Contents

Windows Presentation Foundation

2019-10-26 • 2 minutes to read • Edit Online

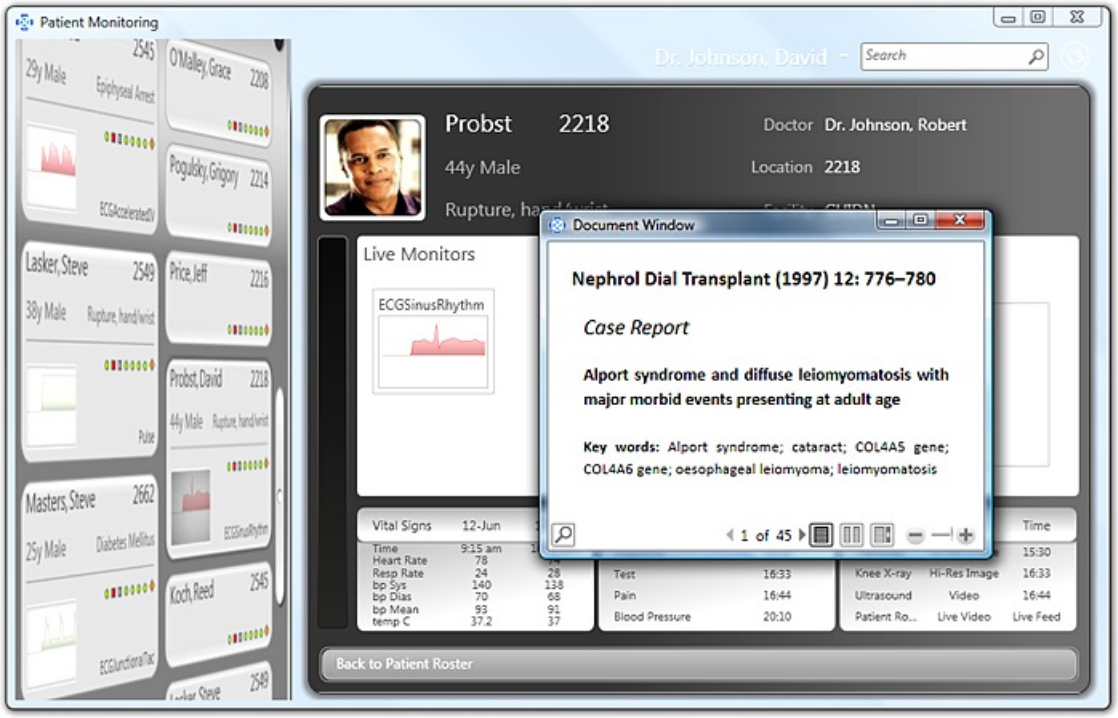
WPF (Windows Presentation Foundation)는 Windows에서 lob (기간 업무) 데스크톱 응용 프로그램을 빌드하기 위한 통합 프로그래밍 모델을 개발자에 게 제공 합니다.

- WPF 소개
- 시작
- 애플리케이션 개발
- 고급
- 컨트롤
- Data
- 그래픽 및 멀티미디어
- Security
- WPF 샘플
- 클래스 라이브러리

WPF 개요

2020-03-21 • 63 minutes to read • Edit Online

WPF(Windows Presentation Foundation)를 사용하면 시각적으로 뛰어난 사용자 환경을 통해 Windows용 데스크톱 클라이언트 애플리케이션을 만들 수 있습니다.



WPF의 핵심은 최신 그래픽 하드웨어를 활용하도록 작성된 해상도 독립적인 벡터 기반 렌더링 엔진입니다. WPF는 XAML(Extensible Application Markup Language), 컨트롤, 데이터 바인딩, 레이아웃, 2D 및 3D 그래픽, 애니메이션, 스타일, 템플릿, 문서, 미디어, 텍스트 및 입력 체계를 포함하는 포괄적인 애플리케이션 개발 기능을 사용하여 핵심을 확장합니다. WPF는 .NET의 일부이므로, .NET API의 다른 요소를 통합하는 애플리케이션을 빌드할 수 있습니다.

이 개요는 초보자를 위한 것이며 WPF의 주요 기능 및 개념에 대해 설명합니다.

WPF로 프로그램

WPF는 대부분 System.Windows 네임스페이스에 있는 .NET 형식의 하위 집합으로 존재합니다. 이전에 ASP.NET 및 Windows Forms와 같은 관리되는 기술을 사용하여 .NET으로 애플리케이션을 빌드한 적이 있는 경우 기본적인 WPF 프로그래밍 환경이 친숙할 것입니다. C# 또는 Visual Basic과 같은 원하는 .NET 프로그래밍 언어를 사용하여 클래스를 인스턴스화하고, 속성을 설정하고, 메서드를 호출하고, 이벤트를 처리합니다.

WPF에는 속성 및 이벤트를 향상시키는 종속성 속성 및 라우트된 이벤트와 같은 추가 프로그래밍 구문이 포함되어 있습니다.

태그 및 코드 숨김

WPF를 사용하면 ASP.NET 개발자에게 익숙한 환경인 *태그* 및 *코드 숨김* 둘 다를 통해 애플리케이션을 개발할 수 있습니다. 일반적으로 XAML 태그를 사용하여 애플리케이션의 모양을 구현하고 관리되는 프로그래밍 언어(코드 숨김)를 사용하여 해당 동작을 구현합니다. 모양 및 동작의 이러한 분리는 다음과 같은 이점이 있습니다.

- 모양 관련 태그가 동작 관련 코드와 밀접하게 결합되지 않으므로 개발 및 유지 관리 비용이 줄어듭니다.
- 디자이너가 애플리케이션의 동작을 구현하는 개발자와 동시에 애플리케이션의 모양을 구현할 수 있으므로

개발이 보다 효율적입니다.

- WPF 애플리케이션에 대한전역화 및 지역화 가 간소화됩니다.

태그

XAML은 선언적으로 애플리케이션의 모양을 구현하는 XML 기반 태그 언어입니다. 일반적으로 창, 대화 상자, 페이지 및 사용자 정의 컨트롤을 만들고 컨트롤, 도형 및 그래픽으로 채우는 데 사용됩니다.

다음 예제에서는 XAML을 사용하여 단일 단추가 포함된 창의 모양을 구현합니다.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Title="Window with Button"
    Width="250" Height="100">

  <!-- Add button to window -->
  <Button Name="button">Click Me!</Button>

</Window>
```

특히, 이 XAML은 각각 `Window` 및 `Button` 요소를 사용하여 창과 단추를 정의합니다. 각 요소는 창의 제목 표시줄 텍스트를 지정하는 `Window` 요소의 `Title` 특성과 같은 특성으로 구성됩니다. 런타임에 WPF는 태그에 정의된 요소와 특성을 WPF 클래스 인스턴스로 변환합니다. 예를 들어 `Window` 요소는 Title 속성이 `Title` 특성의 값인 Window 클래스 인스턴스로 변환됩니다.

다음 그림은 이전 예제의 XAML에 의해 정의된 사용자 인터페이스(UI)를 보여 줍니다.

XAML은 XML 기반이기 때문에 XAML로 작성한 UI는 요소 트리라고 하는 중첩된 요소 계층 구조로 어셈블됩니다. 요소 트리는 UI를 만들고 관리하는 논리적이고 직관적인 방법을 제공합니다.

코드 숨김

애플리케이션의 기본 동작은 이벤트 처리(예: 메뉴, 도구 모음 또는 단추 클릭) 및 응답으로 비즈니스 논리 및 데이터 액세스 논리 호출을 포함하여 사용자 조작에 응답하는 기능을 구현하는 것입니다. WPF에서 이 동작은 태그와 연결된 코드에서 구현됩니다. 이러한 종류의 코드를 코드 숨김이라고 합니다. 다음 예제에서는 이전 예제의 업데이트된 태그와 코드 숨결을 보여 주며 다음과 같은 코드 숨결을 보여 주며 다음과 같은 예제를 보여 주며 다음과 같은 예제를 보여 주며 다음과 같은 경우를 보여 주며 다음과 같은

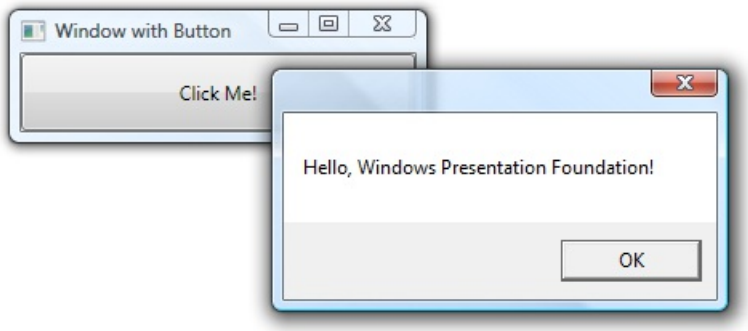
```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.AWindow"
    Title="Window with Button"
    Width="250" Height="100">

  <!-- Add button to window -->
  <Button Name="button" Click="button_Click">Click Me!</Button>

</Window>
```

```
using System.Windows; // Window, RoutedEventArgs, MessageBox

namespace SDKSample
{
    public partial class AWindow : Window
    {
        public AWindow()
        {
            // InitializeComponent call is required to merge the UI
            // that is defined in markup with this class, including
            // setting properties and registering event handlers
            InitializeComponent();
        }

        void button_Click(object sender, RoutedEventArgs e)
        {
            // Show message box when button is clicked.
            MessageBox.Show("Hello, Windows Presentation Foundation!");
        }
    }
}
```

```
Namespace SDKSample

    Partial Public Class AWindow
        Inherits System.Windows.Window

        Public Sub New()

            ' InitializeComponent call is required to merge the UI
            ' that is defined in markup with this class, including
            ' setting properties and registering event handlers
            InitializeComponent()

        End Sub

        Private Sub button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)

            ' Show message box when button is clicked.
            MessageBox.Show("Hello, Windows Presentation Foundation!")

        End Sub

    End Class

End Namespace
```

이 예제에서 코드 숨김은 Window 클래스에서 파생된 클래스를 구현합니다. `x:Class` 특성은 태그를 코드 숨김 클래스와 연결하는 데 사용됩니다. `InitializeComponent`는 코드 숨김 클래스를 사용하여 태그에서 정의된 UI를 병합하기 위해 코드 숨김 클래스의 생성자에서 호출됩니다. (응용 `InitializeComponent` 프로그램이 빌드될 때 생성되기 때문에 수동으로 구현할 필요가 없습니다.) 구현이 `x:Class` 생성될 때마다 구현이 올바르게 초기화되었는지 의 조합및 `InitializeComponent` 확인합니다. 코드 숨김 클래스는 단추의 Click 이벤트에 대한 이벤트 처리기도 구현합니다. 단추를 클릭하면 이벤트 처리기가 System.Windows.MessageBox.Show 메서드를 호출하여 메시지 상자를 표시합니다.

다음 그림은 단추를 클릭할 때 결과를 보여 주며,



컨트롤

애플리케이션 모델에서 제공하는 사용자 환경은 생성된 컨트롤입니다. WPF에서 *컨트롤은* 창이나 페이지에서 호스팅되고 사용자 인터페이스가 있고 일부 동작을 구현하는 WPF 클래스 범주에 적용되는 우산 용어입니다.

자세한 내용은 컨트롤을 참조하십시오.

기능별 **WPF** 컨트롤

기본 제공 WPF 컨트롤은 다음과 같습니다.

- **버튼**: Button RepeatButton및 .
- **데이터** 표시 DataGrid ListView: TreeView, 및 .
- **날짜 표시** 및 Calendar DatePicker선택 : 및 .
- **대화 상자**: OpenFileDialog PrintDialog, SaveFileDialog및 .
- **디지털**잉크 InkCanvas InkPresenter: 및 .
- **문서** DocumentViewer: FlowDocumentPageViewer FlowDocumentReader, FlowDocumentScrollViewer, StickyNoteControl및 .
- **입력** TextBox: RichTextBox, PasswordBox및 .
- **레이아웃** Border: BulletDecorator Canvas' DockPanel Expander, Grid GridView' GridSplitter GroupBox, Panel ResizeGrip' Separator ScrollBar, ScrollViewer StackPanel' Thumb, , , , , , , Viewbox, VirtualizingStackPanel Window WrapPanel, , , , , , , , , ,
- **미디어** Image: MediaElement, SoundPlayerAction및 .
- **메뉴**: ContextMenu Menu, ToolBar및 .
- **탐색** Frame: Hyperlink Page, NavigationWindow, TabControl및 .
- **선택** CheckBox: ComboBox ListBox, RadioButton, Slider, 및 .
- **사용자**정보 AccessText Label: Popup ProgressBar, StatusBar TextBlock, ToolTip, ,

입력 및 명령

컨트롤은 대체로 사용자 입력을 감지하고 응답합니다. WPF 입력 시스템 은 직접 및 라우트된 이벤트를 사용하여 텍스트 입력, 포커스 관리 및 마우스 위치 지정을 지원합니다.

애플리케이션에 복잡한 입력 요구 사항이 있는 경우가 많습니다. WPF는 사용자 입력 작업을 이러한 작업에 응답하는 코드에서 분리하는 명령 시스템을 제공합니다.

레이아웃

사용자 인터페이스를 만들 때 위치 및 크기별로 컨트롤을 정렬하여 레이아웃을 구성합니다. 모든 레이아웃의 주요

요구 사항은 창 크기와 표시 설정의 변경 내용에 맞게 조정되는 것입니다. 이러한 상황에서 레이아웃을 조정하는 코드를 작성하도록 요구하는 대신 WPF는 확장 가능한 뛰어난 레이아웃 시스템을 제공합니다.

이 레이아웃 시스템의 토대는 상대 위치 지정으로, 변화하는 창과 표시 조건에 맞게 조정하는 기능을 향상시킵니다. 또한 이 레이아웃 시스템은 레이아웃을 결정하기 위한 컨트롤 간의 협상을 관리합니다. 협상은 2단계 프로세스입니다. 첫 번째로, 컨트롤이 필요한 위치 및 크기를 부모에게 알립니다. 두 번째로, 부모가 사용할 수 있는 공간을 컨트롤에 알립니다.

이 레이아웃 시스템은 기본 WPF 클래스를 통해 자식 컨트롤에 노출됩니다. 그리드, 스택 및 도킹과 같은 일반적인 레이아웃을 위해 WPF는 여러 레이아웃 컨트롤을 포함합니다.

- Canvas: 자식 컨트롤이 자체 레이아웃을 제공합니다.
- DockPanel: 자식 컨트롤이 패널 가장자리에 맞춰집니다.
- Grid: 자식 컨트롤이 행 및 열별로 배치됩니다.
- StackPanel: 자식 컨트롤이 가로 또는 세로로 포개집니다.
- VirtualizingStackPanel: 자식 컨트롤이 가상화되고 가로 또는 세로 방향인 한 줄에 정렬됩니다.
- WrapPanel: 자식 컨트롤이 왼쪽에서 오른쪽 순서로 배치되고 공간에 허용되는 것보다 더 많은 컨트롤이 현재 줄에 있는 경우 다음 줄로 줄 바꿈됩니다.

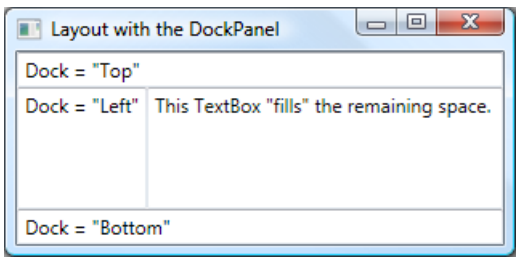
다음 예제에서는 DockPanel a를 사용하여 TextBox 몇 가지 컨트롤을 배치합니다.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.LayoutWindow"
    Title="Layout with the DockPanel" Height="143" Width="319">

  <!--DockPanel to layout four text boxes-->
  <DockPanel>
    <TextBox DockPanel.Dock="Top">Dock = "Top"</TextBox>
    <TextBox DockPanel.Dock="Bottom">Dock = "Bottom"</TextBox>
    <TextBox DockPanel.Dock="Left">Dock = "Left"</TextBox>
    <TextBox Background="White">This TextBox "fills" the remaining space.</TextBox>
  </DockPanel>

</Window>
```

DockPanel 을 사용하면 자식 TextBox 컨트롤이 정렬 방법을 알려줄 수 있습니다. 이렇게 하려면 DockPanel은 각각 도킹 스타일을 지정할 수 있도록 자식 컨트롤에 노출되는 `Dock` 속성을 구현합니다.

> **NOTE**
> 자식 컨트롤에서 사용할 수 있도록 부모 컨트롤이 구현하는 속성은 연결된 속성이라는 WPF 구문입니다.

다음 그림은 앞의 예제에서 XAML 태그의 결과를 보여 주며 있습니다.



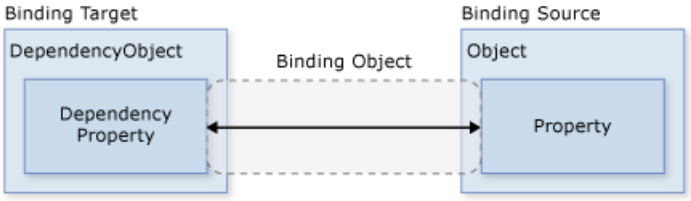
데이터 바인딩

대부분의 애플리케이션은 데이터를 보고 편집할 수 있는 수단을 사용자에게 제공하기 위해 생성됩니다. WPF 애플리케이션의 경우 데이터를 저장 및 액세스하는 작업이 SQL Server 및 ADO .NET과 같은 기술에 의해 이미 제공됩니다. 데이터에 액세스하고 애플리케이션의 관리되는 개체에 로드한 후 WPF 애플리케이션에 대한 힘든 작업이 시작됩니다. 기본적으로 다음 두 가지가 포함됩니다.

1. 관리되는 개체에서 데이터를 표시 및 편집할 수 있는 컨트롤로 데이터 복사

2. 컨트롤을 사용한 데이터 변경 내용이 관리되는 개체에 다시 복사되는지 확인

애플리케이션 개발을 간소화하기 위해 WPF는 이러한 단계를 자동으로 수행하는 데이터 바인딩 엔진을 제공합니다. 데이터 바인딩 엔진의 핵심 단위는 Binding 클래스로, 컨트롤(바인딩 대상)을 데이터 개체(바인딩 소스)에 바인딩하는 작업을 수행합니다. 이 관계는 다음 그림에 나와 있습니다.



다음 예제에서는 TextBox를 사용자 지정 `Person` 개체의 인스턴스에 바인딩하는 방법을 보여 줍니다. `Person` 구현은 다음 코드에 나와 있습니다.

```
Namespace SDKSample

    Class Person

        Private _name As String = "No Name"

        Public Property Name() As String
            Get
                Return _name
            End Get
            Set(ByVal value As String)
                _name = value
            End Set
        End Property

    End Class

End Namespace
```

```
namespace SDKSample
{
    class Person
    {
        string name = "No Name";

        public string Name
        {
            get { return name; }
            set { name = value; }
        }
    }
}
```

다음 태그는 사용자 TextBox 지정 `Person` 개체의 인스턴스에 바인딩합니다.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.DataBindingWindow">

  <!-- Bind the TextBox to the data source (TextBox.Text to Person.Name) -->
  <TextBox Name="personNameTextBox" Text="{Binding Path=Name}" />

</Window>
```

```
Imports System.Windows ' Window

Namespace SDKSample

    Partial Public Class DataBindingWindow
        Inherits Window

        Public Sub New()
            InitializeComponent()

            ' Create Person data source
            Dim person As Person = New Person()

            ' Make data source available for binding
            Me.DataContext = person

        End Sub

    End Class

End Namespace
```

```
using System.Windows; // Window

namespace SDKSample
{
    public partial class DataBindingWindow : Window
    {
        public DataBindingWindow()
        {
            InitializeComponent();

            // Create Person data source
            Person person = new Person();

