第 19 章　WPF 开发基础

在 Vista 和 Windows Seven 火热发布的今天，很多用户都被 Vista 的特效所吸引，Vista 和的 Windows Seven 的 3D 特效，以及毛玻璃等效果给操作系统带来了更新更好的用户体验，在这一系列功劳的背后，WPF 占据着不小的功劳。

19.1　了解 WPF

WPF（Windows Presentation Foundation）原代号为“Avalon”，是微软的新一代图形系统。WPF 基于.NET 3.0 构架，为开发人员进行 Windows 应用程序开发和 2D/3D 图形和多媒体提供了统一的描述方法。对于开发人员而言，WPF 开发非常的简单，只要开发人员有一定的.NET 基础，都能够快速上手 WPF 应用程序开发。

19.1.1　什么是 WPF

WPF（Windows Presentation Foundation）是微软的新一代图形系统，为用户界面、2D/3D 图形、文档和媒体提供了统一的描述和操作方法。基于 DirectX 9 和 Direct 10 技术的 WPF 不仅带来了非常绚丽的 3D 界面，而且其图形向量渲染引擎也大大改进了传统的 2D 界面，使得传统的 2D 界面可以模拟毛玻璃、3D 等特效。

对于开发人员而言 WPF 提供了统一的 Windows Form 应用程序开发方法，并且开发人员通过使用 WPF 技术，能够使得 Windows Form 应用程序像动画一样展现在用户面前，用户能够得到良好的用户体验。WPF 包含两个部分，这两个部分分别为引擎和编程框架。

1．WPF 引擎

WPF 引擎为开发人员和设计人员提供了统一的设计文档，开发人员能够像普通的 Windows Form 应用程序一样进行逻辑编程，设计人员能够通过使用 XAML 语言描述 Windows Form 应用程序中各个控件的风格，以实现动画效果。

WPF 引擎还为设计人员提供了基于浏览器的体验、基于窗体的应用程序、图形、视频、音频和文档提供了一个单一的运行时库，WPF 让传统的 Windows Form 应用程序能够利用起现有的硬件软件资源，充分的利用 Direct 功能和硬件的编码解码功能进行窗体和控件的渲染。

2．WPF 框架

WPF 框架为媒体、用户界面设计和文档提供的解决方案比开发人员现有的解决方案都要好，WPF 框架在设计时考虑了可扩展性和可维护性，开发人员能够在 WPF 中创建自己的控件，还可以通过对现有的 WPF 控件进行改造创建新的 WPF 控件。

WPF 框架是用于形状、图像、视频、动画、文档、三维，以及用于放置控件和内容的面板的一系列控件，这些控件和内容的面板的一系列控件是 WPF 框架的核心。WPF 应用程序提供了若干 WPF 应用程序开发所需要的控件，开发人员同样能够对控件进行拖放操作实现应用程序布局和开发。

3．XAML 基本概念

WPF 应用程序引入了 XAML，XAML 是基于 XML 文档格式的一种标记语言，XAML 能够描述 Windows 应用程序和用户界面。开发人员和设计人员能够使用 XAML 语言进行代码和界面布局的可重用性控制。而对于 Web 开发者而言，XAML 是基于标记语言的，XAML 同样包括属性描述，对于 Web 开发者，也能够轻松的使用 XAML 描述 WPF 应用程序。

WPF（Windows Presentation Foundation）为开发人员和设计人员提供了统一的图形、图像、界面、文档等设计和开发的统一的运行和操作方法，WPF 使现有的 Window 应用程序能够充分的利用硬件软件的资源进行应用程序窗口渲染和优化，给用户以全新的 Windows 窗体应用程序体验。

19.2　WPF 的应用范围

在现有的 Window 应用程序中，对于已经成熟的传统的 WinForm 应用程序而言，为何还要抛弃现有的成熟技术而使用 WPF 技术开发 Window 应用程序呢？在传统 Window 应用程序开发中，应用程序的表现形式往往是非常死板的，应用程序窗体很难实现像 Web 应用和 Flash 中的渲染效果，例如图形图像的渲染和文本的渲染。虽然现今对渲染的方法有很多其他的解决方案，包括遨游等浏览器的 JavaScript 渲染，但是这些都是将 Window 应用程序和 Web 应用程序整合的解决方案，并没有完全的解决 Window 应用程序中对窗体本身的渲染的困难问题。

在 Vista 应用程序开发中，Vista 将应用程序窗体进行了效果的渲染，并没有使用 Web 应用的解决方案，直接通过 WPF 进行窗体和控件的渲染，实现了半透明等效果，让用户耳目一新，提高了用户体验。如图 19-1 所示。

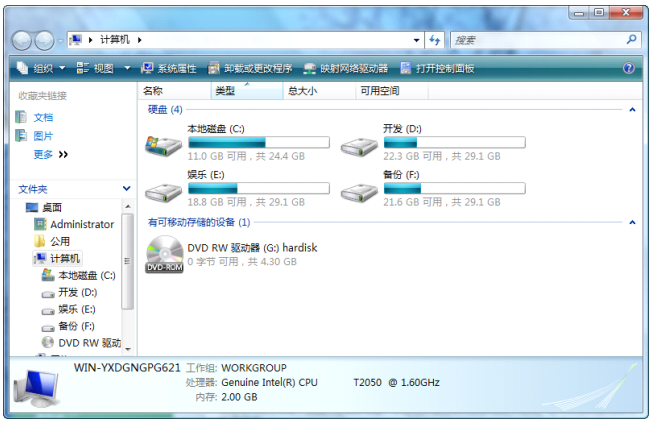


图 19-1　Windows 窗体图形渲染

随着互联网和硬件的发展，显卡等硬件已经能够辅助 CPU 的运算实现动态解码，让 CPU 的使用率变得更低，让 CPU 专注处理内核运算，从而能够让网络游戏等大型的需要图形操作和运算的应用程序能够使用显卡的解码技术流畅运行。

