

**Laboratoire 3**  
**Illustration du modèle MVC**  
**Programmation du contrôleur**  
**Liaison de données**  
**Chargement dynamique des données**

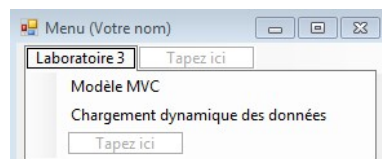
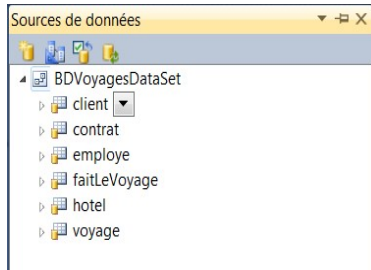
## 1. Objectifs d'apprentissage

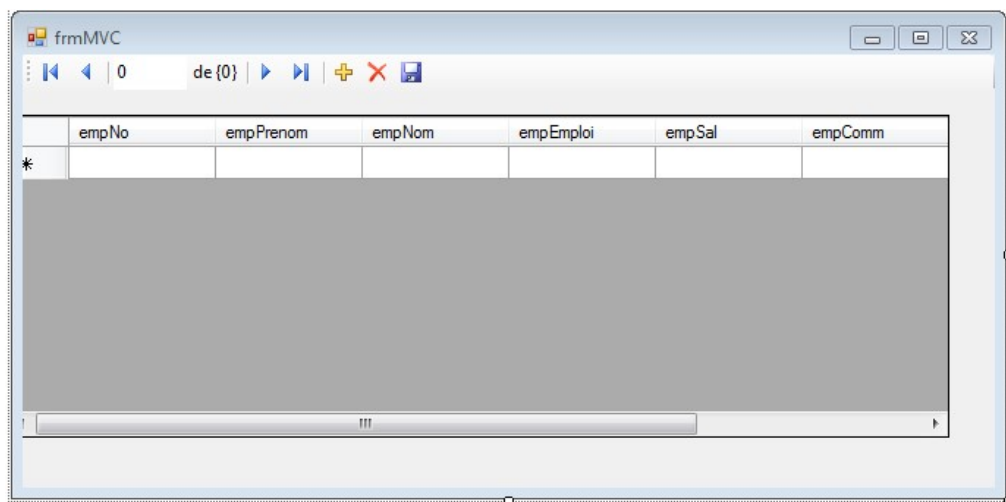
- Illustrer le modèle **MVC** (Modèle-Vue-Contrôleur)
- Programmer le contrôleur
- Comprendre la liaison des données
- Charger dynamiquement des données

## 2. Le modèle MVC (Modèle-Vue-Contrôleur)

En programmation, le modèle **MVC** est très important car il permet de séparer les trois composantes d'une application. Le **modèle** représente les données utilisées par l'application, la **vue** représente l'interface de l'application tandis que le **contrôleur** fait le lien entre le modèle et la vue; c'est le cœur de l'application.

### 2.1 Illustration du modèle MVC dans **ADO.NET**

- Créez un nouveau projet nommé **Labo3** en utilisant le modèle *Application Windows Desktop* (en *Visual C#*)/Application Windows Forms
- Nommez votre formulaire **frmMenu** dont le titre est *Menu (Votre nom)* puis ajoutez un menu (un *MenuStrip*) avec les deux éléments suivants : *Modèle MVC* et *Chargement dynamique des données*.  

- Ajoutez une nouvelle source de données. Cette source de données doit contenir le *DataSet* de toutes les tables de votre base de données **BDVoyagesVotreNom** (voir laboratoire 2).  

- Créez un formulaire nommé **frmMVC** et rattachez ce formulaire à **frmMenu**
- A partir de la *DataTable* **Employe**, ajoutez un *DataGridView* sur le formulaire **frmMVC**.

