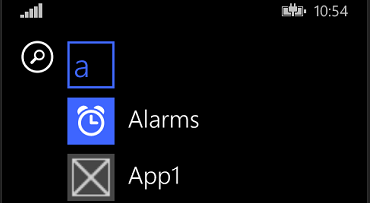
Windows Phone 8.1 App 개발 실습(앱 라이프사이클)

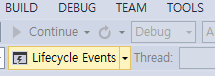
# Suspend / Resume 모드

앱을 일시 중단(Suspend)시키고, 다시 시작(Resume)하는 방법에 대해 알아봅니다.

1. Session 7 폴더 > **Demo1** 폴더안의 **App1**솔루션 파일을 엽니다.
2. 솔루션탐색기에서 Data.cs 파일을 더블클릭하여 엽니다. Data 클래스는 10진수값을 가질 수 있는 Value 속성으로 구성된 클래스입니다.
3. MainPage.xaml.cs 파일을 더블클릭하여 엽니다. MainPage() 생성자에는 0.2초마다 값을 0.2씩 증가시키는 DispatcherTimer가 구현되어 있습니다. (시간을 측정하는 타이머의 역할입니다.)   
   현재 소스코드는 앱을 일시 중단하고 다시 시작할때 적합한 이벤트 처리가 되어 있지 않습니다. 하지만 디버거에서 동작할 때는 앱에서 벗어나도 일시 중단 이벤트가 발생되지 않기 때문에 타이머가 계속 동작합니다. 이를 확인해 보겠습니다.
4. F5 버튼을 눌러 앱을 실행합니다. 타이머의 시간을 확인한 후 뒤로가기 버튼을 선택합니다. 백버튼을 누른 상태에서 나오는 앱 전환 메뉴에서 다시 App1 앱으로 돌아간 후 시간을 확인합니다.
5. 앱이 일시 중단되지 않기 때문에 타이머가 시간의 손실 없이 잘 작동할 것입니다. 비주얼 스튜디오에서 Shift + F5 버튼을 눌러 실행을 종료합니다.
6. 이번에는 디버거를 이용하지 않고 App1을 실행해 보겠습니다. 열려있는 시뮬레이터에서 App1을 선택하여 엽니다.



1. 타이머의 시간을 확인한 뒤 뒤로가기 버튼을 클릭합니다. 다시 App1 앱으로 돌아간 후 타이머의 시간을 확인합니다.
2. 이번에는 기존과는 다르게 시간의 손실이 있었을 것입니다. 앞서 말씀드린 것 처럼 아직 소스코드에 일시 중단 / 다시 시작 이벤트핸들러가 구현되지 않았기 때문입니다.
3. 이번에는 디버거를 이용하면서도 일시 중단 / 다시 시작 이벤트를 발생시키는 방법에 대해 살펴보겠습니다. 다시 비주얼 스튜디오에서 F5 버튼을 눌러 앱을 실행합니다.
4. 비주얼 스튜디오 상단에 보면 아래와 같이 Lifecycle Events라는 버튼이 있습니다. 타이머의 시간을 확인한 뒤Suspend 버튼을 선택하여 앱을 일시 중단시켜 봅니다.



1. 다시 Resume 버튼을 선택하여 앱을 다시 시작합니다. 앱이 일시 중단되었다가 다시 시작되었기 때문에 시간의 손실이 있었을 것입니다. 이러한 방법으로 디버거에서 일시 중단 / 다시 시작 이벤트를 발생시킵니다.
2. 솔루션 탐색기의 App.xaml.cs 파일을 더블클릭하여 엽니다. 첫번째 줄의 주석을 제거합니다.

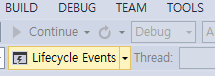


1. ADDHANDLERS 영역의 코드는 앱이 일시 중단되면 타이머의 현재 시간을 저장하고 앱이 다시 시작되면 일시 중단되었던 만큼의 시간을 더해서 시간의 손실을 없애줍니다.
2. 앱을 실행한 후 위 11번의 내용처럼 디버거의 Lifecycle Events 버튼을 활용하여 앱의 동작을 확인합니다.

