Silverlight 4

پژوهش و نگارش: وحید نصیری
Vahid_Nasiri@Yahoo.Com
بهار ۱۳۸۹

فهرست مطالب

فصل ۲۶ – استفاده از WCF RIA Services در Silverlight

مقدمه

Microsoft WCF RIA Services چارچوبی است برای توسعه‌ی ساده‌تر و سریعتر برنامه‌های تجاری مبتنی بر Silverlight (و البته محدود به Silverlight هم نیست). در طی فصل‌های قبل با اصول کاری Web Services، Validation ، Data binding و غیره به شکلی مجزا از هم آشنا شدیم. RIA Services این عناصر را با هم تلفیق کرده و ساخت برنامه‌های N-tier مبتنی بر Silverlight را با ارائه‌ی چارچوبی یکپارچه از سرویس‌های سمت سرور (در یک برنامه‌ی ASP.NET) و کنترل‌های سمت مشتری سهولت می‌بخشد. نگارش نهایی یک این محصول را از آدرس ذیل می‌توانید دریافت کنید:

http://go.microsoft.com/fwlink/?LinkID=169231

البته اگر Silverlight 4 tools را پیشتر نصب کرده باشید (مطابق توضیحات ابتدای کتاب)، این مجموعه به همراه WCF RIA Services ارائه شده است. به علاوه اگر قصد کار با LINQ to SQL را داشته باشید، حتما نیاز به WCF RIA Services Toolkit خواهید داشت:

http://go.microsoft.com/fwlink/?LinkId=185121

همچنین مجموعه‌ای از مثال‌های مرتبط با این فناوری از آدرس بعد قابل دریافت هستند:

http://code.msdn.microsoft.com/RiaServices

مستندات کامل WCF RIA Services را جهت مطالعه‌ی Offline می‌توانید از آدرس ذیل دریافت کنید:

http://tinyurl.com/24bt2mz

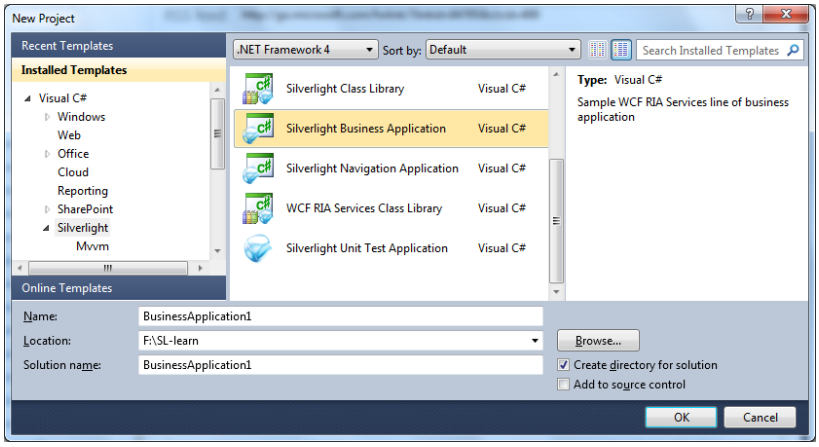
و اگر پیشنهادی جهت بهتر شدن این پروژه دارید می‌توانید نظرات خود را در آدرس ذیل در اختیار تیم Silverlight قرار دهید:

http://riaservices.mswish.net/

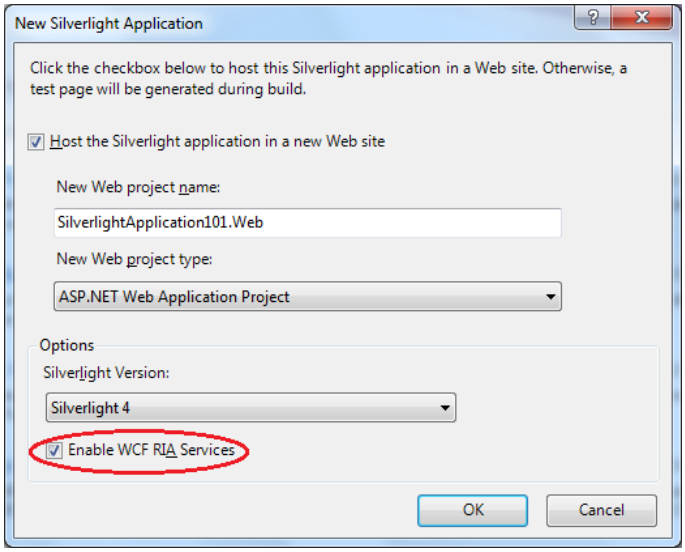
آغاز یک پروژه‌ی مبتنی بر Microsoft WCF RIA Services

پس از نصب مقدمات ذکر شده، حین آغاز یک پروژه‌ی جدید Silverlight در VS.NET 2010 با یک قالب جدید و همچنین یک گزینه‌ی جدید در مورد Microsoft WCF RIA Services مواجه خواهیم شد (شکل‌های ۱ و ۲). قالب جدید برنامه‌های تجاری Silverlight ، کار تنظیم خودکار ارجاعات مورد نیاز را به همراه ایجاد برنامه‌ای

که از امکانات Navigation framework استفاده می‌کند و سرویس‌های اعتبار سنجی کاربران را نیز به همراه دارد، انجام می‌دهد. روش دیگر فعال سازی WCF RIA Services، انتخاب گزینه‌ی مرتبط با آن (شکل ۲) در حین آغاز یک پروژه‌ی معمولی و متداول Silverlight است.



شکل ۱– قالب جدید برنامه‌های تجاری Silverlight در VS.NET 2010 .



شکل ۲– فعال سازی WCF RIA Services در یک پروژه‌ی جدید Silverlight .

اگر نیاز است به پروژه‌ی فعلی خود یک WCF RIA Service را معرفی نمائید تنها کافی است به برگه‌ی خواص پروژه‌ی Silverlight مراجعه کرده و همانند شکل ۳، پروژه‌ی مورد نظر را انتخاب کنید.

شکل ۳– مشخص سازی پروژه‌ی WCF RIA Services یک برنامه‌ی Silverlight.

در طی فصل جاری از روش دوم استفاده کرده (آغاز یک پروژه‌ی ساده Silverlight به همراه انتخاب گزینه‌ی فعال سازی WCF RIA Services) تا با جزئیات بیشتری از عملیات آشنا شویم. در این مثال از بانک اطلاعاتی معروف Northwind استفاده خواهیم نمود. ابتدا با کمک Entity framework تعدادی رکورد را از این بانک اطلاعاتی دریافت کرده و در سمت کاربر نمایش خواهیم داد. سپس امکانات ویرایش رکوردها را فراهم خواهیم نمود. بانک اطلاعاتی Northwind را از آدرس ذیل می‌توانید دریافت کنید:

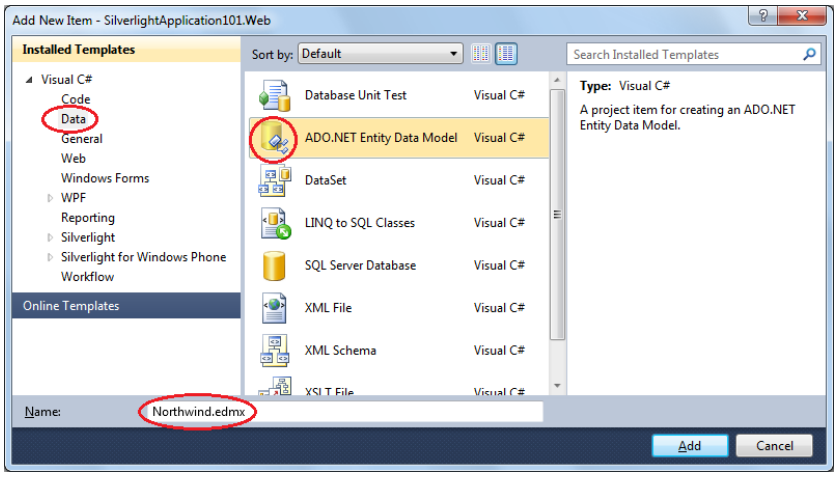
http://tinyurl.com/7vcda

پس از دریافت فایل‌های آن می‌توانید به سادگی با استفاده از اجرای دستور T-SQL بعد فایل‌های mdf. و ldf. آن‌را در SQL Server استفاده نمائید:

T-SQL

```
exec sp_attach_db @dbname=N'Northwind',
                  @filename1=N'C:\path\northwind.mdf',
                  @filename2=N'C:\path\northwind.ldf'
```

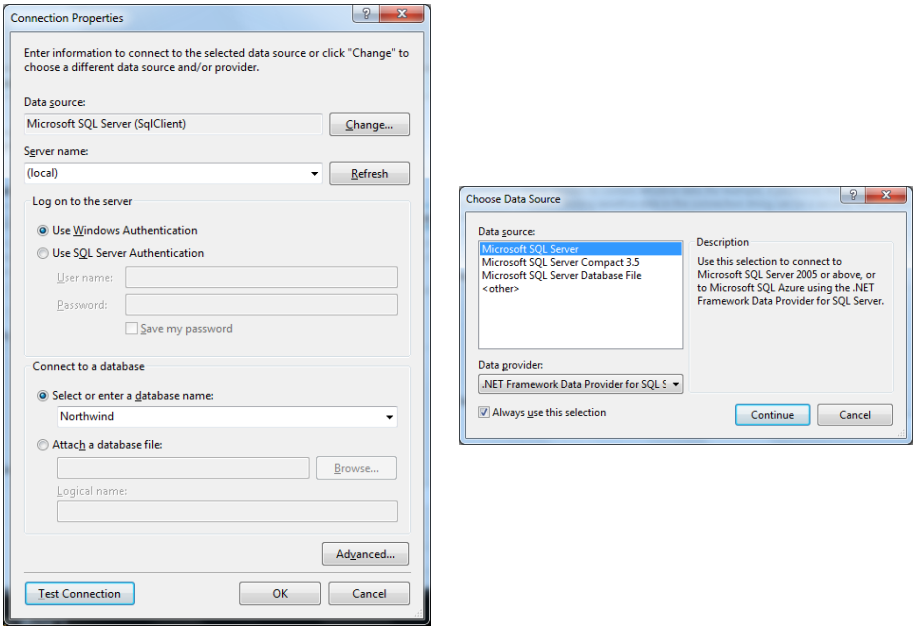
سپس به پروژه‌ی ASP.NET جاری مراجعه کرده و از منوی پروژه، گزینه‌ی افزودن آیتم جدید، یک ADO.NET Entity Data Model را به نام NorthwindDataModel اضافه کنید (شکل ۴). در صفحه‌ی بعد گزینه‌ی Generate from database را انتخاب کرده و بر روی دکمه‌ی بعد کلیک نمائید.