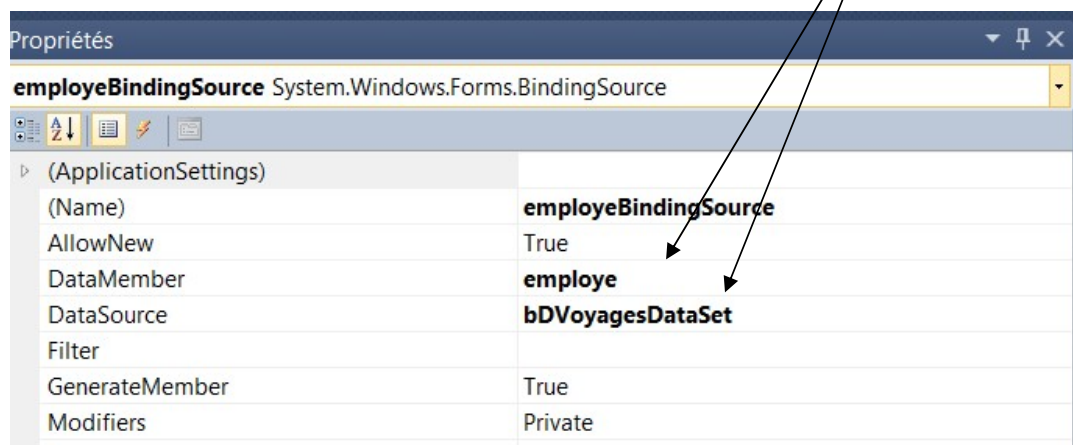


- Observez que 5 objets ont été automatiquement ajoutés sur votre formulaire.

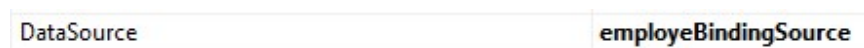


**bDVoyagesVotreNomDataSet** : Cet objet contient le *DataSet* de votre base de données. Au niveau du modèle **MVC**, cet objet représente le **Modèle**.

**employeBindingSource** : Cet objet établit la liaison des données. Au niveau du modèle **MVC**, cet objet représente le **Contrôleur**. La *DataTable* **employe** est directement reliée à ce contrôleur. Observez les propriétés *DataMember* et *DataSource* du contrôleur.



Au niveau du modèle **MVC**, le *DataGridView* est une **Vue**. Cette vue est reliée au contrôleur **employeBindingSource**. Observez la propriété *DataSource* du *DataGridView*.



**employeBindingNavigator** : Cet objet contient la barre de navigation. Au niveau du modèle **MVC**, cet objet est également une **Vue** (avec un mini-contrôleur). Cette vue est également reliée au contrôleur **employeBindingSource**. Observez la propriété *BindingSource* de cet objet.

BindingSource

employeBindingSource

- Maintenant, à partir de *la Source de données*, faites glisser le champ **empNom** situé dans la *dataTable employe* sur le formulaire.



Observez la propriété *DataBindings* du *TextBox* (qui est une **Vue**). Cette vue est également reliée au contrôleur **employeBindingSource**.

(DataBindings)	
(Avancées)	
Tag	(aucun)
Text	employeBindingSource - empNom

- Ici, nous avons un modèle, trois vues et un contrôleur. Le contrôleur fait le lien entre les trois vues et le modèle.
- Exécutez l'application.

- A l'aide de la barre de navigation, passez à l'enregistrement suivant (no 2). Observez, dans le *DataGridView* que l'employé 1002 est sélectionné et que, dans le *TextBox*, le nom de *Bedard* est affiché.



Ici, la barre de navigation, via un événement, a averti le contrôleur de sélectionner le deuxième enregistrement. Le contrôleur, à son tour, a demandé au *DataGridView* et au *TextBox* de sélectionner le deuxième enregistrement de la *DataTable employe*.

- Dans le *DataGridView*, sélectionnez l'employé 1007. Observez, dans la barre de navigation que l'enregistrement no 7 est affiché et que, dans le *TextBox*, le nom de *Lavoie* est affiché.

Ici, la *DataGridView*, via un événement, a averti le contrôleur de sélectionner le septième enregistrement. Le contrôleur, à son tour, a demandé, à la barre de navigation et au *TextBox*, de sélectionner le septième enregistrement de la *DataTable employe*.

- Dans le *TextBox*, remplacez *Lavoie* par *Larue* puis cliquez n'importe où dans le *DataGridView*. Observez, dans le *DataGridView*, que l'employé 1007 se nomme *Larue* au lieu de *Lavoie*.

Ici, le *TextBox*, via un événement, a averti le contrôleur de modifier le nom de *Lavoie* par *Larue*. Le contrôleur, à son tour, a modifié le modèle (la *DataTable* **Employe** du *DataSet*). Suite à cette modification, le *DataGridView* a été mis à jour.

