TP .Net Maui n° 2bis (Suite) : Application à notes multiples

Partie 4 : Fonctionnalités multi-notes

- Affichage des notes :

Dans « Note.xaml.cs », changez le nom de la note en « `"123.notes.txt"` » (en variable globale et constructeur), puis exécutez votre application pour enregistrer cette note.

Dans « Views » ajoutez une nouvelle page « .NET MAUI ContentPage (XAML) » que vous nommez « ListeNotes.xaml ».

Dans « Models » ajoutez une nouvelle classe (Code / Classe C#) que vous nommerez « CListeNotes ». Complétez cette classe da la façon suivante :
- ajoutez une collection publique de notes de type « `ObservableCollection` » nommée « `CollNotes` » avec Getter/Setter et constructeur

```
public ObservableCollection<CNote> CollNotes { get; set; } = new ObservableCollection<CNote>();
```

- ajoutez une méthode publique « `ChargerNotes()` » avec pour code :
  - la déclaration d'une variable « `string` » valant « `FileSystem.AppDataDirectory;` »
  - la déclaration d'une collection « `IEnumerable<CNote>` » nommée « `listenotes` » et valant :

```
Directory.EnumerateFiles(appDataPath, "*.notes.txt")
  .Select(filename => new CNote()
  {
      Filename = filename,
      Text = File.ReadAllText(filename),
      Date = File.GetCreationTime(filename)
  })
  .OrderBy(note => note.Date);
```

  Que va faire ce code ? :

  _____

  _____

  - avec une boucle « `foreach` » listez chaque « `CNote` » présente dans « `listenotes` » et ajoutez là à la collection « `CollNotes` »
- ajoutez un constructeur à la classe qui ne fera qu'appeler « `ChargerNotes` »

```
public CListeNotes() => ChargerNotes();
```

Dans « ListeNotes.xaml », créez une « `CollectionView` » pour afficher la collection d'éléments « `CollNotes` » :

```
<CollectionView x:Name="CollectionDeNotes"
                ItemsSource="{Binding CollNotes}"
                Margin="20"
                SelectionMode="Single">
</CollectionView>
```

Puis définissez le modèle d'affichage des Items :

```
<CollectionView.ItemsLayout>
    <LinearItemsLayout Orientation="Vertical" ItemSpacing="10" />
</CollectionView.ItemsLayout>
```

Puis définissez le contenu de l'affichage des Items :

```
<CollectionView.ItemTemplate>
    <DataTemplate>
        <StackLayout>
            <Label Text="{Binding Text}" FontSize="22"/>
            <Label Text="{Binding Date}" FontSize="14" TextColor="Silver"/>
        </StackLayout>
    </DataTemplate>
</CollectionView.ItemTemplate>
```

Enfin dans « ListeNotes.xaml.cs », ajoutez définissez le contexte dans le constructeur en ajoutant :

```
BindingContext = new Models.CListeNotes();
```

Il suffit maintenant de faire afficher la page « Liste Notes ». Pour cela modifiez le « ShellContent » de la barre de commande.

```
Title="Liste Notes"
ContentTemplate="{DataTemplate view:ListeNotes}"
```

Testez votre application et vérifiez que la page « Liste Notes » affiche bien votre dernière note.

- Ajout de notes :

Dans la page « Liste Notes », ajoutez un bouton « + » :

```
<ContentPage.ToolbarItems>
    <ToolbarItem Text="Ajouter" Clicked="Ajouter_Clicked" IconImageSource="{FontImage
                                                    Glyph='+', Color=White, Size=22}" />
</ContentPage.ToolbarItems>
```

Dans la méthode associée à votre bouton, ajoutez le code d'ouverture de la page Note :

```
await Shell.Current.GoToAsync(nameof(Note));
```

Pour que ce type de redirection fonctionne, il faut ajouter la route dans le construteur de la page qui va l'utiliser :

```
Routing.RegisterRoute($"{nameof(Note)}", typeof(Note));
```

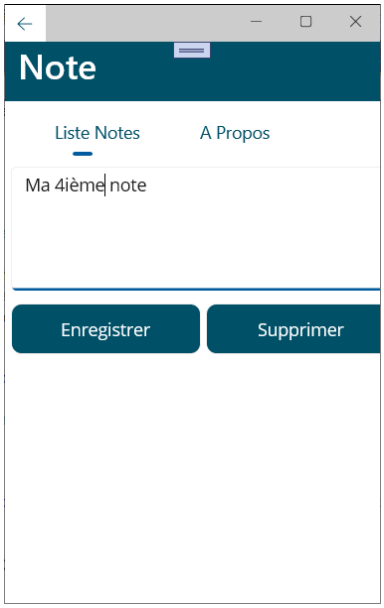
Maintenant, il faut modifier la page de note.
- En premier, il faut associer un nom différent au fichier de chaque note. Pour cela remplacer le nom du fichier de note par un nom aléatoire : `$"{Path.GetRandomFileName()}.notes.txt";`
- Puis, modifiez le code de « `ButtonEnregistrer_Clicked` » pour qu'il sauvegarde la note associée au contexte :