使用 WPF 也能够使用显卡的硬件进行应用程序渲染加速，这也能够让 WPF 应用程序不会占用过多的 CPU 资源，WPF 应用程序能够基于 Direct9/10 进行图形图像编程，而使用显卡加速能够充分的利用 Direct9/10 的资源提升应用程序的用户体验。

19.2　WPF 和 Microsoft Expression

在进行 WPF 应用程序的开发中，需要编写相应的 XAML 文档进行窗体的布局和渲染，在 Visual Studio 2008 中，并没有提供很好的支持 WPF 应用程序设计所需要的功能，例如动画操作和图形渲染。微软提供了 Microsoft Expression 软件套装，在 Microsoft Expression 软件套装中可以使用 Microsoft Expression Blend 2 进行 WPF 应用程序窗体的布局和渲染。

19.2.1　使用 Microsoft Expression Blend 设计 WPF

Microsoft Expression Studio 2 软件套装中，微软提供了 Microsoft Expression Blend 2，用于提供 WPF 应用程序和 Silverlight 应用程序的图形开发和渲染，双击 Microsoft Expression Blend 2 图标打开 Microsoft Expression Blend 2 应用程序，如图 19-2 所示。

Microsoft Expression Blend 2 和 Microsoft Expression Studio 2 软件套装一样，其的界面也是以黑色为主的界面。对于设计人员而言，设计人员更加偏好黑色界面以便将图形图像突出的显式在屏幕中。单击【新建】按钮，Microsoft Expression Blend 2 会弹出一个新建框，开发人员能够选择相应的应用程序进行开发，如图 19-3 所示。

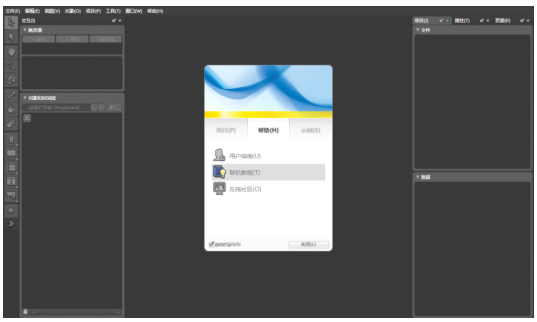


图 19-2　Microsoft Expression Blend 2

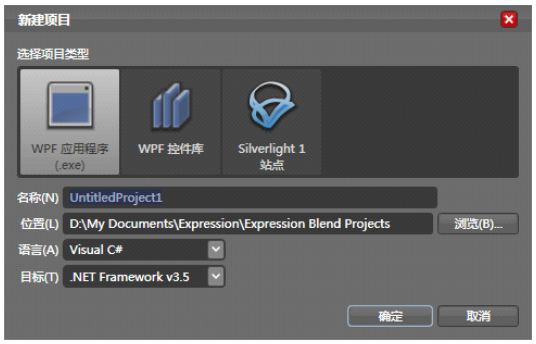


图 19-3　创建新建项

单击【WPF 应用程序】按钮，并选择相应的位置就能够创建 WPF 应用程序。WPF 应用程序创建后，对于开发人员和设计人员而言，其 Windows 应用程序开发窗口很像一张画布。在 WPF 应用程序中，窗体可以想象成是一个画布，这个画布能够承载多媒体、图形、图像甚至是动画。WPF 应用程序提供了默认控件，这些控件包括最常用的 Button 按钮控件，下拉框控件以便开发人员提供用户交互功能，如图 19-4 所示。

从图 19-4 中可以看出，WPF 应用程序提供了 Border、ListBox、Button、Label 等常用控件，在 Microsoft Expression Blend 2 中，同样可以直接拖放到窗体中进行窗体布局，如图 19-5 所示。

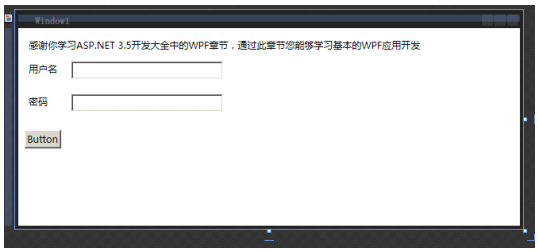
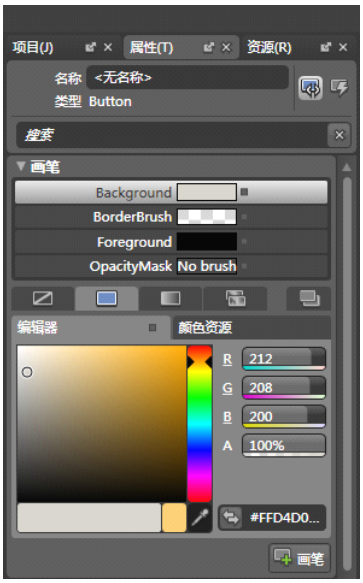


