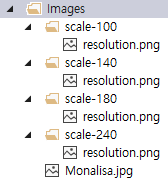
Windows Phone 8.1 App 개발 실습(Adaptive UI)

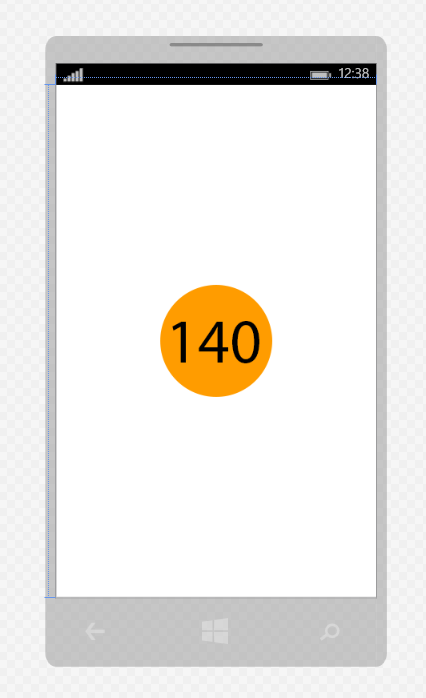
# 이미지 스케일링

이미지의 크기를 화면에 맞게 조절하는법에 대해 알아봅니다.

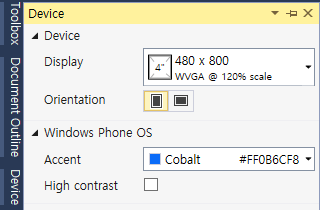
1. Session 6 > **Demo** 폴더안의 **AdaptiveUI**솔루션 파일을 엽니다.
2. 솔루션 탐색기에서 Images 폴더를 확장한 후 각각의 이미지 파일을 더블클릭하여 열어 봅니다. 각기 다른 해상도와 크기의 사진들 입니다.



1. 솔루션 탐색기에서 Pages 폴더를 확장한 후 **Scalefactor.xaml** 파일을 더블클릭하여 엽니다.
2. 코드 편집기 13번째 줄의 Image 컨트롤을 확인합니다. Source 속성에 폴더가 아닌 “/Images/resolution.png” 라는 경로를 지정하고 있습니다. 현재는 디자이너 화면에 ‘140’ scale의 이미지를 보여주고 있습니다.



1. 좌측의 Device tab을 클릭하여 엽니다. 현재 Display 설정이 120% scale로 되어 있습니다.



1. Display 설정을 변경하여 봅니다. Scale이 변경됨에 따라 이미지도 다른 이미지로 바뀌는 것을 보실수가 있습니다.
2. WVGA 4 inch 512MB 에뮬레이터로 변경한 후 앱을 실행해 봅니다. Scale Factor Demo 버튼을 클릭하면 ‘140’ scale의 이미지가 나타날 것입니다.

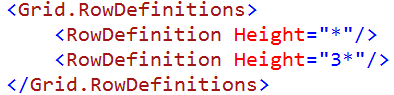


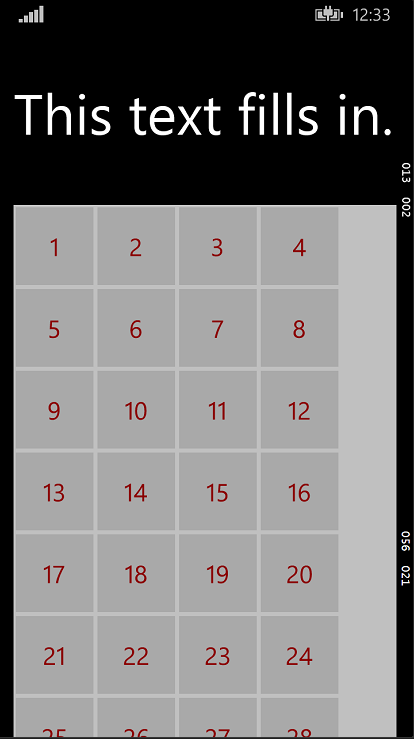
1. 유사하게 720p 4.7 inch 에뮬레이터나 더 큰 화면의 에뮬레이터로 변경한 후 실행하여 봅니다. ‘180’ scale의 이미지가 나타날 것입니다.

