## Giới thiệu

Trong các ứng dụng hiện đại, giao dụng người dùng trực quan chiếm vị trí hết sức quan trọng. Việc trình diễn đúng thông tin, theo đúng cách và vào đúng thời điểm có thể đem lại những giá trị kinh tế xã hội đáng kể. Người sử dụng thường thích những phần mềm có giao diện trực quan, dễ sử dụng, dễ tương tác, ít thao tác phức tạp, … Để có một giao diện người dùng như vậy, việc tích hợp hình ảnh, âm thanh, văn bản và các thành phần trực quan khách thành một thể thống nhất đóng vai trò quan trọng.

Ví dụ một giao diện gồm các thành phần như sau:

* Hiển thị text và hình ảnh người dùng
* Hiển thị và cập nhật liên tục hình ảnh 2 chiều như các biểu đồ, trạng thái …
* Hiển thị hình ảnh 3 chiều
* Trình diễn video.
* Có khả năng hoạt động như một ứng dụng desktop. Đồng thời có thể chạy trên trình duyệt web có giới hạn về phân quyền, cho phép truy cập từ xa qua internet.

Với công nghệ trước 2006, ta sẽ gặp khó khăn nếu xây dựng ứng dụng như vậy bởi các nguyên nhân:

* Có nhiều cách khác nhau để làm việc với hình ảnh, âm thanh nhưng sẽ gặp khó khăn trong quá trình phát triển và bảo trì
* Thiết kế một giao diện như vậy đòi hỏi người thiết kế giao diện chuyên nghiệp, bởi vì lập trình viên đơn thuần sẽ không có đủ các kỹ năng cần thiết.
* Việc cung cấp một giao diện có thể chạy được như một ứng dụng desktop và đồng thời chạy được trên trình duyệt đòi hỏi phải xây dựng hai phiên bản độc lập sử dụng hai công nghệ khác nhau.

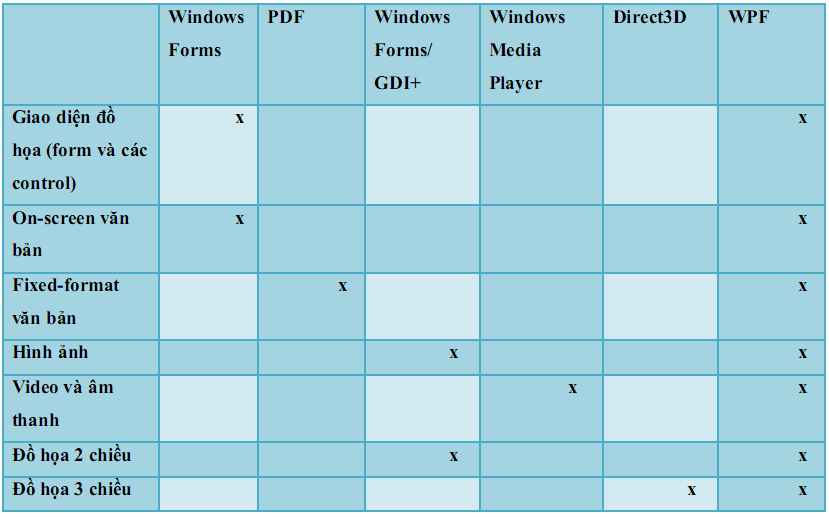
WPF ra đời là để xây dựng một nền tảng chung giải quyết những thách thức đã nêu trên

## WPF là gì?

WPF, viết tắt là Windows Presentation Foundation, là hệ thống API mới hỗ trợ việc xây dựng giao diện đồ hoạ trên nền Windows. Được xem là thế hệ kế tiếp của WinForms, WPF tăng cường khả năng lập trình giao diện của lập trình viên bằng cách cung cấp các API cho phép tận dụng những lợi thế về đa phương tiện hiện đại. Là một phần của .NET Framework 3.0, WPF sẵn có trong Windows Vista, Windows 7 và Windows Server 2008.