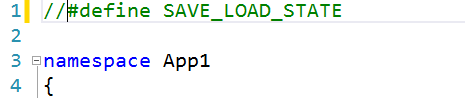
# Suspend/Terminate/Launch 모드

OnSuspending 이벤트를 발생시키고 로컬 파일에 상태를 저장하는 방법에 대해 알아봅니다.

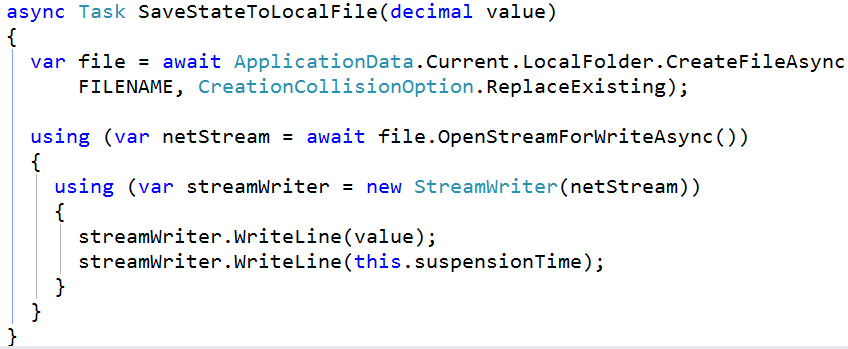
1. Session 7 폴더 > **Demo2** 폴더안의 **App1**솔루션 파일을 엽니다.
2. 앞서 살펴본 코드와 일부 같지만 현재의 상태를 로컬파일에 저장하는 등 추가적인 구현이 되어 있습니다.
3. F5 버튼을 눌러 앱을 실행합니다.
4. 비주얼 스튜디오 상단의 Lifecycle Events 에서 Suspend & Shutdown 버튼을 선택합니다.



1. 앱을 다시 실행합니다. 앱이 다시 실행되었고 타이머의 시간이 초기화 되었음을 확인할 수 있습니다.
2. App.xaml.cs 파일 상단의 주석을 제거한 후 다시 앱을 실행합니다.

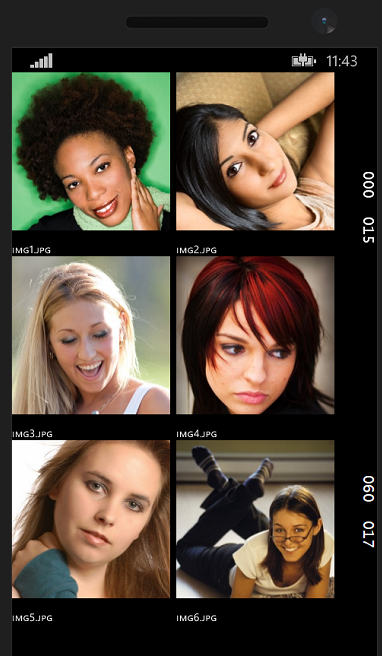


1. 4번과 5번의 과정을 반복합니다. 이번에는 타이머의 시간이 유지되었음을 확인할 수 있습니다. 이와 같은 동작은 아래 코드에 의해 구현되었습니다.

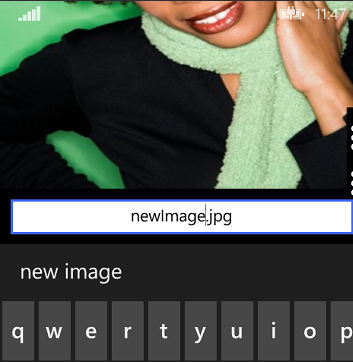


# 네비게이션 상태와 Suspend/Terminate 모드

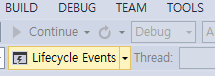
1. Session 7 폴더 > **Demo3** 폴더안의 **PicApp**솔루션 파일을 엽니다.
2. 앱을 실행해 봅니다. 아래와 같이 여섯개의 이미지와 파일명이 나타납니다.



