Xamarin Forms - Usando o Entity Framework Core 2.0

Neste artigo vamos usar os recursos do EF Core 2.0 para acessar dados em uma aplicação Xamarin Forms usando o Visual Studio 2017 e a linguagem C#.

Com o lançamento do Entity Framework Core 2.0, agora podemos executar o EF com um banco de dados SQLite no iOS, Android e na UWP.

Se você já conhece o EF Core vai perceber que é muito simples usar seus recursos com Xamarin Forms.

Como esse vai ser um artigo de apresentação eu vou ser criar um exemplo bem simples mostrando como usar o EF Core 2.0.

A seguir um roteiro básico do que precisa ser feito para isso:

1 - Instalar o .NET Core SDK 2.0 (ou superior);
2 - Definir a class library para usar o .NET Standard 2.0;
3 - Instalar o pacote Microsoft.EntityFrameworkCore.Sqlite em todos os projetos; *(Isso instala todas as dependências)*
4 - Definir o modelo de entidades;
5 - Definir a classe de contexto que herda de DbContext;
6 - Definir o provedor do banco de dados Sqlite e a string de conexão;

Você deve obter o caminho do banco de dados SQLite em cada plataforma. Segue abaixo a sintaxe usada para cada plataforma:

```
// Android
var dbPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Personal), "banco.db");

// iOS
var dbPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments), "..", "Library", "banco.db")

// UWP
var dbPath = Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path, "banco.db");
```

Além disso no iOS você deve incluir o código abaixo no arquivo **AppDelegate.cs :**

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options)
{
        SQLitePCL.Batteries.Init();

        global::Xamarin.Forms.Forms.Init();
        LoadApplication(new App());