图 19-4　WPF 资源库　　图 19-5　WPF 应用程序窗体布局

Microsoft Expression Blend 2 同 Visual Studio 2008 相似，Microsoft Expression Blend 2 允许开发人员直接向窗体中拖动控件以实现控件的布局，但是 Microsoft Expression Blend 2 并不支持控件事件的响应。当开发人员在 Microsoft Expression Blend 2 中双击 Button 按钮控件时，并不会在相应的代码中自动创建方法，Microsoft Expression Blend 2 仅仅为 WPF 应用程序布局提供了良好的支持，若需要为 WPF 应用程序编写事件，需要同 Visual Studio 2008 相配合。

19.2.2　WPF 控件样式

使用 Microsoft Expression Blend 2 进行 WPF 应用程序开发很类似与 Photoshop 中进行图形图像编程。在 Photoshop 中进行图形图像编程时，可以针对某一个图形进行渲染，包括半透明、颜色和渐变等。在传统的 Windows 应用程序的开发中，如果需要让应用程序的背景或者某个按钮控件像动画一样呈现出渐变和半透明效果是非常困难的，在 WPF 中可以进行类似 Photoshop 的操作对 WPF 应用程序中的控件进行样式控制，如图 19-6 和图 19-7 所示。



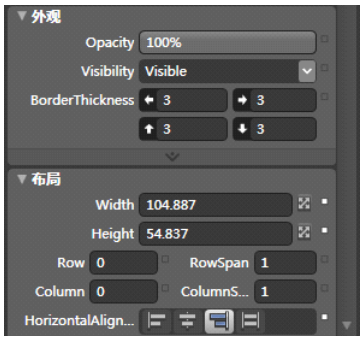


图 19-6　属性控制面板　　图 19-7　外观控制

图 19-6 和图 19-7 都是针对一个控件进行样式控制。使用 Microsoft Expression Blend 2 进行控件样式开发的过程中，每一个控件都包含一个属性面板，属性面板用于 WPF 应用程序中控件的样式的控制。在 WPF 中，窗体都是基于 XAML 文档进行编写和样式控制的，如果需要使用 XAML 文档进行样式开发和控制，不得不记住很多属性，这样就让 WPF 应用程序的开发变得非常困难。在 Microsoft Expression Blend 2 中使用属性控制面板能够快捷的定义相应控件的属性。

使用 Microsoft Expression Blend 2 进行应用程序中进行样式控制非常容易。在 Photoshop 中对图形图像的编程可以直接使用画笔或渐变等工具进行样式控制，同样在 Microsoft Expression Blend 中可以像在 Photoshop 中一样进行属性配置就可以实现控件的不同样式的布局，如图 19-8 所示。

图 19-8　Button 控件样式控制

在图 19-8 中，可以通过样式选择选择不同的控件样式，从左到右分别为渐变样式、平铺样式、纯色样式和半透明样式。使用 XAML 进行样式控制示例代码如下所示。

```
    </Button>
    <Button d:LayoutOverrides="VerticalAlignment" HorizontalAlignment="Right"
        Margin="0,0,519.113,65" VerticalAlignment="Bottom" Width="104.887"
        Height="54.837" Content="Button">
        <Button.Background>
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                <GradientStop Color="#FFC8C8C8" Offset="0"/>
                <GradientStop Color="#FFFFFFFF" Offset="1"/>
            </LinearGradientBrush>
        </Button.Background>
    </Button>
```

上述 XAML 代码就描述了一个渐变控件，其中为控件定义了基本的属性，包括宽度和高度。在子节点中，通过使用 LinearGradientBrush，GradientStop 等属性实现了控件的效果。这些样式在传统的 Windows 应用程序开发中要实现是非常繁琐和困难的，而在 WPF 应用程序中能够方便的实现渐变和图形控件。

在属性控制面板中还包括很多其他的属性配置，这些属性包括不透明度、宽度、高度、文本对齐方式、窗体状态等，极大的方便了开发人员在开发过程中对样式的控制和封装，简化了开发人员对于窗体界面的开发。

19.2.3 浅谈 XAML

使用 Visual Studio 2008 打开项目，就会发现 WPF 应用程序是通过使用 XAML 文档进行描述的。XAML 是 eXtensible Application Markup Language 的英文缩写，其中文名称为可扩展应用程序标记语言，它是微软公司为构建应用程序用户界面而创建的一种新的描述性语言。WPF 应用程序中大量的使用了 XAML 对应用程序窗体进行描述。

XAML 是一种基于 XML 文档格式的标记语言。在 WPF 应用程序中，XAML 用于描述窗体的样式、规则，以及事件的声明等，这种开发模型和 ASP.NET 中的代码隐藏页模型十分类似，这样就让 XAML 提供了一种便于扩展和定位的语法来定义和程序逻辑分离的用户界面的开发模型。

由于 XAML 是一种标记语言，所以开发人员能够很快的学习 XAML 并将 XAML 投入到项目开发中。学习 XAML 只需要开发人员具备一定的 HTML 知识就能够快速的学习 XAML 的语法格式和掌握 XAML 语法规范。

