CHƯƠNG VII:
LÀM VIỆC VỚI DỮ LIỆU TRONG SILVERLIGHT

1 Các công nghệ để truy cập dữ liệu trong Silverlight

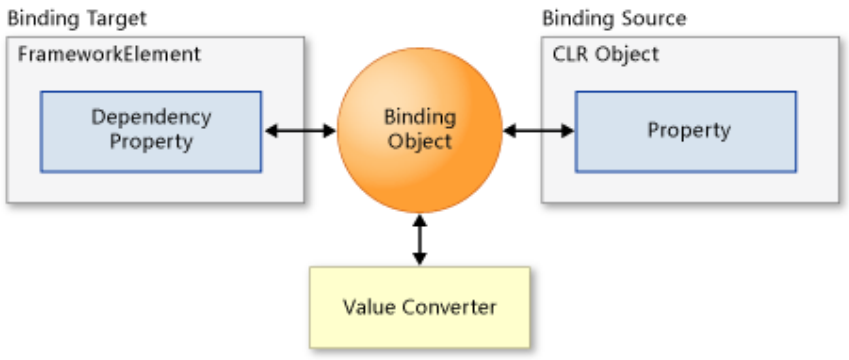
Trong các ứng dụng trên nền tảng Silverlight 2 có cho phép truy cập dữ liệu theo nhiều công nghệ hay phương thức khác nhau. Cách đơn giản nhất để hiển thị và tương tác người dùng là **Data Binding.** Cách khác nữa để truy cập dữ liệu từ ứng dụng máy khách, như là một sự thay thế tối ưu của cookie, chúng ta có thể sử dụng công nghệ **Isolated Storage.** Silverlight cũng cho phép đọc và ghi trên dữ liệu XML bằng cách sử dụng **XmlReader** hoặc **LINQ to XML** . Ngoài ra cách phổ biến trong các ứng dụng hiện nay thường làm đó là truy cập dữ liệu SQL Server thông qua các công nghệ **Web service**, **WCF** và **ADO.Net Data Service.**
Trong chương này chúng ta sẽ tìm hiểu một số công nghệ để truy cập dữ liệu sau:

- Data Binding
- Isolated Storage
- Xử lý dữ liệu XML với LINQ to XML
- Truy cập cơ sở dữ liệu SQL Server với Web Service(WCF)

2 Sử dụng Data Binding

Data Binding cung cấp một cách đơn giảm cho ứng dụng nền tảng silverlight truy cập và tương tác tới dữ liệu. Thông thường Data Binding quản lý dữ liệu theo luồng giữa người dùng và các đối tượng dữ liệu. Tức kà khi một binding được tạo và dữ liệu có thay đổi, thì giao diện người dùng cũng được trình bày và thay đổi theo dữ liệu(tự động ánh xạ). Tương tự khi người dùng thay đổi trên giao diện thì nó cũng có thể làm thay đổi đối tượng dữ liệu. Ví dụ như khi người dùng thay đổi giá trị trên một TextBox, giá trị dữ liệu cũng tự động cập nhật theo thay đổi đó.

- **Sự liên kết giữa giao diện người dùng và dữ liệu**



- Binding Object: Như là một xúc tác ở giữaBindingTarget và Binding Source
- Binding Source: Chứa các thuộc tính dữ liệu, còn có thể là đối tượng của CLR
- Binding Target: Là các thuộc tính giao diệu người dùng để có thể hiển thị hoặc thay đổi tới dữ liệu. Binding target có thể là DependencyProperty của FramworkElement

- Value Converter: Thành phần tùy chọn để áp dụng cho những trường hợp dữ liệu cần phân tích hoặc chuyển đổi
- Điều hướng của luồng dữ liệu: Quyết định tới việc tương tác dữ liệu qua lại giữa Binding Source và Binding Target (xem chi tiết mục dưới)

Ví dụ dưới đây sẽ chỉ rõ cách trình bày thuộc tính màu của chữ (ForeGround Color) của một TextBox:

Mã XAML:

```
<TextBox x:Name="MyTextBox" Text="Text" Foreground="{Binding Brush1,
Mode=OneWay}"/>
```

Mã C# :

```
//Tạo một lớp MyColors đã thực thi giao tiếp INotifyPropertyChanged.
MyColors textcolor = new MyColors();

// Brush1 thiết lập một SolidColorBrush với giá trị màu Red.
textcolor.Brush1 = new SolidColorBrush(Colors.Red);

// Thiết lập DataContext của MyTextBox.
MyTextBox.DataContext = textcolor;
```

Khi sử dụng DataContext trong mã C# trên đây, nó cho phép các thẻ UI thừa kế thông tin về dữ liệu nguồn. Ví dụ giống như trong ASP.NET bạn sử dụng phương thước gán DataSouce để hiển thị dữ liệu trên một Grid.

