بخش اول

آغاز کار با *WPF*

در این فصل موارد زیر را فرا می‌گیرید که چگونه:

- ✓ **ساعت ۱: *WPF* چه هست و چه نیست؟**
- ✓ **ساعت ۲: آشنایی با *XAML***
- ✓ **ساعت ۳: مقدمه‌ای بر نمایشگر فونت**
- ✓ **ساعت ۴: مدیریت طرح‌بندی برنامه**
- ✓ **ساعت ۵: استفاده از کنترل‌های پایه**
- ✓ **ساعت ۶: مقدمه‌ای بر انقیاد داده‌ها**

ساعت ۱

WPF چه هست و چه نیست؟

در این ساعت می‌آموزید:

- *WPF* چیست؟
- چه زمانی باید از *WPF* استفاده کرد؟
- به چه ابزارهایی نیاز دارید؟
- چگونه *WPF* با *Framework*های دیگر مقایسه می‌شود؟
- چه نسخه‌هایی از *NET*. قابل استفاده هستند؟
- *Silverlight*

WPF چیست؟

WPF بسیار بزرگ است! در واقع *WPF* اجزای مؤثر بسیاری دارد که به یکدیگر مرتبطند و همین امر در نگاه اول آن را پیچیده و سخت نشان می‌دهد. پاسخی کوتاه برای این سؤال این است که *WPF* نوعی *API* در *NET Framework*. است که برای ساخت واسط کاربر گرافیکی برنامه‌های تحت ویندوز و وب به کار می‌رود.

اما پاسخ کامل‌تر!

WPF برگرفته از عبارت *Windows Presentation Foundation* و به معنی ساختار نمایش ویندوز است. به طور فیزیکی نیز *WPF* مجموعه‌ای از اسمبلی‌های *NET*. و ابزارهای پشتیبانی شده آن می‌باشد. در واقع هدف *WPF* فراهم‌کردن *API*های یکپارچه برای تولید واسط کاربر حرفه‌ای و جذاب در ویندوز *XP* و *Vista* است.

WPF اجزای مناسبی از طراحی وب نظیر *Style Sheet* و زبان نشانه‌گذاری برای واسط کاربر توصیفی را با اجزای مناسبی از برنامه‌های حرفه‌ای اینترنتی نظیر انیمیشن، بردارهای گرافیکی مقیاس‌پذیر و پشتیبانی چندرسانه ترکیب کرده است. همچنین این اجزا با اجزای مناسب برنامه‌سازی ویندوز نظیر تجمیع قدرتمند با سیستم عامل و انتقیاد داده‌ها ترکیب شده‌اند. در *WPF* این مفاهیم بسیار

www.SourceSara.com سایت سورس سرا - آموزش برنامه نویسی

مستحکم و واحد هستند. علاوه بر این موارد *WPF* جذابیت‌های دیگری نظیر پشتیبانی از طراحی سه‌بعدی، چاپ پیشرفته و اسناد قابل حملی چون *PDF* را نیز دارا می‌باشد.

WPF نوعی *API* یکپارچه می‌باشد. ممکن است برخی از کارهایی که می‌توان در *WPF* انجام داد را از قبل انجام داده باشید. اما انجام تمام آن موارد در یک برنامه، کاری بسیار پیچیده و دشوار خواهد بود. البته ادعا نمی‌کنیم که تنها با *WPF* تمام آن قابلیت‌ها را می‌توان در یکجا گردآوری کرد اما *WPF* به شما امکان می‌داهد تا تمام آن موارد را با استفاده از *API*های قدرتمند و یکپارچه انجام دهید.

WPF تنها بخشی از یک تصویر کلی است یعنی علاوه بر آن سه کتابخانه دیگر نیز همراه با نسخه *3.0* مجموعه *NET. Framework* ارائه شده‌اند. تمامی این چهار کتابخانه هدف مشترکی دارند و آن فراهم آوردن *API*های قدرتمند و واحد در حوزه کاری خودشان است. علاوه بر این ترکیب هر یک از این چهار کتابخانه در یک برنامه می‌تواند نتایج بسیار مؤثری همراه داشته باشد. در جدول ۱-۱ سه کتابخانه دیگر همرده با *WPF* ارائه شده در مجموعه*NET Framework 3.0*. معرفی شده‌اند.