Dans le modèle **MVC**, il existe un lien entre le contrôleur et la vue ainsi qu'un lien entre le contrôleur et le modèle mais il existe également un lien entre chacune des données de la vue et la donnée correspondante dans le modèle (via des pointeurs). Ce lien passe également par le contrôleur. Par exemple, le contenu du *TextBox* est lié au champ **empNom** dans la *DataTable* **employe**. Chacune des colonnes du *DataGridView* est liée au champ correspondant dans la *DataTable* **employe**. Dans le modèle **MVC**, la vue est un miroir du modèle.

- Pour illustrer cela, ajoutez un bouton de commande nommé **btnModifierNom** et un *TextBox* **txtNom** sur le formulaire.

- Programmez l'événement *Click* du bouton de commande de la manière suivante (ne tapez pas **bdVoyagesDataSet** mais **bdVoyagesVotreNomDataSet**) :

```
private void btnModifierNom_Click(object sender, EventArgs e)
{
    bdVoyagesDataSet.employe[0].empNom = txtNom.Text;
}
```

Ici, à chaque fois que l'utilisateur va cliquer sur le bouton de commande, le nom de 1<sup>er</sup> employé (l'employé #0) va être modifié dans la *DataTable* **employe** et, par lien, dans chacune des vues qui contient ce nom.

- Exécutez l'application.
  - Dans le *TextBox* **txtNom**, tapez *Tremblay* puis cliquez sur le bouton de commande **btnModifierNom**.

- Observez, dans le *DataGridView*, que l'employé 1001 se nomme *Tremblay* au lieu de *Lebeau*.

empNo	empPrenom	empNom
1001	toto	Tremblay

- Observez également que, dans le *TextBox*, cet employé se nomme également *Tremblay*.

## 2.2 Le contrôleur

Dans le modèle *MVC*, le contrôleur est un objet très important car c'est lui qui fait le lien entre le modèle et la vue. Dans cette section, nous allons utiliser quelques méthodes du contrôleur.

- Sur le formulaire, ajoutez deux boutons de commande **btnSuspendreLiaison** et **btnReprendreLiaison**



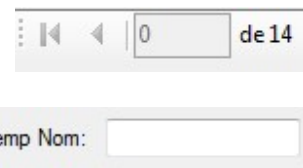
- Programmez l'événement *Click* de chacun des boutons de commandes de la manière suivante.

```
private void btnSuspendreLiaison_Click(object sender, EventArgs e)
{
    employeBindingSource.SuspendBinding();
    this.btnSuspendreLiaison.Enabled = false;
    this.btnReprendreLiaison.Enabled = true;
}

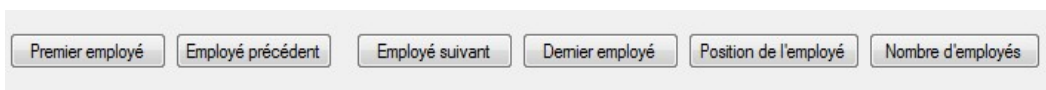
private void btnReprendreLiaison_Click(object sender, EventArgs e)
{
    employeBindingSource.ResumeBinding();
    this.btnReprendreLiaison.Enabled = false;
    this.btnSuspendreLiaison.Enabled = true;
}
```

- Exécutez l'application.

- Cliquez sur le bouton *Suspendre la liaison*. Comme vous pouvez le constater, les vues ne sont plus reliées au contrôleur. Si vous sélectionnez un enregistrement dans le *DataGridView*, le numéro d'enregistrement est toujours 0 dans la barre de navigation et le *TextBox emp Nom* ne contient aucun nom.



- Cliquez sur le bouton *Reprendre la liaison*. Observez le comportement.
- Sur le formulaire, ajoutez six boutons de commande dont les noms sont respectivement **btnPremierEmploye**, **btnEmployePrecedent**, **btnEmployeSuivant**, **btnDernierEmploye**, **btnPositionEmploye** et **btnNbEmployes**.



