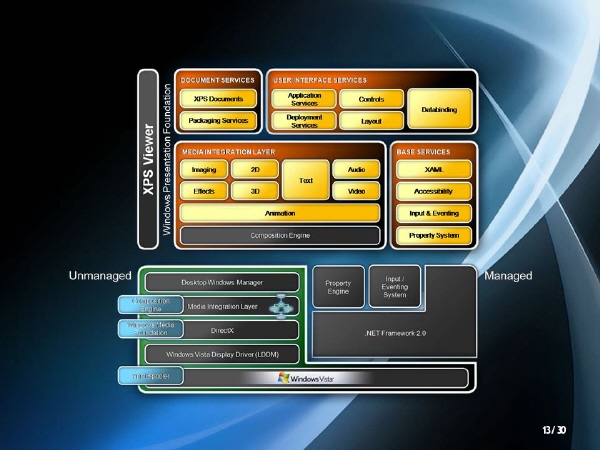
**01 WPF란?**



.Net Framework 2.0에서 3.x로 업그레이드 되면서 위 그림과 같이, WPF, WCF, WF, Cardspace와 같은 기술이 추가되었다.

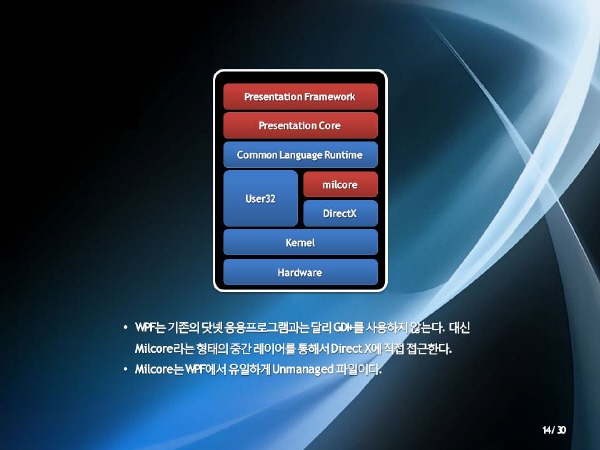
.Net Framework 버전을 3.x로 표현한 이유는 .Net Framework 2.0 이후 버전 부터는 하위 호환성을 가지고 있기 때문이다.

2.0에 조금 기능을 추가해서 3.0이 만들어지고 조금 더 추가해 3.5가 만들어진 것이다. 이 말은, 만약 개발자가 2.0으로 개발을 하고 있었다 하더라도 아무런 문제 없이 3.x로 버전업이 가능하다는 뜻이다.  
  
다시 본론으로 돌아가서 우리가 공부할 WPF는 .Net Framework 3.x에 추가된 새로운 기술 중 하나로, Windows Vista의 새로운 UX를 제공하기 위해 탄생 하였다. 기존 UI제작 방식과 달리, XML을 기반으로 한XAML이라는 언어를 통해 UI를 구현하며, 하드웨어 가속을 통해 성능을 최적화 할 수 있다.  
(Windows Vista를 위해 탄생했다고 했는데, Windowos XP SP2이상의 OS면 모두 사용가능하다.)



조금 더 내부를 살펴보면, Media Integration Layer에 다양한 엔진이 사실을 확인 할 수 있다.

기본적인 Imaging 과 Effect이외에 2D, 3D, Audio, Video, Animation 등의 엔진을 자체적으로 제공하여, 개발자가 쉽게 원하는 효과를 구현 할 수 있도록 돕고 있다.



위 그림에서 빨간색 박스 부분이 WPF에서 사용하는 부분이다.

.Net Framework 를 기반으로 작동되기 때문에, CLR위에 Presentation Framework와 Presentation Core가 있다는 것을 확인 할 수 있다.

여기서 한가지 중요한 점은 milcore라는 것인데, milcore는 WPF의 보다 나은 성능을 위하여,  
기존의 응용프로그램과 달리 GDI+를 사용하지 않고 DirectX에 직접 접근 하도록 돕는 레이어 이다.

milcore는 DirectX를 통해 kernel에 직접 접근 함으로, 어쩔 수 없이 unmanaged Code로 작성되어 있지만, 일반 개발자입장에서 milcore를 직접 다룰 일이 흔치 않으므로, 이정도 까지만 알고 있으면 될 것이다.



