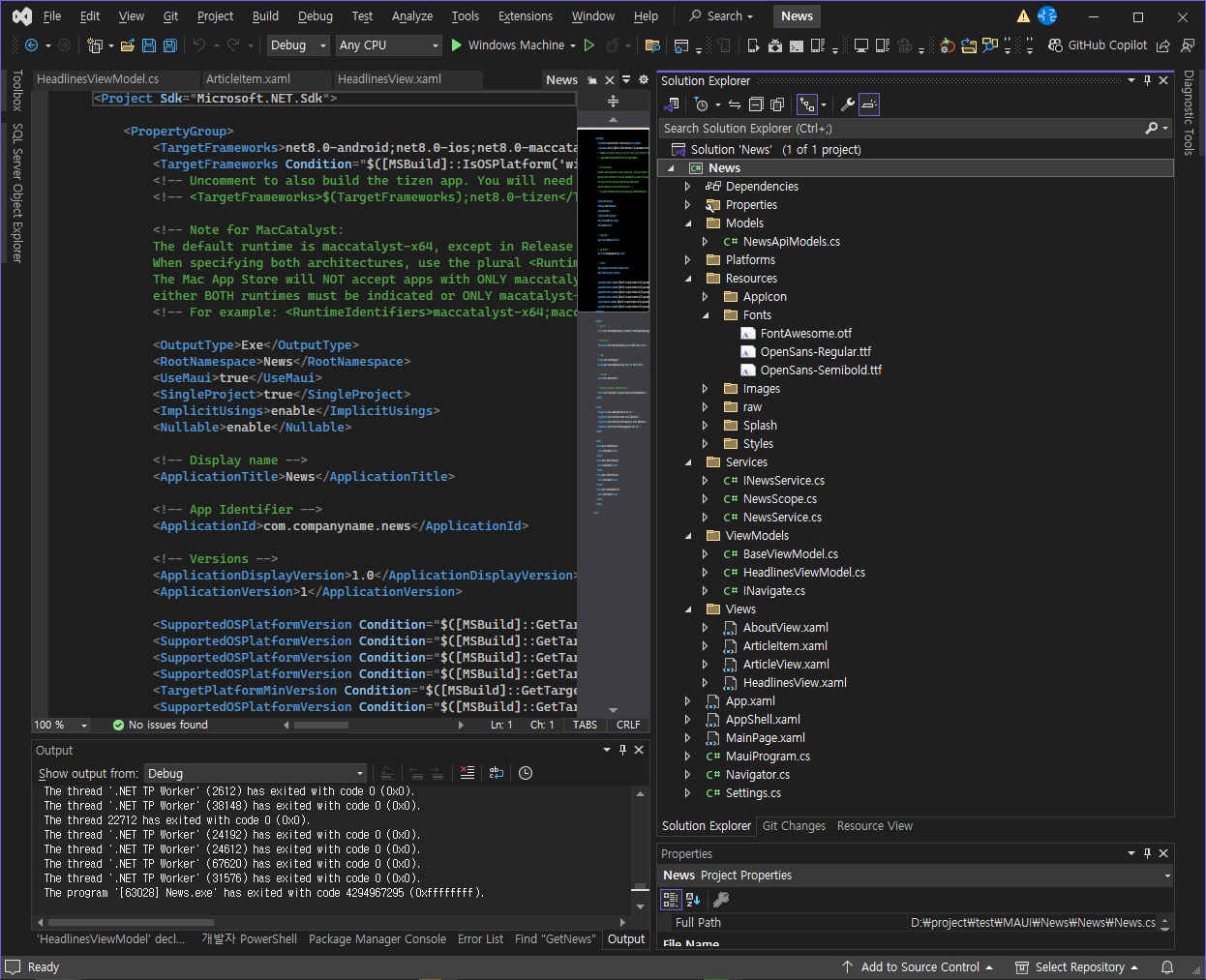
MAUI – NEWS

사전 준비

<https://newsapi.org/> 에 회원가입 한다.





