What's New With .NET MAUI & EF Core in .NET9?

Engincan VESKE

Software Engineer at Volosoft

@EngincanVeske

EngincanV

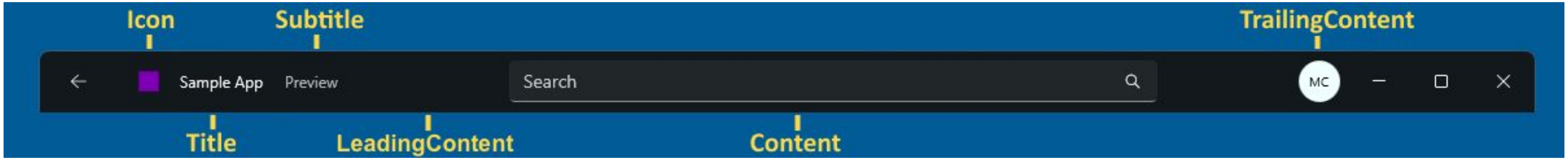
.NET MAUI

- New Controls
- Control Enhancements
- Compiled Bindings in XAML & Code

.NET MAUI - New Controls

.NET MAUI 9 includes two new controls: **Titlebar** & **HybridWebView**

Titlebar for Windows:



```
<Window.TitleBar>
    <TitleBar x:Name="TitleBar"  Title="Sample App"  Icon="icon.png" HeightRequest="46">
        <TitleBar.Content>
            <SearchBar Placeholder="Search"
                       PlaceholderColor="White"
                       MaximumWidthRequest="300"
                       HorizontalOptions="Fill"
                       VerticalOptions="Center" />
        </TitleBar.Content>
    </TitleBar>
</Window.TitleBar>
```

.NET MAUI - New Controls

HybridWebView:

- Using HybridWebView (file -> web content of the app)

```
<HybridWebView
    x:Name="hwv"
    DefaultFile="index.html"
    HybridRoot="wwwroot"
    RawMessageReceived="OnRawMessageReceived" />
```

- Invoking JS methods from C#

```
var sum = await hwv.InvokeJsMethodAsync<int>("JsAddNumbers", 123, 456);
```

- Invoking C# methods from JavaScript

```
const returnValue = await window.HybridWebView.InvokeDotNet('DoSyncWorkParamsReturn', [123, 'hello']);
```

.NET MAUI - Control Enhancements

- CollectionView and CarouselView handlers for IOS & MACCATALYST:

```
#if IOS || MACCATALYST
builder.ConfigureMauiHandlers(handlers =>
{
    handlers.AddHandler<Microsoft.Maui.Controls.CollectionView, Microsoft.Maui.Controls.Handlers.Items2.CollectionViewHandler2>();
    handlers.AddHandler<Microsoft.Maui.Controls.CarouselView, Microsoft.Maui.Controls.Handlers.Items2.CarouselViewHandler2>();
});
#endif
```

- **Button** controls on IOS now respects spacing, padding, border width and margins

- **BackButtonBehavior** binding mode changes as **BindingMode.OneWay** instead of **BindingMode.OneTime**

abp.io

.NET MAUI - Compiled Bindings in XAML & Code

XAML

```xml
<ContentPage x:DataType="local:AnimalsPageViewModel">
    <!-- binds to AnimalsPageViewModel.Animals -->
    <CollectionView ItemsSource="{Binding Animals}">
        <CollectionView.ItemTemplate>
            <DataTemplate x:DataType="local:Animal">
                <!-- binds to Animal.Name -->
                <Label Text="{Binding Name}" />
            </DataTemplate>
        </CollectionView.ItemTemplate>
    </CollectionView>
</ContentPage>
```

Code

```csharp
// in .NET 8
MyLabelName.SetBinding(Label.TextProperty, "Text");
```

```csharp
// in .NET 9
MyLabelName.SetBinding(
   Label.TextProperty,
   static (Entry entry) => entry.Text
);
```

abp.io

EF Core

- Azure Cosmos DB for NoSQL
- AOT & Pre-Compiled Queries
- LINQ & SQL Translation
- Migrations & Model Building

