Silverlight 4

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

فهرست مطالب

فصل ۱۳- استفاده از Web Services در Silverlight

مقدمه

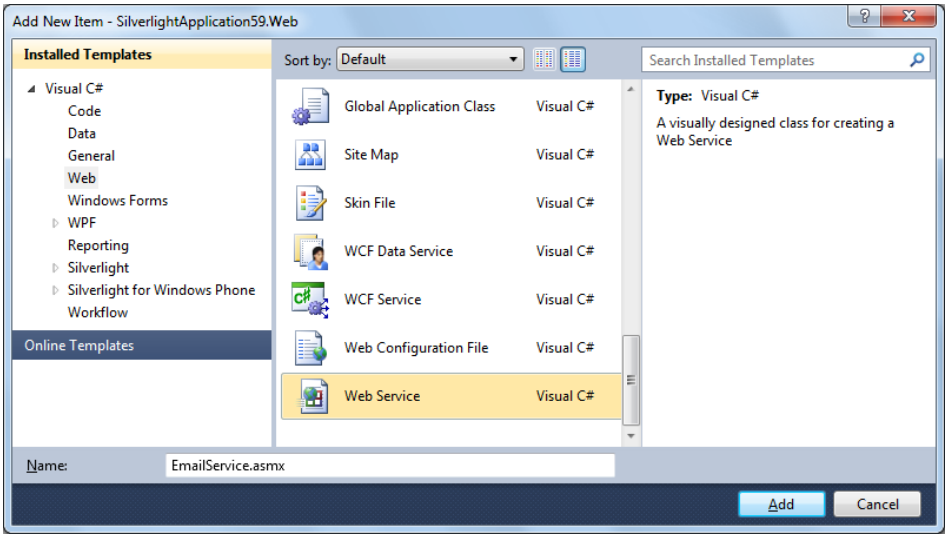
Silverlight امکان استفاده از بسیاری از سرویس‌های موجود در شبکه و یا Internet را دارا است؛ برای مثال RESTful services ، syndicated feeds ، SOAP services ، WCF Services و غیره. در این فصل با نحوه‌ی ایجاد ASMX web services و همچنین WCF web services آشنا شده و سپس نحوه‌ی استفاده از آن‌ها را در Silverlight بررسی خواهیم نمود.

ایجاد یک ASMX Web Service جدید

استفاده از Web services از این جهت حائز اهمیت می‌باشد که Silverlight یک فناوری سمت کاربر است و به دلایل امنیتی، محدودیت‌های بسیاری بر آن اعمال شده است. برای مثال Silverlight دسترسی به فضای نام متداول ارسال ایمیل در .NET را ندارد. اما می‌توان یک Web Service ارسال ایمیل را ایجاد نمود و توانایی‌های سمت سرور آن‌را در اختیار برنامه‌های Silverlight قرار داد. در اینجا قصد داریم در طی یک مثال، یک Web Service ارسال ایمیل را ایجاد کرده و سپس نحوه‌ی ایجاد کلاس proxy جهت دسترسی به این Web Service و سپس ارسال و دریافت اطلاعات به آن‌را در Silverlight بررسی نمائیم.

برای این منظور ابتدا یک پروژه‌ی جدید Silverlight را آغاز نموده و زمانیکه در حین ایجاد پروژه در VS.NET سؤال می‌شود آیا از یک Web Site برای مدیریت آن استفاده شود، این گزینه را نیز انتخاب نمائید؛ زیرا از ASP.NET Web Site اضافه شده جهت ایجاد ASMX Web Service مورد نظر استفاده خواهیم نمود.

در ادامه به پروژه‌ی ASP.NET اضافه شده مراجعه نموده و از منوی پروژه، گزینه‌ی Add New Item ، یک Web Servive جدید را به نام EmailService اضافه نمائید (شکل ۱).



شکل ۱– افزودن یک Web Service جدید به پروژه ASP.NET

سپس یک کلاس جدید به نام SendMail را به این پروژه‌ی ASP.NET اضافه کنید :

SendMail.cs

```
using System.Net.Mail;
namespace SilverlightApplication59.Web
{
    public class SendMail
    {
        //TODO: read these values form web.config
        private readonly string host = "mail.yoursmtpserver.net";
        private readonly int port = 25;
        public void SendMessage(string from, string to,
            string subject, string body)
        {
            using (var message =
            new MailMessage
            {
                From = new MailAddress(from), Body = body, Subject = subject
            })
            {
                message.To.Add(to);
                var smtpClient = new SmtpClient(host, port);
                smtpClient.Send(message);
            }
        }
    }
}
```

در این مثال مقدارهای مرتبط با SMTP Server و Port آن به صورت صریح در کلاس ذکر شده‌اند (و باید مطابق اطلاعات SMTP Server شما تغییر نمایند). روش صحیح کار با اینگونه مقادیر، قرار دادن آن‌ها در فایل Web.Config و سپس فراخوانی آن‌ها در برنامه است. به این ترتیب بدون نیاز به Compile مجدد برنامه می‌توان در صورت نیاز، تغییرات جدید تنظیمات را صرفا با ویرایش فایل Web.Config اعمال نمود.

در ادامه به کلاس EmailService.asmx.cs مراجعه نموده و کدهای آن را مطابق اطلاعات زیر تغییر دهید (متد SendMessage کلاس SendMail را به صورت یک سرویس ارائه خواهد داد):

EmailService.asmx.cs

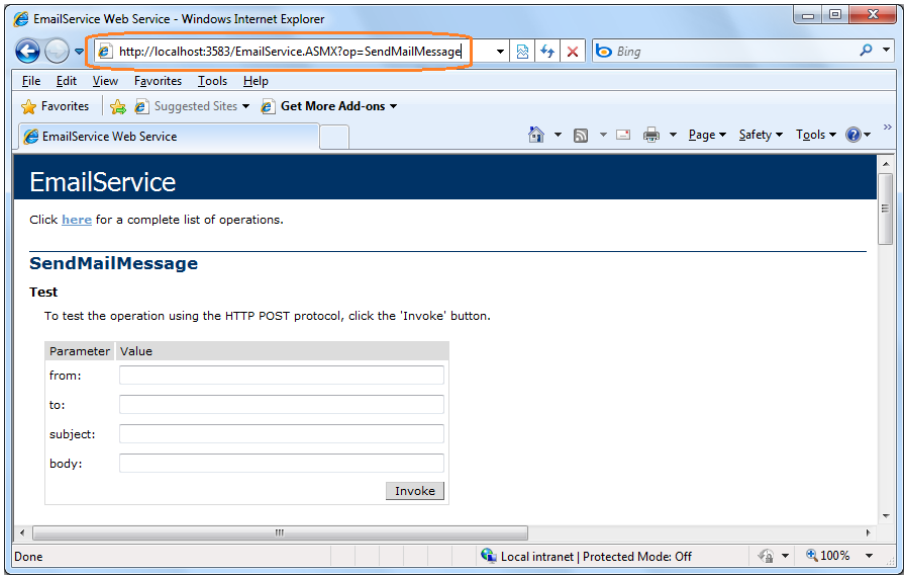
```
using System.Web.Services;

namespace SilverlightApplication59.Web
{
    /// <summary>
    /// Summary description for EmailService
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    public class EmailService : WebService
    {
        [WebMethod]
        public void SendMailMessage(string from, string to,
            string subject, string body)
        {
            new SendMail().SendMessage(from, to, subject, body);
        }
    }
}
```

همانطور که ملاحظه می‌نمائید ایجاد یک ASMX Web Service توسط ASP.NET بسیار ساده بوده و تنها کافی متد مورد نظر خود را با ویژگی WebMethod مزین نمائیم تا توسط Web Service فوق، قابل ارائه گردد.