WPF có 3 mục tiêu cơ bản:

* Cung cấp nền tảng thống nhất để xây dựng giao diện người dùng
* Cho phép người lập trình và người thiết kế giao diện làm việc cùng nhau một cách dễ dàng
* Cung cấp một công nghệ chung để xây dựng giao diện người dùng trên cả Windows và trình duyệt Web.



Bảng so sánh các chức năng trong WPF

WPF cung cấp nhiều tính năng lập trình giao diện trong cùng một công nghệ đơn nhất. Điều này giúp cho quá trình tạo giao diện người dùng dễ dàng hơn. Ngoài việc hợp nhất nhiều tính năng lại với nhau, WPF còn có ưu điểm là có thể tận dụng được sức mạnh về phần cứng của công nghệ hiện đại ngày nay như card đồ hoạ. WPF dựa hoàn toàn trên đồ hoạ vector, cho phép nâng cao chất lượng hình ảnh trong ứng dụng hơn đồ hoạ bitmap.

Với tất cả các ưu điểm trên, WPF đơn giản hoá công việc lập trình giao diện. WPF góp phần làm giảm chi phí cho việc xây dựng và bảo trì ứng dụng. Và bằng cách cho phép tích hợp đa dạng cách biểu diễn thông tin trên giao diện người dùng, WPF cũng góp phần nâng cao chất lượng sản phẩm.

## XAML

WPF đưa ra ngôn ngữ đặt tả eXtensible Application Makeup Language (XAML). XAML định ra các phần tử giao diện như button, label, textbox … để mô tả giao diện của ngươi dùng. Việc phát triển giao diện độc lập hoàn toàn với lập trình chức năng nên người thiết kế và lập trình có thể làm việc với nhau một cách dễ dàng.

Người thiết kế có thể dùng công cụ (như Microsoft Expression Interactive Designer) để tạo những giao diện thông qua ngôn ngữ XAML. Và Microsoft Visual Studio có thể dễ dàng sử dụng những đoạn mô tả giao diện bằng XAML và tạo giao diện người dùng một cách chính xác như những gì đã mô tả.

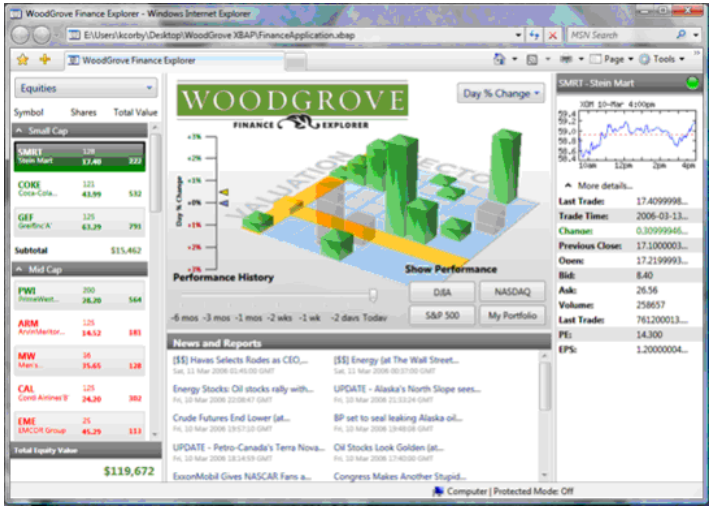
## Công nghệ chung cho giao diện trên Windows và trên trình duyệt.

Trong thời đại internet đang phát triển mạnh, các ứng dụng chạy trên trình duyệt ngày một phát triển. nên ứng dụng có thể chạy trên trình duyệt web sẽ mở ra thêm thị trường người sử dụng. Tuy nhiên, với công nghệ truyền thống, Ứng dụng desktop và ứng dụng trên trình duyệt web sử dụng công nghệ hoàn toàn khác nhau. Vì thế nếu sử dụng công nghệ truyền thống, ta sẽ phải thực hiện 2 phiên bản độc lập sẽ làm tốn chi phí và thời gian cho việc thực hiện.

