WPF和Silverlight

项目设计实例

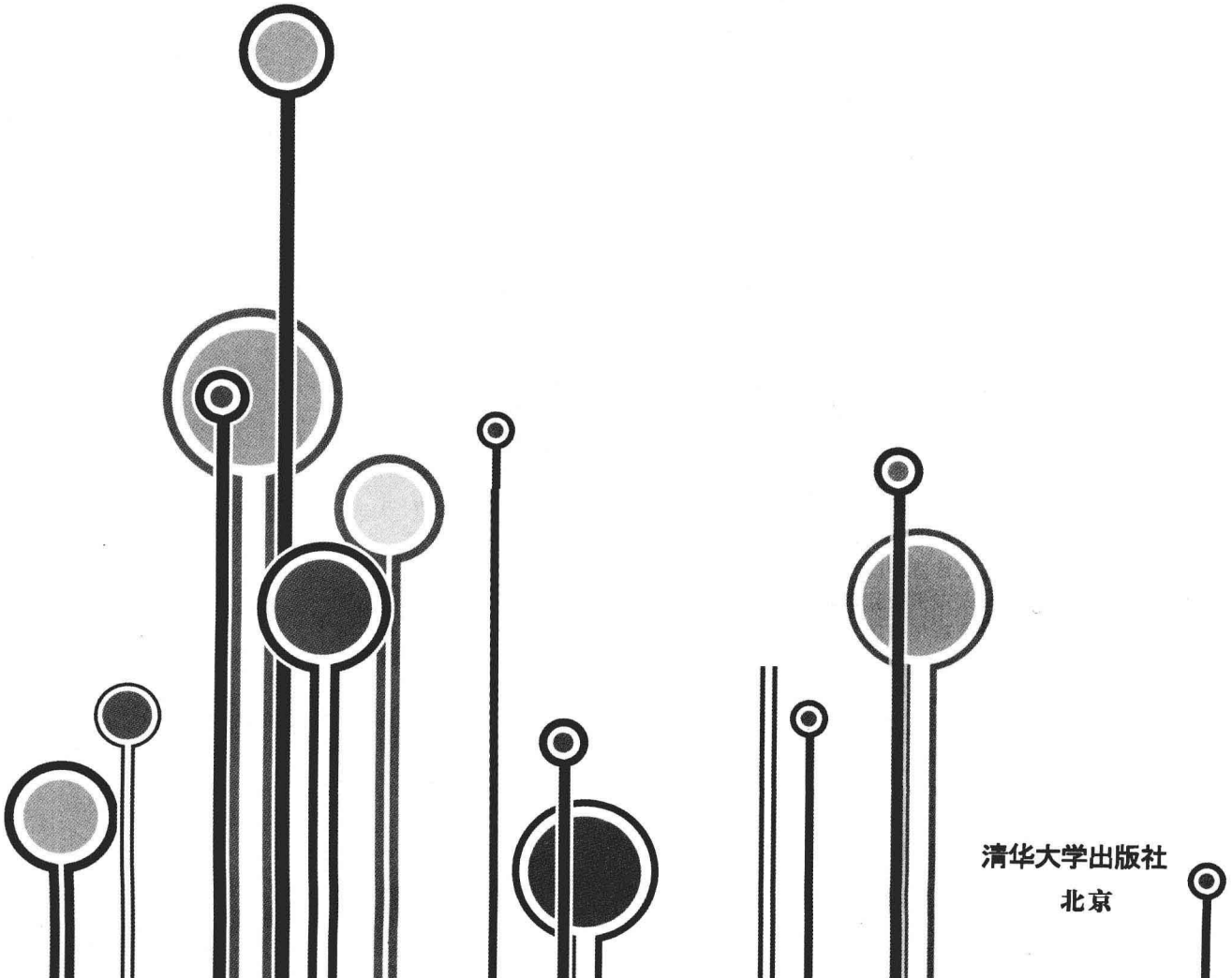
张洪定 主编
郭早早 梁竣 高飞 副主编

清华大学出版社

WPF和Silverlight

项目设计实例

张洪定 主编
郭早早 梁竣 高飞 副主编



清华大学出版社
北京

内 容 简 介

清华大学出版社2011年出版的《基于Expression Blend 4中文版WPF和Silverlight项目设计基础》是本书的基础。本书列举的实例是对前一本书的补充和深化，实例涉及“设计基础”一书没有介绍的控件，也涉及动画设计、二维图形描绘、图像动画处理、插件引入、视频播放、三维空间对象使用、XML文件应用、数据通信和数据库应用等内容，还涉及很多C#的编程技术和技巧。绝大部分实例设计难度均超过前本书，编程量加大了，适用范围也扩展了，应该是前一本书的进阶篇。掌握本书的实例设计对提高WPF和Silverlight的设计应用水平、构建大型项目是非常有益的。