使用 XAML 进行窗体样式开发能够更加方便的协调开发人员和设计人员的工作，设计人员能够专注于窗体的样式控制使得开发人员能够专注于代码的编写，这样就有利于应用程序的开发。在 WPF 应用程序开发中，XAML 能够方便的对应用程序中的控件，样式进行描述，示例代码如下所示。

```
<Button HorizontalAlignment="Left" Margin="86,0,0,65" VerticalAlignment="Bottom"
Width="104.887" Height="54.837" Content="Button">
<Button.Background>
    <DrawingBrush Stretch="Fill" TileMode="None" Viewbox="0,0,20,20" ViewboxUnits="Absolute">
        <DrawingBrush.Drawing>
            <DrawingGroup>
                <GeometryDrawing Brush="#FFD3D3D3">
```

```
                    <GeometryDrawing.Geometry>
                        <RectangleGeometry Rect="0,0,20,20"/>
                    </GeometryDrawing.Geometry>
                </GeometryDrawing>
                <GeometryDrawing Brush="#FF000000">
                    <GeometryDrawing.Geometry>
                        <EllipseGeometry Center="0,0" RadiusX="10" RadiusY="10"/>
                    </GeometryDrawing.Geometry>
                </GeometryDrawing>
                <GeometryDrawing Brush="#FF000000">
                    <GeometryDrawing.Geometry>
                        <EllipseGeometry Center="20,20" RadiusX="10" RadiusY="10"/>
                    </GeometryDrawing.Geometry>
                </GeometryDrawing>
                <GeometryDrawing Brush="#FFFFFFFF">
                    <GeometryDrawing.Geometry>
                        <EllipseGeometry Center="20,0" RadiusX="10" RadiusY="10"/>
                    </GeometryDrawing.Geometry>
                </GeometryDrawing>
                <GeometryDrawing Brush="#FFFFFFFF">
                    <GeometryDrawing.Geometry>
                        <EllipseGeometry Center="0,20" RadiusX="10" RadiusY="10"/>
                    </GeometryDrawing.Geometry>
                </GeometryDrawing>
            </DrawingGroup>
        </DrawingBrush.Drawing>
    </DrawingBrush>
</Button.Background>
</Button>
```

上述代码通过使用 XAML 语法编写了使用平铺画笔进行渲染的按钮控件，其格式是按照 XML 文档的格式进行声明和编写的。创建一个富媒体的半透明控件，同样可以通过 XAML 进行控制，免除了 C++/Java 中复杂的编程实现，示例代码如下所示。

```
<Button Margin="0,114.419,134.797,51.255" Content="Button"
Background="#FFF7F7F7" HorizontalAlignment="Right" Width="148.887" Opacity="0.4"/>
```

上述代码就创建了一个半透明控件，该控件属性分别包括 Margin、Content、Background、HorizontalAlignment、Width 和 Opacity 等，其中 Opacity 属性用于控制控件的半透明度。使用 XAML 文档进行 WPF 应用程序的样式控制可以实现可扩展的样式控制，设计人员可以通过直接对 XAML 文档的修改实现不同的样式控制。WPF 应用程序中的任何控件都包括一些属性，这些属性能够方便的对控件进行样式控制，免除了复杂的编码实现。

19.2.4 WPF 控件层次

同 Photoshop 中的画布相同，WPF 应用程序中也有层次分别，每一个控件都包含一个自己的层次，如图 19-9 所示。

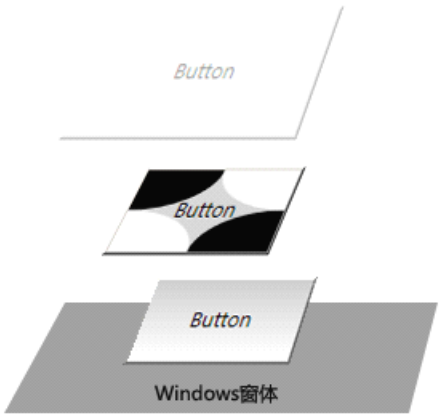
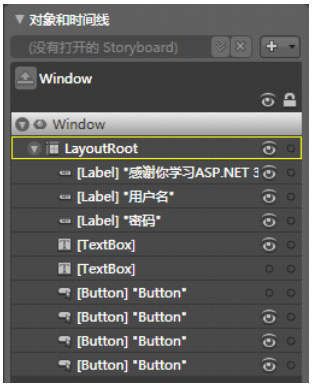


图 19-9　WPF 样式层次结构

无论是在 WPF 应用还是在 Win From 应用开发中，控件之间都有层次结构，如图 19-10 中所示，窗体的层次最低，用于承载控件。在窗体上的控件都是按照一个的层次堆叠在一起的，层次高的控件会遮住层次低的控件，在 WPF 布局中需要注意层次之间的控件的布局，如图 19-10 所示。

图 19-10　两个不同层次叠放的控件

在 Microsoft Expression Blend 2 中，包含层次管理器方便对层次进行管理。创建 WPF 应用程序后，系统会默认的创建一个层，该层作为基层而存在，而控件都是基于该层而存在的，如图 19-11 所示。



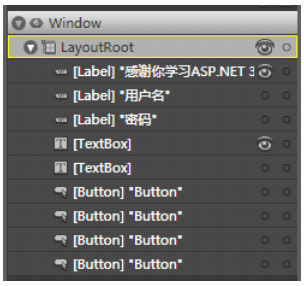


图 19-11　层次管理　　图 19-12　显式和隐藏层

在层次管理器中，可以设置相应的控件是否显式或隐藏，以便在大量的控件中准确的选取相应的控件，如图 19-12 所示。使用层次管理能够方便管理控件所在的层次并且为设计人员提供了控件选取的遍历，在层次管理中还可以选择锁定层或解除层，当一个层被锁定后就无法在窗体的设计中拖动相应的层，这样能够遍历的对不需要经常更改的层进行方便的管理。

