Anatomie d'une Application MAUI (.NET Multi-platform App UI) 🦴

Introduction

MAUI (Multi-platform App UI) est un framework Microsoft permettant de développer des applications multiplateformes (Windows, Android, iOS, macOS) à partir d'une base de code unique. Ce document présente la structure fondamentale d'une application MAUI et explique les rôles de ses principaux composants.

Structure de base d'une application MAUI

Une application MAUI est organisée selon une architecture hiérarchique bien définie. Voici les composants principaux dans leur ordre d'initialisation :

1. MauiProgram.cs - Le point d'entrée

`MauiProgram.cs` est le point d'entrée de l'application. Il contient la classe `MauiProgram` avec la méthode statique `CreateMauiApp()` qui configure l'application :

```
public static class MauiProgram
{
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder
            .UseMauiApp<App>()
            .ConfigureFonts(fonts =>
            {
                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
                fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
            });

        #if DEBUG
            builder.Logging.AddDebug();
        #endif

        return builder.Build();
    }
}
```

Ce fichier est technique et généralement, il n'est pas nécessaire de le modifier sauf pour ajouter des configurations spécifiques comme des services ou des polices de caractères.

2. App.xaml & App.xaml.cs - La définition de l'application

Ces fichiers définissent l'application elle-même :

- **App.xaml** : Contient les ressources globales de l'application (styles, couleurs, templates)
- **App.xaml.cs** : Contient le code d'initialisation de l'application

La classe `App` est définie comme `partial` et hérite de `Application`. Elle est créée par la combinaison de ces deux fichiers pendant la compilation :

```
public partial class App : Application
{
    public App()
    {
        InitializeComponent();

        MainPage = new AppShell();
    }
}
```

Le concept de classes partielles (mot-clé "partial")

MAUI utilise le concept de classes partielles (`partial class`) qui permet de répartir la définition d'une classe entre plusieurs fichiers :

- Le fichier `.xaml` contient la partie UI déclarative
- Le fichier `.xaml.cs` contient la logique de code-behind

Ces deux fichiers sont combinés lors de la compilation pour former une classe unique.

3. AppShell.xaml & AppShell.xaml.cs - La structure de navigation

Le Shell représente la structure de navigation principale de l'application :

- **AppShell.xaml** : Définit la structure de navigation (tabs, flyout menu, routes)
- **AppShell.xaml.cs** : Contient la logique de navigation

```
<Shell
        x:Class="MauiAppBasicDefault.AppShell"
        xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:local="clr-namespace:MauiAppBasicDefault"
        Shell.FlyoutBehavior="Disabled"
        Title="MauiAppBasicDefault">

    <ShellContent
            Title="Home"
            ContentTemplate="{DataTemplate local:MainPage}"
            Route="MainPage"/>
</Shell>
```

AppShell permet de définir:

- Une navigation par onglets (TabBar)
- Un menu latéral (Flyout)
- Des routes de navigation pour la navigation entre pages

4. MainPage.xaml & MainPage.xaml.cs - La page principale

Ces fichiers définissent la page principale de l'application et son comportement :

- **MainPage.xaml** : Définit l'interface utilisateur de la page
- **MainPage.xaml.cs** : Contient la logique de comportement

Exemple de MainPage.xaml :

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MauiAppBasicDefault.MainPage">
    <VerticalStackLayout Spacing="25">
        <Image Source="dotnet_bot.png"
               HeightRequest="185"
               SemanticProperties.Description="dot net bot in a race car number
eight"/>

        <Label Text="Hello, World!"
               Style="{StaticResource Headline}"
               SemanticProperties.HeadingLevel="Level1"/>

        <Label Text="Welcome to .NET Multi-platform App UI"
               Style="{StaticResource SubHeadline}"
               SemanticProperties.HeadingLevel="Level2"
               SemanticProperties.Description="Welcome to dot net Multi platform App
U I"/>

        <Button x:Name="CounterBtn"
                Text="Click me"
                SemanticProperties.Hint="Counts the number of times you click"
                Clicked="OnCounterClicked"/>
    </VerticalStackLayout>
</ContentPage>
```

Exemple de MainPage.xaml.cs :

```
public partial class MainPage : ContentPage
{
    private int count = 0;

    public MainPage()
    {
        InitializeComponent();
    }

    private void OnCounterClicked(object sender, EventArgs e)
    {
        count++;

        if (count == 1)
            CounterBtn.Text = $"Clicked {count} time";
        else
```

```
            CounterBtn.Text = $"Clicked {count} times";

        SemanticScreenReader.Announce(CounterBtn.Text);
    }
}
```

▶ C’est quoi SemanticScreenReader ?

Autres fichiers importants

Dossiers de projet

- **Platforms/** : Contient le code spécifique à chaque plateforme
- **Resources/** : Contient les ressources (images, polices, etc.)
- **Properties/** : Contient les paramètres du projet
- **Dependencies/** : Contient les références et packages utilisés

Flux d'exécution d'une application MAUI

1. L'application démarre via `MauiProgram.CreateMauiApp()`
2. L'instance de `App` est créée et initialisée
3. `App` crée l'instance de `AppShell` comme page principale
4. `AppShell` charge la page définie dans son `ShellContent` (généralement `MainPage`)
5. L'utilisateur interagit avec l'application via l'interface définie dans les pages

Remarques importantes

- **MAUI ≠ MVVM** : Par défaut, les templates MAUI n'implémentent pas le pattern MVVM (Model-View-ViewModel), bien que MAUI soit parfaitement compatible avec celui-ci (d’ailleurs c’est ce qui sera utilisé)
- **Fichiers générés** : Certains fichiers sont générés automatiquement (indiqués par des commentaires `<auto-generated>`) et ne doivent pas être modifiés manuellement
- **Construction de l'UI** : MAUI suggère d’utiliser XAML pour définir l'interface et C# pour la logique même si on peut tout faire en C#

Conclusion

Une application MAUI est structurée autour de plusieurs composants clés qui jouent chacun un rôle spécifique:

1. `MauiProgram.cs` - Configuration et bootstrap
2. `App.xaml/cs` - Définition et initialisation de l'application
3. `AppShell.xaml/cs` - Structure de navigation
4. Pages (MainPage, etc.) - Contenu et interaction utilisateur

Cette architecture permet une séparation claire entre la structure de l'application, sa navigation et le contenu spécifique à chaque page, tout en garantissant une expérience cohérente sur toutes les plateformes.