اتصال userinterface با Data Provider

مقــدمه

تقید داده به تفکیک لایه رابط کاربر در برنامه از سایر لایه های دیگر کمک میکند. مسولیت این تفکیک به وسیله تجزیه (decoupling)هدف، باید ازمنبع آن دوباره بازیابی گردد، هر چند از اشیاء تقید استفاده شود.

در این مقاله با :

1- اتصال داده ها
2- مدهای تقید داده
3- اعمال تغییرات در NotifyProperty
4- نمایش داده
5- بایند کردن TextBlocks
6- DataContext
7- مراحل منطقی تقید داده
8- ListBox و بایند کردن آن به لیست

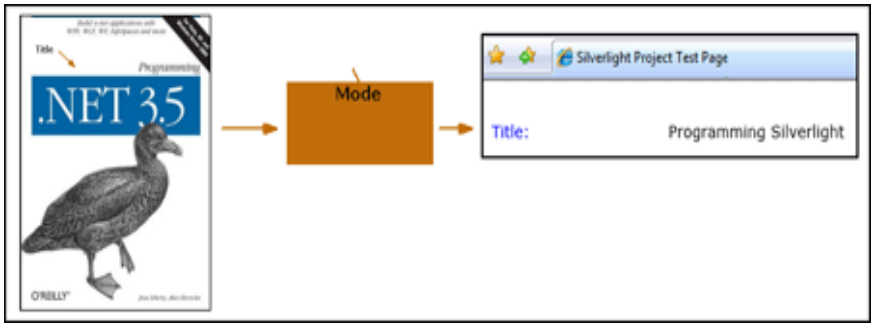
آشنا خواهید شد .

اتصال داده ها

اتصال داده ها (تقید داده ها) رابطه ای بین رابط کاربر و عناصر فراهم آورنده داده میباشد . اصطلاحاً به رابط کاربر، هدف و به فراهم آورنده داده ، منبع میگویند.

تقید داده به تفکیک لایه رابط کاربر در برنامه از سایر لایه های دیگر کمک میکند. مسولیت این تفکیک به وسیله تجزیه (decoupling)هدف، باید ازمنبع آن دوباره بازیابی گردد، هر چند از اشیاء تقید استفاده شود.

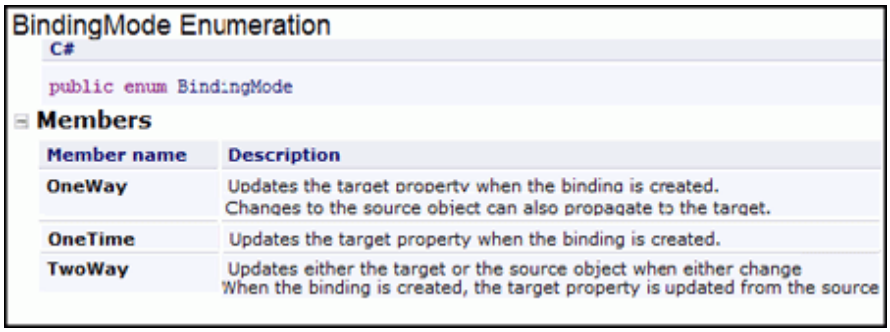
عنصر تقید میتواند مانند یک جعبه سیاه با چند درگاه اتصال عمومی در یک طرف آن به سمت هدف و در اطراف دیگر به سمت منبع تصور شود . چند سوئیچ در بالا وجود دارد که برای تعیین مد سوئیچ از جریان داده ها به کار میروند و بسیار مهم هستند .



شکل 1-تقید داده به عنوان رابط بین هدف و منبع

مدهای تقید داده

مد تقید یک داده شمارشی است که میتواند یکی از سه مقدار زیر را بگیرد .



شکل 2 - متن مد تقید شمارشی

تقید OnTime: هدف را تنظیم و تقید را کامل میکند . این تقید برای نمایش داده هایی که به ندرت یا هیچ گاه تغییر نمیکنند ، بسیارعالی است .