جهت بررسی WebService ایجاد شده، برنامه‌ی ASP.NET را در حالت Debug در VS.NET اجرا نمائید و سپس به مسیر زیر مراجعه کنید (شکل ۲) :

http://localhost:3583/EmailService.ASMX?op=SendMailMessage

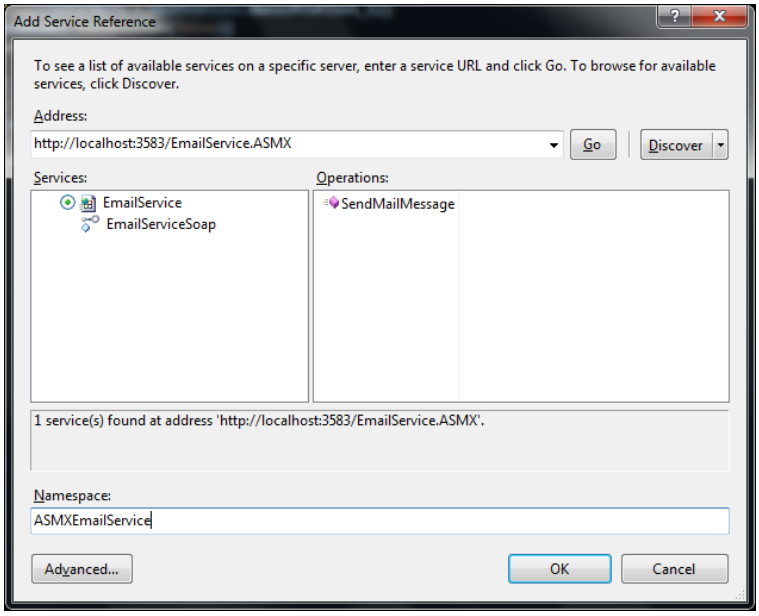


شکل ۲– بررسی امکان ارسال ایمیل توسط Web Service

بدیهی است Port شماره 3583 توسط Web Server آزمایشی VS.NET انتساب داده شده است و این شماره بر روی کامپیوتر شما به طور قطع مورد دیگری خواهد بود. همچنین اگر از IIS جهت مدیریت وب سایت فوق استفاده نمائید، عموما Port شماره ۸۰ مورد استفاده قرار می‌گیرد که نیازی به ذکر آن در URL فوق نخواهد بود.

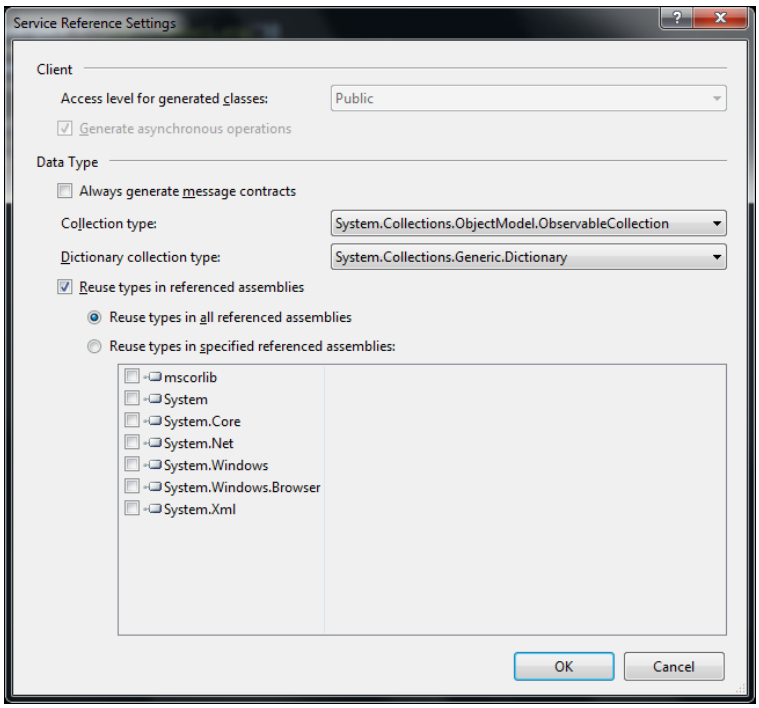
استفاده از یک ASMX Web Service در Silverlight

در قسمت قبل کار ایجاد یک EmailService به پایان رسید. اکنون می‌خواهیم از توانایی‌های آن در پروژه‌ی Silverlight جاری استفاده نمائیم. برای این منظور به منوی پروژه، گزینه‌ی Add Service reference مراجعه نمائید و سپس مسیر Web Service ایجاد شده را وارد نموده و بر روی دکمه‌ی Go کلیک نمائید (شکل ۳).
پس از مدتی، اطلاعات Web Service دریافت شده و متد SendMailMessage آن تشخیص داده خواهد شد.
پیش از کلیک بر روی دکمه‌ی OK ، بر روی دکمه‌ی Advanced در صفحه‌ی افزودن ارجاع به Web Service کلیک نمائید تا صفحه‌ی مربوطه نمایش داده شود (شکل ۴). همانطور که در این شکل ملاحظه می‌نمائید، VS.NET به صورت خودکار خروجی یک Web Service از نوع List<T> را به اشیایی از نوع ObservableCollection<T> جهت سهولت استفاده از آن‌ها در عملیات پیشرفته‌ی انقیاد داده‌ها در Silverlight تبدیل خواهد نمود.



شکل ۳– افزودن ارجاعی به Web Service ایجاد شده در پروژه‌ی Silverlight

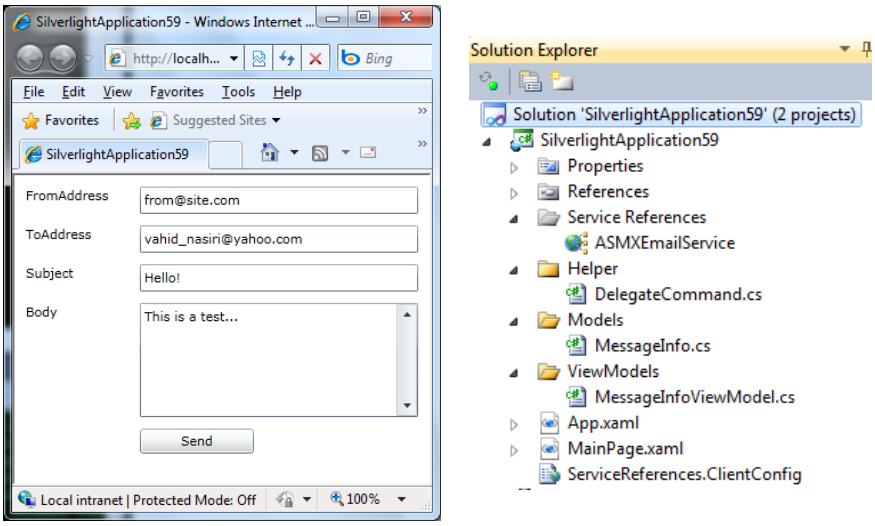
اکنون صفحه‌ی تنظیمات پیشرفته‌ی افزودن ارجاعی به Web Service را بسته و در صفحه‌ی اصلی، نام ASMXEmailService را وارد نموده و بر روی دکمه‌ی Ok کلیک نمائید تا کار ایجاد Proxy های لازم جهت دسترسی به Web Service ایجاد شده در برنامه‌ی Silverlight ما به پایان برسد.