本书适合从事WPF和Silverlight项目开发的设计人员参考，适用于计算机类、信息技术类、多媒体技术类、计算机艺术类、动画类、教育技术类等专业有关课程的教学或参考。

图书在版编目(CIP)数据

WPF和Silverlight项目设计实例/张洪定主编. --北京：清华大学出版社，2012.6
ISBN 978-7-302-28655-4

Ⅰ. ①W…　Ⅱ. ①张…　Ⅲ. ①Windows操作系统－程序设计　②网页制作工具－程序设计
Ⅳ. ①TP316.7　②TP393.092

中国版本图书馆CIP数据核字(2012)第077030号

责任编辑：索　梅　赵晓宁
封面设计：常雪影
责任校对：时翠兰
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：17.25　　**字　　数**：420千字
附光盘1张
版　　次：2012年6月第1版　　**印　　次**：2012年6月第1次印刷
印　　数：1～3000
定　　价：35.00元

产品编号：046012-01

FOREWORD

前　言

本书是在《基于 Expression Blend 4 中文版 WPF 和 Silverlight 项目设计基础》(本书涉及该书时简称“设计基础”)基础上编写的应用实例,每个实例均从应用设计出发做了较详细的介绍,给出了完整的设计程序,同时在光盘中给出了实例的项目源文件,供读者参考。本书对涉及“设计基础”的基本操作不再详细介绍,每个实例相对独立,涉及的知识点均有注解。

WPF 和 Silverlight 的项目设计平台可以同时使用 Expression Blend 或 Visual Studio 2010,考虑到 Expression Blend 具有非常友好的 UI 设计功能,所以本书的实例主要以此平台为主,当此平台“力所不能及”时,使用 Visual Studio 2010 平台做前期开发,后期再回到 Expression Blend 平台。Expression Blend 平台使用的是 2010 年微软公司发布的简体中文版第 4 版。好在都是微软的产品,在两个平台交叉使用时可以实现“无缝连接”。

WPF 和 Silverlight 的项目设计涉及界面设计和界面的逻辑设计,前者使用 XAML 语言;后者使用 C# 语言(本书使用此语言)。因为在设计中界面设计的 XAML 代码是自动生成的,目前没有必要花费很多笔墨去描述它,所以本书的所有实例均没有列出 XAML 代码,需要时也只列出了有关部分,有兴趣的读者可以查看光盘中的源文件。这是本书写作的出发点,也是和其他同类书籍不一样的地方。当然,应该了解 XAML 语言的语法结构,这对局部修改、整体调试和其他工具软件连接是有帮助的。

本书分为 2 章。第 1 章是 WPF 项目设计实例;第 2 章是 Silverlight 项目设计实例。实例涉及“设计基础”一书没有介绍的控件,也涉及动画设计、二维图形描绘、动画特效处理、插件引入、自定义视频播放器设计、三维空间对象使用、xml 文件应用、数据通信和数据库应用等内容,还涉及很多 C# 的编程技术和技巧。总体来说,这些实例不大,有的还很小,但解决了一些应用中的实际问题,可以搭成大型项目的应用案例。当然实例程序重在目标实现,没有仔细做优化工作。

目前国内利用 WPF 和 Silverlight 技术开发项目的应用还不多,但国外研究开发者较多。作者经过几年的教学和应用,认为这确实是“RIA”设计的优秀主流软件,软件将多媒体呈现,包括二维、三维对象的呈现、数据访问集成开发,节省了很多中间设计环节,改变了很多传统软件在桌面项目开发和网络项目开发的设计面貌,使得有些原来常用的软件“黯然失色”,这本书交

稿时微软公司的第5版还没有发布，相信第5版的功能将会更加强大，在下一代操作系统面向多媒体环境的应用设计、面向企业的项目设计、面向微软移动设备的应用设计和面向新一代网络设计标准HTML5中发挥更大作用。

