

<b>Laboratoire 10</b> <b><i>LINQ To SQL – Partie 2</i></b>
---

## 1. Objectifs d'apprentissage

- Programmer une application complète qui utilise *LINQ To SQL*

## 2. L'application

Dans ce laboratoire, nous allons programmer une application qui utilise *LINQ to SQL*. Il s'agit d'ajouter des nouveaux contrats ainsi que de nouveaux voyageurs.

- Créez un nouveau projet **Labo 10** de type *Application Windows Forms Visual C#..*

Le titre du formulaire doit être *Ajout de contrats et de voyageurs* suivi de votre nom entre parenthèses.

- Ajouter un élément *Classes LINQ to SQL* à votre projet.

- À partir de *l'Explorateur des serveurs*, glisser toutes les tables de votre base de données **BDVoyagesVotreNom** dans la partie gauche du *Concepteur Objet/Relationnel*.

## 3. L'ajout d'un nouveau contrat

### La date

- Ajoutez, sur le formulaire, une étiquette et un *DateTimePicker* **dtpDate** qui offre la possibilité à l'utilisateur de choisir une date.
- La date ne doit pas être postérieure à la date actuelle. Ajouter un *ErrorProvider* **errMessage**. Programmez l'événement *validating* du contrôle *dtpDate* comme suit :

```
private void dtpDate_Validating(object sender, CancelEventArgs e)
{
    if (dtpDate.Value.Date > DateTime.Now.Date)
    {
        errMsg.SetError(dtpDate, "La date ne doit pas être postérieure à la date actuelle.");
        e.Cancel = true;
    }
    else errMsg.SetError(dtpDate, "");
}
}
```

- Exécutez pour tester.

## Les montants d'argent

- Ajoutez, sur le formulaire, trois étiquettes et trois *NumericUpDown* qui vont permettre de sélectionner des montants d'argent. Les noms des *NumericUpDown* sont respectivement **nudAcompte**, **nudMontant** et **nudPaye**.

- Pour les trois *NumericUpDown*, affecter les valeurs suivantes :

Propriété	Valeur
<i>DecimalPlaces</i>	2
<i>Increment</i>	0,1
<i>Maximum</i>	9999,99
<i>Minimum</i>	1

- Programmez l'événement *validating* de chacun des contrôles *NumericUpDown* comme suit:

```
private void nudAcompte_Validating(object sender, CancelEventArgs e)
{
    if (nudAcompte.Value > nudMontant.Value)
    {
        errMsg.SetError(nudAcompte, "L'acompte ne doit pas dépasser le montant du contrat");
        e.Cancel = true;
    }
    else errMsg.SetError(nudAcompte, "");
}

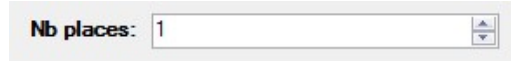
private void nudMontant_Validating(object sender, CancelEventArgs e)
{
    if (nudMontant.Value < nudPaye.Value)
    {
        errMsg.SetError(nudMontant, "Le montant payé ne doit pas dépasser le montant du contrat");
        e.Cancel = true;
    }
    else errMsg.SetError(nudMontant, "");
}

private void nudPaye_Validating(object sender, CancelEventArgs e)
{
    if (nudPaye.Value < nudAcompte.Value)
    {
        errMsg.SetError(nudPaye, "L'acompte ne doit pas dépasser le montant payé");
        e.Cancel = true;
    }
    else errMsg.SetError(nudPaye, "");
}
```

- Exécuter pour tester.

### Le nombre de places

- Ajoutez, sur le formulaire, une étiquette et un *NumericUpDown* **nudNbPlaces** qui va permettre de sélectionner le nombre de places.



- Pour ce *NumericUpDown*, affecter les valeurs suivantes :

Propriété	Valeur
<i>DecimalPlaces</i>	0
<i>Increment</i>	1
<i>Maximum</i>	4
<i>Minimum</i>	1

- Ne pas programmer pas son événement *validating* (puisque tout est déjà validé par le *NumericUpDown*).
- Exécutez pout tester.

