

Développement .Net Maui

(3iL I3 - B. Chervy) **TP Maui N° 2** Année: 2024/2025 Temps: 2 séances Nombre de page: 5

TP .Net Maui n° 2 : Gestion de notes sous .Net Maui

Objectif: Prise en main

Partie 1 : Création du projet

Lancez « Visual Studio » et créez un nouveau projet de type « Application .NET MAUI ».

Lancez l'exécution en sélectionnant « Windows Machine ». Testez l'application.

Allez sous « Windows Machine » et faites « Gestionnaire d'appareils Android », faites « Nouveauté », sélectionnez « Pixel 2 (+Store), x86, Android 9 - API28 » et laissez les autres paramètres par défaut, faites ensuite « Créer ».

Démarrez votre émulateur.

Revenez à Visual Studio et lancez votre application sur votre émulateur.

Partie 2 : Création des premières pages

- Création de la page A Propos

Dans l'Explorateur de solutions, cliquez avec le bouton droit sur le projet et faites « Ajouter un nouvel élément ». Dans la fenêtre, sélectionnez .NET MAUI et .NET MAUI ContentPage (XAML). Nommez votre page « APropos.xaml ». Changez le code de la façon suivante :

Avec le bouton droit cliquez sur le répertoire « Resources/Images » et faites « Ajoutez/Elément Existant », et ajoutez le fichier « logo3il.png ».

Sur le nom de la méthode « APropos_Clicked », cliquez avec le bouton droit pour faire « Atteindre la définition ». Dans la méthode créée, ajoutez le code :

```
await Launcher.Default.OpenAsync("https://www.3il-ingenieurs.fr/");
```

Attention la méthode devra être passée en « async »

```
Ajoutez les 2 images « icon_about.png » et « icon_notes.png » au répertoire « Images ». Ouvrez « AppShell.xaml » et remplacez la balise « ShellContent » par le code par : 

<TabBar>
```

```
<ShellContent</pre>
```

Notez que <Shell> est l'objet racine du balisage XAML, <TabBar> est une barre de commande avec 2 éléments de type <ShellContent>

Lancez votre application est vérifiez que la page APropos s'ouvre et que le bouton « En savoir plus sur 3iL » ouvre bien le navigateur.

Vous devez avoir le rendu suivant :





- Création de la page Notes

Ajoutez une nouvelle page « .NET MAUI ContentPage (XAML) » que vous nommez « Note.xaml ».

```
Changez le code de la façon suivante :
    <VerticalStackLayout Spacing="10" Margin="5">
        <Editor x:Name="TextEditor"
                 Placeholder="Taper votre note"
                HeightRequest="100" />
        <Grid ColumnDefinitions="*,*" ColumnSpacing="4">
            <Button Text="Enregistrer"</pre>
                     Clicked="ButtonEnregistrer_Clicked" />
            <Button Grid.Column="1"</pre>
                     Text="Supprimer"
                     Clicked="ButtonSupprimer_Clicked" />
        </Grid>
    </VerticalStackLayout>
Construisez les 2 méthodes des 2 boutons. Dans la classe de votre fenêtre ajoutez la variable globale :
      string _fileName = Path.Combine(FileSystem.AppDataDirectory, "notes.txt");
Dans le constructeur, ajoutez :
      if (File.Exists(_fileName))
            TextEditor.Text = File.ReadAllText(_fileName);
Enfin complétez les 2 méthodes des boutons comme cela :
      private void SaveButton_Clicked(object sender, EventArgs e)
               File.WriteAllText(_fileName, TextEditor.Text);
      private void DeleteButton_Clicked(object sender, EventArgs e)
```

if (File.Exists(_fileName))

```
File.Delete(_fileName);
TextEditor.Text = string.Empty;
}
```

La page de note étant terminée, il faut la rendre accessible à l'utilisateur. Dans le fichier « AppShell.xaml », modifiez le premier élément en spécifiant : ContentTemplate="{DataTemplate local:Note}"

Testez votre application, saisissez une note, enregistrez là, arrêtez et redémarrez l'application pour vérifier que la note est bien sauvegardée.

Note

Note

Notes A Propos

Test de note|

Partie 3 : Fonctionnalités MVC

Vous allez réorganiser (Refactoriser) votre projet pour passer en mode MVC.