تقید One Way: هدف را تنظیم و تغییرات منبع را به صورت به روز شده نگه میدارد. این تقید برای نمایش داده هایی است که کاربراجازه تغییر داده ها را ندارد.

تقید Twoway:هدف را تنظیم و تغییرات منبع را به روز کرده و تغییراتی که کاربر روی هدف انجام میدهد و یا تغییراتی که باعث تغییر روی منبع میشود، را نگهداری میکند.

اگریک کتاب فروشی آنلاین و نمایش اطلاعات یک کتاب را داشته باشید، ممکن است از تقید OnTime روی عنوان و نام نویسنده وتقید OnWay روی قیمت و تقید TwoWay روی تعداد در دسترس کتابها ، استفاده نمایید .
هدف تقید شما میتواند یک خصوصیت عمومی از اشیاء مجازی CLR باشد .
این تقیدها را میتوانید در یک مثال کوچک مشاهده کنید، اما در این تمرین از یک شیوه سه مرحله ای برای جدا سازی استفاده کرده ایم:

- لایه کاربر
- لایه مجازی
- لایه ماندگاری

لایه کاربرمتشکل ازکنترل هایی است که ما از جعبه ابزار به دست آورده و به عنوان پیشنهاد هم اکنون استفاده میکنیم ، اگرچه یک خود اموزبعداً نحوه استفاده از Style ها و قالب ها را خواهد اموخت.
لایه تجاری بوسیله کلاسی به نام Book ارائه خواهدشد .
فعلا لایه ماندگاری را نادیده گرفتیم ، هرچند این جزئی بزرگ در خودآموز میباشد.

اعمال تغییرات در NotifyProperty

یک برنامه جدید در سیلور لایت بسازید. این برنامه باید شامل دو فایل Page.xaml و Page.xaml.cs باشد.
به برنامه یک book.cs اضافه نمایید . که این در لایه تجاری نمایش داده خواهد شد .
آنچه یک شی تجاری ، سیلورلایت را ازیک پروژه ای که مثلاً در Asp.net ساخته شده ، تفکیک میکند ، چیزی است که میخواهیم درشی تجاری برنامه خود با یکی ازتقیدهای OneWay یا TwoWay در لایه UI به کار گیریم .
در کنترلی که مشخص کننده تغییرات شی تجاری است ، شی جاری باید با INotifyPropertyChanged پیاده سازی شود .

این رابط فقط به یک چیز نیاز دارد : کلاسی که یک رویداد روی نوع ChangedEventHandler داشته باشد . این تقید به طور ضمنی مورد پشتیبانی میباشد . هرچند به طور معمول، هنگامی که هر یک از خاصیت های کنترل های رابط کاربری از کار افتاده وتغییر کرده باشند، باید شی تجاری رویداد PropertyChanged شروع کند (وبه طورعمده راه تغییر آنها تنظیم همین مقادیر است).

```
public class Book : INotifyPropertyChanged
 {
    private string bookTitle;

    public event PropertyChangedEventHandler PropertyChanged;
```

```
    public string Title
    {
        get { return bookTitle; }
        set
        {
            bookTitle = value;
            if ( PropertyChanged != null )
            {
                PropertyChanged(this, new PropertyChangedEventArgs("Title");
            }
        }
    }
                                                                                    }
```

اولین چیزی که مشاهده میشود استفاده از دستور System.ComponentModel است که برای رابط INotifyPropertyChanged کلاس مذکور ضروری میباشد . این رابط را در تعریف کلاس باید انجام داد.

```
public class Book : INotifyPropertyChanged
```

در رویداد PropertyChangedEventHandler وجود الگویی با نام PropertyChanged لازم میباشد.