جدول ۱-۱: کتابخانه‌های همرده *WPF* در مجموعه ۳,۰ *NET Framework*.

کتابخانه	توضیحات
WCF	*Windows Communication Foundation* که به اختصار *WCF* نامگذاری شده به موضوع پیام رسانی تمرکز دارد. *API* این کتابخانه به طور بسیار جالبی تمامی وظایف و کارهای مربوط به ارتباطات و شبکه را تسهیل و ساده کرده است. همچنین تمامی مطالب حوزه سرویس‌های تحت وب راه دور و *P2P* را نیز پوشش داده است.
WF	کتابخانه‌ای قدرتمند برای تولید برنامه‌های با قابلیت جریان کار می‌باشد. *WCF* از زبان نشانه‌گذاری برای تعریف جریان کار در برنامه استفاده می‌کند. بنابراین از مبدل شدن جریان کار به کدهای پیچیده و دشوار جلوگیری می‌کند. همچنین ایجاد وظایف جریان کار دلخواه را برای طراحان و توسعه‌دهندگان آسان کرده است.
CardSpace	این کتابخانه از سه کتابخانه دیگر از شهرت کمتری برخوردار است. *CardSpace* سیستم شناسایی و تعیین هویت متعارفی را فراهم آورده است که توسط برنامه‌های تحت ویندوز یا برنامه‌های تحت وب قابل استفاده می‌باشد.

پیش از *WPF* از فرم‌های ویندوز استفاده می‌شد. فرم‌های ویندوز نوعی *API* گرافیکی هستند که توسعه‌دهندگان در *NET 2.0*. و نسخه‌های قبل از آن استفاده می‌کنند. فرم‌های ویندوز طرح و قالب قابل مدیریتی را ارائه می‌کنند که به کمک آن می‌توان به توابع گرافیکی *API* ویندوز دسترسی پیدا کرد. اما *WPF* اساساً با فرم‌های ویندوز تفاوت دارد زیرا بر پایه *DirectX* طراحی شده است. در اصل *DirectX API* بر ساختارهای برنامه‌نویسی چندرسانه‌ای و بازی‌سازی متمرکز شده است. بدین ترتیب با استفاده از *WPF* می‌توان فنون و ترفندهای جذابی را اجرا کرد که پیاده‌سازی آنها با استفاده از فرم‌های ویندوز تقریباً غیرممکن است. علت اصلی این موضوع نیز در استفاده *WPF* از حداکثر توان و کارایی سخت‌افزار موجود می‌باشد.

www.SourceSara.com سایت سورس سرا - آموزش برنامه نویسی

WPF شباهت‌هایی نیز با فرم‌های ویندوز و همچنین فرم‌های ASP.NET دارد. مایکروسافت کتابخانه‌ای از کنترل‌های پایه نظیر دکمه و جعبه متن را در آن ارائه کرده است. همچنین در WPF با مفاهیم مشابهی نظیر انقیاد داده‌ها و کد همراه مواجه خواهید شد. تمامی این مفاهیم در WPF کارآمدتر و بهینه‌تر شده‌اند.

معرفی قابلیت‌های WPF

در این قسمت به طور کلی قابلیت‌های مهم WPF را معرفی می‌کنیم اما در ادامه و در ساعات و بخش‌های بعدی این قابلیت‌ها را بیشتر بررسی خواهیم کرد.

- واسط کاربر توصیفی

WPF این امکان را فراهم کرده است تا واسط کاربر برنامه خود را با استفاده از زبان نشانه‌گذاری جدید XAML (زمل) ایجاد کنید. در ساعت ۲ "آشنایی با XAML" بیشتر با XAML آشنا خواهید شد. البته اگر از قبل با HTML کار کرده باشید با مفاهیم XAML نیز آشنا خواهید بود. XAML زبان نشانه‌گذاری غنی‌تر و کاراتر نسبت به HTML است و ابهام کمتری نسبت به آن دارد. Visual Studio همانند برخی از محصولات خانواده Expression نظیر Expression Blend امکان تولید محتوای XAML را دارا می‌باشد. همچنین XAML واسطی را برای تراکنش با طراح فراهم می‌کند.

