

Laboratoire 4**ADO.NET en mode déconnecté par programmation
Développer une application ADO.NET
Programmation de transactions****1. Objectifs d'apprentissage**

- Utiliser *ADO.NET* en mode déconnecté mais par programmation.
- Développer une application à l'aide des assistants de *Visual Studio*, de la programmation et du mode connecté.
- Programmer des transactions.

2. Utilisation d'ADO.NET par programmation

Toutes les vues (*TextBox*, *DataGridView*) créées dans les laboratoires 2 et 3 sont liées à un modèle (*DataSet*). C'est un contrôleur (*DataBindings*) qui fait le lien entre les vues et le modèle.

Cela ne pose aucun problème lorsque la *DataTable* présente dans le *DataSet* est reliée à une table réelle dans la base de données. Il est possible, pour le contrôleur, de gérer les mises à jour, les insertions et les suppressions.

Par contre, si la *DataTable* est reliée à plusieurs tables réelles dans la base de données, le contrôleur ne peut pas gérer les mises à jour, les insertions et les suppressions.

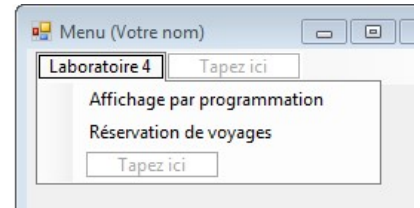
Lorsqu'un formulaire utilise des vues qui affichent des données à partir de plusieurs tables, la meilleure façon est de créer manuellement ces vues et de programmer les opérations de manipulation de la base de données sans passer par le contrôleur.

L'exemple qui suit va vous permettre de vous familiariser avec les objets utilisés pour la mise en œuvre de ces opérations.

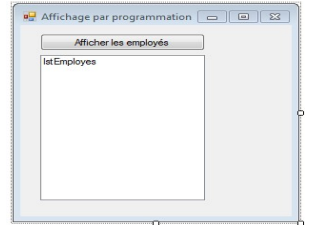
2.1. Affichage, par programmation, en mode déconnecté, des noms et prénoms de tous les employés

- Créez un nouveau projet nommé **Labo4** en utilisant le modèle *Application Windows Desktop* (en *Visual C#*)/*Application Windows Forms*

- Nommez votre formulaire **frmMnenu** dont le titre est *Menu (Votre nom)* puis ajoutez un menu (un *MenuStrip*) avec les deux (2) éléments suivants : *Affichage par programmation* et *Réservation de voyages*.



- Ajoutez un nouveau formulaire nommé **frmAffichageParProgrammation** dont le titre est *Affichage par programmation* composé d'un bouton de commande **btnAfficherEmployes** (dont le titre est *Afficher les employés*) et d'une liste simple (un *ListBox*) nommée **lstEmployes**.



- Pour éviter que les utilisateurs puissent ajouter des éléments dans la liste **lstEmployes**, mettre la valeur de la propriété *Locked* à *True*.
- Rattachez ce formulaire à **frmMenu**.
- Pour pouvoir utiliser les classes fournies par le fournisseur de données *SQL Serveur*, au début de la fenêtre de code, tapez *using System.Data.SqlClient*;
- Déclarez les objets suivants. Ils seront utilisés dans plusieurs événements. Votre connexion doit être une connexion authentifiée par *SQL Serveur*. Vous n'êtes pas obligés de taper les commentaires mais lisez-les pour comprendre.

```
public partial class frmAffichageParProgrammation : Form
{
    // La chaîne de connexion pour se connecter à la base de données
    String maChaineDeConnexion = "Tapez ici votre chaîne de connexion. La BD utilisée est BDVoyagesVotreNom";

    // Objet SqlConnection pour établir une connexion à la BD
    SqlConnection maConnexion = new SqlConnection();

    // Objet DataSet dans lequel nous allons ajouter des DataTables
    DataSet monDataSet = new DataSet();

    public frmAffichageParProgrammation()
    {
        InitializeComponent();
    }
}
```