EF Core - Azure Cosmos DB for NoSQL

EF Core 9 introduces significant updates for Azure Cosmos DB, focusing on NoSQL scenarios. The improvements provide better query performance and enhanced flexibility when working with Cosmos DB, making it easier to handle complex data models in cloud environments. For example:

- 🚀**Improved: Querying with partition keys**

```
var sessions = await context.Sessions
    .Where(b => b.PartitionKey == "someValue" && b.Username.StartsWith("x"))
    .ToListAsync();
```

Output:

```
Executed ReadNext (189.8434 ms, 2.8 RU) ActivityId='...', Container='test', Partition='["someValue"]', Parameters=[]
SELECT VALUE c
FROM root c
WHERE STARTSWITH(c["Username"], "x")
```

EF Core - AOT & Pre-Compiled Queries

⚠️ **NativeAOT & Query Precompilation is experimental and not ready to use for production!**

- Support for Ahead-of-Time (AOT) compilation
- Pre-compiled queries for faster execution

For example:

```
var blogs = await context.Blogs.Where(b => b.Name == "foo").ToListAsync();
```

```
var efCoreInterception = ((IRelationalCommandTemplate)(new RelationalCommand(materializerLiftableConstantContext.CommandBuilderDependencies,
   "SELECT [b].[Id], [b].[Name]\nFROM [Blogs] AS [b]\nWHERE [b].[Name] = N'foo'",
new IRelationalParameter[] { })));
```

EF Core - LINQ & SQL Translation

LINQ Query:

```
var customers = await context.Customers.Where(o => o.Orders.Any()).ToListAsync();
```

Prior to .NET 9:

```
SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE EXISTS (
    SELECT 1
    FROM [Orders] AS [o]
    LEFT JOIN [DiscountedOrders] AS [d] ON [o].[Id] = [d].[Id]
    WHERE [c].[Id] = [o].[CustomerId])
```

👉 Table Pruning

.NET 9+:

```
SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE EXISTS (
    SELECT 1
    FROM [Orders] AS [o]
    WHERE [c].[Id] = [o].[CustomerId])
```

abp.io

GroupBy with complex types:

```
var groupedAddresses = await context.Stores
    .GroupBy(b => b.StoreAddress)
    .Select(g => new { g.Key, Count = g.Count() })
    .ToListAsync();
```

```
SELECT [s].[StoreAddress_City], [s].[StoreAddress_Country], [s].[StoreAddress_Line1],
[s].[StoreAddress_Line2], [s].[StoreAddress_PostCode], COUNT(*) AS [Count]
FROM [Stores] AS [s]
GROUP BY [s].[StoreAddress_City], [s].[StoreAddress_Country], [s].[StoreAddress_Line1],
[s].[StoreAddress_Line2], [s].[StoreAddress_PostCode]
```

ExecuteUpdate with complex types:

```
var newAddress = new Address("Gressenhall Farm Shop", null, "Beetley", "Norfolk", "NR20 4DR");

await context.Stores
    .Where(e => e.Region == "Germany")
    .ExecuteUpdateAsync(s => s.SetProperty(b => b.StoreAddress, newAddress));
```

```
UPDATE [s]
SET [s].[StoreAddress_City] = @__complex_type_newAddress_0_City,
    [s].[StoreAddress_Country] = @__complex_type_newAddress_0_Country,
    [s].[StoreAddress_Line1] = @__complex_type_newAddress_0_Line1,
    [s].[StoreAddress_Line2] = NULL,
    [s].[StoreAddress_PostCode] = @__complex_type_newAddress_0_PostCode
FROM [Stores] AS [s]
WHERE [s].[Region] = N'Germany'
```

EF Core - Migrations & Model Buildings

- Enhanced migrations for handling schema changes
- New model-building APIs (support for **Read-Only Primitive Collections** and more…)
- 🚀**Improved: Data Seeding with UseSeeding & UseAsyncSeeding extension methods**

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    => optionsBuilder
        .UseSqlServer(@"...")
        .UseAsyncSeeding(async (context, _, cancellationToken) =>
        {
             context.Set<Blog>().Add(new Blog { Url = "http://test.com" });
             await context.SaveChangesAsync(cancellationToken);
        });
```

THANKS FOR LISTENING