Các phương thức điều hướng của luồng dữ liệu

Mỗi một kiểu binding data đều có một phương thức điều hướng luồng dữ liệu(Mode propery). Cái này quyết định khi nào và làm thế nào để dữ liệu chảy tràn. Có 3 phương thức điều hướng luồng dữ liệu sau

- **OneTime:** binding dữ liệu tới target (giao diện người dùng) một lần và sau đó kết thúc kết nối không binding nữa. Phương thức này phù hợp khi hiển thị dữ liệu mà ít khi hoặc không bao giờ cần thay đổi.
- **OneWay:** Binding dữ liệu tới target và giữ nó đến khi Source (nguồn dữ liệu) thay đổi thì target cũng thay đổi theo. Phương thức này phù hợp khi trình bày dữ liệu mà người dùng không được phép thay đổi
- **TwoWay:** Binding dữ liệu tới target và giữ nó đến khi source thay đổi thì target cũng thay đổi theo, nhưng khi target thay đổi thì source cũng thay đổi theo. Phương thức này phù hợp khi trình bày dữ liệu mà cho phép người dùng thay đổi dữ liệu nguồn

Change Notification

Để cho hợp lệ với các thay đổi giá trị giữa Source object và Target objects thì chúng ta phải thực thi một interface là InotifyPropertyChanged. Trong InotifyPropertyChanged có cung cấp một event PropertyChanged.

Trong ví dụ dưới đây. Class MyColors thực thi giao tiếp InotifyPropertyChanged cho phương thức OneWay

```
namespace DataBinding
{
    //Tao mot class thua ke interface INotifyPropertyChanged
    public class MyColors : INotifyPropertyChanged
    {
        private SolidColorBrush _Brush1;
```

```
        // Khai bao su kien PropertyChanged.
        public event PropertyChangedEventHandler PropertyChanged;

        //Tao thuoc tinh cua SolidColorBrush
        public SolidColorBrush Brush1
        {
            get { return _Brush1; }
            set
            {
                _Brush1 = value;
                // Goi NotifyPropertyChanged khi thuoc tinh nguon duoc cap nhap
                NotifyPropertyChanged("Brush1");
            }
        }
        public void NotifyPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
            }
        }
    }
}
```

3 Sử dụng Isolated Storage

Silverlight dùng Isolated Storage như một hệ thống file ảo để lưu trữ dữ liệu trong một thư mục ẩn trên máy tính của bạn. Nó chia dữ liệu vào làm 2 phân riêng biệt: Phần thứ nhất chứa các thông tin quản lý như dung lượng cho phép và phần thứ 2 chứa dữ liệu thực sự. Mỗi ứng dụng Silverlight được phân bố một vùng riêng trong hệ thống lưu trữ này với dung lượng lưu trữ mặc nhiên là 1 MB.

- **Ưu điểm:**
  - Isolated Storage là một thay thế tuyệt vời cho cookie, đặc biệt là nếu bạn đang làm việc với một tập lớn dữ liệu. Một số ví dụ về khả năng của Isolated Storage bao gồm undo lại một số thao tác của ứng dụng, hay giỏ hàng, các cài đặt về cửa sổ hay bất kỳ cài đặt nào mà bạn muốn truy cập vào lần nạp tiếp theo.
  - Isolated Storage lưu trữ theo người dùng cho phép các ứng dụng lưu giữ các cài đặt cho từng người dùng riêng biệt.
- **Một số vấn đề có thể xảy ra:**
  - Người quản trị có thể đặt giới hạn đĩa trên mỗi người dùng và assembly, không có bất kỳ cảnh báo nào về không gian trống còn lại. Vì lý do này, bạn nên kiểm tra và bắt các ngoại lệ trong ứng dụng của bạn.
  - Dù rằng Isolated Storage được đặt trong một thư mục ẩn trong hệ thống, nhưng không có nghĩa người dùng không thể tìm ra thư mục này, do vậy Isolated Storage không hoàn toàn an toàn và người dùng có thể thay đổi hoặc xóa các file. Tuy nhiên, nếu muốn người dùng không thể thay đổi các file, bạn có thể dùng các lớp mã hóa để mã lại các file này trước khi lưu.
  - Các máy tính có thể bị khóa bởi các chính sách bảo mật và các ứng dụng sẽ không thể lưu được vào IsolatedStorage. Cụ thể hơn, bạn cần có IsolatedStorageFilePermission để có thể làm việc được với IsolatedStorage

Giờ hãy xem qua ví dụ về việc lưu và đọc dữ liệu từ IsolatedStorage. Nhớ rằng bạn cần dùng using để tham chiếu đến namespace System.IO.IsolatedStorage cũng như là System.IO.

```
using System.Windows.Controls;
using System.IO.IsolatedStorage;
using System.IO;
using System;


namespace Samples_Chap7
{
    public partial class Page : UserControl
    {
        public Page()
        {
            InitializeComponent();
            //Luu du lieu vao file dulieu.txt
            SaveData("Day la du lieu cua toi", "dulieu.txt");
            //Lay du lieu tu file dulieu.txt
            string test = LoadData("dulieu.txt");
        }
        private void SaveData(string data, string fileName)
        {
            //Lay Isolated Storage cua nguoi dung danh cho ung dung
            using (IsolatedStorageFile isf =
IsolatedStorageFile.GetUserStoreForApplication())
            {
                using (IsolatedStorageFileStream isfs = new
IsolatedStorageFileStream(fileName, FileMode.Create, isf))
                {
                    using (StreamWriter sw = new StreamWriter(isfs))
                    {
                        //Luu du lieu
                        sw.Write(data);
                        sw.Close();
                    }
                }
            }
        }

        private string LoadData(string fileName)
        {
            string data = String.Empty;
            //Lay Isolated Storage cua nguoi dung danh cho ung dung
            using (IsolatedStorageFile isf =
IsolatedStorageFile.GetUserStoreForApplication())
            {
                using (IsolatedStorageFileStream isfs = new
IsolatedStorageFileStream(fileName, FileMode.Open, isf))
                {
                    using (StreamReader sr = new StreamReader(isfs))
                    {
                        string lineOfData = String.Empty;
                        while ((lineOfData = sr.ReadLine()) != null)
                            //Doc du lieu
                            data += lineOfData;
                    }
                }
            }
            return data;
        }

    }
}
```

