Bài mở đầu
TỔNG QUAN VỀ WINDOWS REPRESENTATION FOUNDATION

Bài này giới thiệu tổng quan về công nghệ Windows Presentation Foundation (WPF). Phần đầu nói về những thách thức trong việc xây dựng giao diện người dùng hiện đại, từ đó dẫn đến sự ra đời của công nghệ WPF, công nghệ xây dựng giao diện mới của Microsoft. Sau đó sẽ giới thiệu những khái niệm, mục tiêu căn bản và các thành phần quan trọng của WPF. Phần cuối sẽ giới thiệu các công cụ cần thiết để phát triển ứng dụng WPF và giúp học viên làm quen với WPF bằng việc hướng dẫn phát triển một ứng dụng đơn giản cụ thể.

Giao diện người dùng hiện đại và những thách thức

Trong các ứng dụng hiện đại, giao diện 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. Với những ứng dụng thương mại, chẳng hạn một ứng dụng bán hàng trực tuyến, việc cung cấp một giao diện người dùng mạnh có thể tạo nên sự khác biệt giữa một công ty với các đối thủ cạnh tranh, góp phần làm tăng tăng doanh số và giá trị thương hiệu của hãng này so với hãng khác. Để có được một giao diện người dùng như vậy, việc tích hợp đồ họa, media, văn bản và các thành phần trực quan khác như một thể thống nhất đóng đóng vai trò mấu chốt.

Hãy xem xét một ứng dụng cụ thể trong quản lý và theo dõi bệnh nhân của một bệnh viện nào đó. Với sự phát triển của công nghệ đa phương tiện hiện nay, yêu cầu về giao diện người dùng cho hệ thống mới này sẽ bao gồm:

- Hiển thị hình ảnh và text về bệnh nhân.
- Hiển thị và cập nhật hình ảnh 2 chiều cho biết trạng thái của bệnh nhân như nhịp tim, huyết áp.
- Cung cấp hình ảnh chồng lớp 3 chiều về thông tin của người bệnh.
- Trình diễn những đoạn video siêu âm và những chẩn đoán khác, trong đó, cho phép bác sỹ hay y tá thêm vào các ghi chú.
- Cho phép nhân viên bệnh viện đọc và ghi chú trên những tài liệu mô tả về bệnh nhân và tình trạng của người đó.
- Có khả năng hoạt động như một ứng dụng Windows, trong đó, các nhân viên bệnh viện đều được sử dụng đầy đủ các tính năng, đồng thời có thể chạy trên trình duyệt Web có giới hạn về an ninh, cho phép các bác sỹ truy nhập có hạn chế từ xa qua mạng Internet.

Với công nghệ từ trước năm 2006, một giao diện như vậy trên Windows đã có thể xây dựng được, tuy nhiên, sẽ gặp không ít khó khăn bởi một số nguyên nhân chính sau:

- Có rất nhiều công nghệ khác nhau được sử dụng để làm việc với hình ảnh âm thanh và video. Tìm được những lập trình viên có khả năng sử dụng tốt nhiều công nghệ như vậy không dễ và chi phí cao cho cả quá trình phát triển cũng như bảo trì ứng dụng.
- Thiết kế một giao diện biểu diễn có hiệu quả tất cả những tính năng như vậy cũng là một thách thức. Nó đòi hỏi phải có những người thiết kế giao diện chuyên nghiệp, bởi lập trình viên phần mềm đơn thuần sẽ không có đủ các kỹ năng cần thiết. Điều này lại dẫn tới những khó khăn phát sinh khi người thiết kế và người lập trình làm việc chung.
- Việc cung cấp một giao diện đầy đủ tính năng, hoạt động được như một ứng dụng Windows riêng biệt trên máy desktop, đồng thời có thể được truy nhập thông qua trình duyệt có thể đòi hỏi phải xây dựng hai phiên bản độc lập sử dụng hai công nghệ khác nhau. Ứng dụng Windows trên desktop sử dụng Windows Forms và các công nghệ thuần Windows khác, trong khi ứng dụng trên trình duyệt lại sử dụng HTML và JavaScript. Do đó, cần phải có hai nhóm phát triển với hai phần kỹ năng khác nhau.

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

WPF là gì?

WPF, viết tắt của *Windows Presentation Foundation*, là hệ thống API mới hỗ trợ việc xây dựng giao diện đồ hoạ trên nền Windows. Được xem như 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 bộ phận của .NET Framework 3.0, WPF sẵn có trong Windows Vista và Windows Server 2008. Đồng thời, WPF cũng có thể hoạt động trên nền Windows XP Service Pack 2 hoặc mới hơn, và cả Windows Server 2003.

WPF được xây dựng nhằm vào ba mục tiêu cơ bản: 1) Cung cấp một nền tảng thống nhất để xây dựng giao diện người dùng; 2) Cho phép người lập trình và người thiết kế giao diện làm việc cùng nhau một cách dễ dàng; 3) Cung cấp một công nghệ chung để xây dựng giao diện người dùng trên cả Windows và trình duyệt Web.

1.1 Nền tảng thống nhất để xây dựng giao diện người dùng

Trước khi WPF ra đời, việc tạo giao diện người dùng theo những yêu cầu mô tả ở ví dụ trên đòi hỏi sử dụng rất nhiều công nghệ khác nhau (xem Bảng 2.1). Để tạo form, các control và các tính năng kinh điển khác của một giao diện đồ họa Windows, thông thường lập trình viên sẽ chọn Windows Forms, một phần của .NET Framework. Nếu cần hiển thị văn bản, Windows Forms có một số tính năng hỗ trợ văn bản trực tiếp hoặc có thể sử dụng Adobe's PDF để hiển thị văn bản có khuôn dạng cố định. Đối với hình ảnh và đồ họa 2 chiều, lập trình viên sẽ dùng GDI+, một mô hình lập trình riêng biệt có thể truy nhập qua Windows Forms. Để hiển thị video hay phát âm thanh, lập trình viên lại phải sử dụng Windows Media Player, và với đồ họa 3 chiều, anh ta lại phải dùng Direct3D, một thành phần chuẩn khác của Windows. Tóm lại, quá trình phát triển giao diện người dùng theo yêu cầu trở nên phức tạp, đòi hỏi lập trình viên quá nhiều kỹ năng công nghệ.

	Windows Forms	PDF	Windows Forms/ GDI+	Windows Media Player	Direct3D	WPF
Giao diện đồ họa (form và các control)	x					x
On-screen văn bản	x					x
Fixed-format văn bản		x				x
Hình ảnh			x			x
Video và âm thanh				x		x
Đồ họa 2 chiều			x			x
Đồ họa 3 chiều					x	x

Bảng 0.1 – Thành phần giao diện theo yêu cầu và những công nghệ chuyên biệt cần thiết để tạo chúng.

WPF là giải pháp hợp nhất nhằm giải quyết tất cả những vấn đề công nghệ nêu trên, hay nói cách khác, 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 trở nên dễ dàng hơn đáng kể. Hình 2.2 cho thấy một giao diện quản lý và theo dõi bệnh nhân có sự kết hợp của hình ảnh, text, đồ họa 2 chiều/3 chiều và nhiều thông tin trực quan khác. Tất cả đều được tạo ra bằng WPF – lập trình viên không cần viết code để sử dụng các công nghệ chuyên biệt như GDI+ hay Direct3D.

Hình 0.1 – Một giao diễn người dùng quản lý và theo dõi bệnh nhân sử dụng WPF có thể kết hợp hình ảnh, text, đồ họa 2 chiều/3chiều và nhiều tính năng trực quan khác

Tuy nhiên, WPF ra đời không có nghĩa là tất cả những công nghệ nêu trên bị thay thế. Windows Forms vẫn có giá trị, thậm trí trong WPF, một số ứng dụng mới vẫn sẽ sử dụng Windows Forms. Windows Media Player vẫn đóng một vai trò công cụ độc lập để chơi nhạc và trình chiếu video. PDF cho văn bản vẫn tiếp tục được sử dụng. Direct3D vẫn là công nghệ quan trọng trong games và các dạng ứng dụng khác (Trong thực tế, bản thân WPF dựa trên Direct3D để thực hiện mọi biểu diễn đồ họa).

Việc tạo ra một giao diện người dùng hiện đại không chỉ là việc hợp nhất các công nghệ sẵn có khác nhau. Nó còn thể hiện ở việc tận dụng lợi điểm của card đồ họa hiện đại. Để giải phóng những hạn chế của đồ họa bitmap, WPF dựa hoàn toàn trên đồ họa vector, cho phép hình ảnh tự động thay đổi kích thước để phù hợp với kích thước và độ phân giải của màn hình mà nó được hiển thị.