در این نسخه ازکلاس Book فقط یک میدان BookTitle داریم که خاصیت Title را نشان میدهد. نمیتوانیم از دستور خاصیت C#3.0 استفاده کنیم .

```
public class Book : INotifyPropertyChanged
 {
    public event PropertyChangedEventHandler PropertyChanged;

    public string Title { get; set; }
                                                                                    }
```

می توان با اینکه در Get کاری انجام نمی شود مقداری را به برنامه برگرداند یا در Set مقداری را تنظیم کرد. برنامه تنظیم کننده ما یک قسمت از این عمل را انجام می دهد . چک کردن این که با رویدادی که ما نوشته ایم کسی ثبت نام شده یا نه و اگر متد ثبت نام فراخوانی شده , متد Delegate فعال شده یا خیر.

چنانچه خواسته باشیم خاصیت های بیشتری به برنامه اضافه کنیم ، متد set هر کدام ، رویداد رجیستر را چک خواهد کرد که کدام فراخوانی شده است . انتقال این کد Skin ی را به نام Craw1 خواهد ساخت . این کدها که اندازه های خیلی خوبی ندارند ، مقداردهی های درست و خطایابی انها کمی سخت است . عاملی که مسئولیت این کار را به عهده دارد در متدی که برای هر خاصیت فراخوانی میشود، قرار دارد .

```
public class Book : INotifyPropertyChanged
  {
    private string bookTitle;

    private string bookAuthor;

    private int quantityOnHand;

    private bool multipleAuthor;

    private string authorURL;

    private string authorWebPage;

    private List<string> myChapters;


    // implement the required event for the interface

    public event PropertyChangedEventHandler PropertyChanged;

    public string Title
    {
```

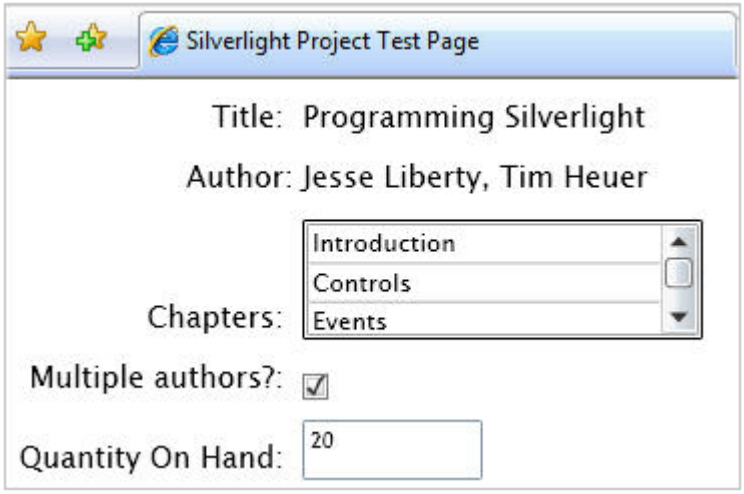
```
    get { return bookTitle; }
    set
    {
        bookTitle = value;
        NotifyPropertyChanged("Title");
    }       // end set
}           // end property

public string Author
{
    get { return bookAuthor; }
    set
    {
        bookAuthor = value;
        NotifyPropertyChanged("Author");
    }       // end set

}

public List<string> Chapters
{
    get { return myChapters; }
    set
    {
        myChapters = value;
        NotifyPropertyChanged("Chapters");
    }
}


public bool MultipleAuthor
{
    get { return multipleAuthor; }
    set
```

```
        {
            multipleAuthor = value;
            NotifyPropertyChanged("MultipleAuthor");
        }       // end set
    }



    public int QuantityOnHand
    {
        get { return quantityOnHand; }
        set
        {
            quantityOnHand = value;
            NotifyPropertyChanged("QuantityOnHand");
        }       // end set
    }


    // factoring out the call to the event
    public void NotifyPropertyChanged(string propertyName)
    {
        if (PropertyChanged != null)
        {
            PropertyChanged(this,
                new PropertyChangedEventArgs(propertyName));
        }

    }
}
```

توسط عامل PropertyChanged ، میتوان از نام هر خاصیت عبور کرد وdelegate آن را دوباره مورد استفاده قرار داد .