4 Khái quát về làm việc với dữ liệu XML, LINQ to XML

Trong Silverlight bạn có thể phân tích dữ liệu XML bằng một trong hai cách là LINQ to XML hoặc XmlReader
Đối với những tài liệu XML lớn đòi hỏi việc tải dữ liệu về tốn nhiều bộ nhớ trong làm giảm hiệu suất chương trình. Trong trường hợp này chúng ta nên sử dụng XmlReader với những đặc tính như đọc dữ liệu nhanh, forward-only, non-caching parser. Ngược lại với những tài liệu XML nhỏ hơn bạn nên sử dụng LINQ to XML, nó cung cấp những tính năng và tiện ích hữu dụng hơn XmlReader
Trong phần này chúng tôi sẽ hướng dẫn các bạn về cách xử lý dữ liệu xml với LINQ to XML(Silverlight).

○ Khái quát về LINQ to XML

LINQ to XML cũng giống như Document Object Model(DOM) trong đó nó đem tài liệu XML này vào trong bộ nhớ. Bạn có thể truy vấn và thay đổi tài liệu này, sau khi thay đổi nó bạn có thể lưu nó và một file hoặc chuyển đổi hóa nó (Serialize) rồi gửi nó thông qua mạng Internet. Tuy nhiên LINQ to XML khác biệt so với DOM ở chỗ: Nó cung cấp một đối tượng mới nhẹ nhàng hơn và dễ dàng làm việc với chúng hơn, ngoài ra LINQ to XML còn là sự cải tiến của ngôn ngữ Visual C# 2008.
Một lợi thế quan trọng nhất của LINQ to XML là nó tích hợp với Language-Integrated Query (LINQ). Với sự tích hợp này nó cho phép bạn viết câu truy vấn tài liệu XML khi ở trong bộ nhớ
để trả về tập hợp các Element và attribute. LINQ to XML có thể so sánh được với các chức năng của XPath và Xquery.

○ Tạo một XAML động với LINQ to XML

Trong mục này chúng ta sẽ chỉ rõ các tạo một control TextBlock bằng LINQ to XML.
Trước tiên để tùy chỉnh project silverlight trong Visual Studio chúng ta chạy ví dụ dưới đây

1. Trong Solution Explorer, thêm assembly reference tới System.Xml.Linq.dll

2. Thay đổi nội dung file page.xaml của bạn theo mã dưới đây

```
<UserControl x:Class="SilverlightApplication1.Page"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Canvas x:Name="LayoutRoot" Background="White">
    </Canvas>
</UserControl>
```

3. Trong file page.xaml.cs thêm các khai báo using sau đây

```
using System.Windows.Markup;
using System.Xml.Resolvers;
using System.Xml;
using System.Xml.Linq;
using System.IO;
using System.Text;
```

Dưới đây là ví dụ chi tiết về tạo một đối tượng XElement và lưu nó với thông tin liên hệ.

```
  // Tao XElement de luu thong tin lien he ban be
            XElement contacts =
                new XElement("Contacts",
                    new XElement("Contact1",
                        new XElement("Ten", "Pham Chi Cuong"),
                        new XElement("DienThoai", "0906 123 480"),
                        new XElement("DiaChi",
                            new XElement("DuongPho", "Nguyen Hong"),
```

```
                            new XElement("ThanhPho", "HaNoi")
                        )
                    ),
                    new XElement("Contact2",
                        new XElement("Ten", "Tran Duy Bien"),
                        new XElement("DienThoai", "0904 252 161"),
                        new XElement("DiaChi",
                            new XElement("DuongPho", "Pham Van Dong"),
                            new XElement("ThanhPho", "HaNoi")
                        )
                    )
                );

            // Tao TextBlock1
            // Luu y rang Element nay phai khai bao 2 XAML namespace
            XElement textBlock1 = XElement.Parse(
                    @"<TextBlock
                    xmlns='http://schemas.microsoft.com/client/2007'
                    xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml'
                    TextWrapping= 'Wrap'
                    Width = '400'
                    Canvas.Top = '10'
                    Text=''/>");

            // Lay child Element cua contact1
            XElement contact1 = contacts.Element("Contact1");

            // Gan gia tri vao thuoc tinh Text o cuoi cung voi noi dung cua cac
contact trong xml
            textBlock1.LastAttribute.SetValue(contact1.ToString());
            // Lay child element thu hai
            XElement contact2 = contacts.Element("Contact2");

            // Tao TextBlock2
            // Luu y rang Element nay phai khai bao 2 XAML namespace
            XNamespace xmlns = "http://schemas.microsoft.com/client/2007";
            XElement textBlock2 = new XElement(xmlns + "TextBlock",
                new XAttribute(XNamespace.Xmlns + "x",
"http://schemas.microsoft.com/winfx/2006/xaml"),
                new XAttribute("Canvas.Top", 250),
                new XAttribute("Width", "600"),
                new XAttribute("Text", contact2.ToString())
                );


            // Them TextBlock1 vao trong trang

            LayoutRoot.Children.Add(XamlReader.Load(textBlock1.ToString()) as
UIElement);
            // Them TextBlock2 vao trong trang
            LayoutRoot.Children.Add(XamlReader.Load(textBlock2.ToString()) as
UIElement);
```