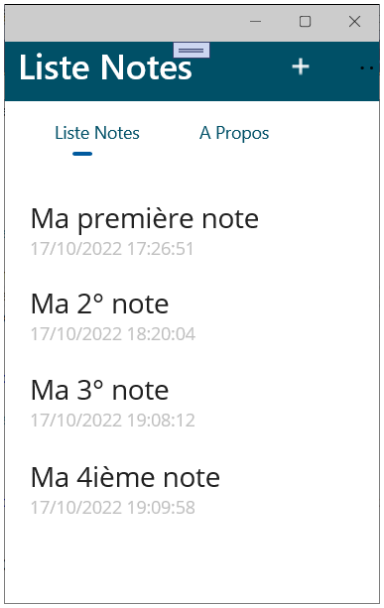
```
if (BindingContext is Models.CNote note)
    File.WriteAllText(note.Filename, TextEditor.Text);
```

Et qu'il revienne à la page précédente :

```
await Shell.Current.GoToAsync("..");
```

Testez l'ajout de notes et vérifiez qu'elles sont bien visibles dans la page « Liste Notes ».





- Modification d'une note

Dans la page « Liste Notes », interceptez l'événement « `SelectionChanged` » de « `CollectionDeNotes` » :

```
SelectionChanged="CollectionDeNotes_SelectionChanged"
```

Dans la variable « e » qui est passée en paramètre se trouve une collection d'éléments sélectionnés nommée « `CurrentSelection` » de type « object ». Récupérez dans une variable de type « `var` » la note sélectionnée en lisant le premier élément sélectionné de cette collection (indice 0) et en le castant en « `Models.CNote` ».
Encadrez ce code d'une condition « if » qui vérifie que le nombre d'éléments de cette collection n'est pas zéro (Count).

Maintenant, comment faire pour faire passer la note sélectionnée à la page « Note » pour la supprimer ou la modifier ? Vous pourriez utiliser une variable globale mais en MVC ce n'est pas très conforme.

Pour cela vous allez faire passer l'information comme dans le cas d'une navigation Web, c'est-à-dire en ajoutant des paramètres lors de l'appel de la page « Note ».
Il va donc falloir construire une chaîne type requête qui va inclure le nom de la page de destination et un paramètre définissant une propriété de cette page.
En premier dans « Note.xaml.cs » ajoutez un paramètre de requête sous la forme (juste sous le namespace) :

```
[QueryProperty(nameof(ItemId), nameof(ItemId))]
```

Et ajoutez la propriété « `ItemId` » à la classe qui chargera la note à son initialisation (setter) :

```
public string ItemId
{
    set { ChargerNote(value); }
}
```

Si vous regardez la déclaration de « `ChargerNote` », vous devinez que « `ItemId` » devra contenir le chemin du fichier contenant la note qui a été sélectionnée.

Dans la page « Liste Notes », il faut donc faire passer le paramètre lors de l'appel de la page « Note ». Pour cela ajoutez après l'initialisation de la variable « note » :

```
await Shell.Current.GoToAsync($"{nameof(Note)}?{nameof(Note.ItemId)}={note.Filename}");
```

Remarquez que c'est toujours « `GoToAsync` » qui est appelée, que l'argument passé correspond au nom de la page suivi du symbole « ? » et de « `ItemId = nom du fichier` » (codé en expression Maui)

Pour finir, mettez « `CollectionDeNotes.SelectedItem` » à nul pour éviter de passer plusieurs fois dans la méthode.

Testez l'application pour valider la modification de notes, tout en gardant opérationnel l'ajout de notes.

- Suppression de notes :

La suppression de note va se faire sur cette même page « Note » avec le bouton existant. Il suffit de modifier le code de la même façon que pour « `ButtonEnregistrer_Clicked` » :
- Testez dans un « if » si la note associée au contexte de données est bien une CNote en l'intialisant à « note »
- Si le fichier « `note.Filename` » existe, supprimez-le
- Revenez à la page précédente

Testez l'application pour valider la suppression de notes.