Bằng việc hợp nhất tất cả các công nghệ cần thiết để tạo ra một giao diện người dùng vào một nền tảng đơn nhất, WPF đơn giản hóa đáng kể công việc của lập trình viên giao diện. Với việc yêu cầu lập trình viên học một môi trường phát triển duy nhất, WPF góp phần làm giảm chi phí cho việc xây dựng và bảo trì ứng dụng. Và bằng việc cho phép tích hợp đa dạng nhiều cách biểu diễn thông tin trên giao diện người dùng, WPF góp phần nâng cao chất lượng, và theo đó là giá trị công việc, của cách thức người dùng tương tác với ứng dụng trên Windows.

1.2 Khả năng làm việc chung giữa người thiết kế giao diện và lập trình viên

Trong thực tế, việc xây dựng một giao diện người dùng phức hợp như trong ví dụ về ứng dụng quản lý bệnh nhân trên đòi hỏi những kỹ năng ít thấy ở những lập trình viên đơn thuần, mà chỉ có thể tìm thấy ở những người thiết kế giao diện chuyên nghiệp. Nhưng câu hỏi đặt ra là làm sao để người thiết kế và lập trình viên có thể làm việc cùng nhau?

Thông thường, người thiết kế giao diện sử dụng một công cụ đồ họa để tạo ra những ảnh tĩnh về cách bố trí giao diện trên màn hình. Những hình ảnh này sau đó được chuyển tới lập trình viên với nhiệm vụ tạo ra mã trình để hiện thực hóa giao diện đã thiết kế. Đôi lúc vẽ ra một giao diện thì đơn giản với người thiết kế, nhưng để biến nó thành hiện thực có thể là khó khăn hoặc bất khả thi với lập trình viên. Hạn chế về công nghệ, sức ép tiến độ, thiếu kỹ năng, hiểu nhầm hoặc đơn giản là bất đồng quan điểm có thể khiến lập trình viên không đáp ứng được đầy đủ yêu cầu từ người thiết kế. Do vậy, điều cần thiết ở đây là một cách thức để hai nhóm công tác độc lập này có thể làm việc với nhau mà không làm thay đổi chất lượng của giao diện đã thiết kế.

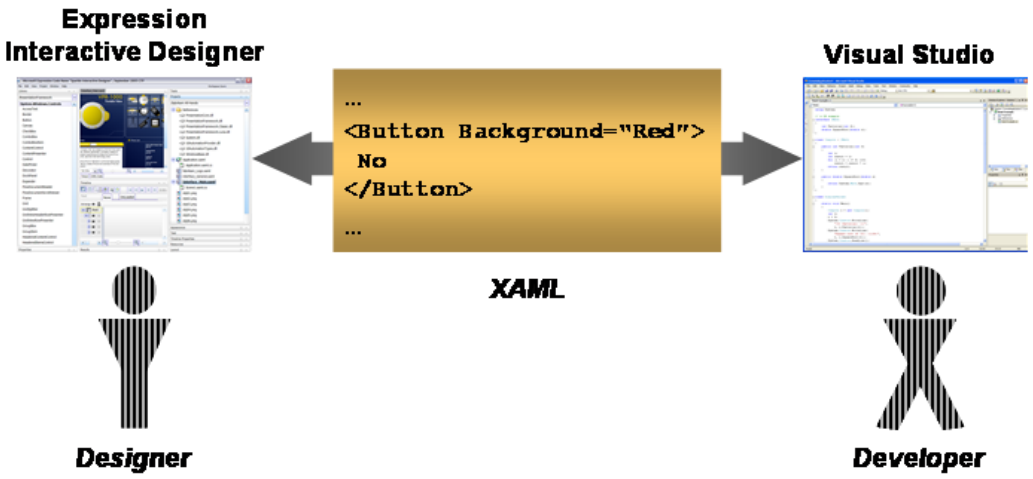
Để thực hiện được điều này, WPF đưa ra ngôn ngữ đặc tả *eXtensible Application Markup Language* (XAML). XAML định ra một tập các phần tử XML như Button, TextBox, Label…, nhằm định nghĩa các đối tượng đồ họa tương ứng như nút bấm, hộp thoại, nhãn…, và nhờ đó cho phép mô tả chính xác diện mạo của giao diện người dùng. Các phần tử XAML cũng chứa các thuộc tính, cho phép thiết lập nhiều tính chất khác nhau của đối tượng đồ họa tương ứng. Ví dụ, đoạn mã sau sẽ tạo ra một nút bấm màu đỏ có nhan đề "Bỏ qua".

```
<Button Background="Red">No</Button>
```

Mỗi phần tử XAML lại tương ứng với một lớp WPF, và mỗi thuộc tính của phần tử đó lại tương ứng với thuộc tính hay sự kiện của lớp này. Chẳng hạn, nút bấm màu đỏ trong ví dụ trên có thể tạo bằng C# code như sau:

```
Button btn = new Button();
btn.Background = Brushes.Red;
btn.Content = "No";
```

Nếu như mọi thứ có thể biểu diễn bằng XAML thì cũng có thể biểu diễn bằng đoạn mã, thì câu hỏi đặt ra là XAML có ý nghĩa gì? Câu trả lời là việc xây dựng các công cụ sinh và sử dụng các đặc tả bằng XML dễ dàng hơn nhiều so với xây dựng một công cụ tương tự làm việc với đoạn mã. Bởi vậy, XAML mở ra một cách thức tốt hơn để lập trình viên và người thiết kế làm việc với nhau. Hình 2.3 minh họa quá trình này.



Hình 0.2 – XAML hỗ trợ lập trình viên và người thiết kế làm việc chung.

Người thiết kế có thể mô tả giao diện người dùng và tương tác với nó thông qua một công cụ, chẳng hạn như *Microsoft Expression Interactive Designer*. Chỉ tập trung vào việc định ra diện mạo và cảm quan cho giao diện đồ họa WPF, công cụ này sinh các đoạn mô tả giao diện thể hiện qua ngôn ngữ XAML. Lập trình viên sau đó sẽ nhập đoạn mô tả XAML đó vào môi trường lập trình, chẳng hạn như Microsoft Visual Studio. Thay vì lập trình viên phải tái tạo lại giao diện từ đầu dựa trên một ảnh tĩnh mà người thiết kế cung cấp, bản thân các đoạn XAML này sẽ được Microsoft Visual Studio biên dịch để tái tạo thành giao diện đồ họa đúng theo mô tả. Lập trình viên chỉ tập trung vào việc viết mã trình cho giao diện được sinh ra, chẳng hạn như xử lý các sự kiện, theo những chức năng đề ra của ứng dụng.

Việc cho phép người thiết kế và lập trình viên làm việc chung như vậy sẽ hạn chế những lỗi phát sinh khi hiện thực hóa giao diện từ thiết kế. Thêm vào đó, nó còn cho phép hai nhóm công tác này làm việc song song, khiến mỗi bước lặp trong quy trình phát triển phần mềm ngắn đi và việc phản hồi được tốt hơn. Vì cả hai môi trường đều có khả năng hiểu và sử dụng XAML, ứng dụng WPF có thể chuyển qua lại giữa hai môi trường phát triển để sửa đổi hay bổ sung giao diện. Với tất cả những lợi điểm này, vai trò của người thiết kế trong việc xây dựng giao diện được đặt lên hàng đầu.

