Wpf چیست؟

WPF سر آغاز سه کلمه Windows Presentation Foundation می باشد

هر کسی که تا به حال در محیط های گرافیکی و یا به اصلاح برنامه نویسان، محیط های ویژوال، برنامه نویسی کرده باشد، یقینا با مفاهیم Windows Application ها که گاها به صورت مخفف WinApp نیز نامیده می شوند، آشنا می باشد. این نوع برنامه نویسی همزمان با ورود سیستم عامل های ویندوز در دنیای کامپیوتر شروع شد و روز به روز با به وجود آمدن زبان های متفاوت جایگاه محبوبتری نزد برنامه نویسان پیدا کرد.

Windows Application ها از API های سیستم عامل مربوطه (که اکثرا ویندوز XP نیز می باشد) برای ترسیم عناصر گرافیکی یا همان عناصر ویژوال، استفده می کنند. به عنوان مثال برای ترسیم انواع دکمه ها، فرم ها و بسیاری از عناصردیگری که با آن ها آشنا هستید، از توابع API ویندوز کمک گرفته می شود. همین مسئله باعث ایجاد محدودیت برای برنامه نویسان در ایجاد کنترل های سفارشی با ظاهر دلخواه خود شده بود. اگر چه با ابزار های گرافیکی که در دات نت فریم ورک 2.0 نیز وجود داشت، می توانستیم تا حد خوبی اقدام به ایجاد کنترل های مورد دلخواه خود را بکنیم، اما این موضوع نیاز به دانستن اطلاعات زیاد در مورد ایجاد کنترل های سفارشی و همچنین نوشتن گاها کد های بسیار زیادی جهت ایجاد کنترل مورد نظر می بود. این به آن دلیل بود که قالب و اساس اولیه کنترل ها بسته بود و نمی توانستید به راحتی کنترل ها را شخصی سازی نمایید. در بهترین حالت، یک برنامه نویس ماهر میتوانست با ارث بری از کلاس Control اقدام به ایجاد یک کنترل جدید با ظاهر و امکانات مورد نظر خود بکند.

بنابراین، wpf به وجود اومد تا دیگه برای طراحی فرم های application هیچ محدودیتی وجود نداشته باشد.

اساس WPF

ساختار ویژگی های جدید در WPF، در واقع یک ساختار جدید و قدرتمند که براساس DirectX و APIهایِ گرافیکی سریعِ سخت افزاری، که معمولاً در اکثر بازی های کامپیوتری مدرن استفاده می شوند، پایه گذاری شده است.

پایه و اساس WPF بر Directx استوار می باشد. این موضوع سبب می شود که بتوان از بسیاری از جنبه های گرافیکی بدون ایجاد سربار اضافی بر روی برنامه بهره برد و در واقع برنامه هایی با ظاهر هایی بسازید که ساختن آن ها با برنامه نویسی های پیشین یا غیر ممکن و یا متحمل کار بسیار زیادی بوده است. اگرچه نقطه قوت این تکنولوژی اعمال گرافیکی، انیمیشن و .. می باشد، ولی این بدان معنی نیست که نمی توان با WPF اقدام به ایجاد فرم ها و کنترل های سابق نمود.

WPF هنوز از User32 برای انجام سرویس های خاصی مانند اجرای دریافت ورودی ها ودسته بندی اینکه هر برنامه دربردارنده چه قسمتی از صفحه نمایش می باشد. با این حال کلیه عملیات ترسیمی توسط DirectX انجام می پذیرد.