本书涉及程序内容较多，书中的变量均用正体。本书没有给出习题，书中全是实例，如果读者能模仿书中的实例设计自己的应用，也就达到目的了。书中光盘的所有源文件只要复制到读者的计算机中，就可以直接在开发平台中打开运行、调试。笔者的博客网址为http://blog.sina.com.cn/u/2502744874，主要讨论WPF和Silverlight的项目设计和应用，欢迎读者参与。在那里可以更深入讨论书中有关问题。

感谢清华大学出版社对本书出版的支持。感谢南开大学滨海学院、计算机科学系在书稿编写过程中的支持。

编者

2012年5月

CONTENTS

目 录

第1章 WPF应用实例

本章的 WPF 的项目设计实例涉及部分控件应用、故事板的控制方法，部分动画设计示例、动画缓动曲线、二维图形描绘、文字和图像动画特效、图像色彩动画处理、嵌入插件和 EXE 文件、自定义视频播放器设计、三维空间对象应用等。实例中的编程都有注解，有的实例的具体操作需要参考“设计基础”，本书的出发点是应用，不做过多理论探讨。

1.1 WPF Expander 控件

Expander 控件可以展开、卷缩对象内容，卷缩时可以只占一行高度，展开大小可以设定，它使界面设计具有灵活性。

本实例项目中有两个 Window 窗口，一个是 MainWindow，一个是 Window1，两个窗口有不同的示例。

图 1-1 所示是展示 Expander 控件基本功能的示例(在 Window1 中)。图 1-1 左边是 Expander 控件，现在是向下展开状态，卷缩时只看到“Expander 示例”，其左侧有一个指示箭头，目前箭头向上，表示单击它就可以卷缩。图 1-1 右边是展开方向的设置选择。

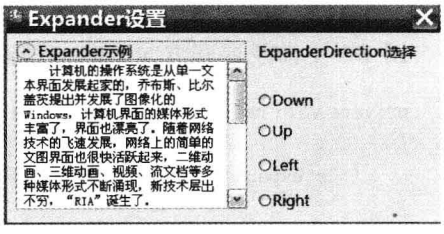


图 1-1 Expander 展开方向设置

1. 典型属性

Herder：Expander 的标题(在“公共属性”栏中)。

ExpanderDirection：展开方向设置有 4 种(在“公共属性”栏中)，Down 表示向下展开，Up 表示向上展开，Left 表示向左展开，Right 表示向右展开。

IsExpanded：控制 Expander 是否展开(在“公共属性”栏中)。bool 类型，选中为 false 表示卷缩，true 表示可以展开，默认值可以展开。

2. 典型事件

Expanded：当单击 Expander 控件从卷缩到展开时发生。

Collapsed：当单击 Expander 控件从展开到卷缩时发生。

3. 基本功能演示

这里说的基本功能演示是图 1-1 的示例，在项目的 Window1 窗口中，主要介绍 Expander 控件的基本功能。

从【工具】面板的“资产”栏中选择 Expander 控件拖入【设计面板】，这时的【对象和时间线】面板如图 1-2 所示。

图 1-2　Expander 结构

从图中看出 Expander 是个容器，内部默认包含标题 Herder 和一个控件 Grid，Expander 只能放置一个控件。如果在 Expander 内放入控件可能会出现两种情况：一种是替换了 Grid 的位置；另一种情况是放入 Grid 中，因为 Grid 本身可以容纳多个控件对象。以图 1-1 中的 Expander 为例。

在【对象和时间线】面板选中 Expander，在“公共属性”栏中修改 Header 属性，拖动 Expander 边缘放大到图 1-1 的窗口高度，可以看到内部的 Grid 的大小是伴随变化的。

选中【对象和时间线】中的 LayoutRoot 根布局，将一个 RichTextbox 文本框(命名为 rbt1)放入 LayoutRoot 中，完成文本录入。如果在【设计面板】直接将 rtb1 拖进 Expander 可能会将其中的 Grid 替换，在【对象和时间线】面板中用鼠标将 rbt1 拖进 Grid 就没有问题了，这里采用的是后一种操作方法，所以在图 1-1 中，看到的 rbt1 是在 Expander 的 Grid 控件中。