شکل ۴– افزودن یک ADO.NET Entity Data Model جدید به پروژه ASP.NET .

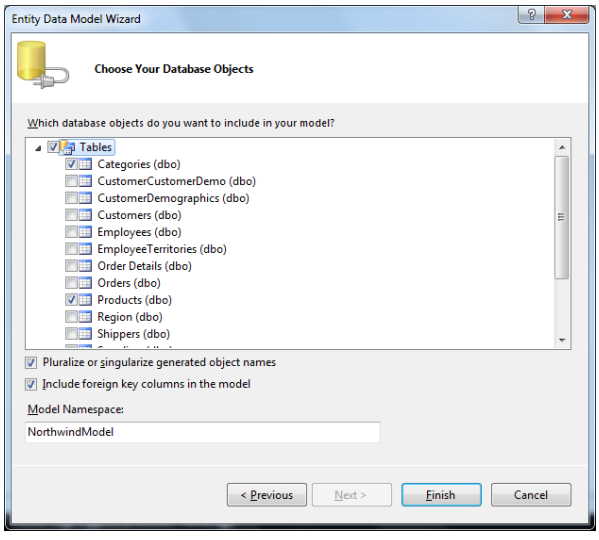
در ادامه نیاز خواهد بود تا یک Connection string را جهت اتصال به بانک اطلاعاتی تعریف نمائیم یا از نمونه‌های ذخیره شده قبلی استفاده کنیم. اگر رشته اتصالی از قبل جهت انتخاب وجود ندارد، ابتدا بر روی دکمه‌ی New connection کلیک کرده، سپس در صفحه‌ی باز شده (شکل ۵)، گزینه‌ی Microsoft SQL Server را انتخاب نموده و بر روی دکمه‌ی ادامه کلیک کنید. در صفحه‌ی بعد نام Server را وارد کرده (برای مثال (local))

و نوع اعتبار سنجی مورد نظر را نیز مشخص کنید (برای مثال حالت اعتبار سنجی یکپارچه با ویندوز که در محیط یک Intranet بسیار مناسب است) و سپس بانک اطلاعاتی Northwind را از لیست مربوطه انتخاب کنید.



شکل ۵– ایجاد رشته اتصالی به بانک اطلاعاتی Northwind .

اکنون با استفاد از این رشته‌ی اتصالی ساخته شده می‌توان از صفحه‌ی مشخص سازی Connection string به مرحله‌ی بعد عبور کرد. در صفحه‌ی بعد دو جدول گروه‌ها و محصولات را انتخاب خواهیم نمود (شکل ۶) و در آخر بر روی دکمه‌ی Finish کلیک نمائید.

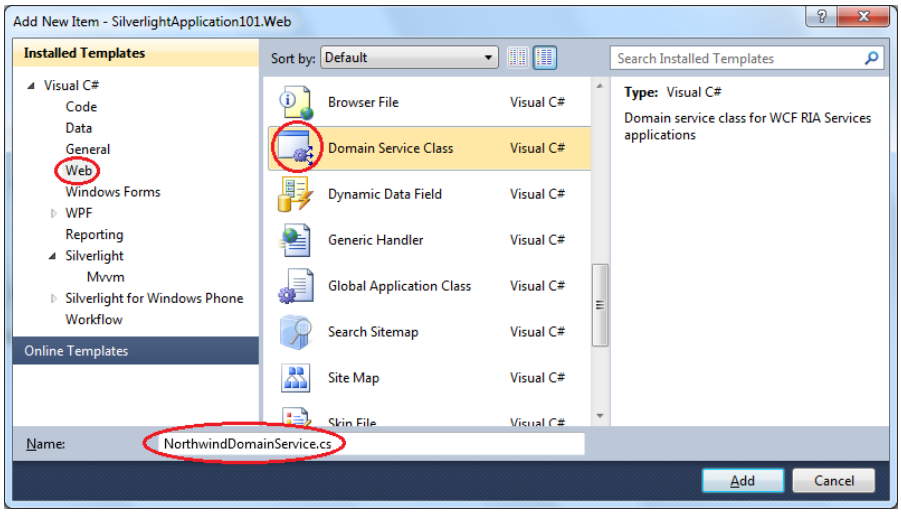


شکل ۶– انتخاب جداول گروه‌ها و محصولات از بانک اطلاعاتی Northwind.

اگر به شکل ۶ دقت نمائید در Entity framework 4 به صورت خودکار اسامی جمع به مفرد تبدیل خواهند شد و این مورد در حین کارکردن با موجودیت‌ها در طراح آن‌ها کار را ساده‌تر می‌کند؛ زیرا مرسوم است که از اسامی مفرد بجای اسامی جمع متناظر با جداول، جهت نامگذاری موجودیت‌ها استفاده گردد.

پس از اینکار یکبار پروژه را Compile نمائید، در غیر اینصورت در قسمت بعد امکان انتخاب موجودیت‌ها را نخواهیم داشت.

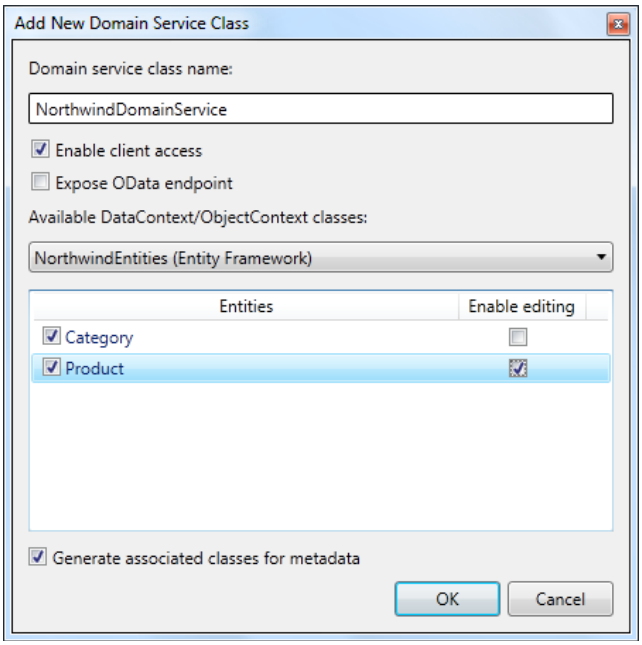
اکنون یک پوشه‌ی جدید را به نام Services به برنامه‌ی ASP.NET خود اضافه نمائید. به این پوشه قصد داریم یک Domain service class را مطابق شکل ۷ اضافه نمائیم (قالب جدید آن به همراه WCF RIA Services نصب گردیده است). نام این کلاس را NorthwindDomainService وارد نمائید. درصفحه‌ی بعد (شکل ۸) اگر لیست موجودیت‌های آن خالی می‌باشد به این معنا است که هنوز پروژه را یکبار Compile نکرده‌اید. در این صفحه هر دو موجودیت را انتخاب نمائید. در این مثال قصد داریم تنها امکان ویرایش محصولات را فراهم آوریم. به همین جهت گزینه‌ی Enable editing آن‌را نیز مطابق شکل ۸ انتخاب نمائید. به این صورت متدهای متناظر انجام اینکار به صورت خودکار تهیه خواهند شد. پس از کلیک بر روی OK دو کلاس جدید به برنامه اضافه خواهند شد. علت ایجاد کلاس metadata ، امکان افزودن ویژگی‌های اعتبار سنجی به آن در ادامه‌ی توضیحات فصل جاری است.



شکل ۷– افزودن یک Domain service class جدید به پروژه.

اگر به کدهای کلاس تولید شده‌ی NorthwindDomainService دقت نمائیم، تنها امکان انتخاب اطلاعات جدول گروه‌ها به صورت خودکار اضافه شده است؛ اما از آنجائیکه پیشتر امکان ویرایش اطلاعات جدول محصولات را نیز انتخاب کرده بودیم، بنابراین متدهای متناظر افزودن، ویرایش و حذف اطلاعات آن نیز فراهم شده‌اند.

فضای نام کلاس‌های اضافه شده SilverlightApplicationXYZ.Web.Services نام دارد که پس از Compile پروژه به سادگی و بدون نیاز به هیچ نوع تمهیدات دیگری در برنامه‌ی Silverlight قابل استفاده خواهد بود.



شکل ۸– صفحه‌ی تنظیمات یک Domain service class جدید.

