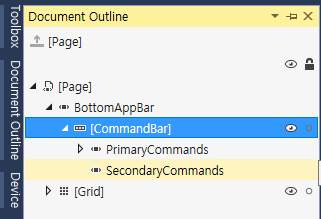
Windows Phone 8.1 App 개발 실습(ContosoCookbookSimple)

이번 세션에서는 페이지간 이동과 캐시모드, 데이터 바인딩에 대해 다룹니다.

# Navigation (페이지 이동) 다뤄 보기

페이지간 이동에 대해 알아 봅니다.

1. Session 3 Demos 폴더 > 1 - Page Navigation BEGIN 폴더 안의 ContosoCookbookSimple 솔루션 파일을 엽니다.
2. Solution Explorer에서 MainPage.xaml 파일을 더블클릭하여 엽니다.
3. **Document Outline** 메뉴를 열고 **BottomAppBar** 아래의 **SecondaryCommands** 를 마우스 우클릭 합니다. 이후 **Add AppBarButton 버튼을 선택합니다.**



1. 버튼을 선택하면 앱바 아래 텍스트 버튼이 하나 추가 된 것을 보실수가 있습니다. Windows 8.1 과 같은 XAML을 사용하지만 Windows 8.1 프로젝트 처럼 아이콘버튼을 추가하실 수 없고 텍스트 버튼만 추가가 가능합니다.
2. 아래 그림과 같이 추가된 버튼의 Label 속성값을 **about** 으로 변경합니다.

<CommandBar>

            <CommandBar.SecondaryCommands>

                <AppBarButton Label="about"/>

1. 디자이너 편집기에서 추가된 버튼을 더블클릭 합니다. 더블클릭하게 되면 MainPage.xaml.cs 파일로 넘어가면서 자동으로 버튼을 클릭했을 때 발생하는 이벤트핸들러를 생성해 줍니다.
2. 이벤트 핸들러 코드를 작성하기 전에, AboutPage.xaml 파일을 프로젝트에 추가해줍니다.
3. Solution Explorer 에서 마우스 우클릭 후 Add > Existing Item.. 순으로 선택합니다.
4. 위로 버튼을 통해 Session 3 Demos 폴더로 이동합니다. Items 폴더를 연 후 AboutPage.xaml 파일을 더블클릭하여 프로젝트에 추가합니다.
5. 아래의 코드를 이벤트 핸들러에 복사하여 넣습니다.

private void AppBarButton\_Click(object sender, RoutedEventArgs e)

{

Frame.Navigate(typeof(AboutPage)); //붙여 넣을 코드

}

1. F5 버튼을 눌러 애플리케이션을 실행합니다. about 버튼이 작동하는지 확인합니다.
2. 시뮬레이터에 뒤로가기 버튼을 눌러봅니다. 이때 앱이 정지되게 되는데 이는 이번 프로젝트가 Blank App 프로젝트 템플릿을 통해 만들어졌고, 하드웨어의 뒤로가기 버튼을 처리하는 부분이 없기 때문입니다. 디버깅을 종료합니다.
3. 뒤로가기 버튼 처리를 구현하기 위해 App.xaml 파일을 확장하고 App.xaml.cs 파일을 더블클릭하여 엽니다.
4. 아래와 같이 App 클래스의 생성자 구현부분 안에 아래코드를 복사하여 넣습니다.

public App()

{

    this.InitializeComponent();

    this.Suspending += this.OnSuspending;

    Windows.Phone.UI.Input.HardwareButtons.BackPressed += HardwareButtons\_BackPressed;

}

1. 그 다음줄에 아래의 이벤트 핸들러 코드를 복사하여 넣습니다.

private void HardwareButtons\_BackPressed(object sender, Windows.Phone.UI.Input.BackPressedEventArgs e)

{

Frame rootFrame = Window.Current.Content as Frame;

if (rootFrame != null && rootFrame.CanGoBack)

{

rootFrame.GoBack();

e.Handled = true;

}

}

1. 애플리케이션을 다시 실행하여 About 페이지와 뒤로가기 버튼의 작동을 확인합니다.

# 캐시모드

간단한 데모를 통해 캐시모드가 페이지 사용에 어떤 영향을 주는지 살펴보겠습니다.

1. Session 3 Demos > **2 – CacheModeDemo – BEGIN** 폴더의 **CacheModeDemo** 솔루션 파일을 엽니다.
2. F5 버튼을 눌러 애플리케이션을 실행합니다. 간단한 페이지 이동을 하는 애플리케이션입니다.
3. **To Page 2** 버튼을 눌러 다음 페이지로 이동합니다.
4. Page 2에서 임의의 값과 버튼을 선택하고 **To Page 3** 버튼을 눌러 다음 페이지로 이동합니다.
5. 뒤로가기 버튼을 눌러 다시 Page 2로 돌아가봅니다. 기존에 선택했던 값들이 없어졌음을 볼 수 있습니다. 디버깅을 종료합니다.
6. Page2.xaml 파일을 확장한 후 Page2.xaml.cs 파일을 더블클릭하여 엽니다.
7. Page2의 생성자 부분에는 아래와 같은 코드가 있습니다.

this.NavigationCacheMode = Windows.UI.Xaml.Navigation.NavigationCacheMode.Disabled;

1. 위 코드의 Disabled 를 Required 로 바꿔준 후 실행해봅니다. 4~5번 과정을 똑같이 재현해보면 기존 선택값들이 남아 있음을 보실 수 있습니다.