نمایش داده

برای نمایش بهتر و آسانتر داده ها باید یک فرم با دو ستون ساخت. در ستون سمت چپ یک Prompt و در ستون سمت راست داده ها قرار میگیرند .



شکل3- نمایش خصوصیات یک کتاب

درمورد چگونگی گرفتن اطلاعات در حال حاضر بحثی وجود ندارد و مهم نحوه نمایش داده ها میباشد. تمام برچسب های ستون سمت چپ در فرم به عنوان TextBlocks ساخته میشوند. دو برچسب بالایی در سمت راست بلاکهای متنی هستند که هر کدام توسط یک checkBox وListBox ویک TextBoxساخته میشوند. تفاوت بین بلاک متنی و جعبه متن در ورود داده ها است . در خود آموز قبلی روی کنترل ها طریقه ساخت Grid را با کمک ردیف ها و ستون ها و محل قرار گرفتن کنترل ها و محل نمایش داده ها در آنها مشاهده نمودید. در این قسمت کدهایی برای تمام کنترل هایی که این سوال را باقی میگذارند که از کجا به اتصال داده ها نیاز داریم، وجود دارد .

```
<UserControl x:Class="BookProperties.Page"

   xmlns="http://schemas.microsoft.com/client/2007"

   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

   Width="400" Height="300">

   <Grid x:Name="LayoutRoot" Background="White">
```

```
    <Grid.RowDefinitions>
        <RowDefinition MaxHeight="30" />
        <RowDefinition MaxHeight="30" />
        <RowDefinition MaxHeight="70" />
        <RowDefinition MaxHeight="30" />
        <RowDefinition MaxHeight="40" />
        <RowDefinition MaxHeight="50" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition MaxWidth="150"/>
        <ColumnDefinition MaxWidth="200" />
    </Grid.ColumnDefinitions>
    <TextBlock x:Name="TitlePrompt" Text="Title:  "
        VerticalAlignment="Bottom"
        HorizontalAlignment="Right"
        Grid.Row="0"  Grid.Column="0" />
   <TextBlock x:Name="Title"
        Text="?"
```

```
    VerticalAlignment="Bottom"

    HorizontalAlignment="Left"

    Grid.Row="0" Grid.Column="1" />

  <TextBlock x:Name="AuthorPrompt" Text="Author: "

    VerticalAlignment="Bottom"

    HorizontalAlignment="Right"

    Grid.Row="1"  Grid.Column="0" />

 <TextBlock x:Name="Author"

    Text="?"
    VerticalAlignment="Bottom"

    HorizontalAlignment="Left"

    Grid.Row="1" Grid.Column="1" />

<TextBlock x:Name="ChapterPrompt" Text="Chapters:  "

    VerticalAlignment="Bottom"

    HorizontalAlignment="Right"

    Grid.Row="2" Grid.Column="0"  />

  <ListBox x:Name="Chapters"
```

```
    ItemsSource="?"

    VerticalAlignment="Bottom"

    HorizontalAlignment="Left"

    Height="60" Width="200"

    Grid.Row="2" Grid.Column="1" />

<TextBlock x:Name="MultipleAuthorPrompt"

    Text="Multiple authors?:  "

     VerticalAlignment="Bottom"

    HorizontalAlignment="Right"

    Grid.Row="3" Grid.Column="0"  />

<CheckBox x:Name="MultipleAuthor"

    IsChecked="?"

    VerticalAlignment="Bottom"

    HorizontalAlignment="Left"

    Grid.Row="3" Grid.Column="1"/>

<TextBlock x:Name="QOHPrompt"

Text="Quantity On Hand:  "
```

```
      VerticalAlignment="Bottom"

      HorizontalAlignment="Right"

      Grid.Row="4" Grid.Column="0" />

      <TextBox x:Name="QuantityOnHand"

         Text="?"
        VerticalAlignment="Bottom"

         HorizontalAlignment="Left"

         Height="30" Width="90"

         Grid.Row="4" Grid.Column="1" />

   </Grid>
</UserControl>
```