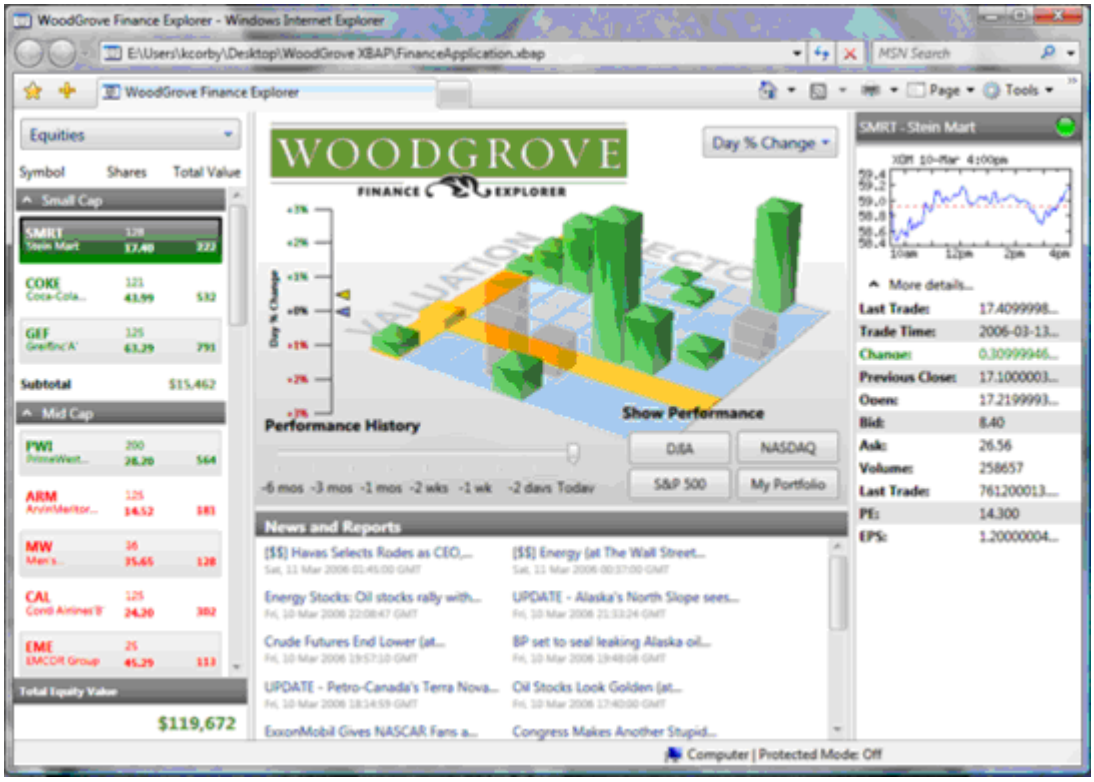
1.3 Công nghệ chung cho giao diện trên Windows và trên trình duyệt Web

Trong thời đại bùng nổ của Internet, các ứng dụng Web ngày một phát triển. Việc trang bị giao diện người dùng với đầy đủ tính năng như một ứng dụng desktop sẽ thu hút nhiều người sử dụng;, và do đó góp phần làm tăng giá trị doanh nghiệp. Tuy nhiên, như đã nêu trong phần đầu, với những công nghệ truyền thống, để phát triển một giao diện đồ họa vừa hoạt động trên desktop vừa trên trình duyệt Web, đòi hỏi phải sử dụng những công nghệ

hoàn toàn khác nhau, giống như việc xây dựng hai giao diện hoàn toàn độc lập. Điều này tạo ra chi phí không cần thiết để phát triển giao diện.

WPF là một giải pháp cho vấn đề này. Lập trình viên có thể tạo ra một *ứng dụng trình duyệt XAML* (XBAP) sử dụng WPF chạy trên Internet Explore. Trên thực tế, cùng đoạn code này có thể được dùng để sinh ứng dụng WPF chạy độc lập trên Windows. Hình 0.4 minh họa một ứng dụng dịch vụ tài chính hoạt động như một ứng dụng WPF độc lập. Trong khi đó, hình 0.4 minh họa giao diện của cùng ứng dụng chạy trên Internet Explore dưới dạng XBAP.

Hình 0.3. Ứng dụng WPF độc lập cung cấp dịch vụ tài chính chạy trong cửa sổ riêng.



Hình 0.4. Giao diện của cùng ứng dụng nêu trên dưới dạng một XBAP chạy trên Internet Explore.

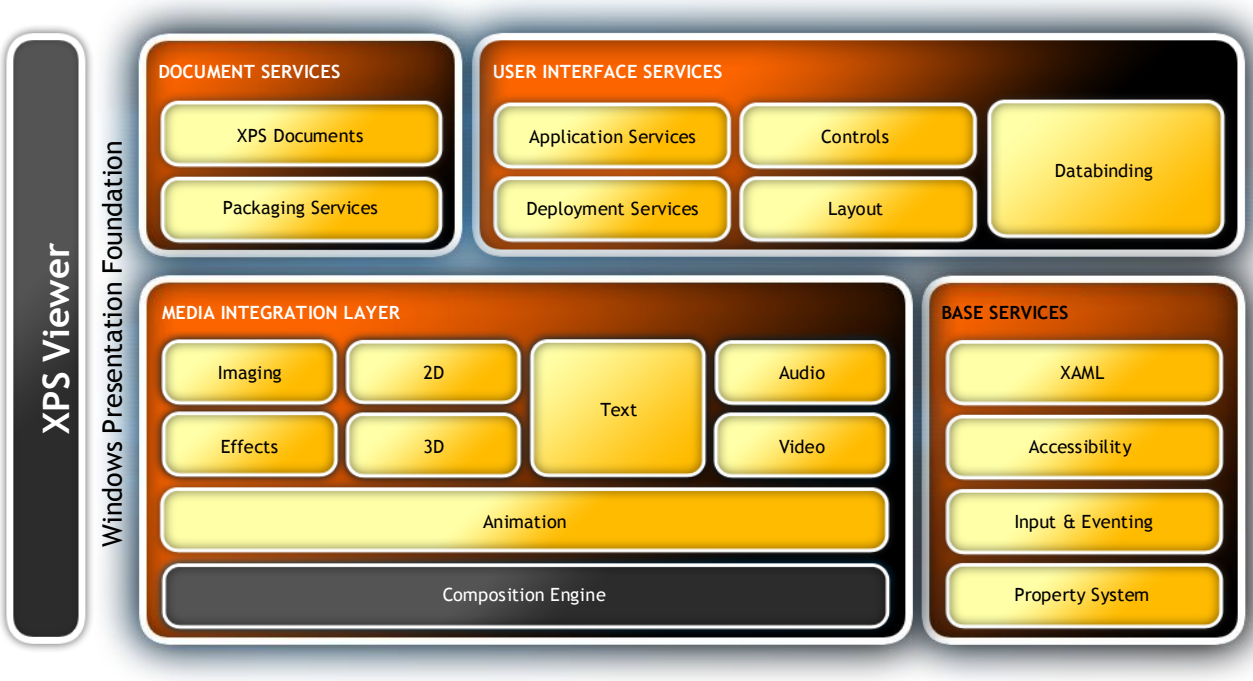
Như đã thấy trong Hình 0.4, phần giao diện của ứng dụng dạng XBAP được trình duyệt chia thành các frame thay vì chạy trên các cửa sổ riêng, ngoài ra, các chức năng đều được bảo toàn. Cùng một đoạn mã được sử dụng chung cho cả hai trường hợp sẽ làm giảm khối lượng công việc cần thiết để phát triển hai dạng giao diện. Thêm vào đó, sử dụng cùng một đoạn mã cũng có nghĩa là sử dụng cùng kỹ năng của lập trình viên. Do đó, lập trình viên chỉ cần có học một kiến thức chung là có thể sử dụng trong cả hai trường hợp. Một lợi điểm nữa của việc dùng chung công nghệ cho cả giao diện Windows và giao diện Web là người xây dựng ứng dụng không nhất thiết phải quyết định trước loại giao diện nào được sử dụng. Miễn là máy client đáp ứng được những yêu cầu hệ thống để chạy XBAP, một ứng dụng có thể cung cấp cả giao diện Windows và giao diện Web, mà chỉ sử dụng phần lớn những đoạn mã giống nhau.

Mỗi ứng dụng XBAP được download khi cần từ một Web server, nên nó phải tuân theo những yêu cầu về an ninh khắt khe hơn đối với một ứng dụng Windows độc lập. Theo đó, XBAP chạy trong phạm vi sandbox an ninh do hệ thống an ninh truy nhập mã của .NET Framework cung cấp. XBAP chỉ chạy với các hệ thống Windows có cài đặt WPF và chỉ với Internet Explore phiên bản 6 và 7 trở lên.

3. Các thành phần của WPF

Giống như các thành phần khác của .NET Framework, WPF tổ chức các chức năng theo một nhóm *namespace* cùng trực thuộc namespace **System.Windows**. Bất kể chức năng nào được sử dụng, cấu trúc cơ bản của mọi ứng dụng WPF đều gần như nhau. Là ứng dụng Windows độc lập hay là một XBAP, một ứng dụng WPF điển hình bao giờ cũng gồm một tập các trang XAML và phần code tương ứng được viết bằng C# hoặc Visual Basic, còn gọi là các file *code-behind*. Tất cả các ứng dụng đều kế thừa từ lớp chuẩn **Application** của WPF. Lớp này cung cấp những dịch vụ chung cho mọi ứng dụng, chẳng hạn như các biến lưu trữ trạng thái của ứng dụng, các phương thức chuẩn để kích hoạt hay kết thúc ứng dụng.