- طرح‌بندی هوشمند

چیدمان و آرایش کامپوننت‌های مختلف در صفحه ممکن کمی پیچیده باشد. همچنین از آنجا که ترکیب نمایشی آنها بسیار زیاد و متنوع است این پیچیدگی در چیدمان افزایش می‌یابد. برای حل این مشکل WPF سیستم طرح‌بندی قابل توسعه‌ای را برای چیدمان بصری عناصر واسط کاربر ارائه کرده است. این سیستم به طور هوشمند و براساس طرح‌بندی تعریف شده کاربر قادر است تغییر اندازه داده و سازگاری یابد. این سیستم را در ساعت چهارم "مدیریت طرح‌بندی برنامه" در تشریح پانل‌ها بیشتر بررسی خواهیم کرد.

- گرافیک‌های مقیاس‌پذیر

در WPF گرافیک‌ها مبتنی بر بردار (و نه مبتنی بر محل تصویر Raster based) هستند. گرافیک‌های برداری ذاتاً مقیاس‌پذیرند و برخلاف گرافیک‌های مبتنی بر محل تصویر به فضای ذخیره‌سازی کمتری نیاز دارند. با این همه WPF پشتیبانی‌های متعددی را از گرافیک‌های مبتنی بر محل تصویر انجام داده است اما گرافیک‌های مبتنی بر بردار برای ساختار واسط کاربر مناسب‌ترند.

گرافیک‌های برداری در وب به خاطر محصولی همچون Adobe Flash و در ابعاد کوچکتر مشخصه‌های SVG (Scalable Vector Graphics) بسیار مشهور و پر طرفدارند.

حاصل نهایی WPF برای کاربران خود برنامه‌هایی است که قابلیت تغییر مقیاس دارند اما به هیچوجه از کیفیت نمایشی آنها کاسته نمی‌شود.

- الگوها و قالب‌ها

www.SourceSara.com سایت سورس سرا - آموزش برنامه نویسی

WPF ساخت عناصر واسط کاربر با قابلیت استفاده مجدد را ساده و آسان ساخته است. در *WPF* دو نوع الگو و قالب وجود دارد که عبارتند از الگوهای کنترلی و الگوهای داده‌ای. الگوهای کنترلی به شما امکان می‌دهند تا نحوه نمایش کنترل را مجدداً تعریف کنید. به عنوان مثال اگر در برنامه بخواهید تمامی کنترل‌های *Listbox* دارای پس‌زمینه آبی و حاشیه قرمز باشند می‌توانید از الگوی کنترلی برای تعریف مجدد ظاهر نمایشی کنترل‌های *Listbox* استفاده کنید. الگوهای کنترلی انجام این کار را برای طراحان بسیار آسان ساخته است زیرا قادرند تا بدون آنکه کوچکترین تأثیری بر فرایند توسعه برنامه داشته باشند ظاهر نمایشی (*look*) موردنظر را برای کنترل *Listbox* در الگوی کنترلی ایجاد کنند.

الگوهای داده‌ای نیز مشابه الگوهای کنترلی هستند با این تفاوت که به جای تعریف نحوه نمایش کنترل، نوع داده‌های لازم جهت نمایش را تعریف می‌کنند. فرض کنید که با برنامه‌ای کار می‌کنید که سروکار آن با افراد مختلف است نظیر برنامه مدیریت تماس‌ها. و در کد این برنامه هریک از افراد نمونه‌ای از کلاس *Person* باشند. در این حالت با استفاده از الگوهای داده‌ای می‌توانید نحوه نمایش هریک از نمونه‌های کلاس *Person* را که در واقع افراد ثبت شده در برنامه هستند در وسط کاربر برنامه تعیین کنید. به عنوان مثال هر نمونه از کلاس *Person* ممکن است در واسط کاربر برنامه به صورت یک کارت شغلی به عمراه تصویر کاربر، نام، نام‌خانوادگی و شماره تلفن وی نمایش داده شود. در صورت استفاده از چنین الگوی داده‌ای، هنگام انقیاد نمونه از کلاس *Person* به عنصری از واسط کاربر نظیر *Listbox*، *WPF* از الگوهای داده‌ای مشابهی استفاده خواهد کرد. در عمل خواهید دید که الگوهای داده‌ای هنگام کار با لیست‌ها یا سایر مجموعه‌های داده‌ای بسیار مناسب‌ترند.