Với WPF, lập trình viên có thể tạo ứng dụng trình duyệt XAML (XBAP) sử dụng WPF để chạy trên trình duyệt Web. Ngoài ra, cùng đoạn code này có thể được dùng để chạy độc lập trên Windows.



Ứng dụng WPF chạy độc lập trên windows



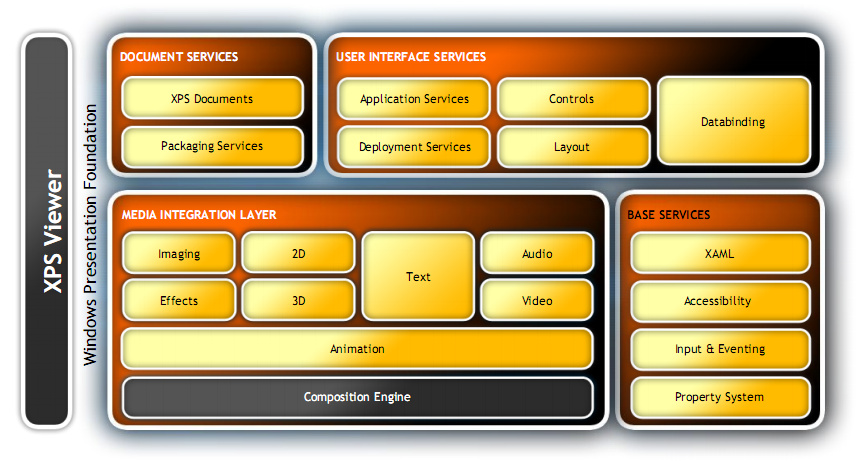
Ứng dụng nêu trên khi chạy trên trình duyệt Internet Explorer

Việc cùng một giao diện duy nhất có thể chạy được trên cả Windows và trình duyệt giúp giảm khối lượng lớn công việc nếu phải phát triển 2 giao diện song song. Ngoài ra còn có lợi thế về công nghệ, lập trình viên chỉ cần có kiến thức về WPF là có thể sử dụng ở cả 2 trường hợp.

# Các thành phần của WPF

WPF tổ chức các chức năng theo một nhóm namespace cùng trực thuộc System.Windows. Là ứng dụng windows hay XBAP, một ứng dụng WPF bao giờ cũng là một tập các trang XAML và phần code tương ứng được viết bằng c# hoặc visual basic. Tất cả các ứng dụng đều kế thừa từ lớp chuẩn Application của WPF

Nhân của WPF là cơ chế tạo sinh đồ họa dựa trên vector và độc lập với độ phân giải nhằm tận dụng những lợi thế của phần cứng đồ họa hiện đại. WPF được mở rộng với các tập tính năng phát triển ứng dụng bao gồm XAML, các control, cơ chế móc nối dữ liệu, layout, đồ họa 2 chiều, ba chiều, hoạt họa, style, khuôn dạng mẫu, văn bản, media, text và in ấn. WPF nằm trong .NET Framework, nên ngoài ra, ứng dụng WPF có thể kết hợp các thành phần khác có trong thư viện lớp của .NET Framework.



Các thành phần cơ bản của WPF

## Layout và control

### Layout

Ứng dụng WPF sử dụng panel để sắp đặt các thành phần khác nhau trên giao diện. Mỗi panel có thể chứa các thành phần khác như button, textbox … và cả panel khác. Có các loại panel:

#### Stack Panel

StackPanel là loại layout dễ sử dụng và hữu ích trong WPF. Nó chứa các thành phần con của nó theo hàng dọc hoặc hàng ngang theo tuỳ chọn. Loại này thích hợp để biểu diễn các phần tử con theo dạng danh sách. Các control như ComboBox, ListBox, hoặc Menu đều sử dụng Stack Panel