- Dans l'événement *Click* du bouton de commande **btnAfficherEmployes**, tapez les instructions de la page suivante.

```

private void btnAfficherEmployes_Click(object sender, EventArgs e)
{
    maConnexion.ConnectionString = maChaineDeConnexion;

    string maRequeteSQL = "SELECT empPrenom, empNom from employee";

    SqlDataAdapter dataAdapterEmployes = new SqlDataAdapter(maRequeteSQL, maConnexion);

    // détruit dataTable prenomNomEmployes si elle existe déjà dans le Dataset
    if (monDataSet.Tables.Contains("prenomNomEmployes"))
    {
        monDataSet.Tables.Remove("prenomNomEmployes");
    }
    // crée la DataTable prenomNomEmployes et la remplit
    dataAdapterEmployes.Fill(monDataSet, "prenomNomEmployes");

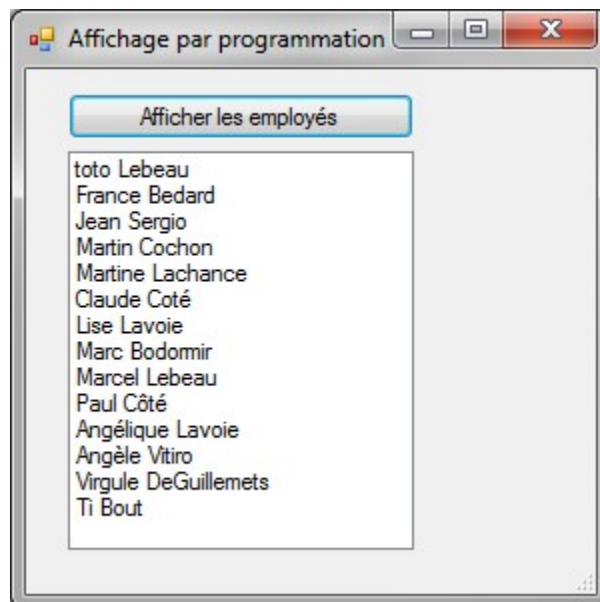
    if (monDataSet.Tables["prenomNomEmployes"].Rows.Count == 0)
    {
        MessageBox.Show("La table des employés est vide", "Erreur");
    }
    else
    {
        // vide la liste
        lstEmployes.Items.Clear();

        foreach (DataRow ligneEmploye in monDataSet.Tables["prenomNomEmployes"].Rows)
        {
            string prenomNomEmploye = ligneEmploye["empPrenom"] + " " + ligneEmploye["empNom"];

            lstEmployes.Items.Add(prenomNomEmploye);
        }
    }
}

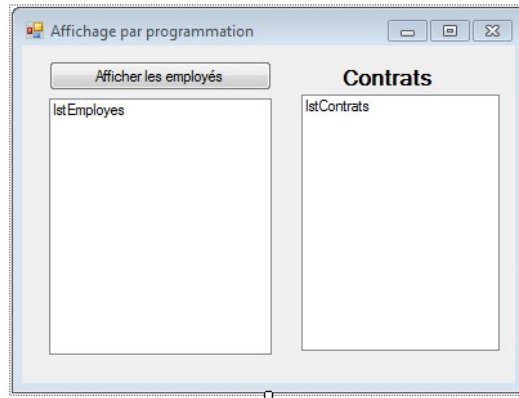
```

➤ Exécutez. Voici le résultat.



2.2. Affichage par programmation, en mode déconnecté, des contrats d'un employé sélectionné dans la liste des employés

➤ Complétez le formulaire **frmAffichageParProgrammation** comme suit :



- Pour éviter que les utilisateurs puissent ajouter des éléments dans la liste **lstContrats**, mettre la valeur de la propriété *Locked* à *True*.
- Vous devez programmer ce qui suit dans l'événement *SelectedIndexChanged* de la liste des employés **lstEmployes**. Cet événement se produit lorsqu'il y a une nouvelle sélection dans la liste **lstEmployes**.

```
private void lstEmployes_SelectedIndexChanged(object sender, EventArgs e)
{
}

```

- Vérifiez si la *DataTable* **prenomNomEmployes** existe dans votre *DataSet*. Si elle n'existe pas, affichez un message d'erreur.
- Si elle existe, faire ce qui suit :
 - Récupérez le prénom ainsi que le nom de l'employé dans la *DataTable* **prenomNomEmployes** et ce, à partir de la sélection qui a été faite par l'utilisateur dans la liste. Vous devez utiliser l'indice de l'employé dans la liste **lstEmployes** (c'est le même que celui de la *DataTable* **employes**).

Programmez cela de la manière suivante :

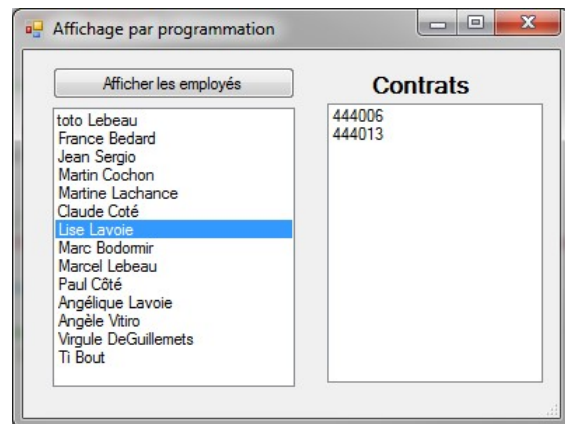
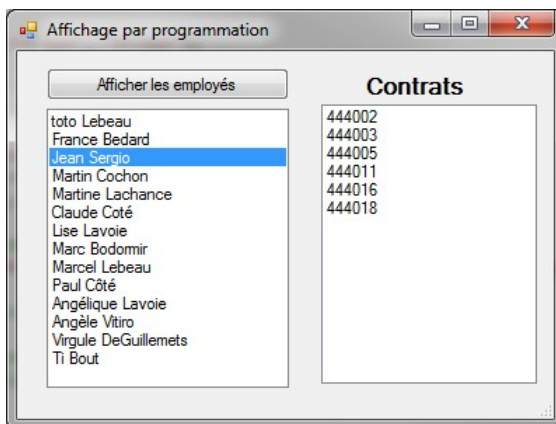
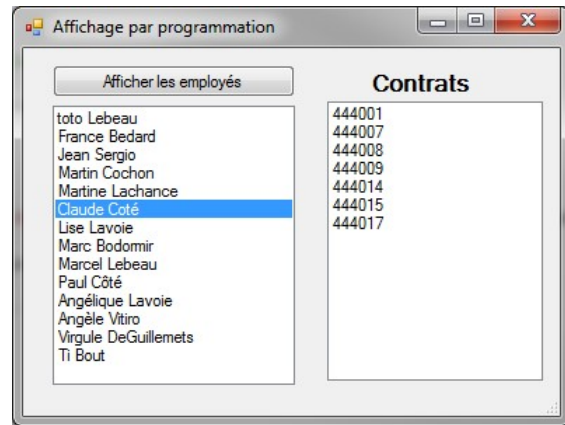
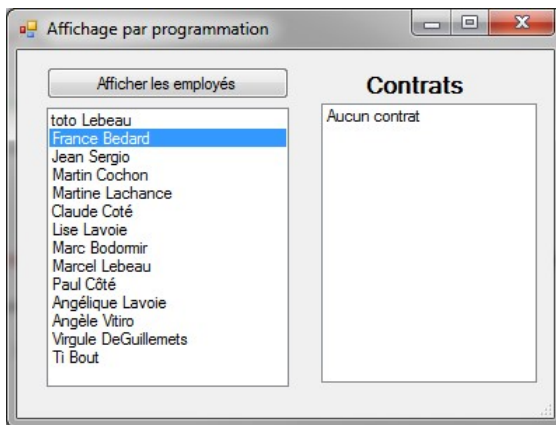
```
int indiceSelection = lstEmployes.SelectedIndex;
DataRow ligneEmploye = monDataSet.Tables["prenomNomEmployes"].Rows[indiceSelection];
String prenomEmploye = ligneEmploye["empPrenom"].ToString();
String nomEmploye = ligneEmploye["empNom"].ToString();