19.3　WPF 应用程序开发

WPF 不仅提供了强大的布局功能和窗体渲染功能，在 WPF 应用程序开发中，还能够实现如 Flash 一样的动画效果，这就使得在 Windows 窗体中能够实现 Flash 动画效果，Microsoft Expression Blend 2 提供了动画轴，动画事件处理面板，方便了开发人员在 WPF 中实现动画效果。

19.3.1　WPF 动画事件

WPF 可以像 Flash 一样支持动画开发，与普通的事件不同的是，WPF 包括一个动画事件，这个动画事件描述的是当用户执行某个操作时所触发的动画事件。首先需要创建一个动画对象，这个对象可以是一个图片，也可以是一个控件，其 XAML 文档如下所示。

```
<Button HorizontalAlignment="Left" Margin="267,103,0,0" VerticalAlignment="Top"
Content="会变的按钮" Height="52.687" Width="86" Opacity="0.4"/>
```

上述代码创建了一个按钮控件，并为按钮控件配置了相应的属性，这些属性包括对齐方式，大小，文本，以及不透明度。如果开发人员希望当用户的鼠标单击该控件时，该控件的宽度和高度都会变化，并且不透明度也会变化，在 Microsoft Expression Blend 2 中的交互控制面板可以完成该事件的配置，如图 19-13 所示。

单击【+事件】按钮可以为 WPF 应用程序添加动画事件，Microsoft Expression Blend 2 能够智能的识别 WPF 应用中的控件并为相应的控件选择方法，为了实现开发人员所希望实现的效果。在下拉菜单中，这里可以选择【Click】事件如图 19-14 所示。

图 19-13　交互面板

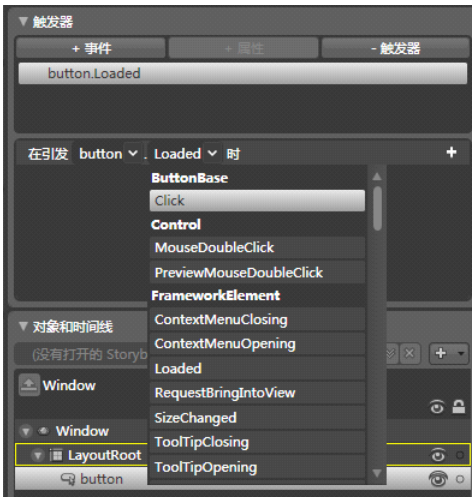


图 19-14　选择事件

选择事件后单击【时】按钮旁边的加号能够为动画事件添加新操作，如果 WPF 应用程序没有创建时间轴时，系统会提示是否添加一个时间轴，单击【确定】按钮即可创建一个默认的时间轴以供开发人员进行动画开发，如图 19-15 所示。

图 19-15　添加时间轴

注意：Story board 可以翻译成节目播出表，其概念同时间轴基本相同，都会规定对象的播放顺序和方法。

19.3.2　WPF 时间轴

在 Flash 动画的制作中，有一个时间轴的概念。时间轴是用来控制在动画运行中相应的时间时，某个或某些对象所需要执行的操作。例如一个 Flash 动画，在动画运行后，第 5 秒的时候有一个小人出现在动画中，并且拿出了一朵花。在这个过程中，就需要在第 5 秒的时候对相应的对象（这里包括人，花）进行相应的操作。

在 WPF 应用程序中同样包括一个时间轴，这个时间轴用于定制 WPF 动画事件中某一个时刻所需要实现的动画效果，开发人员能够通过使用 WPF 时间轴快速的实现动画效果。例如上一节中讲到的开发人员希望实现一个按钮的动画效果，则可以通过时间轴编写相应的事件，时间轴如图 19-16 所示。

在对象和时间轴面板中，可以选择相应的对象进行动画事件的操作。在时间轴面板中，可以看到在时间轴上方包括从 0 开始的数字，这些数字就是时间控制。当触发某个动画事件时，时间轴就开始运算，从 0 开始向右移动，分别执行相应的路径以实现动画效果。如果开发人员希望用户单击按钮时能够实现按钮形状和透明度的变化，可以通过时间轴方便的生成动画。在进行时间轴操作前，可以通过样式控制控件的相应位置，样式和内容等，如图 19-17 所示。

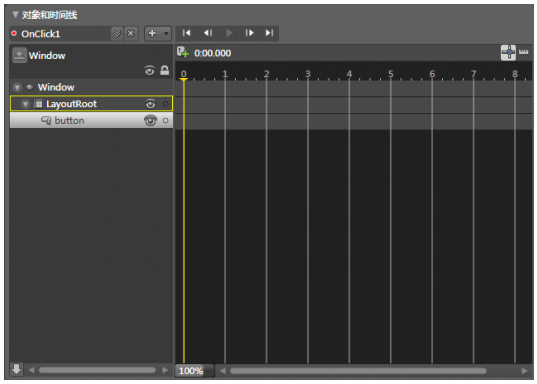


图 19-16　对象和时间轴

图 19-17　控件初始状态

初始状态确定后 XAML 文档代码如下所示。

```
<Button HorizontalAlignment="Left" Margin="267,103,0,0" VerticalAlignment="Top" Content="会变的按钮"
Height="52.687" Width="86" Opacity="0.4" x:Name="button" RenderTransformOrigin="0.5,0.5">
<Button.RenderTransform>
    <TransformGroup>
        <ScaleTransform ScaleX="1" ScaleY="1"/>
        <SkewTransform AngleX="0" AngleY="0"/>
        <RotateTransform Angle="0"/>
        <TranslateTransform X="0" Y="0"/>
    </TransformGroup>
</Button.RenderTransform>
</Button>
```

