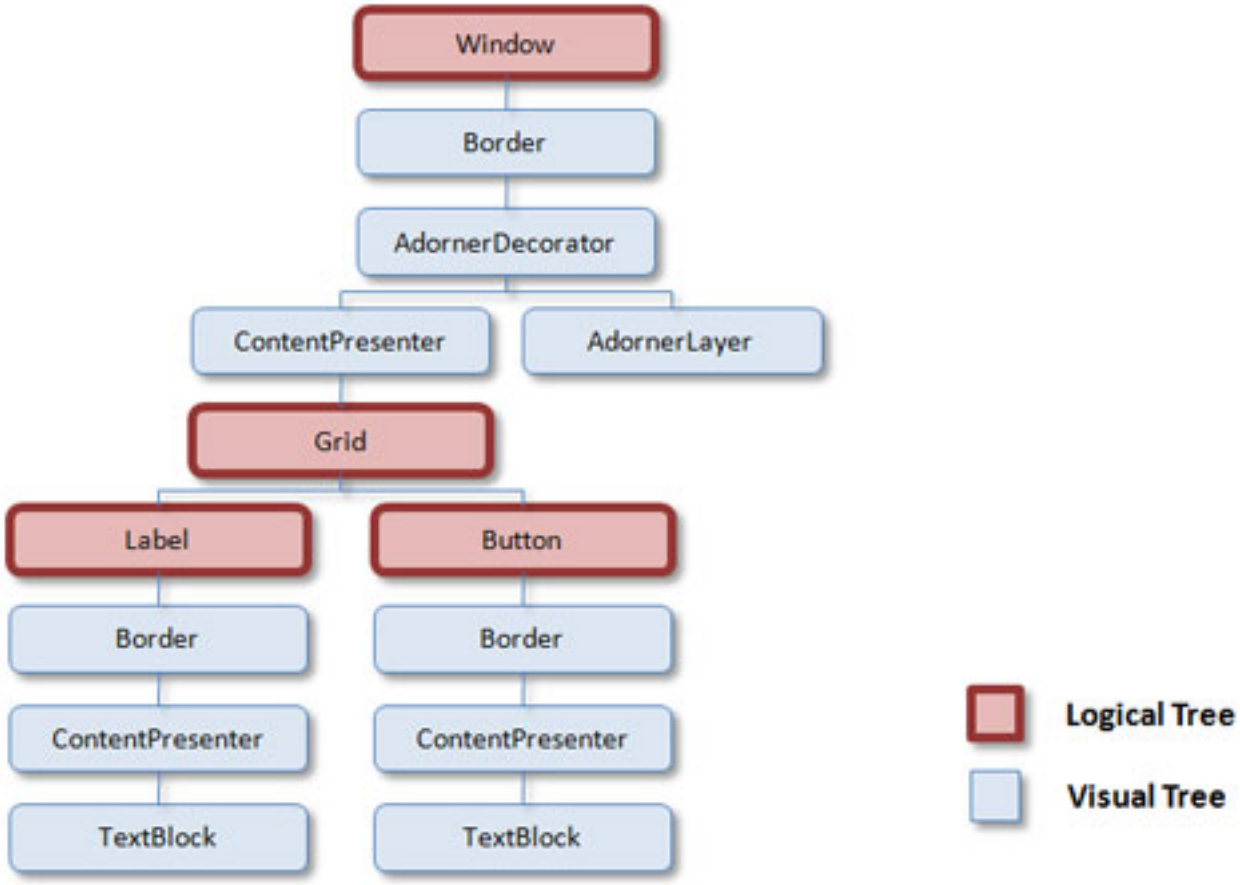
عنوان: **آشنایی با WPF قسمت اول : ساختار سلسله مراتبی**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۰۷ ۲:۰
آدرس: www.dotnettips.info
گروه‌ها: WPF