```

- Créez la requête *SQL* qui sélectionne tous les numéros de contrat de l'employé à partir de son nom et de son prénom.
- Créez un *DataAdapter* nommé **dataAdapterContratsEmploye** pour qu'il accepte votre requête *SQL* et votre connexion.
- Vérifiez si la *DataTable* **contratsEmploye** existe dans votre *DataSet*. Si c'est le cas, la détruire.

- À l'aide de votre *DataAdapter*, créez la *DataTable* **contratsEmploye** dans votre *DataSet* puis remplissez-la.
- Placez les numéros de contrats dans la liste **lstContrats** sans oublier de la vider au départ. S'il n'y a pas de contrats pour l'employé sélectionné, vous devez mettre *Aucun contrat* dans la liste **lstContrats** (voir des exemples d'exécution plus bas).

Quatre exemples d'exécution :

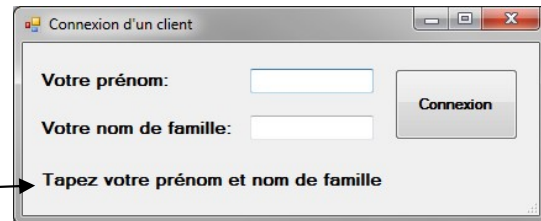


3. La réservation de voyages

On veut permettre à un client d'accéder à un formulaire qui consiste à réserver ou annuler des voyages.

3.1. Création d'un formulaire pour la connexion d'un client en mode connecté

Lorsque le client clique sur le bouton *Connexion*, si les informations qu'il a tapées sont correctes, on affiche le message « *Connexion réussie* » sinon on affiche « *Connexion échouée* » (affichez votre message ici)



- Ajoutez le formulaire nommé **frmConnexionClient** dans votre projet.
- Rattachez ce formulaire pour la *Réservation de voyages* dans **frmMenu**.
- Vous devez programmer ce qui suit dans l'événement *Click* du bouton **Connexion** :
 - Créez un objet *SqlConnection* dans lequel vous devez mettre votre chaîne de connexion *SQL Seveur*. Par la suite, ouvrez votre connexion (voir démo du cours 1). Notez qu'ici on doit ouvrir explicitement la connexion (ce n'est pas le cas avec le *DataAdapter*).
 - Créez un objet *SqlCommand* avec des paramètres *SQL Serveur* (voir démo du cours 1 pour le paramétrage). La requête SQL doit sélectionner le no de client dans la table **client** qui correspond au prénom et au nom de famille.
 - Exécutez la commande en utilisant la méthode *ExecuteScalar()* puis selon le résultat retourné, affichez le bon message : si le no de client existe, vous devez afficher « *Connexion réussie* » sinon vous devez afficher « *Connexion échouée* »
 - Fermez la connexion (pour libérer les ressources).
- Testez avec des clients qui existent (par exemple, le client **Ned Triton** et le client **Achille Talon**). Testez également avec des clients qui n'existent pas.



3.2. Création d'un formulaire en utilisant les assistants et un peu de programmation

Illustration de ce que vous allez devoir programmer:

Lorsque la connexion de la section précédente est réussie, un formulaire qui permet au client de réserver des voyages s'affiche.

The screenshot shows a window titled "Réservation de voyages". At the top, there is a "Destination:" label followed by a dropdown menu currently showing "Boston". To the right of this is a button labeled "Ajouter le voyage sélectionné à vos réservations". Below the destination field is a section titled "Voyages" containing a table with the following data:

Numéro	Date de départ	Date d'arrivée	animateur	Destination	Hôtel
3001	2019-04-04	2019-04-08	Marc Bodomir	Boston	Ritz, Boston
3003	2019-08-07	2019-08-12	Marc Bodomir	Boston	Days INN, Boston
3008	2019-06-07	2019-06-23	Ti Bout	Boston	Delta, Boston
3009	2019-06-06	2019-06-23	Virgule DeGuillemets	Boston	Delta, Boston

Le client choisit une destination dans une liste combinée.

A close-up of the "Destination:" dropdown menu, which is currently set to "Boston".