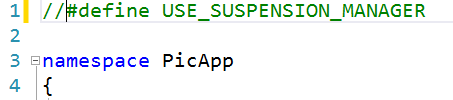
1. 이미지를 선택하면 비주얼 스튜디오에 걸려 있는break point로 이동하게 됩니다. F5버튼을 눌러 계속 실행하면 확대된 이미지와 함께 파일명을 수정할 수 있는 메뉴가 나타납니다.
2. 
3. 아래와 같이 파일명을 변경한 후 뒤로 버튼을 선택하여 변경사항이 적용되었는지 확인합니다.



1. 비주얼 스튜디오 상단의 Lifecycle Events 에서 Suspend & Shutdown 버튼을 선택합니다.



1. F5버튼을 눌러 앱을 다시 실행해 봅니다. 변경사항이 적용되지 않았음을 확인할 수 있습니다.
2. App.xaml.cs 파일 상단의 주석을 제거한 후 다시 앱을 실행합니다.



1. 3~9번의 과정을 반복하여 파일명 변경이 적용되는지를 살펴봅니다.
2. 아래의 코드는 파일명 변경을 가능하게 하는 코드입니다. 하나하나 살펴보도록 하겠습니다.

protected async override void OnLaunched(LaunchActivatedEventArgs e)

{

#if DEBUG

if (System.Diagnostics.Debugger.IsAttached)

{

this.DebugSettings.EnableFrameRateCounter = true;

}

#endif

Frame rootFrame = Window.Current.Content as Frame;

if (rootFrame == null)

{

// Create a Frame to act as the navigation context and navigate to the first page

rootFrame = new Frame();

SuspensionManager.RegisterFrame(rootFrame, "AppFrame"); 🡸 (1)

// Set the default language

rootFrame.Language = Windows.Globalization.ApplicationLanguages.Languages[0];

if (e.PreviousExecutionState == ApplicationExecutionState.Terminated)

{

// Restore the saved session state only when appropriate

try

{

#if USE\_SUSPENSION\_MANAGER

await SuspensionManager.RestoreAsync(); 🡨 (2)

#endif // USE\_SUSPENSION\_MANAGER

}

catch (SuspensionManagerException)

{

//Something went wrong restoring state.

//Assume there is no state and continue

}

}

...

}

async void OnSuspending(object sender, SuspendingEventArgs e)

{

var deferral = e.SuspendingOperation.GetDeferral();

#if USE\_SUSPENSION\_MANAGER

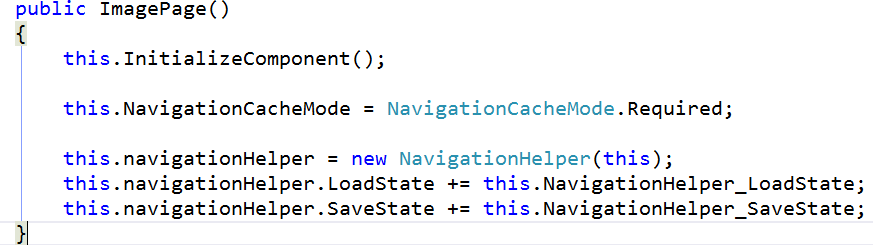
await SuspensionManager.SaveAsync(); 🡨 (3)

#endif // USE\_SUSPENSION\_MANAGER

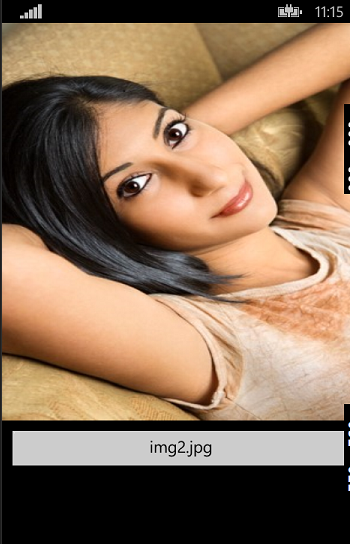
deferral.Complete();