- انقیاد

هنگامیکه از انقیاد در *WPF* سخن می‌گوییم ممکن است انقیاد داده‌های در ذهنتان مجسم شود. انقیاد داده‌ها در فرم‌های *NET*. بسیار پرکاربرد و سودمند بوده است. اگرچه *WPF* دارای قابلیت‌های انقیاد داده‌ای مهمی است اما امکان انقیاد توصیفی موارد دیگری نظیر فرامین، دکمه‌ها، انیمیشن و رویدادها را نیز فراهم کرده است. به عنوان مثال می‌توان به صورت توصیفی برای الصاق اطلاعات (*Pasting*)، کنترل *Button* را به یک فرمان جهت انجام این کار قید کرد تا با کلیک روی کنترل *Button* عملیات الصاق اطلاعات صورت گیرد.

- سبک‌سازی

اهمیت و ارزش *WPF* به جهت فراهم کردن امکان جذاب‌ترکردن ظاهر و واسط کاربر برنامه است. مثلاً به شما این امکان را می‌دهد تا پس‌زمینه کنترل *TextBox* را به رنگ قرمز در آورید یا حاشیه کنترل *Button* را با نوار آبی رنگ ضخیمی پوشش دهید. سبک‌ها (*Styles*) در *WPF* مشابه کلاس‌های نمایشی *CSS* در *HTML* می‌باشند. با این وجود سبک‌ها در *WPF* بسیار حرفه‌ای‌تر بوده و ابهام و پیچیدگی کمتری دارند. آنها تمامی خصوصیات و مشخصه‌های قابل تصور مانند لایه‌گذاری (*Padding*)، حاشیه‌گذاری (*Margin*)، موقعیت مکانی، رنگ و نظایر آنها را شامل می‌شوند. اما می‌توان علاوه بر این از سبک‌ها برای تعریف مشخصه‌ها و خصوصیات غیربصری نیز استفاده کرد.

www.SourceSara.com سایت سورس سرا - آموزش برنامه نویسی

از دیگر نکات جالب توجه در مورد سبک‌ها امکان استفاده مجدد از آنهاست. با ترکیب سبک‌ها به همراه الگوها می‌توان کارهای جالب و جذابی انجام داد.

- تریگرها (*Triggers*)

در *WPF* هم الگوها و هم سبک‌ها از مفهوم تریگر پشتیبانی می‌کنند. به عنوان مثال با استفاده از تریگر می‌توان به *WPF* گفت که: "در زمان قرارگیری نشانگر ماوس روی دکمه، رنگ پس‌زمینه آن تغییر کند!". در واقع تریگرها این امکان را فراهم می‌کنند تا بتوان تغییر وضعیت عناصر را مدیریت کرد. همچنین به کمک آنها می‌توان کار با انیمیشن را آغاز کرد.

- انیمیشن

ساختار و *Framework* انیمیشن در *WPF* بسیار جالب‌تر و جذاب‌تر از آنچه که ممکن است تصور کنید می‌باشد. بسیاری از خصوصیات در *WPF* می‌توانند متحرک‌سازی شوند. همچنین از مواردی نظیر *Timeline* و فریم‌های کلیدی نیز پشتیبانی می‌شود. می‌توان انیمیشن‌ها را به راحتی با الگوها و سبک‌ها تلفیق کرد. به عنوان مثال می‌توان سبکی را برای کنترل *Button* تعریف کرد که در زمان قرارگیری نشانگر ماوس روی آن متحرک شود!

- سه‌بعدی‌سازی

نهایتا *WPF* امکان انجام مدل‌سازی و متحرک‌سازی سه‌بعدی پایه را فراهم کرده است. علت اینکه می‌گوییم سه‌بعدی‌سازی پایه آن است که *WPF* برای ایجاد برنامه‌های سه‌بعدی با قابلیت بالا طراحی نشده است. با این همه قابلیت‌های سه‌بعدی‌سازی در *WPF* گنجانده شده و می‌توان از آنها در واسط کاربر برنامه‌ها استفاده کرد. در این کتاب در مورد قابلیت‌های سه‌بعدی‌سازی *WPF* صحبت نخواهیم کرد اما به طور مقدماتی در بخش ضمیمه مطالبی در این مورد ارائه شده است.