اکنون به پروژه‌ی Silverlight جاری رجوع کرده و یک کنترل DataGrid را از جعبه ابزار آن کشیده و بر روی فرم رها کنید تا ارجاعات لازم به آن به صورت خودکار به پروژه اضافه شوند. کدهای XAML صفحه‌ی اصلی پروژه اکنون به صورت زیر خواهند بود:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication101.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
    "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400"
   xmlns:
   sdk="http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk">
    <Grid x:Name="LayoutRoot" Background="White">
        <sdk:DataGrid  Name="dataGrid1"  />
    </Grid>
</UserControl>
```

سپس فقط با استفاده از سه سطر کد نویسی در متد mainPageLoaded ، اطلاعات WCF RIA Service خود را دریافت و توسط کنترل DataGrid نمایش خواهیم داد:

MainPage.xaml.cs

```
using System.Windows;
using SilverlightApplication101.Web.Services;

namespace SilverlightApplication101
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
            this.Loaded += mainPageLoaded;
        }

        void mainPageLoaded(object sender, RoutedEventArgs e)
        {
            var ctx = new NorthwindDomainContext();
            ctx.Load(ctx.GetProductsQuery());
            dataGrid1.ItemsSource = ctx.Products;
        }
    }
}
```

NorthwindDomainContext تعریف شده در پروژه‌ی ASP.NET تنها پس از افزودن سطر فضای نام متناظر با آن به کلاس جاری، قابل استفاده خواهد بود. از این طریق می‌توان به کلاس‌ها و متدهای این WCF RIA Service دسترسی داشت.

این نوع پروژه‌ها کاندیدای مناسب جهت انتخاب کاهش حجم فایل XAP نهایی با استفاده از ویژگی library caching می‌باشند که در مورد آن در طی فصول قبل بیشتر توضیح داده شد.

تا اینجا با اصول مقدماتی کار با WCF RIA Services آشنا شدیم. در ادامه، مباحث تکمیلی آن‌را مرور خواهیم نمود.

معرفی کنترل DomainDataSource

پس از نصب مجموعه‌ی WCF RIA Services ، کنترل جدیدی به نام DomainDataSource به جعبه ابزار VS.NET اضافه خواهد شد که از آن جهت تامین منبع داده‌ای نمایش و کار با اطلاعات استفاده خواهیم کرد. این کنترل را از جعبه ابزار VS.NET کشیده و بر روی فرم رها کنید تا ارجاعات لازم به اسمبلی‌های آن و همچنین فضاهای نام متناظر با آن (با نام مستعار riaControls) به پروژه و صفحه‌ی جاری برنامه اضافه شوند.

اکنون قصد داریم بجای روش مثال قبل، بدون استفاده از کد نویسی، NorthwindDomainContext و کوئرهای مربوطه را در اختیار کنترل DataGrid قرار دهیم. به همین منظور متد mainPageLoaded مثال قبل را حذف کرده و کدهای XAML برنامه را به صورت بعد تغییر دهید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication101.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc=
    "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400"
    xmlns:
     sdk="http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk"
    xmlns:riaControls=
    "clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.DomainServices"
    xmlns:localServices=
        "clr-namespace:SilverlightApplication101.Web.Services"
    xmlns:my="clr-namespace:SilverlightApplication101.Web">
    <StackPanel>
        <riaControls:DomainDataSource
            Name="nwProducts"
            LoadSize="4"
            LoadDelay="0:0:0.25"
            PageSize="10"
            d:DesignData=
                "{d:DesignInstance my:Product, CreateList=true}"
            LoadedData="productDomainDataSource_LoadedData"
            QueryName="GetProductsQuery"
            AutoLoad="True">
            <riaControls:DomainDataSource.DomainContext>
                <localServices:NorthwindDomainContext />
            </riaControls:DomainDataSource.DomainContext>
        </riaControls:DomainDataSource>
        <sdk:DataGrid
            Name="dataGrid1"
            ItemsSource="{Binding Path=Data, ElementName=nwProducts}"
            Width="Auto"
            AutoGenerateColumns="True"
            Height="200"
            />
        <sdk:DataPager
            Source="{Binding Path=Data, ElementName=nwProducts}"
            PageSize="10"
            />
    </StackPanel>
</UserControl>
```

توضیحات:

فضاهای نام متناظر با کنترل‌های اضافه شده به صورت خودکار توسط VS.NET پس از کشیدن و رها کردن آن‌ها بر روی صفحه ایجاد شده‌اند. تنها دو فضای نام مرتبط با کلاس‌های WCF RIA Service به صورت دستی اضافه شده‌اند.

ابتدا نیاز است تا منبع داده مشخص شود؛ به همین جهت کنترل DomainDataSource تعریف شده است. خاصیت LoadSize آن مشخص می‌کند که در طی یک حلقه غیرهمزمان هربار چند رکورد از بانک اطلاعاتی دریافت و نمایش داده شود. فواصل زمانی این دریافت اطلاعات توسط خاصیت LoadDelay مشخص شده است. از خاصیت PageSize در ادامه جهت صفحه بندی اطلاعات DataGrid استفاده خواهیم کرد. ویژگی DesignData سبب نمایان شدن ستون‌های نهایی DataGrid خواهند شد. QueryName نیز متناظر است با نام متدهایی که در WCF RIA Service به صورت خودکار تولید شدند. در ادامه روش معرفی DomainContext را ملاحظه می‌فرمائید. پس از مشخص شدن منبع داده مورد استفاده، با کمک عملیات Binding آن‌را به کنترل‌های DataGrid و همچنین DataPager معرفی کرده‌ایم.

روال رخداد گردان LoadedData از این جهت اضافه شده است که اگر در حین بارگذاری اطلاعات خطایی رخ داد، بتوان مشکلات موجود را بهتر دریافت نمود.

MainPage.xaml.cs

```
using System.Windows;
using System.Windows.Controls;

namespace SilverlightApplication101
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void productDomainDataSource_LoadedData(object sender,
            LoadedDataEventArgs e)
        {
            if (e.HasError)
            {
                MessageBox.Show(e.Error.ToString(),
                        "Load Error", MessageBoxButton.OK);
                e.MarkErrorAsHandled();
            }
        }
    }
}
```

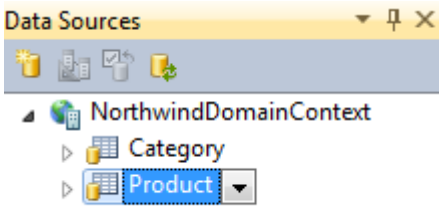
برای صفحه بندی نیاز است تا خروجی کوئری LINQ ما مرتب شده باشد. به همین جهت به پروژه‌ی ASP.NET مراجعه کرده و تغییر ذیل را اعمال نمائید:

NorthwindDomainService.cs

```
public IQueryable<Product> GetProducts()
{
  return this.ObjectContext.Products.OrderBy(a=>a.ProductName);
}
```

استفاده از امکانات جدید VS.NET 2010 برای تولید خودکار منبع داده

تمام این عملیات را به صورت خودکار نیز می‌توان انجام داد. این مورد جزو ویژگی‌های جدید VS.NET 2010 است. برای این منظور به برگه‌ی Data Sources (شکل ۹) رجوع کرده (منوی Data گزینه‌ی Show Data Sources)، اندکی صبر کنید تا به صورت خودکار اطلاعات موجودیت‌ها تشخیص داده شوند.



شکل ۹– برگه‌ی Data Sources در VS.NET 2010 .

سپس موجودیت Product را از برگه‌ی Data Sources کشیده و بر روی فرم برنامه رها کنید. به صورت خودکار دو کنترل DomainDataSource و DataGrid به صفحه اضافه شده و همچنین تنظیمات ابتدایی آن‌ها نیز انجام خواهد شد. DataGrid حاصل نیز بسیار شکیل‌تر بوده و قالب ستون‌های آن به صورت خودکار تشکیل شده‌اند. حتی کدهای Binding مربوط به DataPager را نیز می‌توان با کشیدن و رها کردن موجودیت Product بر روی کنترل DataPager به صورت خودکار ایجاد نمود.

نمایش جزئیات هر ردیف انتخاب شده در DataGrid

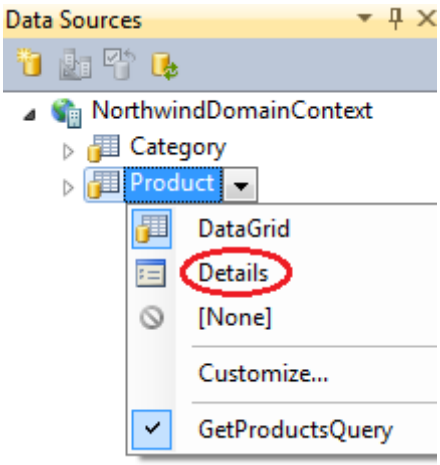
در ادامه‌ی مثال قبل، قصد داریم پس از انتخاب هر ردیف در DataGrid ، جزئیات فیلدهای آن‌را در پایین صفحه نمایش دهیم. برای سهولت کار، از امکانات جدید VS.NET 2010 استفاده خواهیم کرد. به برگه‌ی Data Sources مراجعه کرده و بر روی موجودیت Product کلیک کنید (شکل ۱۰). حالت پیش فرض آن، DataGrid

است که پیشتر در مورد آن توضیح داده شد. اکنون ابتدا حالت Details را انتخاب کنید، سپس موجودیت Product را کشیده و بر روی فرم رها نمائید. به این صورت یک Grid نمایانگر جزئیات تک تک فیلدهای موجودیت Product به صفحه اضافه می‌شود. بدیهی است کدهای XAML این Grid را به سادگی می‌توان ویرایش کرد و تغییرات دلخواه را به آن اعمال نمود. برای مثال از آنجائیکه می‌خواهیم جزئیات ردیف انتخاب شده در DataGrid را در این Grid جدید مشاهده کنیم، تغییر ذیل را به این Grid جدید اعمال کنید:

MainPage.xaml

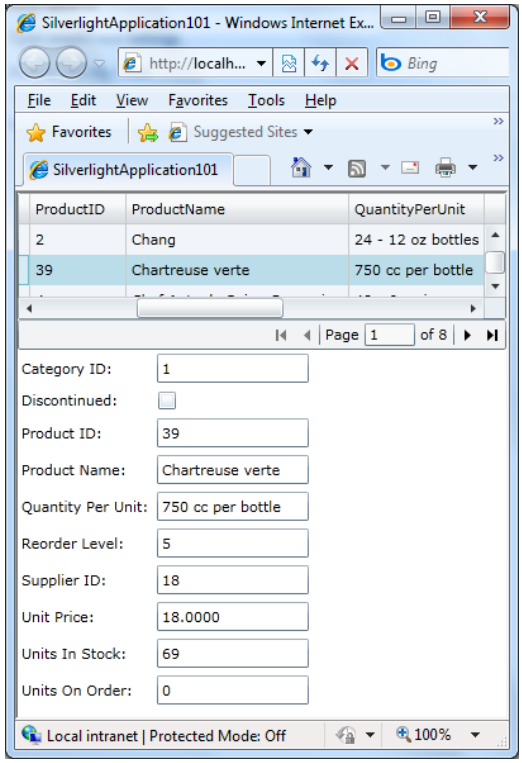
```
...
<Grid DataContext="{Binding ElementName=dataGrid1, Path=SelectedItem}"
...
```

نتیجه‌ی حاصل را در شکل ۱۱ می‌توانید مشاهده نمائید.

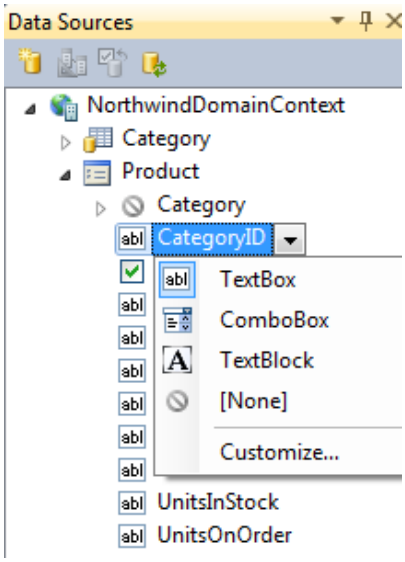


شکل ۱۰– امکان انتخاب جزئیات فیلدهای یک کلاس جهت نمایش

شاید از کنترل‌های پیش فرضی که در این حالت تولید شدند راضی نباشید. برای مثال علاقمند باشید تعدادی را به صورت TextBlock نمایش دهید و غیره. دو راه برای انجام اینکار وجود دارد. ویرایش دستی کدهای XAML حاصل و یا مجددا به برگه‌ی Data Sources مراجعه کنید؛ اینبار گزینه‌ی Product را گشوده (شکل ۱۲) و فیلدهای مورد را نظر را انتخاب کنید. با انتخاب هر فیلد می‌توان کنترل پیش فرض را تغییر داد (اگر کنترل مورد نظر شما در منو نمایش داده نشده است بر روی گزینه‌ی Customize کلیک کرده و کنترل دلخواه خود را انتخاب نمائید). اکنون اگر مجددا موجودیت Product را کشیده و بر روی فرم رها کنید، از کنترل‌های جدید انتخابی استفاده خواهد شد.



شکل ۱۱– نمایی از نمایش جزئیات یک ردیف انتخاب شده در DataGrid .



شکل ۱۲– امکان تغییر کنترل‌های پیش فرض نمایش داده شده در حالت انتخاب Details .

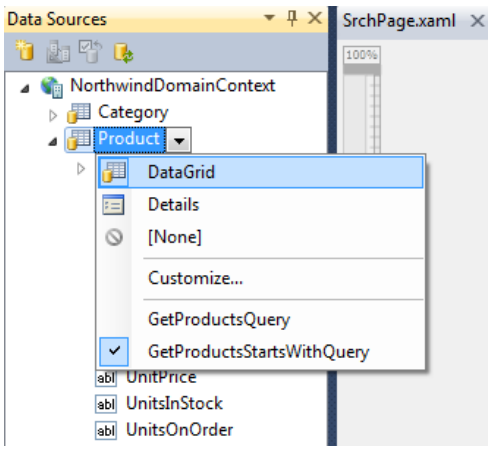
جستجو بر روی اطلاعات دریافتی از بانک اطلاعاتی

فرض کنید در مثال قبل قصد داریم امکانات جستجو بر اساس نام محصولات را نیز اضافه کنیم. به همین منظور ابتدا به پروژه‌ی ASP.NET مراجعه کرده و کوئری LINQ آن‌را مطابق کدهای بعد به کلاس NorthwindDomainService اضافه خواهیم نمود. پس از افزودن متد GetProductsStartsWith ، Compile پروژه را فراموش نکنید.

NorthwindDomainService.cs

```
public IQueryable<Product> GetProductsStartsWith(string name)
{
   return this.ObjectContext.Products.Where(
                          a => a.ProductName.StartsWith(name));
}
```

اکنون اگر به پروژه‌ی Silverlight رجوع کرده و مجددا برگه‌ی Data Sources را باز نمائیم ، می‌توان کوئری LINQ جدید را مشاهده نمود (شکل ۱۳). ابتدا حالت انتخابی را بر روی DataGrid قرار دهید، سپس کوئری GetProductsStartsWithQuery را انتخاب نمائید. حال می‌توان مجددا موجودیت Product را از این برگه کشیده و بر روی یک فرم جدید برنامه رها کرد. این بار به صورت خودکار قسمت جستجوی برنامه نیز تشکیل خواهد شد (شکل ۱۴).



شکل ۱۳– امکان انتخاب کوئری LINQ جدید به عنوان منبع داده DataGrid .

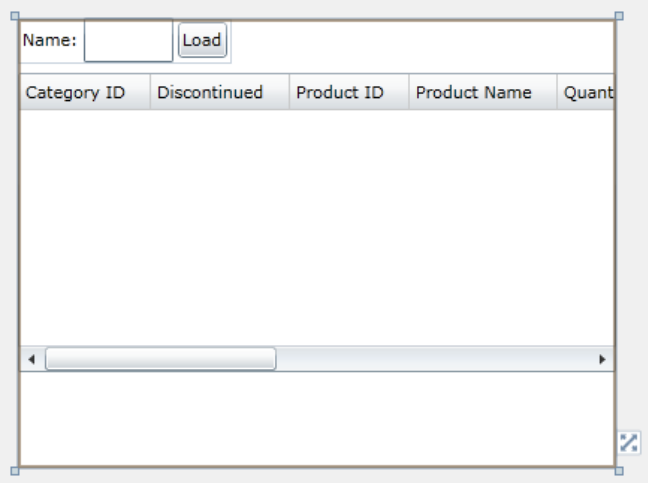
این مثال از لحاظ نحوه‌ی تعریف و ارسال پارامتر به کنترل DomainDataSource حائز اهمیت است:

SrchPage.xaml

```
<riaControls:DomainDataSource
     AutoLoad="False"
```

```
            d:DesignData="{d:DesignInstance my:Product, CreateList=true}"
    Height="0"
      LoadedData="productDomainDataSource_LoadedData"
      Name="productDomainDataSource"
      QueryName="GetProductsStartsWithQuery"
      Width="0">
    <riaControls:DomainDataSource.DomainContext>
       <my1:NorthwindDomainContext />
    </riaControls:DomainDataSource.DomainContext>
    <riaControls:DomainDataSource.QueryParameters>
         <riaControls:Parameter
            ParameterName="name"
            Value="{Binding ElementName=nameTextBox, Path=Text}" />
    </riaControls:DomainDataSource.QueryParameters>
</riaControls:DomainDataSource>
```

در اینجا روش تعریف یک پارامتر جدید و همچنین دریافت اطلاعات آن‌را از یک کنترل TextBox قرار گرفته بر روی صفحه می‌توان ملاحظه نمود.



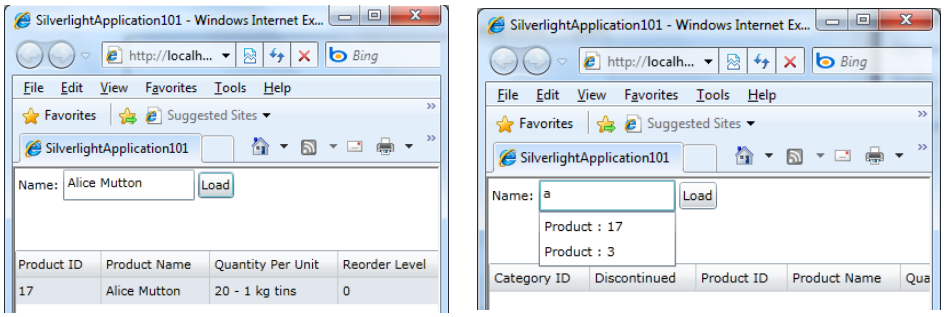
شکل ۱۴– تشکیل خودکار فرم جستجوی اطلاعات در بانک اطلاعاتی

در این مثال فرض کنید بجای یک TextBox ساده نیاز است تا از یک AutoCompleteBox استفاده گردد (شکل ۱۵). برای این منظور یک کنترل DomainDataSource را بر اساس لیست محصولات تشکیل داده و اطلاعات آن‌را در اختیار کنترل AutoCompleteBox خواهیم گذاشت:

SrchPage2.xaml

```
...
        <riaControls:DomainDataSource
            AutoLoad="True"
            d:DesignData=
            "{d:DesignInstance my:Product, CreateList=true}"
```

```
            Height="0"
            LoadedData="productDomainDataSource_LoadedData"
            Name="productDomainDataSource"
            QueryName="GetProductsQuery"
            Width="0">
            <riaControls:DomainDataSource.DomainContext>
                <my1:NorthwindDomainContext />
            </riaControls:DomainDataSource.DomainContext>
        </riaControls:DomainDataSource>
...
            <sdk:AutoCompleteBox
                Height="28"
                FilterMode="StartsWith"
                ItemsSource=
                  "{Binding Data, ElementName=productDomainDataSource}"
                ValueMemberBinding="{Binding ProductName}"
                Name="nameTextBox"
                Width="120" />
...
```



شکل ۱۵– استفاده از AutoCompleteBox به همراه کنترل DomainDataSource .