ویژگیهای WPF

گفتیم تکنولوژی WPF به روشی متفاوت از winappو netframework. عمل می کند. در واقع قدرت WPF در این است که اساس و پایه هر کنترلی مانند برنامه نویسی قبل، بسته نیست و این شما هستید که به WPF خواهید گفت که متن روی کنترل را به چه صورتی طراحی کنید. یا پس زمینه کنترل یا کناره های آن را به آن صورتی که شما می گویید طراحی کند. به همین منظر نیز دارای ابزارهای بسیار زیادی جهت کار برای طراحی کنترل های شما مهیا می کند. ابزارهایی مانند قلم موهای گرادیان با تعداد رنگهای نامحدود، انواع ابزارهای گرافیکی برای ترسیم شکل دلخواه شما، امکان ایجاد افکت های بسیار زیبا و متنوع بر روی هر قسمتی از کنترل که بخواهید، وجود افکت های از پیش تعریف شده، امکان طراحی های 2 بعدی و نیز 3 بعدی، امکان ایجاد انیمیشن و... علاوه بر این موارد، WPF امکان کار با اسناد متنی، کنترل کردن بر روی نحوه Print شدن آن ها و ... را برای شما مهیا می سازد.

انیمیشن، صدا و تصویر:

همانطور که پیش تر نیز توضیح داده شد، علاوه بر انجام اعمال بسیاری که می توانید، با اشکال انجام دهید، اعم از چرخش، بزگر نمایی، کوچک نمایی و ...، نیز می توانید اقدام به ایجاد انیمیشن های زیبا توسط WPF نمایید. همچنین قادر خواهید فایل های صوتی و ویدیویی را به خوبی به کار بگیرید.

عدم وابستگی WPF به رزولوشن:

یکی از جنبه های فوق العاده مفید و قوی WPF عدم وابستگی آن به رزولوشن صفحه نمایش است. یک

برنامه نویس حرفه ای در WPF حتی المقدور از خواص Width و Height عناصر برای چیدمان آن ها استفاده نخواهد کرد. این به این دلیل است که برنامه های تحت ویندوزی که تا کنون و با تکنولوژی های موجود نوشته می شدند(می شوند) وابستگی زیادی به رزولوشن صفحه نمایش دارند. به عنوان مثال فرم های شما، که در صفحه نمایش شما با رزولوشن 1024 * 768 به خوبی طراحی شده اند، ممکن است در یک کامپیوتر دیگری با رزولوشن بالاتر از ان (این امر در Laptop ها بسیار معمول می باشد. علاوه بر اینکه ان ها در بیشتر مواقع از تراکم 120 DPI استفاده می کنند. در صورتی که مونیتور های CRT معمولا از تراکم 96 DPI استفاده می کنند. "گر چه قابل تغییر می باشد") کوچک شود، و بر عکس، در یک سیستم با رزولوشن پایین، قسمتی از فرم های شما از صفحه نمایش خارج گردد.

اما با WPF این مشکلات مرتفع می گردد. دلیل آن هم استفاده از سیستم خاصی برای اندازه گیری اجزاء و عناصر برنامه شما، می باشد. عناصر، اعم از دکمه ها، فرم ها و هر شی قابل اندازه گیری با واحدی با نام DIU Unit (Device Independent) اندازه گیری می شوند. هر یک DIU معادل با 96/1 (1 تقسیم بر 96) هر اینچ می باشد. در واقع می توان گفت هر DIU در صفحه نمایشی با تراکم پیکسل استاندارد یعنی 96 DPI ، دقیقا برابر با 1 پیکسل فیزیکی در صفحه نمایش می باشد. حال اگر از DPI بالاتری استفاده گردد، طبیعتا هر یک DIU (در همان رزولوشن قبلی)کمتر از 1 پیکسل خواهد شد.

حال WPF با اندازه گیری DPI در هر رزولوشنی که با فرمول مشخصی محاسبه می شود، می توانید سایز مناسب عناصر شما را محاسبه کند. این روش باعث می شود که نمایش یک کنترل مانند Button در رزولوشن 1024*786 و با 96 DPI تراکم، با نمایش آن در رزولوشن 1600*1200 و با تراکم 120 DPI یکسان باشد.