- **Cách load file XML từ một URI bất kỳ**

1. Tạo một đối tượng WebClient, thêm trình xử lý (handler), Gọi phương thức OpenReadAsync. Làm theo mã lệnh dưới đây

```
    WebClient wc = new WebClient();
    wc.OpenReadCompleted += wc_OpenReadCompleted;
    wc.OpenReadAsync(new Uri(uriString));
```

2. Thực thi hàm callback **wc_OpenReadCompleted**

```
private void wc_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
//Kiem tra thuoc tinh Error cho cac loi
     if (e.Error != null)
     {
        OutputTextBlock.Text = e.Error.Message;
        return;
     }
//Neu khong co loi, Lay du lieu stream ve va phan tich chung toi XDocument thong
qua phuong thuc Load
    using (Stream s = e.Result)
    {
       XDocument doc = XDocument.Load(s);
       OutputTextBlock.Text = doc.ToString(SaveOptions.OmitDuplicateNamespaces);
    }
}
```

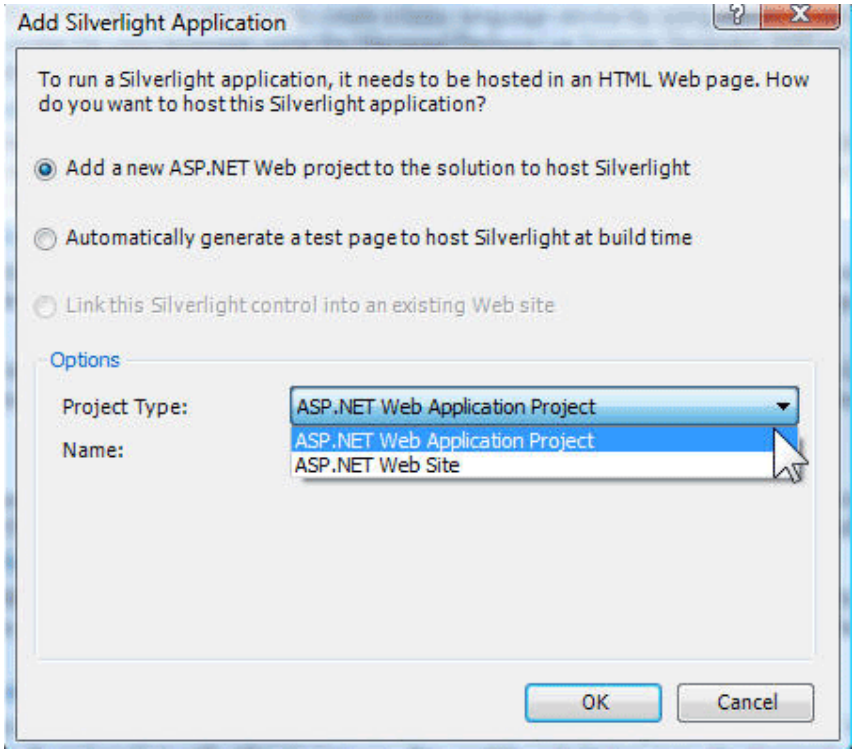
5 Truy cập dữ liệu SQL Server với WCF

Trong phần này chúng ta sẽ xây dựng một ứng dụng nhỏ để kết nối với CSDL SQL Server. Để hoàn thiện được ứng dụng này chúng ta cần tập hợp được những kỹ năng sau

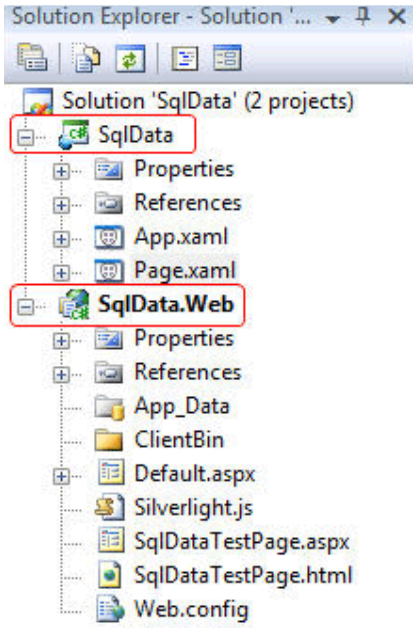
- Kết nối tới một WCF Web Service
- Sử dụng LINQ để truy vấn và trả về dữ liệu
- Sử dụng Control DataGrid để hiển thị dữ liệu

○ Khởi tạo chương trình

Tạo một Project Web Application (File -> New ->Project -> Silverlight Application) . Chúng ta đặt tên cho ứng dụng này là SQLData



