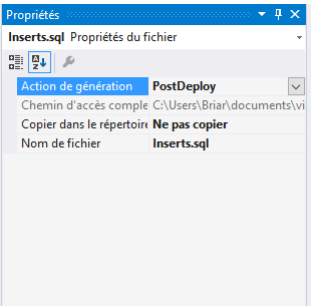
Une fois celui-ci créer, écrivons le code T-SQL à réaliser.

« Set Identity\_Insert Employe On » permet de spécifier nous même les données de la colonne « ID » malgré le fait qu'elle soit en auto-incrémentation.

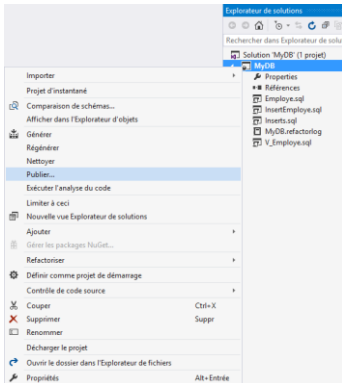
# CRÉER UN SCRIPT POST-DÉPLOIEMENT

Dans la partie propriété, l'action de génération détermine le moment d'exécution du script.

Dans notre cas « PostDeploy ».



## DÉPLOYER SA BASE DE DONNÉES



Voilà notre base de données est « terminée ». Nous devons donc la déployer sur le serveur.

- Cliquons droit sur le projet
- Publier...

COGNITIC - MORRE THIERRY ©2020

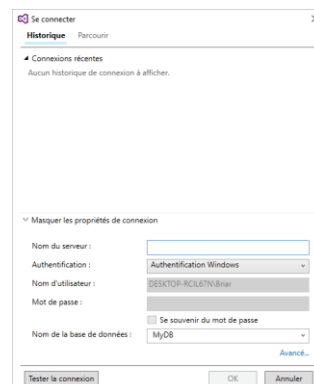
41

41

## DÉPLOYER SA BASE DE DONNÉES

Avant tout chose nous devons spécifier :

- L'adresse du serveur
- Le mode d'authentification (Windows ou Sql Server)
- Eventuellement le login et le mot de passe
- Ainsi que le nom de la base de données.

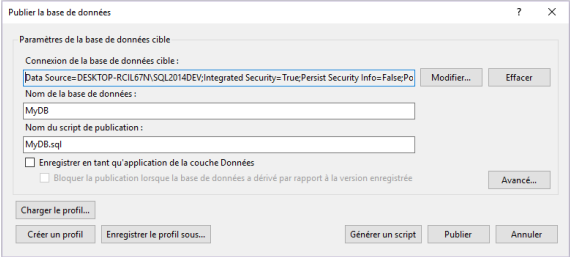


COGNITIC - MORRE THIERRY ©2020

42

42

# DÉPLOYER SA BASE DE DONNÉES



Si l'écran précédent n'est pas apparu, cliquons sur modifier et revenons sur le slide précédent...

Une fois les informations de connexion spécifiée, cliquons sur publier.



Bien entendu, assurez-vous d'avoir les droits nécessaires.

COGNITIC - MORRE THIERRY ©2020

43

43

# DÉPLOYER SA BASE DE DONNÉES

Dans la fenêtre « Opérations des outils de données », nous pourrons suivre l'évolution de la publication.

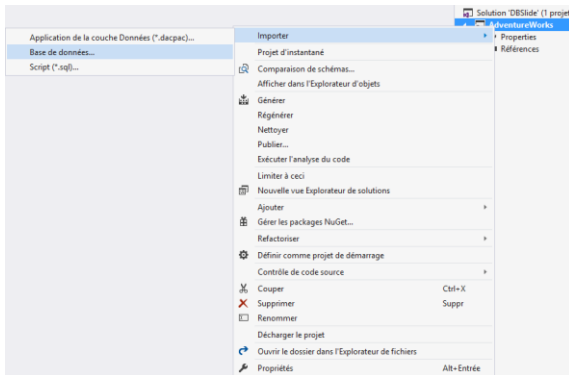


COGNITIC - MORRE THIERRY ©2020

44

44

## IMPORTER UNE BASE DE DONNÉES EXISTANTE



Nous venons de créer une base de données en partant de rien.

Nous avons, cependant, la possibilité d'importer dans notre projet une base de données existante.

- Pour ce faire cliquons droit sur notre projet
- Choisissons importer

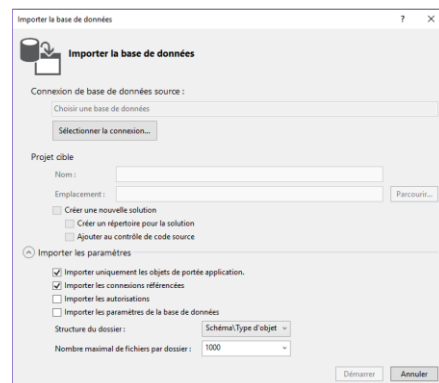
COGNITIC - MORRE THIERRY ©2020

45

45

## IMPORTER UNE BASE DE DONNÉES EXISTANTE

Commençons par sélectionner la connexion.

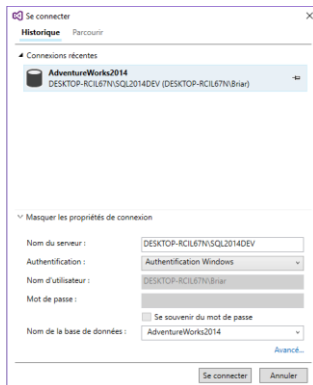


COGNITIC - MORRE THIERRY ©2020

46

46

## IMPORTER UNE BASE DE DONNÉES EXISTANTE



COGNITIC - MORRE THIERRY ©2020

Pour cette étape, nous devons fournir toutes les informations de connexion comme pour la publication.

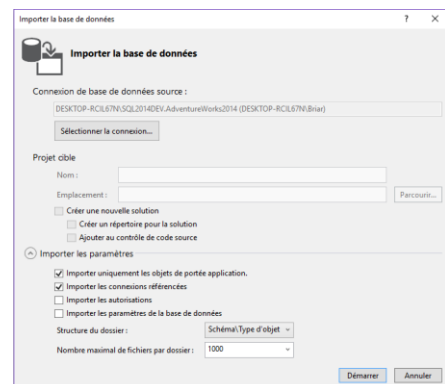
- L'adresse du serveur
- Le mode d'authentification (Windows ou Sql Server)
- Eventuellement le login et le mot de passe
- Ainsi que le nom de la base de données.

47

47

## IMPORTER UNE BASE DE DONNÉES EXISTANTE

Nous pouvons spécifier quelques options supplémentaires et cliquons sur « Démarrer ».



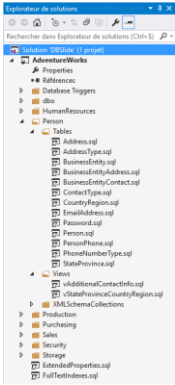
COGNITIC - MORRE THIERRY ©2020

48

48



# IMPORTER UNE BASE DE DONNÉES EXISTANTE



Nous voici donc avec notre base de données importées dans Visual Studio.