1. PresentationFramwork.dll

- 최상위 WPF 타입들을 포함한다.

- 윈도우와 패널을 비롯해서, 여러 종류의 클래스가 속한다.

- 추가적으로 고수준의 프로그래밍 추상화를 구현한 스타일 클래스도 포함

- 대부분의 클래스가 여기에 포함됨

- 도형개체와 컨트롤도 포함됨

2. PresentationCore.dll

- 모든 도형들의 개체들과 컨트롤이 파생되는 UIElement타입과 Visual 타입 포함

3. WindowBase.dll

- 기본적인 구성요소를 관리한다.

- Dispatcherobject 타입, DependencyObject 타입의 경우 WPF 외부에서 재사용이 될 수 있다.

4. milcore.dll

- WPF 랜더링 시스템의 핵심, 미디어 통합계층의 토대

- WPF일부임과 동시에 OS 필수적인 시스템 구성요소

5. WindowsCodect.dll

- 저수준 API

6. Direct3D

- 저수준 API, WPF 모든 그래픽 요소들은 Direct3D를 통해 랜더링한다.

7. User32

- 윈도우를 관리하거나 입력을 처리하는 프로그램의 상태와 관련된 작업을 처리

여전히 이전에서와 같이 WPF에서도 필요하지만, User32의 일반적인 컨트롤은 WPF에서 랜

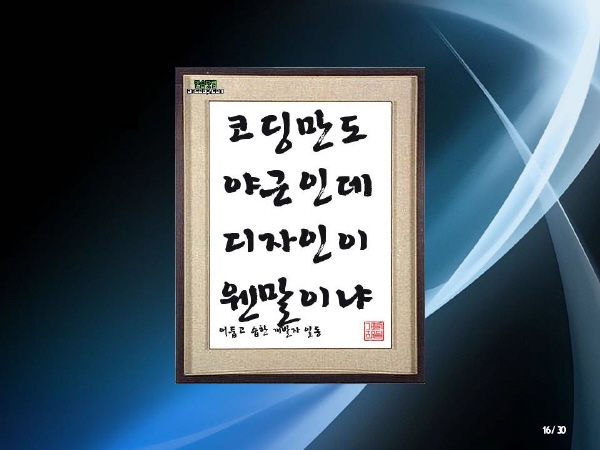
더링하는 부분은 제공되지 않는다.

**02. 왜 WPF를 사용해야 하는가 그리고 WPF를 사용하면 왜 좋은가?**



보통 우리는 디자인이 요구되는 프로젝트를 할때 디자이너와 함께 작업을 한다.  
(최근 UX가 발전하면서 Application의 UI 디자인은 필수요소라고 할 수 있죠.)  
디자이너는 Photoshop이나 Illustrator, Powerpoint 등을 사용해서 디자인을 제작합니다. (그림 가장왼쪽)  
그리고나서 각각의 세부항목들을 잘라서 JPG,PNG,PPT등의 포멧으로 변환하여 개발자에게 넘겨주죠?  
(간혹가다 잘라주지 않고 '통이미지'로 보내주는 사람이 있는데. 이건뭐...직접 잘라다 쓰란소리죠 ㅠ )  
뭐 어쨋든 개발자는 이렇게 이미지 파일을 받으면 C++,C#, VB와 같은 언어를 사용해 UI를 구현하게 된다.  
  
많은 개발자 분들이 느끼시겠지만, 개발의 한계를 이해 하지 못하는 디자이너 분들의 환상적인 디자인 때문에, 수많은 삽질과 노가다를 시작한다. 현실은 언제나 그렇듯이 냉혹하다ㅠ   
열심히 한다고 원하는 결과가 뚝딱 나오는건 아니죠ㅜ   
  
엄청난 삽질과 노가다를 통해 디자이너의 요구대로 구현을 하다보면 실제로 구현하기 너무 힘든 부분이 발생하는 경우도 있다. 위에서 예시를 든 콤보박스를 보면..  
  