### Les numéros d'employé, de voyage et de client

- L'employé, le voyage et le client seront choisis à partir de listes combinées
- Créer trois nouvelles classes comme suit :

```

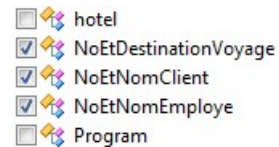
class NoEtNomEmploye
{
    public decimal noEmploye { get; private set; }
    public string nomCompleetEmploye { get; private set; }
}

class NoEtDestinationVoyage
{
    public decimal noVoyage { get; private set; }
    public string destinationVoyage { get; private set; }
}

class NoEtNomClient
{
    public decimal noClient { get; private set; }
    public string nomCompleetClient { get; private set; }
}

```

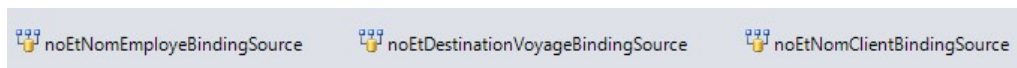
- Ajoutez ces trois classes comme source de données (en tant qu'objet).  
(regénérez la solution si les nouvelles classes n'apparaissent pas)



- Ajoutez, sur le formulaire, trois étiquettes et trois *ComboBox*: **cboEmploye**, **cboVoyage** et **cboClient** (la valeur de leur propriété *DropDownStyle* doit être *DropDownList*).

- Glissez la classe **NoEtNomEmploye** sur le *ComboBox* **cboEmploye**. Assurez-vous que le membre affiché est *nomCompleetEmploye* et sa valeur *noEmploye*.

- Répétez l'opération précédente pour le voyage et le client.
- Observez que trois contrôleurs ont été automatiquement ajoutés.



Nous allons utiliser ces contrôleurs et *LINQ to SQL* pour remplir les *ComboBox*.

- Créez un *DataContext* pour avoir accès aux données de la base de données.

```
public partial class Form1 : Form
{
    private DataClasses1DataContext dataContext = new DataClasses1DataContext();
}
```

- À l'aide de *LINQ*, dans l'événement *Load* du formulaire, remplir les *ComboBox* de comme suit :

```

// chargement des employés
noEtNomEmployeBindingSource.DataSource =
    from unEmploye in dataContext.employe
    select new
    {
        noEmploye = unEmploye.empNo,
        nomCompleetEmploye = unEmploye.empPrenom + " " + unEmploye.empNom
    };

// chargement des clients
noEtNomClientBindingSource.DataSource =
    from unClient in dataContext.client
    select new
    {
        noClient = unClient.cliNo,
        nomCompleetClient = unClient.cliPrenom + " " + unClient.cliNom
    };

// chargement des voyages
noEtDestinationVoyageBindingSource.DataSource =
    from unVoyage in dataContext.voyage
    let dateDepart = (DateTime)unVoyage.voyDateDepart
    let dateArrivee= (DateTime)unVoyage.voyDateArrive
    select new
    {
        noVoyage = unVoyage.voyNo,
        destinationVoyage = unVoyage.voyDestination.Trim() + " [" + dateDepart.Date+ "-->" + dateArrivee.Date + "]"
    };

```

- Exécutez pour tester :

The screenshot shows a form with three dropdown menus. The first dropdown is labeled 'Employé :' and contains the text 'toto Lebeau'. The second dropdown is labeled 'Voyage :' and contains the text 'Boston[2019-04-04-->2019-04-08]'. The third dropdown is labeled 'Client :' and contains the text 'Ned Triton'.

- Nous sommes maintenant prêts à ajouter ce contrat dans la base de données.

### L'ajout du contrat dans la base de données

- Ajoutez, sur le formulaire, un bouton de commande **btnEnregistrerContrat**.



- Programmez son événement *click* (ne pas oublier d'écrire *using System.Data.Linq* au début du formulaire) comme suit :

```

// récupère tous les numéros de contrats à partir de la base de données
var tousLesNosContrats =
    from contrat in dataContext.contrat
    select contrat.conNo;

// numéro du nouveau contrat
decimal noContrat = tousLesNosContrats.Max() + 1;

var nouveauContrat = new contrat
{
    conNo = noContrat,
    conDate = dtpDate.Value.Date,
    conAcompte = nudAcompte.Value,
    conMontant = nudMontant.Value,
    conPaye = nudPaye.Value,
    empNo = (decimal)cboEmploye.SelectedValue,
    voyNo = (decimal)cboVoyage.SelectedValue,
    cliNo = (decimal)cboClient.SelectedValue
};

//insère le contrat dans le contexte
dataContext.contrat.InsertOnSubmit(nouveauContrat);

// enregistre le contrat dans la base de données.
try
{
    dataContext.SubmitChanges();
    MessageBox.Show(" le contrat " + noContrat + "a été enregistré. ", "enregistrement du contrat");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, " impossible de modifier la base de données");
}

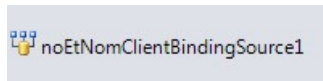
```

- Exécutez le tout et ajoutez des nouveaux contrats. Vérifiez que les contrats ont bien été ajoutés dans la base de données.