Le client a la possibilité de réserver un nouveau voyage. Pour ce faire, il doit sélectionner le voyage dans la liste des voyages disponibles puis cliquer sur le bouton *Ajouter le voyage sélectionné à vos réservations*.

Par exemple, voici ce que cela donne lorsque *Achille Talon* réserve le voyage 3009 à destination de *Boston*.

Réservation de voyages

Destination:

Voyages

Numéro	Date de départ	Date d'arrivée	animateur	Destination	Hôtel
3001	2019-04-04	2019-04-08	Marc Bodomir	Boston	Ritz, Boston
3003	2019-08-07	2019-08-12	Marc Bodomir	Boston	Days INN, Boston
3008	2019-06-07	2019-06-23	Ti Bout	Boston	Delta, Boston
3009	2019-06-06	2019-06-23	Virgule DeGuillemets	Boston	Delta, Boston

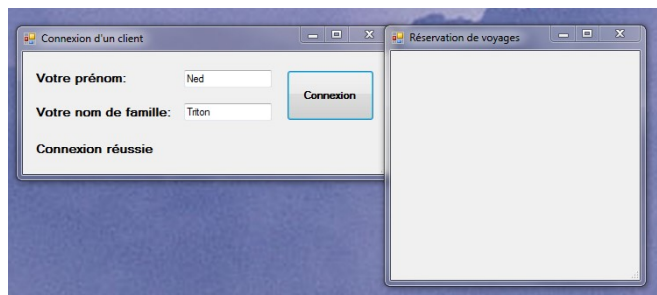
Vos voyages réservés:

No de contrat	Date du contrat	No de client	No de voyage	Date de départ	Date d'arrivée	Destination	Hôtel
444007	2019-02-23	123009	3001	2019-04-04	2019-04-08	Boston	Ritz, Boston ...
444019	2019-08-28	123009	3009	2019-06-06	2019-06-23	Boston	Delta, Boston ...

Le client a également la possibilité de supprimer une réservation. Pour ce faire, doit sélectionner un des voyages qu'il a réservés puis de cliquer sur le bouton *Supprimer la réservation sélectionnée*.

Voici les étapes pour réaliser cette partie :

- Ajoutez le formulaire nommé **frmReservationVoyages** dans votre projet. Son titre doit être *Réservation de voyages*.
- Dans le formulaire **frmConnexionClient**, si la connexion est réussie, instanciez un formulaire **frmReservationVoyages** puis affichez-le dans une fenêtre modale (méthode **ShowDialog()**).
- Pour que, dans le formulaire **frmReservationVoyages**, on ait accès au numéro de client, déclarez dans ce formulaire un attribut public dans la classe.



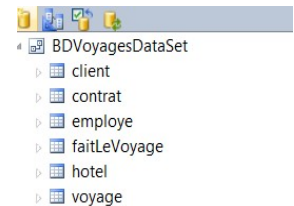

```
public partial class frmReservationVoyages : Form
{
    // Le numéro du client qui va effectuer des réservations
    public String noClient;
```

- Puis, dans le formulaire **frmConnexionClient**, juste avant d'afficher le formulaire **frmReservationVoyages**, tapez ce qui suit :

```
frv.noClient = noClient.ToString();
```

Frv est une instance de **frmReservationVoyages** et **noClient** une variable qui contient le numéro du client.

- À l'aide de l'assistant, ajoutez une nouvelle source de données (un *DataSet*) à votre projet pour la base de données **BDVoyagesVotreNom**. Utilisez une connexion authentifiée par *SQL Serveur*.
- À l'aide du concepteur du *DataSet*, ajoutez un *TableAdapter* pour le *DataSet* qui utilise la requête suivante pour récupérer toutes les destinations à partir de la table **voyage**.



Quelles données doivent être chargées dans la table ?

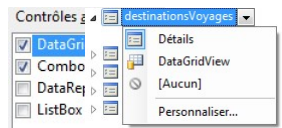
```
SELECT DISTINCT voyDestination FROM voyage
```

Notez l'utilisation du mot *DISTINCT* car plusieurs voyages peuvent avoir la même destination.

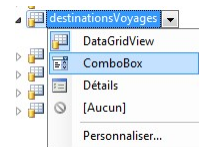
- Renommez la nouvelle DataTable, **destinationsVoyages**.
- Pour afficher les destinations dans une liste combinée (un *ComboBox*) qui va suivre les étapes suivantes :

- Dans la liste de la *DataTable* **destinationsVoyages**, sélectionnez *Personnaliser...*

- Dans les options, ajoutez *ComboBox* puis fermez la boîte de dialogue.



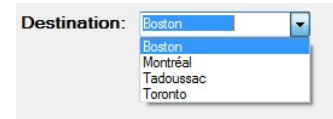
- Assurez-vous que la liste combinée (le *ComboBox*) est sélectionnée pour la *DataTable* **destinationsVoyages**.



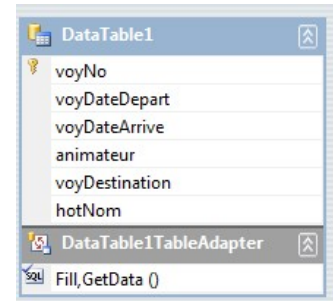
- Glissez la *DataTable* **destinationsVoyages** sur le formulaire puis supprimez complètement la barre de navigation située sur le formulaire (on n'en a pas de besoin ici).