شکل ۴– صفحه‌ی تنظیمات پیشرفته‌ی افزودن ارجاعی به یک Web Service

تنظیمات مرتبط با مسیر این Web Service مورد نظر را در فایل جدید ServiceReferences.ClientConfig اضافه شده به پروژه جاری می‌توان یافت. برای مثال اگر آدرس Web Service مورد استفاده تغییر نماید، تنها کافی است مسیر جدید را در فایل ذکر شده ویرایش نمائید.

در ادامه قصد داریم پیاده سازی استفاده از Web Service ارسال ایمیل فوق را با استفاده از الگوی MVVM انجام دهیم (شکل ۵). به همین منظور پوشه‌های جدید Helper ، Models و ViewModels را به پروژه‌ی Silverlight اضافه نمائید. در پوشه‌ی Helper همان کلاس معروف DelegateCommand که در فصل‌های قبل معرفی گردید، قرار خواهد گرفت و از تکرار مجدد کدهای آن در اینجا صرفنظر خواهد شد. تنها View برنامه همان صفحه‌ی اصلی و فایل MainPage.xaml می‌باشد. در ادامه کدهای Model و ViewModel برنامه را بررسی خواهیم نمود.



شکل ۵– نمایی از ساختار پروژه‌ی ارسال ایمیل به کمک یک Web Service

مدل برنامه

کلاس ساده MessageInfo مدل برنامه را تشکیل می‌دهد :

MessageInfo.cs

```
using System.ComponentModel;

namespace SilverlightApplication59.Models
{
    public class MessageInfo : INotifyPropertyChanged
    {
        public string FromAddress { get; set; }
        public string ToAddress { get; set; }
        public string Subject { get; set; }
```

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

        string _result;
        public string Result
        {
            get { return _result; }
            set
            {
                if (_result == value) return;
                _result = value;
                raisePropertyChanged("Result");
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;

        void raisePropertyChanged(string propertyName)
        {
            var handler = PropertyChanged;
            if (handler == null) return;
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

خاصیت Result جهت نمایش حاصل عملیات به کاربر اضافه شده است و در برنامه به صورت مستقیم مقدار دهی می‌گردد. سایر خواص از طریق انقیاد دو طرفه صرفا توسط کاربر در جعبه‌های متنی برنامه وارد شده و اطلاعات آن‌ها به این صورت دریافت می‌گردد.

View برنامه

کدهای XAML صفحه‌ی اصلی برنامه به شرح بعد می‌باشند:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication59.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"
    xmlns:vm="clr-namespace:SilverlightApplication59.ViewModels"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Resources>
```

```
        <vm:MessageInfoViewModel x:Key="vmMessageInfoViewModel" />
</UserControl.Resources>
<Grid x:Name="LayoutRoot" Background="White" Margin="5"
      DataContext="{StaticResource vmMessageInfoViewModel}"
      >
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="100"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition  Height="Auto"/>
        <RowDefinition  Height="Auto"/>
        <RowDefinition  Height="Auto"/>
        <RowDefinition  Height="Auto"/>
        <RowDefinition  Height="Auto"/>
        <RowDefinition  Height="*"/>
    </Grid.RowDefinitions>

    <TextBlock Text="FromAddress"
           Grid.Column="0" Grid.Row="0" Margin="5"  />
    <TextBox Grid.Row="0" Grid.Column="1" Margin="5"
        Text="{Binding Mode=TwoWay, Path=MessageInfo.FromAddress}"/>

    <TextBlock Text="ToAddress"
            Grid.Column="0" Grid.Row="1" Margin="5"  />
    <TextBox Grid.Row="1" Grid.Column="1" Margin="5"
        Text="{Binding Mode=TwoWay, Path=MessageInfo.ToAddress}"/>

    <TextBlock Text="Subject"
                Grid.Column="0" Grid.Row="2" Margin="5"  />
    <TextBox Grid.Row="2" Grid.Column="1" Margin="5"
            Text="{Binding Mode=TwoWay, Path=MessageInfo.Subject}"/>

    <TextBlock Text="Body" Grid.Column="0"
            Grid.Row="3" Margin="5"  />
    <TextBox Grid.Row="3" Grid.Column="1" Margin="5"
       Height="100" VerticalScrollBarVisibility="Visible"
       AcceptsReturn="True" TextWrapping="Wrap"
      Text="{Binding Mode=TwoWay, Path=MessageInfo.Body}"/>

    <Button Grid.Column="1" Grid.Row="4" Width="100"
            HorizontalAlignment="Left" Margin="5"
            VerticalAlignment="Top" Content="Send"
            Command="{Binding SendMessage}"
            CommandParameter="{Binding MessageInfo}"
            />

    <TextBlock Text="{Binding Path=MessageInfo.Result}"
               VerticalAlignment="Top"
               TextWrapping="Wrap" MaxWidth="250"
```

```
                Grid.Column="1" Grid.Row="5" Margin="5"
                />
    </Grid>
</UserControl>
```

View برنامه، اطلاعات ViewModel را از طریق منبع ثابتی به نام vmMessageInfoViewModel که تعریف آن را در قسمت UserControl.Resources ملاحظه می‌نمائید و در نهایت به DataContext گرید صفحه انتساب داده شده است، دریافت می‌کند.

کلیه TextBox های قرار گرفته بر روی صفحه اطلاعات خود را از طریق Binding دریافت می‌کنند و در پایان صفحه، دکمه‌ی ارسال اطلاعات، داده‌های این شیء را از طریق اشیاء Command به ViewModel ارسال خواهد نمود.

ViewModel برنامه

کار ViewModel برنامه، در اختیار قرار دادن اطلاعات شیء MessageInfo به View برنامه بوده و همچنین مدیریت رخدادهای دریافتی از طریق شیء Command . کدهای کلاس MessageInfoViewModel را در ادامه ملاحظه خواهید نمود :

MessageInfoViewModel.cs