간지가 좔좔 흐르는 콤보 박스에 각각 아이템들에는 체크박스도 포함되어 있네요..  
체크박스도 보통 체크박스는 아닌듯 하고.. 위와 같이 구현을 하려면 보통은 콤보박스를 새로 구현을 해야한다.  
체크 박스따로 만들고 체크박스 포함된 콤보박스 아이템 만들고 거기다 콤보박스 틀에 내부 기능까지... 해야 할 일이 너무 많다. ( 이럴경우를 보고 배보다 배꼽이 크다고 하죠.)    
  
개발자 입장에서는 이것도 해보고 저것도 해보다가 결국 타협을 하기 시작한다.

이건 기술적으로 안되, 저거 하려면 시간이 너무 오래 걸려 이렇게 자신과의 타협을 하면서 나온 결과물은 그림 오른쪽, 기존 콤보박스와 별다른 차이가 없죠.   
  
이제부터 디자이너와 개발자의 싸움이 시작됩니다. 디자이너는 왜 자기가 디자인 해준데로 구현을 못하냐고 소리치고, 개발자는 그게 말처럼 쉽게 되는게 아니니까 니가 해봐라 하고, 그럼 가능한선에서 예쁘게 만들라고 한다. 이말은 마치 디자이너가 어느정도 컨셉을 정해주면 개발자가 알아서 개발이 가능한 형태로 새로 디자인해서 개발을 해라 라는 말과 같죠..   
  
그럼 개발자들은 이렇게 말한다.



개발만으로도 할 일이 태산같은데, 디자이너가 할 일까지 해야 하겠냐는거죠..   
WPF는 이런 상황에 놓인 개발자들을 위한 해결책이라 할 수 있다.  
아래 그림을 살펴보도록 하자



아까와 비슷한 그림이지만 잘 살펴보면 이번에는 개발자의 콤보박스가 디자이너가 요구한데로 그대로 구현이 되어 있다는 것을 확인 하실 수 있다.

WPF를 사용한 결과인데, 어떻게 된일일까?   
  
눈에 띄는 것을 보면, 이전에 Photoshop이나 Illustrator를 JPG,PNG등의 포멧으로 전달 햇던 작업이 딸랑 **XAML**이라는 것 하나로 해결된 것을 확인 할 수 있다. XAML이란 WPF에서 UI를 표현하기 위한 언어로 디자이너와 개발자가 모두 사용 가능한 형태의 중간 언어라고 할 수 있다.   
  
"중간 언어" 라고 표현을 했는데, 그렇다면 디자이너들은 그럼 우리가 언어를 배워야 하는거 아니냐 라고 말할 수 있겟죠? 배워서 남주는건 아니니 배운다고 나쁠건 없지만, 배우기 싫으면 굳이 배우지 않아도 언어를 충분히 다룰 수 있다. Exression Studio와 같은 Tool을 사용해 디자인을 하고, XAML으로 저장만하면 되기 때문이다.



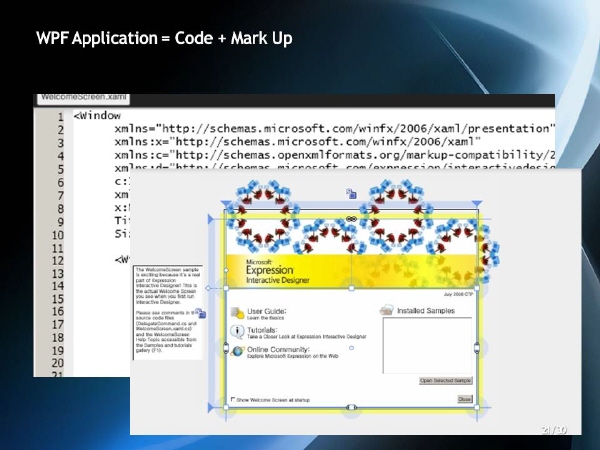
XAML은 디자이너의 디자인을 객체단위로 XML로 표현하는 방법이라 할 수 있다. 개발자는 이렇게 XML형태로 만들어진 디자인을 가져다가 그대로 사용하면 되는것이다.  정리하면 디자이너가 위와 같은 버튼을 하나 그렷다고 가정을 해보자.  
  