Vous allez créer d'abord créer 2 répertoires dans votre projet : Models et Views.

Ensuite vous allez déplacer les 2 pages APropos.xaml et Note.xaml dans ce répertoire View. Si Visual vous demande d'ajuster les espaces de noms, répondez NON.

Votre application risque de ne plus fonctionner normalement. Pour corriger cela :

- dans Note.xaml changez «x:Class="TPMaui1.Note" » en «x:Class="TPMaui1.Views.Note" »
- dans APropos.xaml changez «x:Class="TPMaui1.APropos" » en «x:Class="TPMaui1.Views.APropos" »
- dans Note.xaml.cs changez « namespace TPMaui1; » par « namespace TPMaui1.Views; »
- dans APropos.xaml.cs changez « namespace TPMaui1; » par « namespace TPMaui1.Views; »
- dans AppShell.xaml changez « xmlns:local="clr-namespace:TPMaui1" » par

```
« xmlns:view="clr-namespace:TPMaui1.Views" »
et « {DataTemplate local:Note} » par « {DataTemplate
```

et « {DataTemplate local:Note} » par « {DataTemplate view:Note} »
et « {DataTemplate local:APropos} » par « {DataTemplate view:APropos} »

(Attention ne modifier pas votre espace de nom principal qui est dans cet exemple « TPMaui1 »)

Testez votre application pour vérifier que les modifications sont correctes.

En cliquant sur le répertoire « Models », faites « Ajouter un nouvel élément » et sélectionnez « Code / Classe » que vous nommerez « CNote ». Complétez cette classe avec le code suivant :

```
public string Filename { get; set; }
public string Text { get; set; }
public DateTime Date { get; set; }
```

Ajoutez ensuite une 2° classe nommée « CAPropos ». Complétez cette classe avec le code suivant :

```
public string Title => "Notes 3iL - " + AppInfo.Name;
public string Version => AppInfo.VersionString;
public string MoreInfoUrl => "https://www.3il-ingenieurs.fr/";
public string Message => "Application XAML/C# écrite en .NET MAUI.";
```

```
Dans « APropos.xaml », faites les modifications suivantes :
- Au-dessus de « x:Class="TPMaui1.Views.APropos" » ajoutez
                                                « xmlns:models="clr-namespace:TPMaui1.Models" »
- En dessous de la balise « ContentPage » (après « Title="APropos"> ») ajoutez ;
    <ContentPage.BindingContext>
        <models:CAPropos />
    </ContentPage.BindingContext>
- Remplacez les codes des 3 « Label » par :
        <Label FontSize="22" FontAttributes="Bold" Text="{Binding Title}" VerticalOptions="End"</pre>
                    TextColor="{StaticResource Secondary}"/>
        <Label FontSize="22" Text="{Binding Version}" VerticalOptions="End"</pre>
                    TextColor="{StaticResource Secondary}"/>
        <Label Text="{Binding Message}" TextColor="{StaticResource Primary}"/>
Enfin dans la méthode « APropos_Clicked », remplacez le code par :
        if (BindingContext is Models.CAPropos apropos)
            await Launcher.Default.OpenAsync(apropos.MoreInfoUrl);
        }
Testez votre application, et vérifier que la page « A Propos » est toujours fonctionnelle.
A quoi sert l'instruction « xmlns:models="clr-namespace:TPMaui1.Models" » ?:
A quoi sert le bloc d'instructions « BindingContext » ? :
A quoi sert l'instruction « {Binding variable} » ?:
Pour la page Note, dans « Note.xaml », ajoutez le code « Text="{Binding Text}" » au contrôle « Editor ».
Dans « « Note.xaml.cs », ajoutez la méthode :
    private void ChargerNote(string fileName)
        Models.CNote noteModel = new Models.CNote();
        noteModel.Filename = fileName;
        if (File.Exists(fileName))
             noteModel.Date = File.GetCreationTime(fileName);
             noteModel.Text = File.ReadAllText(fileName);
        BindingContext = noteModel;
    }
Puis dans le constructeur de cette page, ajoutez :
        string appDataPath = FileSystem.AppDataDirectory;
        string sFileName = "notes.txt";
        ChargerNote(Path.Combine(appDataPath, sFileName));
```

Testez votre application, et vérifiez que la page « Note » est toujours fonctionnelle.