COGNITIC - MORRE THIERRY ©2020

49

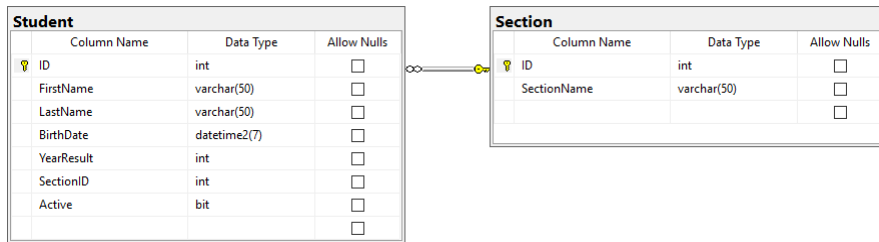
# EXERCICES

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

50

## EXERCICES



51

51

## EXERCICES

- Depuis Visual Studio et à l'aide d'un projet SQL Serveur, créez une base de données appelée « ADO » d'après le schéma du slide précédent. En sachant que :
  - ID de Student est auto-incrémenté
  - « Active » à pour valeur par défaut « 1 »
- Créez une vue « V\_Student » n'affichant que les étudiants actifs
- Ajoutez la clé étrangère de « SectionID » dans « Student » vers « ID » de « Section »
- Ajoutez les contraintes suivantes :
  - « YearResult » doit-être compris entre 0 et 20
  - « BirthDate » doit-être supérieure ou égale au 1<sup>er</sup> Janvier 1930
- Ajouter les procédures :
  - AddStudent
  - UpdateStudent (ne peut modifier que la SectionID et le résultat annuel)
  - DeleteStudent
  - AddSection

COGNITIC - MORRE THIERRY ©2020

52

52

## EXERCICES

- Ajoutez un trigger qui remplace un ordre « Delete from Student » par un ordre « Update Student set Active = 0 where ... »
- Créer un script Post-Déploiement reprenant le contenu du fichier « ADO\_LoadData.sql »
- Déployer votre base de données sur « SQL Serveur »

COGNITIC - MORRE THIERRY ©2020

53

53

## LES ESPACES DE NOMS

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

54

54

## LES ESPACES DE NOMS

ADO.NET est réparti dans plusieurs espaces de noms, les principaux sont les suivants :

- System.Data : L'espace de noms System.Data permet d'accéder aux types qui représentent l'architecture ADO.NET
- System.Data.Common : Cet espace de noms contient des classes abstraites héritées par les fournisseurs de données .NET Framework.
- System.Data.OracleClient : L'espace de noms System.Data.OracleClient contient les types nécessaires pour travailler avec Oracle.
- System.Data.SqlClient : L'espace de noms System.Data.SqlClient contient les types nécessaires pour travailler avec SQL Server.
- System.Data.SqlTypes : Cet espace de noms fournit des classes pour les types de données natifs dans SQL Server.

COGNITIC - MORRE THIERRY ©2020

55

55

## NOTION DE CONNEXIONS

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

56

56

- La « `ConnectionString` »
- La classe « `SqlConnection` »

## LES ESPACES DE NOMS

COGNITIC - MORRE THIERRY ©2020

57

57

## LA « `CONNECTIONSTRING` »

```
string WinAuthConnectionString = @"Server=DESKTOP-RCIL67N\SQL2014DEV;" +
    "Database=AdventureWorks2014;" +
    "Trusted_Connection=True;";

string SqlAuthConnectionString = @"Server=DESKTOP-RCIL67N\SQL2014DEV;" +
    "Database=AdventureWorks2014;" +
    "User Id=Thierry;Password=Test1234=";
```

Avant toute chose, pour nous connecter à une base de données spécifique, nous avons besoin de différentes informations.

- L'adresse du serveur avec éventuellement son instance.
- Le nom de la base de données
- Les informations d'identification de l'utilisateur.

Ces informations sont regroupées et structurées en ce que nous appelons communément une chaîne de connexion (`ConnectionString`).

Chaque SGBD possède sa propre structure de « `ConnectionString` »

Toutes les retenir par cœur, est difficile, par conséquent, je conseille le site « <http://www.connectionstrings.com> » qui reprend les différentes chaînes de connexion en fonction du type de serveur sur lequel vous vous connecterez.

COGNITIC - MORRE THIERRY ©2020

58

58

## LA CLASSE « SQLCONNECTION »

Un objet de type « SqlConnection » représente une connexion à une base de données SQL Server.

Cette classe est définie « sealed », hérite de la classe « DbConnection » et implémente l'interface « IDisposable ».

Elle possède des propriétés telles que :

- **ConnectionString** : Permet de spécifier la chaîne de connexion
- **State** : retourne le statut de la connexion (Closed, Open, Connecting, Executing, Fetching ou Broken)

Elle possède également des méthodes telles que :

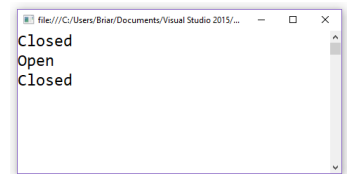
- **Open** : ouvre la connexion.
- **Close** : ferme la connexion.
- **CreateCommand** : crée une commande pour cette connexion.

```
static void Main(string[] args)
{
    string WinAuthConnectionString = @"Server=DESKTOP-RCIL67N\SQLEDEV;" +
        "Database=AdventureWorks2014;" +
        "Trusted_Connection=True;";

    SqlConnection c = new SqlConnection();
    c.ConnectionString = WinAuthConnectionString;

    Console.WriteLine(c.State);
    c.Open();
    Console.WriteLine(c.State);
    c.Close();
    Console.WriteLine(c.State);

    Console.ReadLine();
}
```



59

## EXERCICE

C# - ADO.NET

60

## EXERCICE

- Etablissez la connexion à votre base de données « ADO »

61

61

## LA CLASSE « SQLCOMMAND »

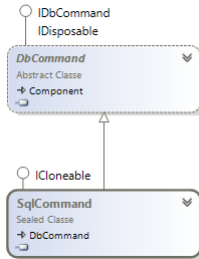
C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

62

62

## LA CLASSE « SQLCOMMAND »



La classe « SqlCommand » représente une instruction T-SQL ou une procédure stockée à exécuter par rapport à une base de données SQL Server.

Cette classe est définie « sealed », hérite de la classe « DbCommand » et implémente l'interface « ICloneable » et par son héritage de l'interface « IDisposable ».

Elle possède des propriétés telles que :

- **CommandText** : Obtient ou définit l'instruction T-SQL
- **CommandType** : Obtient ou définit une valeur indiquant la manière dont la propriété **CommandText** doit être interprétée
- **Connection** : Obtient ou définit la connexion utilisée par la commande
- **Parameters** : Obtient la collection de paramètres
- **Transaction** : Obtient ou définit la transaction dans laquelle la commande s'exécute.

63

## LA CLASSE « SQLCOMMAND »

Elle possède également diverses méthodes d'exécutions que nous verrons en détail dans les chapitres suivants.

Cependant, pour instancier une commande, nous avons deux possibilités

- Via le constructeur
- Via la méthode « CreateCommand » de la connexion.

Il est à savoir que si nous passons par le constructeur, nous serons obligé remplir la propriété « Connection » de la commande nous-même.

```

static void Main(string[] args)
{
    using (SqlConnection c = new SqlConnection())
    {
        c.ConnectionString = "Mettez votre chaîne de connexion ici";

        using (SqlCommand cmd = new SqlCommand())
        {
            cmd.Connection = c;

            using (SqlCommand cmd = c.CreateCommand())
            {
                // ...
            }
        }
    }
}
  
```

64



# SÉLECTION DE DONNÉES

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

65

65

- Mode connecté
  - La méthode « ExecuteScalar »
  - La classe « SqlDataReader »
  - La méthode « ExecuteReader »
- Mode déconnecté
  - Avant de commencer
  - La classe « SqlDataAdapter »
  - Lecture des données
- Avantages et inconvénients

## LES ESPACES DE NOMS

COGNITIC - MORRE THIERRY ©2020

66

66

# MODE CONNECTÉ

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

67

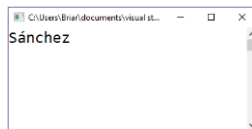
67

## MODE CONNECTÉ : LA MÉTHODE « EXECUTESCALAR »

```
using (SqlConnection c = new SqlConnection())
{
    c.ConnectionString =
        @"Data Source=DESKTOP-RCIL67N\SQL2014DEV;"+
        "Initial Catalog=AdventureWorks2014;"+
        "Integrated Security=True";

    using (SqlCommand cmd = c.CreateCommand())
    {
        cmd.CommandText = "SELECT LastName " +
            "FROM Person.Person " +
            "WHERE BusinessEntityID = 1;";

        c.Open();
        string LastName = (string)cmd.ExecuteScalar();
        c.Close();
        Console.WriteLine(LastName);
    }
}
```



Par « mode connecté » on entend que la connexion doit rester ouverte au moment de l'exécution de la requête et durant la récupération du résultat.