- Programmez l'événement *Click* de chacun des boutons de commandes de la manière suivante puis exécutez l'application.

```

private void btnPremierEmploye_Click(object sender, EventArgs e)
{
    employeBindingSource.MoveFirst();
}

private void btnEmployePrecedent_Click(object sender, EventArgs e)
{
    employeBindingSource.MovePrevious();
}

private void btnEmployeSuivant_Click(object sender, EventArgs e)
{
    employeBindingSource.MoveNext();
}

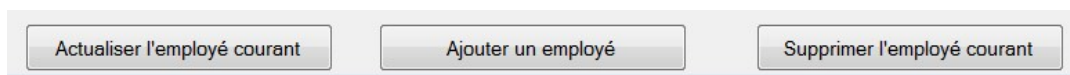
private void btnDernierEmploye_Click(object sender, EventArgs e)
{
    employeBindingSource.MoveLast();
}

private void btnPositionEmploye_Click(object sender, EventArgs e)
{
    MessageBox.Show( "Position de l'employé: " + employeBindingSource.Position);
}

private void btnNbEmployes_Click(object sender, EventArgs e)
{
    MessageBox.Show("Nombre d'employés: " + employeBindingSource.Count);
}

```

- À l'exécution, cliquez sur chacun des boutons de commande pour voir son effet.
  - Noter que la navigation ne fonctionne pas si les vues ne sont plus reliées au contrôleur (si la liaison des données est suspendue). Testez.
- Sur le formulaire, ajoutez trois boutons de commande dont les noms sont respectivement **btnActualiser**, **btnAjouter** et **btnSupprimer**.



- Programmez l'événement *Click* de chacun des boutons de commande comme suit puis exécutez l'application.



```

private void btnActualiser_Click(object sender, EventArgs e)
{
    employeBindingSource.ResetCurrentItem();
}

private void btnAjouter_Click(object sender, EventArgs e)
{
    employeBindingSource.AddNew();
}

private void btnSupprimer_Click(object sender, EventArgs e)
{
    employeBindingSource.RemoveCurrent();
}

```

- À l'exécution, tapez *Tremblay* dans la zone de texte puis cliquez sur le bouton **Actualiser l'employé courant**. Observez, que le nom de l'employé s'est automatiquement mis à jour dans le *DataGridView*.

emp Nom:

empNo	empPrenom	empNom
1001	toto	Tremblay

Il est important de noter que le nom de l'employé a été modifié dans la *DataTable employe* du *DataSet* (dans le modèle) mais n'a pas été modifié dans la vraie base de données.

- À l'exécution, cliquez sur le bouton **Ajouter un employé**. Observez l'apparition d'un enregistrement vide à la fin du *DataGridView*.

	empNo	empPrenom	empNom	empEmploi	empSal	empComm	empSup	empBureau
	1009	Marcel	Lebeau	SC	12,75		1011	bpt
	1010	Paul	Côté	AN	145,00		1002	stf
	1011	Angélique	Lavoie	GR	18,25		1001	lév
	1012	Angèle	Vitiro	AN	125,00		1002	lév
	1013	Virgule	DeGuillemets	AN	115,00		1002	stf
	1014	Ti	Bout	AN	115,00		1002	lév

Vous pouvez ainsi remplir tous les champs pour ajouter un nouvel employé

	empNo	empPrenom	empNom	empEmploi	empSal	empComm	empSup	empBureau
	1009	Marcel	Lebeau	SC	12,75		1011	bpt
	1010	Paul	Côté	AN	145,00		1002	stf
	1011	Angélique	Lavoie	GR	18,25		1001	lév
	1012	Angèle	Vitiro	AN	125,00		1002	lév
	1013	Virgule	DeGuillemets	AN	115,00		1002	stf
	1014	Ti	Bout	AN	115,00		1002	lév
	9999	Jean	Tremblay	GR	125000		1002	lév

Il est important de noter que ce nouvel enregistrement est ajouté dans *DataTable employe* du *DataSet* (dans le modèle) mais pas dans la vraie base de données. Pour le vérifier, fermer le formulaire **frmMVC** puis le recharger.

- À l'exécution, sélectionnez le deuxième enregistrement dans le *DataGridView* puis cliquez sur le bouton **Supprimer l'employé courant**. Observez que l'employé 1002 n'existe plus.

	empNo	empPrenom	empNom
	1001	toto	Tremblay
▶	1003	Jean	Sergio

Il est important de prendre note que l'employé a été supprimé dans la *DataTable employe* du *DataSet* (dans le modèle) mais n'a pas été supprimé dans la vraie base de données. Vérifiez.