Bấm OK chúng ta sẽ có một Solution như hình vẽ dưới đây



- **Tìm hiểu về LINQ to SQL**

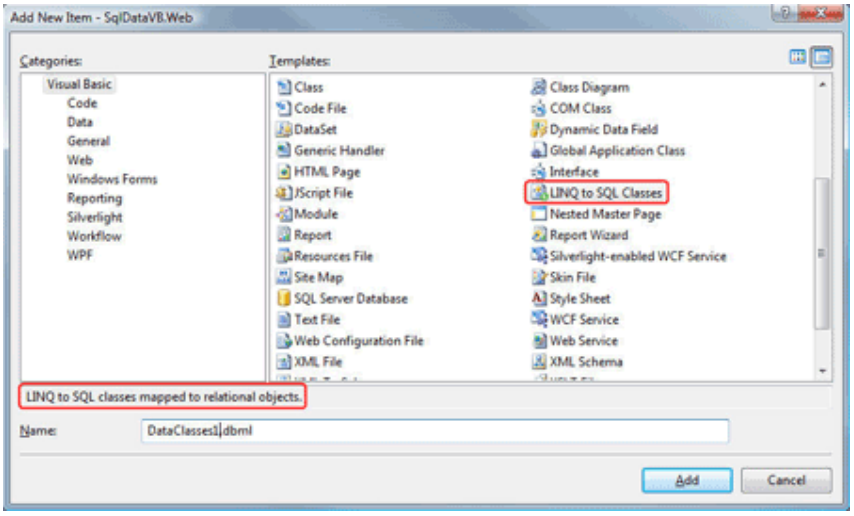
LINQ to SQL là một phiên bản hiện thực hóa của O/RM (object relational mapping) có bên trong .NET Framework bản "Orcas" (nay là .NET 3.5), nó cho phép bạn mô hình hóa một cơ sở dữ liệu dùng các lớp .NET. Sau đó bạn có thể truy vấn cơ sở dữ liệu (CSDL) dùng LINQ, cũng như cập nhật/thêm/xóa dữ liệu từ đó.

LINQ to SQL hỗ trợ đầy đủ transaction, view và các stored procedure (SP). Nó cũng cung cấp một cách dễ dàng để thêm khả năng kiểm tra tính hợp lệ của dữ liệu và các quy tắc vào trong mô hình dữ liệu của bạn.

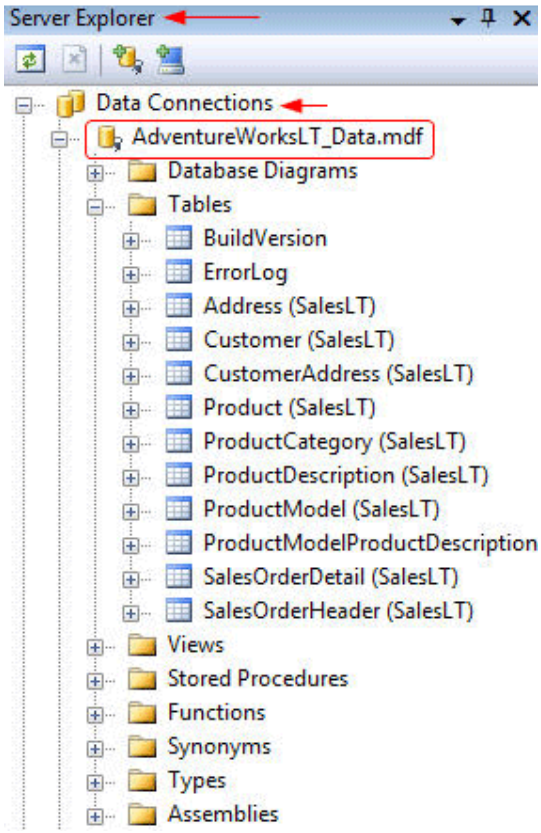
Để biết thêm về công nghệ này bạn có thể xem chi tiết tại ScottGu's tutorial

- **Thêm LINQ to SQL Classes**

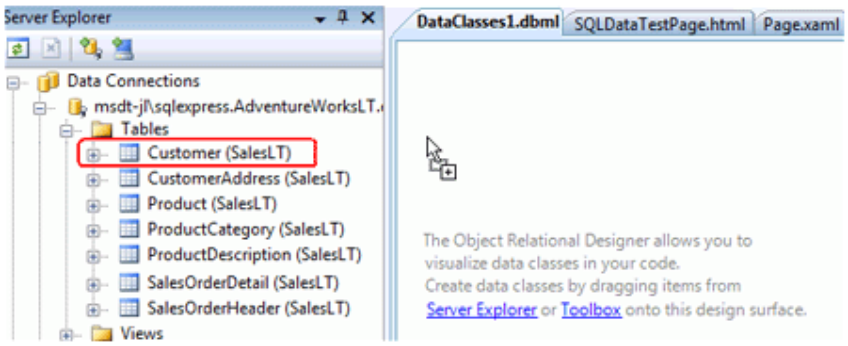
- Bằng cách chuột phải vào Server Project và chọn Add -> New Item sau đó chọn LINQ to SQL Classes. class này có tên là **DataClasses1.dbml**