```
using System.ComponentModel;
using System.Windows.Input;
using SilverlightApplication59.Helper;
using SilverlightApplication59.Models;

namespace SilverlightApplication59.ViewModels
{
    public class MessageInfoViewModel
    {
        public MessageInfo MessageInfo { set; get; }
        public ICommand SendMessage { set; get; }

        public MessageInfoViewModel()
        {
            MessageInfo = new MessageInfo
                          {
                              FromAddress = "from@site.com",
                              ToAddress = "vahid_nasiri@yahoo.com",
                              Subject = "Hello!",
                              Body = "This is a test...",
                              Result = string.Empty
                          };
```

```
            SendMessage =
             new DelegateCommand<MessageInfo>(sendEmail, canSendEmail);
        }


        private static bool canSendEmail(MessageInfo enteredInfo)
        {
            //TODO: Validation
            return enteredInfo != null;
        }

        private void sendEmail(MessageInfo enteredInfo)
        {
            var srv = new ASMXEmailService.EmailServiceSoapClient();
            srv.SendMailMessageCompleted +=
                                srv_SendMailMessageCompleted;
            srv.SendMailMessageAsync(
                enteredInfo.FromAddress,
                enteredInfo.ToAddress,
                enteredInfo.Subject,
                enteredInfo.Body
                );
        }

        void srv_SendMailMessageCompleted(object sender,
            AsyncCompletedEventArgs e)
        {
            string msg = "Message was ";
            if (e.Error != null)
                msg += string.Format("not sent.\n\n{0}",
                                 e.Error.Message);
            else
                msg += "sent using ASMX";

            MessageInfo.Result = msg;
        }
    }
}
```

مهمترین قسمت‌های این ViewModel که مقصود اصلی برنامه نیز هستند، متدهای ارسال ایمیل و پایان ارسال ایمیل می‌باشند. در متد sendEmail یک وهله از شیء Web Service ارسال ایمیل که ارجاعی از آن را پیشتر به برنامه اضافه کرده بودیم، ساخته شده و سپس از طریق متد SendMailMessageAsync آن نسبت به ارسال ایمیل اقدام خواهد شد. نکته‌ای را که باید به آن دقت داشت امکان اعتبار سنجی اطلاعات دریافت شده در متد

canSendEmail است. اگر خروجی این متد False باشد، دکمه‌ی ارسال ایمیل تا زمان برآورده شدن شرایط لازم، غیرفعال خواهد گردید.

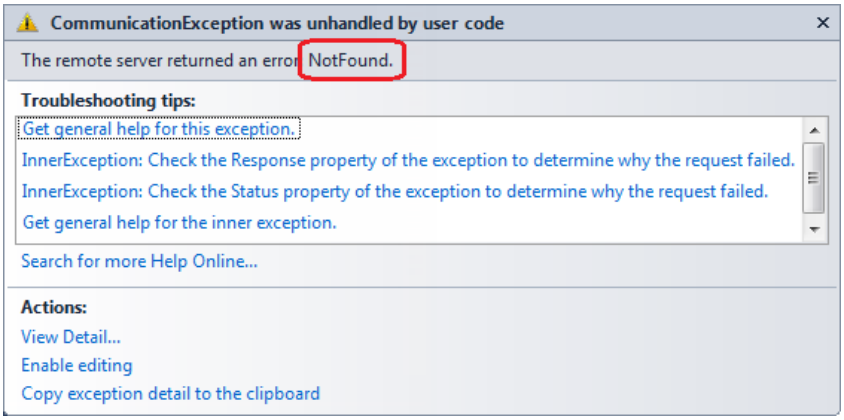
کلیه اعمال انجام شده با Web Service ها در Silverlight از نوع غیرهمزمان و Asynchronously می‌باشند. به همین جهت تنها متد SendMailMessage**Async** در لیست متدهای ارسال ایمیل موجود در این Web service در برنامه ظاهر شده است (و متدی به نام SendMailMessage وجود ندارد). به این صورت رابط گرافیکی کاربر برنامه در حین کاربر با یک Web Service از فعالیت باز نخواهد ایستاد.

در ادامه برای مشخص شدن زمان پایان کار، از رخداد SendMailMessageCompleted استفاده گردیده است. این متد اطلاعات نهایی را طریق آرگومانی از نوع AsyncCompletedEventArgs دریافت می‌کند. برای مثال خاصیت e.Error آن مشخص می‌سازد که آیا عملیات با موفقیت به پایان رسیده است یا خیر.

اکنون برنامه را اجرا نمائید. بلافاصله پیغام Not found مبتنی بر عدم یافت شدن Web Service مورد نظر را دریافت خواهید نمود (شکل ۶). البته پیغام خطای مورد نظر آنچنان گویا نبوده و منظور اصلی از آن، عدم یافت شدن cross-domain policy file در کنار Web Service برنامه می‌باشد. همچنین این خطا به معنای بروز استثناء در Web Service تعریف شده نیز می‌تواند باشد. در ادامه این موضوع را با جزئیات بیشتری بررسی خواهیم کرد.

فراخوانی‌های بین Domain ها و مسایل امنیتی مرتبط با آن‌ها

به صورت پیش فرض برنامه‌های Silverlight نمی‌توانند Web Service ایی را که بر روی Domain دیگری بجز Domain ایی که بر روی آن قرار گرفته اند، فراخوانی نمایند. برای مثال اگر برنامه‌ی Silverlight شما بر روی Domain ایی به نام sl1.com قرار گرفته است، تنها به وب سرویس‌های موجود بر روی sl1.com دسترسی خواهد داشت (در اینجا Port مورد استفاده نیز مد نظر خواهد بود).



شکل ۶– خطای حاصل در حین اجرای اولین درخواست از Web Service.

برای دسترسی به Web Service قرار گرفته بر روی Domain و Port دیگری بجز Domain و Port برنامه‌ی Silverlight ما، نیاز است تا آن سایت مقصد توسط فایلی به نام clientaccesspolicy.xml ، مشخص نماید که کدام Domain ها مجوز دسترسی به خدمات Web Service او را دارند. برنامه‌های Flash نیز از یک چنین سیستم امنیتی استفاده کرده و نیاز به فایلی به نام crossdomain.xml در ریشه‌ی سایت دارند. نکته‌ی مهمی که در مورد محل قرارگیری این فایل اهمیت دارد آن است که محل آن حتما باید در ریشه‌ی سایت تعریف گردد و نه در ریشه‌ی اصلی برنامه که می‌تواند از یک زیر پوشه‌ی سایت نیز تشکیل شده باشد.

این محدودیت بر روی ASMX Web services ، WCF web services ، درخواست‌های WebClient و HttpWebRequest نیز اعمال می‌گردد. اگر به مثال دریافت اطلاعات یک feed در فصل معرفی MVVM light toolkit دقت کرده باشید، مشکلی از لحاظ دسترسی به اطلاعات سایت feedburner.com وجود نداشت، زیر این سایت، فایل‌های XML مذکور را تعریف و دسترسی‌های لازم را اعطاء نموده است.

در ادامه یک فایل XML جدید به نام clientaccesspolicy.xml را به پروژه‌ی ASP.NET دربرگیرنده‌ی Web Service ارسال ایمیل اضافه نمائید. سپس محتویات آن‌را به صورت ذیل تعریف نمائید :

clientaccesspolicy.xml

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
  <cross-domain-access>
    <policy>
      <allow-from http-request-headers="*">
        <domain uri="*"/>
      </allow-from>
      <grant-to>
        <resource path="/" include-subpaths="true"/>
      </grant-to>
    </policy>
  </cross-domain-access>
</access-policy>
```

تنظیمات امنیتی فوق بدین معنا است که کلیه Domain ها به تمامی Web Service های قرار گرفته بر روی Domain جاری دسترسی خواهند داشت (و بدیهی است جهت محیط کاری اصلا توصیه نمی‌شود). در اینجا خاصیت uri المان domain به ستاره تنظیم شده است و به معنای مجاز بودن کلیه Domain های عالم است.