- Ajoutez l'étiquette *Destination* : sur le formulaire à gauche de la liste combinée.
- Exécutez le tout. Vous devriez voir les quatre destinations s'afficher dans la liste combinée.

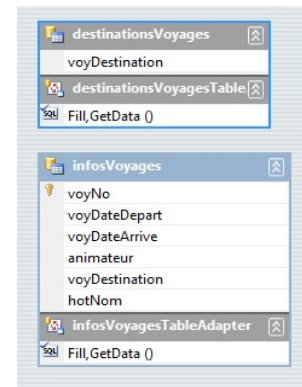


- Pour obtenir les informations sur les voyages, à l'aide du concepteur du *DataSet*, ajoutez un *TableAdapter* pour le *DataSet* qui permettrait d'obtenir le numéro de voyage, la date de départ, la date d'arrivée, le prénom et le nom de famille de l'animateur (de l'employé), la destination et le nom de l'hôtel.



Utilisez le générateur de requêtes. Le numéro de voyage, la date de départ, la date d'arrivée et la destination sont situés dans la table **voyage**. Le prénom et le nom de famille de l'animateur sont situés dans la table **employe**. Le nom de l'hôtel est situé dans la table **hotel**.

Modifiez la requête de manière à ce que le prénom et le nom de l'animateur soient sur une seule colonne. Donnez le nom *animateur* à cette colonne.



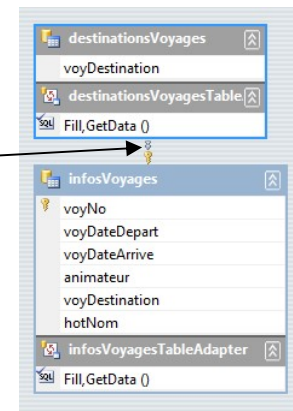
- Renommez la *DataTable*, **infosVoyages**.
- Pour l'instant, la *DataTable* **destinationsVoyages** contient les destinations et la *DataTable* **infosVoyages** contient les informations sur tous les voyages. Il n'y a aucune relation entre les deux. Vous pouvez voir cela dans le concepteur du *DataSet*.
- Pour que le contrôleur ne sélectionne que les voyages qui concernent la destination sélectionnée dans la liste combinée (*ComboBox*), nous devons relier les deux *DataTables*. Pour ce faire, sélectionnez les deux *DataTables*, cliquez-droit puis *Ajouter* puis *Relation*.

Remplissez la relation de la manière suivante puis cliquez sur *OK*.

Table parente :	Table enfant :
destinationsVoyages	infosVoyages
Colonnes :	
Colonnes clés	Colonnes clés étrangères
voyDestination	voyDestination

En faisant cela, dans la *DataTable* **destinationsVoyages**, le contrôleur va sélectionner seulement les voyages qui concernent la destination sélectionnée dans la liste combinée.

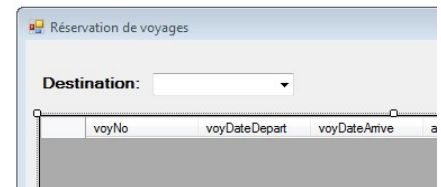
Vous pouvez observer la relation qu'il y a entre les deux *DataTables*.



Fermez cette fenêtre.

- À présent, il faut créer le *DataGridView* qui va afficher les informations sur les voyages à partir de la destination sélectionnée :

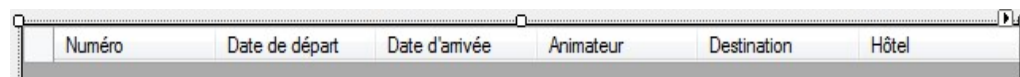
- Dans la fenêtre *Sources de données*, développez la *DataTable* **destinationsVoyages**, choisissez la *DataTable* **infosVoyages** puis choisissez *DataGridView* dans la liste de choix puis faites glisser la *DataTable* **infosVoyages** sur le formulaire en dessous de la liste combinée.



- Modifiez les propriétés du *DataGridView* comme suit :

Propriété	Valeur
Name	dgInfosVoyages
AllowUserToAddRows	False
AllowUserToDeleteRows	False
AutoSizeColumnsMode	Fill
ReadOnly	True
RowHeadersWidth	20
SelectionMode	FullRowSelect

- Modifiez également les titres de colonnes :



- Juste au-dessus de ce *DataGridView*, sur le formulaire, ajoutez une étiquette *Voyages*:
- Exécutez et vérifiez que tout fonctionne bien.

Réservation de voyages

Destination: Ajouter le voyage sélectionné à vos réservations

Voyages

Número	Date de départ	Date d'arrivée	animateur	Destination	Hôtel
3002	2019-07-17	2019-07-18	Paul Côté	Montréal	Voyageur, Montréal
3006	2019-06-08	2019-06-11	Ti Bout	Montréal	Universel, Montréal
3007	2019-06-03	2019-06-05	Ti Bout	Montréal	Universel, Montréal