Mặc dù WPF cung cấp một nền tảng thống nhất để tạo giao diện người dùng, những công nghệ mà WPF chứa đựng có thể phân chia thành những thành phần độc lập. 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.



Hình 0.5. Các thành phần cơ bản của WPF

Phần tiếp theo sẽ giới thiệu sơ lược những thành phần và khái niệm quan trọng của WPF.

3.1 Layout và Control

Để sắp đặt các thành phần khác nhau trên giao diện, ứng dụng WPF sử dụng *panel*. Mỗi panel có thể chứa các thành phần con, bao gồm các control như nút bấm hay hộp thoại, hay bản than những panel khác. Những loại panel khác nhau cho phép sắp xếp thành phần con theo những cách khác nhau. Ví dụ, **DockPanel** cho phép các thành phần con có thể được đặt dọc theo cạnh của panel đó, trong khi **Grid** cho phép sắp đặt các thành phần con của nó trên một lưới tọa độ.

Giống như bất kỳ một công nghệ giao diện người dùng nào, WPF cung cấp một số lượng lớn các control. Ngoài ra, người dùng có thể tùy ý định nghĩa các control theo ý mình. 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. Trong khi các control và các thành phần giao diện khác có thể được đặc tả đầy đủ bằng XAML, các sự kiện bắt buộc phải được xử lý bằng mã trình.

3.2 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 ButtonStyle có thể được định nghĩa như sau:

```
<Style x:Key="ButtonStyle">
  <Setter Property="Control.Background" Value="Red"/>
  <Setter Property="Control.FontSize" Value="16"/>
</Style>
```

Bất kỳ nút bấm nào sử dụng kiểu này sẽ có nền màu đỏ và sử dụng font chữ kích thước 16. Ví dụ:

```
<Button Style="{StaticResource ButtonStyle}">
  Click Here
</Button>
```

Một Style có thể được dẫn xuất từ một Style khác, thừa kế hoặc chồng lên những thuộc tính đã thiết lập. Mỗi style có thể định nghĩa các *trigger* cho phép tạo ra những hiệu ứng tương tác đặc biệt, chẳng hạn như khi lướt chuột qua nút bấm, nút bấm chuyển thành màu vàng.

WPF cũng hỗ trợ sử dụng *template*. Mỗi template tương tự như một style, và ở hai dạng:

- *Template cho dữ liệu*: sử dụng thành phần **DataTemplate** của XAML để thiết lập một nhóm thuộc tính hiển thị của dữ liệu như màu sắc, phương thức căn lề...
- *Template cho control*: sử dụng thành phần **ControlTemplate** của XAML để định ra diện mạo của một control.

3.3 Text

Giao diện người dùng ít nhiều đều hiển thị chữ hay text. Đối với phần lớn mọi người, đọc text trên màn hình thường khó hơn đọc trên giấy in. Đó là do chất lượng hiển thị text trên màn hình kém hơn so với khi in ra giấy. WPF tập trung giải quyết vấn đề này, làm chất lượng text hiển thị trên màn hình tương đương trên giấy in. Cụ thể, WPF hỗ trợ các font chữ OpenType chuẩn, cho phép sử dụng các thư viện font đã có. WPF cũng hỗ trợ công nghệ font chữ mới ClearType, cho phép hiển thị các ký tự mịn hơn đối với mắt người, đặc biệt là trên màn hình tinh thể lỏng (LCD).

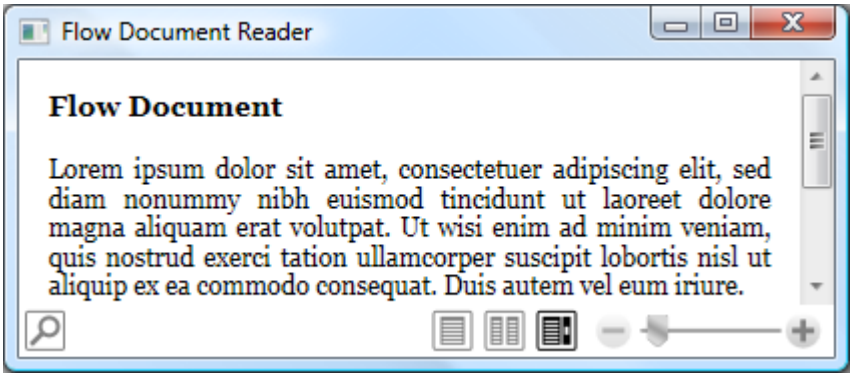
Để nâng cao hơn nữa chất lượng hiển thị text, WPF cho phép một số công nghệ khác như chữ ghép, theo đó một nhóm ký tự được thay thế bằng một ảnh đơn nhất, tạo tâm lý thoải mái hơn khi đọc đối với người dùng.

3.4 Văn bản

WPF hỗ trợ ba dạng văn bản: văn bản cố định (fixed), văn bản thích nghi (flow/adaptive) và văn bản XPS (XML Paper Specification). Kèm theo đó, WPF cũng cung cấp các dịch vụ để tạo, xem, quản lý, ghi chú, đóng gói và in ấn văn bản.

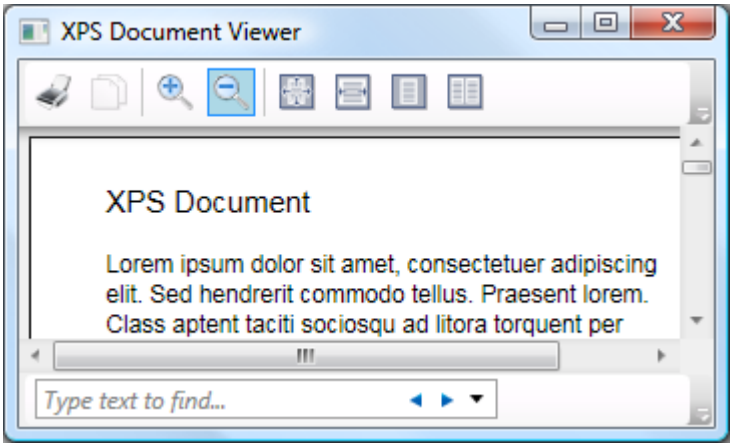
Văn bản cố định trông không đổi bất kể chúng được hiển thị trên màn hình hay in ra máy in. Trong WPF, những văn bản dạng này được định nghĩa bằng phần tử **FixedDocument** trong XAML và được hiển thị bằng control **DocumentViewer**.

Trong khi đó, văn bản thích nghi thường chỉ dùng để đọc trên màn hình, và có khả năng tự động thay đổi các thuộc tính hiển thị ảnh và text cho phù hợp với kích thước cửa sổ hay các yếu tố môi trường khác nhằm nâng cao chất lượng đọc cho người dùng. Văn bản thích nghi được định nghĩa bằng phần tử **FlowDocument.** Để hiển thị văn bản thích nghi, WPF sử dụng một số control khác nhau, chẳng hạn như **FlowDocumentPageViewer, FlowDocumentScrollViewer, FlowDocumentReader…**



Hình 0.6. – Một minh họa về văn bản thích nghi trong WPF.

Văn bản XPS xây dựng trên cơ sở văn bản bất động của WPF. XPS là một định dạng mở theo đặc tả XML, có khả năng sử dụng trên nhiều nền tảng khác nhau, được thiết kế nhằm tạo thuận lợi cho việc xây dựng, chia sẻ, in ấn và lưu trữ văn bản. Cũng như văn bản cố định, văn bản XPS được hiển thị bằng **DocumentViewer**.



Hình 0.7. Một minh họa về văn bản XPS trong WPF.

3.5 Hình ảnh

Trong WPF, hình ảnh được hiển thị nhờ control **Image,** ví dụ:

```
<Image
 Width="200"
 Source="C:\Documents and Settings\All Users\Documents\My Pictures\Ava.jpg" />
```

Control **Image** có thể hiển thị hình ảnh lưu trữ dưới nhiều khuôn dạng khác nhau, bao gồm JPEG, BMP, TIFF, GIF và PNG. Nó cũng có thể hiển thị hình ảnh dạng Windows Media Photo mới được sử dụng trong Windows Vista. Bất kể ở khuôn dạng nào, WPF sử dụng Windows Imaging Component (WIC) để tạo ra hình ảnh. Cùng với các *codec* dùng cho các khuôn dạng ảnh kể trên, WIC cũng cung cấp một nền tảng chung để bổ sung codec khác.