## 4. L'ajout des voyageurs dans la base de données

- Ajoutez, sur le formulaire, une étiquette et un *ListBox* **IstVoyageurs**. Sa propriété *SelectionMode* doit être *MultiExtended* (l'utilisateur pourra sélectionner plusieurs voyageurs).
- Glissez la source de données **NoEtNomClient** sur ce *ListBox*.

Un nouveau contrôleur a été créé.



- Modifiez l'événement *Load* du formulaire de la manière suivante :

```
// chargement des clients et des voyageurs
noEtNomClientBindingSource.DataSource = noEtNomClientBindingSource1.DataSource=
from unClient in dataContext.client
select new
{
    noClient = unClient.cliNo,
    nomCompletClient = unClient.cliPrenom + " " + unClient.cliNom
};
```

- Exécutez le tout et vérifiez que le *ListBox* est bien rempli.

Avant d'ajouter ces voyageurs dans la base de données, il faut vérifier que :

- Le nombre de voyageurs ne dépasse pas le nombre de places disponibles
- Le client qui signe le contrat fait partie du voyage.

- Programmez l'événement *validating* de ce *ListBox* comme suit :

```
if (lstVoyageurs.SelectedItems.Count != nudNbPlaces.Value)
{
    errMessage.SetError(lstVoyageurs, $"Vous devez sélectionner {nudNbPlaces.Value} voyageur(s)");
    e.Cancel = true;
}
else
{
    bool present = lstVoyageurs.SelectedItems.Cast<NoEtNomClient>()
        .Any(c => c.NoClient == (decimal) cboClient.SelectedValue);

    if (!present)
    {
        errMessage.SetError(lstVoyageurs, "Le responsable doit faire partie du voyage");
        e.Cancel = true;
    }
    else
    {
        errMessage.SetError(lstVoyageurs, string.Empty);
    }
}
```

- Exécutez le tout et vérifiez les deux messages d'erreur.

### La sauvegarde des voyageurs dans la base de données

- modifiez le titre du bouton **btnEnregistrerContrat**.



- Dans l'événement *click* du bouton **btnEnregistrerContrat**, juste avant le *try*, ajoutez les instructions suivantes et changez le message après le *submitChanges* comme suit :



```

dataContext.contrats.InsertOnSubmit(nouveauContrat);
// enregistre le contrat dans la base de données.

lstVoyageurs.SelectedItems.Cast<NoEtNomClient>().ToList()
    .ForEach(c =>
        dataContext.faitLeVoyages.InsertOnSubmit(new faitLeVoyage {conNo = noContrat, cliNo = c.NoClient}));

try
{
    dataContext.SubmitChanges();
    MessageBox.Show($"Le contrat {noContrat} et les voyageurs ont été enregistrés.", "Enregistrement du contrat");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Impossible de modifier la base de données");
}
}

```

- Exécutez. Vérifiez que les contrats et les voyageurs reliés au contrat ont bien été ajoutés dans la base de données.

**Remarque** : si vous affectez la valeur 4 au nombre de voyageurs et que vous cliquez sur enregistrer le contrat... l'enregistrement du contrat en cours sera réussi alors que vous n'avez pas sélectionné de voyageurs.

**Voyageurs:**

L'événement *Validating* de **lstVoyageurs** n'a pas été appelé. Pour qu'un événement *Validating* soit appelé, il faut que le contrôle soit en train de perdre le focus. Or, ici, **lstVoyageurs** n'a jamais reçu le focus.

Nous devons donc forcer l'appel à l'événement **Validating** de tous les contrôles situés sur le formulaire. Pour ce faire, nous pourrions donner le focus (par programmation) à chacun des contrôles situés sur le formulaire et vérifier s'il y a un message d'erreur dans le contrôle **errMessage**. S'il y a un message d'erreur, il y a au moins une donnée non valide. Cette solution n'est pas très élégante.

