NAVIGATION

Vincent Uhlmann
IT-Akademie Dr. Heuer GmbH

NAVIGATION

- Navigation bezieht sich auf die Struktur und das Wechseln von Seiten innerhalb einer Anwendung
- Es gibt zwei Hauptmethoden zur Navigation in .NET MAUI

- Navigation ohne Shell
- Navigation mit Shell

WICHTIGE KONZEPTE DER NAVIGATION

- **Navigation Stack**: Ein Stapelkonzept, das es erlaubt, Seiten zu "stapeln" und in umgekehrter Reihenfolge darauf zuzugreifen
- **Forward Navigation**: Bewegen zu einer neuen Seite, oft durch das Auswählen eines Elements oder das Ausführen einer Aktion
- **Backward Navigation**: Rückkehr zur vorherigen Seite, oft über einen Zurück-Button oder eine Geste wie Wischen

NAVIGATION OHNE SHELL

- **NavigationPage** ist ein Container, der eine Navigationssymbolleiste bietet und den Benutzern ermöglicht, zwischen Seiten vor- und zurückzunavigieren
- Eine **ContentPage** wird als Kind einer NavigationPage hinzugefügt, wodurch ein Stapel von Seiten (Navigation Stack) verwaltet wird

```
// Eine NavigationPage als Hauptseite setzen
Application.Current.MainPage = new NavigationPage(new MainPage());

// Navigation zu einer neuen Seite
await Navigation.PushAsync(new DetailPage());

// Navigation zurück
await Navigation.PopAsync();
```

NAVIGATION MIT SHELL

- **Shell** bietet eine hochgradig konfigurierbare Infrastruktur zur Definition der Navigation und der App-Architektur in einer einzigen XAML-Datei
- Vereinfacht die Erstellung von Tab- und Flyout-basierten Layouts
- Erlaubt das Definieren von Routen, die wie URLs behandelt werden, um einfache Navigation und tiefe Verlinkungen zu ermöglichen

```
<Shell>
    <TabBar>
        <Tab Title="Home" Icon="home.png">
            <ShellContent Route="home" ContentTemplate="{DataTemplate local:HomePage}" />
        </Tab>
        <Tab Title="Settings" Icon="settings.png">
            <ShellContent Route="settings" ContentTemplate="{DataTemplate local:SettingsPage}" />
        </Tab>
    </TabBar>
</Shell>
```

NAVIGATION MIT SHELL

- Routen, die nicht in der visuellen Hierarchie aufgeführt werden, können manuell hinzugefügt werden
- Hierzu wird die Shell.Current.RegisterRoute() Methode genutzt
- Meistens im Konstruktor der AppShell.cs Datei

```
Routing.RegisterRoute("detail", typeof(DetailPage));
```

ROUTENSYNTAX

- In der Shell wird die Navigation durch eine spezielle Syntax gesteuert, die es ermöglicht, die Hierarchie und Beziehung zwischen den Seiten klar zu definieren
- **route:** In der Routenhierarchie wird von der aktuellen Position aus nach oben nach der angegebenen Route gesucht. Die übereinstimmende Seite wird per Push an den Navigationsstapel übertragen
- /**route:** In der Routenhierarchie wird von der aktuellen Position aus nach unten nach der angegebenen Route gesucht. Die übereinstimmende Seite wird per Push an den Navigationsstapel übertragen
- //**route:** In der Routenhierarchie wird von der aktuellen Position aus nach oben nach der angegebenen Route gesucht. Die übereinstimmende Seite ersetzt den Navigationsstapel
- ///**route:** In der Routenhierarchie wird von der aktuellen Position aus nach unten nach der angegebenen Route gesucht. Die übereinstimmende Seite ersetzt den Navigationsstapel

SHELL NAVIGATION

- GoToAsync() ist die Methode, die in .NET MAUI Shell für die Navigation verwendet wird
- Sie erlaubt die asynchrone Navigation zwischen Seiten, basierend auf Routennamen oder Routenmustern
- Die Methode akzeptiert eine Zeichenkette, die die Route beschreibt, zu der navigiert werden soll
- Diese Zeichenkette kann einfache Routennamen oder komplexe Ausdrücke mit Routenparametern und Navigationsbefehlen enthalten

```
await Shell.Current.GoToAsync("settings");
```

RÜCKWÄRTSNAVIGATION

- Die Rückwärtsnavigation kann erfolgen, indem Sie ".." als Argument zur GoToAsync-Methode hinzufügen

```
await Shell.Current.GoToAsync("..");
```

- Die Rückwärtsnavigation mit ".." kann auch mit einer Route kombiniert werden

```
await Shell.Current.GoToAsync("../route");
```

PARAMETERÜBERGABE MIT QUERY

- **URL-basierte Navigation** in Shell ermöglicht das Hinzufügen von Query-Parametern direkt in der Route
- Diese Methode ist besonders nützlich, wenn Sie primitive Datentypen oder einfache Strings übergeben wollen

```
await Shell.Current.GoToAsync($"detail?id={item.Id}");
```

PARAMETERÜBERGABE MIT DICTIONARY

- **URL-basierte Navigation** in Shell ermöglicht das Hinzufügen von Parametern durch ein Dictionary
- Diese Methode ist besonders nützlich, wenn Sie Objekte oder Sammlungen von Daten zwischen Seiten übermitteln müssen, die nicht einfach als primitive Datentypen oder einfache Strings übergeben werden können

```
var navigationParameters = new Dictionary<string, object>
{
    { "model", myModel },
    { "isNew", true }
};

await Shell.Current.GoToAsync($"detailpage", navigationParameters);
```

EMPFANGEN VON NAVIGATIONSDATEN

- Die Klasse, die die Seite darstellt, zu der navigiert wird, oder die Klasse für den BindingContext der Seite kann um ein QueryPropertyAttribute für jeden Abfrageparameter ergänzt werden

```
[QueryProperty(nameof(ItemId), "id")]
public class Test
{
    public int ItemId { get; set; }
}
```

- Die Klasse, die die Seite darstellt, zu der navigiert wird, oder die Klasse für den BindingContext der Seite kann die IQueryAttributable-Schnittstelle implementieren

```
public class MealsViewModel : IQueryAttributable
...
if(query.ContainsKey("newMeal"))
{
    object o = query["newMeal"];
    Meal meal = (Meal)o;
}
```