3.6 Video và Âm thanh

Khi tốc độ của các bộ xử lý và truyền thông mạng ngày một nâng cao, video trở thành một phần tương tác lớn của người dùng với phần mềm. Người dùng cũng sử dụng nhiều thời gian để nghe nhạc và các dạng âm thanh khác trên máy tính. Do đó, WPF cung cấp tính năng hỗ trợ cả hai dạng media này thông qua phần tử **MediaElement.** Control này có thể chơi các định dạng video WMV, MPEG và AVI, và nhiều định dạng âm thanh khác nhau. Việc lập trình để chạy một đoạn video trở nên khá đơn giản, như trong ví dụ sau:

```
<MediaElement
 Source="C:\Documents and Settings\All Users\Documents\
  My Videos\Ruby.wmv" />
```

3.7 Đồ họa hai chiều

Trong 20 năm gần đây, việc tạo ra đồ họa hai chiều trên Windows dựa trên Graphics Device Interface (GDI) và phiên bản sau của nó GDI+. Các ứng dụng Windows Forms phải sử dụng chức năng này thông qua một namespace khác hoàn toàn, bởi bản thân Windows Forms không tích hợp đồ họa 2 chiều. Đối với đồ họa 3 chiều thì càng tồi hơn, Windows Forms phải dựa trên công nghệ hoàn toàn biệt lập là Direct3D. Với WPF, vấn đề trở nên đơn giản hơn nhiều. Cả đồ họa 2 chiều và 3 chiều đều có thể được tạo ra trực tiếp trong XAML hoặc trong code sử dụng thư viện WPF tương ứng.

Đối với đồ họa 2 chiều, WPF định ra nhóm control của các khuôn hình (shapes) mà ứng dụng có thể sử dụng để tạo nên hình ảnh, gồm:

- **Line**: vẽ đường thẳng qua 2 điểm.

- **Elllipse**: vẽ ellipse.
- **Rectangle**: vẽ chữ nhật.
- **Polygon**: vẽ đa giác.
- **Polyline**: vẽ đa giác mở.
- **Path**: vẽ hình theo một đường bất kỳ.

Mỗi khuôn hình đều có các thuộc tính phong phú cho phép hiển thị với nhiều tính chất khác nhau: màu nền, màu biên… Một đặc điểm quan trọng trong WPF là: vì mọi thứ đều được xây dựng trên một nền chung, việc kết hợp các đặc tính và đối tượng khác nhau, chẳng hạn, lồng một ảnh vào một hình chữ nhật, trở nên đơn giản. Điểm thú vị nữa là các đối tượng hình học này còn có thể thu nhận các sự kiện từ phía người dùng như một control, chẳng hạn sự kiện nhắp chuột.

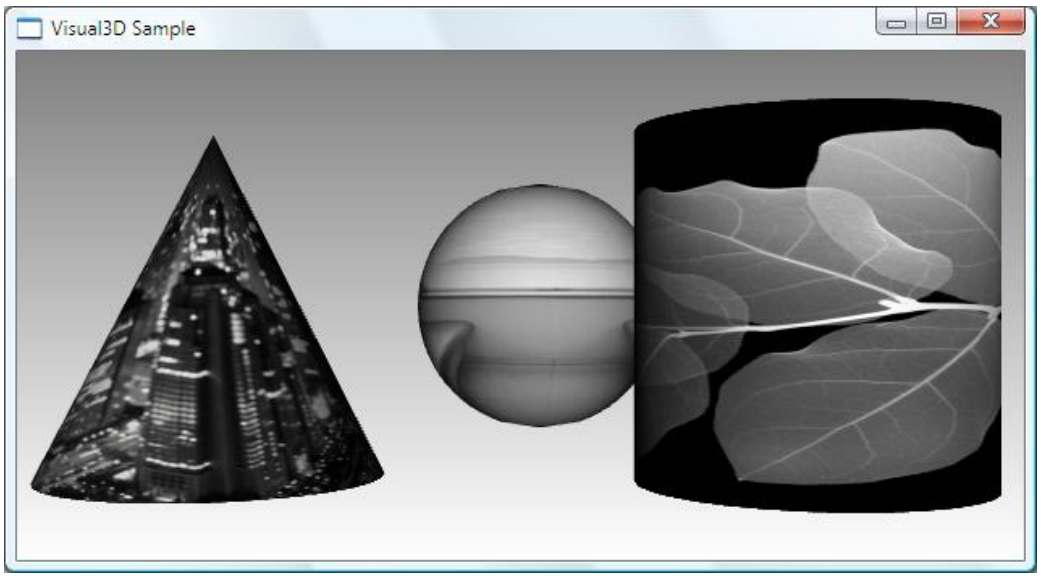
Ngoài ra, WPF cũng cung cấp một nhóm chức năng hình học khác, gọi là *geometries*, để làm việc với đồ họa hai chiều, như **LineGeometry**, **RectangleGeometry**, **EllipseGeometry**, và **PathGeometry.** Dạng hình học này có nhiều thuộc tính và chức năng tương tự như các khuôn hình đã nêu trên. Điểm khác biệt quan trọng nhất là các geometries không được dùng để hiển thị, chúng được dùng chủ yếu để tính toán hình học, ví dụ như để định ra các vùng miền, theo dõi vị trí bấm chuột...

Thêm vào đó, WPF cung cấp lớp **Transform** cho phép thực hiện các biến đổi hình học như xoay, dịch chuyển, co giãn đối tượng đồ họa; hoặc cho phép thực hiện các hiệu ứng hoạt họa theo thời gian thông qua các lớp **Animation** và **Timing.**

3.8 Đồ họa ba chiều

WPF hỗ trợ đồ họa 3 chiều bằng việc gói các lời gọi API của Direct3D, và do vậy, việc sử dụng chúng trở nên thống nhất và đơn giản hơn đáng kể. Để hiển thị đồ họa ba chiều, ứng dụng WPF sử dụng control **Viewport3D**. Để tạo ra các cảnh ba chiều, lập trình viên mô tả một hay nhiều *mô hình*, sau đó, phân định cách thức các mô hình này được chiếu sáng hay hiển thị. Như thường lệ, điều này được thực hiện bằng XAML, bằng code hay trộn cả hai. Để mô tả mô hình, WPF cung cấp lớp **GeometryModel3D** để tạo ra hình dạng của mô hình. Khi mô hình đã được định hình, diện mạo bên ngoài của nó có thể được điều khiển bằng việc phủ lên các *vật liệu* (material). Chẳng hạn, lớp **SpecularMaterial** cho phép tạo bóng trên bề mặt mô hình.

Bất kể được làm từ vật liệu gì, một mô hình có thể được chiếu sáng theo nhiều cách. Lớp **DirectionalLight** cho phép ánh sáng tới từ một hướng xác định, trong khi lớp **AmbientLight** tạo ra ánh sáng đồng đều trên mọi vật trong cảnh. Cuối cùng, để định ra cách nhìn cảnh, lập trình viên phải định ra một *camera*. Ví dụ, **PerspectiveCamera** cho phép phân định khoảng cách từ vị trí nhìn tới vật thể và kiểu nhìn phối cảnh (tuân theo luật gần xa).



Hình 0.8. Tạo lập đối tượng đồ họa ba chiều với WPF.

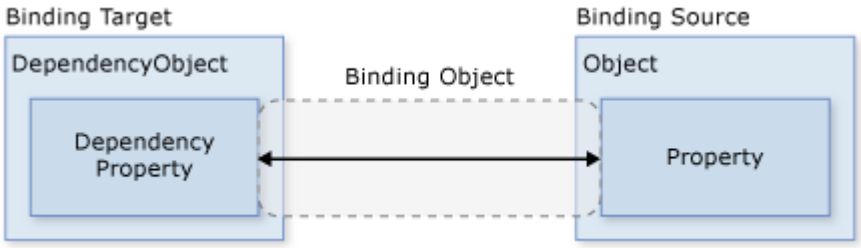
Xây dựng cảnh ba chiều trực tiếp bằng XAML hay mã trình đều không đơn giản. Do đó, chỉ nên dùng ứng dụng WPF để hiển thị cảnh ba chiều, việc xây dựng cảnh nên được thực hiện bằng những công cụ đồ họa chuyên biệt.