<StackPanel>

<Button Margin="10">Stack Panel</Button>

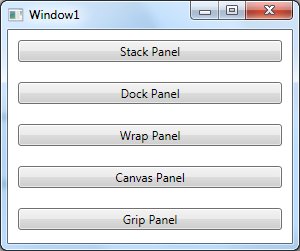
<Button Margin="10">Dock Panel</Button>

<Button Margin="10">Wrap Panel</Button>

<Button Margin="10">Canvas Panel</Button>

<Button Margin="10">Grip Panel</Button>

</StackPanel>



Stack Panel

#### Dock Panel

Dock Panel là loại layout sắp xếp các phần tử của nó bằng theo các hướng Trái, Phải, Trên, Dưới của Panel. Để xác định hướng cho phần tử, ta sử dụng thuộc tính DockPanel.Dock.

<DockPanel>

<Button DockPanel.Dock="Top"> Top Button 1</Button>

<Button DockPanel.Dock="Top">Top Button 2</Button>

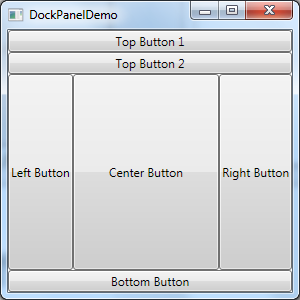
<Button DockPanel.Dock="Bottom">Bottom Button</Button>

<Button DockPanel.Dock="Right"> Right Button</Button>

<Button DockPanel.Dock="Left">Left Button</Button>

<Button > Center Button </Button>

</DockPanel>



Dock Panel

#### Wrap Panel

Wrap Panel sắp xếp các phần từ theo chiều từ trái sang phải, nếu các phần tử đã đầy khoảng không gian cho phép theo chiều ngang thì Wrap Panel sẽ tự xếp phần từ tiếp theo xuống đầu dòng tiếp theo.

<WrapPanel>

<Button>Button</Button>

<Button>Button</Button>

<Button>Button</Button>

<Button>Button</Button>

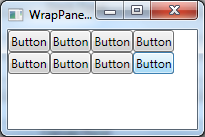
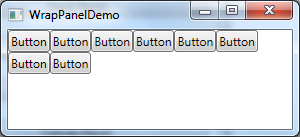
<Button>Button</Button>

<Button>Button</Button>

<Button>Button</Button>

<Button>Button</Button>

</WrapPanel>



Wrap Panel

#### Canvas Panel

Cavas Panel là loại Panel căn bản nhất của WPF. Mỗi phần tử của nó sẽ được xác định vị trí cố định bằng toạ độ. Toạ độ được xác định bằng các giá trị của các thuộc tính Canvas.Left, Canvas.Right, Canvas.Top, Canvas.Bottom:

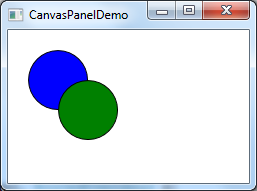
Loại Panel này thường được sử dụng để gom nhóm các hình ảnh 2D lại với nhau và không hay sử dụng cho các thành phần giao diện người dùng. Đây là vấn đề quan trọng vì khi gán toạ độ cố định sẽ gây ra vấn đề khi thay đổi kích thước của ứng dụng.

<Canvas>

<Ellipse Height="60" Width="60" Canvas.Top="20" Canvas.Left="20" Fill="Blue" Stroke="Black"></Ellipse>

<Ellipse Height="60" Width="60" Canvas.Top="50" Canvas.Left="50" Fill="Green" Stroke="Black"></Ellipse>

</Canvas>



Canvas Panel

#### Grid Panel

Grid cho phép ta phân định các dòng và cột theo dạng một lưới kẻ ô, và sau đó sẽ sắp đặt các phần tử UI vào các ô tùy ý. Grid sẽ tự động chia đều các dòng và cột (dựa trên kích thước của phần nội dung). Tuy nhiên, ta có thể sử dụng dấu sao (\*) để phân định kích thước theo tỉ lệ hoặc phân định giá trị tuyệt đối về chiều cao hoặc chiều rộng cho hàng và cột. Ta có thể nhận biết sự khác biệt của 2 dạng phân định kích thước nêu trên bằng cách thay đổi kích thước của form chứa panel Grid. Thêm vào đó, thuộc tính ShowGridLines được đặt bằng True cho phép hiển thị các đường kẻ ô.