- **Remarque** : En mémoire, la *DataTable* **infosVoyages** est complète. Elle contient tous les voyages sans exceptions. Lorsque vous sélectionnez une destination, le contrôleur filtre les enregistrements de cette *DataTable* de manière à ce que l'utilisateur ne voie que ceux qui concernent la destination sélectionnée.
- Maintenant, nous devons programmer « l'obtention » des informations sur les voyages qui sont présentement réservés par le client qui vient de réussir la connexion.
- Pour ce faire, ajoutez un *TableAdapter* pour le *DataSet* qui va permettre d'obtenir le numéro de contrat et la date du contrat (dans la table **contrat**), le numéro du client qui fait le voyage (dans la table **faitLeVoyage**), le numéro du voyage, la date de départ, la date d'arrivée et la destination (dans la table **voyage**) ainsi que le nom de l'hôtel (dans la table **hotel**).
 - Utilisez le générateur de requêtes puis ajoutez la clause **WHERE** (pour que les réservations s'appliquent à ce client seulement).

```
SELECT  contrat.conNo, contrat.conDate, faitLeVoyage.cliNo, voyage.voyNo, voyage.voyDateDepart,
voyage.voyDateArrive, voyage.voyDestination, hotel.hotNom
FROM    contrat INNER JOIN
        voyage ON contrat.voyNo = voyage.voyNo INNER JOIN
        hotel ON voyage.hotno = hotel.hotNo INNER JOIN
        faitLeVoyage ON contrat.conNo = faitLeVoyage.conNo
WHERE   (faitLeVoyage.cliNo = @noClient)
```

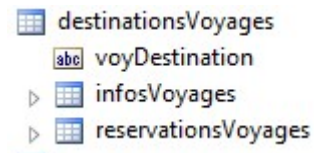
- Renommez la *DataTable*, **reservationsVoyages**.
- Ajoutez une relation entre la *DataTable* **destinationsVoyages** (qui contient les destinations) et la *DataTable* **reservationsVoyages** qui contient les informations sur les voyages présentement réservés par le client.

Table parente : destinationsVoyages Table enfant : reservationsVoyages

Colonnes :

Colonnes clés	Colonnes clés étrangères
voyDestination	voyDestination

- Pour afficher les voyages réservés par le client, dans la fenêtre *Sources de données*, développez la **DataTable destinationsVoyages**, choisissez la **DataTable reservationsVoyages** puis choisissez **DataGridView** dans la liste de choix puis faites glisser la **reservationsVoyages** sur le formulaire en dessous du **DataGridView** précédent.



- Observez qu'une barre d'outils a été automatiquement ajoutée sur le formulaire. Cette barre d'outils est présente pour donner la possibilité à l'utilisateur d'entrer un numéro de client. Dans le cas qui nous concerne, nous n'avons pas de besoin de cette barre d'outils car ce numéro de client est un attribut du formulaire (voir page 9).



- Supprimez cette barre d'outils et puis, à la fin de l'événement **Load** du formulaire **frmReservationVoyages**, ajoutez l'instruction suivante :

```
reservationsVoyagesTableAdapter.Fill(bdVoyagesVotreNomDataSet.reservationsVoyages, Convert.ToDecimal(noClient));
```

Au chargement du formulaire, la **DataTable reservationsVoyages** va être automatiquement remplie par tous les voyages vers la destination sélectionnée qui ont été réservés par le client **noClient**.

- Au-dessus du **DataGridView**, sur le formulaire, ajoutez une étiquette *Vos voyages réservés* :
- Modifiez les propriétés du **DataGridView** comme suit :

Propriété	Valeur
Name	dgReservationsVoyages
AllowUserToAddRows	False
AllowUserToDeleteRows	False
AutoSizeColumnsMode	Fill
ReadOnly	True
RowHeadersWidth	20
SelectionMode	FullRowSelect

- Modifiez également les titres de colonnes :

No de contrat	Date du contrat	No de client	No de voyage	Date de départ	Date d'arrivée	Destination	Hôtel

- Exécutez l'application pour tester. Testez-la avec le client *Achille Talon* (client no 123009). Testez différentes destinations dont celle de *Montréal*.

Réservation de voyages

Destination: Montréal
Ajouter le voyage sélectionné à vos réservations

Voyages

Número	Date de départ	Date d'arrivée	animateur	Destination	Hôtel
3002	2018-07-17	2018-07-18	Paul Côté	Montréal	Voyageur, Montréal
3006	2018-06-08	2018-06-11	Ti Bout	Montréal	Universel, Montréal
3007	2018-06-03	2018-06-05	Ti Bout	Montréal	Universel, Montréal

Vos voyages réservés:

No de contrat	Date du contrat	No de client	No de voyage	Date de départ	Date d'arrivée	Destination	Hôtel
444005	2018-05-24	123009	3002	2018-07-17	2018-07-18	Montréal	Voyageur, Montr...
444018	2017-12-10	123009	3007	2018-06-03	2018-06-05	Montréal	Universel, Montr...

Supprimer la réservation sélectionné

- Il faut maintenant programmer le bouton *Ajouter le voyage sélectionné à vos réservations*. L'ajout d'une réservation de voyage se fait par l'intermédiaire de deux tables : la table **contrat** (qui contient la liste de toutes les réservations de voyages) et la table **faitLeVoyage** (qui contient tous les numéros de clients qui font un voyage et qui sont liés à un contrat).

Nous devons donc ajouter un nouveau contrat pour le client avec le voyage sélectionné dans la table **contrat** puis une ligne dans la table **faitLeVoyage** qui contient le numéro du client et le numéro de contrat nouvellement inséré. Nous allons le faire en mode connecté.

Ajouter le voyage sélectionné à vos réservations

Ajoutez ce bouton sur le formulaire à la droite de la liste combinée et programmez ce qui suit dans son événement click.

- Créez un nouvel objet *Connexion* puis ouvrez-la. Votre chaîne de connexion doit être une connexion authentifiée par *SQL Serveur*.