        return base.FinishedLaunching(app, options);
}
```

Agora vamos ao projeto.

Recursos usados:

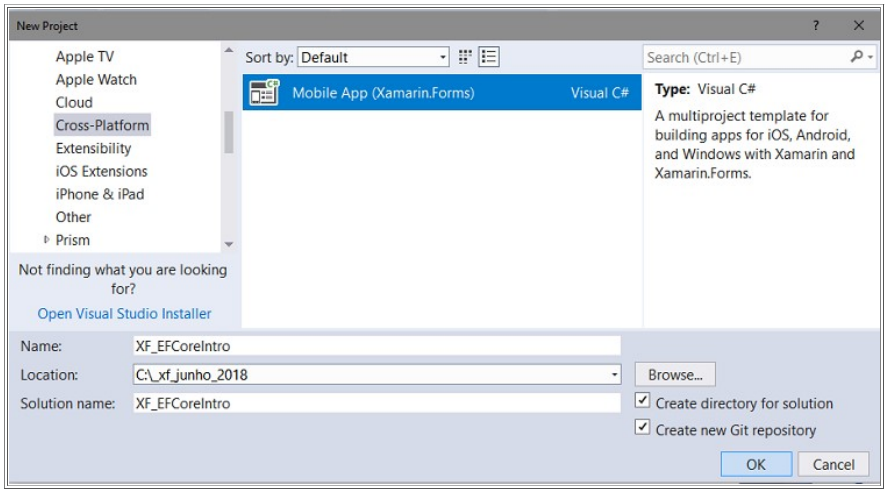
- Visual Studio Community 2017 ou Xamarin Studio
- Xamarin

Criando o projeto no Visual Studio 2017 Community e definindo a página principal

Abra o Visual Studio Community 2017 e clique em **New Project;**

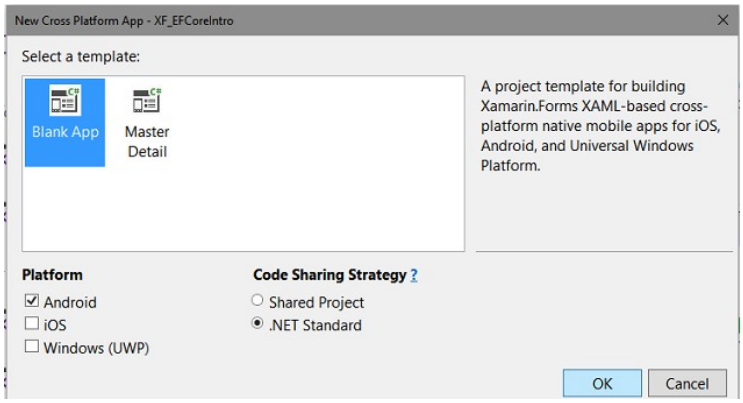
Selecione Visual C#, o template **Cross Plataform** e a seguir **Mobile App (Xamarin.Forms);**

Informe o nome **XF_EFCoreIntro** e clique no botão OK;



A seguir selecione **Blank App** e marque as opções - **Plataform -> Android (iOS e Window UWP)** e **.NET Standard** e clique em OK;

Nota: *Como eu não tenho um Mac marquei somente o projeto Android.*



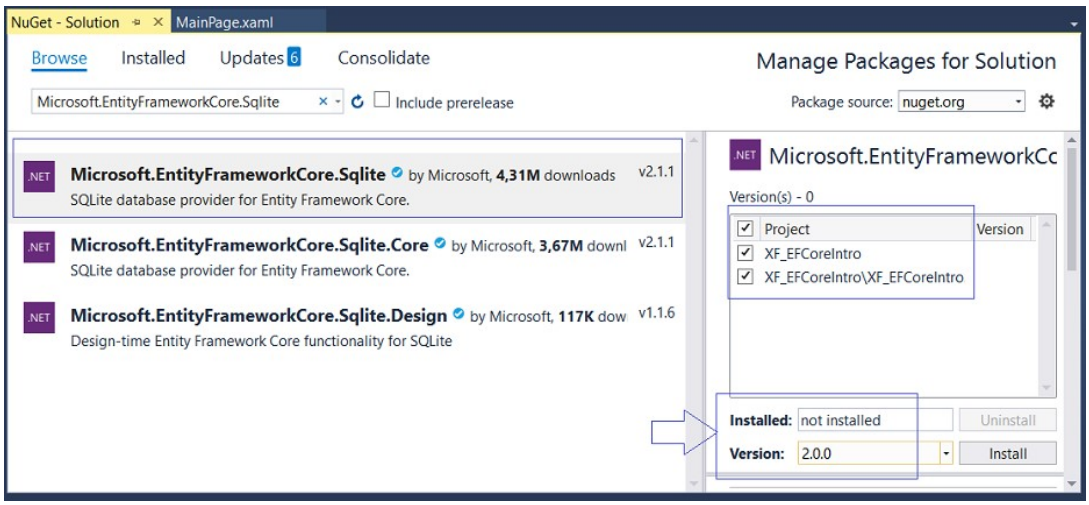
Será criado um projeto contendo no projeto **Portable** as páginas **App.xaml e MainPage.xaml.**

No code-behind do arquivo **App.xaml** temos a classe **App.xaml.cs** que irá conter o código compartilhado e que vamos usar neste artigo.

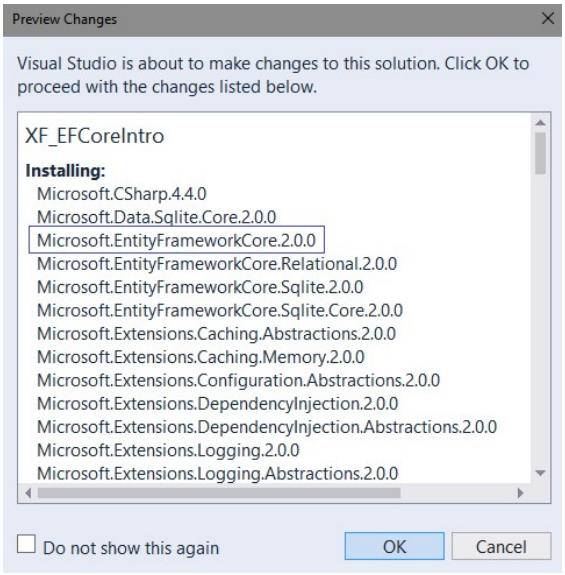
Incluindo pacote do EF Core no projeto

Agora vamos incluir uma referência ao pacote do Entity Frameworkcore para o Sqlite.

No menu **Tools** clique em **Nuget Package Manager -> Manage Nuget Packages for Solution** e selecione o pacote **Microsoft.EntityFrameworkCore.Sqlite**:



Observe que estou usando a versão 2.0.0 do pacote que vai instalar as dependências :



Selecione todos os projetos e clique no botão **Install**.

Definindo o modelo de domínio da aplicação : A classe Cliente

Vamos criar uma aplicação para gerenciar informações de clientes, logo vamos definir uma classe chamada **Cliente** que será o nosso modelo de domínio.

Crie uma pasta **Models** no projeto (**Project-> New Folder)** e a seguir crie a classe **Cliente.cs** conforme mostra o código abaixo:

```
public class Cliente
 {
       public int ClienteId { get; set; }
       public string Nome { get; set; }

       public override string ToString()
       {
          return Nome;
       }
}
```

A seguir vamos criar na pasta **Models** a classe de contexto: **AppDbContext** com o seguinte código:

```
using Microsoft.EntityFrameworkCore;

namespace XF_EFCoreIntro.Models
{
   public class AppDbContext : DbContext
   {
      public DbSet<Cliente> Clientes { get; set; }
      private string _databasePath;

      public AppDbContext(string databasePath)
      {
         _databasePath = databasePath;
      }

      protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
      {
         optionsBuilder.UseSqlite($"Filename={_databasePath}");
      }
   }
}
```

Na classe de contexto que herda de **DbContext** definimos o mapeamento da entidade **Cliente** para a tabela Clientes e no construtor estamos recebendo o caminho do banco de dados que iremos acessar.

No método **OnConfiguring**() definimos o provedor do banco de dados usado e definimos a string de conexão com o banco de dados.

Obtendo o caminho do banco de dados

Para obter o caminho do banco de dados no projeto Android abra o arquivo **MainActivity.cs** e inclua o código abaixo:

```
[Activity(Label = "XF_EFCoreIntro", Icon = "@mipmap/icon",
Theme = "@style/MainTheme", MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation)]
public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        TabLayoutResource = Resource.Layout.Tabbar;
        ToolbarResource = Resource.Layout.Toolbar;

        base.OnCreate(bundle);

        global::Xamarin.Forms.Forms.Init(this, bundle);

        var dbPath = Path.Combine(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "TesteDB.db");

        LoadApplication(new App(dbPath));
    }
}
```

Obtemos o caminho para o banco de dados **TesteDB.db** e passamos para o construtor da **App**. Se o banco de dados não existir será criado.

Definindo o código do arquivo App.xaml.cs

Agora vamos definir o código do arquivo **App.xaml.cs** conforme abaixo:

```
using System.Collections.Generic;
using System.Linq;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using XF_EFCoreIntro.Models;

[assembly: XamlCompilation(XamlCompilationOptions.Compile)]
namespace XF_EFCoreIntro
{
    public partial class App : Application
    {
        public App(string dbPath)
        {
            InitializeComponent();

            List<Cliente> listaClientes;

            // Cria o banco de dados e as tabelas
            using (var db = new AppDbContext(dbPath))
            {
                // Verifica se o banco de dados foi criado
                db.Database.EnsureCreated();

                // Inserindo dados : usando o método Add e SaveChanges
                db.Add(new Cliente() { Nome = "Maria da Silva" });
                db.Add(new Cliente() { Nome = "José Rocha Siqueira" });
                db.Add(new Cliente() { Nome = "Pedro dos Santos"});
                db.SaveChanges();

                // Retornando os dados
                listaClientes = db.Clientes.ToList();
            }