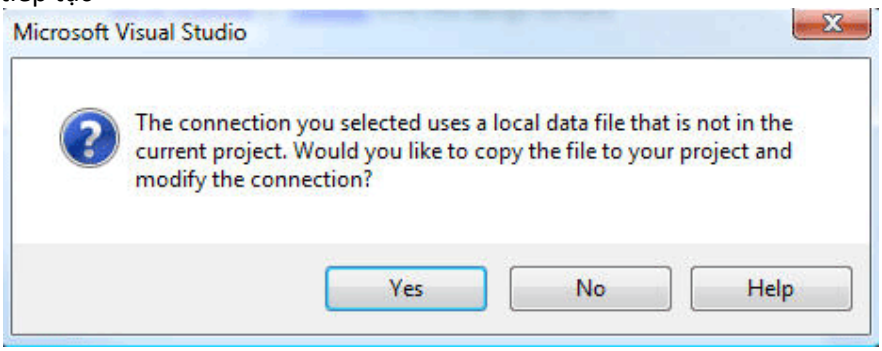
- Chúng ta mở Server Explorer (View->Server Explorer) và chọn cơ sở dữ liệu AdventureWorkLTs (đã cài đặt sẵn trong SQL Server hoặc có thể download ở Microsoft.com)



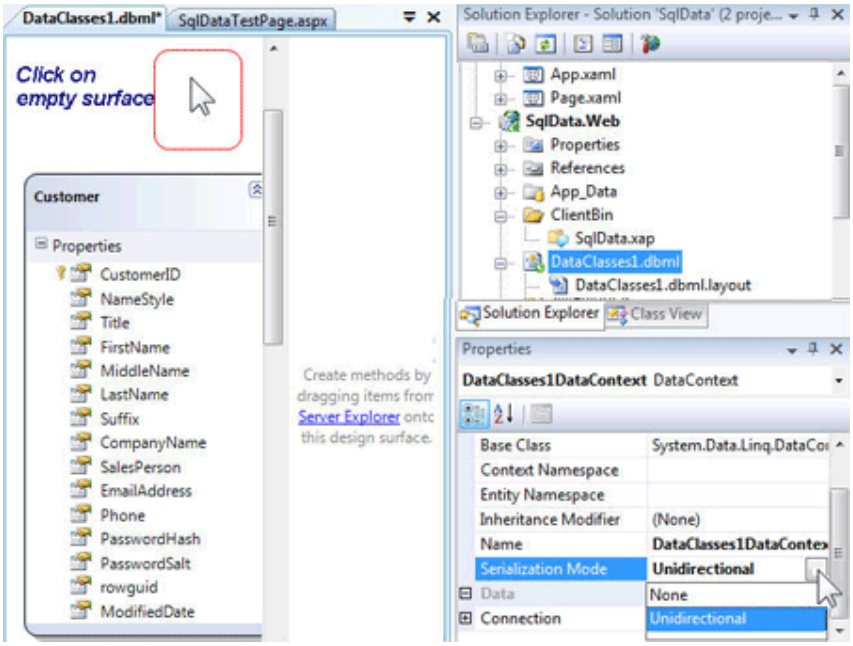
- Sau khi đã mở kết nối tới CSDL AdventureWorkLTs, chúng ta tiếp tục kéo bảng Customer vào DataClasses1.dbml trong đã mở trong Designer



- Nhiều khả năng sẽ bạn sẽ nhận được cảnh báo như hình vẽ dưới đây, bạn bấm vào Yes để tiếp tục

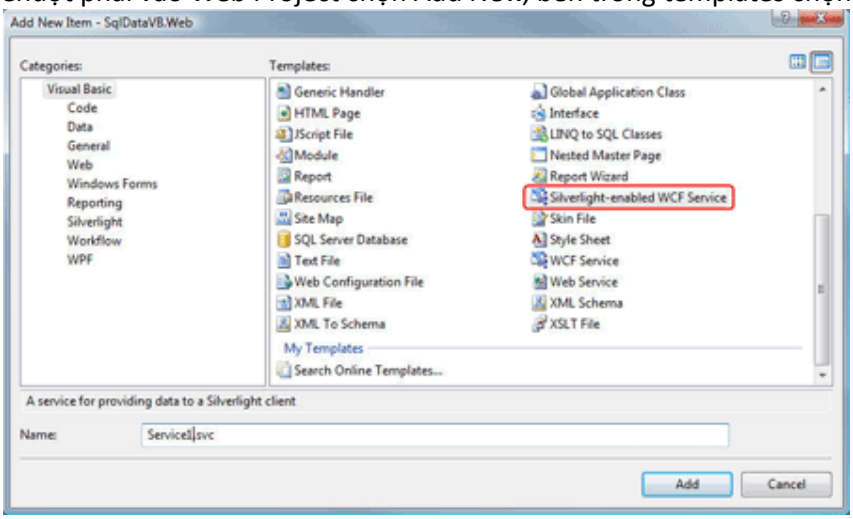


- Sau khi kéo vào vùng trống của DataClasses1.dbml sẽ nhìn thấy được khả năng chuyển hóa từ đối tượng bảng (Customer table) sang mô hình lớp như hình vẽ dưới đây

