Control

WPF에서 "Control"이란 창이나 페이지에서 호스팅되고 UI(사용자 인터페이스)가 있으며 몇몇 동작을 구현하는 WPF 클래스의 범주를 포괄적으로 지칭하는 용어이다. 사용자와 상호작용하는 모든 사용자 인터페이스 요소를 말하기도 한다. 윈도우 프로그램 (C#이나 MFC 등)에서는 컨트롤에 사용자의 입력을 처리할 때 하나하나의 이벤트를 생성해 줘야 하지만 WPF에서는 WPF의 기본으로 포함되어 있는 객체를 사용하여 좀더 직관적인 코딩을 할 수 있는 장점이 있다. WPF의 Control 개념에서는 하나의 객체에 여러가지 프로퍼티를 사용하여 같은 객체에서 각각 다른 결과를 얻어 낼 수 있다. 컨트롤은 대개 사용자 입력을 감지하여 이에 응답한다. WPF 입력 시스템에서는 직접 이벤트와 라우트된 이벤트를 모두 사용하여 텍스트 입력, 포커스 관리 및 마우스 위치 지정을 지원한다. 응용 프로그램에는 종종 복잡한 입력 요구 사항도 있다. WPF에서는 사용자 입력 작업과 이러한 작업에 응답하는 코드를 구별하는 명령 시스템을 제공한다.

**표준 WPF 컨트롤** (표준 WPF컨트롤은 다음과 같다.)

|  |  |
| --- | --- |
| **컨트롤 이름** | **설명** |
| **System.Windows.Controls.Border** | 콘텐츠 주위에 테두리를 표시합니다. |
| **System.Windows.Controls.Button** | 사용자가 단추를 클릭하여 작업을 수행할 수 있습니다. Button을 클릭하면 Buttonbase.Click 이벤트가 발생합니다. |
| **System.Windows.Controls.CheckBox** | Yes/No 또는 True/False 값을 나타내는 확인란을 사용자가 선택하거나 선택 취소할 수 있습니다. |
| **System.Windows.Controls.ComboBox** | 사용자가 드롭다운 목록에서 항목을 선택할 수 있습니다. 목록은 사용자가 드롭다운 화살표를 클릭하면 표시됩니다. |
| **System.Windows.Controls.Grid** | 열과 행으로 구성되는 유연한 표 영역을 정의합니다. |
| **System.Windows.Controls.Image** | 이미지를 표시합니다. |
| **System.Windows.Controls.Label** | 폼에 텍스트를 표시합니다. 이 컨트롤은 선택키를 지원합니다. |
| **System.Windows.Controls.ListBox** | 사용자가 목록에서 항목을 선택할 수 있습니다. |
| **System.Windows.Controls.RadioButton** | 사용자가 함께 사용할 수 없는 항목 중에서 선택할 수 있습니다. 같은 컨테이너에 포함된 라디오 단추 중에서는 라디오 단추 하나만 선택할 수 있습니다. |
| **System.Windows.Controls.StackPanel** | 자식 컨트롤을 가로 또는 세로로 스택 방식으로 배치할 수 있습니다. |
| **System.Windows.Control.TabControl** | 시각적 콘텐츠를 표 형식으로 정렬할 수 있습니다. |
| **System.Windows.Controls.TextBox** | 서식 없는 텍스트를 표시하고 사용자가 텍스트를 입력할 수 있습니다. |

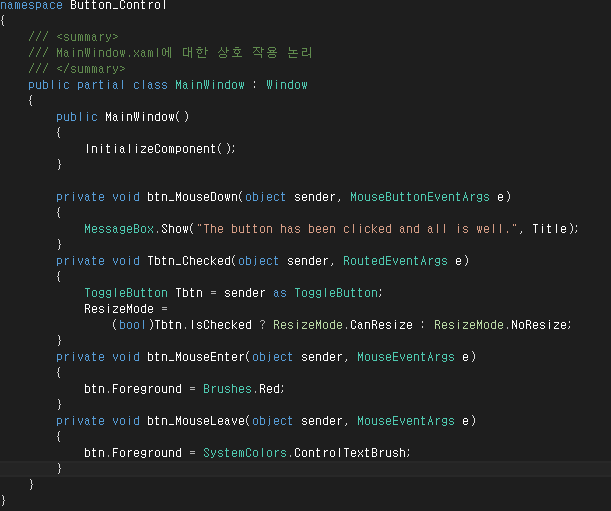
WPF에서 가장 기본적인 컨트롤은 Button이다. 버튼은 ButtonBase를 기반 클래스로 파생되며 ButtonBase 클래스는 ContentControl에서 파생되므로 모두 Content모델을 지원한다. 이것은 Windows의 Contents에 대입 할 수 있는 모든 객체는 버튼의 Content로 사용될 수 있으며 Button을 비트맵, Shape객체, 서식이 있는 텍스트로 표시하거나 Button 객체의 Content에 다른 객체를 연결할 수 있다. WPF에서 ButtonBase 클래스는 Button, RepeatButton, ToggleButton와 같은 컨트롤들로 파생된다.