چرا از *WPF* استفاده می‌کنیم؟

WPF مشابه سایر کتابخانه‌های هم خانواده خود در *NET 3.0*. خوش‌ساخت بوده و از *API*های مستحکم و قدرتمندی برخوردار است. در مجموع آنها با ترکیب مفاهیم برنامه‌نویسی بسیاری از پیچیدگی‌های توسعه برنامه را کاهش دادند. اما در هر صورت این نکته را نیز باید در نظر داشته باشید که لزوماً *WPF* برای هر پروژه‌ای گزینه مناسبی نیست. برخی از برنامه‌ها را می‌توان به راحتی به صورت فرم‌های ویندوزی ایجاد کرد. اما آشنایی با *WPF* و قابلیت‌های آن امکانات بیشتری را در تولید برنامه‌ها فراهم می‌آورد. یادگیری *WPF* برای توسعه‌دهندگان برنامه‌های ویندوزی ضروری است زیرا سرانجام *WPF* تا اندازه‌ای رشد خواهد کرد که به زور کامل جایگزین فرم‌های ویندوز شود.

در قسمت "معرفی قابلیت‌های *WPF*" با برخی از مزایا و قابلیت‌های کلیدی *WPF* آشنا شدید. موارد زیر سناریوهایی هستند که لزوم استفاده از *WPF* در آنها خودنمایی می‌کند:

- پروژه شما نیازمند همکاری با طراحان است. در این مورد استفاده از *XAML* و ابزارهای پشتیبانی شده توسط آن واقعاً سودمند است. پس از آشنایی توسعه‌دهندگان و طراحان تیم با ابزاهای پشتیبانی شده، آنها قادر خواهند بود تا به تجربیات بسیار کارآمد و مؤثری دست یابند.

www.SourceSara.com سایت سورس سرا - آموزش برنامه نویسی

- برنامه شما چند رسانه‌ای است. اگر نیازمند تجمیع صدا و تصویر در محصول خود هستید قطعاً باید از *WPF* استفاده کنید.
- سخت‌افزار موردنیاز برای اجرای برنامه شما از *DirectX 9* یا نسخه‌های بعدی آن پشتیبانی می‌کند. *WPF* از قابلیت‌ها و امکانات *DirectX* استفاده کرده و براساس آن ساخته شده است. لذا برنامه‌های ساخته شده به کمک *WPF* از قابلیت‌های سخت‌افزاری به خوبی استفاده می‌کنند.
- برنامه شما نیاز به پشتیبانی از چاپ پیشرفته دارد. *WPF* از قابلیت‌ها و فنون مختلف چاپ پشتیبانی می‌کند که این مورد توسط فرم‌های ویندوز پشتیبانی نمی‌شوند.

پس از آنکه به موارد استفاده از *WPF* پی بردید با یادگیری آن خواهید توانست تا کارهای زیادی را در مدت زمان کمی انجام دهید. در بخش اول با ساخت برنامه‌ای ساده با اصول اولیه کار با *WPF* آشنا خواهید شد.

مقایسه *WPF* با سایر فناوری‌ها

اگر تنها با *.NET* کار کرده‌اید و توسعه‌دهنده آن هستید به جز *WPF* دو گزینه دیگر یعنی فرم‌های ویندوز و فرم‌های *ASP.NET* را در اختیار دارید. در این ساعت به طور کامل به مقایسه *WPF* و فرم‌های ویندوز خواهیم پرداخت. تنها مزیت فرم‌های ویندوز نسبت به *WPF* کتابخانه قدرتمند کنترل‌های آن و پشتیبانی مناسب از آن توسط ابزارها و محیط‌های دیگر است. زیرا از آنجا که *WPF* فنآوری نوینی است لذا هنوز بخش عمده‌ای از ابزارها و اجزاء قابل پشتیبانی توسط آن تولید نشده‌اند.