그럼 XAML(왼쪽)으로 표현될 것이며, 개발자가 버튼을 만들면 C#,VB.NET(중앙,오른쪽) 과 같이 코드로 표현하게 된다. 차이점은 코드를 통해 UI를 구현 할 경우 디자이너가 보내준 그림을 기준으로 개발자가 새로 구현 해야 하지만, 디자이너가 만들어준 XAML을 사용하면 개발자가 UI를 새로 구현할 필요 없이 내부 Logic만 구현하면 바로 사용이 가능 하다는 것이다.   
  
아직까지는 살짝 애매할 수 도 있는 부분이지만, XAML에 대한 이야기와 디자이너와 협력 부분은 뒷부분에서 더 자세히 다루도록 하겠다. (지금은 일단 XAML이라는게 있구나 라고 만 이해해 두면 되겠다.)



XAML은 Expression Designer나 Expression Blend와 같은 Tool에서 쉽게 제작이 가능하며 Expression Studio에서 만들어진 XAML은 곧바로 VisualStudio에서 사용이 가능하다.

한가지 덧붙이자면, VisualStudio에서 개발한 프로젝트를 Expression Blend에서 불러와 디자인 작업을 할 수 도 있고, Expression Blend에서 작업한 프로젝트를 Visual Studio에서 불러와 사용할 수 있다.  
  
여기서 중요한 점은 바로, UI와 Logic의 분리이다.  
이제 디자이너는 디자인만 하고, 개발자는 내부 Logic만 구현하면 되는 시대가 된것이다.

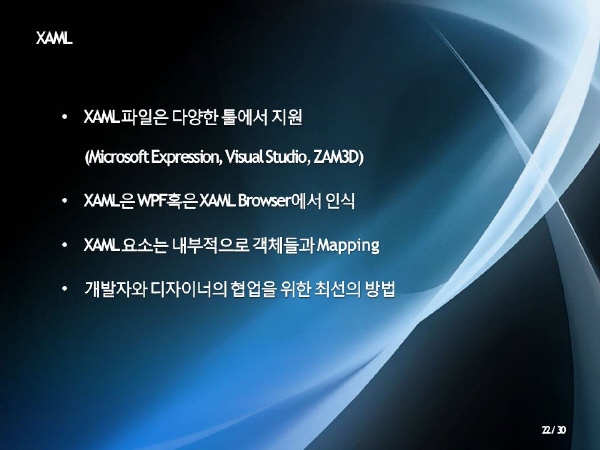
**03. WPF의 작동방식과 활용 방법**



WPF는 Code와 Markup의 조화라고 할 수 있다.

C#이나 , VB.Net을 사용하여 Logic을 구현하고 XAML을 사용하여 UI를 구현한다. WPF를 사용하는데 있어서, UI를 꼭 XAML로 구현 해야 할 필요는 없지만, 좀더 직관적이고 편리한 XAML을 사용하여 UI를 구현하는 것을 권장한다.  
  
XAML이 XML을 기반으로 한 Markup Language다 보니, XAML에서 표현되는 모든 UI는 계층적 구조를 가지고 있다. 위 그림에서는 잘 보이지 않지만, 최상위 노드가 Window가 되고, 최상위 노드의 자식 객체들이 올라가는 형식이다.   
  
기존의 Win32나 Winform등도 어느정도 계층적 구조를 가지고 있지만, 특수한경우(Form,Frame, Panel 등) 를 제외 하곤 자식객체를 포함 할 수 없는 것이 대부분이다.

하지만, WPF는 가장 기본적인 객체(Label, Textblock)등을 제외 하곤 모두 자식 객체를 포함 할 수 있기 때문에 별다른 수정 없이 원하는 형태로 변환 하여 구현 할 수 있다. 객체를 개발자 임의로 모양 등을 변경 하는 방법에 대해서는 뒷부분에서 설명하도록 하겠다.



XAML에 대해 정리를 하면 위와 같이 정리 할 수 있다.

XAML은 Expression Blend나 Expression Designer, Visual Studio등과 같은 Microsoft 공식 Tool에서 지원하며 ZAM3D와 같은 3rd Tool에서도 지원한다.