}

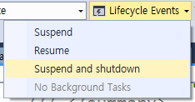
1. (1)번 코드는 기본 프레임인 rootFrame을 SuspensionManager에 등록하여 앱이 일시 중단 되었을 때 모든 페이지 네비게이션을 추적하고 히스토리를 저장할 수 있도록 합니다.
2. (2)번 코드는 앱이 OS에 의해 종료된 후에 다시 실행되었을 때 실행됩니다. 앱이 종료된 페이지에서 다시 실행될 수 있도록 합니다.
3. (3)번 코드는 앱이 일시 중단되었을 때의 상태를 파일에 저장합니다. 이 때 앱은 OS에 의해 종료되는지 유저에 의해 종료되는지 알 수 없습니다만 종료되었던 상태에서 앱이 다시 시작될 수 있습니다.
4. 앱 시나리오에 따라 앞서 살펴본 이 로직은 변경될 필요가 있습니다. 긴 시간동안 앱을 사용하지 않았던 사용자가 방문했을 때는 마지막 페이지에서 앱을 재 시작하는 것보다 MainPage에서 다시 시작하는 것이 좋을 수도 있습니다.
5. 솔루션 탐색기에서 ImagePage.xaml.cs 파일을 더블클릭 한 후 코드를 살펴봅니다. 이 페이지는 이미지파일명을 변경하는 코드가 구현되어 있습니다.
6. 20번째 줄부터 선언되어 있는 페이지의 생성자는 NavigationHelper의 인스턴스를 만들고 NavigationHelper 클래스의 LoadState / SaveState 이벤트 핸들러의 수신자에 ImagePage 클래스의 LoadState / SaveState 메서드를 각각 추가합니다.



1. 앱을 실행한 후 이미지를 클릭해 봅니다.
2. OnNavigatedTo 메서드에 break가 걸릴 것입니다. 이 메서드에서는 Image ID를 넘기고 새로운 ViewModel 인스턴스를 생성한 후 DataItem 객체를 저장합니다. F5 버튼을 눌러 계속 실행합니다.
3. 이번에는 NavigationHelper\_LoadState 메서드에 break가 걸릴 것입니다. F10 버튼을 눌러 한 줄씩 수행해 보면, 현재 저장된 상태가 없으므로 if문 블록을 거치지 않고 지나가게 됩니다.
4. 다시 F5 버튼을 눌러 계속 실행합니다.
5. 아래와 같이 에뮬레이터의 Edit 버튼을 눌러 이미지의 제목을 바꿔봅니다.



1. Lifecycle Events 메뉴의 Suspend and shutdown 메뉴를 선택하여 앱을 종료합니다. 이 때 앱은 OnNavigatedFrom 메서드에서 break가 걸리게 됩니다.



1. F10 버튼을 눌러 OnNavigatedFrom 메서드를 한 줄씩 실행해 봅니다. 뒤로 버튼을 누른것이 아니기 때문에 If문 블록을 거치지 않고 지나가게 됩니다. F5버튼을 눌러 계속 실행합니다.
2. 이번에는 NavigationHelper\_SaveState 메서드에서 break가 걸립니다. 이 메서드에서는 viewModel의 데이터를 편집하고 있는지를 확인합니다. 앞서 이미지의 제목을 바꾸려 했기 때문에 if문의 블록이 수행되고 PageState객체에 바뀐 이미지의 제목이 저장됩니다.
3. 앱이 중지되었을 때 PageState 객체와 Frame의 네비게이션 히스토리는 SuspensionManager에 의해 저장됩니다. SuspensionManager는 DataContractSerializer를 통해 상태 관련 데이터를 직렬화 합니다.
4. 앱을 다시 실행합니다. BreakPoint가 걸려 있으므로 F5를 눌러 앱을 계속 실행하면, 앞서 변경한 내용이 저장되어 있음을 보실수가 있습니다.