بایند کردن TextBlocks

برای راحتی کار تقید را با دو بلاک متنی شروع میکنیم . الگویی برای تقید یک بلاک متنی به نام و خاصیت وجود دارد . این الگوها قادر به تعیین نوع تقید نیز خواهند بود . پیش فرض تقید OneWay میباشد. یک تمرین خوب برنامه نویسی ساخت یک استثناء است . این تقید روی میدان متنی است و با { } در برنامه مورد استفاده قرار میگیرد .

```
Text="{Binding Title, Mode=OneWay }"
```

در نتیجه دو ردیف اول مانند این کدها خواهد بود .

```
<TextBlock x:Name="TitlePrompt" Text="Title:  "
    VerticalAlignment="Bottom"
    HorizontalAlignment="Right"
    Grid.Row="0"  Grid.Column="0" />
<TextBlock x:Name="Title"
    Text="{Binding Title, Mode=OneWay }"
    VerticalAlignment="Bottom"
    HorizontalAlignment="Left"
    Grid.Row="0" Grid.Column="1" />


 <TextBlock x:Name="AuthorPrompt" Text="Author: "
    VerticalAlignment="Bottom"
    HorizontalAlignment="Right"
    Grid.Row="1"  Grid.Column="0" />
<TextBlock x:Name="Author"
    Text="{Binding Author, Mode=OneWay }"
    VerticalAlignment="Bottom"
    HorizontalAlignment="Left"
    Grid.Row="1" Grid.Column="1" />
```

هنوز در مورد خاصیت های اشیاء عنوان و نام نویسنده و تقید آنها چیزی نمیدانیم اما باید اشیاء را با خاصیت های آنها در اختیار داشته باشیم . این موضوع برای اشیاء دیگر نیز مصداق دارد .ListBox در حال عادی به یک ItemSource مقید میشود . اما ما انتظار داریم به یک مجموعه مقید گردد .

```
<ListBox x:Name="Chapters"
   ItemsSource="{Binding  Chapters, Mode=OneWay}"
   VerticalAlignment="Bottom"
   HorizontalAlignment="Left"
   Height="60" Width="200"
   Grid.Row="2" Grid.Column="1" />
```

ChechBox نیز به یک مقدار Boolean مقید است اما میتوان آن را به یک شی ء تقید داد .

```
<CheckBox x:Name="MultipleAuthor"
  IsChecked="{Binding MultipleAuthor, Mode=TwoWay}"
   VerticalAlignment="Bottom"
   HorizontalAlignment="Left"
   Grid.Row="3" Grid.Column="1"/>
```

DataContext

در زمان طراحی , اتصال داده ها روی اهداف نیاز است ، اما ممکن است فقط تقید را روی بعضی از عناوین کتاب ها بخواهید داشته باشید . DataContext یک کتاب مشخص است که در زمان اجرا انتخاب میشود و به خاصیت DataContext در عناصر چارچوب کاری تخصیص داده میشود . به عبارت خلاصه تر عناوین از عنصر جاری This استفاده میکنند .

DataContext میتواند چند ردیف داده داشته باشد اما اغلب یک شی از نوع تقید تخصیص داده میشود .یک شیء تقید میداند که چگونه داده هایی را که نیاز دارد توسط هدف از منبع بگیرد. اتصال عمومی با مد سوئیچ در شیء تقید مشخص میشود .