اهداف WPF

یکی از مهمترین اهداف WPF استفاده از GPU به جای CPU جهت انجام روتین های پیچیده گرافیکی می باشد که این امر باعث آزاد بودن CPU بوده که میتواند به پردازش های دیگر در سیستم رسیدگی کند.

به طور کلی برنامه های ویندوزی از دو امکان، توابع User32 و GDI/GDI+ برای ترسیم عناصر گرافیکی استفاده می کنند که User32 امکان ترسیم عناصر ویژوال را با ظاهر عادی مهیا می کند. عناصری مانند فرم ها، دکمه ها و ... و GDI/GDI+ امکانات گسترده تری را جهت ایجاد برخی اعمل گرافیکی مانند ایجاد گرادیان ها و ... را مهیا می کنند.

شرکت ماکروسافت به دلیل محدودیت هایی که در هر یک از دوبخش فوق، وجود داست، اقدام به ایجاد کتابخانه سطح بالایی به نام DirectX کرد.(حرف X می تواند جایگزین کلماتی مانند Sound و .. شود).
این ابزار که امروزه نیز از آن استفاده های زیادی میشود،(از جمله در ایجاد بازی های سه بعدی و ...) با بهره گیری از توان کارت های گرافیکی با بهره بری بالا، حداکثر توان آن را برای ایجاد گرافیک های قوی به کار می برد.
اما با قدرت زیاد این کتابخانه، به دلیل برقراری ارتباط مشکل با ان و نیاز به کد نویسی های زیاد، این ابزار بیشتر در تهیه بازی ها و برنامه های گرافیکی مورد استفاده قرار گرفت و جایگاه زیادی در توسعغه برنامه های تجاری پیدا نکرد.
تکنولوژی WPF تمامی این مشکلات را مرتفع کرد و در واقع کاربر را از درگیر کردن نوشتن کدهای زیاد و گاه طاقت فرسا به صورت مستقیم در DirectX ، رهایی داد. WPF از تمامی قدرت DirectX جهت ایجاد گرافیک های 2 بعد، 3 بعدی، ایجاد انیمیشن ها، استفاده می کند. همچنین ابزار های بسیاری را جهت طراحی کردن در اختیار شما قرار می دهد. علاوه بر این DirectX به جهت اینکه به خوبی با مفاهیم Texture، Gradient و ... تطبیق پیدا می کند، درارای سرعت بالاتری نسبت GDIو GDI+ می باشد. به این دلیل که این تکنولوژی ها برای رندر کردن از روش پیکسلی و الگوریتم های آن که اصطلاحا Pixel By Pixel Instruction گفته می شود، استفاده می کنند.
یکی دیگر از مشکلاتی که کار کردن با DirectX به صورت مستقیم وجود داشت(دارد) به دلیل نوع بهینه سازی و نحوه رندر کردن اشکال توسط کارت های ویدیویی متفاوت بود، که با WPF این مشکل نیز مرتفع شده است
نکته: WPF زبان XAML را برای ایجاد واسط های کاربری برنامه های خود به کار می گیرد.
هر سند XAML در WPF می تواند نگهدارنده آبجکت های WPF باشد. این آبجکت های می توانند در بالاترین سطح، پنجره های باشند و یا تنها یک آبجکت خط و یا یک مستطیل طراحی شده توسط شما باشد.

WPF Projects

بعد از آشنایی مختصری که با WPF پیدا کردیم، میخواهیم ببینیم که WPF چه پروژه هایی رو support می کند

WPF بطور کلی دو نوع مختلف از پروژه را پشتیبانی می‌کند، اما این بدین معنی نیست که تنها دو نوع پروژه را می‌توان با آن ایجاد کرد! برخلاف آنچه در IDE می‌بینیم شما قادرید که سه نوع پروژه مختلف را بوسیله **WPF** ایجاد نمایید.