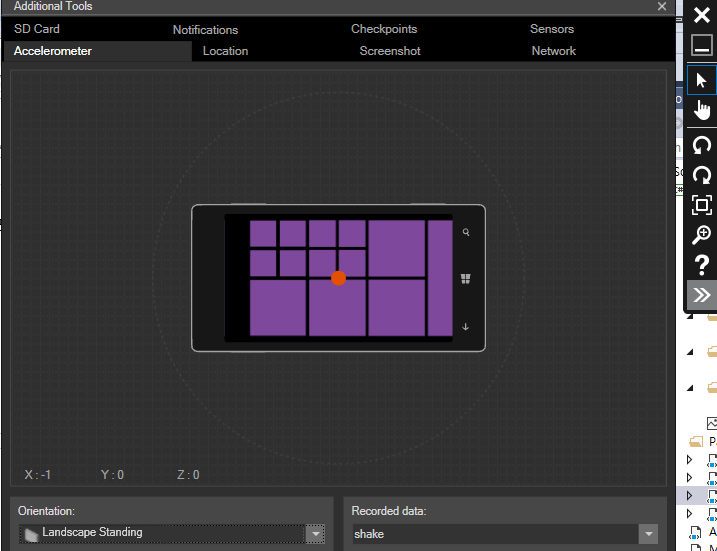
# 반응형 UI

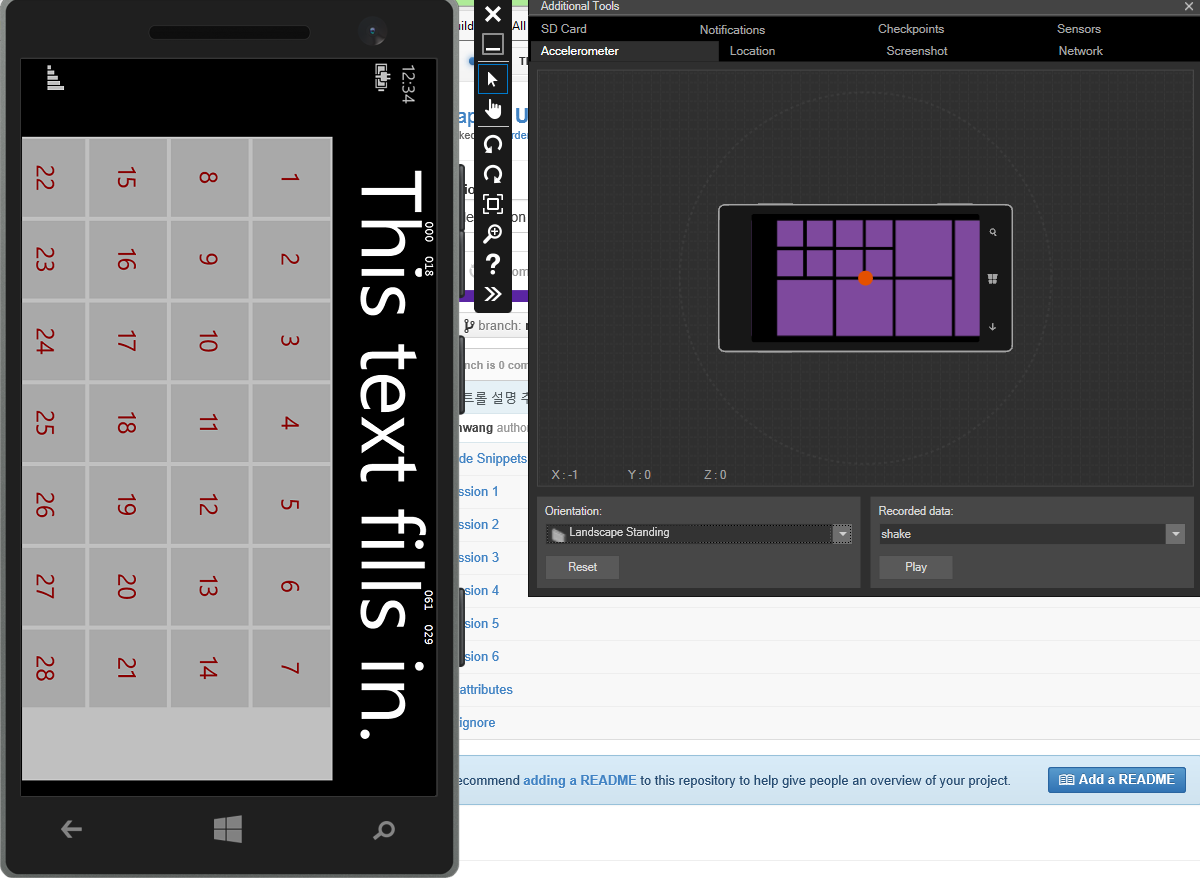
반응형 UI를 구현하는 방법에 대해 살펴봅니다.