عناصر رابط کاربری WPF با یکدیگر یک رابطه‌ی سلسله مراتبی دارند. به این رابطه، درخت منطقی یا Logical Tree می‌گویند که به توصیف ارتباط اجزای رابط کاربری می‌پردازد. نوع دیگری از درخت نیز وجود دارد که به آن درخت بصری یا Visual Tree می‌گویند. این درخت شامل عناصری است که باعث نمایش کنترل پدر می‌شوند و کنترل پدر بدون آن‌ها هیچ ظاهر نمایشی ندارد. به عنوان مثال شما یک دکمه را در نظر بگیرید. این دکمه شامل عناصری چون Border,Block Text,Content Presenter می‌شود تا بتواند به عنوان یک دکمه نمایش یابد و بدون وجود این عناصر، کنترل دکمه هیچ ظاهری ندارد و در واقع با رندر شدن کنترل‌های فرزندان، دکمه معنا پیدا می‌کند. به تصویر بالا دقت کنید که به خوبی مرز بین درخت منطقی و درخت بصری را نمایش می‌دهد.
شکل سلسله مراتبی بالا از طریق کد زیر به دست آمده است:

```
<Window>
    <Grid>
        <Label Content="Label" />
        <Button Content="Button" />
    </Grid>
</Window>
```