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

<RowDefinition Height="28" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="200" />

</Grid.ColumnDefinitions>

<Label Grid.Row="0" Grid.Column="0" Content="Name:"/>

<Label Grid.Row="1" Grid.Column="0" Content="E-Mail:"/>

<Label Grid.Row="2" Grid.Column="0" Content="Comment:"/>

<TextBox Grid.Column="1" Grid.Row="0" Margin="3" />

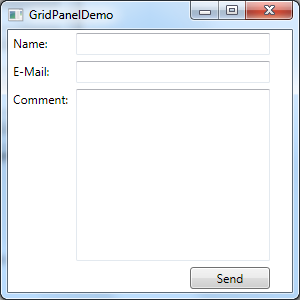
<TextBox Grid.Column="1" Grid.Row="1" Margin="3" />

<TextBox Grid.Column="1" Grid.Row="2" Margin="3" />

<Button Grid.Column="1" Grid.Row="3" HorizontalAlignment="Right"

MinWidth="80" Margin="3" Content="Send" />

</Grid>



Grid Panel

### Control

Là một công nghệ giao diện, WPF cung cấp một số lượng lớn các control. Các control chuẩn gồm Button, Label, TextBox, ListBox, Menu, Slider, hay phức tạp hơn có SpellCheck, PasswordBox… Các sự kiện do người dùng tạo ra, như di chuyển chuột hay ấn phím, có thể được các control nắm bắt và xử lý.

## Style và template

Giống như sử dụng Cascading Style Sheets (CSS) đối với HTML, việc định ra thuộc tính đồ họa cho các đối tượng giao diện một lần, rồi sau đó áp dụng lại cho các đối tượng khác cùng loại thường rất tiện lợi. WPF cũng cung cấp tính năng tương tự bằng việc sử dụng thành phần Style của XAML. Ví dụ, kiểu MyStyle có thể được định nghĩa như sau:

<Window.Resources>

<Style x:Key="MyStyle" TargetType="Button">

<Setter Property="Background" Value="Orange"/>

<Setter Property="FontStyle" Value="Italic" />

<Setter Property="Padding" Value="8,4" />

<Setter Property="Margin" Value="4" />

</Style>

</Window.Resources>

Để sử dụng style đã được định nghĩa, ta dùng thuộc tính Style=”{StaticResource StyleName}”. Ví dụ:

<WrapPanel>

<Button Style="{StaticResource MyStyle}">Styles</Button>

<Button Style="{StaticResource MyStyle}">Are</Button>

<Button Style="{StaticResource MyStyle}">Cool</Button>

</WrapPanel>



Kết quả của đoạn mã trên

Style có thể thay đổi giao diện của control như màu sắc, kiểu chữ, … Nhưng lại không thể thay đổi được hình dạng của control như nút bấm thay vì hình chữ nhật đổi thành hình eclipse. Để làm được điều đó ta phải sử dụng Template.

Template trong WPF cho phép tuỳ chỉnh các thể hiện của các control hay các tập dữ liệu một cách linh hoạt hơn. Ví dụ như thay đổi hình dạng của một nút bấm, hoặc phức tạp hơn là có một tập danh sách người dùng và ta muốn trình bày danh sách theo định nghĩa riêng của ta như hiển thị hình ảnh kế bên họ và tên.

Có 2 dạng template

* Data Template: Được sử dụng để định nghĩa cách trình bài của một tập dữ liệu
* Control Template: Được sử dụng để định nghĩa lại hình dạng của các control.

## Text

## 