Ensuite, exécuter un ordre « select » sur une base de donnée peut retourner deux types de résultats.

Le premier d'entre eux est un résultat dit « scalaire » qui par définition n'est composé que d'une seule valeur.

Afin de récupérer ce type de résultat, nous allons utiliser la méthode « ExecuteScalar », cette méthode retourne une valeur de type « object » en raison que le type de valeur dépend de notre ordre « Select ».

```
public override object ExecuteScalar();
```

68

## MODE CONNECTÉ : LA CLASSE « SQLDATAREADER »

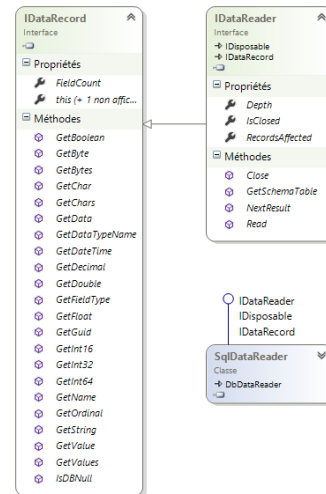
Le deuxième type de résultat est un résultat dit « tabulaire » qui par définition est composé de plusieurs colonnes et/ou plusieurs lignes.

Pour ce faire, nous utiliserons cette fois la méthode « ExecuteReader »

Cette méthode retourne un objet de type « SqlDataReader » qui hérite de « DbDataReader » et implémente deux interfaces :

- « IDataRecord » qui nous donne accès aux indexeurs
- « IDataReader » qui nous donne accès à la méthode « Read »

```
public SqlDataReader ExecuteReader();
```



69

## MODE CONNECTÉ : LA MÉTHODE « EXECUTEREADER »

```

using (SqlCommand cmd = c.CreateCommand())
{
    cmd.CommandText = "SELECT LastName, FirstName " +
        "FROM Person.Person " +
        "WHERE BusinessEntityID < 10;";

    c.Open();
    using (SqlDataReader reader = cmd.ExecuteReader())
    {
        while (reader.Read())
        {
            Console.WriteLine($"{reader["LastName"]} {reader[1]}");
        }
    }
    c.Close();
}
  
```

Une fois notre connexion ouverte, nous pouvons donc appeler la méthode « ExecuteReader ».

Chaque fois que nous voudrions lire une ligne du résultat, nous appellerons la méthode « Read » qui nous renvoi, une valeur de type « bool », si la lecture a aboutie.

Quant aux données, elles deviennent accessibles via les indexeurs de type « int » (pour l'indice) ou de type « string » (pour le nom de colonne).

70

# MODE DÉCONNECTÉ

C# - ADO.NET

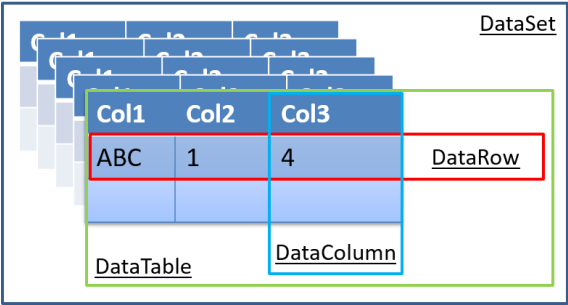
COGNITIC - MORRE THIERRY ©2020

71

71

## MODE DÉCONNECTÉ : AVANT DE COMMENCER

- DataSet : Le « DataSet » est l'élément central de l'architecture ADO.NET. Chaque « DataSet » peut contenir plusieurs objets « DataTable », chacune de ces « DataTable » pouvant être reliées entre elle par des « DataRelation ».
- DataTable : Représente une table de données en mémoire, elle contient des colonnes de type « DataColumn » et, si il n'est pas vide, des enregistrements de type « DataRow ».
- DataRow : Représente un enregistrement dans une « DataTable », cette classe possède des indexeurs nous permettant de récupérer les données de nos champs.
- DataColumn : Représente une colonne (Nom, type de donnée, etc.) d'une DataTable.
- DataRelation : représente une relation entre deux « DataTable » et ajoute une contrainte relationnelle (notion de « Foreign Key »)

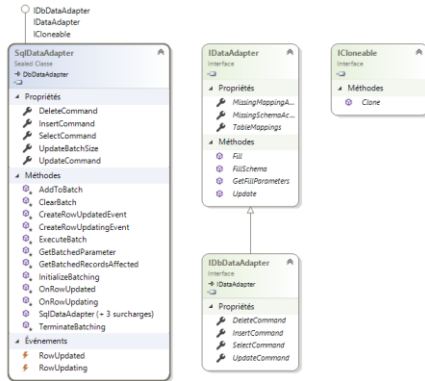


COGNITIC - MORRE THIERRY ©2020

72

72

## MODE DÉCONNECTÉ : LA CLASSE « SQLDATAADAPTER »



COGNITIC - MORRE THIERRY ©2020

Une autre possibilité pour récupérer des données est d'utiliser le mode déconnecté. Bien entendu le terme « déconnecté » ne concerne pas la récupération de données mais il met en avant le fait que ces données seront intégralement rapatriées en mémoire nous dissociant de la base de données.

Ce « mode » consiste à utiliser la classe « SqlDataAdapter » et de lui demander de remplir soit un objet de type « DataSet » ou « DataTable ».

Cette classe a pour rôle de faire la passerelle entre les « DataSet » ou « DataTable » et notre base de données.



73

73

## MODE DÉCONNECTÉ : LECTURE DES DONNÉES

Afin de lire les données nous devons tout d'abord instancier notre « SqlDataAdapter », ceci étant fait nous devons lui passer notre commande à la propriété « SelectCommand ».

Ensuite, il nous reste à invoquer la méthode « Fill ».

Cette méthode « Fill » va se charger :

- D'ouvrir la connexion
- D'exécuter l'ordre de sélection
- De rapatrier les données dans le « DataSet » ou la « DataTable »
- De refermer la connexion

Pour terminer il nous reste à parcourir la « DataRowCollection » de nos « DataTable » pour récupérer et manipuler nos données.

```
using (SqlConnection c = new SqlConnection())
{
    c.ConnectionString =
        @"Data Source=DESKTOP-RCIL67N\SQL2014DEV;"+
        "Initial Catalog=AdventureWorks2014;"+
        "Integrated Security=True";

    using (SqlCommand cmd = c.CreateCommand())
    {
        cmd.CommandText = "SELECT LastName, FirstName " +
            "FROM Person.Person " +
            "WHERE BusinessEntityID < 10;";

        SqlDataAdapter da = new SqlDataAdapter();
        da.SelectCommand = cmd;
        DataSet ds = new DataSet();
        // ou DataTable dt = new DataTable();
        da.Fill(ds);
        // ou da.Fill(dt);

        if(ds.Tables.Count > 0)
        {
            foreach(DataRow dr in ds.Tables[0].Rows)
            {
                Console.WriteLine($"{dr["LastName"]} {dr["FirstName"]}");
            }
        }
    }
}
```