반대로 Page 2에서 MainPage로 이동한 후 다시 돌아오면 어떤 상태 일까요?

1. Page 2에서 한번 더 뒤로가기 버튼을 눌러 MainPage로 이동한 후 다시 **To Page 2** 버튼을 눌러 돌아와 봅니다.
2. 역시 기존에 입력했던 값들이 남아 있음을 보실 수가 있습니다. 디버깅을 종료합니다.

앱의 시나리오에 따라 이 경우에는 값들이 초기화 되야 할 수도 있습니다. 추후 이런 구현에 대해 알아 보도록 하겠습니다.

# NavigationHelper

NavigationHelper를 사용하여 페이지의 상태를 관리하는 방법에 대해 알아봅니다.

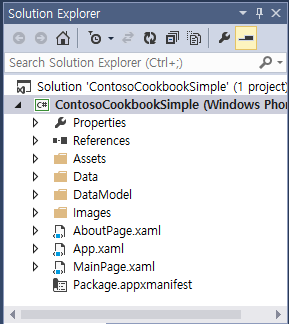
1. Session 3 Demos 폴더 > **3 – CacheModeNavigationHelperDemo – End** 폴더 안의 **CacheModeDemo** 솔루션 파일을 엽니다.
2. 이 솔루션 파일은 앞서 살펴보았던 솔루션 파일의 수정된 버전입니다. MainPage.xaml 파일을 확장한 후 MainPage.xaml.cs 파일을 더블클릭하여 엽니다.
3. Ctrl + F 버튼을 눌러 검색창을 띄운 후 검색어로 Navigationhelper 를 입력해 봅니다. Navigationhelper가 어떻게 선언되고 사용되었는지 확인합니다.
4. App.Xaml.cs 파일을 더블클릭하여 엽니다. 앞서 추가하였던 뒤로가기 버튼의 이벤트가 주석처리 되어 있음을 보실수 있습니다.
5. F5 버튼을 눌러 애플리케이션을 실행합니다. 앞서 실행 했던 것처럼 Page 2에서 입력한 값들이 유지 되는 것을 보실 수가 있습니다. 디버깅을 종료합니다.

이는 NavigationHelper가 같은 작업을 수행해 주기 때문입니다. 이어질 강의에서는 NavigationHelper와 함께 사용되는 SuspensionManager에 대해서도 함께 알아 보도록 하겠습니다.

# 데이터 바인딩

Data로 사용되는 텍스트 파일과 DataModel을 추가하여 데이터를 바인딩 하는 방법에 대해 알아봅니다.

1. Session 3 Demos 폴더 > **4 – Data Binding – BEGIN** 폴더안의 **ContosoCookbookSimple** 솔루션 파일을 엽니다.
2. Session 3 Demos 폴더 > Items 폴더 안의 **Data** 폴더와 **DataModel** 폴더를 선택하고 Solution Explorer의 프로젝트 파일에 끌어다 놓습니다. 추가된 모습은 아래와 같습니다.



1. App.xaml.cs 파일을 더블클릭하여 엽니다.
2. Ctrl + F 버튼을 눌러 검색창을 띄운 후 검색어로 **OnLaunched** 를 입력해 봅니다. 함수에 아래와 같이 **async** 키워드를 추가합니다.

protected async override void OnLaunched(LaunchActivatedEventArgs e)

1. **OnLaunched** 함수의 구현부 첫줄에 아래와 같이 코드를 추가해 줍니다.

protected async override void OnLaunched(LaunchActivatedEventArgs e)

{

**// Load recipe data**

**await ContosoCookbookSimple.DataModel.RecipeDataSource.LoadLocalDataAsync();**

**//…**

**}**

1. MainPage.xaml.cs 파일을 더블클릭하여 엽니다. **OnNavigatedTo** 함수를 찾아서 아래와 같이 ingredients 리스트 선언부와 바인딩 구현 부분을 주석처리 해 줍니다.

//List<string> ingredients = new List<string> {

// "Shortcrust Pastry: ","220g plain flour","120g cold butter",

// "1 tablespoon very cold water","1 large egg",

// "separated into yolk and white","Filling: ","1 teaspoon oil",

// "1 small onion","220g bacon","finely chopped 5 large eggs",

// "125ml cream","1/4 teaspoon ground nutmeg Salt and pepper",

// "110g grated tasty cheddar cheese"

//};

//IngredientsListBox.ItemsSource = ingredients;

1. 아래의 코드를 그 다음 라인에 추가해 줍니다.