مقایسه *WPF* با *ASP.NET* کمی پیچیده است. سؤال مطرح در این مورد مبنی بر نحوه نصب و توزیع آنهاست. در حال حاضر *WPF* برای استفاده و به کارگیری در زیرساخت ویندوز دارای محدودیت‌هایی است که این محدودیت‌ها در مورد *ASP.NET* وجود ندارند. به عنوان مثال برنامه‌های *WPF* برای نصب و اجرا در ویندوز نیازمند*.NET Framework 3.0* یا نسخه‌های پیشرفته‌تر از آن هستند. از سوی دیگر اگر لازم باشد تا برنامه خود را به صورت متمرکز طراحی کنید یعنی برنامه نیازمند یک یا چند کنترل سمت سرور باشد به احتمال زیاد برای کاهش پیچیدگی این نوع برنامه‌ها از برنامه‌نویسی تحت وب استفاده خواهید کرد.

در خارج از دنیای *.NET* برخی از قابلیت‌های مشابه آن مخصوصاً رسانه و انیمیشن در محصولاتی همچون *Adobe Flash* وجود دارند. در گذشته *Flash* محصولی بود که در زمینه وب بسیار پرکاربرد بود. اما زیر ساخت *Adobe AIR* محصول *Flash* را برای توسعه در زیر ساخت‌های مختلف و همچنین برنامه‌های کاربردی مورد استفاده قرار داده است. با آنکه محیط توسعه‌کاری *Flash* به قدرت و کارایی محیط توسعه *.NET* نیست اما برای بسیاری از طراحان محیط آشنایی است. اما از سوی دیگر کتابخانه کنترل‌های *Flash* بسیار محدود و استفاده از کنترل‌های آن نیز دشوار و پیچیده است. با این همه ممکن است *Adobe AIR* رقیبی جدی برای *WPF* باشد.

www.SourceSara.com سایت سورس سرا - آموزش برنامه نویسی

اجزاء .NET Framework

متأسفانه در حال حاضر واژه‌ها و نسخه‌های متعددی از .NET Framework وجود دارد که بررسی جزءبه‌جزء آنها کاری دشوار است. بهتر است کمی تأمل کرده و به بررسی اجزاء .NET Framework و همچنین نحوه ارتباط آنها با نسخه‌های مختلف آن بپردازیم.

.NET Framework را می‌توان همچون خانواده‌ای از محصولات در نظر گرفت که شامل کامپایلر، کتابخانه‌های کد و زمان اجرا می‌باشند.

منظور از زمان اجرا Common Language Runtime یا CLR می‌باشد. CLR ماشین مجازیست که برنامه‌های .NET را میزبانی می‌کند. همچنین بسیاری از سرویس‌های اصلی همچون مدیریت حافظه، پاک‌کردن خانه‌های بلا استفاده حافظه (Garbage Collection) و امنیت را فراهم می‌آورد. بررسی CLR در حیطه مباحث این کتاب نیست اما همین‌قدر بدانید که CLR زمان اجرای .NET بوده و نسخه‌های آن نیز برحسب نسخه‌های .NET Framework متفاوت است.

دو زبان برجسته و پرکاربر دنیای .NET زبان‌های برنامه‌سازی C# و Visual Basic می‌باشند. این دو زبان دارای نسخه‌های مختلفی هستند که شماره این نسخه‌ها با شماره نسخه‌های .NET Framework یکسان نیست.

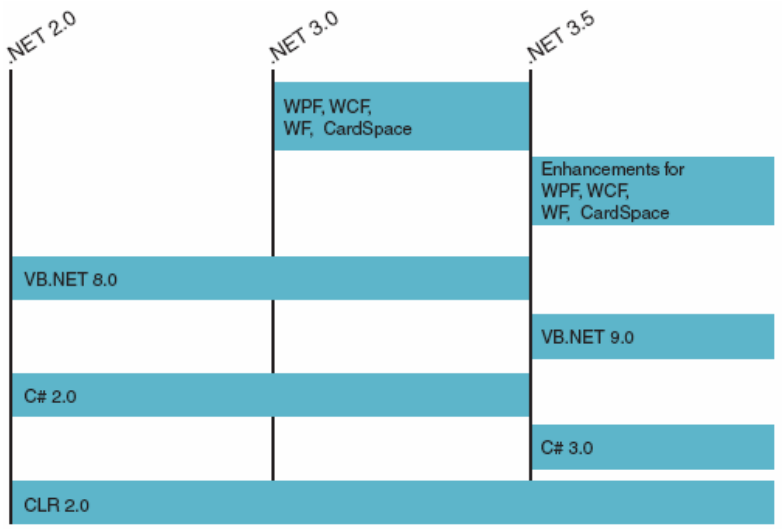
همچنین ممکن است درباره Basic Class Library (BCL) و Framework Class Library (FCL) مطالبی شنیده باشید. BCL مجموعه‌ای از کلاس‌های موجود در هریک از زبان‌های برنامه‌سازی خانواده .NET است. FCL نیز واژه‌ای است که شامل هم BCL و هم کتابخانه‌های رایج در فضای نام Microsoft می‌باشد.