3.9 Móc nối dữ liệu

Phần lớn các ứng dụng được tạo ra đều cung cấp cho người dùng phương tiện để xem và sửa đổi dữ liệu. Trong các ứng dụng WPF, việc lưu trữ và truy xuất dữ liệu đã được thực hiện bởi các công nghệ như Microsoft SQL Server và ADO.NET. Sau khi dữ liệu được truy xuất và tải vào các đối tượng quản lý dữ liệu trên ứng dụng, phần việc khó khăn của ứng dụng WPF mới bắt đầu. Về cơ bản, có hai công việc phải thực hiện:

1) Sao chép dữ liệu từ các đối tượng quản lý dữ liệu vào các control trên giao diện, qua đó, dữ liệu có thể được hiển thị hay sửa đổi.
2) Đảm bảo rằng những thay đổi trên dữ liệu từ các control được cập nhật trở lại các đối tượng quản lý dữ liệu.

Để đơn giản hóa quá trình phát triển ứng dụng, WPF cung cấp một cơ chế móc nối dữ liệu để thực hiện tự động những bước này. Phần nhân của cơ chế móc nối dữ liệu là lớp **Binding** mà nhiệm vụ của nó là liên kết control trên giao diện (đích) với đối tượng quản lý dữ liệu (nguồn). Mối quan hệ này được minh họa trong hình dưới đây:



Hình 0.9. Quan hệ giữa đối tượng dữ liệu và đối tượng phụ thuộc.

Việc hỗ trợ móc nối dữ liệu được xây dựng ngay từ nhân của WPF. Tất cả các đối tượng đồ họa trong WPF đều kế thừa từ **DependencyObject**, chúng là các *đối tượng phụ thuộc*. Chức năng mà lớp cơ sở này hỗ trợ cho phép thực hiện hiệu ứng hoạt họa, tạo kiểu mẫu (styling) và móc nối dữ liệu. Các đối tượng này đều mang một thuộc tính đặc biệt gọi là **DependencyProperty**, *thuộc tính phụ thuộc*. Phần lớn các thuộc tính hay dùng như **Text, Content, Width, Height**, vân vân đều là các thuộc tính phụ thuộc. Tất cả các thuộc tính phụ thuộc đều có thể tạo hiệu ứng hoạt họa, tạo kiểu và kết nối dữ liệu.

Cơ chế móc nối dữ liệu trong WPF còn cung cấp thêm những tính năng như xác thực tính hợp lệ, sắp xếp, lọc và phân nhóm dữ liệu. Thêm vào đó, tính năng móc nối dữ liệu cũng hỗ trợ sử dụng khuôn mẫu dữ liệu (data template) để tạo ra các đối tượng giao diện tùy biến có kết nối dữ liệu, khi các control chuẩn không phù hợp. Móc nối dữ liệu và khuôn dạng dữ liệu có thể được coi là tính năng mạnh nhất của WPF.

4. Công cụ phát triển WPF

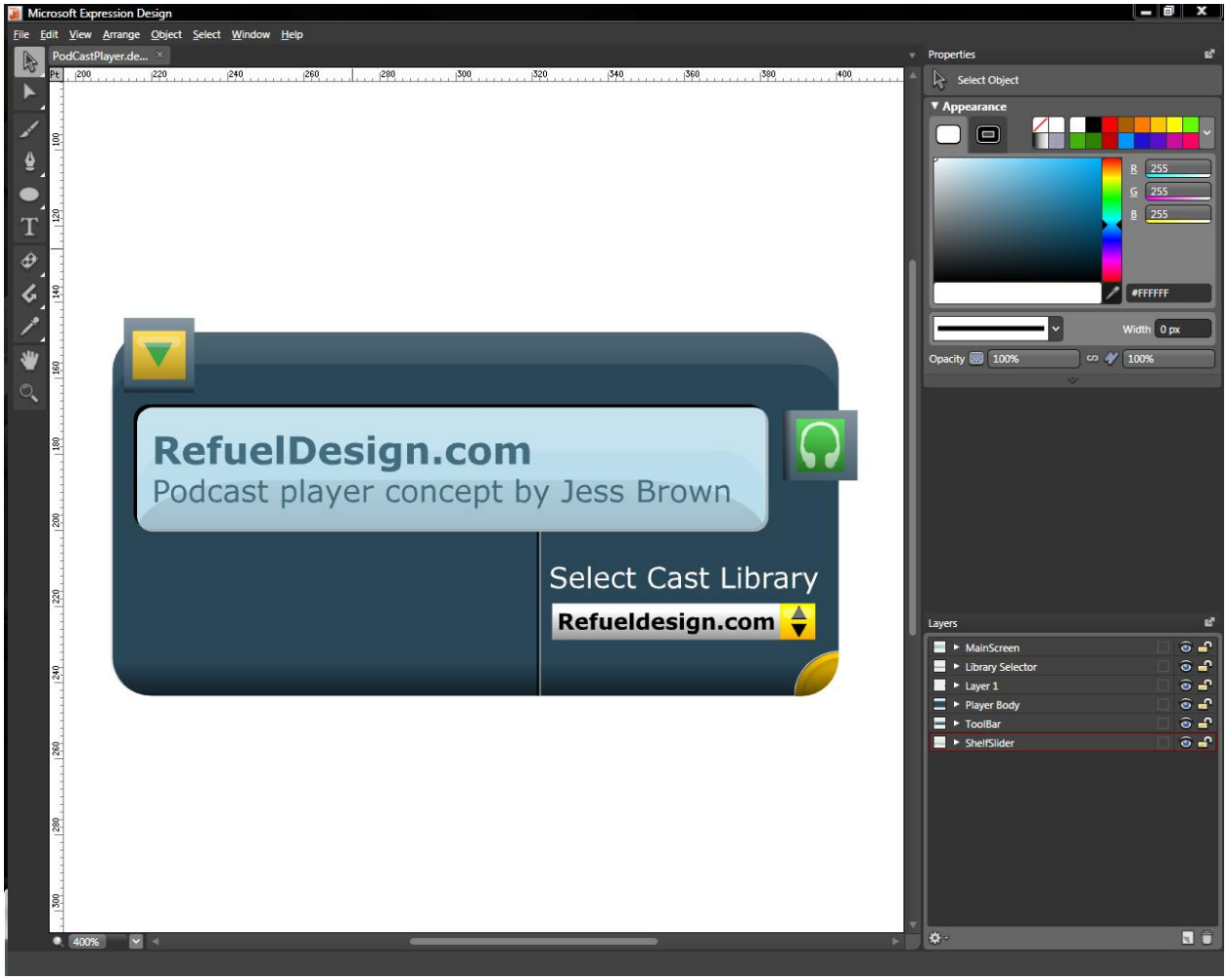
Như đã trình bày ở trên, WPF cung cấp rất nhiều tính năng cho những lập trình viên. Tuy nhiên, một công nghệ dù có hữu dụng đến đâu cũng cần một công cụ và môi trường tốt để phát huy những lợi điểm của nó. Đối với WPF, Microsoft cung cấp một công cụ chuyên dùng cho lập trình viên, và một công cụ khác phục vụ người thiết kế giao diện. Phần dưới đây đề cập ngắn gọn về những công cụ này.

4.1 Microsoft Visual Studio - Công cụ cho lập trình viên

Visual Studio là công cụ chủ đạo của Microsoft dành cho lập trình viên phần mềm. Microsoft cung cấp thành phần mở rộng cho Visual Studio 2005 cho phép lập trình viên có thể tạo ra các ứng dụng WPF. Phiên bản tiếp theo của Visual Studio (2008) có bổ sung thêm các tính năng phát triển ứng dụng WPF, trong đó bao gồm Visual Designer, môi trường thiết kế giao diện cho WPF. Sử dụng công cụ này, lập trình viên có thể tạo ra giao diện WPF một cách trực quan, trong khi sản sinh các đặc tả XAML tương ứng một cách tự động.

4.2 Microsoft Expression Design – Công cụ cho người thiết kế

Như đã giới thiệu trong phần trước, mục tiêu cơ bản của WPF là nâng cao vị thế của người thiết kế trong việc tạo giao diện người dùng. Để đạt mục tiêu này, ngoài XAML là công nghệ cốt lõi, Microsoft cũng đưa ra một công cụ mới cho phép người thiết kế làm việc thuận tiện hơn, đó là *Microsoft Expression Design* (Hình 0.10).



Hình 0.10. Giao diện của công cụ Microsoft Expression Design.