上述代码初始化了一个按钮控件并声明。在确定了初始状态后就需要拖动时间轴来确定动画播放的顺序，如图 19-18 所示。

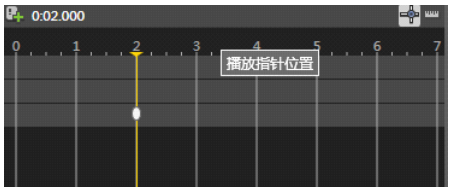


图 19-18　拖动时间轴

如图 19-18 所示，可以将时间轴拖放在 2 秒的位置。拖放后，可以直接在当前位置修改控件的属性 。修改后当触发动画事件后，时间轴开始移动并且会随着时间轴进行控件属性的更改。当时间轴的时间指针移动到 2 秒位置时，属性就会更改成 2 秒时设置的样式，如图 19-19 和图 19-20 所示。

图 19-19　初始状态

图 19-20　实现动画

通过使用时间轴能够快速定义 WPF 动画效果，开发人员能够使用时间轴进行相应的动画操作而无需通过编程实现，这样就简化了开发人员对底层动画实现的复杂的操作，节约了开发周期。另外，设计人员也能够设计动画事件并专注与 WPF 动画的实现，而开发人员能够专注逻辑处理，可以将动画事件的实现交付给设计人员，形成明确的分工。

19.3.3　WPF 事件处理

在 Microsoft Expression Blend 2 中只能控制 WPF 应用程序的样式，却无法进行事件处理开发，若需要进行 WPF 应用程序开发，就必须使用 Visual Studio 2008。使用 Visual Studio 2008 打开 Microsoft Expression Blend 2 创建的解决方案，能够进行 WPF 应用程序事件开发，如图 19-21 所示。

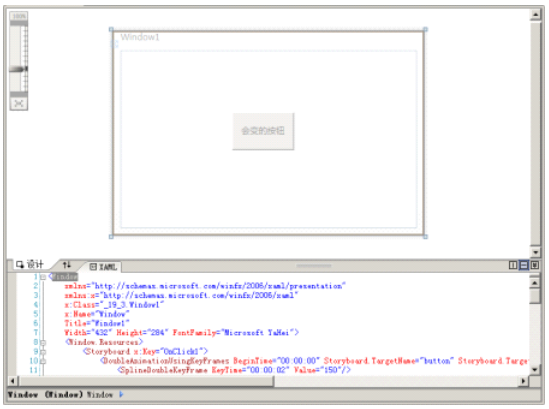


图 19-21　使用 Visual Studio 2008 打开解决方案

在 Visual Studio 2008 中进行 WPF 应用程序开发会呈现两个窗口，一个窗口用于直接进行 Windows 窗体布局，另一个窗口用于呈现相应的 XAML 文档。在 Visual Studio 2008 中，可以直接修改 XAML 文档进行 WPF 样式控制，示例代码如下所示。

```
<Grid x:Name="LayoutRoot">
<Button Margin="155,86,169,107" Content="更改后的按钮" Opacity="0.6"
      x:Name="button" RenderTransformOrigin="0.5,0.5">
      <Button.RenderTransform>
            <TransformGroup>
                  <ScaleTransform ScaleX="1" ScaleY="1"/>
                  <SkewTransform AngleX="0" AngleY="0"/>
                  <RotateTransform Angle="0"/>
                  <TranslateTransform X="0" Y="0"/>
            </TransformGroup>
      </Button.RenderTransform>
</Button>
</Grid>
```

上述代码直接修改 XAML 代码就可以实现 Windows 窗体样式的控制。与 Microsoft Expression Blend 2 不同的是，在 Visual Studio 2008 中双击按钮控件，在就会在 cs 文件中自动创建相应的事件代码，开发人员可以在相应的区域中编写代码，示例代码如下所示。

```
        private void button_Click(object sender, RoutedEventArgs e)
        {
            button.Content = "按钮控件被按";                                    //触发事件
        }
```

上述代码运行后如图 19-22 和图 19-23 所示。

图 19-22　按钮初始化

图 9-23　动画事件和按钮事件

注意：WPF 应用程序中的一些属性可能和 Win From 和 ASP.NET 中的一些属性不同，例如在 Win From 和 ASP.NET 中按钮控件上的文本是通过 Text 属性控制的，而在 WPF 中使用的是 Content 属性。

WPF 应用的开发和 Win Form 应用程序的开发没有特殊的区别，但是 WPF 应用提供了更好的用户体验。WPF 不仅能够提供动画事件同样也能够执行 Win From 应用程序开发中所需要的事件。

19.4　WPF 系统开发

WPF 能够开发用户体验更好的 Windows 应用程序，通过使用 WPF 技术，能够实现可扩展的容易维护并且用户体验友好的 Windows 应用程序。在微软本身的产品中，很多应用也使用了 WPF 技术，包括

Vista 以及 Expression。

19.4.1 WPF 系统需求

在 Windows 应用程序开发中，常常需要进行数据查询，例如一个图书管理系统，借读的读者往往很难在诸多图书当中寻找一个适合自己的书，例如如果读者希望借一本关于 ASP.NET 的书，但是图书馆中包含了很多关于 ASP.NET 的书，读者曾经看过了一本关于 ASP.NET 3.5 的书，希望能够找到该书，但是在图书馆中找了半天都找不到这本书，读者就会想“如果能够查询该书就好了”。