            //Cria e exibe o ListView com os dados
            MainPage = new MainPage()
            {
                Content = new ListView()
                {
                    ItemsSource = listaClientes
                }
            };
        }

        protected override void OnStart ()
        {
         // Handle when your app starts
        }
```

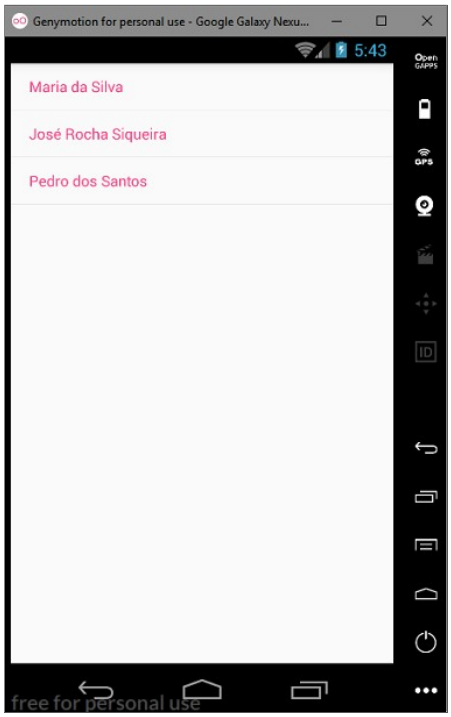
```
    protected override void OnSleep ()
    {
      // Handle when your app sleeps
    }

   protected override void OnResume ()
   {
     // Handle when your app resumes
   }
  }
}
```

Neste código estamos recebendo o caminho do banco de dados e criando uma instância do contexto informando o seu valor.

A seguir criamos 3 instâncias da entidade **Cliente** e usando o método **Add** do EF Core incluimos as entidades no contexto e a seguir, dando o **SaveChanges()** persistimos as informações no banco de dados.

Executando o projeto obteremos o seguinte resultado:



Observe que o controle ListView exibe os dados que foram incluidos na tabela do banco de dados pelo EF Core.

Em outro artigo vou mostrar como fazer um CRUD completo usando o EF Core 2.0 com Xamarin Forms.

Aguarde...

Pegue o código do projeto compartilhado aqui : **XF_EFCoreIntro.zip** *(sem as referências)*

'Disse-lhe Jesus: Não te hei dito que, se creres, verás a glória de Deus? '
João 11:40

- **Curso C# Basico - Video Aulas**

Quer aprender os conceitos da Programação Orientada a objetos ?

- **Curso Fundamentos da Programação Orientada a Objetos com VB .NET** NEW

Quer aprender o gerar relatórios com o ReportViewer no VS 2013 ?

- **Curso - Gerando Relatórios com o ReportViewer no VS 2013 - Vídeo Aulas** NEW

Referências:

- Seção VB .NET do Site Macoratti.net
- Super DVD .NET - A sua porta de entrada na plataforma .NET
- Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#
- Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#
- Seção C# do site Macoratti.net
- Seção ASP .NET do site Macoratti .net
- Curso Básico VB .NET - Vídeo Aulas
- Curso C# Básico - Vídeo Aulas
- Curso Fundamentos da Programação Orientada a Objetos com VB .NET NEW
- **Macoratti .net | Facebook**
- **macoratti - YouTube**
- **Jose C Macoratti (@macorati) | Twitter**
- VB.NET 2005 - Controles - Macoratti.net
- Seção de Jogos do site Macoratti .net
- Xamarim - Desenvolvimento Multiplataforma com C# ... - Macoratti.net
- Xamarin - Apresentando Xamarin.Forms - Macoratti.net
- Xamarin.Forms - Olá Mundo - Criando sua primeira ... - Macoratti.net
- Xamarin.Forms - Olá Mundo - Anatomia da aplicação - Macoratti.net
- https://developer.xamarin.com/api/type/Android.App.AlertDialog/
- Xamarin Android - Tratando eventos de forma declarativa
- Seção Mobile/Xamarin do site Macoratti .net
- https://developer.android.com/reference/android/app/Activity.html
- https://developer.xamarin.com/api/type/Android.Widget.ProgressBar/

José Carlos Macoratti