图 1-1 中涉及展开方向的动态设置，使用了下面的代码：

```
private void rb1_Click(object sender, System.Windows.RoutedEventArgs e)
{
    this.expander.ExpandDirection = ExpandDirection.Down;
}
private void rb2_Click(object sender, System.Windows.RoutedEventArgs e)
{
    this.expander.ExpandDirection = ExpandDirection.Up;
}
private void rb3_Click(object sender, System.Windows.RoutedEventArgs e)
{
    this.expander.ExpandDirection = ExpandDirection.Left;
}
private void rb4_Click(object sender, System.Windows.RoutedEventArgs e)
{
    this.expander.ExpandDirection = ExpandDirection.Right;
}
```

这些代码是 Expander 的展开方向设置。

4. 应用实例

图 1-3 和图 1-4 展示了 Expander 的一个应用实例，在项目的 MainWindow 窗口中。

图 1-3 中有两个 Expander，展开后界面大小布满窗口，上方一个，名为 expander1，标题是“内容介绍”，展开方向 ExpanderDirection 设为 Down；下方一个，名为 expander2，标题为“操作视频”，展开方向 ExpanderDirection 设为 Up，其中只有一个视频，在 MediaElement 控件中，命名为 me，其中的视频文件在项目添加的文件夹 picture 中，me 的属性 LoadedBehavior(在属性“媒体”栏中)设为 Manual，这样其播放、停止要用代码控制。me 位于 expander2 的 Grid 控件中。在图 1-3 中目前显示的是 expander1 卷缩，expander2 向上展开的情况。

图 1-3 expander2 展开

在图 1-4 中，expander1(展开方向 ExpanderDirection 设为 Down)内包含三个控件，均位于 expander1 的 Grid(名为 grid1)中，其中两个 RichTextbox 和一个 Image 控件。两个 RichTextbox 中一个称为 rtb1，位于左上方，内部是文本；一个称为 rtb2，位于下方，内部有文本和一个图片；Image 控件(名为 image)位于右上方。图片文件从文件夹 picture 插入。

向 Expander 中放置控件注意上面提到的操作，保证对象放置在 Grid 中。

在【对象和时间线】面板中的排列顺序，expander1 位于 expander2 的下层，因为这两个控件都布满窗口(否则不能完全显示)，项目运行时需要动态设置它们的层次；否则有些界面不能看到或发生重叠。下面是程序设计。

```
//expander1 展开事件
private void expander1_Expanded(object sender, System.Windows.RoutedEventArgs e)
{
    //设置 expander1 到最上层，根布局只有 2 个元素，本例中 1 是上层序号
    Grid.SetZIndex(this.expander1,1);
    //卷缩 expander2
    this.expander2.IsExpanded = false;
    //停止 expande2 中的视频播放
```

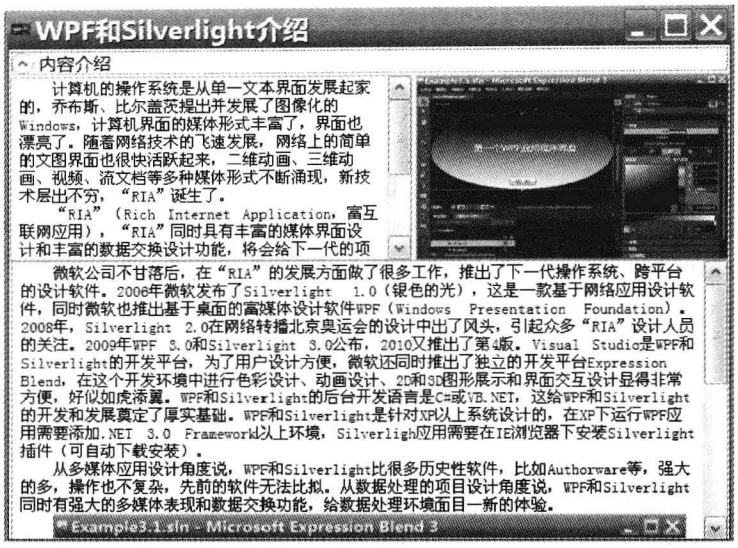


图 1-4　expander1 展开

```
    this.me.Stop();
}
//expander1 的卷缩事件
private void expander1_Collapsed(object sender, System.Windows.RoutedEventArgs e)
{
    //expander1 设置为底层,0 是底层序号
    Grid.SetZIndex(this.expander1,0);
}
//expander2 展开事件
private void expander2_Expanded(object sender, System.Windows.RoutedEventArgs e)
{
    //设置 expander2 到最上层
    Grid.SetZIndex(this.expander2,1);
    //视频播放
    this.me.Play();
}
//expander2 的卷缩事件
private void expander2_Collapsed(object sender, System.Windows.RoutedEventArgs e)
{
    //expander2 设置为底层
    Grid.SetZIndex(this.expander2,0);
}
```