XAML은 당연한 말이지만, XAML을 인식할 수 있는 XAML Browser에서만 작동되며, Internet Explorer이 대표적입니다.  
  
XAML로 표현한 객체들은 모두 Code Level에서 접근이 가능하기 때문에 디자이너가 XAML로 디자인을 표현하면, 개발자가 별다른 수정 과정 없이 디자이너의 결과를 그대로 활용할 수 가 있다.

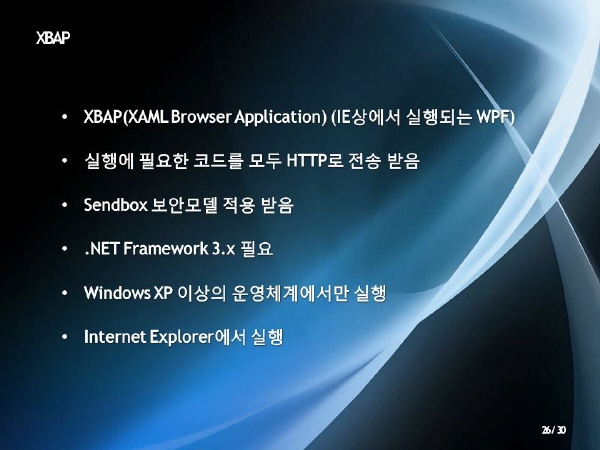


WPF를 활용하는 방법으로는 크게 3가지로 나눌 수 있다.   
처음부터 끝까지 WPF를 사용하여 Application을 구현하는 Full WPF Application방식과 Internet Explorer등에서 다운로드 받는 형식으로 작동되는 XBAP방식, 그리고 플랫폼에 종속적이지 않고 Web Browser만 있다면 사용이 가능한 Silverlight 방식이 있다.



먼저 Full WPF Application 이다.

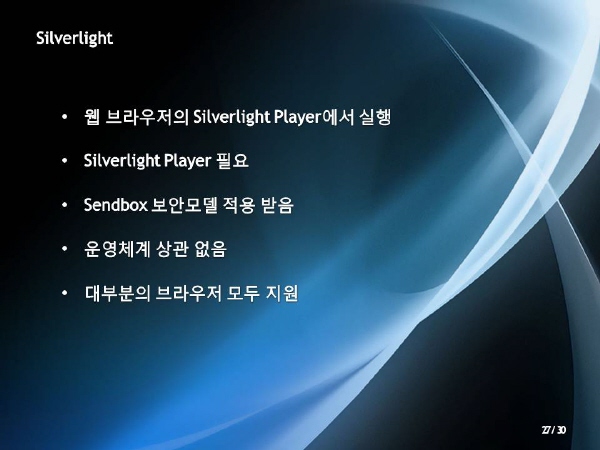
이름에서도 알 수 있듯이, WPF의 모든 성능을 이끌어 낼 수 있는 방식이다. Window에서 시작하여 하나하나 객체를 추가 해서 구현을 하며, Full WPF Application을 사용하기 위해서는 사용자 컴퓨터에 .Net Framework 3.x이상의 버전이 필요하다. 그리고 Smart Client방식으로도 활용이 가능하다.  
  
WPF는 Vista의 UI를 위해 탄생한 기술이지만, Windows XP SP2 이상의 OS에서도 실행이 가능하다.



다음은 XBAP형식이다.

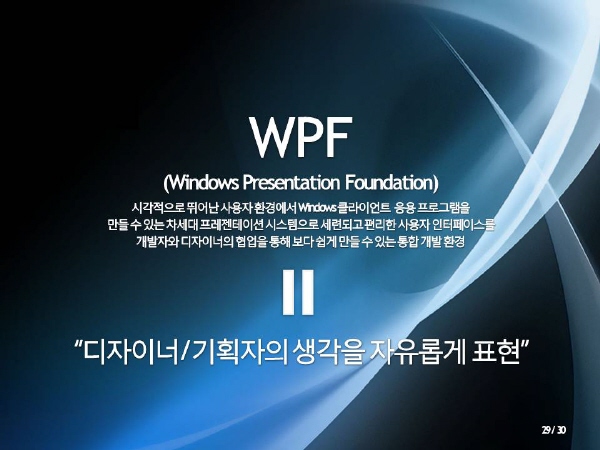
XBAP은 XAML Browser Application의 약어로 XAML을 Browsing 할 수 있는 XAML Browser에서 동작하는 형태의 Application으로 Windows에서는 Internet Explorer를 사용 할 수 있다.