            // Make data source available for binding
            this.DataContext = person;
        }
    }
}
```

이 예제에서 `Person` 클래스는 코드 숨김에서 인스턴스화되고 `DataBindingWindow`에 대한 데이터 컨텍스트로 설정됩니다. 태그에서 TextBox의 Text 속성은 "`{Binding ... }`" XAML 구문을 사용하는 `Person.Name` 속성에 바인딩됩니다. 이 XAML은 WPF가 TextBox 컨트롤을 해당 창의 DataContext 속성에 저장된 `Person` 개체에 바인딩되도록 합니다.

WPF 데이터 바인딩 엔진은 유효성 검사, 정렬, 필터링 및 그룹화를 포함하는 추가 지원을 제공합니다. 또한 데이터 바인딩은 표준 WPF 컨트롤에 의해 표시되는 사용자 인터페이스가 적절하지 않은 경우 데이터 템플릿을 사용하여 바인딩된 데이터에 대한 사용자 지정 사용자 인터페이스를 만들 수 있도록 지원합니다.

자세한 내용은 데이터 바인딩 개요를참조하십시오.

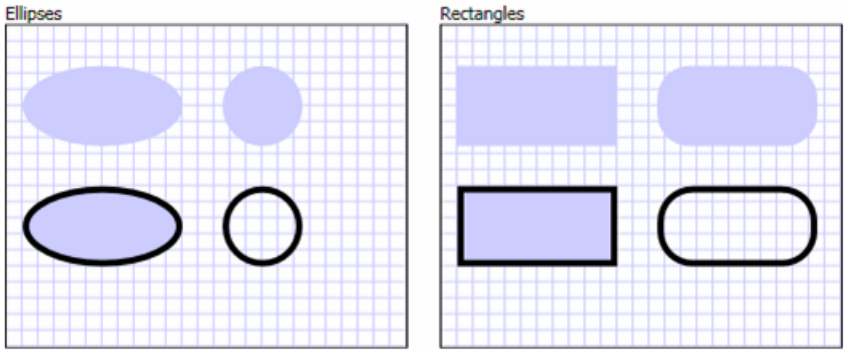
그래픽

WPF는 다음과 같은 이점이 있는 광범위하고 확장 가능하며 유연한 그래픽 집합을 도입합니다.

- **해상도 및 디바이스 독립적인 그래픽**. WPF 그래픽 시스템의 기본 측정 단위는 실제 화면 해상도에 관계 없이 1/96인치인 디바이스 독립적 픽셀이며, 해상도 및 디바이스 독립적인 렌더링을 위한 토대를 제공합니다. 각 디바이스 독립적 픽셀은 렌더링되는 시스템의 dpi(인치당 도트 수) 설정에 맞게 자동으로 확장됩니다.
- **향상된 정밀도**. WPF 좌표계는 단정밀도 대신 배정밀도 부동 소수점 숫자로 측정됩니다. 변환 및 불투명도 값도 배정밀도로 표현됩니다. 또한 WPF는 광범위한 색 영역(scRGB)을 지원하며 여러 색 공간의 입력을 관리하기 위한 통합 지원을 제공합니다.
- **고급 그래픽 및 애니메이션 지원**. WPF는 애니메이션 장면을 관리하여 그래픽 프로그래밍을 간소화합니다. 장면 처리, 렌더링 루프 및 쌍선형 보간에 대해 걱정할 필요가 없습니다. 또한 WPF는 적중 테스트 및 전체 알파 합치기를 지원합니다.
- **하드웨어 가속**. WPF 그래픽 시스템은 그래픽 하드웨어를 활용하여 CPU 사용량을 최소화합니다.

2D 도형

WPF는 다음 그림에 표시된 사각형 및 타원과 같은 일반적인 벡터 기반의 2D 도형 라이브러리를 제공합니다.



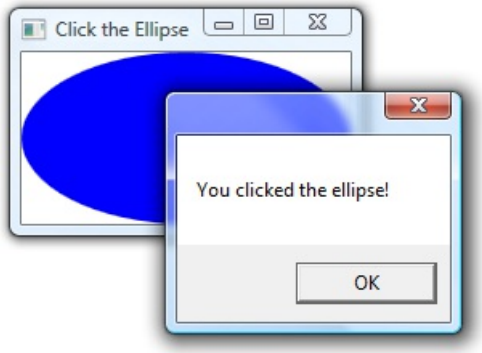
도형의 흥미로운 기능은 표시에 사용되는 것뿐 아니라 키보드 및 마우스 입력을 비롯하여 컨트롤에서 기대하는 기능을 대부분 구현한다는 것입니다. 다음 예제에서는 MouseUp 처리 Ellipse 되는 이벤트를 보여 주면 됩니다.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.EllipseEventHandlingWindow"
    Title="Click the Ellipse">
    <Ellipse Name="clickableEllipse" Fill="Blue" MouseUp="clickableEllipse_MouseUp" />
</Window>
```

```
Imports System.Windows ' Window, MessageBox
Imports System.Windows.Input ' MouseButtonEventArgs

Namespace SDKSample

    Public Class EllipseEventHandlingWindow
        Inherits Window

        Public Sub New()
            InitializeComponent()
        End Sub

        Private Sub clickableEllipse_MouseUp(ByVal sender As Object, ByVal e As MouseButtonEventArgs)
            MessageBox.Show("You clicked the ellipse!")
        End Sub

    End Class

End Namespace
```

```
using System.Windows; // Window, MessageBox
using System.Windows.Input; // MouseButtonEventHandler

namespace SDKSample
{
    public partial class EllipseEventHandlingWindow : Window
    {
        public EllipseEventHandlingWindow()
        {
            InitializeComponent();
        }

        void clickableEllipse_MouseUp(object sender, MouseButtonEventArgs e)
        {
            // Display a message
            MessageBox.Show("You clicked the ellipse!");
        }
    }
}
```

다음 그림은 앞의 코드에서 생성된 내용을 보여 주며,



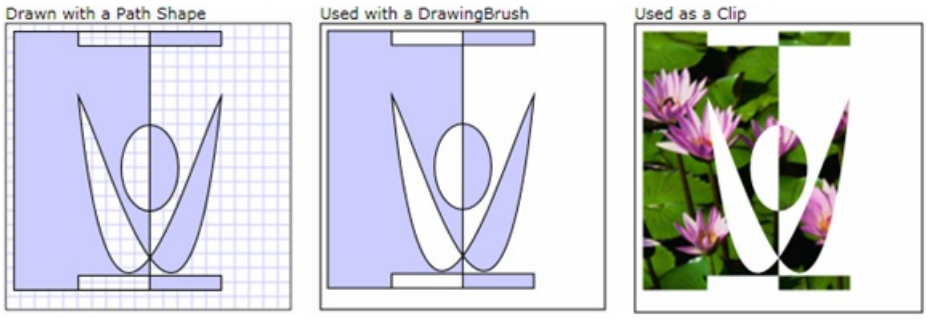
자세한 내용은 WPF 개요에서 셰이프 및 기본 도면을참조하십시오.

2D 기하 도형

WPF에서 제공하는 2D 도형은 기본 도형의 표준 집합을 포함합니다. 그러나 사용자 지정 사용자 인터페이스의 디자인이 용이하도록 사용자 지정 도형을 만들어야 할 수도 있습니다. 이 용도로 WPF는 기하 도형을 제공합니다. 다음 그림은 기하 도형을 사용하여 직접 그리거나, 브러시로 사용하거나, 다른 도형 및 컨트롤을 자르는 데 사용할 수 있는 사용자 지정 도형을 만드는 과정을 보여 줍니다.

Path 개체는 닫혔거나 열린 도형, 여러 도형 및 곡선 도형을 그리는 데 사용할 수 있습니다.

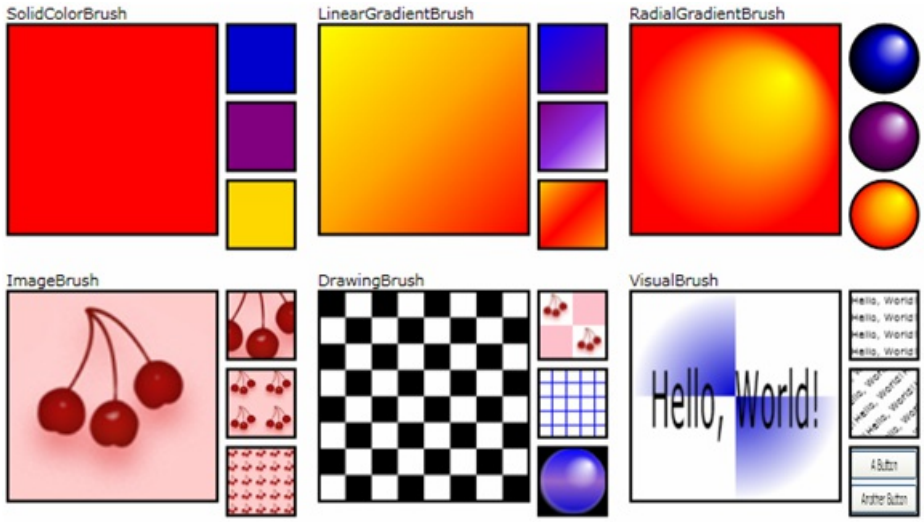
Geometry 개체는 자르기, 적중 테스트 및 2D 그래픽 데이터 렌더링에 사용할 수 있습니다.



자세한 내용은 기하 도형 개요를 참조하세요.

2D 효과

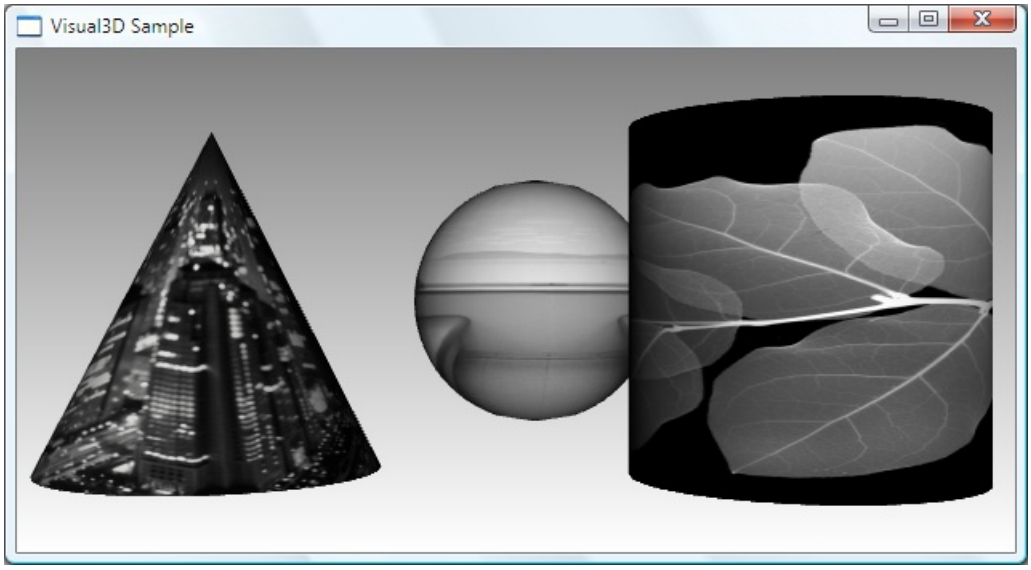
WPF 2D 기능의 하위 집합에는 그라데이션, 비트맵, 그리기, 비디오로 그리기, 회전, 크기 조정 및 기울이기와 같은 시각 효과가 포함됩니다. 이들은 모두 브러시로 달성됩니다. 다음 그림은 몇 가지 예를 보여줍니다.



자세한 내용은 WPF 브러시 개요를 참조하세요.

3D 렌더링

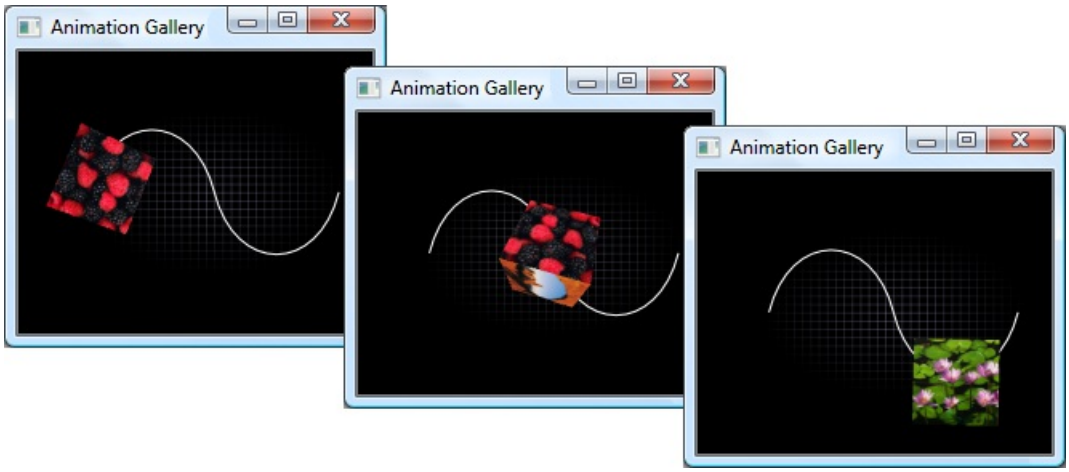
또한 WPF에는 2D 그래픽과 통합되는 3D 렌더링 기능이 포함되어 있어 더욱 흥미롭고 흥미로운 사용자 인터페이스를 만들 수 있습니다. 예를 들어 다음 그림은 3D 셰이프에 렌더링된 2D 이미지를 보여 주어집니다.



자세한 내용은 3D 그래픽 개요를 참조하세요.

애니메이션

WPF 애니메이션 지원을 사용하면 컨트롤이 커지거나, 흔들리거나, 회전하거나, 사라지도록 하여 흥미로운 페이지 전환 등을 만들 수 있습니다. 사용자 지정 클래스를 비롯한 대부분의 WPF 클래스에 애니메이션 효과를 줄 수 있습니다. 다음 그림은 간단한 애니메이션을 보여 주며 다음과 같은 작업을 수행합니다.



자세한 내용은 애니메이션 개요를 참조하세요.

미디어

풍부한 콘텐츠를 전달하는 한 가지 방법은 시청각 미디어를 사용하는 것입니다. WPF는 이미지, 비디오 및 오디오에 대한 특별한 지원을 제공합니다.

이미지

이미지는 대부분의 애플리케이션에서 공통적으로 사용되며 WPF는 이미지를 사용하는 여러 방법을 제공합니다. 다음 그림에서는 미리 보기 이미지를 포함하는 목록 상자가 있는 사용자 인터페이스를 보여 줍니다. 미리 보기를 선택하면 이미지가 전체 크기로 표시됩니다.

자세한 내용은 이미징 개요를 참조하세요.

비디오 및 오디오

MediaElement 컨트롤은 비디오와 오디오를 둘 다 재생할 수 있으며 사용자 지정 미디어 플레이어의 토대가 될 수 있을 정도로 유연합니다. 다음 XAML 태그는 미디어 플레이어를 구현합니다.

```
<MediaElement
  Name="myMediaElement"
  Source="media/wpf.wmv"
  LoadedBehavior="Manual"
  Width="350" Height="250" />
```

다음 그림의 창에는 MediaElement 작업 컨트롤이 표시됩니다.

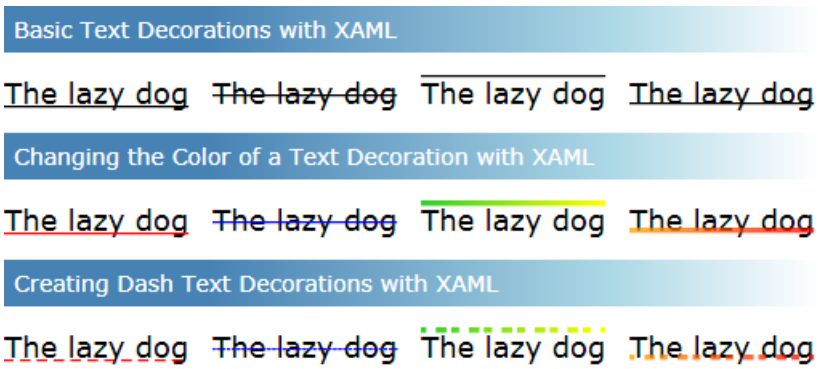
자세한 내용은 그래픽 및 멀티미디어를 참조하세요.

텍스트 및 입력 체계

고품질 텍스트 렌더링이 용이하도록 WPF는 다음 기능을 제공합니다.

- OpenType 글꼴 지원
- ClearType 향상
- 하드웨어 가속을 활용하는 고성능
- 미디어, 그래픽 및 애니메이션과 텍스트 통합
- 국가별 글꼴 지원 및 대체(fallback) 메커니즘

그래픽과의 텍스트 통합을 보여 주면 다음 그림에서 텍스트 장식의 적용을 보여 주며 다음과 같은 그림이 표시됩니다.



자세한 내용은 Windows Presentation Foundation의 입력 체계를 참조하세요.

WPF 앱 사용자 지정

지금까지 애플리케이션을 개발하기 위한 핵심 WPF 구성 요소를 살펴봤습니다. 애플리케이션 모델을 사용하여 주

로 컨트롤로 구성된 애플리케이션 콘텐츠를 호스트 및 제공합니다. 사용자 인터페이스에서 컨트롤 정렬을 간소화하고 창 크기 및 디스플레이 설정이 변경되어도 정렬이 유지되도록 하기 위해 WPF 레이아웃 시스템을 사용합니다. 대부분의 애플리케이션은 사용자의 데이터 조작을 허용하므로 데이터 바인딩을 사용하여 사용자 인터페이스와 데이터 통합 작업을 줄입니다. 애플리케이션의 시각적 모양을 개선하려면 WPF에서 제공하는 광범위한 그래픽, 애니메이션 및 미디어 지원을 사용합니다.

하지만 기본 사항이 고유하고 시각적으로 멋진 사용자 환경을 만들고 관리하는 데 충분하지 않은 경우도 많습니다. 표준 WPF 컨트롤이 원하는 애플리케이션 모양과 통합되지 않을 수 있습니다. 데이터가 가장 효율적인 방식으로 표시되지 않을 수도 있습니다. 애플리케이션의 전반적인 사용자 환경이 Windows 테마의 기본 모양과 느낌에 적합하지 않을 수 있습니다. 여러 측면에서 프레젠테이션 기술은 다른 형식의 확장성만큼 시각적 확장성을 필요로 합니다.

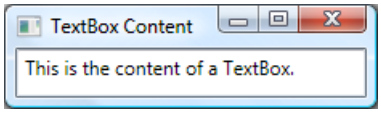
이런 이유로 WPF는 컨트롤, 트리거, 컨트롤 및 데이터 템플릿, 스타일, 사용자 인터페이스 리소스, 테마 및 스킨에 대한 풍부한 콘텐츠 모델을 포함하여 고유한 사용자 환경을 만들기 위한 다양한 메커니즘을 제공합니다.

콘텐츠 모델

대부분의 WPF 컨트롤은 주로 콘텐츠를 표시하는 데 사용됩니다. WPF에서 컨트롤의 콘텐츠를 구성할 수 있는 항목의 유형과 개수를 컨트롤의 *콘텐츠 모델*이라고 합니다. 일부 컨트롤은 단일 항목 및 유형의 콘텐츠를 포함할 수 있습니다. 예를 들어 TextBox 의 콘텐츠는 Text 속성에 할당되는 문자열 값입니다. 다음 예제는 TextBox다음의 내용을 설정합니다.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.TextBoxContentWindow"
    Title="TextBox Content">

    <TextBox Text="This is the content of a TextBox." />
</Window>
```

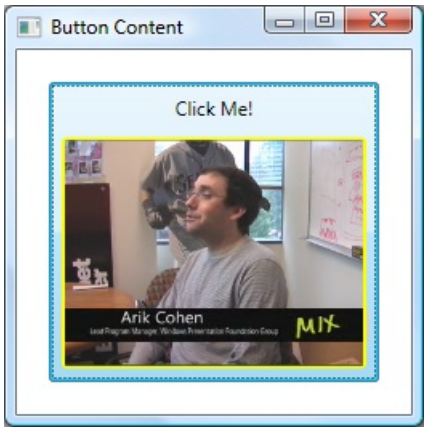
다음 그림은 결과를 보여 주며, 그 결과는 다음과 같은 것입니다.



그러나 다른 컨트롤은 다양한 콘텐츠 형식의 여러 항목을 포함할 수 있습니다. Content 속성으로 지정된 Button의 콘텐츠에는 레이아웃 컨트롤, 텍스트, 이미지 및 도형을 포함하여 다양한 항목이 포함될 수 있습니다. 다음 예제에서는 Button DockPanel을 포함하는 콘텐츠가 Label표시됩니다. Border MediaElement

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.ButtonContentWindow"
    Title="Button Content">

  <Button Margin="20">
    <!-- Button Content -->
    <DockPanel Width="200" Height="180">
      <Label DockPanel.Dock="Top" HorizontalAlignment="Center">Click Me!</Label>
      <Border Background="Black" BorderBrush="Yellow" BorderThickness="2"
        CornerRadius="2" Margin="5">
        <MediaElement Source="media/wpf.wmv" Stretch="Fill" />
      </Border>
    </DockPanel>
  </Button>
</Window>
```

다음 그림은 이 단추의 내용을 보여 주며 있습니다.



다양한 컨트롤에서 지원하는 콘텐츠 종류에 대한 자세한 내용은 WPF 콘텐츠 모델을 참조하세요.

트리거

XAML 태그의 주요 용도는 애플리케이션의 모양을 구현하는 것이지만 XAML을 사용하여 애플리케이션 동작의 일부 측면을 구현할 수도 있습니다. 한 가지 예는 트리거를 사용하여 사용자 조작에 따라 애플리케이션의 모양을 변경하는 것입니다. 자세한 내용은 스타일 및 템플릿을 참조하세요.

컨트롤 템플릿

WPF 컨트롤의 기본 사용자 인터페이스는 일반적으로 다른 컨트롤 및 도형에서 구성됩니다. 예를 들어 Button 은 ButtonChrome 및 ContentPresenter 컨트롤 둘 다로 구성됩니다. ButtonChrome 은 표준 단추 모양을 제공하는 반면, ContentPresenter 는 Content 속성에 지정된 대로 단추의 콘텐츠를 표시합니다.

컨트롤의 기본 모양이 애플리케이션의 전반적인 모양에 맞지 않을 수도 있습니다. 이 경우 ControlTemplate 을 사용하여 해당 콘텐츠 및 동작을 변경하지 않고 컨트롤의 사용자 인터페이스 모양을 변경할 수 있습니다.

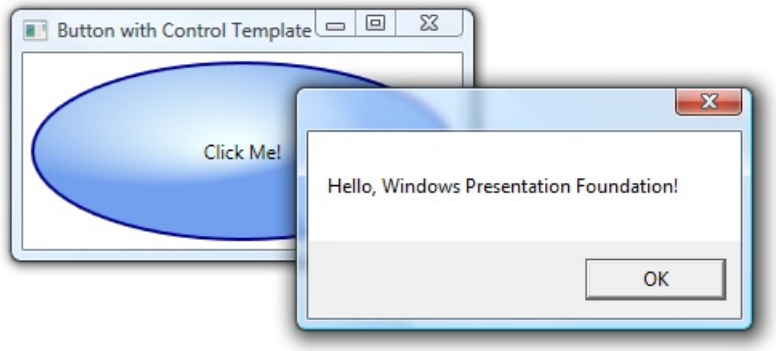
다음 예제에서는 Button ControlTemplate다음을 사용하여 a의 모양을 변경하는 방법을 보여 주며 있습니다.