درخت بصری می‌تواند به ما کمک کند تا بتوانیم بر روی عناصر تشکیل دهنده، یک کنترل قدرت عمل داشته باشیم و آن‌ها را مورد تغییر قرار دهیم.

Dependency Properties

خاصیت‌های وابستگی همان خاصیت‌ها یا property هایی هستند در ویندوزفرم با آن‌ها سر و کله می‌زدید ولی در اینجا تفاوت‌هایی با پراپرتی‌های قبلی وجود دارد که باعث ایجاد مزایای زیادی شده است.

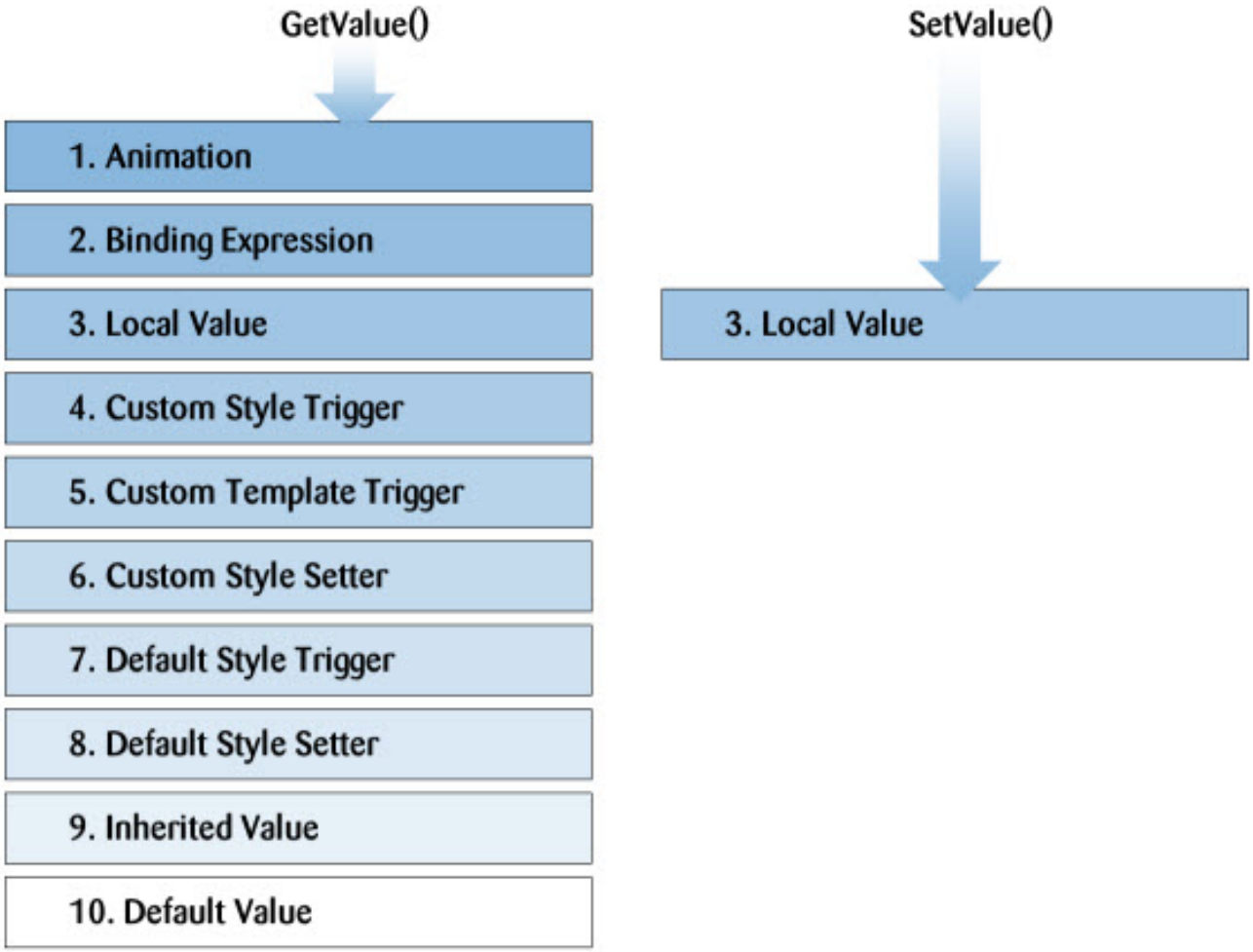
اول اینکه بر خلاف پراپرتی‌های ویندوز فرم که در خود فیلدهای تعیین شده همان کنترل ذخیره می‌شدند، در این روش کلید (نام پراپرتی) و مقدار آن داخل یک شیء دیکشنری قرار می‌گیرند که از شیء DependencyObject ارث بری شده است و این شیء والد یک متد با نام GetValue برای دریافت مقادیر دارد. مزیت این روش این است که بیخود و بی‌جهت مانند روش قبلی، ما فیلدهایی را تعریف نمی‌کنیم که شاید به نصف بیشتر آن‌ها، حتی نیازی نداریم. در این حالت تنها فیلدهایی از حافظه را دریافت و ذخیره می‌کنیم که واقعا به آن‌ها نیاز داریم. فیلدها یا مقادیر پیش فرض موقع ایجاد شیء در آن ذخیره می‌شوند.