74

COGNITIC - MORRE THIERRY ©2020

74



# AVANTAGES ET INCONVÉNIENTS

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

75

75



# AVANTAGES ET INCONVÉNIENTS

	+	-
Mode connecté	<ul style="list-style-type: none"><li>• Léger en mémoire (on ne reçoit qu'un record à la fois)</li><li>• On travail avec les données présentent en base de données</li></ul>	<ul style="list-style-type: none"><li>• La connexion doit rester ouverte le temps du traitement.</li><li>• Accroît la charge au niveau communication réseau</li></ul>
Mode déconnecté	<ul style="list-style-type: none"><li>• Permet de diminuer niveau communication réseau</li></ul>	<ul style="list-style-type: none"><li>• Peut s'avérer très lourd en mémoire</li><li>• On travail sur une copie des données et non sur la base de données</li></ul>

COGNITIC - MORRE THIERRY ©2020

76

76

# EXERCICES

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

77

77

## EXERCICES

- Afficher l'« ID », le « Nom », le « Prenom » de chaque étudiant depuis la vue « V\_Student » en utilisant la méthode connectée
- Afficher l'« ID », le « Nom » de chaque section en utilisant la méthode déconnectée
- Afficher la moyenne annuelle des étudiants

78

78

# INSERTION, MODIFICATION ET SUPPRESSION DE DONNÉES

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

79

79

## LA MÉTHODE « EXECUTENONQUERY »

```
CREATE TABLE Test (
    ID INT NOT NULL IDENTITY,
    Nom VARCHAR(50) NOT NULL,
    CONSTRAINT PK_Test PRIMARY KEY(ID)
);

using (SqlConnection c = new SqlConnection())
{
    c.ConnectionString = @"Data Source=ASUS-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO;" +
        "Integrated Security=true";

    using (SqlCommand cmd = c.CreateCommand())
    {
        cmd.CommandText = "insert into Test (Nom) values ('Doe');";

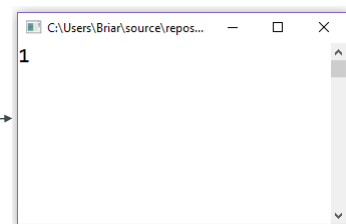
        c.Open();
        int rows = cmd.ExecuteNonQuery();

        Console.WriteLine(rows);
    }
}
```

Lorsque nous exécutons un ordre dit « DML », le serveur nous retourne le nombre de lignes affectées par notre requête.

Depuis notre application nous appellerons, la méthode « ExecuteNonQuery » qui nous retournera une valeur de type « int » qui contiendra ce nombre.

```
public override int ExecuteNonQuery();
```



COGNITIC - MORRE THIERRY ©2020

80

80



---

## CAS PARTICULIER : UTILISATION DU MOT CLÉ « OUTPUT »

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

81

81

- 
- Les tables « inserted » et « deleted »
  - Utilisation du mot clé « output »

## LES ESPACES DE NOMS

COGNITIC - MORRE THIERRY ©2020

82

82

# LES TABLES « INSERTED » ET « DELETED »

L'utilisation du mot clé output au niveau de nos requêtes implique l'utilisation de deux tables temporaires spécifiques.

Ces deux tables sont créées lors de l'exécute d'ordre DML, ont la structure de la table sur laquelle on travaille et disparaissent une fois l'action réalisée.

Il s'agit des tables « inserted » et deleted ».

La table « inserted » n'est utilisable que dans l'utilisation d'un ordre « insert » ou « update ».

Tandis que la table « deleted » ne pourra être utilisée que dans le cadre d'un « update » ou d'un « delete ».



Les noms « inserted » et « deleted » sont spécifiques à SQL Server.

COGNITIC - MORRE THIERRY ©2020

83

83

Pour plus d'informations sur l'utilisation du mot-clé output dans les ordres DML en T-SQL

<https://docs.microsoft.com/fr-fr/sql/t-sql/queries/output-clause-transact-sql>

Ordre DML	Inserted	Deleted
Insert	Nouvelles données	NA
Update	Nouvelles valeurs	Anciennes valeurs
Delete	NA	Données supprimées

# UTILISATION DU MOT CLÉ « OUTPUT »

```
using (SqlConnection c = new SqlConnection())
{
    c.ConnectionString = @"Data Source=ASUS-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO;" +
        "Integrated Security=true";

    using (SqlCommand cmd = c.CreateCommand())
    {
        cmd.CommandText =
            "insert into Test (Nom) output inserted.ID values ('Zorro');";

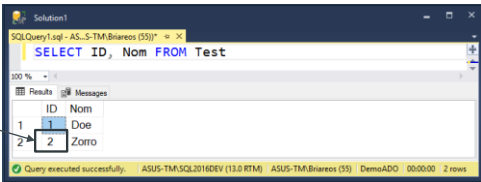
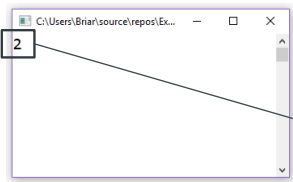
        c.Open();
        int ID = (int)cmd.ExecuteScalar();

        Console.WriteLine(ID);
    }
}
```

COGNITIC - MORRE THIERRY ©2020

Le hic du coups c'est que notre requête, en plus de retourner le nombre de ligne, peut nous retourner une autre valeur.

Ce faisant, il nous faudra faire attention au nombre de valeurs retournées et invoquer la méthode « ExecuteScalar » ou « ExecuteReader » en fonction du cas.



84

84

## EXERCICES

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

85

85

## EXERCICES

- Instanciez un objet de type « Student » contenant vos informations
- Insérez votre objet en base de données en récupérant son « ID » au passage

86

86

# L'INJECTION SQL

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

87

87

## L'INJECTION SQL

L'injection SQL est l'exploitation de faille de sécurité d'une application interagissant avec une base de données.

Elle permet d'injecter dans la requête SQL en cours un morceau de requête non prévu par le système et pouvant compromettre la sécurité.

Cette faille est exploitable en raison de la création dynamique de la requête par concaténation.

Imaginons la situation suivante, j'ai un formulaire de login et je demande à l'utilisateur de me fournir son login et son mot de passe.

```
//Informations venant du formulaire
string Login = "root";
string Passwd = "Ad165fhm@f$53dg10dg";

//Création de la commande
string Query = "select ID, Nom, Prenom from Utilisateur " +
    "where Login='" + Login + "' and Passwd='" + Passwd + "'";

Console.WriteLine(Query);
```

Jusque là tout va bien

```
C:\Users\Briar\source\repos\ExampleAdo\ExampleAdo\bin\Debug\ExampleAdo.exe
select ID, Nom, Prenom from Utilisateur where Login='root' and Passwd='Ad165fhm@f$53dg10dg';
```

COGNITIC - MORRE THIERRY ©2020

88

88

## L'INJECTION SQL

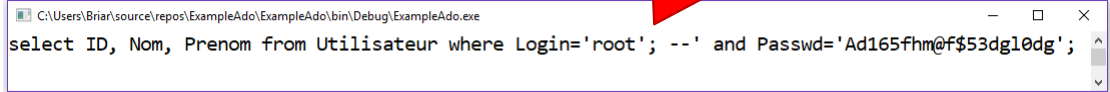
```
//Informations venant du formulaire
string Login = "root'; --";
string Passwd = "Ad165fhm@f$53dgl0dg";

//Création de la commande
string Query = "select ID, Nom, Prenom from Utilisateur " +
    "where Login='" + Login + "' and Passwd='" + Passwd + "'";

Console.WriteLine(Query);
```

Ajoutons cette fois l'injection SQL.

**Houston, on a un problème!!**



C:\Users\Briar\source\repos\ExampleAdo\ExampleAdo\bin\Debug\ExampleAdo.exe

```
select ID, Nom, Prenom from Utilisateur where Login='root'; --' and Passwd='Ad165fhm@f$53dgl0dg';
```

COGNITIC - MORRE THIERRY ©2020

89

89

## LES REQUÊTES PARAMÉTRÉES

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

90

90

## LES REQUÊTES PARAMÉTRÉES

Afin de nous prémunir contre l'injection SQL, il nous est recommandé d'utiliser les requêtes paramétrées.