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.ControlTemplateButtonWindow"
  Title="Button with Control Template" Height="158" Width="290">

  <!-- Button using an ellipse -->
  <Button Content="Click Me!" Click="button_Click">
    <Button.Template>
      <ControlTemplate TargetType="{x:Type Button}">
        <Grid Margin="5">
          <Ellipse Stroke="DarkBlue" StrokeThickness="2">
            <Ellipse.Fill>
              <RadialGradientBrush Center="0.3,0.2" RadiusX="0.5" RadiusY="0.5">
                <GradientStop Color="Azure" Offset="0.1" />
                <GradientStop Color="CornflowerBlue" Offset="1.1" />
              </RadialGradientBrush>
            </Ellipse.Fill>
          </Ellipse>
          <ContentPresenter Name="content" HorizontalAlignment="Center"
            VerticalAlignment="Center"/>
        </Grid>
      </ControlTemplate>
    </Button.Template>

  </Button>

</Window>
```

```
using System.Windows; // Window, RoutedEventArgs, MessageBox

namespace SDKSample
{
    public partial class ControlTemplateButtonWindow : Window
    {
        public ControlTemplateButtonWindow()
        {
            InitializeComponent();
        }

        void button_Click(object sender, RoutedEventArgs e)
        {
            // Show message box when button is clicked
            MessageBox.Show("Hello, Windows Presentation Foundation!");
        }
    }
}
```

```
Imports System.Windows ' Window, RoutedEventArgs, MessageBox

Namespace SDKSample

    Public Class ControlTemplateButtonWindow
        Inherits Window

        Public Sub New()

            InitializeComponent()

        End Sub

        Private Sub button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
            MessageBox.Show("Hello, Windows Presentation Foundation!")
        End Sub

    End Class

End Namespace
```

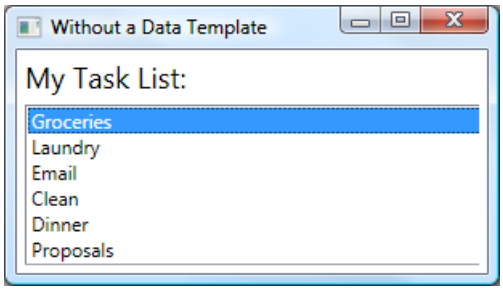
이 예제에서는 기본 단추 사용자 인터페이스가 진한 파란색 테두리가 있는 Ellipse로 대체되고 RadialGradientBrush를 사용하여 채워집니다. ContentPresenter 컨트롤은 Button의 콘텐츠인 "Click Me!"를 표시합니다. Button 을 클릭하면 Click 이벤트가 Button 컨트롤의 기본 동작의 일부로 여전히 발생합니다. 결과는 다음 그림에 나와 있습니다.



데이터 템플릿

컨트롤 템플릿을 사용하면 컨트롤의 모양을 지정할 수 있는 반면 데이터 템플릿을 사용하면 컨트롤 콘텐츠의 모양을 지정할 수 있습니다. 데이터 템플릿은 바인딩된 데이터가 표시되는 방식을 개선하는 데 자주 사용됩니다. 다음 그림은 각 작업에 이름, 설명 및 우선 `Task` 순위가 있는 개체 컬렉션에 바인딩된 a의 ListBox 기본 모양을 보여 주

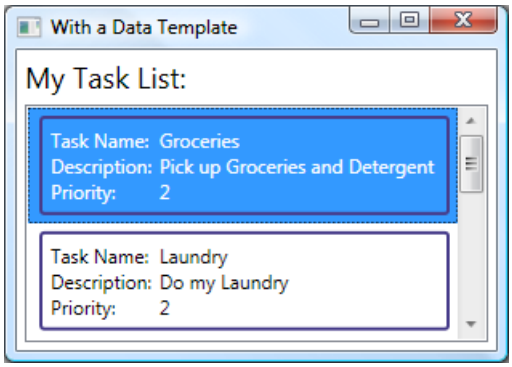
며, 이 그림에서는 이름, 설명 및 우선 순위가 있습니다.



기본 모양은 ListBox에서 예상되는 모양입니다. 그러나 각 작업의 기본 모양은 작업 이름만 포함합니다. 작업 이름, 설명 및 우선 순위를 표시하려면 DataTemplate을 사용하여 ListBox 컨트롤의 바인딩된 목록 항목에 대한 기본 모양을 변경해야 합니다. 다음 XAML은 DataTemplate ItemTemplate 특성을 사용하여 각 작업에 적용되는 이러한 를 정의합니다.

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.DataTemplateWindow"
  Title="With a Data Template">
  <Window.Resources>
    <!-- Data Template (applied to each bound task item in the task collection) -->
    <DataTemplate x:Key="myTaskTemplate">
      <Border Name="border" BorderBrush="DarkSlateBlue" BorderThickness="2"
        CornerRadius="2" Padding="5" Margin="5">
        <Grid>
          <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
          </Grid.RowDefinitions>
          <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition />
          </Grid.ColumnDefinitions>
          <TextBlock Grid.Row="0" Grid.Column="0" Padding="0,0,5,0" Text="Task Name:"/>
          <TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding Path=TaskName}"/>
          <TextBlock Grid.Row="1" Grid.Column="0" Padding="0,0,5,0" Text="Description:"/>
          <TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding Path=Description}"/>
          <TextBlock Grid.Row="2" Grid.Column="0" Padding="0,0,5,0" Text="Priority:"/>
          <TextBlock Grid.Row="2" Grid.Column="1" Text="{Binding Path=Priority}"/>
        </Grid>
      </Border>
    </DataTemplate>
  </Window.Resources>

  <!-- UI -->
  <DockPanel>
    <!-- Title -->
    <Label DockPanel.Dock="Top" FontSize="18" Margin="5" Content="My Task List:"/>

    <!-- Data template is specified by the ItemTemplate attribute -->
    <ListBox
      ItemsSource="{Binding}"
      ItemTemplate="{StaticResource myTaskTemplate}"
      HorizontalContentAlignment="Stretch"
      IsSynchronizedWithCurrentItem="True"
      Margin="5,0,5,5" />

  </DockPanel>
</Window>
```

다음 그림은 이 코드의 효과를 보여 주며, 이 코드의 효과를 보여 주며,



ListBox 의 동작과 전반적인 모양은 유지되고 목록 상자에 표시되는 콘텐츠의 모양만 변경되었습니다.

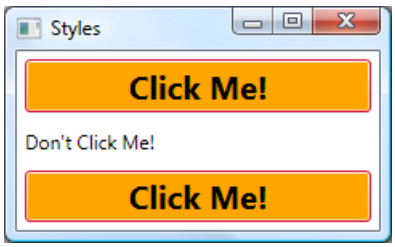
자세한 내용은 데이터 템플릿 개요를 참조하세요.

스타일

스타일을 사용하면 개발자와 디자이너가 해당 제품에 대해 특정 모양을 표준화할 수 있습니다. WPF는 Style 요소를 기반으로 하는 강력한 스타일 모델을 제공합니다. 다음 예제는 창의 모든 Button 배경 색을 다음과 `Orange` 같은 스타일로 설정하는 스타일을 만듭니다.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.StyleWindow"
    Title="Styles">
  <!-- Style that will be applied to all buttons -->
  <Style TargetType="{x:Type Button}">
    <Setter Property="Background" Value="Orange" />
    <Setter Property="BorderBrush" Value="Crimson" />
    <Setter Property="FontSize" Value="20" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Margin" Value="5" />
  </Style>
  <!-- This button will have the style applied to it -->
  <Button>Click Me!</Button>

  <!-- This label will not have the style applied to it -->
  <Label>Don't Click Me!</Label>

  <!-- This button will have the style applied to it -->
  <Button>Click Me!</Button>
</Window>
```

이 스타일은 모든 Button 컨트롤을 대상으로 하기 때문에 다음 그림과 같이 창에 있는 모든 단추에 스타일이 자동으로 적용됩니다.



자세한 내용은 스타일 및 템플릿을 참조하세요.

리소스

애플리케이션의 컨트롤은 글꼴 및 배경색부터 컨트롤 템플릿, 데이터 템플릿 및 스타일까지 모든 항목을 포함할 수 있는 동일한 모양을 공유해야 합니다. 사용자 인터페이스 리소스에 대한 WPF 지원을 사용하여 재사용을 위해 이러한 리소스를 단일 위치에 캡슐화할 수 있습니다.

다음 예제는 a 및 Button Labela에서 공유하는 공통 배경색을 정의합니다.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.ResourcesWindow"
    Title="Resources Window">

  <!-- Define window-scoped background color resource -->
  <Window.Resources>
    <SolidColorBrush x:Key="defaultBackground" Color="Red" />
  </Window.Resources>

  <!-- Button background is defined by window-scoped resource -->
  <Button Background="{StaticResource defaultBackground}">One Button</Button>

  <!-- Label background is defined by window-scoped resource -->
  <Label Background="{StaticResource defaultBackground}">One Label</Label>
</Window>
```

이 예제에서는 `Window.Resources` 속성 요소를 사용하여 배경색 리소스를 구현합니다. 이 리소스는 Window의 모든 자식에서 사용할 수 있습니다. 다음을 포함하여 다양한 리소스 범위가 있습니다(확인되는 순서대로 나열됨).

1. 개별 컨트롤(상속된 System.Windows.FrameworkElement.Resources 속성 사용)
2. Window 또는 Page (또한 상속된 System.Windows.FrameworkElement.Resources 속성 사용)
3. Application (System.Windows.Application.Resources 속성 사용)

다양한 범위는 리소스를 정의 및 공유하는 방법과 관련해서 유연성을 제공합니다.

리소스를 특정 범위에 직접 연결하는 대신, 애플리케이션의 다른 부분에서 참조할 수 있는 별도 ResourceDictionary 를 사용하여 하나 이상의 리소스를 패키지할 수 있습니다. 예를 들어 다음 예제에서는 리소스 사전에서 기본 배경색을 정의합니다.

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <!-- Define background color resource -->
  <SolidColorBrush x:Key="defaultBackground" Color="Red" />

  <!-- Define other resources -->
</ResourceDictionary>
```

다음 예제에서는 응용 프로그램 간에 공유되도록 이전 예제에서 정의된 리소스 사전을 참조합니다.

```
<Application
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.App">

  <Application.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="BackgroundColorResources.xaml"/>
      </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
</Application>
```

리소스 및 리소스 사전은 테마 및 스킨에 대한 WPF 지원의 기반이 됩니다.

자세한 내용은 리소스를 참조하세요.

사용자 지정 컨트롤

WPF는 다양한 사용자 지정 지원을 제공하지만 기존 WPF 컨트롤이 애플리케이션 또는 해당 사용자의 요구를 충족하지 않는 경우가 발생할 수 있습니다. 이 오류는 다음과 같은 경우에 발생할 수 있습니다.

- 사용자가 요구하는 사용자 인터페이스가 기존 WPF 가 구현하는 모양이나 느낌으로 만들 수 없는 경우
- 필요한 동작이 기존 WPF 구현에서 지원되지 않는 경우(또는 쉽게 지원되지 않는 경우)

그러나 이제 세 가지 WPF 모델 중 하나를 활용하여 새 컨트롤을 만들 수 있습니다. 각 모델은 특정 시나리오를 대상으로 하며, 특정 WPF 기본 클래스에서 사용자 지정 컨트롤을 파생시켜야 합니다. 세 가지 모델은 다음과 같습니다.

- **사용자 정의 컨트롤 모델**. 사용자 지정 컨트롤은 UserControl 에서 파생되며 하나 이상의 다른 컨트롤로 구성됩니다.
- **컨트롤 모델**. 사용자 지정 컨트롤은 Control 에서 파생되며 대부분의 WPF 컨트롤과 마찬가지로 템플릿을 사용하여 모양과 동작을 구분하는 구현을 작성하는 데 사용됩니다. Control 에서 파생시키는 경우 사용자 정의 컨트롤보다 더 자유롭게 사용자 지정 사용자 인터페이스를 만들 수 있지만 더 많은 노력이 필요할 수 있습니다.
- **프레임워크 요소 모델**. 사용자 지정 컨트롤은 모양이 사용자 지정 렌더링 논리(템플릿 아님)에 의해 정의되는 경우 FrameworkElement 에서 파생됩니다.

다음 예제에서는 다음에서 UserControl파생되는 사용자 지정 숫자 업/다운 컨트롤을 보여 주며 있습니다.

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="SDKSample.NumericUpDown">

  <Grid>

    <Grid.RowDefinitions>
      <RowDefinition/>
      <RowDefinition/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition/>
    </Grid.ColumnDefinitions>

    <!-- Value text box -->
    <Border BorderThickness="1" BorderBrush="Gray" Margin="2" Grid.RowSpan="2"
      VerticalAlignment="Center" HorizontalAlignment="Stretch">
      <TextBlock Name="valueText" Width="60" TextAlignment="Right" Padding="5"/>
    </Border>

    <!-- Up/Down buttons -->
    <RepeatButton Name="upButton" Click="upButton_Click" Grid.Column="1"
      Grid.Row="0">Up</RepeatButton>
    <RepeatButton Name="downButton" Click="downButton_Click" Grid.Column="1"
      Grid.Row="1">Down</RepeatButton>

  </Grid>

</UserControl>
```

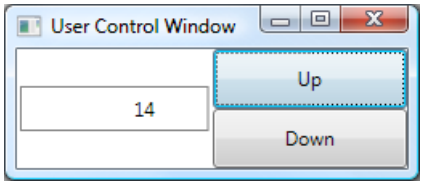
```
using System; // EventArgs
using System.Windows; // DependencyObject, DependencyPropertyChangedEventArgs,
                      // FrameworkPropertyMetadata, PropertyChangedCallback,
                      // RoutedPropertyChangedEventArgs
using System.Windows.Controls; // UserControl

namespace SDKSample
{
    public partial class NumericUpDown : UserControl
    {
        // NumericUpDown user control implementation
    }
}
```

```
imports System 'EventArgs
imports System.Windows 'DependencyObject, DependencyPropertyChangedEventArgs,
                       ' FrameworkPropertyMetadata, PropertyChangedCallback,
                       ' RoutedPropertyChangedEventArgs
imports System.Windows.Controls 'UserControl

Namespace SDKSample

    ' Interaction logic for NumericUpDown.xaml
    Partial Public Class NumericUpDown
        Inherits System.Windows.Controls.UserControl

        'NumericUpDown user control implementation

    End Class

End Namespace
```

다음 예제에서는 사용자 컨트롤을 Window다음 에 통합하는 데 필요한 XAML을 보여 줍니다.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="SDKSample.UserControlWindow"
    xmlns:local="clr-namespace:SDKSample"
    Title="User Control Window">

  <!-- Numeric Up/Down user control -->
  <local:NumericUpDown />

</Window>
```

다음 그림은 다음 `NumericUpDown` 에서 호스팅되는 컨트롤을 Window보여 주며 있습니다.



사용자 지정 컨트롤에 대한 자세한 내용은 컨트롤 제작 개요를 참조하세요.

WPF 모범 사례

모든 개발 플랫폼과 마찬가지로 WPF를 다양한 방법으로 사용하여 원하는 결과를 얻을 수 있습니다. WPF 애플리케이션이 필요한 사용자 환경을 제공하고 일반적인 사용자 요구를 충족하도록 하는 한 가지 방법으로 접근성, 전역

화 및 지역화, 성능에 대한 권장 모범 사례가 있습니다. 자세한 내용은 다음을 참조하세요.

- 접근성
- WPF 전역화 및 지역화
- WPF 앱 성능
- WPF 보안

다음 단계

WPF의 주요 기능을 살펴보았습니다. 이제 첫 번째 WPF 앱을 만들 순서입니다.

연습: 내 첫 번째 WPF 데스크톱 앱

참고 항목

- WPF 시작
- Windows Presentation Foundation
- WPF 커뮤니티 리소스

시작(WPF)

2020-04-24 • 2 minutes to read • Edit Online

WPF(Windows Presentation Foundation)는 데스크톱 클라이언트 애플리케이션을 만드는 UI 프레임워크입니다. WPF 개발 플랫폼은 애플리케이션 모델, 리소스, 컨트롤, 그래픽, 레이아웃, 데이터 바인딩, 문서 및 보안을 포함하여 다양한 애플리케이션 개발 기능 집합을 지원합니다. .NET Framework의 하위 집합이므로 이전에 ASP.NET 또는 Windows Forms를 사용하여 .NET Framework로 애플리케이션을 빌드한 경우 프로그래밍 환경이 비슷합니다. WPF는 XAML(Extensible Application Markup Language)을 사용하여 애플리케이션 프로그래밍을 위한 선언적 모델을 제공합니다. 이 섹션의 항목에서는 WPF를 소개하고 WPF를 시작하는 데 유용한 정보를 제공합니다.

어디서 시작해야 합니까?

바로 시작	연습: 내 첫 WPF 데스크톱 애플리케이션
애플리케이션 UI를 디자인하려면 어떻게 해야 하나요?	Visual Studio에서 XAML 디자인
.NET을 처음 사용하세요?	.NET Framework의 개요 Visual C# 및 Visual Basic 시작
WPF에 대한 자세한 설명…	Visual Studio에서의 WPF 소개 XAML 개요(WPF) 컨트롤 데이터 바인딩 개요
Windows Forms 개발자인가요?	Windows Forms 컨트롤 및 해당 WPF 컨트롤 WPF 및 Windows Forms 상호 운용성

참고 항목

- 클래스 라이브러리
- 응용 프로그램 개발
- .NET Framework 개발자 센터

애플리케이션 개발

2020-03-21 • 15 minutes to read • Edit Online

WPF(Windows 프레젠테이션 파운데이션)는 다음과 같은 유형의 응용 프로그램을 개발하는 데 사용할 수 있는 프레젠테이션 프레임워크입니다.

- 독립 실행형 응용 프로그램(클라이언트 컴퓨터에서 설치되고 실행되는 실행 형 어셈블리로 빌드된 기존 스타일 Windows 응용 프로그램)
- XAML 브라우저 응용 프로그램(XBAPs) (실행 가능한 어셈블리로 빌드되고 Microsoft 인터넷 익스플로러 또는 모질라 파이어폭스와 같은 웹 브라우저에서 호스팅되는 탐색 페이지로 구성된 응용 프로그램).
- 사용자 지정 컨트롤 라이브러리(재사용 가능한 컨트롤을 포함하는 실행 불가능한 어셈블리)
- 클래스 라이브러리(재사용 가능한 클래스를 포함하는 실행 불가능한 어셈블리)

> **NOTE**
> Windows 서비스에서 WPF 유형을 사용해서는 안 됩니다. Windows 서비스에서 이러한 기능을 사용하려고 하면 예상대로 작동하지 않을 수 있습니다.

이 응용 프로그램 집합을 빌드하기 위해 WPF는 다양한 서비스를 구현합니다. 이 항목에서 이러한 서비스의 개요를 제공하며 자세한 정보를 찾을 수 있습니다.

애플리케이션 관리

실행 가능한 WPF 응용 프로그램에는 일반적으로 다음을 포함하는 핵심 기능 집합이 필요합니다.

- 일반적인 애플리케이션 인프라 만들기 및 관리(시스템 및 입력 메시지를 수신하는 Windows 메시지 루프 및 진입점 메서드 만들기 포함)
- 애플리케이션의 수명 추적 및 상호 작용
- 명령줄 매개 변수 검색 및 처리
- 애플리케이션 범위 속성 및 UI 리소스 공유
- 처리되지 않은 예외 검색 및 처리
- 종료 코드 반환
- 독립 실행형 애플리케이션에서 창 관리
- XAML 브라우저 응용 프로그램(XBAPs) 및 탐색 창 및 프레임이 있는 독립 실행형 응용 프로그램에서 탐색추적.

이러한 기능은 *애플리케이션 정의*를 사용하여 애플리케이션에 추가되는 Application 클래스에 의해 구현됩니다.

자세한 내용은 애플리케이션 관리 개요를 참조하세요.

WPF 애플리케이션 리소스, 콘텐츠 및 데이터 파일

WPF는 리소스, 콘텐츠 및 데이터라는 세 가지 종류의 실행 불가능한 데이터 파일을 지원하면서 포함된 리소스에 대한 Microsoft .NET 프레임워크의 핵심 지원을 확장합니다. 자세한 내용은 WPF 애플리케이션 리소스, 콘텐츠 및 데이터 파일을 참조하세요.

WPF 실행 불가능한 데이터 파일에 대한 지원의 핵심 구성 요소는 고유한 URI를 사용하여 식별하고 로드하는 기능입니다. 자세한 내용은 WPF의 팩 URI를참조하십시오.

창 및 대화 상자

사용자는 창을 통해 WPF 독립 실행형 응용 프로그램과 상호 작용합니다. 창의 목적은 애플리케이션 콘텐츠를 호스트하고 일반적으로 사용자가 콘텐츠와 상호 작용할 수 있도록 애플리케이션 기능을 노출하는 것입니다. WPF에서 창은 다음을 지원하는 Window 클래스에 의해 캡슐화됩니다.

- 창 만들기 및 표시
- 소유자/피소유자 창 관계 설정
- 창 모양 구성(예: 크기, 위치, 아이콘, 제목 표시줄 텍스트, 테두리)
- 창의 수명 추적 및 상호 작용

자세한 내용은 WPF 창 개요를 참조하세요.

Window는 대화 상자로 알려진 특별한 유형의 창을 만드는 기능을 지원합니다. 모달 및 모덜리스 유형의 대화 상자를 모두 만들 수 있습니다.

WPF는 편의를 위해 응용 프로그램 전체에서 재사용 가능성과 일관된 사용자 환경의 이점을 위해 OpenFileDialog SaveFileDialog일반적인 PrintDialogWindows 대화 상자 세 가지를 노출합니다.

메시지 상자는 중요한 텍스트 정보를 보여 주고 간단한 예/아니요/확인/취소 질문을 묻는 특별한 유형의 대화 상자입니다. MessageBox 클래스를 사용하여 메시지 상자를 만들고 표시합니다.

자세한 내용은 대화 상자 개요를 참조하세요.

탐색

WPF는 페이지 ()Page및 하이퍼링크()를 사용하여 웹 스타일 탐색을Hyperlink지원합니다. 다음을 포함한 다양한 방법으로 탐색을 구현할 수 있습니다.

- 웹 브라우저에 호스트되는 독립 실행형 페이지
- 웹 브라우저에서 호스팅되는 XBAP로 컴파일된 페이지입니다.
- 독립 실행형 애플리케이션으로 컴파일되고 탐색 창(NavigationWindow)에서 호스트하는 페이지
- 프레임()에Frame의해 호스팅되는 페이지()로, 독립 실행형 페이지에서 호스팅되거나 XBAP 또는 독립 실행형 응용 프로그램으로 컴파일된 페이지입니다.

탐색을 용이하게 하기 위해 WPF는 다음을 구현합니다.

- NavigationService에서 응용 프로그램 내 탐색을 지원하는 Frame및 NavigationWindowXBAPs에서 사용되는 탐색 요청을 처리하기 위한 공유 탐색 엔진입니다.
- 탐색을 시작하는 탐색 메서드
- 탐색 수명을 추적하고 상호 작용하는 탐색 이벤트
- 저널을 사용하여 뒤로/앞으로 탐색 저장(검사하고 조작할 수도 있음)

자세한 내용은 탐색 개요를 참조하세요.

WPF는 구조화 된 탐색이라고하는 특별한 유형의 탐색도 지원합니다. 구조적 탐색을 사용하여 호출 함수와 일치하는 구조적이고 예측 가능한 방식으로 데이터를 반환하는 하나 이상의 페이지를 호출할 수 있습니다. 이 기능은 PageFunction<T> 클래스에 따라 달라지며 구조적 탐색 개요에 자세히 설명되어 있습니다. PageFunction<T>은 복잡한 탐색 토폴로지 생성을 간소화하는 역할도 합니다. 이러한 탐색 토폴로지에 대한 설명은 탐색 토폴로지 개

요에 나와 있습니다.

Hosting

XBAPs는 마이크로 소프트 인터넷 익스플로러 또는 파이어 폭스에서 호스팅 할 수 있습니다. 각 호스팅 모델에는 호스팅에서 다루는 고유한 고려 사항 및 제약 조건 집합이 있습니다.

빌드 및 배포

간단한 WPF 응용 프로그램은 명령줄 컴파일러를 사용하여 명령 프롬프트에서 빌드할 수 있지만 WPF는 Visual Studio와 통합되어 개발 및 빌드 프로세스를 간소화하는 추가 지원을 제공합니다. 자세한 내용은 WPF 애플리케이션 빌드를 참조하세요.

빌드하는 애플리케이션의 유형에 따라 선택할 수 있는 하나 이상의 배포 옵션이 있습니다. 자세한 내용은 WPF 애플리케이션 배포를 참조하세요.

관련 항목

제목	DESCRIPTION
애플리케이션 관리 개요	애플리케이션 수명, 창, 애플리케이션 리소스 및 탐색 관리를 비롯하여 Application 클래스에 대해 개략적으로 설명합니다.
WPF의 창	Window 클래스 및 대화 상자의 사용 방법을 비롯하여 애플리케이션의 창 관리에 대해 세부적으로 설명합니다.
탐색 개요	애플리케이션 페이지 간의 탐색 관리에 대해 개략적으로 설명합니다.
호스팅	XAML 브라우저 응용 프로그램(XBAP)에 대한 개요를 제공합니다.
빌드 및 배포	WPF 애플리케이션을 빌드하고 배포하는 방법에 대해 설명합니다.
Visual Studio에서의 WPF 소개	WPF의 주요 기능에 대해 설명합니다.
연습: 첫 번째 WPF 데스크톱 응용 프로그램	페이지 탐색, 레이아웃, 컨트롤, 이미지, 스타일 및 바인딩을 사용하여 WPF 애플리케이션을 만드는 방법을 보여 주는 연습입니다.

고급(Windows Presentation Foundation)

2020-03-26 • 2 minutes to read • Edit Online

이 섹션의 WPF에서 일부 고급 영역을 설명합니다.

섹션 내용

컨트롤

2020-04-24 • 16 minutes to read • Edit Online

WPF(Windows Presentation Foundation)의 거의 모든 Windows 응용 Button프로그램에서 Label TextBox Menu사용되는 많은 공통 UI 구성 요소와 ListBox함께 제공됩니다. 지금까지 이러한 개체는 컨트롤이라고 불렀습니다. WPF SDK는 응용 프로그램에서 보이는 개체를 나타내는 클래스를 느슨하게 의미하기 위해 "control"이라는 용어를 계속 사용하지만 클래스가 보이는 Control 존재를 갖기 위해 클래스에서 상속할 필요가 없다는 점에 유의해야 합니다. Control 클래스에서 상속하는 클래스에는 ControlTemplate컨트롤의 소비자가 새 하위 클래스를 만들지 않고도 컨트롤의 모양을 근본적으로 변경할 수 있는 을 포함합니다. 이 항목에서는 Control 컨트롤(클래스에서 상속하는 컨트롤과 클래스에서 상속하지 않는 컨트롤 WPF모두)이 에서 일반적으로 사용되는 방법에 대해 설명합니다.

컨트롤의 인스턴스 만들기

XAML(Extensible Application Markup Language) 또는 코드를 사용하여 응용 프로그램에 컨트롤을 추가할 수 있습니다. 다음 예제에서는 사용자에게 이름과 성을 묻는 간단한 애플리케이션을 만드는 방법을 보여 줍니다. 이 예제에서는 XAML에서 두 개의 레이블, 두 개의 텍스트 상자 및 두 개의 단추등 6개의 컨트롤을 만듭니다. 모든 컨트롤을 유사하게 만들 수 있습니다.

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
    <RowDefinition Height="30"/>
    <RowDefinition/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition/>
    <ColumnDefinition/>
  </Grid.ColumnDefinitions>

  <Label>
    Enter your first name:
  </Label>
  <TextBox Grid.Row="0" Grid.Column="1"
           Name="firstName" Margin="0,5,10,5"/>

  <Label Grid.Row="1" >
    Enter your last name:
  </Label>
  <TextBox Grid.Row="1" Grid.Column="1"
           Name="lastName" Margin="0,5,10,5"/>

  <Button Grid.Row="2" Grid.Column="0"
          Name="submit" Margin="2">
    View message
  </Button>

  <Button Grid.Row="2" Grid.Column="1"
          Name="Clear" Margin="2">
    Clear Name
  </Button>
</Grid>
```