فایل clientaccesspolicy.xml بعد، تنظیمات معقول‌تری را ارائه می‌دهد. در این مثال تنها Web Service های قرار گرفته در مسیر MyAwesomeServices به اشتراک گذاشته خواهند شد (همچنین زیر پوشه‌های آن نیز با توجه به تنظیم include-subpaths به true نیز قابل دسترسی خواهند بود) . به علاوه توسط مقدار دهی http-request-headers ، تنها درخواست‌های خاص ذکر شده مجاز خواهند بود. علاوه بر آن تنها یک Domain مشخص شده به نام silverlight-data.com امکان دسترسی به منابع Web service مذکور را خواهد داشت.

clientaccesspolicy.xml

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
  <cross-domain-access>
    <policy>
      <allow-from http-request-headers="SOAPAction, Content-Type">
        <domain uri="http://silverlight-data.com"/>
      </allow-from>
      <grant-to>
        <resource path="/MyAwesomeServices/" include-subpaths="true"/>
      </grant-to>
    </policy>
  </cross-domain-access>
</access-policy>
```

لازم به ذکر است که همیشه Web services مورد استفاده، توسط ما تولید نخواهند شد. در این حالات، Silverlight ابتدا به دنبال فایل clientaccesspolicy.xml خواهد گشت و اگر این فایل بر روی ریشه سایت یافت نشد، فایل crossdomain.xml را جستجو خواهد نمود. این قالب توسط Web Service های برنامه‌های مبتنی بر Flash مورد استفاده قرار می‌گیرد. برای مثال فایل XML ذیل، قسمتی از محتوای فایل سایت زیر را نمایش می‌دهد:

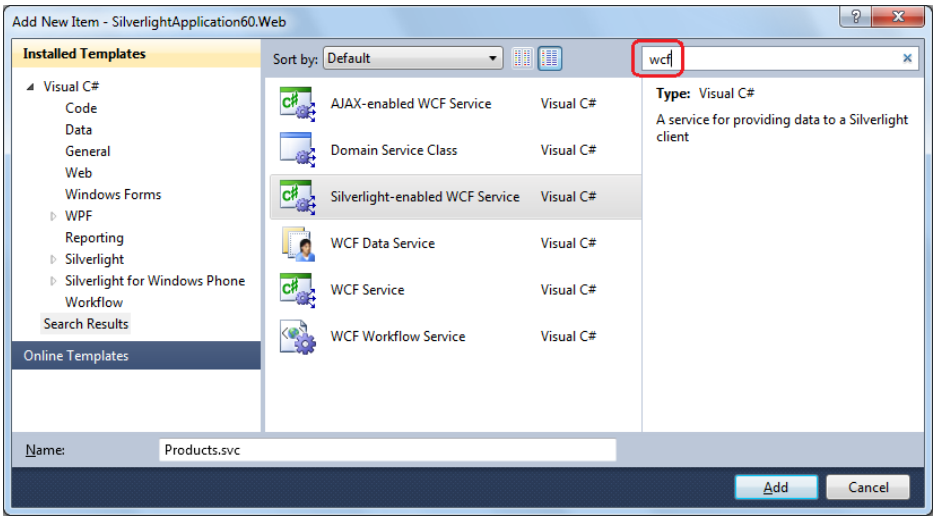
amazon.com/crossdomain.xml

crossdomain.xml

```
<?xml version="1.0" ?>
<cross-domain-policy>
  <allow-access-from domain="*.amazon.com" secure="true" />
  <allow-access-from domain="amazon.com" secure="true" />
  <allow-access-from domain="www.amazon.com" secure="true" />
  <allow-access-from domain="pre-prod.amazon.com" secure="true" />
  <allow-access-from domain="devo.amazon.com" secure="true" />
  <allow-access-from domain="anon.amazon.speedera.net" secure="true" />
  <allow-access-from domain="*.images-amazon.com" secure="true" />
  <allow-access-from domain="*.ssl-images-amazon.com" secure="true" />
  <allow-access-from domain="*.amazon.ca" secure="true" />
  <allow-access-from domain="*.amazon.de" secure="true" />
  <allow-access-from domain="*.amazon.fr" secure="true" />
  <allow-access-from domain="*.amazon.jp" secure="true" />
  <allow-access-from domain="*.amazon.co.jp" secure="true" />
  <allow-access-from domain="*.amazon.uk" secure="true" />
  <allow-access-from domain="*.amazon.co.uk" secure="true" />
</cross-domain-policy>
```

نحوه‌ی ایجاد یک WCF Service سازگار با Silverlight

WCF به عنوان جایگزینی یکپارچه برای سرویس‌های ASMX ، MSMQ ، WSE ، .NET Remoting و COM+ Enterprise services مطرح است. یکی از مزیت‌های مهم استفاده از WCF Services ، محدود نبودن آن به برنامه‌های ASP.NET و IIS جهت ارائه‌ی خدمات خویش می‌باشند. برای مثال می‌توان از یک سرویس ویندوز NT یا یک برنامه‌ی Console نیز به عنوان Host یک WCF Service استفاده کرد. WCF Services نسبت به ASMX Web Services بیش از ۲۵ درصد سریعتر هستند. همچنین تنظیمات امنیتی بیشتری نیز در این نوع قابل استفاده و اعمال است و پروتکل‌های بیشتری پشتیبانی می‌گردند. WCF خود می‌تواند عنوان یک کتاب جامع باشد.



شکل ۷– افزودن یک WCF Service سازگار با Silverlight

برای آشنایی با نحوه‌ی استفاد از WCF Services در Silverlight یک مثال کاربردی را با هم مرور خواهیم کرد. در این مثال WCF Service تعریف شده، لیستی از محصولات را بازگشت داده و سپس پروژه‌ی Silverlight ما آن‌هارا توسط یک DataGrid نمایش خواهد داد.

برای این منظور یک پروژه‌ی جدید Silverlight را به همراه Web Site آن آغاز نمائید. سپس از منوی پروژه، گزینه‌ی Add new item ، یک Silverlight-enabled WCF Service را به نام Products.svc مطابق شکل ۷ به پروژه‌ی ASP.NET برنامه اضافه کنید. مزیت استفاده از گزینه‌ی Silverlight-enabled WCF Service ، انجام تنظیمات خودکار سازگار با Silverlight می‌باشد که صرفه جویی زمانی قابل ملاحظه‌ای را به همراه خواهد داشت. به این صورت تنظیمات لازم (در مورد پروتکل سازگار و همچنین تعاریف دیگر) به صورت

خودکار به Web.Config برنامه اضافه شده و علاوه بر آن ویژگی AspNetCompatibilityRequirements نیز به فایل Products.svc.cs برنامه اضافه خواهد شد.

در ادامه یک کلاس جدید به نام Product را به پروژه‌ی ASP.NET اضافه نمائید :

Product.cs