1.2　WPF ToolBar 和 ToolBarTray 控件

Toolbar 控件是工具条控件，是可以容纳多个控件的容器，可以在其中放置多个按钮控件组成 WPF 应用的工具。ToolBarTray 是 ToolBar 的工具盒，是可以放置多个 ToolBar 的容器，和 Word 软件中的工具条类似。

1. ToolBar 控件

图 1-5 所示是 ToolBar 控件的外观，其左侧有一排纵向虚线标记，当鼠标停在此处时鼠标形状会变为带箭头的十字形，表示可以移动，当程序运行时可以拖动 ToolBar 的虚线标记，移动 ToolBar，从而改变布局；其右侧下方有一个向下的三角箭头标志，当放入 ToolBar 的控件溢出时，单击此三角箭头可以弹出溢出的控件供使用。

其典型属性如下所示。

Header：可以输入文本作为 ToolBar 的标题(公共属性栏中)，一般位于 ToolBar 的左侧。

图 1-5 ToolBar 控件

Background：背景色。Foreground：Header 的颜色。

2. ToolBarTray 控件

ToolBarTray 刚放入设计面板时也仅仅是个空白的矩形控件。

其典型属性如下。

Orientation：排列方式，有水平 Horizontal 和垂直 Vertical 两种选择，这样工具条可以横向或纵向排列在界面中。

Background：背景色。

3. 应用实例

图 1-6 所示是 ToolBar 和 ToolBarTray 控件应用的一个实例，上方是菜单，菜单下方是工具栏。工具栏由一个工具盒 TooBarTray(图中矩形线框)和两个包含其中的工具条 ToolBar 组成，每个工具条包含标题、6 个图形按钮和 1 个分割线。下面说明制作过程。

在项目中添加了一个文件夹 icon，其中添加了图标文件(png 格式，大小 25×25)。

从图形按钮说起。从【工具】中将按钮 Button(命名 b1)放入【设计面板】，调整其大小，比图标文件稍大一点，从 icon 文件夹拖 1 个图标到此按钮上，按钮也是只能容纳 1 个控件的容器，如图 1-7 所示。

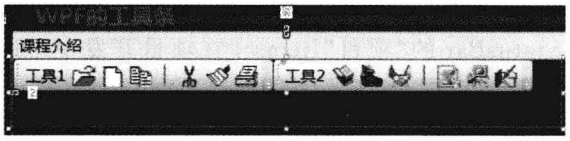


图 1-6 实例界面

图 1-7 图形按钮

从“资产”中找到 ToolBar 控件(命名为 tooolbar1)拖入【设计面板】中，在【对象和时间线】内选中 b1(注意不是其中的图标)，将 b1 拖进 toolbar1(接近边框时有提示为“按 Alt 键以放入 toolbar1 中”)，也可以在【对象和时间线】面板中完成这个操作。然后复制 b1、粘贴，这时自动粘贴到 toolbar1 内部并在 b1 的后侧排列，更名为 b2，从 icon 文件夹再拖 1 个图标到 b2。同样的办法生成 b3。3 个图形按钮完成。

从“资产”中找到分割线 Separator 控件，拖入【设计面板】，这时 Separator 是水平方向的一条横线，调整其宽度小于 toolbar1 的高度，设置其背景色 Background。在【属性】面板的“转换”栏中将其旋转 90°，Separator 变为垂直方向，将其拖入 toolbar1，会自动排列在 b3 后面(也可以在【对象和时间线】面板中完成这个操作)。

用上面复制、粘贴、更换图标的办法制作 b4、b5 和 b6。最后选中 toolbar1，将“公共属性”栏中 Herder 属性赋值为“工具 1”，toolbar1 制作完成，如图 1-6 的左边。Header 属性的默认值是空，赋值操作应该在分割线放入以后，否则有可能造成分割线不可见。