다음 예제에서는 코드로 동일한 애플리케이션을 만듭니다. 간결하게 하기 위해 Grid에서 `grid1`를 만들면 샘플에서 제외됩니다. `grid1`이전 XAML 예제와 같이 동일한 열 및 행 정의가 있습니다.

```
Label firstNameLabel;
Label lastNameLabel;
TextBox firstName;
TextBox lastName;
Button submit;
Button clear;

void CreateControls()
{
    firstNameLabel = new Label();
    firstNameLabel.Content = "Enter your first name:";
    grid1.Children.Add(firstNameLabel);

    firstName = new TextBox();
    firstName.Margin = new Thickness(0, 5, 10, 5);
    Grid.SetColumn(firstName, 1);
    grid1.Children.Add(firstName);

    lastNameLabel = new Label();
    lastNameLabel.Content = "Enter your last name:";
    Grid.SetRow(lastNameLabel, 1);
    grid1.Children.Add(lastNameLabel);

    lastName = new TextBox();
    lastName.Margin = new Thickness(0, 5, 10, 5);
    Grid.SetColumn(lastName, 1);
    Grid.SetRow(lastName, 1);
    grid1.Children.Add(lastName);

    submit = new Button();
    submit.Content = "View message";
    Grid.SetRow(submit, 2);
    grid1.Children.Add(submit);

    clear = new Button();
    clear.Content = "Clear Name";
    Grid.SetRow(clear, 2);
    Grid.SetColumn(clear, 1);
    grid1.Children.Add(clear);
}
```

```
Private firstNameLabel As Label
Private lastNameLabel As Label
Private firstName As TextBox
Private lastName As TextBox
Private submit As Button
Private clear As Button

Sub CreateControls()
    firstNameLabel = New Label()
    firstNameLabel.Content = "Enter your first name:"
    grid1.Children.Add(firstNameLabel)

    firstName = New TextBox()
    firstName.Margin = New Thickness(0, 5, 10, 5)
    Grid.SetColumn(firstName, 1)
    grid1.Children.Add(firstName)

    lastNameLabel = New Label()
    lastNameLabel.Content = "Enter your last name:"
    Grid.SetRow(lastNameLabel, 1)
    grid1.Children.Add(lastNameLabel)

    lastName = New TextBox()
    lastName.Margin = New Thickness(0, 5, 10, 5)
    Grid.SetColumn(lastName, 1)
    Grid.SetRow(lastName, 1)
    grid1.Children.Add(lastName)

    submit = New Button()
    submit.Content = "View message"
    Grid.SetRow(submit, 2)
    grid1.Children.Add(submit)

    clear = New Button()
    clear.Content = "Clear Name"
    Grid.SetRow(clear, 2)
    Grid.SetColumn(clear, 1)
    grid1.Children.Add(clear)


End Sub
```

컨트롤의 모양 변경

일반적으로 애플리케이션의 모양과 느낌에 맞게 컨트롤의 모양을 변경합니다. 수행하려는 작업에 따라 다음 중 하나를 수행하여 컨트롤의 모양을 변경할 수 있습니다.

- 컨트롤의 속성 값을 변경합니다.
- 컨트롤에 Style 대한 a를 만듭니다.
- 컨트롤에 ControlTemplate 대한 새 만들기

컨트롤의 속성 값 변경

대부분의 컨트롤에는 의 와 같이 컨트롤이 표시되는 방식을 Background 변경할 Button수 있는 속성이 있습니다. 값 속성과 코드 모두에서 XAML 설정할 수 있습니다. 다음 예제에서는 Background에서 FontSize에서 FontWeight 의 Button 에서 XAML의 및 속성을 설정합니다.

```
<Button FontSize="14" FontWeight="Bold">
  <!--Set the Background property of the Button to
    a LinearGradientBrush.-->
  <Button.Background>
    <LinearGradientBrush StartPoint="0,0.5"
                         EndPoint="1,0.5">
      <GradientStop Color="Green" Offset="0.0" />
      <GradientStop Color="White" Offset="0.9" />
    </LinearGradientBrush>

  </Button.Background>
  View message
</Button>
```

다음 예제에서는 코드로 동일한 속성을 설정합니다.

```
LinearGradientBrush buttonBrush = new LinearGradientBrush();
buttonBrush.StartPoint = new Point(0, 0.5);
buttonBrush.EndPoint = new Point(1, 0.5);
buttonBrush.GradientStops.Add(new GradientStop(Colors.Green, 0));
buttonBrush.GradientStops.Add(new GradientStop(Colors.White, 0.9));

submit.Background = buttonBrush;
submit.FontSize = 14;
submit.FontWeight = FontWeights.Bold;
```

```
Dim buttonBrush As New LinearGradientBrush()
buttonBrush.StartPoint = New Point(0, 0.5)
buttonBrush.EndPoint = New Point(1, 0.5)
buttonBrush.GradientStops.Add(New GradientStop(Colors.Green, 0))
buttonBrush.GradientStops.Add(New GradientStop(Colors.White, 0.9))

submit.Background = buttonBrush
submit.FontSize = 14
submit.FontWeight = FontWeights.Bold
```

컨트롤에 대한 스타일 만들기

WPF을 만들어 응용 프로그램의 각 인스턴스에 속성을 설정하는 대신 컨트롤 도매의 모양을 Style지정할 수 있습니다. 다음 예제에서는 Style 응용 프로그램의 각 Button 에 적용 되는 a를 만듭니다. Style정의는 XAML 일반적으로 ResourceDictionary에 정의됩니다. Resources FrameworkElement

```
<Style TargetType="Button">
  <Setter Property="FontSize" Value="14"/>
  <Setter Property="FontWeight" Value="Bold"/>
  <Setter Property="Background">
    <Setter.Value>
      <LinearGradientBrush StartPoint="0,0.5"
                           EndPoint="1,0.5">
        <GradientStop Color="Green" Offset="0.0" />
        <GradientStop Color="White" Offset="0.9" />
      </LinearGradientBrush>

    </Setter.Value>
  </Setter>
</Style>
```

스타일에 키를 할당하고 컨트롤 속성에 해당 키를 지정하여 특정 형식의 특정 컨트롤에만 `Style` 스타일을 적용할 수도 있습니다. 스타일에 대한 자세한 내용은 스타일 지정 및 템플릿을참조하십시오.

ControlTemplate 만들기

A를 Style 사용하면 한 번에 여러 컨트롤에 속성을 설정할 수 있지만 Control Style때로는 을 만들어 수행할 수 있는 것 이상으로 모양을 사용자 지정할 수 있습니다. Control 클래스에서 상속하는 클래스에는 ControlTemplate Control의 구조와 모양을 정의하는 .가 있습니다. a의 Template Control 속성은 공용이므로 기본값과 Control ControlTemplate 다른 a를 제공할 수 있습니다. 컨트롤에서 상속하는 ControlTemplate Control 대신 에 대한 새 를 지정하여 Control의 모양을 사용자 지정할 수 있습니다.

매우 일반적인 컨트롤을 Button고려하십시오. a의 Button 주요 동작은 사용자가 클릭할 때 응용 프로그램이 일부 작업을 수행할 수 있도록 하는 것입니다. 기본적으로 in은 Button WPF 발생한 사각형으로 나타납니다. 응용 프로그램을 개발하는 동안 단추의 클릭 이벤트를 처리하여 Buttona-의 동작을 활용할 수 있지만 단추의 속성을 변경하여 수행할 수 있는 것 이상으로 단추모양을 변경할 수 있습니다. 이 경우 새 ControlTemplate을 만들 수 있습니다.

다음 예제는 ControlTemplate 에 대 한는 Button합니다. 는 ControlTemplate 둥근 Button 모서리와 그라데이션 배경을 가진 을 만듭니다. 에는 ControlTemplate 두 Border Background 개의 GradientStop LinearGradientBrush 개체가 있는 있는 개체가 포함되어 있습니다. 첫 GradientStop 번째는 데이터 바인딩을 Color 사용하여 GradientStop 단추의 배경 색에 의 속성을 바인딩합니다. 의 Background 속성을 설정하면 해당 값의 색상이 첫 번째 GradientStop로 사용됩니다. Button 데이터 바인딩에 대한 자세한 내용은 데이터 바인딩 개요를 참조하세요. 또한 이 Trigger 예제는 Button when의 IsPressed 모양을 변경하는 `true` a를 만듭니다.

```
<!--Define a template that creates a gradient-colored button.-->
<Style TargetType="Button">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="Button">
        <Border
          x:Name="Border"
          CornerRadius="20"
          BorderThickness="1"
          BorderBrush="Black">
          <Border.Background>
            <LinearGradientBrush StartPoint="0,0.5"
                                 EndPoint="1,0.5">
              <GradientStop Color="{Binding Background.Color,
                    RelativeSource={RelativeSource TemplatedParent}}"
                            Offset="0.0" />
              <GradientStop Color="White" Offset="0.9" />
            </LinearGradientBrush>
          </Border.Background>
          <ContentPresenter
            Margin="2"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            RecognizesAccessKey="True"/>
        </Border>
        <ControlTemplate.Triggers>
          <!--Change the appearance of
          the button when the user clicks it.-->
          <Trigger Property="IsPressed" Value="true">
            <Setter TargetName="Border" Property="Background">
              <Setter.Value>
                <LinearGradientBrush StartPoint="0,0.5"
                                     EndPoint="1,0.5">
                  <GradientStop Color="{Binding Background.Color,
                    RelativeSource={RelativeSource TemplatedParent}}"
                                Offset="0.0" />
                  <GradientStop Color="DarkSlateGray" Offset="0.9" />
                </LinearGradientBrush>
              </Setter.Value>
            </Setter>
          </Trigger>

        </ControlTemplate.Triggers>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

```
<Button Grid.Row="2" Grid.ColumnSpan="2" Name="submitName"
        Background="Green">View message</Button>
```

> **NOTE**
> 예제가 Background 제대로 Button 작동하려면 의 SolidColorBrush 속성을 a로 설정해야 합니다.

이벤트 구독

XAML 또는 코드를 사용하여 컨트롤의 이벤트를 구독할 수 있지만 코드에서만 이벤트를 처리할 수 있습니다. 다음 예제에서는 `Click` Button의 이벤트를 구독하는 방법을 보여 주며 있습니다.

```
<Button Grid.Row="2" Grid.ColumnSpan="2" Name="submitName" Click="submit_Click"
  Background="Green">View message</Button>
```

```
submit.Click += new RoutedEventHandler(submit_Click);
```

```
AddHandler submit.Click, AddressOf submit_Click
```

다음 예제는 `Click` Button의 이벤트를 처리합니다.

```
void submit_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show("Hello, " + firstName.Text + " " + lastName.Text);
}
```

```
Private Sub submit_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
    MessageBox.Show("Hello, " + firstName.Text + " " + lastName.Text)

End Sub
```

컨트롤의 풍부한 콘텐츠

Control 클래스에서 상속되는 대부분의 클래스에는 풍부한 콘텐츠를 포함할 수 있는 능력이 있습니다. 예를 들어 Label a는 문자열, Image. 또는 와 같은 Panel모든 개체를 포함할 수 있습니다. 다음 클래스는 리치 콘텐츠를 지원하고 의 대부분의 컨트롤에 대한 WPF기본 클래스 역할을 합니다.

- ContentControl-- 이 클래스에서 상속되는 클래스의 Label Button몇 ToolTip가지 예는 .
- ItemsControl-- 이 클래스에서 상속되는 클래스의 ListBox Menu몇 StatusBar가지 예는 .
- HeaderedContentControl-- 이 클래스에서 상속되는 클래스의 TabItem GroupBox몇 Expander가지 예는 .
- HeaderedItemsControl--이 클래스에서 상속되는 클래스의 MenuItem몇 TreeViewItem가지 ToolBar예는 " 및 .

이러한 기본 클래스에 대한 자세한 내용은 WPF 콘텐츠 모델을참조하십시오.

참고 항목

- 스타일 지정 및 템플릿
- 컨트롤 범주
- 컨트롤 라이브러리
- 데이터 템플릿 개요
- 데이터 바인딩 개요
- 입력
- 명령 사용
- 연습: 사용자 지정 애니메이션 효과가 있는 단추 만들기
- 컨트롤 사용자 지정

데이터

2020-04-24 • 2 minutes to read • Edit Online

WPF(Windows Presentation Foundation) 데이터 바인딩은 애플리케이션이 데이터를 제공하고 상호 작용할 수 있는 간단하고 일관된 방법을 제공합니다. 요소는 공통 언어 런타임(CLR) 개체 및 XML 의 형태로 다양한 데이터 원본의 데이터에 바인딩될 수 있습니다. WPF(Windows Presentation Foundation) 또한 끌어서 놓기 작업을 통해 데이터 전송을 위한 메커니즘을 제공합니다.

섹션 내용

데이터 바인딩
끌어서 놓기

참조

System.Windows.Data

Binding

DataTemplate

DataTemplateSelector

관련 단원

컨트롤

스타일 지정 및 템플릿

데이터 바인딩

참고 항목

- 연습: 내 첫 WPF 데스크톱 애플리케이션
- 연습: WPF 애플리케이션에서 애플리케이션 데이터 캐싱

그래픽 및 멀티미디어

2020-03-21 • 15 minutes to read • Edit Online

WPF(Windows Presentation Foundation)에서는 멀티미디어, 벡터 그래픽, 애니메이션 및 콘텐츠 컴퍼지션을 지원하여 개발자가 흥미로운 사용자 인터페이스 및 콘텐츠를 쉽게 작성할 수 있도록 합니다. Visual Studio를 사용하여 벡터 그래픽 또는 복잡한 애니메이션을 만들고 응용 프로그램에 미디어를 통합할 수 있습니다.

이 항목에서는 그래픽, 전환 효과, 소리 및 비디오를 애플리케이션에 추가할 수 있도록 하는 WPF의 그래픽, 애니메이션 및 미디어 기능을 소개합니다.

> **NOTE**
> Windows 서비스에서 WPF 유형을 사용해서는 안 됩니다. Windows 서비스에서 WPF 유형을 사용하려고 하면 서비스가 예상대로 작동하지 않을 수 있습니다.

WPF 4에 포함된 그래픽 및 멀티미디어의 새로운 기능

그래픽 및 애니메이션에 관련해서 몇 가지 사항이 변경되었습니다.

- 레이아웃 조정

  개체 가장자리가 픽셀 디바이스 중앙에 놓이면 DPI 독립적 그래픽 시스템은 흐릿한 가장자리 또는 반투명 가장자리와 같은 렌더링 아티팩트를 만들 수 있습니다. 이전 버전의 WPF에는 이러한 경우를 처리하는 데 도움이 되는 픽셀 맞추기 기능이 포함되어 있습니다. Silverlight 2에서는 가장자리가 전체 픽셀 경계에 딱 맞게 요소를 이동하는 또 다른 방법인 레이아웃 조정이 도입되었습니다. 이제 WPF는 에 UseLayoutRounding FrameworkElement연결된 속성으로 레이아웃 반올림을 지원합니다.

- 캐시된 컴퍼지션

  새 BitmapCache 클래스와 BitmapCacheBrush 클래스를 사용하면 시각적 트리의 복잡한 부분을 비트맵으로 캐시하고 렌더링 시간을 크게 향상시킬 수 있습니다. 비트맵은 마우스 클릭 같은 사용자 입력에 응답하며, 브러시처럼 다른 요소에 칠할 수 있습니다.

- 픽셀 셰이더 3 지원

  WPF 4는 응용 프로그램이 ShaderEffect 픽셀 셰더(PS) 버전 3.0을 사용하여 효과를 작성할 수 있도록 하여 WPF 3.5 SP1에 도입된 지원 위에 빌드됩니다. PS 3.0 셰이더 모델은 PS 2.0보다 더 정교해졌으며 지원되는 하드웨어에 더 많은 영향을 미칠 수 있습니다.

- 감속/가속 함수

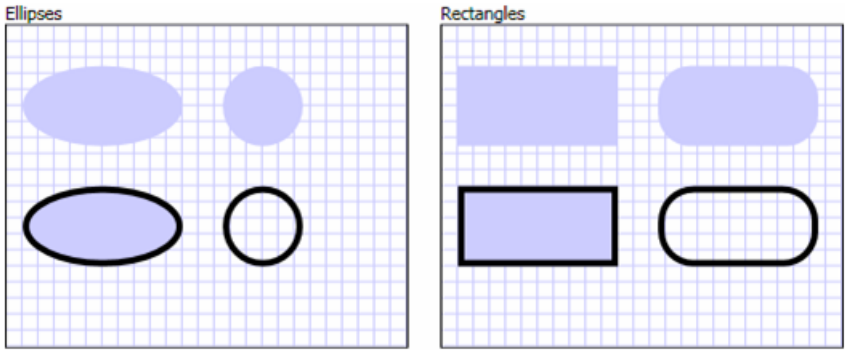
  감속/가속 함수를 사용하여 애니메이션을 개선하고 동작을 좀 더 강력히 제어할 수 있습니다. 예를 들어 애니메이션에 ElasticEase 애니메이션을 적용하여 애니메이션에 탄력적인 동작을 제공할 수 있습니다. 자세한 내용은 System.Windows.Media.Animation 네임스페이스의 감속/가속 형식을 참조하십시오.

그래픽 및 렌더링

WPF에는 고품질 2D 그래픽에 대한 지원이 포함되어 있습니다. 기능으로는 브러시, 기하 도형, 이미지, 도형 및 변환 기능이 있습니다. 자세한 내용은 그래픽을 참조하세요. 그래픽 요소의 렌더링은 클래스를 Visual 기반으로 합니다. 화면의 시각적 개체 구조는 시각적 트리로 설명됩니다. 자세한 내용은 WPF 그래픽 렌더링 개요를 참조하세요.

2D 셰이프

WPF는 다음 그림에서 보여 드리는 사각형 및 타원과 같이 일반적으로 사용되는 벡터 로그린 2D 셰이프 라이브러리를 제공합니다.



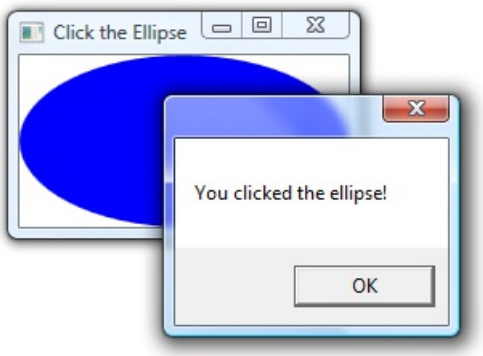
이러한 고유 WPF 셰이프는 단순한 셰이프가 아니라 키보드 및 마우스 입력을 포함한 가장 일반적인 컨트롤에서 기대하는 많은 기능을 구현하는 프로그래밍 가능한 요소입니다. 다음 예제에서는 MouseUp Ellipse 요소를 클릭하여 발생한 이벤트를 처리하는 방법을 보여 주며 있습니다.

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="Window1" >
  <Ellipse Fill="LightBlue" MouseUp="ellipseButton_MouseUp" />
</Window>
```

```
public partial class Window1  : Window
{
    void ellipseButton_MouseUp(object sender, MouseButtonEventArgs e)
    {
        MessageBox.Show("You clicked the ellipse!");
    }
}
```

```
Partial Public Class Window1
    Inherits Window
    Private Sub ellipseButton_MouseUp(ByVal sender As Object, ByVal e As MouseButtonEventArgs)
        MessageBox.Show("You clicked the ellipse!")
    End Sub
End Class
```