- Windows Application
- Navigation Application
- XBAP Application

Windows Application

این برنامه‌ها از کلاس **Window**به ارث رفته‌اند و آن کلاس نیز از **ContentControl**به ارث رفته است. این نوع برنامه‌ها تقریبا همان جایگزین‌های برنامه‌های ویندوزی سابق هستند و کاربرد مشابه دارند. نکته مهمی که باید در نظر گرفت آن است که این نوع از برنامه‌ها به تمامی منابع سیستم‌عامل اعم از سیتم فایل، رجیستری و ... دسترسی کامل دارند.

برای ایجاد چنین برنامه‌هایی کافیست که از منوی File بر روی گزینه New>Projectکلیک کند و از پنجره باز شده آیتم WPF Application را انتخاب کنید.

نحوه نمایش پنجره‌ها به دو صورت **Show** و **ShowDialog** است که اولی اجازه کار بر روی سایر پنجه‌ها را هم به کاربر می‌دهد و دومی برای ادامه کار از کاربر درخواست انجام یک عملیات (مثل فشردن دکمه بله و ...)را می‌کند.

Navigation Application

این برنامه‌ها هم مانند Windows Application ها هستند. با این تفاوت که بجای اینکه از کلاس Windowبه ارث رفته باشند از کلاس **Page**به ارث رفته‌اند. ظاهر این برنامه‌ها مانند برنامه‌های تحت وب است (چراکه در مرورگر اجرا می‌شوند) . نکته مهم آن است که این برنامه‌ها هم به تمامی منابع اصلی ویندوز مانند سیستم فایل، رجیستری و ... دسترسی دارند.

این برنامه‌ها بطور پیش‌فرض وقتی ایجاد می‌شوند توسط **NavigationWindow**میزبانی می‌شوند که یکسری امکانات خاص مانند دکمه‌های back و forward و) journalصفحاتی که دیده شده‌اند) را برای کاربر نهایی فراهم کرده است .این برنامه‌ها برای استفاده در **Internet Explorer**کاملا بهینه‌سازی و تست شده‌اند.

برای ایجاد یک NavigationAplication باید همه مراحلی را که برای ایجاد Windows Application سپری کردید را مجددا سپری کنید.

در انتها باید فایل Window1.Xaml را **حذف** (delete)کنید و سپس اقدام به اضافه کردن یک آیتم جدید بکنید.

برای اضافه کردن یک آیتم جدید بر روی پروژه کلیکراست کرده و گزینه Add New Item... را انتخاب کنید و از پنجره باز شده، آیتم Page را انتخاب کرده و بر روی OK کلیک کنید.

در فایل App.Xaml مقدار صفت StartupUri را از Window1.Xaml به Page1.Xaml تغییر دهید.

XBAP Application

این برنامه‌ها هم خیلی به برنامه‌های Navigation شباهت دارند. به این معنی که آنها هم از **Page** Classبه ارث رفته‌اند. شاید یکی از تفاوت‌های اصلی این دو نوع برنامه با هم آن باشد که این برنامه‌ها فایل Installer ندارند به این معنی که نصب نمی‌شوند و مانند یک وب‌سایت با آنها برخورد می‌شود و براحتی یک کلیک کردن در دسترس هستند. و دقیقا به این دلیل که آنها در سیستم نصب نمی‌شوند، **به منابع کاملا محدودی از سیستم‌عامل دسترسی دارند** .آنها **نمی‌توانند** از منابع ویندوز مانند سیستم‌فایل، رجیستری، پایگاه‌داده نصب شده در کلاینت و ... دسترسی پیدا کنند. این برنامه‌ها با مجوز Internet Zone در Internet Explorer اجرا می‌شوند. گرچه این امکان وجود دارد که XBAP ها را در یک محیط **Full Trust**اجرا کرد، اما در این صورت توصیه شده که از **Navigation Application**ها استفاده شود.

در حالت Standard internet Security محدودیت‌های زیر برای برنامه‌های XBAP وجود دارد.