1. 앞서 실행되고 있던 앱을 그대로 둡니다.
2. Visual Studio의 솔루션 탐색기의 ResponsiveUI.xaml 파일을 더블클릭하여 엽니다.
3. 코드 편집기의 13번째 줄의 RowDefinition의 높이가 각각 \*, 3\*로 되어있습니다. 이는 화면을 횡으로 1:3의 비율로 분할함을 의미합니다.



1. 16번째 줄의 Viewbox의 경우 Grid.Row 속성이 “0”으로 되어 있습니다. 그리드는 0부터 시작하는 번호로 영역을 구분할 수 있으며, Row 0으로 지정된 Viewbox는 첫번째 행에 위치하게 됩니다. Viewbox에 별다른 속성 값이 없으므로 자동으로 해당 영역을 채웁니다.
2. 두번째 행 Row 1에는 GridView가 지정되어 있으며 DataTemplate을 활용하여 75x75 픽셀 크기의 박스들을 채워넣도록 구현되어 있습니다.
3. 앱이 실행되고 있는 에뮬레이터에서 뒤로가기 버튼을 클릭하여 메인 페이지로 이동한 후, **Responsive UI Demo** 버튼을 선택합니다. 한 행에 4개의 박스들이 채워져 있습니다.  
   
4. 에뮬레이터 우측의 메뉴중 >> 아이콘을 클릭합니다. 클릭하시면 아래와 같이 추가도구들을 사용할 수 있는 창이 나오게 됩니다.



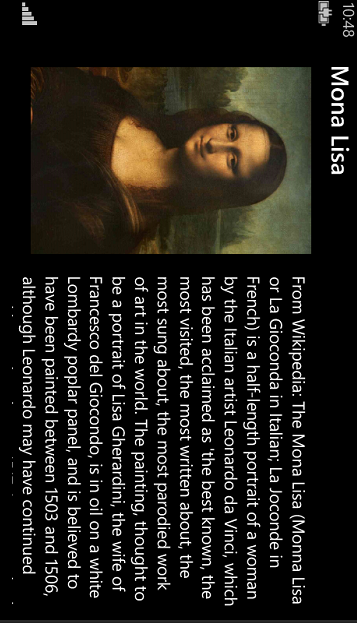
1. 좌측 하단의 메뉴에서 Orientation을 Landscape Standing(가로모드) 로 변경합니다. 에뮬레이터가 회전함을 보실수가 있습니다.  
   

회전된 에뮬레이터를 보시면 상단의 Viewbox는 비율을 유지하면서 상대적으로 텍스트가 커졌고, 박스들의 경우 한 행에 채워지는 박스의 갯수가 변하게 됩니다.

1. 다시 에뮬레이터를 Portrait Standing (세로모드) 로 변경합니다.

# 컨텐츠 오버플로우

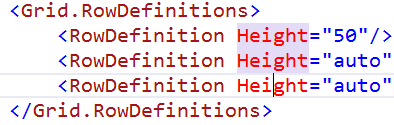
1. 앱이 실행되고 있는 에뮬레이터에서 뒤로가기 버튼을 눌러 메인페이지로 이동합니다.
2. **Content Overflow**버튼을 선택합니다. 에뮬레이터의 세로 모드에서는 모나리자 이미지만 나타납니다.
3. 에뮬레이터를 다시 Landscape Standing(가로모드) 로 변경합니다. 모나리자 이미지와 설명이 나타납니다.