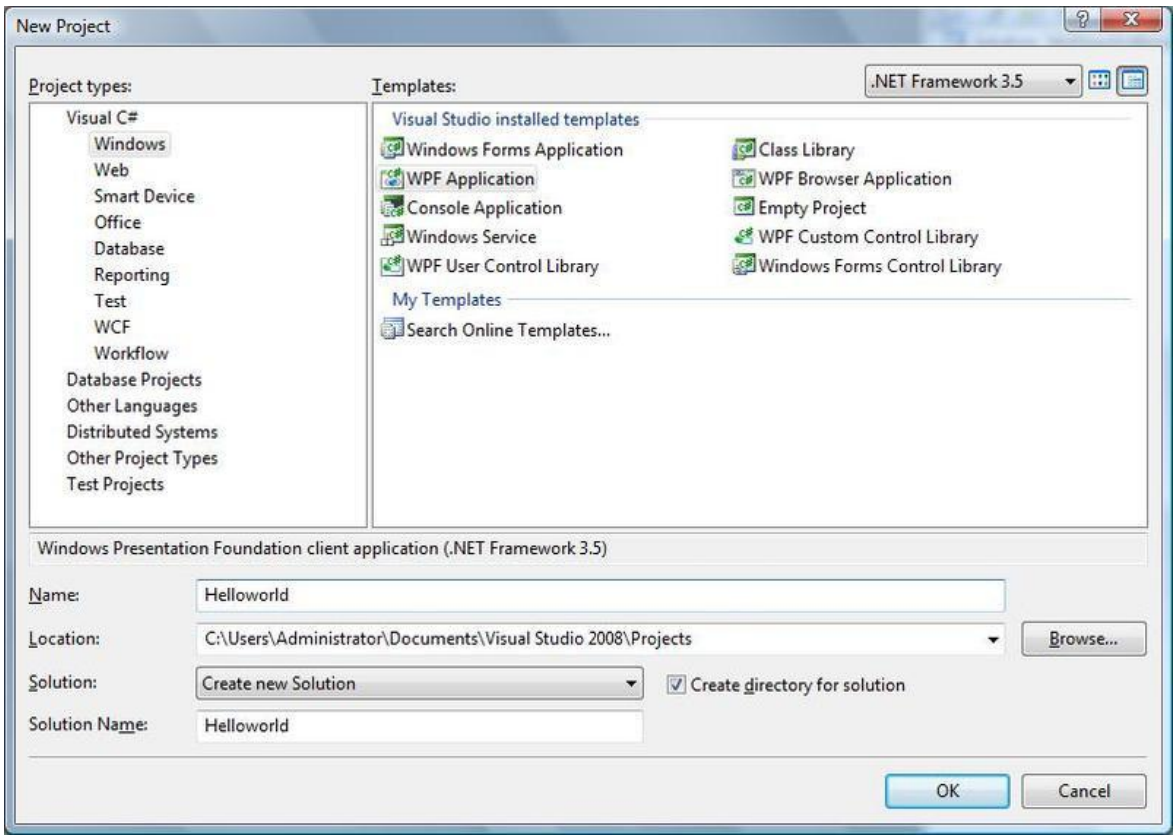
Microsoft Expression Design cung cấp những tính năng truyền thống của một công cụ thiết kế, cho phép người dùng làm việc theo cách quen thuộc. Ngoài ra, công cụ này đặc biệt tập trung vào việc hỗ trợ tạo giao diện cho các ứng dụng WPF. Tất cả các tính năng WPF mô tả ở trên đều sẵn có trong môi trường thiết kế này, và cho phép người dùng thiết kế một cách trực quan. Kết quả thiết kế được biểu diễn dưới dạng file XAML do công cụ này sinh ra, và sau đó có thể được nhập vào môi trường Visual Studio.

5. Ứng dụng đầu tiên với WPF – Hello World

Phần này giúp các bạn làm quen với lập trình WPF thông qua một ví dụ kinh điển: Hello World. Ứng dụng chỉ bao gồm một nút bấm có nhãn ban đầu là Hello World. Khi nhắp chuột vào nút, nút sẽ đổi tên thành "From Hanoi, Vietnam". Môi trường lập trình ở đây là bộ Visual Studio 2008 với .NET Framework 3.5.

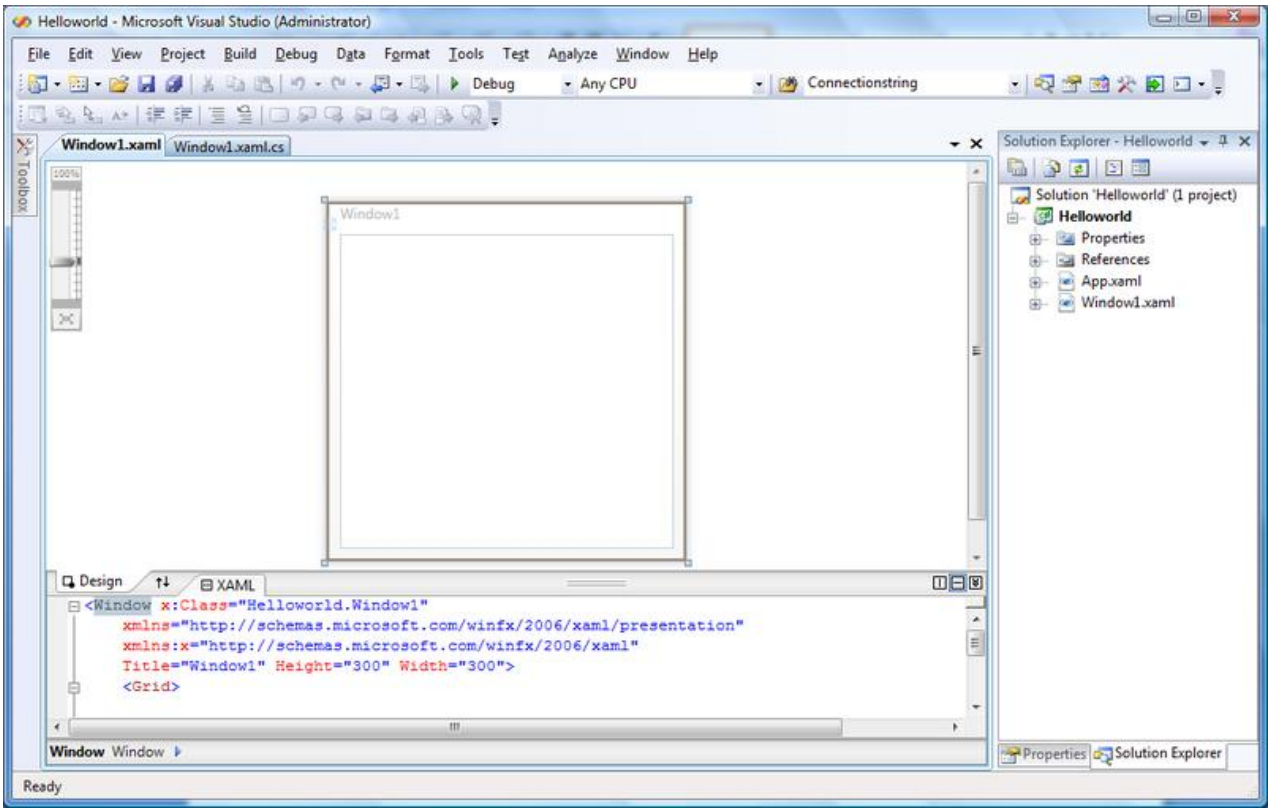
Quy trình thực hiện như sau:

5.1 Tạo ứng dụng WPF



Ở đây, ta chú ý chọn .Net Framework 3.5.

Giao diện thiết kế ứng dụng:



Chúng ta sẽ làm 2 phương pháp, một là viết code C# trực tiếp trong ứng dụng, hai là viết bằng mã XAML.

5.2 Tạo ứng dụng Hello World bằng code C#

Khai báo Button trong lớp Window1. Lớp Button được tạo ra từ namespace: `System.Windows.Controls;`

```
namespace Helloworld
{
    public partial class Window1:Window
    {
      // khai báo 1 button
      Button button;
      public Window1()
      {
          InitializeComponent();
      }
    }
}
```

Ở phương thức khởi tạo, ta sẽ lần lượt đặt thuộc tính cho nút bấm:

```
//tạo mới button
button = new Button();
//xác định thuộc tính cho button
button.Content = "Hello World";
button.LayoutTransform = new ScaleTransform(3, 3);
button.Margin = new System.Windows.Thickness(10);
//thêm phương thức xử lý sự kiện Click cho button
button.Click += new RoutedEventHandler(button_Click);
//đưa button vao Window
this.Content = button;
```

Để tạo phương thức xử lý sự kiện Click cho button, chúng ta chỉ thêm lệnh `button.Click +=` và nhấn Tab 2 lần. Code tự sinh ra như sau:

```
void button_Click(object sender, RoutedEventArgs e)
{
    // xử lý button khi người dùng click
}
```

Trong phương thức này, ta thêm vào dòng lệnh:

```
button.Content = "From Hanoi, Vietnam";//đổi nội dung (caption) của button
```

Nhấn F5 để biên dịch project và chạy kết quả.