App.xaml

|  |
| --- |
| <?xml version = "1.0" encoding = "UTF-8" ?>  <Application xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:News"  x:Class="News.App">  <Application.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="Resources/Styles/Colors.xaml" />  <ResourceDictionary Source="Resources/Styles/Styles.xaml" />  </ResourceDictionary.MergedDictionaries>  <FontImage x:Key="HomeIcon" Glyph="&#xf015;" Size="22" FontFamily="FontAwesome" Color="Black" />  <FontImage x:Key="HeadlinesIcon" Glyph="&#xf70e;" Size="22" FontFamily="FontAwesome" />  <FontImage x:Key="NewsIcon" Glyph="&#xf1ea;" Size="22" FontFamily="FontAwesome" />  <FontImage x:Key="SettingsIcon" Glyph="&#xf013;" Size="22" FontFamily="FontAwesome" Color="Black" />  <FontImage x:Key="AboutIcon" Glyph="&#xf05a;" Size="22" FontFamily="FontAwesome" Color="Black" />      </ResourceDictionary>  </Application.Resources>  </Application> |

App.xaml.cs

|  |
| --- |
| namespace News  {  public partial class App : Application  {  public App()  {  InitializeComponent();  MainPage = new AppShell();  }  #if WINDOWS  protected override Window CreateWindow(IActivationState activationState)  {  var window = base.CreateWindow(activationState);  const int newWidth = 540;  const int newHeight = 960;  window.Width = newWidth;  window.Height = newHeight;  return window;  }  #endif  }  } |

AppShell.xaml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <Shell  x:Class="News.AppShell"  xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:News"  xmlns:view="clr-namespace:News.Views">  <FlyoutItem Title="Home" Icon="{StaticResource HomeIcon}">  <ShellContent Title="Headlines" Route="headlines" Icon="{StaticResource HeadlinesIcon}" ContentTemplate="{DataTemplate view:HeadlinesView}" />  <Tab Title="News" Route="news" Icon="{StaticResource NewsIcon}">  <ShellContent Title="Local" Route="local" ContentTemplate="{DataTemplate view:HeadlinesView}" />  <ShellContent Title="Global" Route="global" ContentTemplate="{DataTemplate view:HeadlinesView}" />  </Tab>  </FlyoutItem>  <FlyoutItem Title="Settings" Icon="{StaticResource SettingsIcon}">  <ContentPage Title="Settings" />  </FlyoutItem>  <ShellContent Title="About" Icon="{StaticResource AboutIcon}">  <ContentPage Title="About"/>  </ShellContent>  </Shell> |

AppShell.xaml.cs

|  |
| --- |
| public partial class AppShell : Shell  {  public AppShell()  {  InitializeComponent();  Routing.RegisterRoute("articleview", typeof(Views.ArticleView));  }  } |

MauiProgram.cs

|  |
| --- |
| public static class MauiProgram  {  public static MauiApp CreateMauiApp()  {  var builder = MauiApp.CreateBuilder();  builder  .UseMauiApp<App>()  .ConfigureFonts(fonts =>  {  fonts.AddFont("FontAwesome.otf", "FontAwesome");  fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");  fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");  })  .RegisterAppTypes();  #if DEBUG  builder.Logging.AddDebug();  #endif  return builder.Build();  }  public static MauiAppBuilder RegisterAppTypes(this MauiAppBuilder mauiAppBuilder)  {  // Services  mauiAppBuilder.Services.AddSingleton<Services.INewsService>((serviceProvider) => new Services.NewsService());  mauiAppBuilder.Services.AddSingleton<ViewModels.INavigate>((serviceProvider) => new Navigator());  // ViewModels  mauiAppBuilder.Services.AddTransient<ViewModels.HeadlinesViewModel>();  //Views  mauiAppBuilder.Services.AddTransient<Views.AboutView>();  mauiAppBuilder.Services.AddTransient<Views.ArticleView>();  mauiAppBuilder.Services.AddTransient<Views.HeadlinesView>();  return mauiAppBuilder;  }  } |

Models/NewsApiModels.cs

|  |
| --- |
| namespace News.Models;  using System;  using System.Collections.Generic;  using System.Text.Json.Serialization;  public class Source  {  [JsonPropertyName("id")]  public string Id { get; set; }  [JsonPropertyName("name")]  public string Name { get; set; }  }  public class Article  {  [JsonPropertyName("source")]  public Source Source { get; set; }  [JsonPropertyName("author")]  public string Author { get; set; }  [JsonPropertyName("title")]  public string Title { get; set; }  [JsonPropertyName("description")]  public string Description { get; set; }  [JsonPropertyName("url")]  public string Url { get; set; }  [JsonPropertyName("urlToImage")]  public string UrlToImage { get; set; }  [JsonPropertyName("publishedAt")]  public DateTime PublishedAt { get; set; }  [JsonPropertyName("content")]  public string Content { get; set; }  }  public class NewsResult  {  [JsonPropertyName("status")]  public string Status { get; set; }  [JsonPropertyName("totalResults")]  public int TotalResults { get; set; }  [JsonPropertyName("articles")]  public List<Article> Articles { get; set; }  } |