در این مثال با استفاده از تعریف یک کوئری LINQ جدید و سپس معرفی پارامتر متد مرتبط به آن توسط اشیاء QueryParameters ، توانستیم جستجوی خود را بر روی داده‌های بانک اطلاعاتی انجام دهیم. روش دیگری نیز برای انجام اینکار وجود دارد و آن استفاده از اشیاء FilterDescriptors می‌باشد:

SrchPage2.xaml

```
...
<riaControls:DomainDataSource
    ...
    <riaControls:DomainDataSource.FilterDescriptors>
         <riaControls:FilterDescriptor
             PropertyPath="ProductName"
             Operator="Contains"
             Value="{Binding ElementName=nameTextBox, Path=Text}"
           />
```

```
        </riaControls:DomainDataSource.FilterDescriptors>
    ...
</riaControls:DomainDataSource>
...
```

در اینجا خاصیت مورد نظر جهت جستجو مشخص می‌شود. سپس نحوه‌ی جستجو تعیین شده و در نهایت مقدار دریافتی باید دقیقا ذکر گردد؛ که می‌تواند مقداری ثابت باشد یا از طریق عملیات Binding دریافت شود. در مثال فوق به ازای هر تغییری که در مقادیر TextBox برنامه داده شود، نتیجه‌ی جستجو به صورت خودکار نمایش داده خواهد شد. بدیهی است مقدار خاصیت PropertyPath را نیز می‌توان از طریق Binding هم دریافت کرد. برای مثال لیستی از فیلدهای یک جدول در یک ComboBox ذکر شوند و سپس بر این اساس جستجو انجام گردد:

```
PropertyPath="{Binding ElementName=ComboBox1,
                Path=SelectedItem.Content}"
```

گروه بندی اطلاعات دریافتی از بانک اطلاعاتی

در طی فصل‌های قبل با استفاده از اشیاء PagedCollectionView موفق به گروه بندی اطلاعات و نمایش آن‌ها در کنترل DataGrid شدیم. در این فصل نیز گروه بندی اطلاعات توسط کنترل DomainDataSource پشتیبانی می‌شود. برای نمونه در مثال قبل، تعریف DomainDataSource را به صورت ذیل تغییر دهید:

SrchPage2.xaml

```
...
<riaControls:DomainDataSource
    ...
    <riaControls:DomainDataSource.GroupDescriptors>
        <riaControls:GroupDescriptor PropertyPath="CategoryID" />
    </riaControls:DomainDataSource.GroupDescriptors>
    ...
</riaControls:DomainDataSource>
...
```

همانطور که ملاحظه می‌نمائید برای گروه بندی اطلاعات باید از اشیاء GroupDescriptor کمک گرفت و سپس فیلد مورد نظر را در ویژگی PropertyPath آن ذکر کرد.

تعیین نحوه‌ی مرتب سازی پیش فرض اطلاعات

هرچند کنترل DataGrid به صورت پیش فرض و بدون کد نویسی خاصی امکانات مرتب سازی اطلاعات را در اختیار ما قرار می‌دهد اما شاید علاقمند باشیم که در اولین بار نمایش آن، اطلاعات بر اساس خاصیت یا

خاصیت‌های مشخصی مرتب شده باشند. برای این منظور یا می‌توان کوئری LINQ آن‌را تدارک دید (مانند مثال صفحه بندی اطلاعات) و یا از اشیاء SortDescriptors کنترل DomainDataSource استفاده کرد. در مثال بعد نحوه‌ی معرفی اشیاء SortDescriptors ، روش مشخص سازی فیلد مورد نظر و صعودی یا نزولی بودن مرتب سازی نیز بیان شده‌اند:

SrchPage2.xaml

```
...
<riaControls:DomainDataSource
    ...
     <riaControls:DomainDataSource.SortDescriptors>
          <riaControls:SortDescriptor
                    PropertyPath="ProductName"
                    Direction="Ascending" />
    </riaControls:DomainDataSource.SortDescriptors>
    ...
</riaControls:DomainDataSource>
...
```

انجام عملیات درج ، به روز رسانی و حذف اطلاعات بانک اطلاعاتی

در این قسمت می‌خواهیم در ادامه‌ی مثال کار با بانک اطلاعاتی Northwind ، با استفاده از امکانات WCF RIA Services ، اطلاعات دریافتی را در یک DataGrid نمایش داده و سپس هر ردیف انتخابی را به یک کنترل DataForm معرفی شده در فصل‌های قبل مقید نمائیم (سناریوی Master-Details). سپس اعمال متداول درج ، به روز رسانی و حذف اطلاعات بانک اطلاعاتی را پیاده سازی کنیم. مزیت استفاده از کنترل DataForm ، صفحه بندی رکوردهای دریافتی، یکپارچگی عملیات اعتبار سنجی و موارد دیگری است که در طی فصل اختصاص داده شده به آن بررسی گردیدند. البته همانطور که پیشتر نیز ذکر شد با استفاده از امکانات جدید VS.NET 2010 و مراجعه به حالت ویژه‌ی Details در برگه‌ی Data Sources ، به سادگی می‌توان با کشیدن و رها کردن یک موجودیت بر روی صفحه‌ی جاری برنامه، فرم متناظری از آن را تشکیل داد و سپس اعمال به روز رسانی بانک اطلاعاتی را بر این اساس تعریف کرد.

برای این منظور یک User control جدید را به نام CRUD.xaml به پروژه‌ی جاری اضافه کرده‌ایم. کدهای XAML آن‌را در ادامه ملاحظه خواهید نمود:

CRUD.xaml

```
<UserControl x:Class="SilverlightApplication101.CRUD"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
        xmlns:mc=
   "http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400"
  xmlns:riaControls=
"clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.DomainServices"
  xmlns:my="clr-namespace:SilverlightApplication101.Web"
  xmlns:my1="clr-namespace:SilverlightApplication101.Web.Services"
  xmlns:sdk=
    "http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk"
  xmlns:toolkit=
"http://schemas.microsoft.com/winfx/2006/xaml/presentation/toolkit">
  <toolkit:BusyIndicator
      Height="auto" Name="busyIndicator1" Width="310"
      IsBusy="{Binding Path=IsBusy,
                    ElementName=productDomainDataSource}"
      HorizontalAlignment="Left">
      <StackPanel>
          <riaControls:DomainDataSource
          AutoLoad="True"
          d:DesignData=
             "{d:DesignInstance my:Product, CreateList=true}"
          Height="0"
          LoadedData="productDomainDataSource_LoadedData"
          Name="productDomainDataSource"
          QueryName="GetProductsQuery"
          Width="0">
              <riaControls:DomainDataSource.DomainContext>
                  <my1:NorthwindDomainContext />
              </riaControls:DomainDataSource.DomainContext>
          </riaControls:DomainDataSource>
          <sdk:DataGrid
          AutoGenerateColumns="False"
          Height="150"
          HorizontalAlignment="Left"
          ItemsSource=
          "{Binding ElementName=productDomainDataSource, Path=Data}"
          Margin="5"
          Name="productDataGrid"
          RowDetailsVisibilityMode="VisibleWhenSelected"
          VerticalAlignment="Top"
          Width="300">
              <sdk:DataGrid.Columns>
                  <sdk:DataGridTextColumn
                  x:Name="productIDColumn"
                  Binding="{Binding Path=ProductID, Mode=OneWay}"
                  Header="Product ID"
                  IsReadOnly="True" Width="SizeToHeader" />
                  <sdk:DataGridTextColumn
                  x:Name="productNameColumn"
                  Binding="{Binding Path=ProductName}"
```

```
                    Header="Product Name" Width="SizeToHeader" />
                </sdk:DataGrid.Columns>
            </sdk:DataGrid>
            <sdk:DataPager
                Source=
                  "{Binding Data, ElementName=productDomainDataSource}"
                PageSize="5"
                />

            <toolkit:DataForm
            Header="Product Details"
            AutoCommit="True"
            AutoEdit="False"
            CommandButtonsVisibility="All"
            CurrentItem=
            "{Binding Path=SelectedItem, ElementName=productDataGrid}"
            ItemsSource=
            "{Binding Path=Data, ElementName=productDomainDataSource}"
            EditEnded="dataForm1_EditEnded"
            DeletingItem="dataForm1_DeletingItem"
            AddingNewItem="dataForm1_AddingNewItem"
            HorizontalAlignment="Left"
            Margin="5"
            Height="Auto"
            Name="dataForm1"
            Width="300"  />
        </StackPanel>
    </toolkit:BusyIndicator>
</UserControl>
```

توضیحات:

- کدهای مرتبط با کنترل DomainDataSource و DataGrid این صفحه به صورت خودکار با کشیدن و رها کردن موجودیت Product از برگه‌ی Data Sources بر روی صفحه‌ی جاری در VS.NET 2010 تولید شده‌اند که در مورد آن‌ها پیشتر توضیح داده شد.
- تمام کدهای صفحه را داخل یک کنترل BusyIndicator مقید شده به خاصیت IsBusy شیء DomainDataSource، قرار داده‌ایم. به این صورت در زمان بارگذاری اولیه صفحه یا ذخیره سازی اطلاعات در Server ، کاربر متوجه طول عملیات خواهد گردید.
- کنترل DataPager نیز جهت نمایش زیباتر و صفحه بندی اطلاعات DataGrid به صفحه اضافه شده است. همچنین خاصیت Source آن به خاصیت Data شیء productDomainDataSource مقید گردیده است.
- قسمت اصلی کار افزودن، ویرایش و حذف رکوردها توسط کنترل DataForm مدیریت خواهد شد. خاصیت AutoCommit آن‌را به True تنظیم کرده‌ایم تا پس از تغییرات و مراجعه به رکورد بعدی،

امکان ثبت خودکار آن‌ها فراهم گردد. همچنین یک سری روال رخدادگردان نیز برای مدیریت اعمال ویرایش ، ثبت و حذف اطلاعات اضافه شده است. خاصیت CurrentItem آن با ردیف جاری انتخاب شده در کنترل DataGrid مقدار دهی گردیده است. ویژگی ItemsSource آن اطلاعات خود را از کنترل productDomainDataSource دریافت می‌کند.