Elles sont plus sécurisées, offrent une meilleure lisibilité et peuvent être utilisées avec tous les types de requêtes.

Cependant l'utilisation de paramètres ne peut se faire que dans le cadre d'une expression.

Les paramètres dans nos requêtes seront précédés d'un '@' et devront être spécifiés dans notre commande de type « SqlCommand ».

Le typage se faisant automatiquement et pour éviter que votre paramètre soit repris comme valeur, nous ne devons pas le mettre entre de simples guillemets.

```
select year_result * @param1
from maTable
where xxx = @param2
```



```
select @MonChamps
from @maTable
where monchamp = '@param1'
```



COGNITIC - MORRE THIERRY ©2020

91

91

## LES REQUÊTES PARAMÉTRÉES

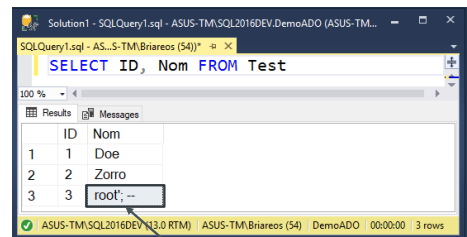
```
using (SqlConnection c = new SqlConnection()) {
    c.ConnectionString = @"Data Source=ASUS-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO;" +
        "Integrated Security=true";

    //Informations venant du formulaire
    string Login = "root'; --";

    using (SqlCommand cmd = c.CreateCommand()) {
        cmd.CommandText = "insert into Test (Nom) output inserted.ID values (@Nom)";
        //Création du paramètre
        SqlParameter PNom = new SqlParameter() {
            ParameterName = "Nom",
            Value = Login
        };
        //Ajout de celui-ci à la commande
        cmd.Parameters.Add(PNom);
        //ou pour SQL Server
        //cmd.Parameters.AddWithValue("Nom", Login);
        c.Open();
        int ID = (int)cmd.ExecuteScalar();
        Console.WriteLine(ID);
    }
}
```

COGNITIC - MORRE THIERRY ©2020

```
exec sp_executesql N'insert into Test (Nom) output inserted.ID values (@Nom);', N'@Nom nvarchar(9)', @Nom=N'root'; --'
```



92

92

# VALEUR « NULL » EN C# VS SQL

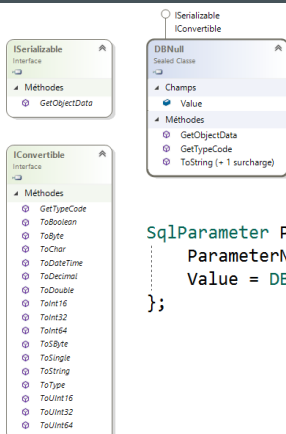
C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

93

93

## VALEUR « NULL » EN C# VS SQL



COGNITIC - MORRE THIERRY ©2020

Les « null » de C# et des SGBD sont différents et ne sont donc pas interprétés de la même manière.

En effet, en C# c'est une absence de référence et en DB c'est une absence de valeur.

Il ne nous est donc pas permis d'envoyer un « null » C# comme valeur ou s'attendre à le recevoir de notre base de données.

C'est pour cela que nous devons travailler avec le type « DBNull ».

Si notre requête nous renvoie un « null » SQL, la conversion C# sera de type « DBNull ».

Lorsque nous souhaiterons envoyer un « null » à notre base de données, nous devrons utiliser la propriété « DBNull.Value »

```

    object o = cmd.ExecuteScalar();
    string Result = (o is DBNull) ? null : (string)o;
  
```

94

94

# EXERCICES

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

95

95

## EXERCICES

- Instanciez un objet de type « Student » contenant les informations de votre voisin(e)
- Insérez votre objet en base de données en utilisant les requêtes paramétrées

96

96



---

# APPELER UNE PROCÉDURE STOCKÉE

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

97

97

- 
- Appel via le mot-clé « exec »
  - L'énumération « CommandType »
  - Appel simplifié
  - Les paramètres « output »
  - Les paramètres de type « Table »

## APPELER UNE PROCÉDURE STOCKÉE

COGNITIC - MORRE THIERRY ©2020

98

98

## APPEL VIA LE MOT-CLÉ « EXEC »

Nous venons de créer une procédure au niveau de notre base de données afin d'accroître la sécurité et la maintenabilité du processus d'ajout d'une personne dans notre table « Test ».

Pour ce faire, nous l'avons nommée « AddPerson » et celle-ci demande un paramètre de type « varchar(50) » appelé « @Name » et nous allons nous servir de ce paramètre pour l'insérer dans notre table.

Une fois cet ajout effectué, nous lui demandons de sélectionner la dernière valeur auto-incrémentée dans notre table « Test ».

```
CREATE PROCEDURE AddPerson
    @Name varchar(50)
AS
BEGIN
    insert into Test (Nom) values (@Name);
    select convert(int, @@identity) from Test;
END
GO
```

COGNITIC - MORRE THIERRY ©2020

99

99

## APPEL VIA LE MOT-CLÉ « EXEC »

```
using (SqlConnection c = new SqlConnection()) {
    c.ConnectionString = @"Data Source=ASUS-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO;" +
        "Integrated Security=true";

    //Informations venant du formulaire
    string Login = "Smith";

    using (SqlCommand cmd = c.CreateCommand()) {
        cmd.CommandText = "exec AddPerson @Name=@Nom;";
        //Création du paramètre
        SqlParameter PNom = new SqlParameter() {
            ParameterName = "Nom",
            Value = Login
        };
        //Ajout de celui-ci à la commande
        cmd.Parameters.Add(PNom);
        //ou pour SQL Server
        //cmd.Parameters.AddWithValue("Nom", Login);
        c.Open();
        int ID = (int)cmd.ExecuteScalar();
        Console.WriteLine(ID);
    }
}
```

COGNITIC - MORRE THIERRY ©2020

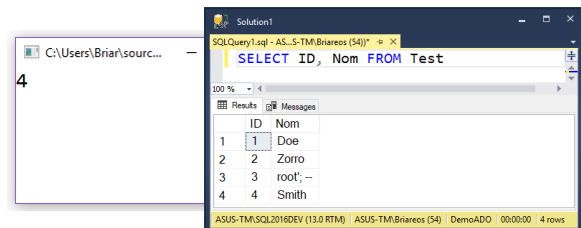
L'appel de cette procédure prendra donc cette forme.

« exec AddPerson @Name=@Nom; »

Exec AddPerson : demande à notre SGBD d'exécuter la procédure « AddPerson »

@Name : est le paramètre que notre procédure demande,

@Nom : est la valeur que nous lui envoyons sous forme de requête paramétrée.



100

## L'ÉNUMÉRATION « COMMANDTYPE »

En y regardant de plus près, dans notre type « SqlCommand », nous retrouvons une propriété « CommandType ».

Cette propriété est de type « CommandType » qui est une énumération faisant partie de l'espace de noms « System.Data ».

Cette énumération propose cas trois constantes.



La valeur « TableDirect » de l'énumération ne peut-être utilisée qu'avec le provider « OleDb »

```
...public enum CommandType
{
    //
    // Résumé :
    //     Une commande de texte SQL. (Par défaut).
    Text = 1,
    //
    // Résumé :
    //     Le nom d'une procédure stockée.
    StoredProcedure = 4,
    //
    // Résumé :
    //     Nom d'une table.
    TableDirect = 512
}
```

COGNITIC - MORRE THIERRY ©2020

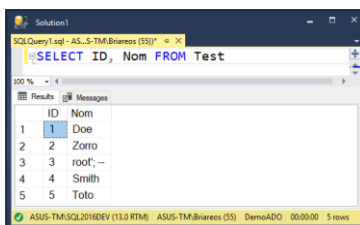
101

101

## APPEL SIMPLIFIÉ

Sachant ceci, nous pouvons donc simplifier notre appel en en affectant à la propriété « CommandType » la valeur « CommandType.StoredProcedure » et en ne spécifiant que le nom de la procédure dans la propriété « CommandText ».