- محدودیت دسترسی به منابع سیستم (کلاینت) بجز) Isolated Storage پایین‌تر در این مورد توضیح داده شده(
- دسترسی به رجیستری
- ایجاد stand-alone windows مانند دیالوگ باکس‌ها
- دسترسی به پایگاه‌داده کلاینت
- استفاده از قابلیت Drag-and-Drop ویندوز
- استفاده از سرویس‌هایWCF

قابل ذکر است که این برنامه‌ها در Internet Explorer 6.0 به بعد بخوبی اجرا می‌شوند.

همانطور که قبلا گفته شد این برنامه‌ها به منابع سیستم عامل دسترسی ندارند و اگر بخواهید که فایلی را در سیستم کلاینت بنویسید و یا بخوانید چنین امکانی وجود ندارد. اما یک محیط کاملا ایزوله شده به حجم **512** kilobytesدر اختیار برنامه‌نویسان است که در آن به نوشتن و خواند فایل‌ها بپردازند. این محیط Isolated Storageنام دارد. (در آینده‌ای نزدیک مقاله‌ای در این ارتباط خواهم نوشت(

برای ایجاد یک XBAP Application از منوی File روی گزینه New>Project کلیک کرده و سپس آیتم WPF Browser Application را انتخاب نمایید.

انتخاب بین انواع مختلف پروژه‌های WPF

باتوجه به مطالب بالا با درنظر گرفتن نوع ظاهر برنامه (برنامه‌های با ظاهر قبلی ویندوز، تحت وب بودن و...) و نیز با توجه به نیازمندی‌های برنامه)مثلا اینکه باید به منابع سیستم دسترسی داشته باشد یا خیر) می‌توان اقدام به انتخاب یکی از این پروژه‌ها کر

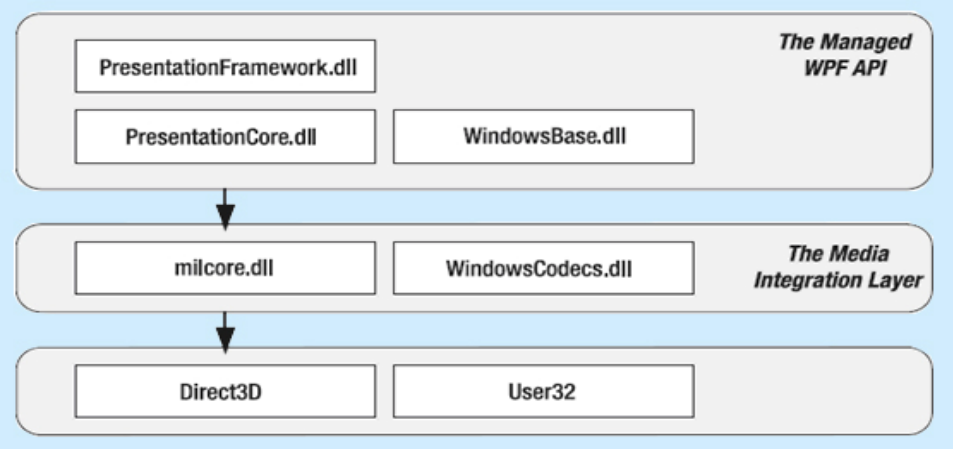
معماری WPF

تکنولوژی WPF یک تکنولوژی چند لایه می باشد.در بالاترین لایه آن اسمبلی های پایه ای و اسای WPF قرا گرفته اند که تماما به صورت کد های مدیریت شده سی شارپ می باشند. این لایه شامل API های PresentationFramework.dll ، WindowsBase.dll و PresentationCore.dll می باشد که در واقع برنامه شما با این اسمبلی ها ارتباط خواهد داشت.