در ادامه کدهای متناظر با این View را مشاهده خواهید نمود:

CRUD.xaml.cs

```
using System.ComponentModel;
using System.Windows;
using System.Windows.Controls;

namespace SilverlightApplication101
{
    public partial class CRUD
    {
        public CRUD()
        {
            InitializeComponent();
            this.Loaded += crudLoaded;
        }

        void crudLoaded(object sender, RoutedEventArgs e)
        {
            productDomainDataSource.SubmittedChanges +=
                productDomainDataSource_SubmittedChanges;
        }

        static void productDomainDataSource_SubmittedChanges(
            object sender, SubmittedChangesEventArgs e)
        {
            if (e.HasError)
            {
                MessageBox.Show(e.Error.ToString(),
                    "SubmittedChanges Error", MessageBoxButton.OK);
                e.MarkErrorAsHandled();
            }
            else
            {
                MessageBox.Show("Submitted!");
            }
        }

        private void productDomainDataSource_LoadedData(object sender,
            LoadedDataEventArgs e)
        {
            if (e.HasError)
```

```
        {
            MessageBox.Show(e.Error.ToString(),
                "Load Error", MessageBoxButton.OK);
            e.MarkErrorAsHandled();
        }
    }

    private void dataForm1_EditEnded(object sender,
        DataFormEditEndedEventArgs e)
    {
        if (e.EditAction != DataFormEditAction.Commit)
            return;

        //save to db (add new item / edit current item)
        if (!productDomainDataSource.IsSubmittingChanges &&
            productDomainDataSource.HasChanges)
        {
            productDomainDataSource.SubmitChanges();
        }
    }

    private void dataForm1_DeletingItem(object sender,
        CancelEventArgs e)
    {
        var res = MessageBox.Show(
            "Do you want to delete this item?",
            "DeletingItem",
            MessageBoxButton.OKCancel);
        if (res == MessageBoxResult.Cancel)
        {
            e.Cancel = true;
            return;
        }

        if (productDomainDataSource.DataView.CanRemove)
        {
            var currentItem =
                  productDomainDataSource.DataView.CurrentItem;
            productDomainDataSource.DataView.Remove(currentItem);
            productDomainDataSource.SubmitChanges();
        }
    }

    private void dataForm1_AddingNewItem(object sender,
        DataFormAddingNewItemEventArgs e)
    {
        //save to db
        if (productDomainDataSource.DataView.CanAdd &&
```

```
                productDomainDataSource.HasChanges)
            {
                productDomainDataSource.SubmitChanges();
            }
        }
    }
}
```

توضیحات:

- ابتدا روال رخدادگردان SubmittedChanges تعریف شده‌است. اگر حین عملیات کار با بانک اطلاعاتی خطایی رخ دهد به این ترتیب می‌توان جزئیات خطا را دریافت کرد. همچنین به کمک متد MarkErrorAsHandled به برنامه اعلام خواهیم نمود که لطفا در اثر بروز خطا به کار خود پایان ندهید.
- روال رخدادگردان dataForm1_EditEnded کار مدیریت ثبت تغییرات رکوردهای اضافه شده یا رکورد جاری ویرایش شده را انجام می‌دهد. در اینجا ابتدا باید بررسی کرد که آیا واقعا تغییری رخ داده است (به کمک خاصیت HasChanges شیء productDomainDataSource) و آیا کنترل productDomainDataSource هم اکنون مشغول به روز رسانی بانک اطلاعاتی نیست (بررسی خاصیت IsSubmittingChanges)، سپس متد SubmitChanges آن‌را جهت ثبت نهایی عملیات در بانک اطلاعاتی فراخوانی خواهیم کرد.
- روال رخدادگردان dataForm1_DeletingItem پیش از حذف یک رکورد و در حین انجام این عملیات فراخوانی می‌گردد. بنابراین می‌توان ابتدا تائید کاربر را در این زمینه اخذ کرد و سپس نسبت به حذف رکورد جاری اقدام نمود. برای این منظور از امکانات کلاس DataView شیء productDomainDataSource استفاده خواهد شد. ابتدا وضعیت جاری بررسی شده و شیء در حال حذف دریافت می‌گردد. سپس این شیء حذف شده و نهایتا تغییرات در بانک اطلاعاتی دائمی خواهد شد (تا زمانیکه متد SubmitChanges فراخوانی نشده است، تغییرات دائمی نخواهند شد).

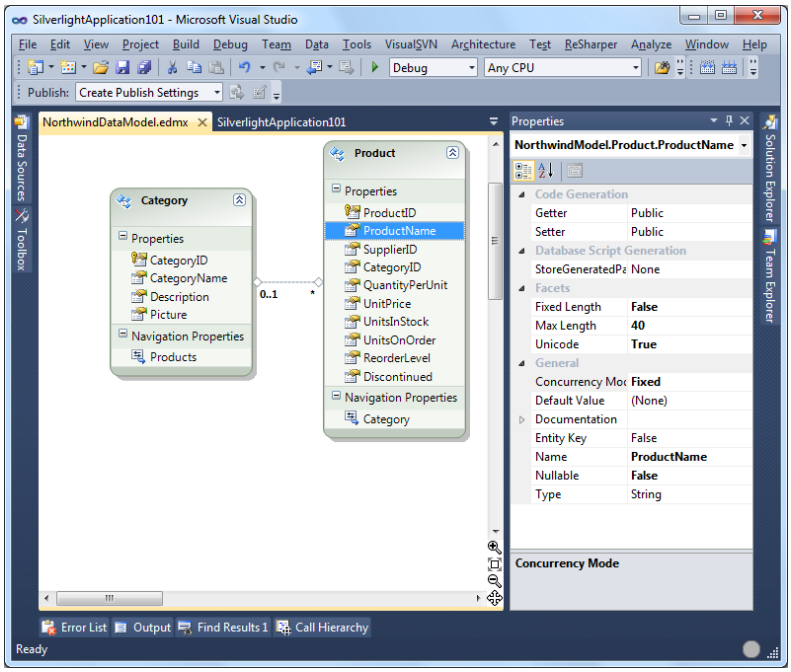
همانطور که ملاحظه می‌کنید کنترل DataForm انجام عملیات متداول کار با بانک اطلاعاتی را به شدت تسهیل می‌بخشد.

اگر علاقمند باشید که قیود اعتبار سنجی ویژه خود را تعریف نمائید باید به کلاس NorthwindDomainService.**metadata**.cs در پروژه‌ی ASP.NET برنامه رجوع کرد. در اینجا می‌توان همانند مباحث مطرح شده در طی فصل آشنایی با روش‌های اعتبار سنجی در Silverlight از Data annotations مناسب و مورد نظر خود استفاده نمود.

مواجه شدن با مسایل همزمانی در WCF RIA Services

مثال قبل را در نظر بگیرید. دو وهله از این برنامه را در دو مرورگر مختلف باز کنید (یا در دو صفحه‌ی جدید از یک مرورگر). در یک مرورگر نام محصولی را ویرایش کنید. همین عملیات را در مرورگر دوم نیز تکرار نمائید. خطایی را دریافت نخواهید کرد. به عبارت دیگر کاربر دوم متوجه نخواهد شد که اطلاعات رکورد مورد نظر پیشتر توسط کاربر دیگری ویرایش شده یا تغییر کرده است.

برای رفع این مشکل به فایل **NorthwindDataModel.edmx** مراجعه کرده و طراح آن‌را باز نمائید (شکل ۱۶). خاصیت ProductName را انتخاب کرده و سپس در برگه‌ی خواص آن، مقدار خاصیت Concurrency mode را از None به Fixed تغییر دهید. اکنون یکبار دیگر پروژه را Compile نمائید تا تغییرات اعمال شوند.



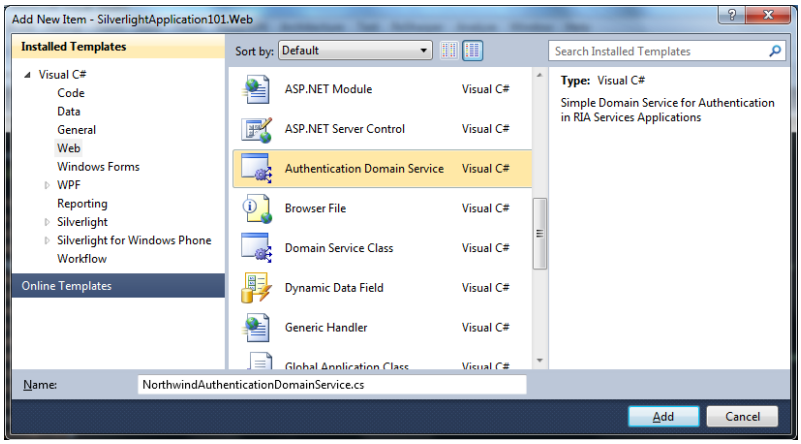
شکل ۱۶– تغییر حالت همزمانی فیلد نام محصول.