Application 실행에 필요한 모든 코드와 리소스를 HTTP를 통해 전송 받으며,Sendbox모델을 적용받는다.   
  
FULL WPF Applicatoin과 구현 할 수 있는 기능은 비슷하지만, 가장 다른 차이점은 SendBox Model을 적용받는다는 점과 XAML Browser에서 실행된다는 점이 있다.   
  
마찬가지로 XBAP형식을 사용하기 위해서는 사용자의 컴퓨터에 .Net Framework 3.x가 설치되어 있어야 하며, Windows XP SP2이상의 버전이 필요하다.



마지막으로 Silverlight이다. Silverlight는 Full WPF Application과 XBAP과 달리 OS,Browser에 상관없이 모두 작동한다.

즉, 플랫폼 종속적이지 않다는 말이다. Silverlight는 Web Browser위에서 작동되며, Silverlight Player가 설치되어 있는 컴퓨터에서만 작동한다. 우리가 Web Browser에서 Flash 를 보기 위해 Flash Player를 설치하는 것과 마찬가지 이다. 마찬가지로 SendBox 보안 모델을 적용 받으며,   
  
위에서 소개한 Full WPF Application과 XBAP과 가장 큰 차이점은 .Net Framework 가 설치되어 있지 않아도 된다. 이유는 Silverlight Player에 .Net Framework의 Core부분이 포함되어 있기 때문인데, Core 만 포함되어 있는 만큼 Full WPF Application에 비해 구현 가능한 범위가 좁다.  
  
WPF Application을 개발하려고 할때 가능한 구현 범위를 정리하면 아래와 같이 정리 할 수 있다.  
  
**Full WPF Application > XBAP > Silverlight**



[정리]

1. WPF

- 윈도우 기반 어플리케이션의 그래픽 랜더링을 위한 플랫폼

- 풍부한 UI, data : 사용자 경험의 폭을 넓이는데 큰 힘이 될 것

- UX : User eXperience 🡺 기술이 인간을 향해 발전하는 공학의 목적(공학의 따뜻한 배려)

🡺 WPF는 시각적으로 뛰어난 사용자 환경, 사용자 경험을 제공하는 응용프로그램을 만들 수

있도록 해주는 차세대 프리젠테이션 시스템

* .NET 3.0 : .NET Framework 2.0 + WPF + WCF + WF + CardSpace

2. WPF의 등장 배경

- 사용자 경험(UX) 증가

- 어플리케이션개발의 디자이너와 프로그래머의 생산성 증대 요구

🡺 XAML, C# 분리로 원활한 협업의 가능

* XAML 제공으로 개발자와 이자이너 모두 손쉬운 개발과 디자인을 할 수 있도록 도와줌

3. WPF의 특징

- 효과적인 GUI, UI 개발을 위한 서브시스템

- Windows7 개념의 UX 제작 기술 제공

- XML로 UI를 설계 가능 ; XAML

- 새로운 디자인 도구 제공(디자이너들의 Expression brend)

- 통합 응용 프로그램 제작 가능 : .NET Framework

- 백터 그래픽 처리방식의 랜더링 엔진 탑재

4. WPF의 기능

- 하드웨어 가속

🡺 모든 WPF 드로잉 작업은 DirectX를 통해 수행 : 비디오카드의 이점을 최대한 활용

* 해상도 독립성
* DPI 설정에 따라 모니터, 디스플레이 선호에 유연성을 제공
* 고정되지 않은 컨트롤 외형 등

5. WPF 어플리케이션의 종류

- Silverright, XBAR, Full WPF application

6. 개발환경

Visual Studio 2008 SP1 설치 : 통합

Microsoft Expression Studio2 : 주로 Expression Blend사용 – 디자이너

기타 : XMAL, PAD, ZAM#D