ویژگی دیگر استفاده از DataContext این است که اجازه میدهد که عناصر داده ای منبع از پدرشان ارث ببرند. به طور مثال میتوانید منبع داده ای برای یک Grid و تمام کنترل های آزاد روی آن برای استفاده DataContext بدون اینکه متد Set داشته باشید ،تنظیم کنید .

شما میتوانید این خط کد را به جای این کدها جایگزین نمایید .

```
Title.DataContext = currentBook;

Author.DataContext = currentBook;

Chapters.DataContext = currentBook;

MultipleAuthor.DataContext = currentBook;

QuantityOnHand.DataContext = currentBook;

LayoutRoot.DataContext = currentBook;
```

مراحل منطقی تقید داده

1- یک شی هدف بسازید و خاصیتی را که میخواهید مقید شود مشخص نمایید .
2- منبع و خاصیتی از منبع را که قرار است مقدار آن به هدف مقید شود ، مشخص نمایید .
3- هدف را به خاصیت مورد استفاده در DataContext منبع نگاشت دهید .

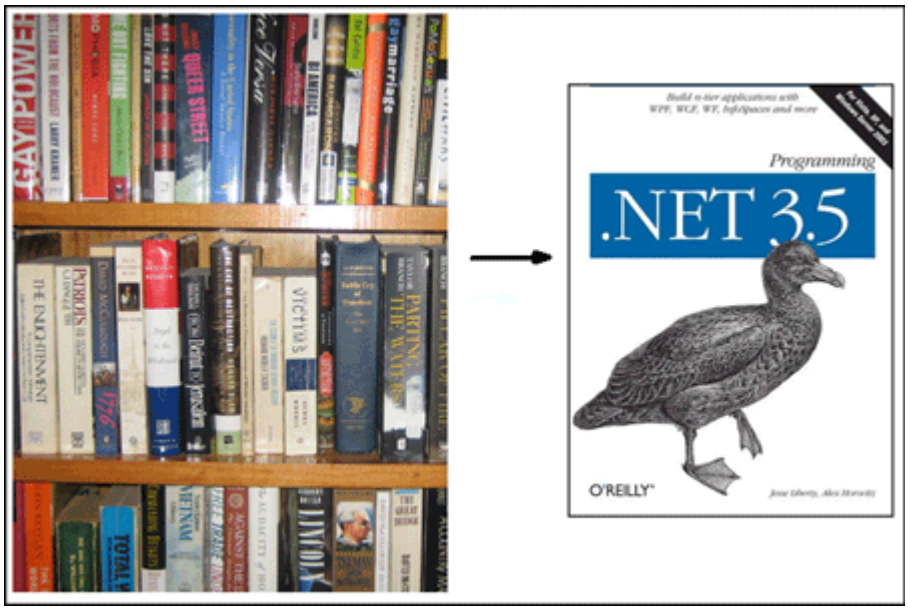
این کار خیلی آسانتر از آنچه میشنویداست . بخصوص این که زمانش بسیار کم است .
میخواهیم در برنامه سیلورلایت قادر باشیم تا هر Book ی راکه ممکن است انتخاب شده باشد با تمام جزئیات نمایش بدهیم . اتصال داده ها ممکن است خصوصیات کتا ب را تنظیم و در زمان اجرا از آنها استفاده نماید .
برای دیدن این خاصیت یک دکمه به برنامه اضافه نمایید . بین نمایش دو کتاب مختلف به تغییرات دقت نمایید .

```
<Button x:Name="Change"

Content="Change Book"

Height="30" Width="80"

HorizontalAlignment="Right"

Grid.Row="5" Grid.Column="0"  />
```

چیزی که مشاهده میشود، در مورد دکمه چیز خاصی نیست بلکه این کد جادویی است ! این جادوی عجیب در چند مرحله کوچک پیاده سازی می شود. ابتدا درون یک متغیر عضو ، یک مرجع به Book بگذارید وسپس سه مقدار جدید عضو از آن بسازید .

```
private Book b1;
private Book b2;
private Book currentBook;
```

باید دو کتاب را در حافظه که اولین آنها با نام b1 ارجاع داده میشود و دومی با نام b2 را مقدار دهی اولیه نمایید و یک کتاب جاری نیز داشته باشید که بین این دو کتاب در جریان است .
فرضیه مقدم در اینجا یک سیستم واقعی برای برداشتن یک کتاب از بین چند کتاب است توجه کنید برداشتن یک کتاب نه دو کتاب.