تشخیص و تمایز BCL و FCL نیازمند بررسی و دقت موشکافانه در نتایج آنهاست و بسیاری از افراد این دو را به جای یکدیگر به کار ببرند.

در شکل ۱-۱ نحوه تغییر BCL و FCL در نسخه‌های مختلف .NET Framework (از نسخه 2.0 به بعد) نشان داده شده است. مطابق این شکل نکات قابل توجهی وجود دارد که به معرفی و بررسی آنها می‌پردازیم:

- CLR تا نسخه .NET Framework 2.0 تغییری نداشته است. از این رو ویژگی‌های اصلی .NET Framework نیز تغییری نداشته‌اند.
- C# 3.0 و VB.NET 9.0 به کدبایت یا IL کامپایل می‌شوند که نوعی کامپایل لحظه‌ای در CLR 2.0 می‌باشد. قابلیت‌های جدید زبان در .NET 3.5 در کامپایلرهای مربوطه بروز شده‌اند.
- WPF کتابخانه کلاس (Class Library) است و چیزی در CLR زیرین آن تغییر نکرده است. این بدان معنی است که برخلاف .NET 2.0 نسخه .NET 3.0 تنها حاوی کتابخانه‌های جدید اضافه شده به آن می‌باشد.

www.SourceSara.com
سایت سورس سرا - آموزش برنامه نویسی



شکل ۱-۱: نسخه‌های مختلف .NET Framework.

ابزارهای کار با *WPF*

در این کتاب مقدمتاً با *Visual Studio 2008* کار خواهیم کرد. *Visual Studio 2008* محیط اولیه و بومی برای *WPF* محسوب می‌شود.

لازم به ذکر است که با *Visual Studio 2005* نیز می‌توان برنامه‌های *WPF* را طراحی کرد که نیازمند به نصب *WPF Extensions* است اما قابلیت‌های نسخه نهایی آن را ندارد. البته قویاً توصیه می‌کنیم که از *Visual Studio 2008* استفاده کنید.

همچنین برای کار با *WPF* می‌توان از *SharpDeveloper* (*#developer* نیز نامیده می‌شود) استفاده کرد. *SharpDeveloper* محیطی کد باز برای کار با *.NET* است که از ساخت برنامه‌های *WPF* در *.NET 3.0* پشتیبانی می‌کند.

پس از *Visual Studio* ابزار اصلی برای ساخت برنامه‌های *WPF* محصولی از مجموعه *Expression* شرکت مایکروسافت یعنی *Expression Blend* است. *Expression Blend* بیشتر مناسب طراحان است تا توسعه‌دهندگان. از مزایای مهم *Expression Blend* آن است که با همان فایل‌ها، پروژه‌ها و راه کارهایی کار می‌کند که *Visual Studio* از آنها استفاده می‌کند. بدین ترتیب هم طراحان و هم توسعه‌دهندگان می‌توانند به طور همزمان در ساخت و تولید برنامه‌های *WPF* همکاری داشته باشند. محیط کار *Expression Blend* قابل مقایسه با محیط کار *Adobe Flash* است. یعنی مشابه *Adobe Flash* در آن ابزارهای ترسیم، انیمیشن‌سازی و نوار زمان (*Timelines*)، الگوها و قالب‌ها و سایر قابلیت‌های اصلی موردنیاز طراحان تعبیه شده است. بدین ترتیب توصیه می‌کنیم که به عنوان توسعه‌دهنده برنامه‌های