5.3 Tạo ứng dụng Hello World bằng XAML

WPF hỗ trợ trên nền tảng XAML nên ta có thể tạo đối tượng hoàn toàn bằng ngôn ngữ XAML. Lưu ý, với cùng project trên, muốn viết đặc tả bằng XAML tương đương ta cần xóa bỏ phần mã trình C# cũ đi, vì C# và XAML không thể cùng sinh một đối tượng.

Cách thực hiện:

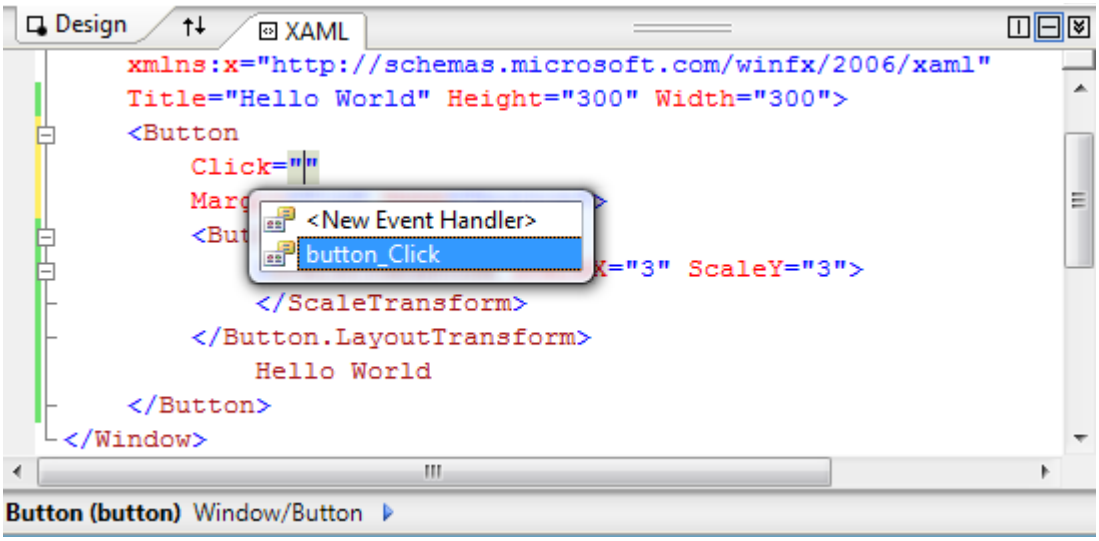
Mở file Window1.xaml tương ứng với file code Window1.xaml.cs ở trên.

Thêm đoạn mã đặc tả XAML tương đương sau:

```
<Window x:Class="Helloworld.Window1"
xmlns="http:://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schema.microsoft.com/winfx/2006/xaml"
Title="Window1" Height="300" Width="300">
  <Button Name="button" Margin="10">
    <Button.LayoutTransform>
       <ScaleTransform  ScaleX="3" ScaleY="3">
       </ScaleTransform>
```

```
    </Button.LayoutTransform>
    Hello World
  </Button>
</Window>
```

Giờ ta gán tên phương thức xử lý sự kiện *Click* cho nút bấm này:



F5 để chạy ứng dụng. Ta sẽ thu được kết quả tương tự như phần 5.2.

Câu hỏi ôn tập

1. ***WPF là gì?***

   WPF, viết tắt của *Windows Presentation Foundation*, là hệ thống API mới hỗ trợ việc xây dựng giao diện đồ hoạ trên nền Windows. WPF tăng cường khả năng lập trình giao diện của lập trình viên bằng cách cung cấp các API cho phép tích hợp nhiều dịch vụ trực quan trên cùng một ứng dụng: các form, control, hình ảnh 2 chiều, 3 chiều, văn bản cố định và thích nghi, in ấn, đồ họa vector, hoạt họa, móc nối dữ liệu, âm thanh và video. Thêm vào đó, các ứng dụng WPF còn có khả năng triển khai như một chương trình độc lập hoặc trên trình duyệt Web mà không phải thay đổi code giao diện. Sử dụng ngôn ngữ markup XAML có bản chất XML, WPF tạo cơ sở làm việc chung giữa người thiết kế giao diện và người lập trình.

   Là một bộ phận của .NET Framework 3.0, WPF sẵn có trong Windows Vista và Windows Server 2008. Đồng thời, WPF cũng có thể hoạt động trên nền Windows XP Service Pack 2 hoặc mới hơn, và cả Windows Server 2003.

2. ***Mục tiêu cơ bản của WPF là gì?***

   A. Cung cấp một nền tảng thống nhất để xây dựng giao diện người dùng.

   B. Cho phép người lập trình và người thiết kế giao diện làm việc cùng nhau một cách dễ dàng.

   C. Cung cấp một công nghệ chung để xây dựng giao diện người dùng trên cả Windows và trình duyệt Web.

   D. Cả ba mục tiêu trên.

   Câu trả lời: D

3. ***XAML là gì?***

   XAML, viết tắt của *Extensible Application Markup Language*, là ngôn ngữ đặc tả dựa trên XML được dùng để định nghĩa các đối tượng và thuộc tính của chúng, mối quan hệ cũng như sự tương tác. XAML đặc biệt được dùng trong các công nghệ của .NET Framework 3.0 (trong đó có WPF) như ngôn ngữ đặc tả giao diện người dùng (user interface - UI) nhằm mô tả cấu trúc và đặc tính của các phần tử UI, sự liên kết dữ liệu, các sự kiện và các đặc tính khác. XAML là mô hình có tính đột phá trong lĩnh vực tính toán trên Internet được chấp nhận rộng rãi trong nhiều hệ thống và bởi nhiều nhà cung cấp phần mềm.

4. ***Vai trò của XAML trong việc tạo môi trường làm việc chung giữa người thiết kế giao diện và người lập trình.***

   XAML đóng vai trò một ngôn ngữ chung giữa môi trường thiết kế giao diện và môi trường lập trình.

   Đối với người thiết kế, XAML không những cho phép người thiết kế dễ dàng mô tả, chỉnh sửa các đối tượng UI và các đặc tính của chúng, mà còn cho phép họ tương tác với các đối tượng này ở mức độ nhất định. Do vậy, tăng khả năng cảm quan của người thiết kế đối với giao diện. Ngoài ra, nó cũng giúp người thiết kế hạn chế những ý tưởng đồ họa không khả thi khi lập trình.

   Đối với người lập trình, nhờ một môi trường lập trình có khả năng tự động tái tạo giao diện đã thiết kế dựa trên file đặc tả bằng XAML do người thiết kế chuyển sang, người lập trình không cần tự mình tái tạo lại giao diện. Điều này giảm đi nhiều công sức và thời gian để phát triển giao diện, cũng như tránh những sai lệch giữa giao diện do người thiết kế và giao diện do người lập trình tái tạo.

   Vì cả môi trường thiết kế và lập trình đều có khả năng hiểu và sử dụng XAML, ứng dụng WPF có thể chuyển qua lại giữa hai môi trường phát triển để sửa đổi hay bổ sung giao diện một cách dễ dàng. Với tất cả những lợi điểm này, vị thế của người thiết kế trong việc xây dựng giao diện được nâng cao.

5. ***Trong ứng dụng WPF, phần giao diện người dùng có thể được viết bằng:***

   A. XAML

   B. Mã trình (C# hoặc Visual Basic)

   C. Cả A và B

   Câu trả lời: C

6. ***Trong ứng dụng WPF, việc xử lý các sự kiện có thể được viết bằng:***

   A. XAML

   B. Mã trình (C# hoặc Visual Basic)

   C. Cả A và B

   Câu trả lời: B

Tài liệu tham khảo

1. Windows Presentation Foundation, URL: http://msdn.microsoft.com/en-us/library/ms754130.aspx.
2. Introducing Windows Presentation Foundation, URL: http://msdn.microsoft.com/en-us/library/aa663364.aspx.
3. WPF Architecture, URL: http://msdn.microsoft.com/en-us/library/ms750441.aspx.
4. WPF Tutorial, URL: http://dotnetslackers.com/articles/silverlight/WPFTutorial.aspx.
5. WPF Tutorials, URL: http://www.wpftutorial.net/.