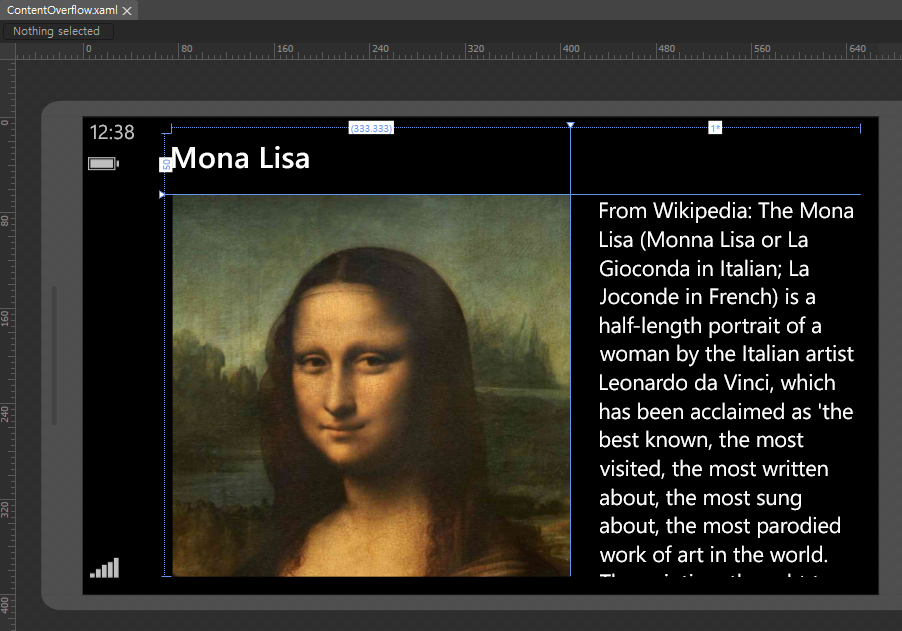
1. 솔루션 탐색기의 ContentOverflow.xaml 파일을 확장한 후 ContentOverflow.xaml.cs 파일을 엽니다.



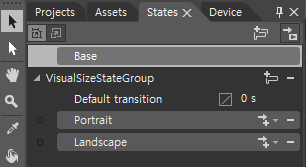
1. 코드 편집기 26번째 줄의 ContentOverflow() 생성자를 보면 VisualStateManager를 “Portrait” 로 설정하고 있습니다. 또한, Orientation이 변경되는 이벤트가 발생할 때 마다 DisplayInfoOrientationChanged 이벤트 핸들러가 VisualStateManager의 값을 “Portrait”와 “Landscape” 를 변경해 줍니다. (34번째 줄 이후로 자세한 이벤트 핸들러의 구현을 보실수 있습니다.)  
   Visual State는 컨텐츠를 여러 상태에 따라서 다르게 표시하기 위해 개발자가 정의할 수 있는 UI 상태입니다. Orientation의 가로/세로 상태에 맞는 UI를 표시하기 위해서 이 예제에서는 “Portrait”와 “Landscape” 상태를 만들어서 정의했습니다. 상태의 이름만 “Portrait”, “Landscape”일 뿐, Orientation의 가로/세로 모드에 연동되는 것은 아니기 때문에 Orientation 변경 이벤트가 발생하였을 때 직접 VisualStateManager로 변경해 주어야 합니다.
2. ContentOverflow.xaml 파일을 더블클릭하여 열어 어떻게 구성되어 있는지 내용을 살펴봅니다.
3. 코드 편집기의 38번째 줄로 이동합니다. Mona Lisa 텍스트와 이미지가 포함되어 있는 Grid 컨트롤의 행 선언 부분입니다. 아래와 같이 첫 번째 행만 50으로 높이가 설정되어 있고, 두번째, 세번째 행의 높이는 자동으로 되어 있습니다.