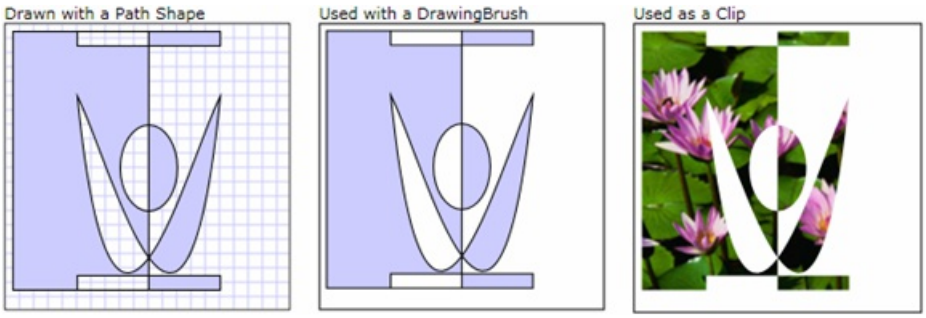
다음 그림에서는 이전 XAML 태그 및 코드 숨김에 대한 출력을 보여 줍니다.



자세한 내용은 WPF에서 Shape 및 기본 그리기 개요를 참조하세요. 기본 샘플을 보려면 도형 요소 샘플을 참조하세요.

2D 지오메트리

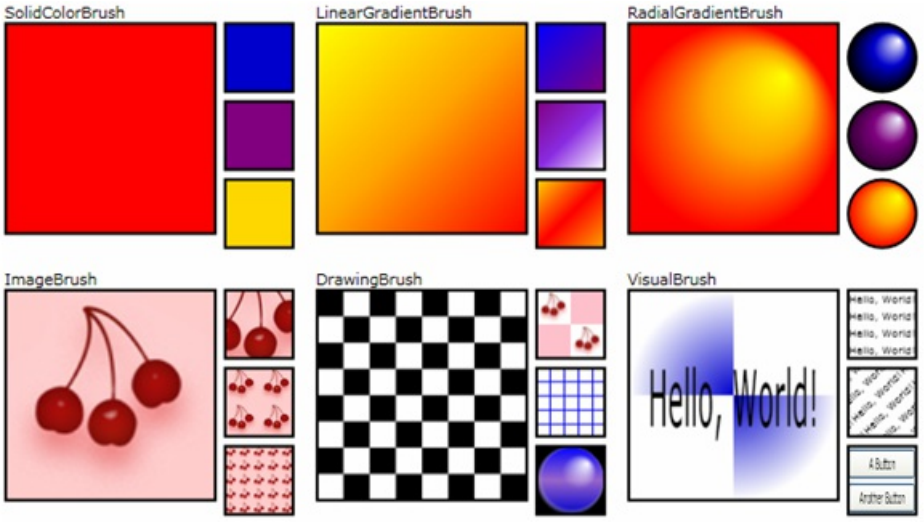
WPF가 제공하는 2D 셰이프가 충분하지 않은 경우 형상 및 패스에 WPF 지원을 사용하여 직접 만들 수 있습니다. 다음 그림에서는 형상을 사용하여 셰이프를 만들고, 드로잉 브러시로, 다른 WPF 요소를 잘라내는 방법을 보여 주어집니다.



자세한 내용은 지오메트리 개요를 참조하십시오. 기본 샘플을 보려면 기하 도형 샘플을 참조하세요.

2D 효과

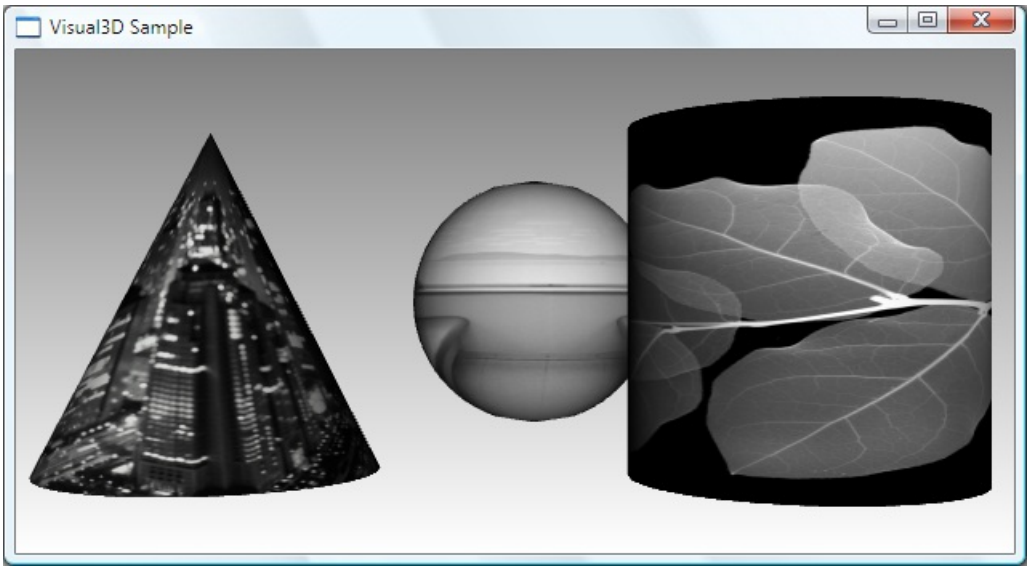
WPF는 다양한 효과를 만드는 데 사용할 수 있는 2D 클래스 라이브러리를 제공합니다. WPF의 2D 렌더링 기능은 그라데이션, 비트맵, 드로잉 및 비디오가 있는 요소를 페인칠하는 UI 기능을 제공합니다. 회전, 배율 조정 및 기울이기를 사용하여 조작할 수 있습니다. 다음 그림에서는 WPF 브러시를 사용하여 얻을 수 있는 다양한 효과의 예를 제공합니다.



자세한 내용은 WPF 브러시 개요를참조하십시오. 기본 샘플을 보려면 브러시 샘플을 참조하세요.

3D 렌더링

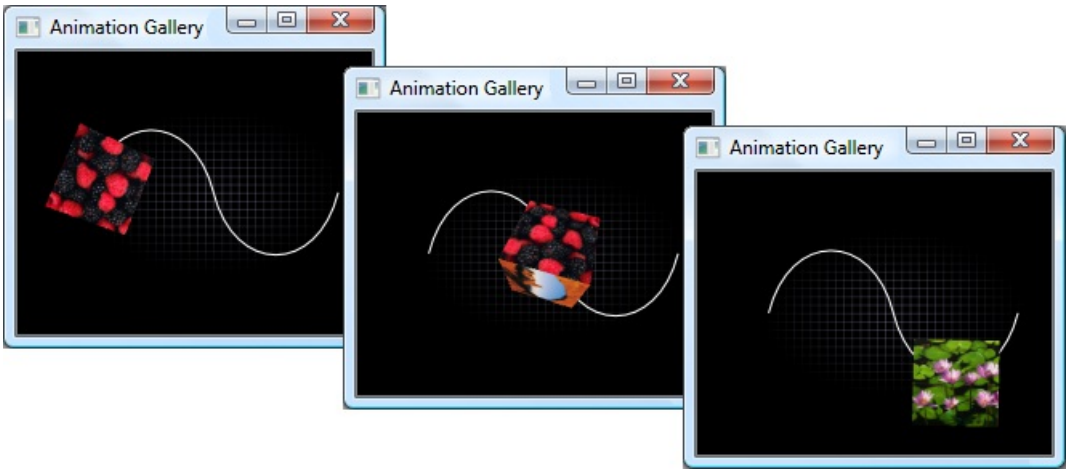
WPF는 WPF의 2D 그래픽 지원과 통합되는 일련의 3D 렌더링 기능을 제공하여 보다 UI흥미로운 레이아웃, 데이터 시각화를 만들 수 있습니다. 스펙트럼의 한쪽 끝에서 WPF를 사용하면 다음 그림에서 보여 주는 3D 셰이프의 표면에 2D 이미지를 렌더링할 수 있습니다.



자세한 내용은 3D 그래픽 개요를참조하십시오. 소개 샘플은 3D 솔리드 샘플을참조하십시오.

애니메이션

애니메이션으로 컨트롤 및 요소가 커지거나, 흔들리거나, 회전하거나, 사라지도록 하여 흥미로운 페이지 전환 등을 만들 수 있습니다. WPF를 사용하면 대부분의 속성에 애니메이션을 만들 수 있을 뿐만 아니라 대부분의 WPF 개체에 애니메이션을 만들 수 있으므로 WPF를 사용하여 만든 사용자 지정 개체에 애니메이션을 애니메이션할 수도 있습니다.



자세한 내용은 애니메이션 개요를참조하십시오. 기본 샘플을 보려면 애니메이션 예제 갤러리를 참조하세요.

미디어

이미지, 비디오 및 오디오는 미디어를 통해 정보 및 사용자 환경을 전달하는 방법입니다.

이미지

아이콘, 배경 및 애니메이션 일부를 포함하는 이미지는 대부분의 애플리케이션에서 핵심적인 부분입니다. 이미지를 자주 사용해야 하기 때문에 WPF는 다양한 방법으로 이미지를 사용할 수 있는 기능을 노출합니다. 다음 그림에서는 해당 방법 중 하나만 보여 줍니다.

자세한 내용은 이미징 개요를참조하십시오.

비디오 및 오디오

WPF 의 그래픽 기능의 핵심 기능은 비디오 및 오디오를 포함하는 멀티미디어 작업에 대한 기본 지원을 제공하는 것입니다. 다음 예제에서는 미디어 플레이어를 애플리케이션에 삽입하는 방법을 보여 줍니다.

```
<MediaElement Source="media\numbers.wmv" Width="450" Height="250" />
```

MediaElement비디오와 오디오를 모두 재생할 수 있으며 사용자 지정 UI를 쉽게 만들 수 있을 만큼 확장할 수 있습니다.

자세한 내용은 멀티미디어 개요를 참조하세요.

참고 항목

- System.Windows.Media
- System.Windows.Media.Animation
- System.Windows.Media.Media3D
- 2D 그래픽 및 이미징
- WPF에서 Shape 및 기본 그리기 개요
- 단색 및 그라데이션을 사용한 그리기 개요
- 이미지, 그림 및 시각적 표시로 그리기
- 애니메이션 및 타이밍 방법 항목
- 3D 그래픽 개요
- 멀티미디어 개요

보안(WPF)

2020-03-21 • 36 minutes to read • Edit Online

WPF(Windows 프레젠테이션 파운데이션) 독립 실행형 및 브라우저 호스팅 응용 프로그램을 개발할 때는 보안 모델을 고려해야 합니다. WPF 독립 실행형 응용 프로그램은 무제한 권한(CAS**FullTrust** 권한 집합)으로 실행되며, Windows 설치 관리자(.msi), XCopy 또는 ClickOnce를 사용하여 배포되었는지 여부입니다. ClickOnce를 포함한 부분 신뢰, 독립 실행형 WPF 애플리케이션 배포가 지원되지 않습니다. 그러나 완전 신뢰 호스트 응용 프로그램은 .NET AppDomain Framework 추가 기능 모델을 사용하여 부분 신뢰를 만들 수 있습니다. 자세한 내용은 WPF 추가 기능 개요를참조하십시오.

WPF 브라우저 호스팅 응용 프로그램은 Windows Internet Explorer 또는 Firefox에서 호스팅하며 XAML 브라우저 응용 XAML(Extensible Application Markup Language) 프로그램(XBAPs) WPF XAML Browser Applications Overview또는 느슨한 문서일 수 있습니다.

WPF 브라우저에서 호스팅되는 응용 프로그램은 기본적으로 기본 CAS**인터넷** 영역 사용 권한 집합으로 제한되는 부분 신뢰 보안 샌드박스 내에서 실행됩니다. 이렇게 하면 일반적인 웹 응용 프로그램이 격리될 것으로 예상되는 것과 동일한 방식으로 WPF 브라우저 호스팅 응용 프로그램을 클라이언트 컴퓨터에서 격리할 수 있습니다. XBAP는 배포 URL 및 클라이언트의 보안 구성의 보안 영역에 따라 권한을 완전 신뢰까지 높일 수 있습니다. 자세한 내용은 WPF 부분 신뢰 보안을 참조하세요.

이 항목에서는 WPF(Windows 프레젠테이션 Foundation) 독립 실행형 및 브라우저 호스팅 응용 프로그램에 대한 보안 모델에 대해 설명합니다.

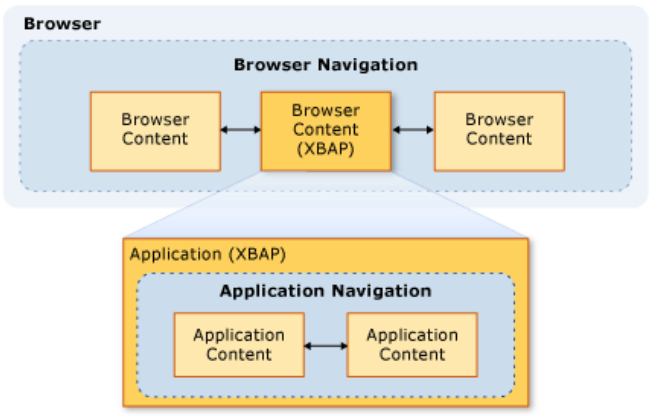
이 항목에는 다음과 같은 섹션이 포함되어 있습니다.

- 안전한 탐색
- 웹 브라우징 소프트웨어 보안 설정
- WebBrowser 컨트롤 및 기능 컨트롤
- 부분적으로 신뢰할 수 있는 클라이언트 애플리케이션에 대한 APTCA 어셈블리를 사용하지 않도록 설정
- XAML 사용 완화 파일에 대한 샌드박스 동작
- 보안을 승격하는 WPF 애플리케이션 개발을 위한 리소스

안전한 탐색

XBAPs의 경우 WPF는 응용 프로그램과 브라우저의 두 가지 유형의 탐색을 구분합니다.

*애플리케이션 탐색*은 브라우저에서 호스트되는 애플리케이션 내의 콘텐츠 항목 간을 탐색합니다. *브라우저 탐색*은 브라우저 자체의 콘텐츠 및 위치 URL을 변경하는 탐색입니다. 응용 프로그램 탐색(일반적으로 XAML)과 브라우저 탐색(일반적으로 HTML) 간의 관계는 다음 그림에 나와 있습니다.



XBAP가 탐색하기에 안전한 것으로 간주되는 콘텐츠 유형은 주로 응용 프로그램 탐색 또는 브라우저 탐색이 사용되는지 여부에 따라 결정됩니다.

애플리케이션 탐색 보안

응용 프로그램 탐색은 다음 네 가지 유형의 콘텐츠를 지원하는 pack URI로 식별할 수 있는 경우 안전한 것으로 간주됩니다.

콘텐츠 형식	DESCRIPTION	URI 예제
리소스	**리소스의** 빌드 유형이 있는 프로젝트에 추가되는 파일 .	`pack://application:,,,/MyResourceFile.xaml`
콘텐츠	**콘텐츠의** 빌드 유형이 있는 프로젝트에 추가되는 파일 .	`pack://application:,,,/MyContentFile.xaml`
원래 사이트	없음의 빌드 유형이 있는 프로젝트에 **None**추가되는 파일입니다.	`pack://siteoforigin:,,,/MySiteOfOriginFile.x`
애플리케이션 코드	컴파일된 코드 숨김이 있는 XAML 리소스. 또는 **페이지**의 빌드 유형이 있는 프로젝트에 추가되는 XAML 파일 .	`pack://application:,,,/MyResourceFile.xaml`

> **NOTE**
> 응용 프로그램 데이터 파일 및 팩 URI에 대한 자세한 내용은 WPF 응용 프로그램 리소스, 콘텐츠 및 데이터 파일을참조하십시오.

이러한 콘텐츠 형식의 파일은 사용자 또는 프로그래밍 방식으로 탐색할 수 있습니다.

- **사용자 탐색**. 사용자는 요소를 클릭하여 탐색합니다. Hyperlink
- **프로그래밍 방식 탐색**. 응용 프로그램은 예를 들어 속성을 설정하여 사용자를 NavigationWindow.Source 포함하지 않고 탐색합니다.

브라우저 탐색 보안

브라우저 탐색은 다음 조건에서만 안전한 것으로 간주됩니다.

- **사용자 탐색**. 사용자는 중첩된 Hyperlink 가 아닌 main NavigationWindow내에 있는 요소를 클릭하여 Frame탐색합니다.
- **영역**. 탐색 대상 콘텐츠는 인터넷 또는 로컬 인트라넷에 있습니다.
- **프로토콜**. 사용 중인 프로토콜은 **http**, **https**, **파일**또는 **mailto**입니다.

XBAP가 이러한 조건을 준수하지 않는 방식으로 콘텐츠로 이동하려고 하면 SecurityException a가 throw됩니다.

웹 브라우징 소프트웨어 보안 설정

컴퓨터의 보안 설정은 웹 브라우징 소프트웨어 권한이 부여된 액세스 권한을 결정합니다. 웹 브라우징 소프트웨어에는 인터넷 익스플로러 및 PresentationHost.exe를 포함하여 WinINet 또는 UrlMon API를 사용하는 모든 응용 프로그램 또는 구성 요소가 포함됩니다.

Internet Explorer는 다음을 포함하여 Internet Explorer에서 실행하거나 실행할 수 있는 기능을 구성할 수 있는 메커니즘을 제공합니다.

- .NET 프레임워크 기반 구성 요소
- ActiveX 컨트롤 및 플러그인

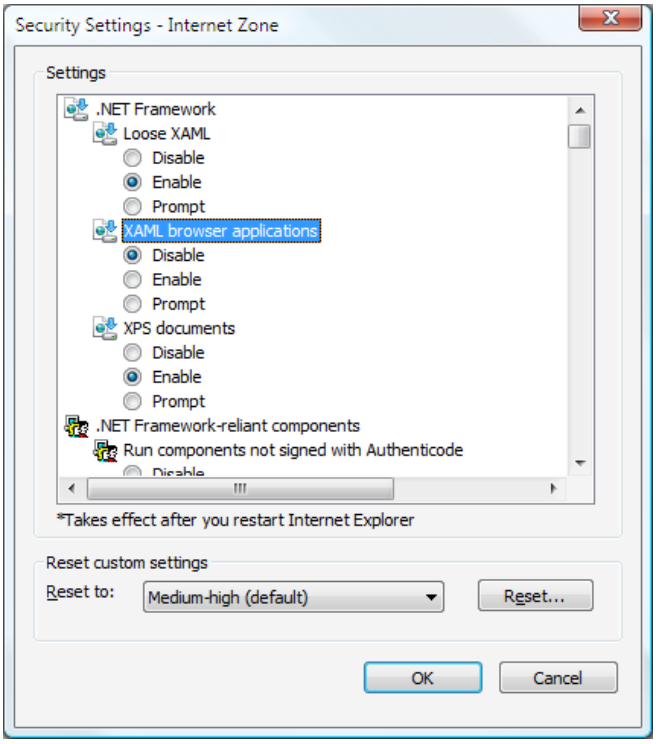
- 다운로드
- 스크립팅
- 사용자 인증

이러한 방식으로 보호할 수 있는 기능 모음은 **인터넷**, **인트라넷**, **신뢰할 수 있는 사이트**및 제한된 **사이트** 영역에 대해 영역별로 구성됩니다. 다음 단계는 보안 설정을 구성하는 방법을 설명합니다.

1. **제어판**을 엽니다.
2. **네트워크 및 인터넷을** 클릭한 다음 인터넷 옵션을 **클릭합니다.**

   인터넷 옵션 대화 상자가 나타납니다.
3. **보안** 탭에서 영역을 선택하여 보안 설정을 구성합니다.
4. 사용자 지정 수준 버튼을 **클릭합니다.**

   보안 설정 대화 상자가 나타나고 선택한 영역에 대한 보안 설정을 구성할 수 있습니다.