```
using System.Runtime.Serialization;

namespace SilverlightApplication60.Web
{
    [DataContract]
    public class Product
    {
        [DataMember]
        public int ProductId { get; set; }
        [DataMember]
        public string ProductName { get; set; }
    }
}
```

توسط این کلاس ساختار اشیایی که ارائه خواهند گشت، تعریف می‌شود. زمانیکه از ویژگی DataContract استفاده گردد، آن شیء توسط مصرف کنندگان قابل دسترسی خواهد بود و زمانیکه هر یک از خواص این کلاس توسط ویژگی DataMember مزین می‌گردند، امکان عملیات serialization آن‌ها توسط WCF مهیا خواهد شد.

سپس اینترفیس جدیدی را به این پروژه به نام IProductServiceContract جهت تعریف ServiceContract و OperationContract متناظر با خدماتی که ارائه خواهیم داد، تعریف نمائید :

IProductServiceContract.cs

```
using System.Collections.Generic;
using System.ServiceModel;

namespace SilverlightApplication60.Web
{
    [ServiceContract]
    public interface IProductServiceContract
    {
        [OperationContract]
        List<Product> GetProductsList();
    }
}
```

متدهای مزین شده به OperationContract ، توسط WCF Service در اختیار سایر برنامه‌ها و مصرف کنندگان آن قرار خواهند گرفت.

اکنون این اینترفیس را در کلاس Products.svc.cs به شکل زیر پیاده سازی خواهیم نمود (ابتدا کدهای پیش فرض آن‌را حذف کرده و سپس کدهای ذیل را اضافه کنید):

Products.svc.cs

```
using System.Collections.Generic;
using System.ServiceModel.Activation;

namespace SilverlightApplication60.Web
{
    [AspNetCompatibilityRequirements(RequirementsMode
        = AspNetCompatibilityRequirementsMode.Allowed)]
    public class Products : IProductServiceContract
    {
        public List<Product> GetProductsList()
        {
            return
                new List<Product>
                    {
                        new Product
                            {
                                ProductId = 1,
                                ProductName = "CD"
                            },
                        new Product
                            {
                                ProductId = 2,
                                ProductName = "DVD"
                            },
                        new Product
                            {
                                ProductId = 3,
                                ProductName = "RAM"
                            }
                    };
        }
    }
}
```

اکنون با توجه به این تغییرات، نیاز است تا Contract جدید را در Web.Config برنامه به شکل زیر معرفی و ویرایش نمائیم :

Web.config

```
...
<endpoint contract="SilverlightApplication60.Web.IProductServiceContract"
...
```

لازم به ذکر است که علاوه بر حالت سفارشی تعریف شده در Web.Config که بدون مشکل کار می‌کند، تنظیم ذیل نیز پشتیبانی می‌شود:

Web.config

```
...
<endpoint address="" binding="basicHttpBinding"
     contract="SilverlightApplication60.Web.IProductServiceContract" />
...
```

تا اینجا کار تعریف یک WCF Service قابل استفاده در Silverlight به پایان می‌رسد. این WCF Service لیستی از اشیاء را بازگشت می‌دهد و بدیهی است که می‌توان انواع و اقسام عملیات مرتبط با بانک‌های اطلاعاتی را نیز به این شکل مدیریت نمود و هدف از این مثال، نمایش ساختار کلی آن است.

ارائه فایل clientaccesspolicy.xml جهت WCF Service های خود میزبان

یکی از قابلیت‌های جالب WCF امکان میزبانی سرویس‌های آن توسط برنامه‌هایی غیر از IIS و یک ASP.NET Web site است. برای مثال استفاده از هر نوع برنامه‌ی تهیه شده با .NET. (یک برنامه Console ، WinForm ، WPF و غیره) و در بهترین حالت استفاده از یک سرویس ویندوز NT . در مورد دسترسی‌های بین Domains و ملاحظات آن در Silverlight نیز پیشتر مطالبی ارائه شد. اکنون سؤالی که مطرح است، این مورد می‌باشد که در حالات ذکر شده، فایل clientaccesspolicy.xml را کجا باید قرار داد؟ زیرا در این روش‌ها که اصطلاحا Self-Hosted WCF Services نیز نامیده می‌شوند، ریشه‌ی وب سایتی جهت قرار دادن فایل مذکور وجود ندارد. برای حل این مشکل، باید سرویسی را ارائه داد که بتواند به درخواست‌هایی که در آن‌ها نام فایل clientaccesspolicy.xml وجود دارد پاسخ داده و محتوای لازم را ارسال نماید. لطفا به مثال بعد دقت بفرمائید :

C#

```
[ServiceContract]
public interface ICrossDomainService
{
    [OperationContract]
    [WebGet(UriTemplate = "/clientaccesspolicy.xml")]
    Stream ProvidePolicyFile();
}

public class CrossDomainService : ICrossDomainService
{
    public Stream ProvidePolicyFile()
    {
        string result = @"<?xml version=""1.0"" encoding=""utf-8""?>
```

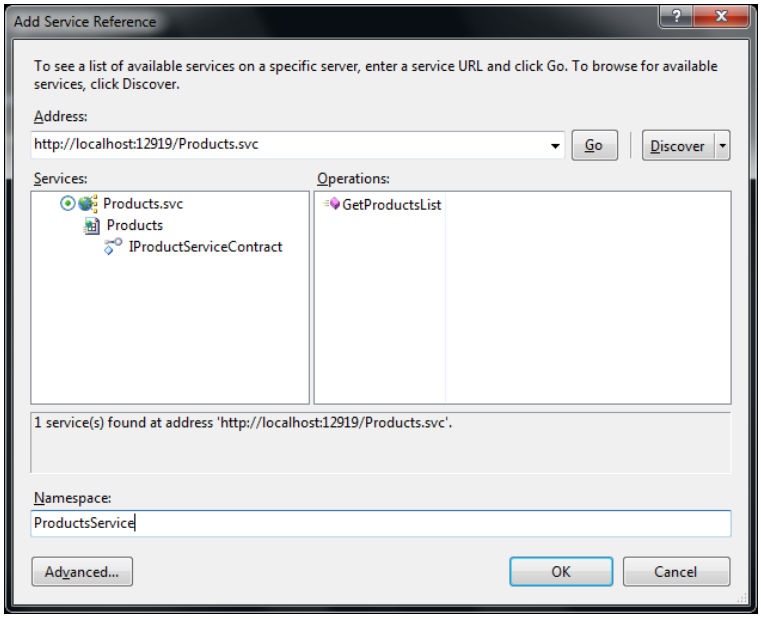
```
      <access-policy>
<cross-domain-access>
    <policy>
        <allow-from http-request-headers=""*"">
            <domain uri=""*""/>
        </allow-from>
        <grant-to>
            <resource path=""/"" include-subpaths=""true""/>
        </grant-to>
    </policy>
</cross-domain-access>
</access-policy>";
        return ConvertToStream(result);
    }