复制 toolbar1，粘贴生成一个新的 ToolBar，更改其中 6 个按钮的图标，命名为 toolbar2，图 1-6 中的右侧。

从“资产”栏中找到工具盒 ToolBarTray 控件(命名为 toolbartray)，拖到【设计面板】，将 toolbar1 和 toolbar2 拖入其中，也可以在【对象和时间线】面板中完成这个操作。

设计时 toolbartray 的高度，应该等于 toolbar1 和 toolbar2 的高度和，便于两个 ToolBar 的上下移动。toolbar1 和 toolabar2 在“布局”属性栏中 Marginde 4 个偏移值均为 0，否则拖放时会出现位置偏移。

菜单设计中只有“退出”项有代码：

```
Application.Current.Shutdown();
```

当项目运行时，当鼠标停在 toolbar2 左侧的虚线标识，鼠标图标变为十字箭头时拖动 toolbar2 向下、向上、向左、向右，toolbar1 和 toolbar2 的排列变化如图 1-8 和图 1-9 所示。

在图 1-8 的位置再拖动 toolbar2 向上，toolbar1 可能会缩短，图形按钮没有丢失，只要单击 toolbar1 右下方的三角标志就会出现。这时再拖动 toolbar2 向右就会恢复到图 1-6 的位置。在图 1-6 中拖动 toolbar2 向左，也会出现图 1-9 中的布局。

图 1-8　拖动 toolbar2 向下

图 1-9　再拖动 toolbar2 向上

1.3　WPF StatusBar 控件

StatusBar 控件是状态栏工具，一般放置在项目窗口的下端。此控件是容器，可以静态或动态向其中放置多个控件，成为 StatusBar 的“项目”Items。在项目开发中可以作为状态栏、提示栏。

1. 典型属性

Background、Width、Height 是常用属性。

BorderBrush：边框颜色。

BorderThickness：4 个矩形边框的宽度，可分别设置，在“外观”属性栏中。

Margin：状态栏的位置偏移(在“布局”属性栏中)。

2. 典型事件

典型事件有 MouseEnter()、MouseLeave()、MouseLeftButtonDown()等，一般较少使用此控件的事件编程。

围绕放入 StatusBar 中的 Items 也有一些属性和事件方法，使用中说明。

3. 应用实例

图 1-10 所示是使用 StatusBar 控件的一个应用实例。

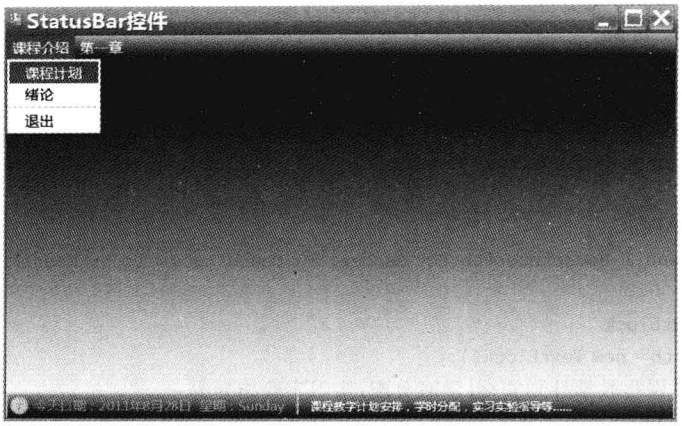


图 1-10 StatusBar 控件应用实例

图 1-10 上方是菜单，下方“状态栏”是 StatusBar 控件，当鼠标停在某一下拉菜单项时状态栏就会出现相关内容提示。图 1-11 是使用的 StatusBar 控件(名为 statusbar)。

图 1-11 statusbar 中的 Items

图 1-11 中显示的 statusbar 控件有 4 个 Items，左边 3 个分别是静态设置的图片(来自项目中添加的文件夹 images 中，clock. png)、statusbar 的本身添加项 StatusBarItem(用于显示日期和星期)和 1 个外添加的分割线 Separator，右边是一个动态添加的文本框，用于显示鼠标所停的下拉菜单项的相关内容提示，是程序中设置的。