Remarque : Si vous avez conservé votre chaîne de connexion dans le fichier *App.config*, utilisez la variable suivante :

Labo4.Properties.Settings.Default.BDVoyagesVotreNomConnectionString

- Vous devez exécuter deux requêtes (à l'aide de la méthode *ExecuteNonQuery()*).

Une première requête pour insérer le nouveau contrat dans la table **contrat**¹:

```
INSERT INTO contrat VALUES (@noContrat, @maintenant, 200.00, 1140.58, 200.00, 1, 1002,
@noVoyage, @noClient);
```

Une seconde requête pour insérer une ligne dans la table **faitLeVoyage**²:

```
INSERT INTO faitLeVoyage VALUES (@noContrat, @noClient);
```

Notez qu'il y a quatre paramètres *SQL Serveur* : *@noContrat*, *@maintenant*, *@noVoyage* et *@noClient*. Vous devez ajouter la valeur de ces quatre paramètres dans votre commande.

- La valeur du paramètre *@noClient* est facile. Ajoutez la valeur de ce paramètre dans votre objet *Command*.
- La valeur du paramètre *@noNoyage* est le numéro de voyage sélectionné directement dans le *DataGridView* **dgInfosVoyages** de la manière suivante.

```
// Le no de voyage
String noVoyage = dgInfosVoyages.CurrentRow.Cells[0].Value.ToString();
```

0 est le numéro de la colonne qui contient le numéro de voyage. On peut aussi utiliser le nom (propriété name) de la colonne numéro de voyage dans le *DataGridView* **dgInfosVoyages**

- La valeur du paramètre *@maintenant* est : :
DateTime.Now.Date (date/ heure actuelle)

¹ Notez, dans cette requête, que les différents montants du contrat ainsi que l'animateur sont déjà déterminés à l'avance. En réalité, on devrait demander, à l'utilisateur, les montants du contrat ainsi que l'animateur. Nous le ferons dans le laboratoire 6. Pour l'instant, nous allons nous contenter de valeurs par défaut.

² Notez, dans cette requête, qu'il n'y a qu'un client qui fait le voyage et que le client qui fait le voyage est le même client qui est responsable de la réservation de voyage (du contrat). En réalité, on devrait demander, à l'utilisateur, tous les clients qui font le voyage. Nous le ferons à la fin de la session. Pour l'instant, nous allons nous contenter de valeurs par défaut.

- La valeur du paramètre *@noContrat* correspond au plus grand numéro de contrat dans la table contrat auquel on ajoute 1.
Pour obtenir le plus grand numéro de contrat, exécutez la requête suivante (en mode connecté à l'aide de la méthode *ExecuteScalar()*) :

SELECT MAX(conNo) FROM contrat

- Après avoir exécuté les deux requêtes *INSERT*, la vraie base de données sur le serveur va être à jour. Par contre, la *DataTable* (située dans la mémoire de l'ordinateur) ne sera pas à jour. Par conséquent, l'utilisateur ne verra pas les résultats dans le *DataGridView*.

Pour mettre à jour la *DataTable* **reservationsVoyages** (qui contient les réservations), tapez la commande suivante :

```
reservationsVoyagesTableAdapter.Fill(bdVoyagesVotreNomDataSet.reservationsVoyages, Convert.ToDecimal(noClient));
```

Remarque : Vérifiez, avant la mise à jour, que l'utilisateur a bien sélectionné un voyage (*infosVoyagesBindingSource.Current != null*)

- Fermez la connexion (pour libérer les ressources).
- Testez votre application :
 - Dans la connexion, entrez *Achille Talon* en tant que client (client 123009).
 - Choisissez les voyages en direction de *Montréal*.
 - Sélectionnez le voyage 3006 et réservez ce voyage (cliquez sur le bouton *Ajouter le voyage...*).
 - Voici ce que vous devriez voir dans les voyages réservés (la date du contrat devrait être la date d'aujourd'hui).

Vos voyages réservés:

No de contrat	Date du contrat	No de client	No de voyage	Date de départ	Date d'arrivée	Destination	Hôtel
444005	2019-05-24	123009	3002	2019-07-17	2019-07-18	Montréal	Voyageur, Montr...
444018	2018-12-10	123009	3007	2019-06-03	2019-06-05	Montréal	Universel, Montr...
444019	2019-08-28	123009	3006	2019-06-08	2019-06-11	Montréal	Universel, Montr...

- Choisissez les voyages en direction de *Tadoussac*.
- Sélectionnez le voyage 3005 et réservez ce voyage (cliquez sur le bouton *Ajouter le voyage...*).
- Voici ce que vous devriez voir dans les voyages réservés.

Vos voyages réservés:

No de contrat	Date du contrat	No de client	No de voyage	Date de départ	Date d'arrivée	Destination	Hôtel
444020	2019-08-28	123009	3005	2019-06-12	2019-06-23	Tadoussac	Maison Dufour, ...

- Finalement, il faut programmer le bouton *Supprimer la réservation sélectionnée*. Nous devons donc, en mode connecté, supprimer la réservation sélectionnée dans la table **faitLeVoyage**. A la suite de cette suppression, s'il n'y a plus aucun client affecté par ce contrat, nous devons supprimer le contrat lui-même dans la table **contrat**.

Supprimer la réservation sélectionnée

Ajoutez ce bouton sur le formulaire en bas du *DataGridView* des réservations et programmez ce qui suit dans son événement *click*.