در لایه زیر آن، کامپوننت مدیریت نشده milcore.dll قرار دارد. تمامی کدهای نوشته شده توسط شما، از طریق لایه اول و ارتباط لایه اول با لایه دوم و کامپوننت مذکور تبدیل آبجکت های مورد نظر می گردد.
در واقع دلیل اینکه کامپوننت milcore.dll به صورت مدیریت نشده می باشد، این است که این کامپوننت بایستی ارتباط تنگاتنگی و مجتمع شده ای با Direct3D داشته باشد و نیز دارای کارایی بسیار بالایی از هر لحاظی باشد.

Direct3D در لایه زیرین milcore.dll قرار گرفته است که به صورت یک API سطح پایین می باشد و در واقع به نوعی موتور WPF به همراه milcore نیز به حساب می آید.

در شکل زیر بخش های مختلف معماری WPF نشان داده شده اند



همانطور که گفته شد، برنامه شما در بالاترین سطح با API های سطح بالا که در واقع پایه واساس WPF را تشکیل می دهند، ارتباط بر قرار می کنند. در ادامه به تشریح هر یک از این کامپوننت ها و ابزار ها خواهم پرداخت:

PresentationFramework.dll : این اسمبلی در واقع تمامی آبجکت های سطحه بالا و در واقع به نوعی بالاترین سطح از آبجکت های WPF مانند Windows ها (که بالاترین سطح در برنامه های WPF را در مدل برنامه نویسی WPFApplication دارا می باشد) و Panel ها که از دیگر اجزاء اساسی برنامه های WPF می باشند،را نگه داری می کند.

می توانید Windows ها را به مانند Form ها در برنامه های معمولی در نظر بگیرید. همچنین Panel، کلاس پایه برای تمامی کنترل های Container از جمله Grid (که مهمترین آن ها و پر کاربرد ترین آن ها می باشد)، StackPanel، Canvas و ... می باشد.

Presentationcore.dll : شامل نوع های پایه از جمله UIElement و Visual می باشد که تمامی اشکال و کنترل های از این کلاس ها ارث بری می کنند. در قسمت بعدی نمودارسلسله مراتبی کلاس های WPF را مشاهده خواهید کرد.

Milcore.dll : در واقع هسته اصلی WPF در رندر کردن آبجکت ها به آبجکت هایی که لایه زیرین خودش یعنی Direct3D نیاز دارد، می باشد. علاوه بر این در ویندوز ویستا، مدیر پنجره های دسکتاپ یعنی Desktop Windows manager (که عمل مدیریت پنجره های دسکتاب را بر عهده دارد) از همین

کامپوننت استفاده می کند. در واقع شما می توانید با فراخوانی DWM ، به فرم ها، یا صحیح تر بگویم به پنجره های برنامه خود، افکت هایی که پنجره های ویندوز ویستا دارا هستند را اضافه نمایید.

نکته:

دقت کنید که این افکت ها بر روی ویندوز ویستا به تنهایی قابل پیاده سازی هستند. گرچه ابزار ها و کامپوننت های Cross نیز برای این کار نوشته شده اند و لی به صورت عادی برنامه هایی که بر روی ویندوز ویستا اجرا می شوند، می توانند قابلیت افکت های ویندوز ویستا را دارا باشند

WindowsCodec.dll : یک API سطح پایین می باشد که قابلیت اعمال، کارهای زیادی را بر روی عکس ها، از قبیل بزرگ نمایی، چرخش و .. را دارد.

Direct3D : نیز یک API سطح پایین است که شامل تمامی گرافیک های رندر شده در WPF می باشد.