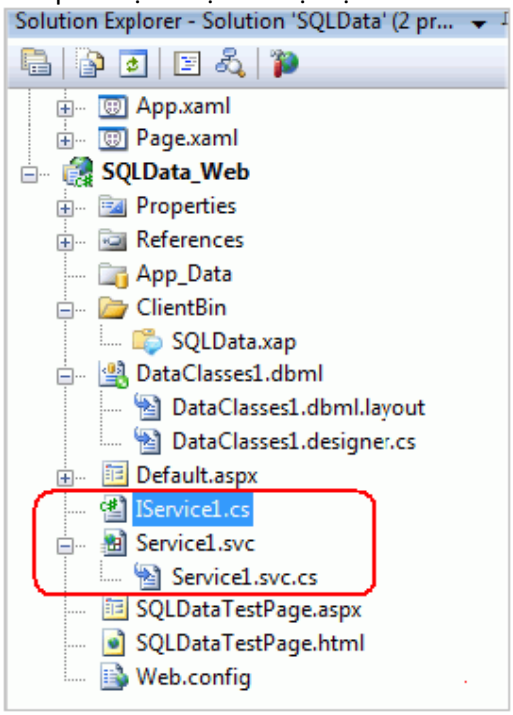


- **Tạo Web Service (WCF)**

  - Chuột phải vào Web Project chọn Add New, bên trong templates chọn WCF Service



- Kết quả nhận được từ việc tạo trên là 3 file theo hình vẽ dưới đây



- Mở file IService1.cs bạn sẽ thấy đoạn mã đã được tạo săn dưới đây

```
        public interface IService1
        {
            [OperationContract]
            void DoWork();
        }
```

- Thay đổi đoạn mã trên bằng đoạn mã dưới đây

```
    public interface IService1
    {
       [OperationContract]
        List<Customer> GetCustomersByLastName(string lastName);
    }
```

- Mở file Service1.svc, thêm viết đoạn mã như dưới đây

```
        public List<Customer> GetCustomersByLastName(string lastName)
        {
            DataClasses1DataContext db = new DataClasses1DataContext();
            var matchingCustomers = from cust in db.Customers
                                    where cust.LastName.StartsWith(lastName)
                                    select cust;
            return matchingCustomers.ToList();
        }
```

- Lưu ý Mặc định của WCF sử dụng wsHttpBinding, xem trong file Web.config

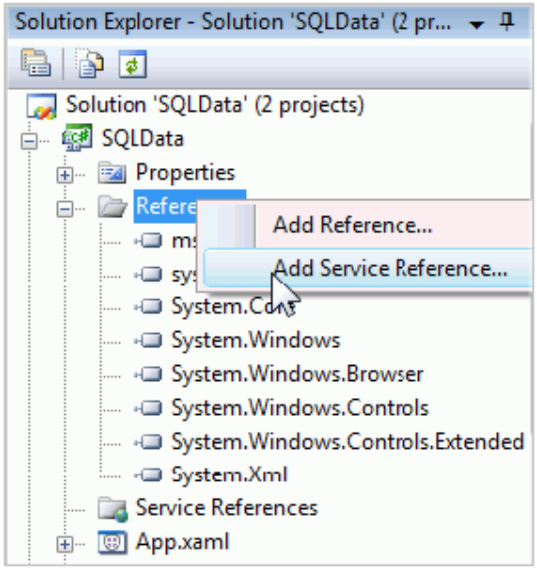
```
<services>
    <service behaviorConfiguration="SQLData_Web.Service1Behavior"
                name="SQLData_Web.Service1">
          <endpoint address="" binding="wsHttpBinding"
                    contract="SQLData_Web.IService1">
                <identity>
                      <dns value="localhost"/>
                </identity>
          </endpoint>
    </service>
</services>
```

- Tuy nhiên nó chỉ hỗ trợ trong basic binding (SOAP 1.1, etc.), vì vậy bạn cần thay đổi binding tương ứng như sau

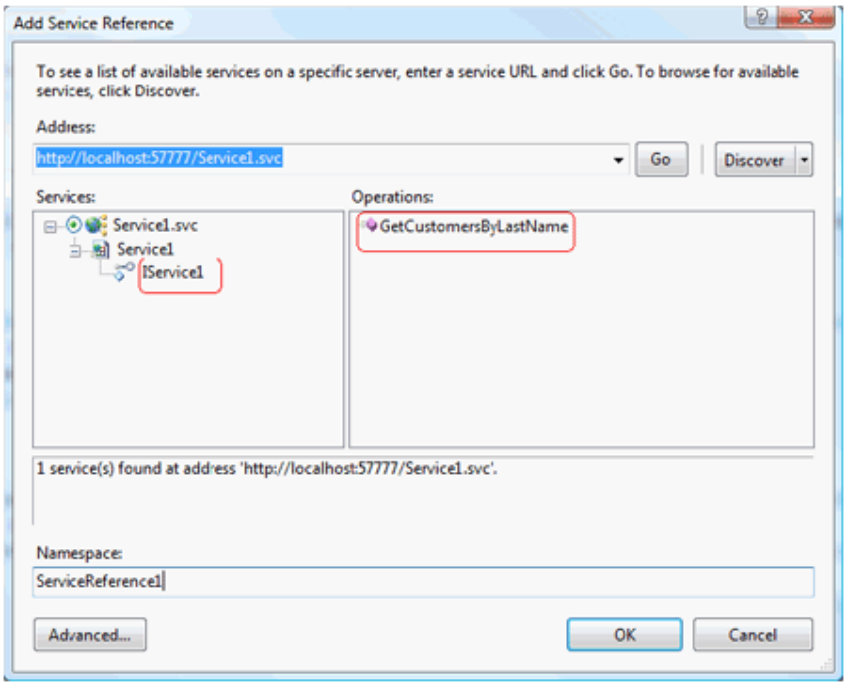
```
<endpoint address="" binding="basicHttpBinding"
contract="SQLData_Web.IService1">
```