    private Stream ConvertToStream(string result)
    {
        WebOperationContext.Current.OutgoingResponse.ContentType =
              "application/xml";
        return new MemoryStream(Encoding.UTF8.GetBytes(result));
    }
}
```

ویژگی WebGet ذکر شده به همراه ServiceContract ، کار عکس العمل نشان دادن به درخواست‌هایی که نیاز به اطلاعات فایل clientaccesspolicy.xml را دارند می‌باشد. لازم به ذکر است که سرویس‌های سازگار با Silverlight باید از نوع BasicHttpBinding باشند، اما این سرویس ویژه باید بر اساس WebHttpBinding ارائه گردد.

نحوه‌ی استفاده از یک WCF Service در Silverlight

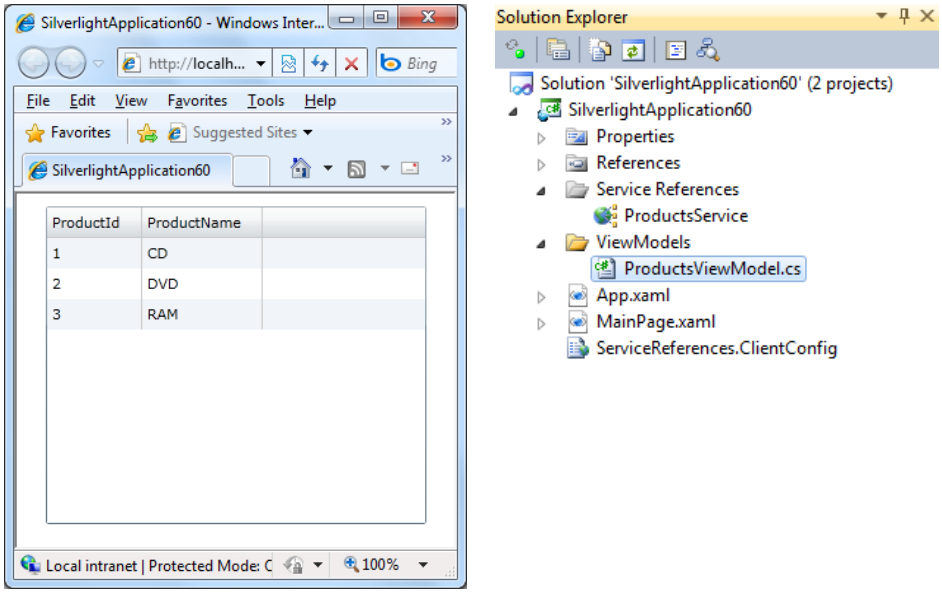
در ادامه به پروژه‌ی Silverlight مراجعه نموده و از منوی پروژه، گزینه‌ی Add service reference را انتخاب کنید. در صفحه‌ی باز شده روی دکمه‌ی Discover کلیک نمائید. این دکمه قابلیت تشخیص سرویس‌های تعریف شده در سطح پروژه‌ی جاری را دارد (شکل ۸).



شکل ۸– افزودن ارجاعی به WCF Service تعریف شده

در این صفحه نام ProductsService را وارد کرده و بر روی دکمه‌ی OK کلیک نمائید تا ارجاعات لازم به این WCF Service اضافه شوند.

همانطور که در مثال بکارگیری ASMX Web Services نیز عنوان شد، List<Product> تعریف شده در پروژه‌ی ASP.NET ، به صورت خودکار به ObservableCollection متناظری تبدیل شده و نیازی به کد نویسی اضافی در این مورد نخواهد بود.



شکل ۹– نمایی از ساختار پروژه‌ی بکارگیری WCF Services .

بدیهی همانند ASMX Web service ها، اگر پروژه‌ی Silverlight شما بر روی یک Domain و پروژه‌ی WCF Service تعریف شده، در Domain دیگری نصب و راه اندازی شوند (و یا حتی هر دو بر روی یک Domain اما با Port های متفاوت)، نیاز به تعریف فایل clientaccesspolicy.xml در ریشه‌ی سایت ارائه دهنده‌ی WCF Service خواهد بود (همانند توضیحاتی که پیشتر ارائه شد) .

این مثال را نیز با کمک الگوی MVVM پیاده سازی خواهیم نمود. برای این منظور پوشه‌ی ViewModels را به پروژه اضافه نمائید (شکل ۹) و فایل بعد را در آن تعریف کنید:

ProductsViewModel.cs

```
using System.Collections.ObjectModel;
using SilverlightApplication60.ProductsService;

namespace SilverlightApplication60.ViewModels
{
    public class ProductsViewModel
    {
        public ObservableCollection<Product> ProductsList { set; get; }
        public ProductsViewModel()
        {
            ProductsList = new ObservableCollection<Product>();
            loadData();
        }

        private void loadData()
        {
            var srv = new ProductServiceContractClient();
            srv.GetProductsListCompleted += srv_GetProductsListCompleted;
            srv.GetProductsListAsync();
        }

        void srv_GetProductsListCompleted(object sender,
            GetProductsListCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                //TODO: Show a general error msg
            }
            else
            {
                foreach (var product in e.Result)
                    ProductsList.Add(product);
            }
        }
    }
}
```

مدل برنامه یا همان شیء Product از فضای نام مرتبط با ProductsService اضافه شده دریافت می‌شود. این ViewModel ، شیء ProductsList را در اختیار View برنامه یا همان صفحه‌ی اصلی پروژه قرار می‌دهد.

روش فراخوانی WCF Services نیز بسیار شبیه به روش فراخوانی ASMX Web Services بوده و عملیات آن نیز در اینجا غیرهمزمان می‌باشد که در متد loadData ذکر گردیده است.

نکته‌ی جدید این مثال آرگومان دوم متد srv_GetProductsListCompleted است. این آرگومان توسط خاصیتی به نام Result ، نتیجه‌ی عملیات یا همان لیست اشیاء دریافت شده را باز می‌گرداند که به این ترتیب می‌توان اطلاعات لازم را از WCF Service دریافت و به شیء ProductsList افزود.

کدهای XAML متناظر با View برنامه در ادامه ذکر شده‌اند :

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication60.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"
    xmlns:vm="clr-namespace:SilverlightApplication60.ViewModels"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400"
    xmlns:sdk="http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk">
    <UserControl.Resources>
        <vm:ProductsViewModel x:Key="vmProductsViewModel" />
    </UserControl.Resources>
    <StackPanel x:Name="LayoutRoot" Margin="5"
          DataContext="{StaticResource vmProductsViewModel}"
          Background="White">
        <sdk:DataGrid AutoGenerateColumns="True"
                      ItemsSource="{Binding ProductsList}"
                      Height="250" Width="300"  Margin="5"  />
    </StackPanel>
</UserControl>
```

در این View ابتدا ViewModel برنامه در قسمت منابع صفحه تعریف گردیده و سپس به DataContext مربوط به StackPanel صفحه انتساب داده شده است.