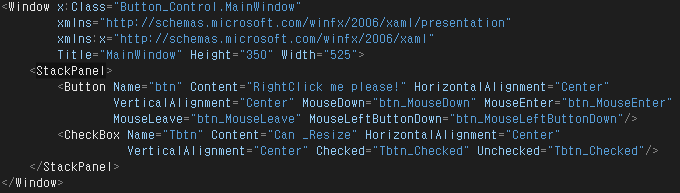
**ButtonBase 클래스**

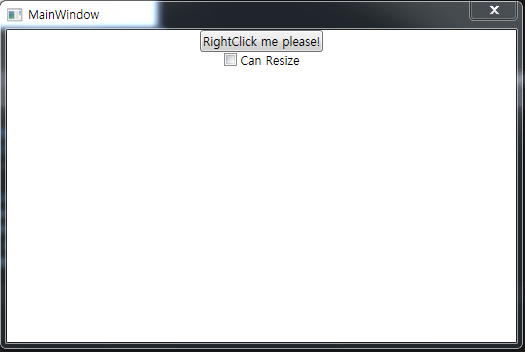
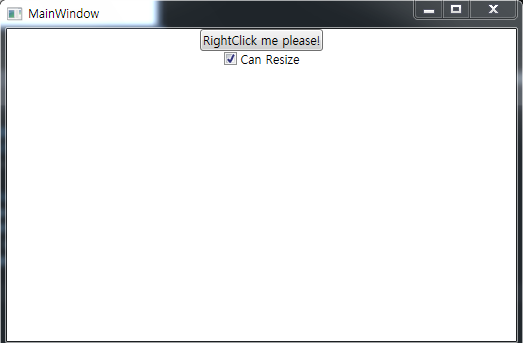
어떤 부모 클래스들과 마찬가지로 ButtonBase 클래스는 파생한 컨트롤들에게 여러 가지 모양의 인터페이스를 제공하고 있다. 또한 Click 이벤트를 정의하고 있고 IsPressed라는 속성 또한 정의하고 있다. 이 속성은 버튼이 파생된 버튼이 눌려진 적이 있는지를 검사하기 위해서 사용한다. 이 밖에도 다음 표에서는 ButtonBase에서 제공하고 있는 멤버들에 대해서 설명하고 있다.

|  |  |
| --- | --- |
| **속성** | **뜻** |
| **ClickMode** | 이 속성은 Click 이벤트를 언제 발생시킬지를 결정하고 Enum형으로 선언되어있다. |
| **Command** | 앞장에서 설명했던 것처럼 많은 UI요소들은 특정 명령을 가질 수 있고 이 명령은 Command 속성을 이용해서 할당하게 된다. |
| **CommandParameter** | 이 속성은 Command 속성에 지정된 파라미터를 전달하기 위한 속성이다. |
| **CommandTarget** | 이 속성은 Command 속성에 설정한 명령을 받을 대상을 지정한다. |

다음 예제소스는 Button Right Click과 CheckBox 이벤트에 대한 예제소스이다. Button Right Click은 메시지박스를 띄우는 이벤트를 알 수 있고 CheckBox는 ToggleButton과 ResizeMode의 관계를 알 수 있다.