شکل 4- استخراج یک کتاب از کتابخانه

هنگامی که برنامه شروع میشودهر کتاب را میتوان در یک مرحله مقدار دهی کرد، و سپس بین آنها مبادله کرد.

```
b1 = new Book();

InitializeBleak(b1);

currentBook = b2 = new Book()

InitializeProgramming(b2);
```

```
private void InitializeProgramming(Book b)
 {
    b.Title = "Programming Silverlight";

    b.Author = "Jesse Liberty, Tim Heuer";

    b.MultipleAuthor = true;
```

```
    b.QuantityOnHand = 20;

    b.Chapters = new List<string>()

        { "Introduction", "Controls", "Events", "Styles" };
}

private void InitializeBleak(Book b)
{
    b.Title = "Bleak House";

    b.Author = "Charles Dickens";

    b.MultipleAuthor = false;


    b.QuantityOnHand = 150;

    b.Chapters = new List<string>()
    {
        "In Chancery",

        "In Fashion",

        "A Progress",

        "Telescoopic Philanthropy",

        "A Morning Adventure",

        "Quite at Home",

        "The Ghosts Walk",
```

```
        "Covering Sins",
        "Signs and Tokens",
        "The Law Writer"
    };
}
```

TextBox

در یک CheckBox اضافه شده فیلد Read/write اضافه مینماییم، جعبه متن مخصوص نمایش است و اجازه میدهد کاربر تعداد کپی های کتاب های در دسترس را به روز کند.

```
<TextBox x:Name="QuantityOnHand"
    Text="{Binding QuantityOnHand, Mode=TwoWay}"
    VerticalAlignment="Bottom"
    HorizontalAlignment="Left"
    Height="30" Width="90"
    Grid.Row="4" Grid.Column="1" />
```

در این طرح هر کتاب میداند چه تعداد کپی از آن نگهداری میشود بنابراین راه های دیگری برای طراحی نیز وجود دارد .

ListBox و بایند کردن آن به لیست

شی کتاب یک لیست از رشته ها را به عنوان فصل های کتاب در نظر می گیرد.

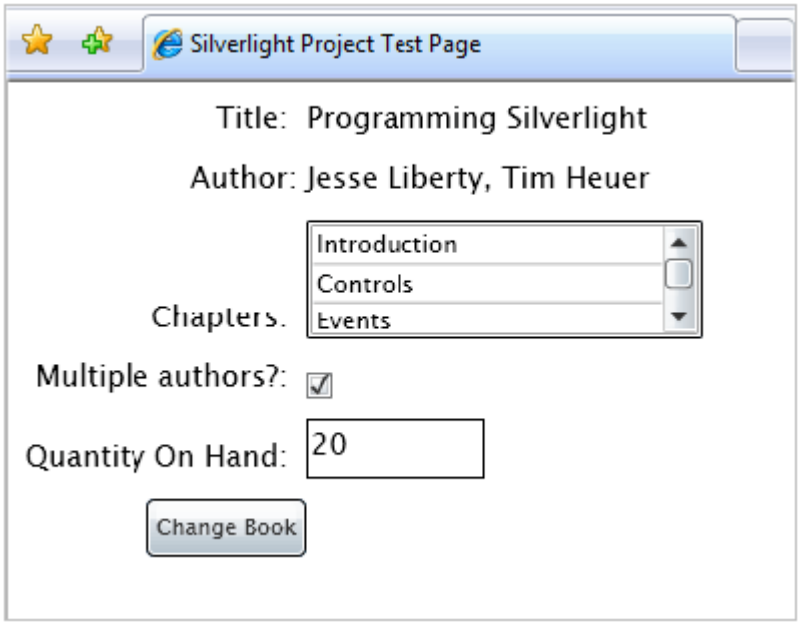
```
public List<string> Chapters
{
   get { return myChapters; }
   set
   {
      myChapters = value;
      NotifyPropertyChanged("Chapters");
   }
}
```