سپس مجددا دو وهله از برنامه را اجرا نمائید. در وهله‌ی اول نام یک محصول را ویرایش کنید. سپس به وهله‌ی دوم برنامه رجوع کرده و سعی نمائید نام قبلی موجود را ویرایش نمائید. بلافاصله پیغام تداخل را دریافت خواهید کرد. در این موارد می‌توان پیغامی را مبنی بر تغییرات اطلاعات توسط کاربرهای دیگر و به روز رسانی منبع داده‌ای با اطلاعات جدید، درنظر گرفت.

اعتبار سنجی کاربران توسط WCF RIA Services

اعتبار سنجی کاربران در حین استفاده از WCF RIA Services بر اساس امکانات اعتبار سنجی ASP.NET مانند Windows authentication ، Forms Authentication و Membership framework می‌باشد که در طی فصل امنیت در Silverlight با نحوه‌ی تعریف و استفاده از Forms authentication بیشتر آشنا شدیم. اگر پروژه‌ی Silverlight خود را از نوع قالب جدید برنامه‌های تجاری Silverlight انتخاب کرده باشید، به صورت پیش فرض دو کلاس AuthenticationService و UserRegistrationService را در پوشه‌ی Services برنامه ASP.NET پروژه می‌توانید پیدا کنید. اساس کار آن استفاده از سیستم Membership تعریف شده در ASP.NET است. به علاوه در سمت مشتری نیز اگر به پروژه‌ی Silverlight دقت نمائید فایل‌ها و صفحات login و ثبت نام کاربر (در پوشه‌ی Views\Login) به صورت خودکار اضافه شده‌اند و بدون نیاز به تنظیم خاصی کار خواهند کرد (فقط باید بانک اطلاعاتی سیستم ASPNET_Membership را پیشتر نصب و فعال کرده باشید).

در ادامه قصد داریم برنامه‌ی فصل را که تاکنون توسعه داده‌ایم با سیستم اعتبار سنجی WCF RIA Services یکپارچه کنیم و جهت توضیحات بیشتر از قالب جدید برنامه‌های تجاری Silverlight استفاده نخواهیم کرد و همچنین روش استفاده از ASPNET_Membership نیز مد نظر نخواهد بود.



شکل ۱۷– تعریف یک کلاس Authentication domain service جدید.

برای این منظور باید یک کلاس AuthenticationDomainService جدید را به پوشه‌ی Services در برنامه‌ی ASP.NET اضافه کرد (شکل ۱۷). از منوی پروژه گزینه‌ی افزودن یک آیتم جدید، تحت قسمت Web ، قالب Authentication domain service را انتخاب نموده و سپس نام NorthwindAuthenticationDomainService را وارد کرده و بر روی دکمه‌ی OK کلیک کنید. اگر سیستم

جاری شما از روش ASPNET_Membership استفاده می‌کند، تعاریف پیش فرض این کلاس کفایت کرده و نیازی به هیچگونه تغییری در آن نیست. اما در اینجا قصد داریم از یک روش Forms Authentication سفارشی استفاده کنیم. پیش از آنکه این روش سفارشی Forms Authentication مورد استفاده قرار گیرد باید این نوع اعتبار سنجی را در فایل Web.config برنامه فعال کرد:

Web.config

```
...
  <system.web>
    <authentication mode="Forms">
      <forms
        name=".403AuthRia"
        />
    </authentication>
...
```

سپس باید متدهای کلاس پایه AuthenticationBase را تحریف (override) کنیم. کدهای این روش سفارشی در ادامه ذکر شده است.

NorthwindAuthenticationDomainService.cs

```
using System;
using System.Security.Principal;
using System.ServiceModel.DomainServices.Hosting;
using System.ServiceModel.DomainServices.Server.ApplicationServices;
using System.Web;
using System.Web.Security;

namespace SilverlightApplication101.Web.Services
{
    [EnableClientAccess]
    public class NorthwindAuthenticationDomainService :
                                        AuthenticationBase<User>
    {
        protected override bool ValidateUser(
            string userName, string password)
        {
            if (string.IsNullOrWhiteSpace(userName))
                return false;

            if (userName == password)
            {
                //todo: read from db...
                return true;
            }

            return false;
        }
```

```
        protected override void IssueAuthenticationToken(
        IPrincipal principal, bool isPersistent)
    {
        //using named parameters of C# 4.0
        var ticket = new FormsAuthenticationTicket
            (
              version: 1,
              name: principal.Identity.Name,
              issueDate: DateTime.Now,
            //todo: read it from config...
              expiration: DateTime.Now.AddMonths(1),
              isPersistent: true,
              userData: string.Empty,
              cookiePath: FormsAuthentication.FormsCookiePath
            );

        string encryptedTicket =
                    FormsAuthentication.Encrypt(ticket);

        var cookie = new HttpCookie(
                        FormsAuthentication.FormsCookieName,
                        encryptedTicket)
                         {
                             Expires = DateTime.Now.AddMinutes(30)
                         };

        HttpContext.Current.Response.Cookies.Add(cookie);
    }

    protected override User GetAuthenticatedUser(
               IPrincipal principal)
    {
        return new User
                    {
                        EmployeeId = principal.Identity.Name,
                        Name = principal.Identity.Name,
                        Roles = MyRoleManager.GetRoles(
                                  principal.Identity.Name)
                    };
    }
}

public class User : UserBase
{
    public string EmployeeId { get; set; }
}

public class MyRoleManager
{
    public static string[] GetRoles(string userName)
```

```
        {
            //todo: read from db...
            return new[] { "Reader" };
        }
    }
}
```

توضیحات:

- با استفاده از متد Login در سمت مشتری که در ادامه در مورد آن توضیح داده خواهد شد، ابتدا متد ValidateUser این کلاس فراخوانی می‌گردد. در اینجا فرصت خواهید داشت تا نام کاربری و کلمه‌ی عبور او را با اطلاعات موجود در یک بانک اطلاعاتی دلخواه مقایسه نمائید و نتیجه را بازگشت دهید.
- سپس متد IssueAuthenticationToken فراخوانی می‌گردد. در اینجا می‌توان یک کوکی ماندگار روش اعتبار سنجی مبتنی بر Forms را صادر کرد.
- و در پایان عملیات به صورت خودکار متد GetAuthenticatedUser فراخوانی می‌گردد. اگر نیاز به اضافه کردن فیلدی سفارشی در اینجا وجود داشته باشد می‌توان آن(ها)را به کلاس User ذکر شده اضافه کرد تا در اختیار برنامه سمت مشتری قرار گیرد.

تا اینجا تنظیمات سمت Server جهت فعال سازی اعتبارسنجی سفارشی به پایان می‌رسد. اکنون نوبت به تنظیمات برنامه‌ی Silverlight است. به فایل App.xaml.cs مراجعه کرده و کدهای متد setAuthenticationContext زیر را به آن اضافه کنید:

App.xaml.cs

```
using System;
using System.ServiceModel.DomainServices.Client.ApplicationServices;
using System.Windows;

namespace SilverlightApplication101
{
    public partial class App
    {

        public App()
        {
            ...
            setAuthenticationContext();
        }

        private void setAuthenticationContext()
        {
            var webcontext = new WebContext
                                {
                                    Authentication =
```

```
                                                new FormsAuthentication()
                                };
            this.ApplicationLifetimeObjects.Add(webcontext);
        }
...
```

و یا روش دوم انجام اینکار افزودن چند سطر بعد به فایل App.XAML است:

App.xaml.cs

```
...
<Application.ApplicationLifetimeObjects>
        <app:RiaContext>
            <app:RiaContext.Authentication>
                <appsvc:FormsAuthentication/>
                <!--<appsvc:WindowsAuthentication/>-->
            </app:RiaContext.Authentication>
        </app:RiaContext>
</Application.ApplicationLifetimeObjects>
...
```

همانطور که ملاحظه می‌کنید در اینجا امکان تنظیم روش اعتبار سنجی مبتنی بر ویندوز نیز وجود دارد و در این حالت نیز همان کلاس ابتدایی NorthwindAuthenticationDomainService کفایت کرده و نیاز به تغییرات خاصی ندارد. این روش به صورت پیش فرض، با Active directory یکپارچه است. بدیهی است در این روش باید اندکی Web.Configبرنامه ASP.NET را به شکل بعد ویرایش نمود:

Web.config

```
...
  <system.web>
    <authentication mode="Windows"/>
...
```

و در فایل App.XAML.cs تنها کافی است متد ذیل را جهت دریافت اطلاعات کاربر جاری وارد شده به سیستم از Active directory ، فراخوانی نمود:

App.XAML.CS

```
...
public App()
{
  ...
  WebContext.Current.Authentication.LoadUser();
}
...
```

اکنون در ادامه، جهت تکمیل همان روش اعتبار سنجی مبتنی بر Froms که توضیح داده شد، دو صفحه‌ی TestFormsAuth.xaml (یک User control جدید) و ChildLoginWindow.xaml (یک Child window جدید) را به برنامه‌ی Silverlight خود اضافه کنید.