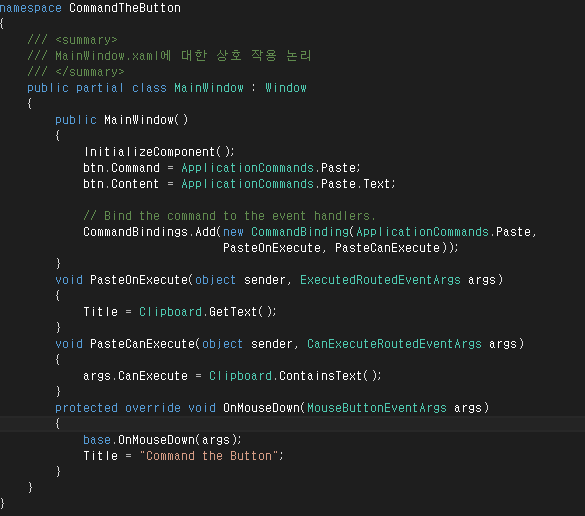
**<C# 소스코드>**

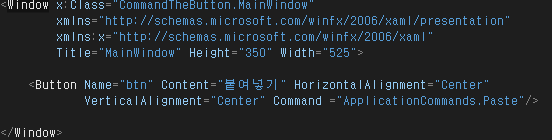
**<XAML소스코드>**

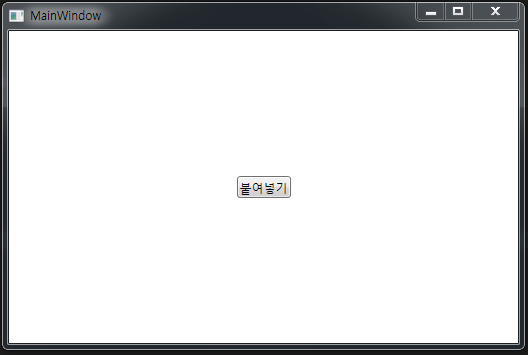
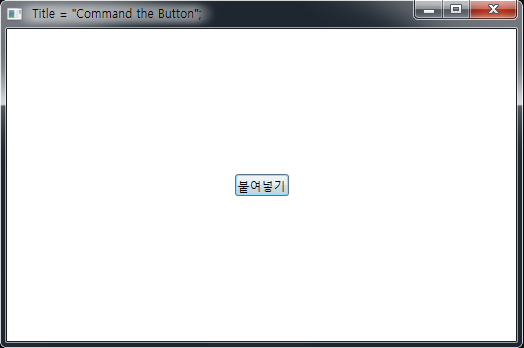
 **<실행화면1> <실행화면2>**

<실행화면1>에서 볼 수 있듯이 CheckBox에 Check가 안돼있을 경우에는 ResizeMode 가 윈도우크기를 설정하지 못하게 막고 <실행화면2>에서 볼 수 있듯이 CheckBox에 Check가 ToggleButton으로 고정되어 있다면 윈도우 크기를 설정할 수 있게 되는 것을 볼 수 있다.

다음 예제는 윈도우 타이틀에 Clipboard 텍스트를 출력하는 예제이다. Command 명령을 바인딩하여 윈도우타이틀에 출력하게 된다. Command 명령으로 Button의 초기화를 비활성화시킨 예제소스이다.

**<C# 소스코드>**

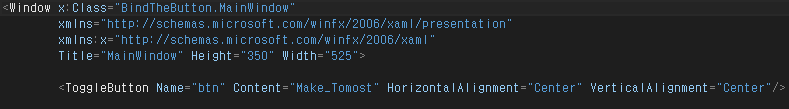
**<XAML 소스코드>**

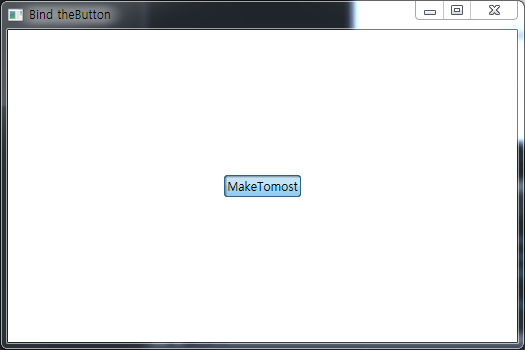
**  <실행화면1> <실행화면2>**

<실행화면1>과 같은 화면이 나오려면 일단 텍스트를 복사하자. 복사하면 <실행화면1>처럼 붙여넣기 Button이 활성화 된 것을 확인할 수 있다. 붙여넣기 Button을 누르게 되면 <실행화면2>에서와 같이 윈도우 타이틀 바에 복사한 텍스트가 출력 되어 보여지는 것을 확인 할 수 있다.

다음 소스 예제는 ToggleButton과 SetBinding을 이용하여 Button 클릭 시 실행화면이 맨 위 화면으로 고정되게 하는 예제소스이다.

**<C# 소스코드>**

**<XAML소스코드>**

**<실행화면>**

XAML의 소스코드를 보면ToggleButton을 지정하여 버튼이 눌린상태를 유지하게 하고, 유지한 상태를 C#소스코드의 btn.SetBinding을 통해 화면을 맨위에 고정 시켜 다른 화면을 띄우더라도 맨 위에 실행화면이 위치하는 것을 확인 할 수 있다.