- Une autre manière de procéder consiste à provoquer tous les événements validating de tous les contrôles du formulaire comme suit:

```

private void btnEnregistrerContrat_Click(object sender, EventArgs e)
{
    if (this.ValidateChildren(ValidationConstraints.ImmediateChildren))
    {
        // Va chercher tous les nos de contrats
        var tousLesNosContrats =
            from unContrat in contexteContrats.contrat

```



## 5. Utilisation d'une transaction

Lors des l'enregistrement d'un contrat et des voyageurs, Il y a au moins deux opérations: l'ajout du contrat et l'ajout de chaque voyageur dans la base de données. Afin d'assurer la cohérence des données, nous allons regrouper ces opérations au sein d'une même transaction.

- Au début du formulaire, ajouter :

```
using System.Transactions;
```

**Note** : Si *Transactions* est souligné en rouge, cliquer sur *Projet*, puis sur *Ajouter une référence...* puis sur, l'onglet *.NET*, sélectionner *System.Transactions*.

- Modifiez l'événement *click* du bouton **btnEnregistrerContrat** comme suit :

```
lstVoyageurs.SelectedItems.Cast<NoEtNomClient>().ToList()
    .ForEach(c =>
        dataContext.faitLeVoyages.InsertOnSubmit(new faitLeVoyage {conNo = noContrat, cliNo = c.NoClient}));

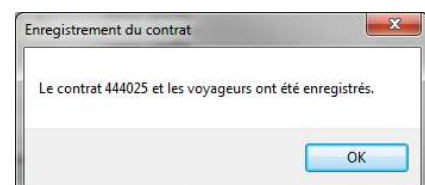
using (var porteeTransaction = new TransactionScope())
{
    try
    {
        dataContext.SubmitChanges();
        MessageBox.Show($"Le contrat {noContrat} et les voyageurs ont été enregistrés.", "Enregistrement du contrat");

        porteeTransaction.Complete();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Impossible de modifier la base de données");
    }
}
```

- Ici, nous avons créé une portée pour la transaction, c'est-à-dire que toutes les opérations (les requêtes) situées à l'intérieur de cette portée font partie de la même transaction. Si une des opérations échoue à l'intérieur de cette portée, toutes les opérations précédentes vont être annulées (*Rollback*).

Observez l'instruction *porteeTransaction.Complete()*. Cette instruction est essentielle. Cela signifie que toutes les opérations situées à l'intérieur de cette portée ont eu lieu avec succès. Si vous n'écrivez pas cette instruction, toutes les opérations vont échouer (par conséquent, par défaut, les opérations sont des échecs).

- Exécutez l'application puis ajoutez un contrat et des voyageurs. Notez le numéro de contrat qui est affiché dans la boîte de message.



- Mettez fin à l'exécution de l'application et, à l'aide de l'*Explorateur de serveurs*, vérifiez, dans la table **contrat** et dans la table **faitLeVoyage**, que ce numéro de contrat existe.

- Mettez en commentaires l'instruction *porteeTransaction.Complete()*.
- Exécutez l'application puis ajoutez un contrat et des voyageurs. Notez le numéro de contrat qui est affiché dans la boîte de message.
- Arrêtez l'application et, à l'aide de l'*Explorateur de serveurs*, vérifiez, dans la table **contrat** et dans la table **faitLeVoyage**, que ce numéro de contrat n'existe pas. Toutes les opérations ont été annulées.
- Enlevez le commentaire de l'instruction *porteeTransaction.Complete()*.

Ajout de contrats et de voyageurs

**Date :** 23 octobre 2019  
**Acompte :** 1,00  
**Montant :** 1,00  
**Payé :** 1,00  
**Nb Places :** 1  
**Employé :** toto Lebeau  
**Voyage :** Boston[2019-04-04-->2019-04-08]  
**Client :** Ned Triton

**Voyageurs :**

- Ned Triton
- Wallace Sefresti
- Gromit Sefresti
- Molton Sefresti
- Achille Talon
- Obelix Le Gaulois
- Hagar L'horrible
- Charlie Brown
- Lucien Tetebeche
- Arthur Laroche
- Délina Cailloux
- Bertha Laroche
- Poncho Sanchez
- Gaston Lagaffe
- Bianca Castafiore
- Capitaine Hadock
- Red Ketchup
- Bart Simpson
- Homer Simpson
- Fred Cailloux
- Toto Lebeau
- Onesime Lebeau

Enregistrer ce contrat et les voyageurs sélectionnées