> **NOTE**
> Internet Explorer에서 인터넷 옵션 대화 상자에 액세스할 수도 있습니다. **도구**를 클릭한 다음 **인터넷 옵션을 클릭합니다.**

Windows 인터넷 익스플로러 7부터 .NET Framework를 위한 다음 보안 설정이 포함되어 있습니다.

- **느슨한 XAML**. Internet Explorer가 파일로 탐색하고 파일을 느슨하게 XAML 할 수 있는지 여부를 제어합니다. (설정, 해제 및 확인 옵션).
- **XAML 브라우저 애플리케이션**. Internet Explorer에서 XBAP로 이동하여 실행할 수 있는지 여부를 제어합니다. (설정, 해제 및 확인 옵션).

기본적으로 이러한 설정은 **모두 인터넷, 로컬 인트라넷**및 **신뢰할 수 있는 사이트** 영역에 대해 활성화되고 제한된 **사이트** 영역에 대해 비활성화됩니다.

보안 관련 **WPF** 레지스트리 설정

인터넷 옵션을 통해 사용할 수 있는 보안 설정 외에 다음 레지스트리 값은 선택적으로 다양한 보안 관련 WPF 기능을 차단하는 데 사용할 수 있습니다. 해당 값은 다음 키에서 정의됩니다.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\Windows Presentation Foundation\Features
```

다음 표에서는 설정할 수 있는 값을 보여 줍니다.

값 이름	값 형식	값 데이터
XBAPDisallow	REG_DWORD	1은 허용되지 않으며 0은 허용됩니다.
LooseXamlDisallow	REG_DWORD	1은 허용되지 않으며 0은 허용됩니다.
WebBrowserDisallow	REG_DWORD	1은 허용되지 않으며 0은 허용됩니다.
MediaAudioDisallow	REG_DWORD	1은 허용되지 않으며 0은 허용됩니다.
MediaImageDisallow	REG_DWORD	1은 허용되지 않으며 0은 허용됩니다.
MediaVideoDisallow	REG_DWORD	1은 허용되지 않으며 0은 허용됩니다.
ScriptInteropDisallow	REG_DWORD	1은 허용되지 않으며 0은 허용됩니다.

WebBrowser 컨트롤 및 기능 컨트롤

WPF WebBrowser 컨트롤을 사용하여 웹 콘텐츠를 호스팅할 수 있습니다. WPF WebBrowser 컨트롤은 기본 웹 브라우저 ActiveX 컨트롤을 래핑합니다. WPF는 신뢰할 수 없는 웹 콘텐츠를 호스트하기 위해 WPF WebBrowser 컨트롤을 사용할 때 응용 프로그램 보안을 위한 몇 가지 지원을 제공합니다. 그러나 일부 보안 기능은 WebBrowser 컨트롤을 사용하는 응용 프로그램에서 직접 적용해야 합니다. WebBrowser ActiveX 컨트롤에 대한 자세한 내용은 웹 브라우저 제어 개요 및 자습서를참조하십시오.

> **NOTE**
> 이 섹션은 HTML Frame 콘텐츠로 이동하는 WebBrowser 데 를 사용하기 때문에 컨트롤에도 적용됩니다.

WPF WebBrowser 컨트롤이 신뢰할 수 없는 웹 콘텐츠를 호스트하는 데 사용되는 AppDomain 경우 응용 프로그램은 부분 트러스트를 사용하여 응용 프로그램 코드를 악의적인 HTML 스크립트 코드로부터 격리해야 합니다. 응용 프로그램이 InvokeScript 메서드 및 ObjectForScripting 속성을 사용하여 호스팅된 스크립트와 상호 작용하는 경우 특히 그렇습니다. 자세한 내용은 WPF 추가 기능 개요를참조하십시오.

응용 프로그램에서 WPF WebBrowser 컨트롤을 사용하는 경우 보안을 강화하고 공격을 완화하는 또 다른 방법은 Internet Explorer 기능 컨트롤을 사용하도록 설정하는 것입니다. 기능 컨트롤은 관리자와 개발자가 WPF WebBrowser 컨트롤이 래핑하는 WebBrowser ActiveX 컨트롤을 호스팅하는 인터넷 익스플로러 및 응용 프로그램의 기능을 구성할 수 있도록 하는 Internet Explorer에 추가된 기능입니다. 기능 컨트롤은 CoInternetSetFeatureEnabled 기능을 사용하거나 레지스트리의 값을 변경하여 구성할 수 있습니다. 피처 컨트롤에 대한 자세한 내용은 기능 컨트롤 및 인터넷 기능 컨트롤 소개를 참조하십시오.

WPF WebBrowser 컨트롤을 사용하는 독립 실행형 WPF 응용 프로그램을 개발하는 경우 WPF는 응용 프로그램에 대해 다음 기능 컨트롤을 자동으로 활성화합니다.

기능 컨트롤
FEATURE_MIME_HANDLING
FEATURE_MIME_SNIFFING
FEATURE_OBJECT_CACHING
FEATURE_SAFE_BINDTOOBJECT

기능 컨트롤
FEATURE_WINDOW_RESTRICTIONS
FEATURE_ZONE_ELEVATION
FEATURE_RESTRICT_FILEDOWNLOAD
FEATURE_RESTRICT_ACTIVEXINSTALL
FEATURE_ADDON_MANAGEMENT
FEATURE_HTTP_USERNAME_PASSWORD_DISABLE
FEATURE_SECURITYBAND
FEATURE_UNC_SAVEDFILECHECK
FEATURE_VALIDATE_NAVIGATE_URL
FEATURE_DISABLE_TELNET_PROTOCOL
FEATURE_WEBOC_POPUPMANAGEMENT
FEATURE_DISABLE_LEGACY_COMPRESSION
FEATURE_SSLUX

이러한 기능 컨트롤은 조건에 상관 없이 사용되므로 이 컨트롤로 인해 완전 신뢰 애플리케이션가 손상될 수 있습니다. 이 경우 특정 애플리케이션 및 호스팅하는 콘텐츠에 대한 보안 위험이 없다면, 해당 기능 컨트롤을 비활성화할 수 있습니다.

기능 컨트롤은 WebBrowser ActiveX 개체를 인스턴스화하는 프로세스에 의해 적용됩니다. 따라서 신뢰할 수 없는 콘텐츠를 탐색할 수 있는 독립 실행형 애플리케이션을 만드는 경우, 추가 기능 컨트롤 사용을 심각하게 고려해야 합니다.

> **NOTE**
> 이 권장 사항은 MSHTML 및 SHDOCVW 호스트 보안을 위한 일반 권장 사항을 기반으로 합니다. 자세한 내용은 MSHTML 호스트 보안 FAQ: II 파트 I 및 MSHTML 호스트 보안 FAQ: II 의 파트 II를참조하십시오.

실행 파일의 경우, 레지스트리 값을 1로 설정하여 다음 기능 컨트롤을 사용하도록 설정하는 것이 좋습니다.

기능 컨트롤
FEATURE_ACTIVEX_REPURPOSEDETECTION
FEATURE_BLOCK_LMZ_IMG
FEATURE_BLOCK_LMZ_OBJECT
FEATURE_BLOCK_LMZ_SCRIPT
FEATURE_RESTRICT_RES_TO_LMZ
FEATURE_RESTRICT_ABOUT_PROTOCOL_IE7

기능 컨트롤
FEATURE_SHOW_APP_PROTOCOL_WARN_DIALOG
FEATURE_LOCALMACHINE_LOCKDOWN
FEATURE_FORCE_ADDR_AND_STATUS
FEATURE_RESTRICTED_ZONE_WHEN_FILE_NOT_FOUND

실행 파일의 경우, 레지스트리 값을 0으로 설정하여 다음 기능 컨트롤을 사용하지 않도록 설정하는 것이 좋습니다.

기능 컨트롤
FEATURE_ENABLE_SCRIPT_PASTE_URLACTION_IF_PROMPT

Windows 인터넷 익스플로러에서 WPF WebBrowser 컨트롤을 포함하는 부분 신뢰 XAML 브라우저 응용 프로그램(XBAP)을 실행하는 경우 WPF는 인터넷 익스플로러 프로세스의 주소 공간에서 WebBrowser ActiveX 컨트롤을 호스팅합니다. WebBrowser ActiveX 컨트롤은 인터넷 익스플로러 프로세스에서 호스팅되므로 Internet Explorer에 대한 모든 기능 컨트롤도 WebBrowser ActiveX 컨트롤에 대해 활성화됩니다.

또한 Internet Explorer에서 실행하는 XBAP는 일반 독립 실행형 애플리케이션보다 추가된 보안 수준이 제공됩니다. 이 추가 보안은 인터넷 익스플로러, 따라서 WebBrowser ActiveX 컨트롤, 기본적으로 Windows Vista 및 Windows 7에서 보호 모드에서 실행 하기 때문에. 보호 모드에 대한 자세한 내용은 보호 모드 인터넷 익스플로러에서 이해 및 작업작업을 참조하십시오.

> **NOTE**
> 파이어 폭스에서 WPF WebBrowser 컨트롤을 포함 하는 XBAP를 실행 하려고 하는 SecurityException 경우, 인터넷 영역에 있는 동안, 던져집니다. 이는 WPF 보안 정책에 의한 것입니다.

부분적으로 신뢰할 수 있는 클라이언트 애플리케이션에 대한 APTCA 어셈블리를 사용하지 않도록 설정

관리되는 어셈블리가 GAC(전역 어셈블리 캐시)에 설치되면 사용자가 설치할 수 있는 명시적 권한을 제공해야 하므로 완전히 신뢰할 수 있습니다. 완전히 신뢰할 수 있기 때문에 완전히 신뢰할 수 있는 관리 클라이언트 애플리케이션에서 사용할 수 있습니다. 부분적으로 신뢰할 수 있는 응용 프로그램을 사용할 수 있도록 AllowPartiallyTrustedCallersAttribute 하려면 APTCA(APTCA)로 표시해야 합니다. 부분 신뢰로 실행하기에 안전한 것으로 테스트된 어셈블리만 이 특성으로 표시되어야 합니다.

그러나 APTCA 어셈블리가 GAC에 설치한 후 보안 결함을 나타낼 수 있습니다. 보안 결함이 발견되면 어셈블리 게시자는 기존 설치에서 문제를 해결하는 보안 업데이트를 생성할 수 있으며, 문제가 발견된 후 발생할 수 있는 설치를 방지할 수 있습니다. 업데이트에 대한 한 가지 옵션은 어셈블리를 제거하는 것이지만 어셈블리를 사용하는 완전히 신뢰할 수 있는 다른 클라이언트 애플리케이션이 중단될 수 있습니다.

WPF는 APTCA 어셈블리를 제거하지 않고 부분적으로 신뢰할 수 있는 XBAP에 대해 APTCA 어셈블리를 비활성화할 수 있는 메커니즘을 제공합니다.

APTCA 어셈블리를 사용하지 않도록 설정하려면 특수한 레지스트리 키를 만들어야 합니다.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\policy\APTCA\<AssemblyFullName>, FileVersion=
<AssemblyFileVersion>
```

이에 대한 예는 다음과 같습니다.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\policy\APTCA\aptcagac, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=215e3ac809a0fea7, FileVersion=1.0.0.0
```

이 키는 APTCA 어셈블리에 대한 항목을 설정합니다. 어셈블리를 사용할지를 지정하는 이 키에서 값을 만들 수도 있습니

다. 다음은 값의 세부 정보입니다.

- 값 이름: **APTCA_FLAG**.
- 값 유형: **REG_DWORD**.
- 값 데이터: **1**을 사용하지 않도록 설정합니다. **0**을 활성화합니다.

어셈블리를 부분적으로 신뢰할 수 있는 클라이언트 애플리케이션에 사용하지 않도록 설정해야 하는 경우, 레지스트리 키와 값을 만드는 업데이트를 작성할 수 있습니다.

> **NOTE**
> Core .NET Framework 어셈블리는 관리되는 응용 프로그램을 실행하는 데 필요하기 때문에 이러한 방식으로 사용하지 않도록 설정하면 영향을 받지 않습니다. APTCA 어셈블리를 사용하지 않도록 설정하는 것에 대한 지원은 기본적으로 타사 애플리케이션을 대상으로 합니다.

느슨한 XAML 파일에 대한 샌드박스 동작

느슨한 XAML 파일은 코드 숨결, 이벤트 처리기 또는 응용 프로그램 별 어셈블리에 의존하지 않는 태그 전용 XAML 파일입니다. 느슨한 XAML 파일이 브라우저에서 직접 탐색되면 기본 인터넷 영역 사용 권한 집합에 따라 보안 샌드박스에 로드됩니다.

그러나 느슨한 XAML 파일이 독립 실행형 응용 프로그램 NavigationWindow 또는 Frame 독립 실행형 응용 프로그램에서 탐색될 때 보안 동작이 다릅니다.

두 경우 모두 XAML 호스트 응용 프로그램의 사용 권한을 상속하기 위해 탐색되는 느슨한 파일입니다. 그러나 이 동작은 보안 관점에서 바람직하지 않을 수 XAML 있으며, 특히 신뢰할 수 없거나 알 수 없는 엔터티에서 느슨한 파일이 생성된 경우 는 그렇지 않을 수 있습니다. 이러한 유형의 콘텐츠를 *외부 콘텐츠라고* Frame 하며 NavigationWindow 탐색할 때 격리되도록 구성할 수 있습니다. 격리는 다음에 대한 예제와 Frame NavigationWindow같이 **SandboxExternalContent** 속성을 true로 설정하여 수행됩니다.

```
<Frame
  Source="ExternalContentPage.xaml"
  SandboxExternalContent="True">
</Frame>
```

```
<!-- Sandboxing external content using NavigationWindow-->
<NavigationWindow
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  Source="ExternalContentPage.xaml"
  SandboxExternalContent="True">
</NavigationWindow>
```

이 설정을 통해 외부 콘텐츠가 애플리케이션을 호스팅하는 프로세스와 별개인 프로세스로 로드됩니다. 이 프로세스는 효과적으로 호스팅 애플리케이션과 클라이언트 컴퓨터에서 격리되어 기본 인터넷 영역 권한 집합으로 제한됩니다.

> **NOTE**
> 프레젠테이션 호스트 프로세스와 관련된 NavigationWindow WPF 브라우저 호스팅 인프라를 기반으로 독립 Frame 실행형 응용 프로그램에서 느슨한 XAML 파일로 탐색하는 것이 구현되더라도 콘텐츠가 Windows Vista 및 Windows 7의 Internet Explorer에서 직접 로드될 때보다 보안 수준이 약간 적습니다(여전히 PresentationHost를 통해 있음). 웹 브라우저를 사용하는 독립 실행형 WPF 애플리케이션은 Internet Explorer의 추가 보호 모드 보안 기능을 제공하지 않습니다.

보안을 승격하는 WPF 애플리케이션 개발을 위한 리소스

다음은 보안을 촉진하는 WPF 응용 프로그램을 개발하는 데 도움이 되는 몇 가지 추가 리소스입니다.

영역	리소스
관리 코드	애플리케이션에 대한 패턴 및 사례 보안 지침
CAS	코드 액세스 보안
ClickOnce	클릭원스 보안 및 배포
WPF	WPF 부분 신뢰 보안

참고 항목

- WPF 부분 신뢰 보안
- WPF 보안 전략 - 플랫폼 보안
- WPF 보안 전략 - 보안 엔지니어링
- 애플리케이션에 대한 패턴 및 사례 보안 지침
- 코드 액세스 보안
- 클릭원스 보안 및 배포
- XAML 개요(WPF)

WPF 부분 신뢰 보안

2020-03-21 • 25 minutes to read • Edit Online

일반적으로 악의적인 손상을 방지하기 위해 중요한 시스템 리소스에 직접 액세스하지 않도록 인터넷 애플리케이션을 제한해야 합니다. 기본적으로 HTML 및 클라이언트 쪽 스크립팅 언어는 중요한 시스템 리소스에 액세스할 수 없습니다. WPF(Windows 프레젠테이션 Foundation) 브라우저에서 브라우저호스팅 응용 프로그램을 시작할 수 있으므로 유사한 제한 집합을 준수해야 합니다. 이러한 제한을 적용하기 위해 WPF는 코드 액세스 보안(CAS) 및 ClickOnce(WPF 보안 전략 - 플랫폼 보안참조)를 모두 사용합니다. 기본적으로 브라우저에서 호스팅되는 응용 프로그램은 인터넷, 로컬 인트라넷 또는 로컬 컴퓨터에서 시작되는지 여부에 관계없이 인터넷 영역 CAS 사용 권한 집합을 요청합니다. 전체 권한 집합보다 적은 권한으로 실행하는 애플리케이션은 부분 신뢰로 실행된다고 할 수 있습니다.

WPF는 가능한 한 많은 기능을 부분 트러스트에서 안전하게 사용할 수 있도록 다양한 지원을 제공하며 CAS와 함께 부분 신뢰 프로그래밍에 대한 추가 지원을 제공합니다.

이 항목에는 다음과 같은 섹션이 포함되어 있습니다.

- WPF 기능 부분 신뢰 지원
- 부분 신뢰 프로그래밍
- 사용 권한 관리

WPF 기능 부분 신뢰 지원

다음 표에는 인터넷 영역 사용 권한 집합의 제한 내에서 안전하게 사용할 수 있는 WPF(Windows 프레젠테이션 Foundation)의 상위 수준 기능이 나열되어 있습니다.

표 1: 부분 신뢰에서 안전한 WPF 기능

기능 영역	기능
일반	브라우저 창 원 사이트 액세스 IsolatedStorage(512KB 제한) UIAutomation 공급자 명령 IME(입력기) 태블릿 스타일러스 및 잉크 마우스 캡처 및 이동 이벤트를 사용하여 시뮬레이션된 끌어서 놓기 OpenFileDialog XamlReader.Load를 통한 XAML Deserialization

기능 영역	기능
웹 통합	브라우저 다운로드 대화 상자 사용자가 시작한 최상위 탐색 mailto:links URI(Uniform Resource Identifier) 매개 변수 HTTPWebRequest IFRAME에 호스팅되는 WPF 콘텐츠 프레임을 사용하여 동일한 사이트 HTML 페이지 호스팅 WebBrowser를 사용하여 동일한 사이트 HTML 페이지 호스팅 웹 서비스(ASMX) 웹 서비스(Windows Communication Foundation 사용) 스크립팅 문서 개체 모델
시각적 개체	2D 및 3D 애니메이션 미디어(원본 사이트 및 도메인 간) 이미지/오디오/비디오
읽기	FlowDocuments XPS 문서 포함된 시스템 글꼴 CFF & 트루타입 글꼴
편집	맞춤법 검사 RichTextBox 일반 텍스트 및 잉크 클립보드 지원 사용자가 시작한 붙여넣기 선택한 콘텐츠 복사
컨트롤	일반 컨트롤

이 표에서는 WPF 기능을 높은 수준에서 다룹니다. 자세한 내용은 Windows SDK에서 WPF의 각 멤버에 필요한 권한을 문서화합니다. 또한 다음 기능에는 특별한 고려 사항을 포함하는 부분 신뢰 실행에 대한 자세한 정보가 있습니다.

- XAML(XAML 개요(WPF)참조)를 참조하십시오.
- 팝업(참조). System.Windows.Controls.Primitives.Popup

- 끌어서 놓기(개요 끌어서 놓기참조).
- 클립보드(참조). System.Windows.Clipboard
- 이미징(참조). System.Windows.Controls.Image
- 직렬화(참조) XamlReader.Load XamlWriter.Save
- 파일 대화 상자 열기(참조). Microsoft.Win32.OpenFileDialog

다음 표에서는 인터넷 영역 사용 권한 집합의 제한 내에서 실행하기에 안전하지 않은 WPF 기능에 대해 간략하게 설명합니다.

표 2: 부분 신뢰에서 안전하지 않은 WPF 기능

기능 영역	기능
일반	창(애플리케이션 정의 창 및 대화 상자) SaveFileDialog 파일 시스템 레지스트리 액세스 끌어서 놓기 XamlWriter.Save를 통한 XAML 직렬화 UIAutomation 클라이언트 소스 창 액세스(HwndHost) 완전한 음성 지원 Windows Forms 상호 운용성
시각적 개체	비트맵 효과 이미지 인코딩
편집	RTF(서식 있는 텍스트) 클립보드 전체 XAML 지원

부분 신뢰 프로그래밍

XBAP 응용 프로그램의 경우 기본 권한 집합을 초과하는 코드는 보안 영역에 따라 다른 동작을 갖습니다. 일부 경우에는 사용자가 설치를 시도할 때 경고를 받게 됩니다. 사용자는 설치를 계속하거나 취소하도록 선택할 수 있습니다. 다음 표에서 각 보안 영역의 동작 및 애플리케이션이 완전 신뢰를 받기 위해 수행해야 하는 작업을 설명합니다.

보안 영역	동작	완전 신뢰 얻기
수집	자동 완전 신뢰	어떤 조치가 필요하지 않습니다.
인트라넷 및 신뢰할 수 있는 사이트	완전 신뢰 확인	사용자가 프롬프트에서 소스를 볼 수 있도록 인증서로 XBAP에 로그인합니다.

보안 영역	동작	완전 신뢰 얻기
인터넷	"신뢰할 수 없음"과 함께 실패	인증서로 XBAP를 서명합니다.

> **NOTE**
> 위의 표에 설명된 동작은 ClickOnce 신뢰 배포 모델을 따르지 않는 완전 신뢰 XBAP에 대한 것입니다.

일반적으로 허용되는 권한을 초과하는 코드는, 독립 실행형 애플리케이션과 브라우저에서 호스트된 애플리케이션 간에 공유되는 일반적인 코드일 수 있습니다. CAS 및 WPF는 이 시나리오를 관리하기 위한 몇 가지 기술을 제공합니다.

CAS를 사용하여 권한 검색

경우에 따라 라이브러리 어셈블리의 공유 코드를 독립 실행형 응용 프로그램과 XBAP 모두에서 사용할 수 있습니다. 이러한 경우 코드는 애플리케이션에서 얻은 권한 집합이 허용하는 것보다 많은 권한이 필요할 수 있는 기능을 실행할 수 있습니다. 응용 프로그램은 Microsoft .NET Framework 보안을 사용하여 특정 권한이 있는지 여부를 검색할 수 있습니다. 특히 원하는 사용 권한의 인스턴스에서 메서드를 Demand 호출하여 특정 권한이 있는지 여부를 테스트할 수 있습니다. 다음 예제는 로컬 디스크에 파일을 저장하는 기능이 있는지 여부에 대해 쿼리하는 코드입니다.