Services/ INewsService.cs

|  |
| --- |
| using News.Models;  namespace News.Services;  public interface INewsService  {  public Task<NewsResult> GetNews(NewsScope scope);  } |

Services/NewsScope.cs

|  |
| --- |
| namespace News.Services;  public enum NewsScope  **{**  Headlines,  Local,  Global  **}** |

Services/NewsService.cs

|  |
| --- |
| using News.Models;  using System.Net.Http.Json;  namespace News.Services;  public class NewsService : INewsService, IDisposable  {  private bool disposedValue;  const string UriBase = "https://newsapi.org/v2";  readonly HttpClient httpClient = new()  {  BaseAddress = new(UriBase),  DefaultRequestHeaders = { { "user-agent", "maui-projects-news/1.0" } }  };  public async Task<NewsResult> GetNews(NewsScope scope)  {  NewsResult result;  string url = GetUrl(scope);  try  {  result = await httpClient.GetFromJsonAsync<NewsResult>(url);  }  catch (Exception ex)  {  result = new() { Articles = new() { new() { Title = $"HTTP Get failed: {ex.Message}", PublishedAt = DateTime.Now } } };  }  return result;  }  private string GetUrl(NewsScope scope) => scope switch  {  NewsScope.Headlines => Headlines,  NewsScope.Global => Global,  NewsScope.Local => Local,  \_ => throw new Exception("Undefined scope")  };  private static string Headlines => $"{UriBase}/top-headlines?country=us&apiKey={Settings.NewsApiKey}";  private static string Local => $"{UriBase}/everything?q=local&apiKey={Settings.NewsApiKey}";  private static string Global => $"{UriBase}/everything?q=global&apiKey={Settings.NewsApiKey}";  protected virtual void Dispose(bool disposing)  {  if (!disposedValue)  {  if (disposing)  {  httpClient.Dispose();  }  disposedValue = true;  }  }  public void Dispose()  {  // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method  Dispose(disposing: true);  GC.SuppressFinalize(this);  }  } |

ViewModes/BaseViewModel.cs

|  |
| --- |
| using CommunityToolkit.Mvvm.ComponentModel;  namespace News.ViewModels;  public partial class BaseViewModel : ObservableObject  **{**  public INavigate Navigation { get; init; }  internal BaseViewModel(INavigate navigation) => Navigation = navigation;  **}** |

ViewModes/ HeadlinesViewModel.cs

|  |
| --- |
| using CommunityToolkit.Mvvm.ComponentModel;  using CommunityToolkit.Mvvm.Input;  using News.Models;  using News.Services;  using System.Web;  namespace News.ViewModels;  public partial class HeadlinesViewModel : BaseViewModel  {  private readonly INewsService newsService;  [ObservableProperty]  private NewsResult currentNews;  public HeadlinesViewModel(INewsService newsService, INavigate navigation) : base(navigation)  {  this.newsService = newsService;  }  public async Task Initialize(string scope) => await Initialize(scope.ToLower() switch  {  "local" => NewsScope.Local,  "global" => NewsScope.Global,  "headlines" => NewsScope.Headlines,  \_ => NewsScope.Headlines  });  public async Task Initialize(NewsScope scope)  {  CurrentNews = await newsService.GetNews(scope);  }  [RelayCommand]  public async Task ItemSelected(object selectedItem)  {  var selectedArticle = selectedItem as Article;  var url = HttpUtility.UrlEncode(selectedArticle.Url);  await Navigation.NavigateTo($"articleview?url={url}");  }  } |

ViewModels/INavigate.cs

|  |
| --- |
| namespace News.ViewModels;  public interface INavigate  {  Task NavigateTo(string route);  Task PushModal(Page page);  Task PopModal();  } |

Views/AboutView.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="News.Views.AboutView"  Title="AboutView">  <VerticalStackLayout>  <Label  Text="AboutView!"  VerticalOptions="Center"  HorizontalOptions="Center" />  </VerticalStackLayout>  </ContentPage> |

Views/AboutView.xaml.cs

|  |
| --- |
| namespace News.Views;  public partial class AboutView : ContentPage  {  public AboutView()  {  InitializeComponent();  }  } |