Bien entendu, nous devons continuer à fournir à notre commande les paramètres demandés par notre procédure stockée.



COGNITIC - MORRE THIERRY ©2020

```
using (SqlConnection c = new SqlConnection()) {
    c.ConnectionString = @"Data Source=ASUS-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO;" +
        "Integrated Security=true";

    //Informations venant du formulaire
    string Login = "Toto";

    using (SqlCommand cmd = c.CreateCommand()) {
        cmd.CommandText = "AddPerson";
        cmd.CommandType = CommandType.StoredProcedure;

        SqlParameter PNom = new SqlParameter() {
            ParameterName = "Name",
            Value = Login
        };
        cmd.Parameters.Add(PNom);
        c.Open();
        int ID = (int)cmd.ExecuteScalar();
        Console.WriteLine(ID);
    }
}
```

102

102

## LES PARAMÈTRES « OUTPUT »

Changeons un peu notre procédure.

Cette fois nous demandons que le paramètre nous soit retourné par le paramètre « @ID » défini en « output ».

Pour récupérer cette valeur dans notre code C#, nous allons devoir spécifier la direction de notre paramètre en utilisant l'énumération « ParameterDirection ».

```
...public enum ParameterDirection
{
    ...Input = 1,
    ...Output = 2,
    ...InputOutput = 3,
    ...ReturnValue = 6
}
```

COGNITIC - MORRE THIERRY ©2020

```
ALTER PROCEDURE [dbo].[AddPerson]
    @ID int output,
    @Name varchar(50)
AS
BEGIN
    insert into Test (Nom) values (@Name);
    select @ID = @@identity from Test;
END
```

103

103

## LES PARAMÈTRES « OUTPUT »

```
using (SqlConnection c = new SqlConnection()) {
    c.ConnectionString = @"Data Source=ASUS-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO; Integrated Security=true";
    //Informations venant du formulaire
    string Login = "Louis XIV";
    using (SqlCommand cmd = c.CreateCommand()) {
        cmd.CommandText = "AddPerson";
        cmd.CommandType = CommandType.StoredProcedure;

        SqlParameter PID = new SqlParameter()
        {
            ParameterName = "ID",
            Value = 0,
            Direction = ParameterDirection.Output
        };

        SqlParameter PNom = new SqlParameter() {
            ParameterName = "Name",
            Value = Login,
        };

        cmd.Parameters.Add(PID);
        cmd.Parameters.Add(PNom);
        c.Open();
        cmd.ExecuteNonQuery();
        int ID = (int)cmd.Parameters["ID"].Value;
        Console.WriteLine(ID);
    }
}
```

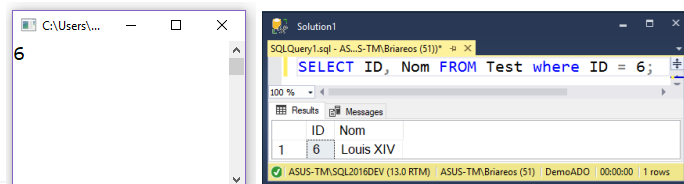
COGNITIC - MORRE THIERRY ©2020

Par défaut lorsque nous créons un paramètre celui-ci a pour direction la valeur « ParameterDirection.Input ».

Pour changer cette dernière, lors de la création de notre paramètre, il nous suffit de changer la propriété « Direction » de notre « SqlParameter ».

Une fois notre commande exécutée, nous pouvons aller rechercher la valeur reçue.

Au travers de la propriété « Parameters » de notre « SqlCommand » en utilisant l'un des indexeurs.



104

104

## LES PARAMÈTRES DE TYPE TABLE

Nous serons parfois amenés à fournir un paramètre de type table.

Pour ce faire nous aurons besoin d'utiliser ce qu'on appelle une « DataTable » typée.

C'est-à-dire une « DataTable » que nous allons structurer de manière à ce qu'elle corresponde au type « Table » défini sous Sql Serveur.

Dans notre cas, notre « DataTable » contiendra une colonne « Nom » de type « string ».

Ensuite, nous devrons remplir cette « DataTable » avant de l'envoyer en paramètre à notre requête.

```
Create Type T_Persons As Table
(
    Nom varchar(50) not null
);
Go

Create Procedure AddPersons
@Persons T_Persons ReadOnly
As
Begin
    insert into Test (Nom) select Nom from @Persons;
End
Go
```

COGNITIC - MORRE THIERRY ©2020

105

105

## LES PARAMÈTRES DE TYPE TABLE

```
using (SqlConnection c = new SqlConnection()) {
    c.ConnectionString = @"Data Source=ASUS-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO; Integrated Security=true";

    using (SqlCommand cmd = c.CreateCommand()) {
        cmd.CommandText = "AddPersons";
        cmd.CommandType = CommandType.StoredProcedure;

        1 DataTable Persons = new DataTable();
        Persons.Columns.Add(new DataColumn("Nom", typeof(string))); 2

        3 Persons.Rows.Add("Kilmer");
        Persons.Rows.Add("Willis");
        Persons.Rows.Add("Moore");

        4 SqlParameter pPersons = new SqlParameter() {
            ParameterName = "Persons",
            Value = Persons,
            TypeName = "T_Persons"
        };
        cmd.Parameters.Add(pPersons);
        cmd.Open();
        cmd.ExecuteNonQuery();
    }
}
```



Que ce soit lors de la création de la « DataTable »  
ou de l'ajout des « DataRow »,  
l'ordre des colonnes est importants

1. Commençons par instancier notre « DataTable »
2. Ajoutons les colonnes à celle-ci
3. Ajoutons nos lignes en utilisant la méthode « Add » sur la propriété « Rows » de notre « DataTable »
4. Ajoutons notre paramètres à la commande en précisant le nom du type « TSql » auquel se rapporte notre « DataTable » n au travers de la propriété « TypeName ».

```
public DataRow Add(params object[] values);
```

ID	Nom
7	Kilmer
8	Willis
9	Moore

106

106

## EXERCICES

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

107

107

## EXERCICES

- Appelez la procédure pour changer de « Section » l'étudiant vous représentant
- Appelez la procédure pour supprimer votre voisin de la base de données
- Vérifiez que le champs « Active » est bien passé à 0

108

108

# GESTION DES TRANSACTIONS

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

109

109

## GESTION DES TRANSACTIONS

Dans le cadre du travail avec les bases de données, les transactions sont monnaie courante.

En C#, le type nous permettant de gérer les transactions est « SqlTransaction ».

Cette classe nous fournit, les méthodes :

- « RollBack » pour annuler une transaction
- « Commit » pour valider la transaction

```
public sealed class SqlTransaction : DbTransaction
{
    public SqlConnection Connection { get; }
    public override IsolationLevel IsolationLevel { get; }
    protected override DbConnection DbConnection { get; }

    public override void Commit();
    public override void Rollback();
    public void Rollback(string transactionName);
    public void Save(string savePointName);
    protected override void Dispose(bool disposing);
}
```



Ces méthodes sont propres à MS Sql Server

110

COGNITIC - MORRE THIERRY ©2020

110

## GESTION DES TRANSACTIONS

```
using (SqlConnection c = new SqlConnection()) {
    c.ConnectionString = @"Data Source=ASUS-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO; Integrated Security=true";

    c.Open();
    using (SqlTransaction Transaction = c.BeginTransaction())
    {
        using (SqlCommand cmd = c.CreateCommand())
        {
            cmd.CommandText = "insert into Test (Nom) values (@Nom)";
            cmd.Transaction = Transaction;

            SqlParameter PNom = new SqlParameter()
            {
                ParameterName = "Nom",
                Value = "Thor",
            };
            cmd.Parameters.Add(PNom);
            cmd.ExecuteNonQuery();

            Transaction.Rollback();
            //ou
            //Transaction.Commit();
        }
    }
}
```

COGNITIC - MORRE THIERRY ©2020

Pour débiter une transaction, une fois notre connexion ouverte, nous demander à notre instance de type « SqlConnection » de commencer une transaction à l'aide de la méthode « BeginTransaction ».