هنگامی که کتاب نمایش داده میشود، فصل های کتاب در یک ListBox در کنار آن نمایش داده میشود . هنگامی که یک کتاب جدید انتخاب میشود، فصل های کتاب جدید نمایش داده میشود. میتوانید متناوباً بین فصل های کتاب جدید رفته و یک ListItem جدید بسازید، اما باید به نکته مقابل این موضوع (مقیاس و خطاها) توجه کنید . در یک کلام تقید دادها یک راه حل رضایت بخش است .

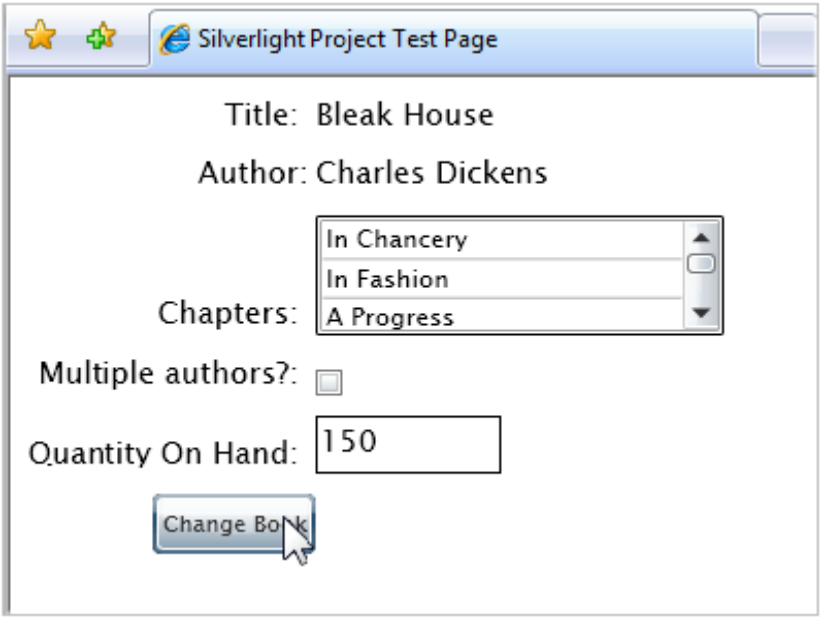
زمانیکه DtatSource در عنوان و نام نویسنده کتاب جاری به کار میرود لیست درست فصلها را خواهیم داشت. میتوانید خاصیت فصل کتاب را به خاصیت ItemSource جعبه لیست نسبت بدهید ، DtatSource که تغییر کند، فصل ها نیز به وضوح تغییر خواهند کرد .

```
<ListBox x:Name="Chapters"

   ItemsSource="{Binding  Chapters, Mode=OneWay}"

   VerticalAlignment="Bottom"

   HorizontalAlignment="Left"

   Height="60" Width="200"

   Grid.Row="2" Grid.Column="1" />
```

دو تصویرزیر نشان دهنده جزئیات جدید دو کتابی است که نمایش داده میشود .



شکل 5- اولین کتاب نمایش داده شده



شکل 6- دومین کتاب نمایش داده شده

کلمات کلیدی

اتصال داده ها - مدهای تقید -تقید شمارشی - تقید onTime - تقید One Way - تقید Twoway - لایه کاربر - لایه مجازی - لایه ماندگاری

Decoupling - INotifyPropertyChanged - System.ComponentModel - DataContext-