WPF یک API با سطح بالاتر

اگر شتاب سخت افزاری با استفاده از DirectX را به عنوان تنها پیشنهاد WPF در نظر بگیریم، آنگاه این تکنولوژی چیزی جز یک بهبود در عمل کامپایل نخواهد بود و نمی توان آن را به عنوان یک حرکت انقلابی در زمینه کارهای ترسیماتی به حساب آورد. WPF برای ایجاد یک حرکت دگرگون ساز سرویس های سطح بالایی را برای برنامه نویسان فراهم آورده است. این سرویس ها عبارتند از:

یک مدل آرایشی تارمانند: WPF به جای ارائه کنترل هایی ثابت و با گوشه های مشخص، بر روی یک صفحه آرایی انعطاف پذیر که در آن می توان کنترل ها را بر اساس محتوایشان تنظیم کرد، تاکید فراوان دارد. نتیجه این تاکید ایجاد یک واسط کاربری که می توان آن را برای نمایش محتویات پویا سازگار کرد می باشد.

یک مدل ترسیماتی غنی: در WPF به جای رسم کردن پیکسل ها با چند مورد از اشکال پایه ای و ابتدایی، بلوک های متنی و عناصر گرافیکی دیگر سروکار خواهیم داشت. البته در میان مزایای فوق نباید پشتیبانی ذاتی از ترسیمات D3 را از یاد برد.

نکته: پشتیبانی از ترسمیات از ترسمیات سه بعدی در WPF قدرت برابری با OpenGL و یا Direct3D را ندارد. اگر قصد دارید که برنامه ای که ترسیمات سه بعدی سنگینی را در بردارد پیاده سازی کنید (مانند بازی های امروزی) WPF 1.0 احتمالاً جوابگوی کار شما نخواهد بود.

یک مدل متنی غنی: سال ها پس از نشان دادن متن ها در کنترل های ضعیفی مانند Labelها، WPF برای برنامه های ویندوز قابلیت نشان دادن متن ها با سلیقه های مختلف در هر نقطه از واسط کاربری را به ارمغان آورده است.

پویانمایی به عنوان بهترین مفهوم برنامه نویسی: تاکنون می تواستید با استفاده از زمان سنج فرم برنامه را مجبور

به دوباره سازی خودش کنید. اما در WPF، پویانمایی یک جز طبیعی و ذاتی در Framework می باشد. برای این منظور ابتدا انیمیشن ها را با استفاده از تگ های اعلانی تعیین می کنیم و سپس WPF آنها را در دستور کار قرار می دهد.

پشتیبانی از رسانه های صوتی و تصویری: ابزاری های ایجاد واسط کاربری که قبلاً مورد استفاده قرار می گرفتند مانند فرم های ویندوزی برای کار با رسانه های صوتی و تصویری به صورت چشمگیری محدود بدوند. اما WPF این مشکل را برای برنامه نویسان رفع کرده است.

سبک ها و قالب ها: با استفاده از قالب ها و سبک ها می توانید نحوه رندر شدن عناصر برنامه و ... را به صورت دلخواه تغییر دهید.

دستورات: برای بسیاری از کاربران انجام یک کار چه از طریق منوها با نوار ابزار فرقی ندارد؛ زیرا نتیجه هر دو یکسان است. به همین منظور با استفاده از WPF می توان دستورات را در جایی قرار داد و به آنها اشاره کرد.

واسط کاربری اعلانی: هر چند برنامه نویسان می توانند پنجره های WPF را با استفاده کدنویسی ایجاد کنند، اما VS.NET از روش دیگری استفاده می کند. VS.NET هر عنصر در پنجره را به صورت تگ های XML در XAML ایجاد می کند. مزیت این کار جداسازی کدهای اصلی برنامه از کدهای مربوط به واسط کاربری می باشد. همچنین update کردن واسط با استفاده از ابزار هایی که برای این منظور در نظر گرفته شده است بسیار راحت تر خواهد بود.

برنامه های براساس صفحه: با استفاده از WPF، می توانید برنامه هایی مانند مرورگر ها ایجاد کنید. این برنامه ها به شما اجازه می دهند در مجموعه صفحات حرکت کنید، اطلاعات را کامل کنید و با استفاده از دکمه ها forward/backward به صفحه های بعد و قبل حرکت کنید