دومین مزیت این روش خاصیت ارث بری مقادیر از عناصر بالاتر درخت منطقی است. موقعی که از طرف شما برای فرزندان این عنصر مقداری تعیین نشده باشد، سیستم به سمت گره‌ها یا عناصر بالا یا والد حرکت می‌کند و اولین عنصری را که مقدارش تنظیم شده باشد، برای فرزندان در نظر می‌گیرد. به این استراتژی یافتن یک مقدار، استراتژی Resolution می‌گویند.

سومین مزیت آن وجود یک سیستم اعلان یا گزارش آنی است. در صورتی که شما یک تابع callback را برای یک پراپرتی ست نمایید، با تغییر این پراپرتی تابع معرفی شده صدا زده خواهد شد.

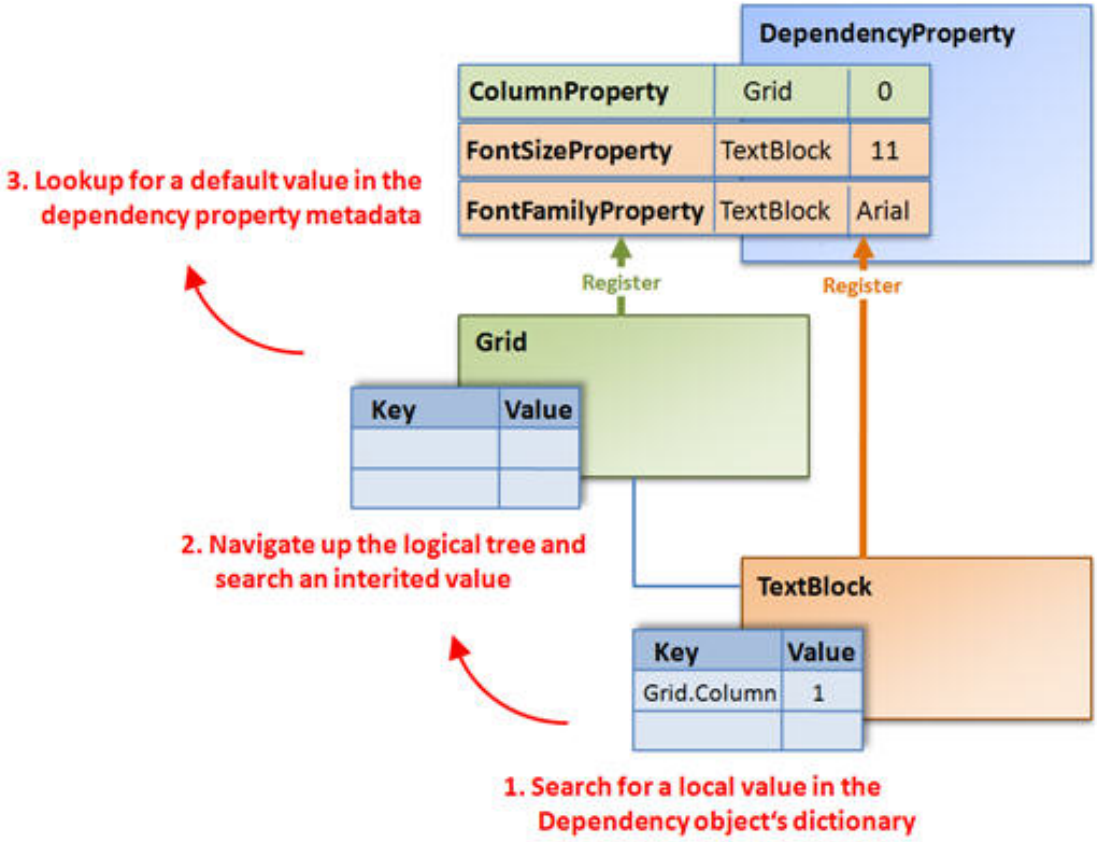
Value Resolution Strategy

همانطور که در بالا اشاره کردیم دریافت مقادیر یک کنترل از طریق یک استراتژی به اسم Resolution انجام می‌شود که طبق تصویر زیر از بالا به پایین بررسی می‌شود. در هر کدام از مراحل زیر اگر مقداری یافت شد، همان مقدار را انتخاب می‌کند. از متد SetValue هم برای درج مقدار استفاده می‌شود. برای مثال در مرحله‌ی سوم بررسی می‌شود که آیا کاربر برای کنترل مورد نظر مقداری را تنظیم کرده است یا خیر؛ اگر آری، پس از آن استفاده می‌کند و اگر پاسخ خیر بود، بررسی می‌کند آیا style برای آن موجود است که مقداری برایش تنظیم شده باشد یا خیر و الی آخر...



جادوی پشت صحنه

مقادیر پراپرتی‌ها در کلاسی استاتیک به اسم Dependency Property ذخیره می‌شوند که این ذخیره در حالت نام و مقدار است و مقدار آن شامل callback و مقدار پیش فرض است. شکل زیر نتیجه‌ی شکل دقیق‌تری را نسبت به قبلی در هنگام پیمایش درخت منطقی به سمت بالا، نشان می‌دهد.



نحوه‌ی تعریف یک خاصیت وابسته که باید به صورت ایستا تعریف شود به صورت زیر است و برای دریافت و درج مقدار جدید از یک پراپرتی معمولی کمک می‌گیریم:

```
// Dependency Property
public static readonly DependencyProperty CurrentTimeProperty =
     DependencyProperty.Register( "CurrentTime", typeof(DateTime),
     typeof(MyClockControl), new FrameworkPropertyMetadata(DateTime.Now));

// .NET Property wrapper
public DateTime CurrentTime
{
    get { return (DateTime)GetValue(CurrentTimeProperty); }
    set { SetValue(CurrentTimeProperty, value); }
}
```

یک قانون در WPF وجود دارد و آن اینست که نام خاصیت‌های وابسته را با کلمه Property به پایان ببرید مثل CurrentTimeProperty.

در مورد خاصیت‌های وابسته و کدنویسی آن ها در مطالب آینده بیشتر بحث خواهیم کرد.