Si une réservation a été bien sélectionnée, effectuer les opérations suivantes :

- Créez un nouvel objet *Connexion* puis ouvrez-la. Votre chaîne de connexion doit être une connexion authentifiée par *SQL Serveur*.
- Vous devez exécuter deux requêtes (à l'aide de la méthode *ExecuteNonQuery()*).

Une première requête pour supprimer ce client dans la table **faitLeVoyage**:

```
DELETE FROM faitLeVoyage WHERE conNo = @noContrat AND cliNo = @noClient;
```

Une seconde requête pour supprimer le contrat s'il n'y a plus aucun client qui fait ce voyage.

```
DELETE FROM contrat WHERE conNo = @noContrat AND conNo NOT IN (SELECT conNo  
FROM faitLeVoyage);
```

Observez qu'il y a deux (2) paramètres *SQL Serveur* : *@noContrat* et *@noClient*. Vous devez ajouter la valeur de ces deux (2) paramètres dans votre commande.

- La valeur du paramètre *@noClient* est facile. Ajoutez la valeur de ce paramètre dans votre objet *Command*.
- La valeur du paramètre *@noContrat* est le numéro de contrat sélectionné directement dans le *DataGridView* **dgReservationsVoyages** de la manière suivante.

```
// Le no de contrat  
String noContrat = dgReservationsVoyages.CurrentRow.Cells[0].Value.ToString();
```

- Après avoir exécuté les deux requêtes *DELETE*, la vraie base de données sur le serveur va être à jour. Par contre, la *DataTable* (située dans la mémoire de l'ordinateur) ne sera pas à jour. Par conséquent, l'utilisateur ne verra pas les résultats dans le *DataGridView*.

Pour mettre à jour la *DataTable* **reservationsVoyages** (qui contient les réservations), tapez la commande suivante :

```
reservationsVoyagesTableAdapter.Fill(bDVoyagesVotrenomDataSet.reservationsVoyages, Convert.ToDecimal(noClient));
```

- Fermez la connexion (pour libérer les ressources).
- Testez votre application :
 - Dans la connexion, entrez *Achille Talon* en tant que client (client 123009).
 - Choisissez les voyages en direction de *Montréal*.
 - Sélectionnez le no de contrat *444019* et supprimez cette réservation (cliquez sur le bouton).
 - Voici ce que vous devriez voir dans les voyages réservés.

Vos voyages réservés:								
No de contrat	Date du contrat	No de client	No de voyage	Date de départ	Date d'arrivée	Destination	Hôtel	
444005	2018-05-24	123009	3002	2018-07-17	2018-07-18	Montréal	Voyageur, Montr...	
444018	2017-12-10	123009	3007	2018-06-03	2018-06-05	Montréal	Universel, Montr...	

- Choisissez les voyages en direction de *Tadoussac*.
- Sélectionnez le no de contrat *444020* et supprimez cette réservation.
- Voici ce que vous devriez voir dans les voyages réservés.

Vos voyages réservés:								
No de contrat	Date du contrat	No de client	No de voyage	Date de départ	Date d'arrivée	Destination	Hôtel	

Il est important de prendre note que ces 2 contrats devraient être supprimés dans la table **contrat**. Vérifiez cela.

3.3. Les transactions

Noter que l'ajout et la suppression de réservations de voyages nécessitent chacune deux opérations dans la base de données. Deux insertions dans le cas de l'ajout et deux suppressions dans le cas de la suppression. Pour assurer un état cohérent de la base de données, il est important que ces opérations soient effectuées au sein d'une transaction.

En *ADO.NET*, il est possible de gérer des transactions. Voici la manière que vous devez procéder.

- Faites cela dans l'événement *click* du bouton de commande qui ajoute le voyage sélectionné dans les réservations.

- Juste après avoir ouvert la connexion, créez votre transaction de la manière suivante.

```
maConnexion.Open();  
SqlTransaction maTransaction = maConnexion.BeginTransaction();
```

- Exécutez toutes vos instructions à l'intérieur d'un *try*. Juste avant d'exécuter une requête quelconque, indiquez que cette requête fait partie de votre transaction de la manière suivante :

```
maCommande.Transaction = maTransaction;
```

- A la suite de la dernière requête exécutée, faites un *Commit* sur la transaction pour indiquer la transaction a réussie.

```
maTransaction.Commit();  
MessageBox.Show("Transaction réussie");
```

- Dans le *catch*, annulez l'exécution des requêtes qui ont déjà été exécutées (*rollback*).

```
catch  
{  
    maTransaction.Rollback();  
    MessageBox.Show("Transaction échouée");  
}
```

- Testez votre application.

- Pour tester l'échec d'une transaction, remplacez la requête:

```
INSERT INTO faitLeVoyage VALUES (@noContrat, @noClient);
```

par la requête suivante:

```
INSERT INTO faitLeVoyage VALUES (10, @noClient);
```

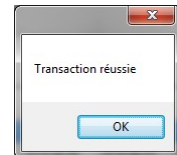
Cette requête va échouer car le contrat numéro 10 n'existe pas.

- A la fin du test, remplacez la requête suivante :

```
INSERT INTO faitLeVoyage VALUES (10, @noClient);
```

par

```
INSERT INTO faitLeVoyage VALUES (@noContrat, @noClient);
```



- Procédez de la même manière dans l'événement *click* du bouton de commande qui supprime une réservation.