- Il est évident que la modification du nom ainsi que la suppression de l'employé ne fonctionnent pas si les vues ne sont plus reliées au contrôleur (liaison des données suspendue). Testez cela.

Suspendre la liaison

- Testez l'ajout dans le cas où liaison des données est suspendue. Quelles sont vos observations?

### 3. Chargement dynamique de données

On veut concevoir une application qui charge les employés dans un *DataSet*. Les contrats des employés seront chargés de manière dynamique à l'aide du contrôleur lors de la sélection d'un employé.

#### Étapes :

- Ajoutez un nouveau formulaire. Le formulaire doit porter le nom de **frmChargementDynamique** et son titre doit être *Chargement dynamique de données*.
- Rattachez ce formulaire à **frmMenu**.



- A partir de la fenêtre *Source de données*, faites afficher, sur le formulaire, dans cet ordre, tous les employés et tous les contrats sous forme tabulaire (dans des *DataGridView*). Pour les employés, n'affichez que les colonnes *empNo*, *empPrenom* et *empNom*. Pour les contrats, n'affichez que les colonnes *conNo*, *conMontant* et *empNo* (supprimez les autres colonnes).

**Attention :** N'utilisez pas la *DataTable contrat* qui est située dans la *DataTable employe*.

- Notez que tous les employés sont affichés. Pour afficher les contrats d'un employé, un filtre de recherche a été ajouté automatiquement. Si vous entrez un numéro d'employé dans la zone de texte *empNo*, puis cliquez sur *Fill*, les contrats de l'employé en question sont affichés. Testez, puis supprimez la zone de texte du filtre et le bouton *Fill* ainsi que la méthode qui gère cet événement (méthode private void fillToolStripButton\_click()....)
- Pour afficher, les contrats de l'employé sélectionné, la *DataTable contrat* doit être remplie dynamiquement en modifiant la requête SQL du *contratTableAdapter* comme suit :

- Cliquez-droit sur le *DataTable contrat*, puis sélectionnez **Configurer...** dans le menu contextuel. Tapez l'instruction suivante :

```
SELECT conNo, conDate, conAcompte, conMontant, ConPaye, conTypeOcc, empNo, voyNo, cliNo  
FROM contrat  
WHERE empNo = @empNo
```

Cela va nous permettre d'obtenir seulement les contrats de l'employé dont le numéro sera passé dans un paramètre *SQL Serveur*.

- Sélectionnez le contrôleur **employeBindingSource**, double-cliquez sur l'événement *PositionChanged*.  
Cet événement est appelé à chaque fois qu'il y a un changement de position au niveau d'une vue quelconque. C'est le cas lorsque l'utilisateur utilise la barre de navigation pour se déplacer d'un enregistrement à un autre ou qu'il sélectionne un enregistrement dans un *DataGridView*.  
Programmez cet événement comme suit :

```

private void employeBindingSource_PositionChanged(object sender, EventArgs e)
{
    {
        int noEnregistrement = employeBindingSource.Position;
        int nbEnregistrements = employeBindingSource.Count;
        if (noEnregistrement >= 0 && noEnregistrement < nbEnregistrements)
        {
            dynamic enregistrementSelectionne = employeBindingSource.Current;
            decimal noEmployeSelectionne = enregistrementSelectionne["empNo"];
            contratTableAdapter.ClearBeforeFill = true;
            contratTableAdapter.Fill(bDVoyagesDataSet.contrat, noEmployeSelectionne);
        }
    }
}

```

Notez l'utilisation du type `dynamic` pour la variable `enregistrementSelectionne`. Son type sera déterminé à l'exécution. Il dépend de la vue utilisée. Notez également la présence du deuxième paramètre de la méthode `Fill`. Le numéro de l'employé sélectionné correspond au paramètre `@empNo` de la requête SQL.

- Exécutez l'application.
- Sélectionnez n'importe quel employé et observez l'affichage de tous ses contrats dans le *DataGridView* des contrats.
- Testez l'application en sélectionnant la dernière ligne vide de la table *employe*. Le programme plante. Identifiez la source du problème et proposez une solution.