//IngredientsListBox.ItemsSource = ingredients;

var item = ContosoCookbookSimple.DataModel.RecipeDataSource.GetItem("2000");

this.DataContext = item;

1. 위 작업을 통해 Data 폴더의 텍스트 파일에서 아이템을 가져와 바인딩을 하게 됩니다.
2. MainPage.xaml 파일을 더블클릭하여 엽니다. 텍스트 편집기에서 XAML 을 수정함으로써 바인딩된 데이터를 표시할 수 있습니다. 아래의 하이라이트 된 부분을 참고하여 코드를 수정합니다.

<Page.BottomAppBar>

<CommandBar>

<CommandBar.SecondaryCommands>

<AppBarButton Label="about" Icon="Help" Click="AppBarButton\_Click"/>

</CommandBar.SecondaryCommands>

<AppBarButton Icon="List" Label="ingredients">

<AppBarButton.Flyout>

<Flyout>

<Grid Height="400" Margin="0,25,0,0">

<ListBox x:Name="IngredientsListBox" ItemsSource="{Binding Ingredients}"/>

</Grid>

</Flyout>

</AppBarButton.Flyout>

</AppBarButton>

</CommandBar>

</Page.BottomAppBar>

<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition/>

</Grid.RowDefinitions>

<StackPanel Margin="10,25,0,30">

<TextBlock Style="{StaticResource TitleTextBlockStyle}" Text="CONTOSO COOKBOOK" Margin="0,20,0,0"/>

<TextBlock Style="{StaticResource HeaderTextBlockStyle}" Text="{Binding ShortTitle}" Margin="0,20,0,0"/>

</StackPanel>

<Grid Grid.Row="1" Margin="10,0,0,70">

<Grid.RowDefinitions>

<RowDefinition Height="109\*"/>

<RowDefinition Height="155\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Image Source="{Binding ImagePath}" Stretch="UniformToFill"/>

<ScrollViewer Grid.Row="1" Margin="0,10,0,0">

<TextBlock x:Name="InstructionsTextBlock" TextWrapping="Wrap"

Style="{StaticResource BodyTextBlockStyle}"

Text="{Binding Directions}"/>

</ScrollViewer>

<StackPanel Grid.Row="2" Orientation="Horizontal">

<TextBlock Text="Prep time:" Style="{StaticResource BodyTextBlockStyle}"

Foreground="{StaticResource PhoneAccentBrush}"/>

<TextBlock x:Name="PrepTimeTextBlock" Text="{Binding PrepTime}" Style="{StaticResource BodyTextBlockStyle}" Margin="10,0,0,10"/>

</StackPanel>

</Grid>

1. F5 버튼을 눌러 애플리케이션을 실행합니다. 정상적으로 실행됨을 보실수가 있습니다. 디버깅을 종료합니다.
2. MainPage.xaml.cs 파일을 더블클릭하여 엽니다. **OnNavigatedTo** 함수를 찾아서 아래와 같이 GetItem함수의 매개변수를 2001로 변경합니다.

var item = ContosoCookbookSimple.DataModel.RecipeDataSource.GetItem("2001");

1. 다시 F5 버튼을 눌러 애플리케이션을 실행합니다. 변경된 그림과 데이터가 나타남을 보실수가 있습니다. 디버깅을 종료합니다.

# 디자인 시점의 데이터

앞서 보여드렸던 데이터 바인딩의 경우 애플리케이션을 실행해야 결과를 나타내기 때문에 디자이너의 입장에서는 결과를 예측하기가 어렵다는 단점이 있습니다. 디자인을 할때 바인딩 된 데이터를 보는 방법에 대해 알아봅니다.

1. Session 3 Demos 폴더 > **5 - Design-time Data BEGIN** 폴더안의**ContosoCookbookSimple** 솔루션 파일을 엽니다.
2. Solution Explorer의 **Assets** 폴더를 마우스 우클릭한 후 **Add > Existing items** 메뉴를 선택합니다.
3. Session 3 Demos 폴더 > **Items >** **Assets** 폴더로 이동하여 3가지 파일(**LIghtGray.png**, **MediumGray.png**, **DarkGray.png**)을 모두 선택하여 추가합니다.
4. MainPage.xaml 파일을 더블클릭하여 엽니다. 텍스트 편집기에서 아래와 같이 하이라이트 된 부분을 수정합니다.

<Page

x:Class="ContosoCookbookSimple.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:ContosoCookbookSimple"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:data="using:ContosoCookbookSimple.DataModel"

mc:Ignorable="d"

d:DataContext="{Binding Source={d:DesignInstance Type=data:SampleDataItem, IsDesignTimeCreatable=True}}">

이 부분을 수정해도 혹시 오류가 나는 경우에는 오류가 나는 부분을 다시 입력해 보시기 바랍니다.

1. **xmlns:data** 속성은 xml 네임스페이스를 선언하여 **ContosoCookbookSimple.DataModel** 네임스페이스 내의 객체들을 참조할 수 있게 합니다.
2. D:DataContext 부분은 런타임에만 수행되던 SampleDataItem의 인스턴스를 만들어 주는 작업을 해 줍니다.
3. 프로젝트를 빌드합니다 (Ctrl + Shift + F5). 실제 데이터와 3번 과정에서 추가한 회색 이미지를 애플리케이션의 실행없이도 확인할 수 있습니다.