Ensuite, à chaque fois que nous voudrions exécuter un ordre DML dans cette transaction, il nous suffira de fournir la valeur (de type « SqlParameter ») à la propriété « Transaction » de notre commande.

Pour terminer, afin de finaliser notre transaction, il nous faudra appeler soit la méthode « Commit » ou « RollBack » en fonction du cas.



La connexion doit rester ouverte tout le long de la transaction sous peine d'être perdue

111

111

## EXERCICE

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

112

112



## EXERCICE

- Analysez les besoins et créez une DLL (Librairie de classe) proposant deux classes :
  - « Command » : représentant toute commande que vous souhaitez exécuter, celle-ci doit être capable de gérer les paramètres et de permettre l'utilisation des procédures stockées
  - « Connection » : représentant une connexion vers SQL Server celle-ci devra implémenter les méthodes « ExecuteScalar », « ExecuteReader », « ExecuteNonQuery » et « GetDataTable ». Chacune d'elle devra recevoir au moins un paramètre de type « Command ». Attention, la fonction « ExecuteReader » devra quant à elle retourner une valeur de type « IEnumerable<T> ».
  - Refaites tous les exercices, depuis la récupération de données, en utilisant vos nouvelles classes.

COGNITIC - MORRE THIERRY ©2020

113

113

## DESIGN PATTERN : ABSTRACT FACTORY

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

114

114

- Schéma UML
- Cas Concret : Bataille navale
- Implémentation

## LES ESPACES DE NOMS

COGNITIC - MORRE THIERRY ©2020

115

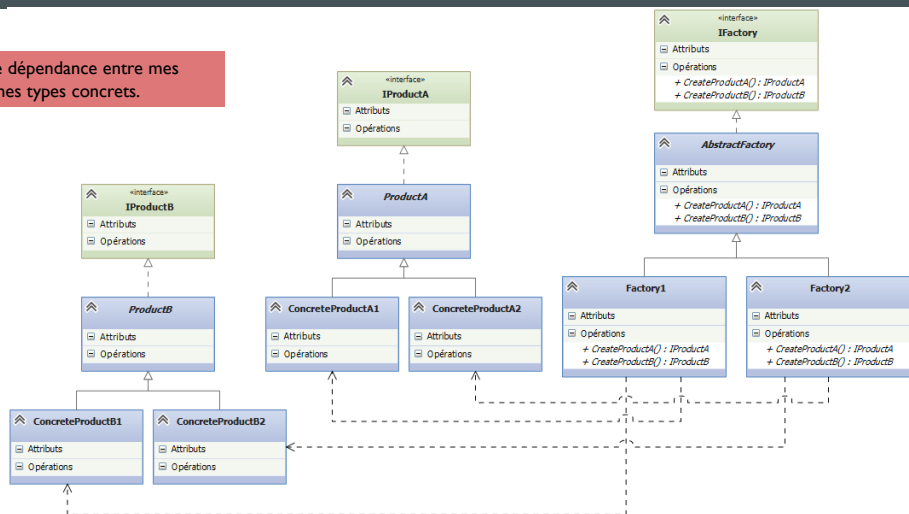
115

## SCHEMA UML



Notez la notion de dépendance entre mes fabriques et mes types concrets.

COGNITIC - MORRE THIERRY ©2020



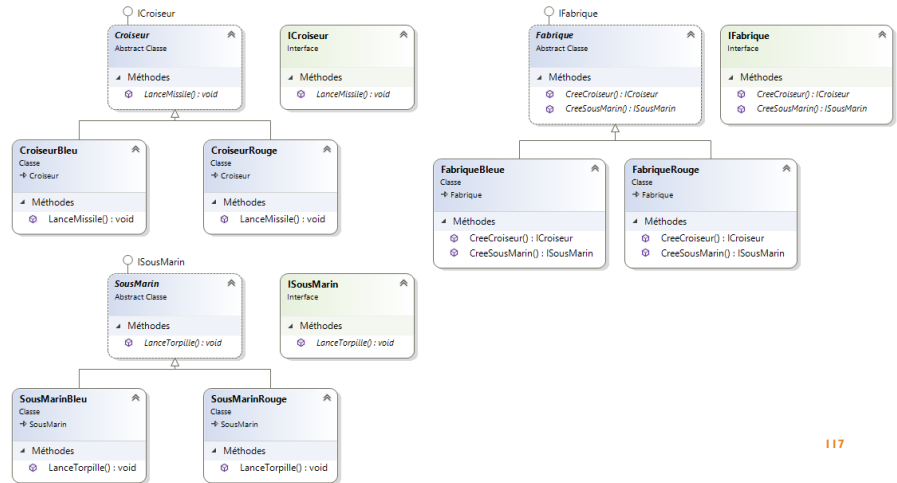
116

## CAS CONCRET : BATAILLE NAVALE

La « bataille navale » est un bon exemple pour présenter ce modèle de conception.

En effet, on y retrouve plusieurs produits : un sous-marin, un croiseur, etc.

Et deux fabriques, une fabrique « Rouge » et une « Bleue »

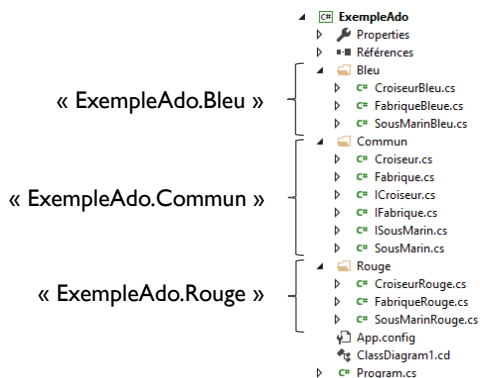


COGNITIC - MORRE THIERRY ©2020

117

117

## IMPLÉMENTATION



Pour l'implémentation, Nous pensons à subdiviser nos classes en trois espace de noms.

- « ExempleAdo.Commune » pour les classes et interfaces communes (« ICroiseur », « ISousMarin », « Fabrique », ...)
- « ExempleAdo.Bleu » pour les classes de l'équipe bleue
- « ExempleAdo.Rouge » pour les classes de l'équipe Rouge

COGNITIC - MORRE THIERRY ©2020

118

118

## IMPLÉMENTATION

### Fabrique bleue

```
public class FabriqueBleue : Fabrique
{
    public override ICroiseur CreeCroiseur()
    {
        return new CroiseurBleu();
    }

    public override ISousMarin CreeSousMarin()
    {
        return new SousMarinBleu();
    }
}
```

COGNITIC - MORRE THIERRY ©2020

### Fabrique rouge

```
public class FabriqueRouge : Fabrique
{
    public override ICroiseur CreeCroiseur()
    {
        return new CroiseurRouge();
    }

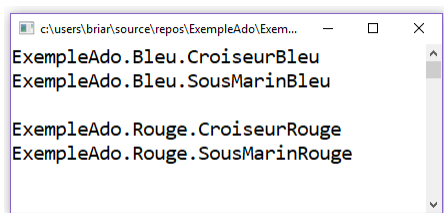
    public override ISousMarin CreeSousMarin()
    {
        return new SousMarinRouge();
    }
}
```

119

119

## IMPLÉMENTATION

Ensuite en fonction que nous instancierons une fabrique bleue ou rouge, celle-ci nous retournera les bateaux spécifiques.