Views/ArticleItem.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="News.Views.ArticleItem"  xmlns:models="clr-namespace:News.Models"  x:DataType="models:Article">  <Grid Margin="0">  <Grid.RowDefinitions>  <RowDefinition Height="10" />  <RowDefinition Height="60" />  <RowDefinition Height="20" />  <RowDefinition Height="10" />  <RowDefinition Height="1" />  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="65" />  <ColumnDefinition Width="\*" />  </Grid.ColumnDefinitions>  <Label Grid.Row="1" Grid.Column="1" Text="{Binding Title}" Padding="10,0" FontSize="Body" FontAttributes="Bold" />  <Label Grid.Row="2" Grid.Column="1" Text="{Binding PublishedAt, *StringFormat*='{0:MMMM d, yyyy}'}" Padding="10,0,0,0" FontSize="Micro" />  <Frame Grid.Row="1" Grid.Column="0" Grid.RowSpan="2" CornerRadius="15" IsClippedToBounds="True" Padding="0" Margin="0,0,0,0" BackgroundColor="#667788" IsVisible="{x:OnPlatform *Default*=False, *iOS*=True}" >  <Image Source="{Binding UrlToImage}" Aspect="AspectFill" HeightRequest="55" HorizontalOptions="Center" VerticalOptions="Center" />  </Frame>  <Border Grid.Row="1" Grid.RowSpan="2" StrokeShape="RoundRectangle 15,15,15,15" Padding="0" Margin="0,0,0,0" BackgroundColor="#667788" IsVisible="{x:OnPlatform *Default*=True, *iOS*=False}">  <Image Source="{Binding UrlToImage}" Aspect="AspectFill" HeightRequest="55" HorizontalOptions="Center" VerticalOptions="Center" />  </Border>  <BoxView Grid.Row="4" Grid.ColumnSpan="2" BackgroundColor="LightGray" />  </Grid>  </ContentView> |

Views/ArticleItem.xaml.cs

|  |
| --- |
| namespace News.Views;  public partial class ArticleItem : ContentView  {  public ArticleItem()  {  InitializeComponent();  }  } |

Views/ArticleView.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="News.Views.ArticleView"  Title="ArticleView">  <WebView Source="{Binding .}" />  </ContentPage> |

Views/ArticleView.xaml.cs

|  |
| --- |
| using System.Web;  namespace News.Views;  [QueryProperty("Url", "url")]  public partial class ArticleView : ContentPage  {  public string Url  {  set  {  BindingContext = new UrlWebViewSource  {  Url = HttpUtility.UrlDecode(value)  };  }  }  public ArticleView()  {  InitializeComponent();  }  } |

Views/HeadlinesView.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="News.Views.HeadlinesView"  xmlns:views="clr-namespace:News.Views"  xmlns:viewModels="clr-namespace:News.ViewModels"  xmlns:models="clr-namespace:News.Models"  x:Name="headlinesview"  x:DataType="viewModels:HeadlinesViewModel"  Title="Home" Padding="14">  <CollectionView ItemsSource="{Binding CurrentNews.Articles}">  <CollectionView.EmptyView>  <Label Text="Loading" />  </CollectionView.EmptyView>  <CollectionView.ItemTemplate >  <DataTemplate x:DataType="models:Article">  <ContentView>  <ContentView.GestureRecognizers>  <TapGestureRecognizer Command="{Binding *BindingContext*.ItemSelectedCommand, Source={x:Reference headlinesview}}" CommandParameter="{Binding .}" />  </ContentView.GestureRecognizers>  <views:ArticleItem />  </ContentView>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </ContentPage> |

Views/HeadlinesView.xaml.cs

|  |
| --- |
| using News.ViewModels;  namespace News.Views;  public partial class HeadlinesView : ContentPage  {  readonly HeadlinesViewModel viewModel;  public HeadlinesView(HeadlinesViewModel viewModel)  {  this.viewModel = viewModel;  InitializeComponent();  Task.Run(async () => await Initialize(GetScopeFromRoute()));  }  private async Task Initialize(string scope)  {  BindingContext = viewModel;  await viewModel.Initialize(scope);  }  private string GetScopeFromRoute()  {  // Hack: As the shell can't define query parameters  // in XAML, we have to parse the route.  // as a convention the last route section defines the category.  var route = Shell.Current.CurrentState.Location  .OriginalString.Split("/").LastOrDefault();  return route;  }  } |