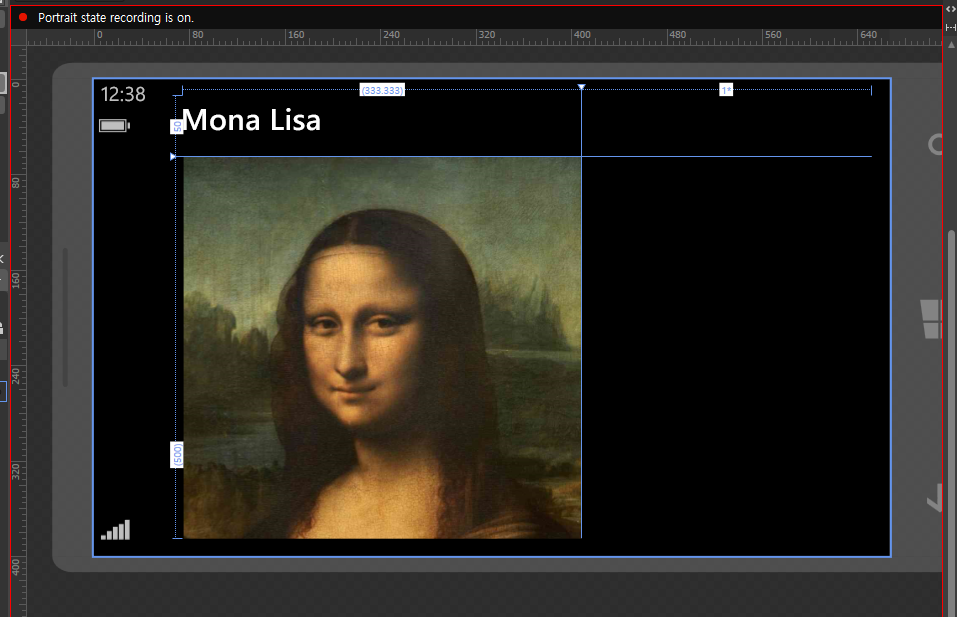
1. Blend를 통해 Visual State가 어떤 식으로 구현이 되어있는지 간단히 알아봅니다. ContentOverflow.xaml 파일을 마우스 우클릭 한후 “Open in Blend…” 메뉴를 선택합니다.
2. 페이지가 아래와 같이 가로모드로 열리고 모든 컨텐츠를 보여줍니다.



1. 좌측의 States 탭을 선택합니다. XAML에 정의해 놓은 Portrait와 Landscape 두 가지 Visual State를 확인할 수 있습니다.



1. Portrait 상태를 선택하면 레코딩이 되고 있다는 표시가 나오는데, 이 때 UI를 수정하면 상태가 적용되었을 때 반영이 됩니다. 화면 왼쪽의 Objects and Timeline창에서 MonaLisaDetails를 선택하고 Properties 창을 확인해 보면 Visibility 속성이 Collapsed로 되어 있어 화면에서 숨겨져 있음을 확인할 수 있습니다.

  
따라서, Orientation이 세로 모드일 때, VisualStateManaer에 Portrait 상태를 적용하고, MonaLisaDetails은 화면에서 숨겨집니다.

1. States 창에서 Landscape를 선택하고, MonaLisaDetails의 Visibility 속성이 Visible로 바뀐 것을 확인합니다.
2. 저장 후 Blend를 종료하면 알림메시지와 함께 변경내용을 저장하게 됩니다. 변경내용은 ContentOverflow.xaml에 반영됩니다.
3. 애플리케이션을 종료합니다.

# 디스플레이 정보

1. 솔루션 탐색기에서 DisplayInfo.xaml.cs 파일을 더블클릭하여 엽니다. 코드내용을 간략히 살펴봅니다.
2. 애플리케이션을 실행한 후 DisplayInfo 버튼을 선택합니다. 현재 디스플레이의 정보를 보여줍니다.
3. 중간의 RawPixelsPerViewPixel 값을 확인합니다. 이 값은 하나의 ‘view pixel’ 에 대응하는 실제 픽셀의 수를 나타냅니다. 720p 4.7인치의 경우 1.8이 됩니다.
4. WVGA 4 inch 512MB 에뮬레이터에서 다시 앱을 실행해 봅니다. RawPixelsPerViewPixel 값이 1.2가 됩니다.