开发人员可以很快的进行图书管理系统的编码并进行查询分析，现在读者可以在图书馆电脑中查询 ASP.NET 3.5 开发大全了，但是查询出来的结果显示的并不那么友好，而且界面颜色单调，这就需要 Windows 应用程序具有较好的用户体验。WPF 应用程序就能够实现较好的用户体验，同样也可以实现普通 Windows 应用程序所能够完成的需求。

19.4.2 WPF 界面开发

为了实现较好的用户体验，首先需要进行良好的 WPF 界面开发和布局。WPF 支持 PNG，JPG 等图片资源作为 WPF 应用程序的背景，所以 WPF 应用程序能够实现半透明等多种渲染效果，WPF 系统登录界面和查询界面如图 9-24 和图 9-25 所示。

图 9-24　图书系统初步布局

图 9-25　用户查询界面布局

WPF 能够支持 PNG，JPG 等格式的图片文件，所以在 WPF 窗体开发中能够使用渐变效果填充窗体并可以直接使用 PNG 图片进行窗体渲染。登录窗体包含了一个图片文件，图片文件的 XAML 代码如下所示。

```
<Image Margin="0,0,2,23" Width="490" Height="450" Source="bg.png" Stretch="Fill"/>
```

注意：PNG 图片支持透明效果，而其他的图片格式的文件可能不支持半透明效果，WPF 支持半透明图片作为资源文件。

通过 XAML 文档能够定义图片文件并定义一些常用控件，为了实现以上的 WPF 界面布局，WPF

应用程序窗体的 XAML 代码如下所示。

```
<Window
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="_19_4.Window1"
x:Name="Window"
Title="图书管理系统"
Width="500"
Height="500"
FontFamily="Microsoft YaHei"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d">
……
</Window>
```

上述代码通过 XAML 实现了一个基本的 Windows 窗体，该窗体的头部信息为图书管理系统，并且定义了窗体的高度和宽度为 500，窗体内的字体样式为微软雅黑。在 WPF 窗体中，还需要定义 Label 控件和 TextBox 等控件，用于实现基本的人机交互，其 XAML 代码如下所示。

```
<Grid x:Name="LayoutRoot">
<Image Margin="0,0,2,23" Width="490" Height="450" Source="bg.png" Stretch="Fill"/>
<Label FontSize="16" FontWeight="Bold" Margin="132,37,116.11,0"
      VerticalAlignment="Top" Height="30.117" Content="读者您好,欢迎查阅我图书馆资料"/>
<Image HorizontalAlignment="Left" Margin="58,25,0,0" VerticalAlignment="Top"
Width="54" Height="54" Source="hello.png" Stretch="Fill"/>
<TabControl IsSynchronizedWithCurrentItem="True" Margin="12,95,21,23">
      <TabItem Header="登录">
            <Grid/>
      </TabItem>
      <TabItem Header="查询">
            <Grid/>
      </TabItem>
</TabControl>
<Label HorizontalAlignment="Left" Margin="110,227,0,0" VerticalAlignment="Top" Content="用户名:"/>
<Label HorizontalAlignment="Left" Margin="122,0,0,173" VerticalAlignment="Bottom" Content="密码:"/>
<TextBox d:LayoutOverrides="HorizontalAlignment" HorizontalAlignment="Left" Margin="175,0,0,213.163"
 VerticalAlignment="Bottom" Text="TextBox" TextWrapping="Wrap" Width="192.89"/>
<PasswordBox Margin="175,0,116.11,179.037" VerticalAlignment="Bottom"/>
<Button HorizontalAlignment="Left" Margin="198,0,0,100" VerticalAlignment="Bottom" Content="登录"
Height="44.837" Width="75.887"/>
</Grid>
```

上述代码实现了 WPF 窗体的基本布局，在 WPF 窗体中包含三个 Label 标签控件，用于显示相应的提示信息，如“用户名”，“密码”等。该窗体还包含了两个 TextBox 控件，其中一个 TextBox 控件用于用户用户名的输入，而另一个 TextBox 控件用于密码的输入。编辑完成登录窗体后，就需要进一步对搜索窗体进行样式控制，搜索窗体 XAML 文档代码如下所示。

```
<Window
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="_19_4.Window2"
x:Name="Window"
```

```
Title="Window2"
Width="500" Height="500" FontFamily="Microsoft YaHei">
……
</Window>
```

上述代码同样创建了一个宽度和高度都为 500 的窗体，在窗体中包括一个图片，一个搜索框和一个搜索结果框，这组控件 XAML 代码如下所示。

```
<Grid x:Name="LayoutRoot">
    <Image Margin="0,0,2,23" Source="bg.png" Stretch="Fill" Width="495" Height="450"/>
    <Image Margin="80,29,0,0" Source="ok.png" Stretch="Fill" Width="91"
        Height="91" VerticalAlignment="Top" HorizontalAlignment="Left"/>
    <Label HorizontalAlignment="Left" Margin="194,54,0,0" VerticalAlignment="Top"
        Content="输出相应书籍名称" FontWeight="Bold"/>
    <TextBox HorizontalAlignment="Left" Margin="194,83.837,0,0" VerticalAlignment="Top"
        Text="" TextWrapping="Wrap" Width="246.487"/>
    <TextBox Margin="44,137,31.513,36" Text="" TextWrapping="Wrap"/>
</Grid>
```