需要说明的是，放入 StatusBar 控件中的项目会自动编号，从 0 开始，图 1-11 中的 4 个 Items 编号从左到右分别是 0～3。

先介绍静态添加的前 3 个 Items。首先从“资产”栏中将 StatusBar 控件拖入设计窗口，设置好位置、长短、边框和背景。

从 images 文件夹将图片 clock. png 拖入 ststusbar，设置好其大小，第一个 Items 添加完成。

选中 statusbar 项，右击，在弹出的快捷菜单中选择 StatusBarItem 选项，命名为 sbt1，添加的 StatusBarItem 也有 Background、Width、Height、BorderBrush、BorderThickness、Margin 等属性设置，还有 Foreground 设置，这里将其设为“绿色”，还有一个属性 Content 可以动态设置，用于显示日期。

从“资产”栏中将 Separater 控件拖入设计窗口，设置其 Background 为“浅灰色”，在“转换”属性栏中将其旋转 90°，变为竖线，拖入 statusbar 中。

前 3 个 Items 添加完成。第 4 个 Items 由程序添加，对每个菜单项的程序设计是一样的，下面只列出一个菜单项的程序。

```
int items;//Items 计数
public MainWindow()
{
```

```
        this.InitializeComponent();
        //显示日期、星期
        this.sbt1.Content = "今天日期: " + System.DateTime.Now.ToLongDateString() + " 星期: " +
    System.DateTime.Now.DayOfWeek.ToString();
        //获取 statusbar 的 Items 总数(本例应该为 3,正好是要添加项的编号)
        items = this.statusbar.Items.Count;
    }
    //鼠标悬停菜单项
    private void menu11_MouseEnter(object sender, System.Windows.Input.MouseEventArgs e)
    {
        //创建 TextBlock
        TextBlock tb = new TextBlock();
        tb.Text = "课程教学计划安排,学时分配,实习实验指导等……";
        //文本颜色设置
        tb.Foreground = Brushes.White;
        //statusbar 添加 1 项
        this.statusbar.Items.Add(tb);
    }
    //鼠标离开菜单项
    private void menu11_MouseLeave(object sender, System.Windows.Input.MouseEventArgs e)
    {
        //从 statusbar 中移除添加项
        this.statusbar.Items.RemoveAt(items);
    }
```

其中,Items.RemoveAt()移除指定编号的 Items,参数 items 指定编号。

1.4 WPF RepeatButton 控件

RepeatButton 属于按钮控件,和普通按钮 Button 的主要差别在:

普通按钮 Button 每单击一次(包括单击并按住不动)只发生 Click 事件一次,而同样情况下,甚至鼠标悬浮在 RepeatButton 上方就可以连续多次发生 Click 事件,这样使得其中的代码被连续反复执行。下面设计了两个实例,一个是秒表计数;另一个是模拟示波器波形垂直位置调整。

1. 典型属性

下面介绍的典型属性全部在"公共属性"栏中。

ClickMode:Click 事件的触发方式,有 3 个选择。Press,鼠标按下时发生;Release,鼠标释放时发生;Hover,鼠标悬浮在按钮上方时发生。

Content:按钮标题。

Delay:事件延发时间,单位毫秒。

Interval:事件连续发生时的时间间隔,单位毫秒。

Cursor:光标设置。

2. 典型事件

Click 事件包括鼠标单击、鼠标悬停。

3. 实例一：秒表

图 1-12 所示是秒表实例的界面。图中左边是两个 RepeatButton 按钮，上方按钮名为 rb1，下方按钮名为 rb2。

图 1-12 秒表实例

当鼠标悬停在 rb1 上时，秒表启动开始计数，以 0.1 秒为计数单位，这时连续发生 rb1 的 Click 时间，计数器连续计数，直到鼠标离开 rb1，计数立即停止。rb1 的 Delay 设为 0，ClickMode 选为 Hover(悬浮在按钮上方)，Interval 设为 100，Cursor 选择的是 Wait。计数显示是 Textblock 控件(textblock1)。计数显示共 5 位，包括小数点在内。为了保证计数过程一直是 5 位显示，程序中在数字不足 5 位时数字前自动补 0，整数时小数点后面自动补 0，程序如下：

```
//rb1 的 Click 事件
private void rb1_Click(object sender, System.Windows.RoutedEventArgs e)
{
    //获取当前计数值,字符串转换为数值
    double js = double.Parse(this.textblock1.Text);
    //计数最大值限制
    if (js<999.5)
    {
      js = js + 0.1;
    }
    //取整数部分,舍去小数
    int js1 = (int)System.Math.Floor(js);
    //如果整数部分只有 1 位
    if (js1<10)
    {
    this.textblock1.Text = "00" + js.ToString();
    //如果计数值是整数
    if (js== System.Math.Floor(js))
    {
    this.textblock1.Text = this.textblock1.Text + ".0";
    }
    return;
    }
    //如果整数部分有 2 位
    if (js1<100)
    {
    this.textblock1.Text = "0" + js.ToString();
    //如果计数值是整数
    if (js== System.Math.Floor(js))
    {
    this.textblock1.Text = this.textblock1.Text + ".0";
    }
    }
}
```