```
using System.IO;
using System.IO.IsolatedStorage;
using System.Security;
using System.Security.Permissions;
using System.Windows;

namespace SDKSample
{
    public class FileHandling
    {
        public void Save()
        {
            if (IsPermissionGranted(new FileIOPermission(FileIOPermissionAccess.Write, @"c:\newfile.txt")))
            {
                // Write to local disk
                using (FileStream stream = File.Create(@"c:\newfile.txt"))
                using (StreamWriter writer = new StreamWriter(stream))
                {
                    writer.WriteLine("I can write to local disk.");
                }
            }
            else
            {
                MessageBox.Show("I can't write to local disk.");
            }
        }

        // Detect whether or not this application has the requested permission
        bool IsPermissionGranted(CodeAccessPermission requestedPermission)
        {
            try
            {
                // Try and get this permission
                requestedPermission.Demand();
                return true;
            }
            catch
            {
                return false;
            }
        }
```

```
Imports System.IO
Imports System.IO.IsolatedStorage
Imports System.Security
Imports System.Security.Permissions
Imports System.Windows

Namespace SDKSample
    Public Class FileHandling
        Public Sub Save()
            If IsPermissionGranted(New FileIOPermission(FileIOPermissionAccess.Write, "c:\newfile.txt")) Then
                ' Write to local disk
                Using stream As FileStream = File.Create("c:\newfile.txt")
                Using writer As New StreamWriter(stream)
                    writer.WriteLine("I can write to local disk.")
                End Using
                End Using
            Else
                MessageBox.Show("I can't write to local disk.")
            End If
        End Sub

        ' Detect whether or not this application has the requested permission
        Private Function IsPermissionGranted(ByVal requestedPermission As CodeAccessPermission) As Boolean
            Try
                ' Try and get this permission
                requestedPermission.Demand()
                Return True
            Catch
                Return False
            End Try
        End Function
```

```
    }
}
```

```
    End Class
End Namespace
```

응용 프로그램에 원하는 권한이 없는 경우 Demand 호출은 보안 예외를 throw합니다. 그렇지 않은 경우 사용 권한이 부여됩니다. `IsPermissionGranted`이 동작을 캡슐화하고 `true` 반환하거나 `false` 적절히 반환합니다.

점진적인 기능 저하

코드에 필요한 작업을 수행할 권한이 있는지 확인하는 것은 다른 영역에서 실행될 수 있는 코드와 관련되어 있습니다. 영역 검색이 하나의 방법이지만, 가능한 경우 사용자를 위한 대안을 제공하는 것이 좋습니다. 예를 들어 부분 신뢰 애플리케이션은 격리된 스토리지에만 파일을 만들 수 있지만, 일반적으로 완전 신뢰 애플리케이션은 사용자가 원하는 모든 곳에 파일을 만들 수가 있습니다. 파일을 만드는 코드가 완전 신뢰(독립 실행형) 애플리케이션과 부분 신뢰(브라우저에서 호스트된) 애플리케이션에서 공유되는 어셈블리에 존재하고 두 애플리케이션에서 사용자가 파일을 만들도록 하는 경우, 공유 코드는 해당 위치에 파일을 만들기 전에 부분 또는 완전 신뢰에서 실행 중인지 감지해야 합니다. 다음 코드는 두 사항을 모두 보여줍니다.

```
using System.IO;
using System.IO.IsolatedStorage;
using System.Security;
using System.Security.Permissions;
using System.Windows;

namespace SDKSample
{
    public class FileHandlingGraceful
    {
        public void Save()
        {
            if (IsPermissionGranted(new FileIOPermission(FileIOPermissionAccess.Write, @"c:\newfile.txt")))
            {
                // Write to local disk
                using (FileStream stream = File.Create(@"c:\newfile.txt"))
                using (StreamWriter writer = new StreamWriter(stream))
                {
                    writer.WriteLine("I can write to local disk.");
                }
            }
            else
            {
                // Persist application-scope property to
                // isolated storage
                IsolatedStorageFile storage = IsolatedStorageFile.GetUserStoreForApplication();
                using (IsolatedStorageFileStream stream =
                    new IsolatedStorageFileStream("newfile.txt", FileMode.Create, storage))
                using (StreamWriter writer = new StreamWriter(stream))
                {
                    writer.WriteLine("I can write to Isolated Storage");
                }
            }
        }

        // Detect whether or not this application has the requested permission
        bool IsPermissionGranted(CodeAccessPermission requestedPermission)
        {
            try
            {
                // Try and get this permission
                requestedPermission.Demand();
                return true;
            }
            catch
            {
                return false;
            }
        }
```

```
Imports System.IO
Imports System.IO.IsolatedStorage
Imports System.Security
Imports System.Security.Permissions
Imports System.Windows

Namespace SDKSample
    Public Class FileHandlingGraceful
        Public Sub Save()
            If IsPermissionGranted(New FileIOPermission(FileIOPermissionAccess.Write, "c:\newfile.txt")) Then
                ' Write to local disk
                Using stream As FileStream = File.Create("c:\newfile.txt")
                Using writer As New StreamWriter(stream)
                    writer.WriteLine("I can write to local disk.")
                End Using
                End Using
            Else
                ' Persist application-scope property to
                ' isolated storage
                Dim storage As IsolatedStorageFile = IsolatedStorageFile.GetUserStoreForApplication()
                Using stream As New IsolatedStorageFileStream("newfile.txt", FileMode.Create, storage)
                Using writer As New StreamWriter(stream)
                    writer.WriteLine("I can write to Isolated Storage")
                End Using
                End Using
            End If
        End Sub

        ' Detect whether or not this application has the requested permission
        Private Function IsPermissionGranted(ByVal requestedPermission As CodeAccessPermission) As Boolean
            Try
                ' Try and get this permission
                requestedPermission.Demand()
                Return True
            Catch
                Return False
            End Try
        End Function
```

```
    }
}
```

```
    End Class
End Namespace
```

대부분의 경우, 부분 신뢰 대체 항목을 찾을 수 있어야 됩니다.

인트라넷과 같은 제어된 환경에서는 사용자 지정 관리 프레임워크를 GAC(전역 어셈블리 캐시)에 클라이언트 기반 전체에 설치할 수 있습니다. 이러한 라이브러리는 완전 신뢰가 필요한 코드를 실행하고 부분 신뢰만 허용하는 응용 AllowPartiallyTrustedCallersAttribute 프로그램에서 참조할 수 있습니다(자세한 내용은 보안 및 WPF 보안 전략 - 플랫폼 보안참조).

브라우저 호스트 검색

CAS를 사용하여 사용 권한을 확인하는 것은 권한단위로 확인해야 할 때 적합한 기술입니다. 이 기술은 일반적인 프로세스의 일부인 예외 감지에 영향을 받으며, 일반적으로 권장되지 않고 성능 문제를 일으킬 수 있습니다. 대신 XAML 브라우저 응용 프로그램(XBAP)이 인터넷 영역 샌드박스 내에서만 BrowserInteropHelper.IsBrowserHosted 실행되는 경우 XAML 브라우저 응용 프로그램(XBAPs)에 대해 true를 반환하는 속성을 사용할 수 있습니다.

NOTE

IsBrowserHosted응용 프로그램이 브라우저에서 실행되고 있는지 여부만 구분하며 응용 프로그램이 실행 중인 사용 권한 집합은 구별하지 않습니다.

사용 권한 관리

기본적으로 XBAPs는 부분 신뢰(기본 인터넷 영역 사용 권한 집합)로 실행됩니다. 그러나 애플리케이션의 요구에 따라 기본값에서 권한 집합을 변경할 수 있습니다. 예를 들어 로컬 인트라넷에서 XBAPs를 시작한 경우 다음 표에 표시된 증가된 사용 권한 집합을 활용할 수 있습니다.

표 3: LocalIntranet 및 인터넷 권한

사용 권한	ATTRIBUTE	LOCALINTRANET	인터넷
DNS	DNS 서버 액세스	yes	예
환경 변수	읽기	yes	예
파일 대화 상자	열기	yes	yes
파일 대화 상자	제한 없음	yes	예
격리된 스토리지	사용자가 어셈블리 격리	yes	예
격리된 스토리지	알 수 없는 격리	yes	yes
격리된 스토리지	무제한 사용자 할당량	yes	예
미디어	안전한 오디오, 비디오 및 이미지	yes	yes
인쇄	기본 인쇄	yes	예
인쇄	안전 인쇄	yes	yes
반사	내보내기	yes	예
보안	관리되는 코드 실행	yes	yes
보안	부여된 권한 어셜션	yes	예
사용자 인터페이스	제한 없음	yes	예
사용자 인터페이스	안전한 최상위 창	yes	yes
사용자 인터페이스	소유 클립보드	yes	yes
웹 브라우저	HTML 안전 프레임 탐색	yes	yes

> **NOTE**
> 잘라내기 및 붙여넣기는 사용자가 시작한 경우 부분 신뢰에서만 허용됩니다.

권한을 높이는 경우 프로젝트 설정 및 ClickOnce 애플리케이션 매니페스트를 변경해야 합니다. 자세한 내용은 WPF XAML 브라우저 응용 프로그램 개요를참조하십시오. 다음 문서도 유용할 수 있습니다.

- Mage.exe (매니페스트 생성 및 편집 도구).
- MageUI.exe (매니페스트 생성 및 편집 도구, 그래픽 클라이언트).
- 보안 ClickOnce 응용 프로그램.

XBAP에 완전 신뢰가 필요한 경우 동일한 도구를 사용하여 요청된 권한을 늘릴 수 있습니다. XBAP는 로컬 컴퓨터, 인트라넷 또는 브라우저의 신뢰할 수 있거나 허용된 사이트에 나열된 URL에서 설치되고 시작된 경우에만 완전 신뢰를 받습니다. 인트라넷 또는 신뢰할 수 있는 사이트에서 애플리케이션이 설치되면, 사용자는 상승된 권한을 알리는 표준 ClickOnce 프롬프트를 받습니다. 사용자는 설치를 계속하거나 취소하도록 선택할 수 있습니다.

또는 모든 보안 영역에서 완전 신뢰 배포를 위한 ClickOnce 신뢰 배포 모델을 사용할 수 있습니다. 자세한 내용은 신뢰할 수 있는 응용 프로그램 배포 개요 및 보안을참조하세요.

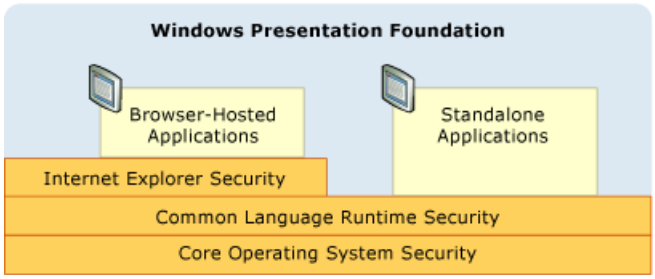
참고 항목

- 보안
- WPF 보안 전략 - 플랫폼 보안
- WPF 보안 전략 - 보안 엔지니어링

WPF 보안 전략 - 플랫폼 보안

2020-03-21 • 43 minutes to read • Edit Online

WPF(Windows 프레젠테이션 재단)는 다양한 보안 서비스를 제공하지만 운영 체제, CLR 및 Internet Explorer를 포함하는 기본 플랫폼의 보안 기능도 활용합니다. 다음 그림과 같이 이러한 계층이 결합되어 WPF에 단일 실패 지점을 피하려고 하는 강력한 심층 방어 보안 모델을 제공합니다.



이 항목의 나머지 부분에서는 WPF와 관련된 각 계층의 기능에 대해 설명합니다.

운영 체제 보안

Windows의 핵심은 WPF로 빌드된 응용 프로그램을 포함하여 모든 Windows 응용 프로그램에 대한 보안 기반을 형성하는 몇 가지 보안 기능을 제공합니다. 이 항목에서는 WPF에 중요한 이러한 보안 기능의 폭과 WPF가 이러한 보안 기능과 통합하여 심층 방어를 제공하는 방법에 대해 설명합니다.

Microsoft Windows XP SP2(서비스 팩 2)

Windows의 일반적인 검토 및 강화 외에도 이 항목에서 논의할 Windows XP SP2의 세 가지 주요 기능이 있습니다.

- /GS 컴파일
- 마이크로 소프트 윈도우 업데이트.

/GS 컴파일

Windows XP SP2는 CLR과 같은 모든 WPF 종속성을 포함하여 많은 핵심 시스템 라이브러리를 다시 컴파일하여 버퍼 오버런을 완화하여 보호합니다. 이 작업을 위해 /GS 매개 변수와 C/C++ 명령줄 컴파일러를 함께 사용합니다. 버퍼 오버런을 명시적으로 피해야 하지만 /GS 컴파일은 실수 또는 악의적으로 만들어진 잠재적인 취약성에 대한 심층 방어의 예를 제공합니다.

지금까지 버퍼 오버런은 많은 강력한 보안 익스플로이트의 원인이 되었습니다. 버퍼 오버런은 공격자는 버퍼의 경계를 지나서 쓰는 악성 코드의 삽입을 허용하는 코드 취약성을 활용하는 경우에 발생합니다. 이 경우 공격자의 코드가 실행되도록 함수의 반환 주소를 덮어써서 코드가 실행되는 프로세스를 공격자가 가로챌 수 있습니다. 그 결과, 가로챈 프로세스와 동일한 권한으로 임의 코드를 실행하는 악성 코드가 생깁니다.

상위 수준에서 -GS 컴파일러 플래그는 로컬 문자열 버퍼가 있는 함수의 반환 주소를 보호하기 위해 특수 보안 쿠키를 삽입하여 일부 잠재적인 버퍼 오버런으로부터 보호합니다. 함수가 반환된 후 보안 쿠키를 이전 값과 비교합니다. 값이 변경된 경우 버퍼 오버런이 발생했을 수 있으며 프로세스가 오류 상태로 중지됩니다. 프로세스를 중지하면 잠재적인 악성 코드의 실행이 방지됩니다. 자세한 내용은 -GS(버퍼 보안 검사)를 참조하십시오.

WPF는 WPF 응용 프로그램에 또 다른 방어 계층을 추가하기 위해 /GS 플래그로 컴파일됩니다.

Windows Vista

Windows Vista의 WPF 사용자는 "최소 권한 사용자 액세스", 코드 무결성 검사 및 권한 격리를 비롯한 운영 체제의 추가 보안 향상의 이점을 누릴 수 있습니다.

UAC(사용자 계정 컨트롤**)**

오늘날 Windows 사용자는 많은 응용 프로그램에서 설치 또는 실행 또는 둘 다에 대해 관리자 권한을 필요로 하기 때문에 관리자 권한으로 실행하는 경향이 있습니다. 한 가지 예로 기본 애플리케이션 설정을 레지스트리에 쓸 수 있습니다.

관리자 권한으로 실행은 실제로 관리자 권한이 부여된 프로세스에서 애플리케이션이 실행됨을 의미합니다. 이 경우 보안에 미치는 영향은 관리자 권한으로 실행되는 프로세스를 가로챈 악성 코드가 중요한 시스템 리소스에 대한 액세스를 포함하여 해당 권한을 자동으로 상속하게 된다는 것입니다.

이 보안 위협으로부터 보호하는 한 가지 방법은 필요한 최소한의 권한으로 애플리케이션을 실행하는 것입니다. 이를 최소 권한 원칙이라고 하며 Windows 운영 체제의 핵심 기능입니다. 이 기능을 UAC(사용자 계정 제어)라고 하며 Windows UAC에서 다음과 같은 두 가지 주요 방법으로 사용됩니다.

- 사용자가 관리자인 경우에도 기본적으로 대부분의 애플리케이션을 UAC 권한으로 실행합니다. 관리자 권한이 필요한 애플리케이션만 관리자 권한으로 실행됩니다. 관리자 권한으로 실행하려면 애플리케이션 매니페스트에서 또는 보안 정책의 항목으로 애플리케이션에 명시적으로 표시해야 합니다.
- 가상화와 같은 호환성 솔루션을 제공합니다. 예를 들어 많은 애플리케이션이 C:\Program Files와 같은 제한된 위치에 쓰려고 합니다. UAC로 실행되는 애플리케이션의 경우 쓰는 데 관리자 권한이 필요하지 않은 사용자별 대체 위치가 있습니다. UAC로 실행되는 애플리케이션의 경우 해당 위치에 쓰고 있다고 생각하는 애플리케이션이 실제로는 사용자별 대체 위치에 쓰도록 UAC에서 C:\Program Files를 가상화합니다. 이러한 종류의 호환성 작업을 통해 운영 체제가 UAC에서 이전에 실행할 수 없었던 많은 애플리케이션을 실행할 수 있습니다.

코드 무결성 검사

Windows Vista는 더 깊은 코드 무결성 검사를 통합하여 악성 코드가 로드/런타임에 시스템 파일이나 커널에 주입되는 것을 방지합니다. 이는 시스템 파일 보호 수준을 벗어납니다.

브라우저에서 호스트된 애플리케이션에 대한 제한된 권한 프로세스

브라우저에서 호스팅되는 WPF 응용 프로그램은 인터넷 영역 샌드박스 내에서 실행됩니다. Microsoft 인터넷 익스플로러와의 WPF 통합은 추가 지원을 통해 이 보호 기능을 확장합니다.

XAML 브라우저 응용 프로그램(XBAPs)은 일반적으로 인터넷 영역 권한 집합에 의해 샌드박스되므로 이러한 권한을 제거해도 XAML 브라우저 응용 프로그램(XBAPs)은 호환성 측면에서 해를 끼치지 않습니다. 대신, 추가 심층 방어 계층이 만들어집니다. 샌드박스 애플리케이션이 다른 계층을 악용하고 프로세스를 가로챌 수 있는 경우 프로세스에 여전히 제한된 권한만 포함됩니다.

최소 권한 사용자 계정 사용을참조하십시오.

공용 언어 런타임 보안

공통 언어 런타임(CLR)은 유효성 검사 및 확인, CAS(코드 액세스 보안) 및 보안에 중요한 방법론을 포함하는 여러 가지 주요 보안 이점을 제공합니다.

유효성 검사 및 확인

어셈블리 격리 및 무결성을 제공하기 위해 CLR은 유효성 검사 프로세스를 사용합니다. CLR 유효성 검사는 어셈블리 외부를 가리키는 주소에 대한 PE(이식 실행 파일) 파일 형식의 유효성을 검사하여 어셈블리가 격리되도록 합니다. CLR 유효성 검사는 어셈블리 내에 포함된 메타데이터의 무결성도 검증합니다.

형식 안전성을 보장하고, 일반적인 보안 문제(예: 버퍼 오버런)를 방지하고, 하위 프로세스 격리를 통해 샌드박싱을 활성화하기 위해 CLR 보안은 확인 개념을 사용합니다.

관리되는 애플리케이션은 MSIL(Microsoft Intermediate Language)로 컴파일됩니다. 관리되는 애플리케이션의 메서드를 실행하면 해당 MSIL이 JIT(Just-In-Time) 컴파일을 통해 네이티브 코드로 컴파일됩니다. JIT 컴파일에는 코드에서 다음이 발생하지 않도록 하는 다양한 안정성 및 견고성 규칙을 적용하는 검증 프로세스가 포함됩니다.

- 형식 계약 위반

- 버퍼 오버런 도입
- 무분별한 메모리 액세스

검증 규칙을 준수하지 않는 관리 코드는 신뢰할 수 있는 코드로 간주되지 않을 경우 실행할 수 없습니다.

확인 가능한 코드의 장점은 WPF가 .NET 프레임워크를 기반으로 하는 중요한 이유입니다. 검증할 수 있는 코드를 사용하면 가능한 취약성이 악용될 가능성이 훨씬 줄어듭니다.

코드 액세스 보안

클라이언트 컴퓨터는 파일 시스템, 레지스트리, 인쇄 서비스, 사용자 인터페이스, 리플렉션 및 환경 변수를 포함하여 관리되는 애플리케이션이 액세스할 수 있는 다양한 리소스를 노출합니다. 관리되는 응용 프로그램이 클라이언트 컴퓨터의 리소스에 액세스하려면 .NET Framework 권한이 있어야 합니다. CAS의 CodeAccessPermission권한은; CAS는 관리되는 응용 프로그램이 액세스할 수 있는 각 리소스에 대해 하나의 하위 클래스를 구현합니다.

실행을 시작할 때 관리되는 응용 프로그램이 부여되는 권한 집합을 사용 권한 집합이라고 하며 응용 프로그램에서 제공하는 증명 파일에 의해 결정됩니다. WPF 응용 프로그램의 경우 제공된 증거는 응용 프로그램이 시작되는 위치 또는 영역입니다. CAS는 다음 영역을 식별합니다.

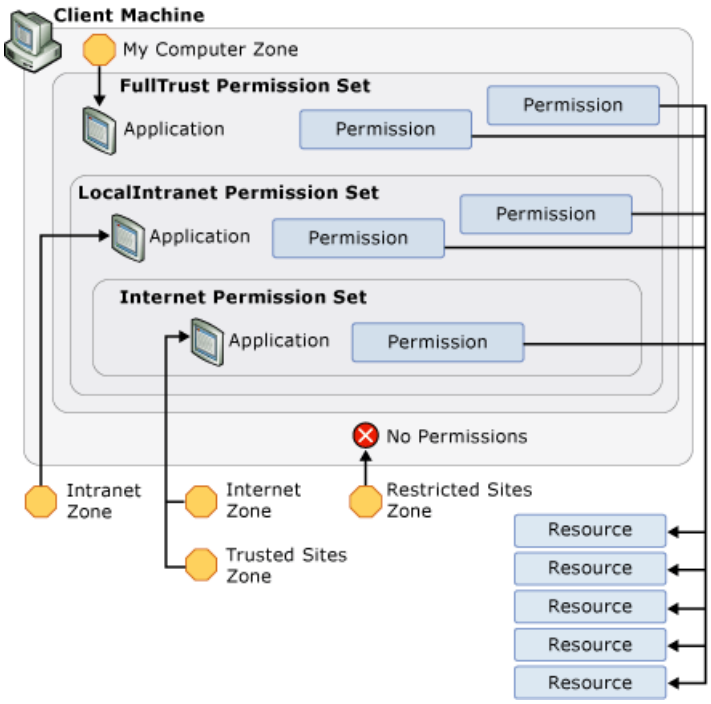
- **내 컴퓨터**. 클라이언트 컴퓨터에서 시작된 애플리케이션입니다(완전 신뢰).
- **로컬 인트라넷** 인트라넷에서 시작된 애플리케이션입니다. (다소 신뢰).
- **인터넷**. 인터넷에서 시작된 애플리케이션입니다. (최소 신뢰).
- **신뢰할 수 있는 사이트** 사용자가 신뢰할 수 있는 것으로 식별한 애플리케이션입니다. (최소 신뢰).
- **신뢰할 수 없는 사이트** 사용자가 신뢰할 수 없는 것으로 식별한 애플리케이션입니다. (신뢰할 수 없음).

이러한 각 영역에 대해 CAS는 각각과 연결된 신뢰 수준과 일치하는 사용 권한을 포함하는 미리 정의된 권한 집합을 제공합니다. 이러한 개체는 다음과 같습니다.

- **FullTrust 내 컴퓨터** 영역에서 시작된 응용 프로그램의 경우. 가능한 모든 권한은 부여됩니다.
- **로컬인트라넷**. **로컬 인트라넷** 영역에서 시작된 응용 프로그램의 경우 격리된 스토리지, 무제한 UI 액세스, 무제한 파일 대화 상자, 제한된 리플렉션, 환경 변수에 대한 제한된 액세스를 포함하여 클라이언트 머신의 리소스에 대해 보통 액세스 권한을 제공하도록 권한 하위 집합이 부여됩니다. 레지스트리와 같은 중요한 리소스에 대한 권한은 제공되지 않습니다.
- **인터넷**. **인터넷** 또는 신뢰할 수 **있는 사이트** 영역에서 시작된 응용 프로그램의 경우 격리된 스토리지, 파일 열기 전용 및 제한된 UI를 포함하여 클라이언트 머신의 리소스에 대해 제한된 액세스 권한을 제공하도록 권한 하위 집합이 부여됩니다. 기본적으로 이 권한 집합은 클라이언트 컴퓨터에서 응용 프로그램을 격리합니다.

신뢰할 수 없는 사이트 영역에서 확인된 응용 프로그램에는 CAS에 의해 사용 권한이 전혀 부여되지 않습니다. 따라서 미리 정의된 해당 권한 집합이 없습니다.

다음 그림에서는 영역, 사용 권한 집합, 사용 권한 및 리소스 간의 관계를 보여 줍니다.



인터넷 영역 보안 샌드박스의 제한은 XBAP가 WPF를 포함하여 시스템 라이브러리에서 가져오는 모든 코드에도 동일하게 적용됩니다. 이렇게 하면 코드의 모든 비트가 잠겨 있고 WPF도 잠깁니다. 안타깝게도 XBAP를 실행하려면 인터넷 영역 보안 샌드박스에서 활성화한 것보다 더 많은 권한이 필요한 기능을 실행해야 합니다.

다음 페이지를 포함하는 XBAP 응용 프로그램을 고려하십시오.