Implémentation dans « ExempleAdo.zip »

```
class Program
{
    static void Main(string[] args)
    {
        IFabrique f = new FabriqueBleue();
        Console.WriteLine(f.CreeCroiseur());
        Console.WriteLine(f.CreeSousMarin());

        Console.WriteLine();

        f = new FabriqueRouge();
        Console.WriteLine(f.CreeCroiseur());
        Console.WriteLine(f.CreeSousMarin());

        Console.ReadLine();
    }
}
```

COGNITIC - MORRE THIERRY ©2020

120

120

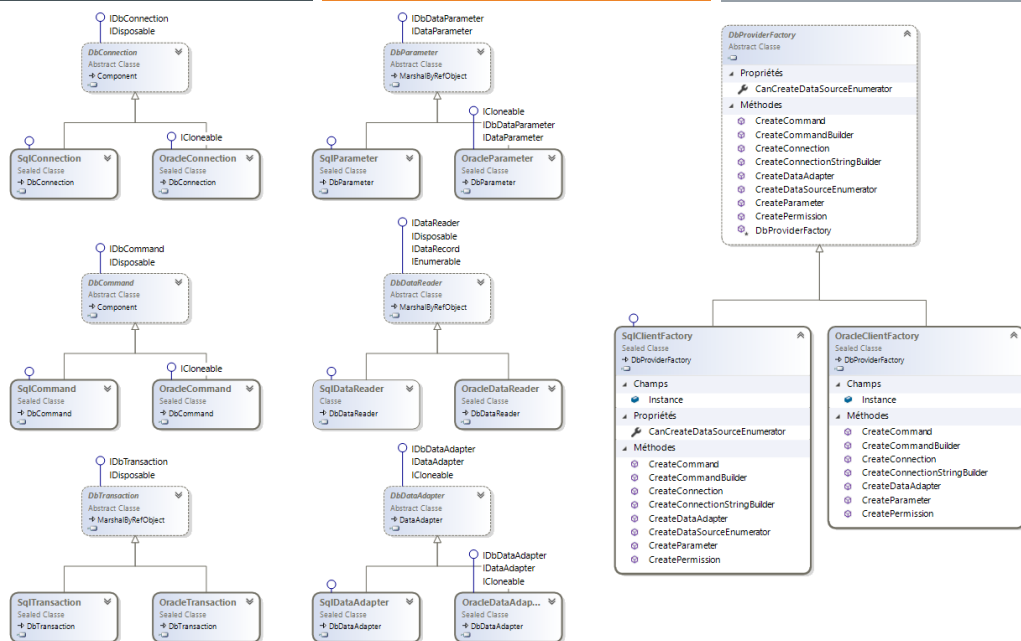
# ABSTRACT FACTORY & ADO.NET

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

121

121



122

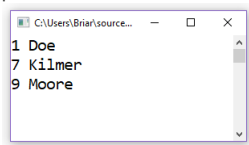
122

## ABSTRACT FACTORY & ADO.NET

L'« ADO.Net » implémente nativement l'« abstract factory » de ce fait nous pouvons directement l'utiliser.

C'est la classe « DbProviderFactory » qui représente toute fabrique au sein du Framework .Net.

Nous sommes donc capable de récupérer notre fabrique et de lui demander de nous créer des connexions, des commandes\*, des paramètres, des adaptateurs de données, etc.



\* Il reste conseillé de demander à la connexion de créer les commandes

COGNITIC - MORRE THIERRY ©2020

```
DbProviderFactory Factory = SqlClientFactory.Instance;
using (DbConnection c = Factory.CreateConnection())
{
    c.ConnectionString = @"Data source=Asus-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO; Integrated Security=true";

    c.Open();
    using (DbCommand cmd = c.CreateCommand())
    {
        cmd.CommandText = "Select ID, Nom from Test where Nom Like @Like";

        DbParameter PLike = Factory.CreateParameter();
        PLike.ParameterName = "Like";
        PLike.Value = "%e%";

        cmd.Parameters.Add(PLike);

        using (DbDataReader reader = cmd.ExecuteReader())
        {
            while(reader.Read())
            {
                Console.WriteLine($"{reader["ID"]} {reader["Nom"]}");
            }
        }
    }
}
```

123

## ABSTRACT FACTORY & ADO.NET

```
...public static class DbProviderFactories
{
    ...public static DbProviderFactory GetFactory(string providerInvariantName);
    ...public static DbProviderFactory GetFactory(DataRow providerRow);
    ...public static DbProviderFactory GetFactory(DbConnection connection);
    ...public static DataTable GetFactoryClasses();
}
```



La classe « DbProviderFactories » n'est pas disponible dans les bibliothèques standard 2.0 (Core 3.1), mais ont été réimplémentées dans les bibliothèques standard 2.1 (.Net 5)

Ceci est presque parfait. Mais nous pouvons récupérer une fabrique particulière plus simplement encore.

En effet, il existe la classe statique « DbProviderFactories » qui sert de 'bibliothèque' de fabriques.

Cette classe propose deux méthodes :

« GetFactoryClasses » : qui retourne une table contenant tous les fournisseurs d'accès installés et disponibles au niveau de votre projet

« GetFactory » : qui retourne une fabrique spécifique soit sur base de son espace de noms, d'une « DataRow » provenant de la « DataTable » retournée par la première méthode ou d'une connexion déjà existante.

COGNITIC - MORRE THIERRY ©2020

124

124

## ABSTRACT FACTORY & ADO.NET

```

DbProviderFactory Factory =
    DbProviderFactories.GetFactory("System.Data.SqlClient");

using (DbConnection c = Factory.CreateConnection())
{
    c.ConnectionString = @"Data source=Asus-TM\SQL2016DEV;" +
        "Initial Catalog=DemoADO; Integrated Security=true";

    c.Open();
    using (DbCommand cmd = c.CreateCommand())
    {
        cmd.CommandText = "Select ID, Nom from Test where Nom Like @Like";

        DbParameter PLike = Factory.CreateParameter();
        PLike.ParameterName = "Like";
        PLike.Value = "%e%";

        cmd.Parameters.Add(PLike);

        using (DbDataReader reader = cmd.ExecuteReader())
        {
            while(reader.Read())
            {
                Console.WriteLine($"{reader["ID"]} {reader["Nom"]}");
            }
        }
    }
}

```

COGNITIC - MORRE THIERRY ©2020

125

125

## EXERCICE

C# - ADO.NET

COGNITIC - MORRE THIERRY ©2020

126

126

## EXERCICE

- Analysez les besoins et modifiez votre DLL de manière à ce que celle-ci utilise le pattern « abstract factory » et puisse de ce fait se connecter à tous types de base de données.

COGNITIC - MORRE THIERRY ©2020

127

127

## RÉFÉRENCES

C# - LES FONDEMENTS

COGNITIC - MORRE THIERRY ©2020

128

128



- 
- O'Reilly :  
C# 4.0 in a nutshell (ISBN-13 : 978-0-596-80095-6)  
C# 5.0 in a nutshell (ISBN-13 : 978-1-4493-2010-2)
  - MSDN Microsoft :  
Spécification du langage C# 5.0 ([http://msdn.microsoft.com/fr-fr/library/vstudio/ms228593\(v=vs.110\).aspx](http://msdn.microsoft.com/fr-fr/library/vstudio/ms228593(v=vs.110).aspx))  
ADO.Net ([https://msdn.microsoft.com/fr-FR/library/e80y5yhx\(v=vs.110\).aspx](https://msdn.microsoft.com/fr-FR/library/e80y5yhx(v=vs.110).aspx))
  - Design Pattern :  
Design Patterns: Elements of Reusable Object-Oriented Software (ISBN-13 : 978-0201633610)  
DoFactory (<http://www.dofactory.com/net/design-patterns>)