○ Tạo ứng dụng Silverlight

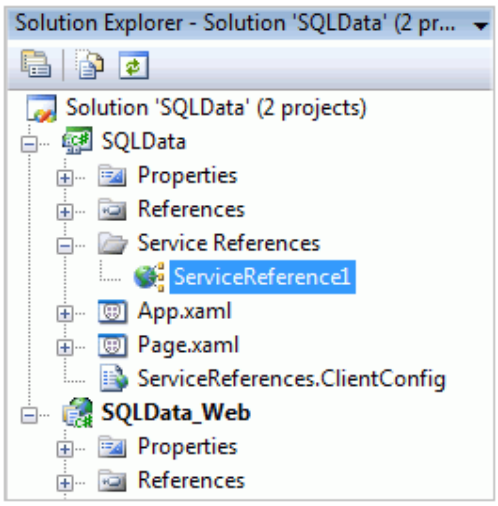
- Tiếp theo chúng ta sẽ tạo một ứng dụng Silverlight và tương tác với web service WCF vừa tạo được. Chuột phải vào references trong silverlight project và chọn Add Service Reference



- Sau khi chọn Add Service Reference, bạn bấm chuột vào Discover và chọn Services in Solution. Service mà bạn vừa tạo sẽ được tìm thấy và hiện thị. Bấm OK



- Service sau khi đã được thêm vào project của bạn, bạn có thể truy cập vào Web service này.



- **Tạo XAML**

- Trong file Page.xaml bạn thêm đoạn mã dưới đây

```
<Grid x:Name="LayoutRoot" Background="White" ShowGridLines="True">
    <Grid.RowDefinitions>
        <RowDefinition Height="10" />
        <!--0 Margin-->
        <RowDefinition Height="50" />
        <!--1 Prompts-->
        <RowDefinition Height="*" />
        <!--2 DataGrid-->
        <RowDefinition Height="10" />
        <!--3 Margin-->
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="10" />
        <!--0 Margin-->
        <ColumnDefinition Width="*" />
        <!--1 Controls-->
        <ColumnDefinition Width="10" />
        <!--2 Margin-->
    </Grid.ColumnDefinitions>
</Grid>
```

- Bên màn hình Designer sẽ thấy như hình ảnh sau

- Đặt các Control sau đây vào hàng thứ nhất của Datagrid

```
        <Border BorderBrush="Black" BorderThickness="2" Grid.Row="1"
Grid.Column="1"/>
        <StackPanel Grid.Row="1" Grid.Column="1" Orientation="Horizontal">
            <TextBlock Text="Last name to search for: " VerticalAlignment="Bottom"
FontSize="18" Margin="15,0,0,0" />
            <TextBox x:Name="LastName" Width="250" Height="30" Margin="2,0,0,4"
VerticalAlignment="Bottom"/>
            <Button x:Name="Search" Width="75" Height="30"      Margin="20,0,0,4"
Content="Search"    VerticalAlignment="Bottom"  Background="Blue" FontWeight="Bold"
FontSize="14" />
        </StackPanel>
```

- Đặt các Control sau đây vào hàng thứ nhất của Grid
- Kéo thả DataGrid trong toolbox vao trong XAML

```
<my:DataGrid x:Name="theDataGrid" AlternatingRowBackground="Beige"
AutoGenerateColumns="True" Width="700" Height="500" Grid.Row="2" Grid.Column="1"
CanUserResizeColumns="True"  />
```

- **Viết xử lý sự kiện cho nút Search**

- Mở file page.xaml.cs thêm đoạn mã dưới đây

```
public Page()
{
   InitializeComponent();
    Loaded += new RoutedEventHandler(Page_Loaded);
}

void Page_Loaded(object sender, RoutedEventArgs e)
{
   Search.Click += new RoutedEventHandler(Search_Click);
}

void Search_Click(object sender, RoutedEventArgs e)
{
    //Gan Service1Client toi mot doi tuong o local là webService
```

```
      ServiceReference1.Service1Client webService = new
SQLData.ServiceReference1.Service1Client();

    //Cai dat trinh xu ly su kien
webService.GetCustomersByLastNameCompleted += new
EventHandler<SQLData.ServiceReference1.
GetCustomersByLastNameCompletedEventArgs>(webService_GetCustomersByLastNameComplete
d);

    //Goi asynchronous
webService.GetCustomersByLastNameAsync(LastName.Text);


void webService_GetCustomersByLastNameCompleted(object sender,
SQLData.ServiceReference1.GetCustomersByLastNameCompletedEventArgs e)
{
   //Trinh bay ket qua vao Datagrid
    theDataGrid.ItemsSource = e.Result;
}
```

- **Kết quả hiển thị**