```
FileIOPermission fp = new FileIOPermission(PermissionState.Unrestricted);
fp.Assert();

// Perform operation that uses the assert

// Revert the assert when operation is completed
CodeAccessPermission.RevertAssert();
```

```
Dim fp As New FileIOPermission(PermissionState.Unrestricted)
fp.Assert()

' Perform operation that uses the assert

' Revert the assert when operation is completed
CodeAccessPermission.RevertAssert()
```

이 XBAP를 실행하려면 기본 WPF 코드가 다음을 포함하여 XBAP 호출에서 사용할 수 있는 것보다 더 많은 기능을 실행해야 합니다.

- 렌더링을 위한 HWND(창 핸들 만들기)
- 메시지 디스패치
- Tahoma 글꼴 로드

샌드박스 애플리케이션에서 이러한 작업에 직접 액세스할 수 있도록 허용하면 보안상 치명적일 수 있습니다.

다행히 WPF는 샌드박스 응용 프로그램을 대신하여 높은 권한으로 이러한 작업을 실행할 수 있도록 하여 이러한 상황을 충족시킵니다. 모든 WPF 작업은 XBAP의 응용 프로그램 도메인의 제한된 인터넷 영역 보안 권한에 대해 검사되지만 WPF(다른 시스템 라이브러리와 마찬가지로)에는 가능한 모든 사용 권한이 포함된 권한 집합이 부여됩니다.

이렇게 하려면 WPF가 호스트 응용 프로그램 도메인의 인터넷 영역 권한 집합에 의해 해당 권한이 제어되지 않도록 하면서 높은 권한을 받아야 합니다.

WPF는 권한의 **Assert** 메서드를 사용하여 이 작업을 수행합니다. 다음 코드에서는 이렇게 되는 방식을 보여 줍니다.

```
FileIOPermission fp = new FileIOPermission(PermissionState.Unrestricted);
fp.Assert();

// Perform operation that uses the assert

// Revert the assert when operation is completed
CodeAccessPermission.RevertAssert();
```

```
Dim fp As New FileIOPermission(PermissionState.Unrestricted)
fp.Assert()

' Perform operation that uses the assert

' Revert the assert when operation is completed
CodeAccessPermission.RevertAssert()
```

어설션은 기본적으로 WPF에 필요한 무제한 사용 권한이 XBAP의 인터넷 영역 권한에 의해 제한되는 것을 방지합니다.

플랫폼 관점에서 WPF는 **어설션을** 올바르게 사용할 책임이 있습니다. **Assert를** 잘못 사용하면 악성 코드가 권한을 상승시킬 수 있습니다. 따라서 필요한 경우에만 **Assert를** 호출하고 샌드박스 제한이 그대로 유지되도록 하는 것이 중요합니다. 예를 들어 샌드박스 코드는 임의의 파일을 열 수 없지만 글꼴을 사용할 수 있습니다. WPF를 사용하면 샌드박스 응용 프로그램에서 **Assert를** 호출하여 글꼴 기능을 사용하고 WPF는 샌드박스 응용 프로그램을 대신하여 해당 글꼴을 포함하는 것으로 알려진 파일을 읽을 수 있습니다.

ClickOnce 배포

ClickOnce는 .NET Framework에 포함되어 있으며 Visual Studio와 통합되는 포괄적인 배포 기술입니다(자세한 내용은 ClickOnce 보안 및 배포 참조). 독립 실행형 WPF 응용 프로그램은 ClickOnce를 사용하여 배포할 수 있으며 브라우저 호스팅 응용 프로그램은 ClickOnce를 사용하여 배포해야 합니다.

ClickOnce를 사용하여 배포된 응용 프로그램에는 CAS(코드 액세스 보안)에 대한 추가 보안 계층이 제공됩니다. 기본적으로 ClickOnce 배포된 응용 프로그램은 필요한 권한을 요청합니다. 애플리케이션이 배포된 소스 영역에 대한 권한 집합을 초과하지 않는 경우에만 해당 권한이 부여됩니다. 사용 권한 집합을 시작 영역의 사용 권한 집합에서 제공하는 것보다 적더라도 필요한 권한 집합으로 줄이면 응용 프로그램에서 액세스할 수 있는 리소스 수가 최소로 줄어듭니다. 따라서 애플리케이션을 가로채는 경우 클라이언트 컴퓨터의 손상 가능성이 줄어듭니다.

보안에 중요한 방법론

XBAP 응용 프로그램에 대한 인터넷 영역 샌드박스를 사용하도록 사용 권한을 사용하는 WPF 코드는 가능한 한 높은 수준의 보안 감사 및 제어를 유지해야 합니다. 이러한 요구 사항을 용이하게 하기 위해 .NET Framework는 권한을 높이는 코드 관리에 대한 새로운 지원을 제공합니다. 특히 CLR을 사용하면 권한을 높이는 코드를 식별하고 을 로 SecurityCriticalAttribute표시할 수 있습니다. 이 방법론을 SecurityCriticalAttribute 사용하여 표시되지 않은 코드는 *투명해집니다.* 반대로, SecurityCriticalAttribute로 표시되지 않은 관리 코드는 권한을 높일 수 있습니다.

보안에 중요한 방법론을 사용하면 권한을 *보안에 중요한 커널로*승격하는 WPF 코드 의 구성을 허용하고 나머지는 투명하게 사용할 수 있습니다. 보안에 중요한 코드를 격리하면 WPF 엔지니어링 팀이 표준 보안 관행 을 넘어 보안에 중요한 커널에 대한 추가 보안 분석 및 소스 제어에 집중할 수 있습니다(WPF 보안 전략 - 보안 엔지니어링참조).

.NET Framework는 개발자가 APTCA(APTCA)로 AllowPartiallyTrustedCallersAttribute 표시되고 사용자의 GAC(전

역 어셈블리 캐시)에 배포된 관리되는 어셈블리를 작성할 수 있도록 하여 신뢰할 수 있는 코드가 XBAP 인터넷 영역 샌드박스를 확장할 수 있도록 합니다. 어셈블리에 APTCA로 표시하는 경우 인터넷의 악성 코드를 비롯한 모든 코드에서 해당 어셈블리를 호출할 수 있으므로 중요한 보안 작업입니다. 이 작업을 수행할 때는 주의해서 최선의 방법을 사용해야 하며, 사용자가 해당 소프트웨어를 신뢰해야 설치됩니다.

Microsoft Internet Explorer 보안

Microsoft Internet Explorer 6(SP2)에는 보안 문제를 줄이고 보안 구성을 단순화하는 것 외에도 XAML 브라우저 응용 프로그램(XBAPs) 사용자의 보안을 강화하는 몇 가지 기능이 포함되어 있습니다. 이러한 기능은 사용자가 검색 환경을 보다 효율적으로 제어할 수 있도록 하기 위한 것입니다.

IE6 SP2 이전에는 다음과 같은 사항의 적용을 받을 수 있습니다.

- 무작위 팝업 창
- 혼란스러운 스크립트 리디렉션
- 일부 웹 사이트의 수많은 보안 대화 상자

경우에 따라 신뢰할 수 없는 웹 사이트는 사용자가 설치를 취소했더라도 설치를 UI(사용자 인터페이스) 스푸핑하거나 Microsoft ActiveX 설치 대화 상자를 반복적으로 표시하여 사용자를 속이려고 합니다. 이러한 기술을 사용하면 다수의 사용자가 속아서 잘못된 결정을 내리고 스파이웨어 애플리케이션을 설치할 수 있습니다.

IE6 SP2에는 이러한 유형의 문제를 완화하는 몇 가지 기능이 포함되어 있으며, 이는 사용자 개시 개념을 중심으로 진행됩니다. IE6 SP2는 사용자가 *작업*시작이라고 하는 작업 전에 링크 또는 페이지 요소를 클릭한 경우를 감지하고 페이지의 스크립트에 의해 유사한 작업이 트리거될 때와 다르게 처리합니다. 예를 들어 IE6 SP2에는 사용자가 팝업을 만들기 전에 단추를 클릭할 때 이를 감지하는 팝업 **차단이** 통합됩니다. 이를 통해 IE6 SP2는 사용자가 요청하거나 원하지 않는 팝업을 방지하는 동시에 대부분의 무해한 팝업을 허용할 수 있습니다. 차단된 팝업은 사용자가 수동으로 블록을 재정의하고 팝업을 볼 수 있는 새로운 **정보 표시줄**아래에 갇혀 있습니다.

Open/Save 보안 프롬프트에도 동일한 사용자 개시 논리가 적용됩니다. ActiveX 설치 대화 상자는 이전에 설치된 컨트롤의 업그레이드를 나타내지 않는 한 항상 정보 표시줄 아래에 갇혀 있습니다. 이러한 조치가 결합되어 사용자에게 더 안전하고 제어된 사용자 환경을 제공합니다. 사용자가 원하지 않는 소프트웨어나 악성 소프트웨어를 설치하도록 유인하는 사이트로부터 보호되기 때문입니다.

또한 IE6 SP2를 사용하여 WPF 응용 프로그램을 다운로드하고 설치할 수 있는 웹 사이트를 탐색하는 고객을 보호합니다. 특히 IE6 SP2는 WPF를 포함하여 어떤 기술을 사용했는지여부에 관계없이 사용자가 악의적이거나 악의적인 응용 프로그램을 설치할 수 있는 더 나은 사용자 환경을 제공하므로 이 기능을 사용할 수 있습니다. WPF는 ClickOnce를 사용하여 인터넷을 통해 응용 프로그램의 다운로드를 용이하게하여 이러한 보호에 추가합니다. XAML 브라우저 응용 프로그램(XBAPs)은 인터넷 영역 보안 샌드박스 내에서 실행되므로 원활하게 시작할 수 있습니다. 반면에 독립 실행형 WPF 응용 프로그램을 실행하려면 완전 신뢰가 필요합니다. 이러한 응용 프로그램의 경우 ClickOnce는 응용 프로그램의 추가 보안 요구 사항을 사용하도록 알리는 시작 프로세스 중에 보안 대화 상자가 표시됩니다. 그러나 사용자가 시작해야 하고, 사용자가 시작한 논리에 의해 제어되며, 취소할 수 있습니다.

Internet Explorer 7은 보안에 대한 지속적인 노력의 일환으로 IE6 SP2의 보안 기능을 통합하고 확장합니다.

참고 항목

- 코드 액세스 보안
- 보안
- WPF 부분 신뢰 보안
- WPF 보안 전략 - 보안 엔지니어링

WPF 보안 전략 - 보안 엔지니어링

2020-03-21 • 12 minutes to read • Edit Online

신뢰할 수 있는 컴퓨팅은 보안 코드 생성을 보장하기 위한 Microsoft 이니셔티브입니다. 신뢰할 수 있는 컴퓨팅 이니셔티브의 핵심 요소는 Microsoft 보안 개발 수명 주기(SDL)입니다. SDL은 보안 코드 의 전달을 용이하게하기 위해 표준 엔지니어링 프로세스와 함께 사용되는 엔지니어링 관행입니다. SDL은 모범 사례와 형식화, 측정 가능성 및 다음과 같은 추가 구조를 결합하는 10단계로 구성됩니다.

- 보안 디자인 분석
- 도구 기반 품질 검사
- 침투 테스트
- 최종 보안 검토
- 릴리스 후 제품 보안 관리

WPF 고유 정보

WPF 엔지니어링 팀은 SDL을 적용하고 확장하며, 이 조합은 다음과 같은 주요 측면을 포함합니다.

위협 모델링

보안 분석 및 편집 도구

테스트 기술

중요한 코드 관리

위협 모델링

위협 모델링은 SDL의 핵심 구성 요소이며 잠재적인 보안 취약점을 확인하기 위해 시스템을 프로파일링하는 데 사용됩니다. 취약상이 식별되면 위협 모델링은 적절한 완화가 적용되었는지도 확인합니다.

높은 수준에서 위협 모델링은 식품점을 예로 들어 다음과 같은 주요 단계를 포함합니다.

1. **자산 식별** 식품점의 자산에는 직원, 금고, 현금 등록기 및 재고가 포함될 수 있습니다.
2. **진입점 열거** 식품점의 진입점에는 앞문과 뒷문, 창, 하역장 및 에어컨 장치가 포함될 수 있습니다.
3. **진입점을 통한 자산 공격 조사** 한 가지 가능한 공격은 *에어컨* 진입점을 통해 식품점의 *금고* 자산을 대상으로 할 수 있습니다. 에어컨 장치의 나사를 풀어 금고를 상점 외부로 끌고 나갈 수 있습니다.

위협 모델링은 WPF 전체에 적용되며 다음을 포함합니다.

- XAML 파서가 파일을 읽고, 텍스트를 해당 개체 모델 클래스에 매핑하고, 실제 코드를 만드는 방법
- 창 핸들(hWnd)이 만들어지고, 메시지를 보내고, 창의 내용을 렌더링하는 데 사용되는 방법
- 데이터 바인딩이 리소스를 가져오고 시스템과 상호 작용하는 방법

이러한 위협 모델은 개발 프로세스 중 보안 디자인 요구 사항과 위협 완화를 식별하는 데 중요합니다.

소스 분석 및 편집 도구

WPF 팀은 SDL의 수동 보안 코드 검토 요소 외에도 소스 분석 및 관련 편집을 위해 여러 도구를 사용하여 보안 취약점을 줄입니다. 다음을 포함하여 다양한 소스 도구가 사용됩니다.

- **FXCop**: 상속 규칙에서 코드 액세스 보안 사용 및 비관리 코드와 안전하게 상호 운용하는 방법에 이르기까지 관리 코드에서 일반적인 보안 문제를 찾습니다. FXCop을 참조하세요.
- **Prefix/Prefast**: 비관리 코드에서 버퍼 오버런, 형식 문자열 문제 및 오류 검사와 같은 보안 취약성 및 일반적인 보안 문제를 찾습니다.
- **금지된 API**: 소스 코드를 검색하여 `strcpy`와 같이 보안 문제가 발생하는 것으로 잘 알려진 함수가 실수로 사용되었는지 식별합니다. 식별되면 이러한 함수는 보다 안전한 대안으로 대체됩니다.

테스트 기술

WPF는 다음을 포함하는 다양한 보안 테스트 기술을 사용합니다.

- **화이트박스 테스트**: 테스터는 소스 코드를 보고 익스플로잇 테스트를 빌드합니다.
- **Blackbox 테스트**: 테스터가 API 및 기능을 검사하여 보안 익스플로이트를 찾은 다음 제품을 공격하려고 합니다.
- **다른 제품의 보안 문제 재발**: 해당하는 경우 관련 제품의 보안 문제를 테스트합니다. 예를 들어, Internet Explorer에 대한 약 60개의 보안 문제의 적절한 변형이 식별되어 WPF에 적용 가능하도록 시도되었습니다.
- **파일 퍼지 테스트를 통한 도구 기반 침투 테스트**: 파일 퍼지 테스트는 다양한 입력을 통해 파일 판독기의 입력 범위를 악용합니다. WPF에서 이 기술이 사용되는 한 가지 예는 이미지 디코딩 코드 오류 검사입니다.

중요한 코드 관리

XAML 브라우저 응용 프로그램(XBAPs)의 WPF 경우 권한을 높이는 보안에 중요한 코드를 표시하고 추적하기 위한 .NET Framework 지원을 사용하여 보안 샌드박스를 빌드합니다(WPF 보안 전략 - 플랫폼 보안의보안 에 중요한 **방법론** 참조). 보안에 중요한 코드의 높은 보안 품질 요구 사항을 감안하여 이러한 코드는 추가 수준의 소스 관리 제어 및 보안 감사를 받습니다. WPF의 약 5%-10%는 전담 검토팀이 검토하는 보안에 중요한 코드로 구성됩니다. 소스 코드 및 체크 인 프로세스는 보안에 중요한 코드를 추적하고 중요한 엔터티(예: 중요한 코드가 포함된 메서드)를 사인오프 상태로 매핑하여 관리합니다. 사인오프 상태에는 하나 이상의 검토자 이름이 포함됩니다. WPF의 각 일별 빌드는 중요한 코드를 이전 빌드의 코드와 비교하여 승인되지 않은 변경 내용을 확인합니다. 엔지니어가 검토팀의 승인 없이 중요한 코드를 수정하는 경우 식별되어 즉시 수정됩니다. 이 프로세스를 통해 WPF 샌드박스 코드에 특히 높은 수준의 감시를 적용하고 유지할 수 있습니다.

참고 항목

- 보안
- WPF 부분 신뢰 보안
- WPF 보안 전략 - 플랫폼 보안
- 신뢰할 수 있는 컴퓨팅
- .NET의 보안

WPF 샘플

2020-02-03 • 2 minutes to read • Edit Online

Windows Presentation Foundation (WPF)을 보여 주는 샘플은 GitHub의 Microsoft/WPF 샘플 리포지토리 를 참조하세요.

클래스 라이브러리(WPF)

2020-02-03 • 2 minutes to read • Edit Online

다음 링크는 WPF(Windows Presentation Foundation) Api를 포함 하는 네임 스페이스를 참조 합니다.

섹션 내용

참조

- Microsoft.Build.Tasks.Windows
- Microsoft.Win32 (공유)
- Microsoft.Windows.Themes
- System.Collections.ObjectModel (공유)
- System.Collections.Specialized (공유)
- System.ComponentModel (공유)
- System.Diagnostics (공유)
- System.IO (공유)
- System.IO.Packaging
- System.Printing
- System.Printing.IndexedProperties
- System.Printing.Interop
- System.Security.Permissions (공유)
- System.Security.RightsManagement
- System.Windows
- System.Windows.Annotations
- System.Windows.Annotations.Storage
- System.Windows.Automation
- System.Windows.Automation.Peers
- System.Windows.Automation.Provider
- System.Windows.Automation.Text
- System.Windows.Controls
- System.Windows.Controls.Primitives
- System.Windows.Converters
- System.Windows.Data

- System.Windows.Documents
- System.Windows.Documents.DocumentStructures
- System.Windows.Documents.Serialization
- System.Windows.Forms.Integration
- System.Windows.Ink
- System.Windows.Input
- System.Windows.Input.StylusPlugIns
- System.Windows.Interop
- System.Windows.Markup (공유)
- System.Windows.Markup.Localizer
- System.Windows.Markup.Primitives
- System.Windows.Media
- System.Windows.Media.Animation
- System.Windows.Media.Converters
- System.Windows.Media.Effects
- System.Windows.Media.Imaging
- System.Windows.Media.Media3D
- System.Windows.Media.Media3D.Converters
- System.Windows.Media.TextFormatting
- System.Windows.Navigation
- System.Windows.Resources
- System.Windows.Shapes
- System.Windows.Threading
- System.Windows.Xps
- System.Windows.Xps.Packaging
- System.Windows.Xps.Serialization
- UIAutomationClientsideProviders

.NET 4의 XAML 지원

다음 네임 스페이스는 System.xaml 어셈블리의 형식을 포함 합니다. System.xaml은 .NET Framework 4에서 빌드된 WPF와 같은 프레임 워크에 대 한 공통 XAML 언어 지원을 제공 합니다.

- System.Windows.Markup (공유)
- System.Xaml
- System.Xaml.Permissions

- System.Xaml.Schema