کدهای XAML صفحه‌ی ChildLoginWindow.xaml در ادامه ذکر شده‌اند. این صفحه شامل دو TextBox جهت دریافت نام کاربری و کلمه‌ی عبور کاربر است؛ به همراه یک برچسب جهت نمایش خطاهای رخ داده و دو دکمه برای انجام عملیات Login و یا بستن صفحه‌ی جاری و لغو عملیات:

ChildLoginWindow.xaml

```
<controls:ChildWindow
     x:Class="SilverlightApplication101.ChildLoginWindow"
     xmlns=
      "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:controls=
         "clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls"
      Width="400" Height="180" Title="Login">
    <Grid x:Name="LayoutRoot" Margin="2">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100" />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>

        <TextBlock Grid.Row="0" Margin="5"
                   Grid.Column="0" Text="User Name: ">
        </TextBlock>
        <TextBox x:Name="UserName"
                 Grid.Row="0"  Margin="5"
                 HorizontalAlignment="Left"
                 Grid.Column="1"
                 Width="150">
        </TextBox>
        <TextBlock Grid.Row="1"  Margin="5"
                   Grid.Column="0"
                   Text="Password: ">
        </TextBlock>
        <PasswordBox x:Name="Password"
                     HorizontalAlignment="Left"
                     Grid.Row="1"  Margin="5"
                     Grid.Column="1"
                     Width="150">
```

```
            </PasswordBox>
        <TextBlock x:Name="LoginResult"
                   TextWrapping="Wrap"
                   Grid.Row="2"
                   Grid.ColumnSpan="2"
                   Foreground="Red">
        </TextBlock>

        <Button x:Name="CancelButton" Content="Cancel"
                Click="CancelButton_Click" Width="75" Height="23"
                HorizontalAlignment="Right" Margin="0,12,0,0"
                Grid.Row="3" Grid.Column="1" />
        <Button x:Name="OKButton" Content="Login" Click="OKButton_Click"
                Width="75" Height="23" HorizontalAlignment="Right"
                Margin="0,12,79,0" Grid.Row="3"  Grid.Column="1"/>
    </Grid>
</controls:ChildWindow>
```

کدهای متناظر با این صفحه را در ادامه ملاحظه خواهید نمود:

ChildLoginWindow.xaml.cs

```
using System.ServiceModel.DomainServices.Client.ApplicationServices;
using System.Windows;

namespace SilverlightApplication101
{
    public partial class ChildLoginWindow
    {
        public ChildLoginWindow()
        {
            InitializeComponent();
        }

        private void OKButton_Click(object sender, RoutedEventArgs e)
        {
            var lp = new LoginParameters(
                UserName.Text, Password.Password);
            WebContext.Current.Authentication.Login(
                lp,
                this.loginOperationCompleted,
                null);
            LoginResult.Text = "";
            OKButton.IsEnabled = false;
        }

        private void loginOperationCompleted(LoginOperation lo)
        {
            if (lo.HasError)
            {
                LoginResult.Text = lo.Error.Message;
```

```
                lo.MarkErrorAsHandled();
                OKButton.IsEnabled = true;
            }
            else if (!lo.LoginSuccess)
            {
                LoginResult.Text =
                  "Login failed. Please check user name and password.";
                OKButton.IsEnabled = true;
            }
            else if (lo.LoginSuccess)
            {
                //refresh user state
                //WebContext.Current.Authentication.LoadUser();

                this.DialogResult = true;
                this.Close();
            }
        }

        private void CancelButton_Click(object sender,
               RoutedEventArgs e)
        {
            this.DialogResult = false;
        }
    }
}
```

توضیحات:

در روال رخدادگردان OKButton_Click ، عملیات Login در سمت مشتری شروع خواهد شد. برای این منظور از متد WebContext.Current.Authentication.Login کمک گرفته می‌شود. آرگومان اول آن باید شامل اطلاعات شیءایی حاوی نام کاربری و کلمه‌ی عبور کاربر باشد. آرگومان دوم آن جهت دریافت نتیجه‌ی این عملیات غیرهمزمان همانند روش‌هایی که در فصل امنیت در Silverlight مرور کردیم می‌باشد.

سپس توسط متد loginOperationCompleted وضعیت عملیات Login دریافت و پردازش می‌شود. اگر خطایی رخ داده باشد یا اگر متد ValidateUser که پیشتر در مورد آن توضیح داده شد، مقدار false یا null را برگرداند، پیغام خطایی به کاربر نمایش خواهیم داد. در غیر اینصورت DialogResult را به True مقدار دهی کرده و صفحه را خواهیم بست.

از این صفحه می‌خواهیم در TestFormsAuth.xaml استفاده نمائیم که کدهای XAML آن در ادامه ذکر شده‌اند:

TestFormsAuth.xaml

```
<UserControl x:Class="SilverlightApplication101.TestFormsAuth"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
        xmlns:mc=
      "http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <StackPanel>
        <StackPanel Orientation="Horizontal">
            <Button Name="btnLogin" Margin="5"
                HorizontalAlignment="Left"
                Content="Login" Width="100"
                Click="btnLogin_Click" />
            <Button Name="btnLogout" Margin="5"
                HorizontalAlignment="Left"
                Content="Logout" Width="100"
                Click="btnLogout_Click" />
        </StackPanel>
        <StackPanel Orientation="Horizontal"
                    Name="userInfoPanel"
                    Visibility="Collapsed">
            <TextBlock Text="UserName:" Margin="5" />
            <TextBlock Margin="5" Name="txtUserName"/>
        </StackPanel>
    </StackPanel>
</UserControl>
```

در اینجا دو دکمه برای نشان دادن صفحه‌ی Login و یا بررسی عملیات خروج کاربر درنظر گرفته شده‌اند. هچنین می‌خواهیم نام کاربر را پس از عملیات موفقیت آمیز Login نمایش دهیم. کدهای C# این صفحه در ذیل آورده شده‌اند:

TestFormsAuth.xaml.cs

```
using System;
using System.ServiceModel.DomainServices.Client.ApplicationServices;
using System.Windows;

namespace SilverlightApplication101
{
    public partial class TestFormsAuth
    {
        public TestFormsAuth()
        {
            InitializeComponent();
            this.Loaded += TestFormsAuth_Loaded;
        }

        void TestFormsAuth_Loaded(object sender, RoutedEventArgs e)
        {
            if(!WebContext.Current.User.IsAuthenticated)
            {
                btnLogin_Click(this, null);
            }
        }
```

```
private void btnLogin_Click(object sender, RoutedEventArgs e)
{
    var loginForm = new ChildLoginWindow();
    loginForm.Closed += loginForm_Closed;
    loginForm.Show();
}

private void loginForm_Closed(object sender, EventArgs e)
{
    txtUserName.Text = string.Empty;

    var dialog = sender as ChildLoginWindow;
    if (dialog == null) return;
    if (dialog.DialogResult == true) //if OK ...
    {
        txtUserName.Text = WebContext.Current.User.Name;
        userInfoPanel.Visibility = Visibility.Visible;

        if (WebContext.Current.User.IsInRole("Managers"))
        {
            //do something ...
        }
    }
    else
    {
        userInfoPanel.Visibility = Visibility.Collapsed;
    }
}

private void btnLogout_Click(object sender, RoutedEventArgs e)
{
    WebContext.Current.Authentication.Logout(
        this.LogoutOperation_Completed,
        null);
}

private void LogoutOperation_Completed(LogoutOperation lo)
{
    if (!lo.HasError)
    {
        userInfoPanel.Visibility = Visibility.Collapsed;
    }
    else
    {
        //Logout failed.
        lo.MarkErrorAsHandled();
    }
```

```
        }
    }
}
```

توضیحات:

در روال آغازین TestFormsAuth_Loaded بررسی خواهیم نمود که آیا کاربر جاری اعتبار سنجی شده است (IsAuthenticated)؟ اگر خیر، صفحه‌ی Login را به صورت خودکار به او نمایش خواهیم داد. پس از پایان عملیات Login، اگر نتیجه‌ی آن مثبت باشد، ابتدا قسمت دربرگیرنده‌ی برچسب‌های نمایش اطلاعات کاربر را نمایان کرده و سپس با کمک خاصیت WebContext.Current.User.Name ، نام او را نمایش خواهیم داد. همچنین با استفاده از متد WebContext.Current.User.IsInRole نیز می‌توان بر اساس نقش‌های انتساب داده شده به کاربر جاری، قسمت‌های بیشتری از صفحه را برای او نمایان کرد.

این نقش‌ها در فایل NorthwindAuthenticationDomainService.cs به شیء User انتساب داده می‌شوند (متد GetAuthenticatedUser).

در ادامه روش خروج یک کاربر به کمک متد WebContext.Current.Authentication.Logout نیز بیان شده است. در صورت موفقیت آمیز بودن عملیات می‌توان قسمت‌های مختلف صفحه را که نیاز به اعتبار سنجی دارند از دید او مخفی کرد.

برای امنیت بیشتر برنامه ، در فایل NorthwindDomainService.cs نیز می‌توان صریحا مشخص ساخت که کدامیک از متدها عمومی بوده و بدون نیاز به اعتبار سنجی قابل دسترسی هستند و اطلاعات کدامیک تنها در اختیار کاربران اعتبار سنجی شده قرار می‌گیرد. برای مثال اگر در این کلاس متد InsertProduct را با ویژگی RequiresAuthentication مزین کنیم، دیگر کاربران اعتبار سنجی نشده دسترسی به آن نخواهند داشت:

NorthwindDomainService.cs

```
...
        [RequiresAuthentication]
        public void InsertProduct(Product product)
...
```

حتی می‌توان علاوه بر این مورد، نقش خاصی را نیز صریحا ذکر نمود:

NorthwindDomainService.cs

```
...
        [RequiresAuthentication]
        [RequiresRole("Managers")]
        public void DeleteProduct(Product product)
...
```

در این مثال تنها کاربران اعتبار سنجی شده‌ای که دارای نقش Managers باشند، مجاز به حذف یک محصول خواهند بود.