همانطور که در فصل‌های قبل نیز ذکر شد، بهتر است جهت افزودن DataGrid موجود در Silverlight toolkit ، این کنترل را از جعبه ابزار VS.NET کشیده و بر روی فرم برنامه رها کرد تا ارجاعات لازم و همچنین تعریف فضای نام مرتبط با آن به صورت خودکار توسط VS.NET انجام شود.

خطاها و استثناهای حاصل از کار با Web Services

زمانیکه یک برنامه‌ی Silverlight با یک Web Service در حال تبادل اطلاعات است ممکن است در این حین، خطاها و استثناهایی نیز رخ دهند. این خطاها شامل موارد زیر خواهند بود:

- خطاهای ارتباطی در شبکه
- خطاهای امنیتی (به دلیل عدم وجود فایل‌های CrossDomain.xml و یاClientAccessPolicy.xml)
- خطاهای برنامه‌ی ارائه دهنده‌ی Web Service

دو خطای اول به صورت مشروحی توسط خاصیت e.Error دومین آرگومان دریافتی روال رخداد گردان پایان کار عملیات با Web Service قابل دریافت است. اما خطای سوم تحت هر شرایطی به خطای ۴۰۴ و Not found در برنامه‌ی Silverlight ختم خواهد شد که باید به این مورد دقت داشت.

مدیریت بهینه‌ی خطاهای یک WCF Service در یک برنامه‌ی Silverlight

روش‌های متعددی برای انتقال پیغام خطا از یک WCF Service به برنامه‌ی Silverlight موجود است که معروفترین آن‌ها به شرح زیر است (ارائه‌ی استثناء به صورت out parameter):

ابتدا یک کلاس جدید را جهت مشخص سازی ساختار شیء خطایی که به مصرف کننده ارائه خواهیم، تعریف می‌نمائیم (در اینجا جهت سهولت کار، همان مثال WCF مطرح شده در قسمت قبل مد نظر است):

MyFaultContract.cs

```
using System.Runtime.Serialization;

namespace SilverlightApplication60.Web
{
    [DataContract]
    public class MyFaultContract
    {
        [DataMember]
        public string FaultType { get; set; }

        [DataMember]
        public string Message { get; set; }
    }
}
```

سپس ServiceContract برنامه را مطابق کدهای زیر به نحوی اصلاح خواهیم نمود که بتوان خطای حاصل را از طریق out parameter تعریف شده، در سمت کاربر دریافت کرد:

IProductServiceContract.cs

```
using System.Collections.Generic;
using System.ServiceModel;

namespace SilverlightApplication60.Web
{
    [ServiceContract]
    public interface IProductServiceContract
    {
        [OperationContract]
        List<Product> GetProductsList(out MyFaultContract myFault);
    }
}
```

اکنون کدهای نهایی Products.svc.cs مثال WCF فصل را به شکل بعد تغییر خواهیم داد. با توجه به تغییرات جدید اینترفیس IProductServiceContract ، امضای متد GetProductsList تغییر نموده است. سپس عملیات اصلی متد مذکور به همراه یک try/catch ارائه گردیده است. زمانیکه خطایی رخ دهد، شیء myFault مقدار دهی خواهد شد. در اینجا جهت آزمایش عملیات، در قسمت try یک استثنای عمدی را تعریف کرده‌ایم:

Products.svc.cs

```
using System;
using System.Collections.Generic;
using System.ServiceModel.Activation;

namespace SilverlightApplication60.Web
{
    [AspNetCompatibilityRequirements(RequirementsMode
        = AspNetCompatibilityRequirementsMode.Allowed)]
    public class Products : IProductServiceContract
    {
        public List<Product> GetProductsList(
               out MyFaultContract myFault)
        {
            List<Product> result = null;
            myFault = null;

            try
            {
                result = new List<Product>
                    {
                        new Product
```

```
                                {
                                    ProductId = 1,
                                    ProductName = "CD"
                                },
                            new Product
                                {
                                    ProductId = 2,
                                    ProductName = "DVD"
                                },
                            new Product
                                {
                                    ProductId = 3,
                                    ProductName = "RAM"
                                }
                        };

                    //ex...
                    throw new NullReferenceException("Table not found");
                }
                catch (Exception ex)
                {
                    myFault = new MyFaultContract
                                {
                                    FaultType = ex.GetType().FullName,
                                    Message = ex.Message
                                };
                }

                return result;
            }
        }
}
```

اکنون به برنامه‌ی Silverlight مراجعه نمائید. در قسمت Service references پروژه‌ی جاری، بر روی ProductsService کلیک راست کرده و گزینه‌ی Update Service reference را انتخاب کنید تا مجددا تعاریف و ساختار جدید WCF Service مطابق تغییرات انجام شده دریافت شود. حال ViewModel برنامه به صورت زیر تغییر خواهد کرد و خطای حاصل را می‌توان دریافت نمود:

ProductsViewModel.cs

```
void srv_GetProductsListCompleted(object sender,
     GetProductsListCompletedEventArgs e)
{
   if (e.Error != null)
   {
       //TODO: Show a general error msg
   }
   else if(e.myFault!=null)
```

```
   {
      //TODO: log  e.myFault.FaultType + "\n" + e.myFault.Message;
   }
   else
   {
      foreach (var product in e.Result)
      {
         ProductsList.Add(product);
      }
   }
}
```

لازم به ذکر است که نمایش کامل متن خطای دریافت شده به کاربر از دیدگاه امنیتی به هیچ عنوان صحیح نبوده و در این حالت می‌توان امکان ارسال ایمیل محتوای آن خطا را به تیم فنی برنامه درنظر گرفت و مواردی از این دست. بنابراین در اینجا عدم ارسال جزئیات خطا به کاربر از دیدگاه امنیتی کاملا صحیح بوده و اینگونه جزئیات وقایع مرتبط با خطاها را باید در همان سمت سرور جهت بررسی‌های بعدی تیم فنی ثبت نمود.

تحت نظر قرار دادن ارتباط با شبکه در Silverlight

برای مدیریت بهتر خطاهای حالت اول (خطاهای ارتباطی در شبکه) می‌توان وضعیت ارتباطی شبکه را زیر نظر قرار داد. برای این منظور باید از کلاس NetworkChange مطابق کدهای بعد کمک گرفت:

MainPage.xaml.cs

```
using System;
using System.Net.NetworkInformation;

namespace SilverlightApplication60
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
            NetworkChange.NetworkAddressChanged += NetworkChanged;
        }

        private void NetworkChanged(object sender, EventArgs e)
        {
            if (NetworkInterface.GetIsNetworkAvailable())
            {
                // Currently online.
            }
```

```
            else
        {
            // Currently offline.
        }
    }
}
```

هر چند می‌توان وضعیت اتصال به شبکه را در این حالت تحت نظر قرار داد اما آنلاین بودن در این حالت به معنای اتصال به اینترنت نیست و همچنین هیچ الزامی هم ندارد که سایت ارائه دهنده‌ی Web Service هم اکنون مشغول به سرویس دهی باشد.