rb2 是复位键，ClickMode 选择 Press，实际普通按钮就可以，这里用的也是 RepeatButton，代码简单：

```
private void rb2_Click(object sender, System.Windows.RoutedEventArgs e)
{
    this.textblock1.Text = "000.0";
}
```

4. 实例二：示波器波形垂直位置调整

图 1-13 所示是设计界面，其中用的图片来自项目添加的文件夹 images 中。左边是一个示波器屏幕图片，被放在布局控件 Canvas(名为 canvas1)中，示波器屏幕图片和 canvas1 的大小一致，屏幕中有一条绿色水平指示线，代表波形(1 条直线，名为 lineh，其 Stroke 画笔属性设为绿色，线宽 StrokeThickness 为 2)，通过它可以看到位置调整情况，Canvas 的 GetLeft()、GetTop()、SetLeft() 和 SetTop()等方法使得波形移动的定位获取和设置非常方便。图 1-13 右边的图由旋钮图片和刻度图片组成，全部组合在布局控件 Grid 中，这里关键是旋钮。旋钮实际上是由两个对接完整的图片组成(一个完整的按钮利用 Photoshop 被平均划分成两张图)，左边一半名为 imageLeft.png，右边一半名为 imageRight.Png，这两个图片分别被“构成控件”RepeatButton，左边的按钮称为 repeatbuttonLeft，右边的按钮称为 repeatbuttonRight(构成控件的方法在“设计基础”中讲过，选中图片，右击，在弹出的快捷菜单选“构成控件”，在弹出的窗口中选择 RepeatButton 项)。这两个按钮被组合在布局控件 Grid(命名为 gridxn)，利用这两个按钮事件中的代码控制波形的垂直位置调整，同时为获得较好的仿真效果当调整时旋钮也要转动，实际上是 gridxn 转动。对象的转动使用旋转变换 RotateTransform 可以实现，需要编程，这里采用了“取巧”的办法跳过并简化了有关程序设计，具体过程如下：

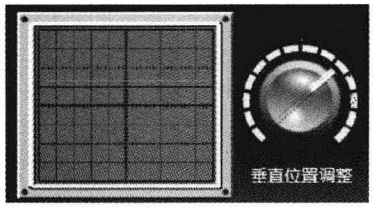


图 1-13 示波器波形垂直位置调整

当 gridxn 组成后在 MainPage.xaml 文件中有一段对它的描述：

```
<Grid x:Name = "gridxn" Margin = "13,14.328,17,4.672">
< RepeatButton x: Name = " repeatbuttonLeft" Content = "" Margin = " 0, 0, 34, 0" Style =
"{DynamicResource RepeatButtonStyle1}" Cursor = "ScrollSW" Click = "repeatbuttonLeft_Click"/>
< RepeatButton x:Name = "repeatbuttonRight" Content = "" HorizontalAlignment = "Right" Style =
"{DynamicResource RepeatButtonStyle2 }" Width = " 34 " Cursor = " ScrollSE " Click =
"repeatbuttonRight_Click"/>
</Grid>
```

这一段没有 RotateTransform 的描述，因为还没有定义。

首先确定 RotateTransform 的旋转中心，当两个图片构成按钮并组合成 gridxn 后其中心点会出现偏移，也许偏移值很小，可以微调。选中 gridxn 项，在属性“转换”栏中找到“中心点”，微调其数值，使旋钮的中心点位于正中位置，如图 1-14 所示。

微调后的 X、Y 数值分别是 0.52，0.52。

中心点调整后继续选择属性“转换”中的“旋转”，任意设置 1 个角度值，这时 xaml 文件