窗体基本布局完成后，就可以为窗体中的控件进行动画事件的编写，创建动画事件能够提高用户的体验并且使应用程序更加绚丽。

19.4.3 WPF 动画制作

在图书管理系统中，希望读者首先登录，如果登录成功了，就能够进行查询；如果登录没有成功，则不允许用户开始查询，只有用户登录成功后才有查询权限。在读者单击登录按钮时，应用程序可以播放一段动画以提示用户正在登录，如图 9-26 和图 9-27 所示。

图 9-26　登录框位置下移

图 9-27　登录框位置上移

当用户单击登录按钮进行登录时，登录框会上下移动以提示用户该应用程序正在处理。在动画处理代码中，必须为其中的每一个控件进行动画处理描述，而写控件的动画处理的 XAML 文档基本相同，示例代码如下所示。

```
<Window.Resources>                                                    //窗体资源文件
    <Storyboard x:Key="OnClick1">                                     //定义了动画事件
    <DoubleAnimationUsingKeyFrames
        BeginTime="00:00:00"
        Storyboard.TargetName="label" Storyboard.TargetProperty=
        "(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">
        <SplineDoubleKeyFrame KeyTime="00:00:01" Value="-85"/>
        <SplineDoubleKeyFrame KeyTime="00:00:02" Value="49"/>
        <SplineDoubleKeyFrame KeyTime="00:00:03" Value="-86"/>
    </DoubleAnimationUsingKeyFrames>
```

上述代码定义了动画处理中变换的操作，在XAML文档中，动画处理都会被作为窗体资源而存在，而动画事件作为窗体触发器而存在，示例代码如下所示。

```
<Window.Triggers>                                                     //窗体触发器
<EventTrigger RoutedEvent="ButtonBase.Click" SourceName="button">
    <BeginStoryboard Storyboard="{StaticResource OnClick1}"/>
</EventTrigger>
</Window.Triggers>
```

上述代码定义了窗体触发器，当用户操作OnClick1事件后则会触发动画处理事件。开发人员能够在<Storyboard x:Key="OnClick1">标记中定义控件动画事件的其他内容以扩展WPF动画事件。

19.4.4 WPF事件编写

在WPF应用程序控件动画制作中，不能为了实现绚丽的动画而放弃了实用的功能。该应用程序希望用户能够进行登录并对用户的身份进行验证操作，如果验证成功则能够执行操作，而如果身份验证不成功，则无法执行搜索操作。在Visual Studio 2008中，双击按钮控件以进行登录验证操作，示例代码如下所示。

```
private void button_Click_1(object sender, RoutedEventArgs e)
{
    if ((textBox.Text == "admin") && (passwordBox.Password == "admin"))        //如果是管理员
    {
        Window2 w2 = new Window2();                                            //打开新窗口
        w2.ShowDialog();
    }
}
```

上述代码定义了用户如果输入了用户名和密码分别为admin/admin时，则验证成功，就会呈现搜索框，如果用户名和密码不正确，则无法验证进行搜索。进入搜索窗口时，用户可以在书籍搜索框中输入相应信息，当用户输入信息后，结果框就能够及时反映相应的搜索结果，示例代码如下所示。

```
public string[] books = { "ASP.NET 开发大全", "ASP 开发指南", ".NET 应用程序", "组件开发指南",
                          "PHP 新手入门", "C++学习" };
private void TextBox_TextChanged(object sender, TextChangedEventArgs e)        //用户查询
{
    if (!String.IsNullOrEmpty(search.Text))                                    //如果输入不为空
    {
        result.Clear();                                                        //清空结果
        for (int i = 0; i < books.Length; i++)                                 //遍历书籍
        {
            if (books[i].Contains(search.Text))                                //如果匹配则输出
```

```
                {
                    result.Text += books[i].ToString() + "\n";                    //填充结果控件
                }
            }
        }
    }
```

上述代码定义了一个数组以存储书籍的相应信息，当用户在搜索框中输入相应的信息时，系统就会遍历数据库进行书籍查询，运行结果如图 9-28 和图 9-29 所示。

图 9-28　搜索 C++相关书籍

图 9-29　清空结果后再搜索

19.5　小结

本章简单的讲解了 WPF 的基础知识，包括 WPF 和 WPF 的适用范围，WPF 是微软近几年力推的技术，随着 Vista 的普及，WPF 应用已经被越来越多的个人和企业接受，了解 WPF 技术在今后的项目开发中会起到很好的作用。本章还包括：

- 什么是 WPF：讲解了什么是 WPF，以及 WPF 引擎和 WPF 构架。
- 使用 Microsoft Expression Blend 设计 WPF：讲解了如何使用 Microsoft Expression Blend，以及如何使用 Microsoft Expression Blend 设计 WPF 应用程序。
- XAML 文档：讲解了 XAML 基本概念，以及如何通过 XAML 进行样式控制。
- WPF 控件层次：讲解了 WPF 中控件的层次概念。
- WPF 动画事件：讲解了 WPF 动画事件的概念，以及如何使用 Microsoft Expression Blend 开发动画事件。
- WPF 时间轴：讲解了时间轴的概念，以及使用时间轴进行动画开发。
- WPF 事件处理：讲解了如何使用 WPF 进行事件处理。

由于本书并不详细的讲解 WPF 应用开发，本书只是对 WPF 进行了简单的介绍。WPF 应用程序的开发和 Win From 开发基本相同，但是 WPF 提供了更好的开发和布局方案，使用 WPF 能够开发用户体验更好的应用程序。