

Silverlight 2 チュートリアル

チュートリアル #4 SQL データベース データを LINQ および WCF を使用して DataGrid に表示する

執筆者 : [Jesse Liberty](#)

SQL データベースからのデータへのアクセス

最初の 3 つのチュートリアルの読者の多くは、DataBinding を使用して SQL データベースにデータへのバインドを使用する方法に関する質問を記入しました。このチュートリアルでは、Web サービスを作成して SQL データにアクセスし、LINQ を使用してバインド可能なデータ ソースを作成します。バインド先とするコントロールは DataGrid です。

新しいスキル

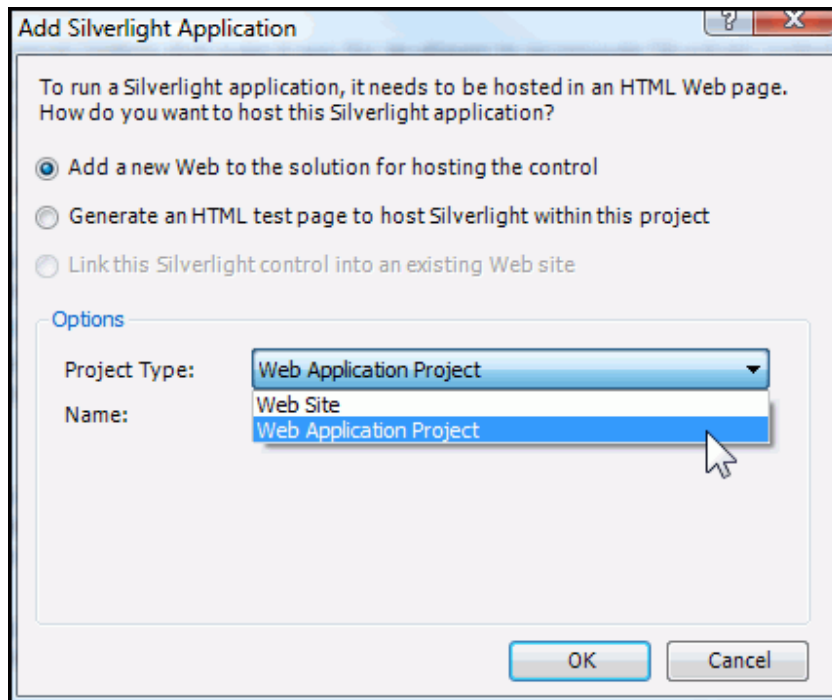
構築するアプリケーションは、多数のさまざまな新しいスキルを組み合わせたものになります。

- WCF Web サービスへの接続
- Silverlight アプリケーションが使用できるデータを、LINQ を使用してクエリおよび取得する
- DataGrid コントロールを使用してデータを表示する

Silverlight を使用するには Web サービスの構築と LINQ の使用は関連スキルですが、このチュートリアルでは取り上げません。とはこのものの、Silverlight とどのように相互作用するかについては確認します。

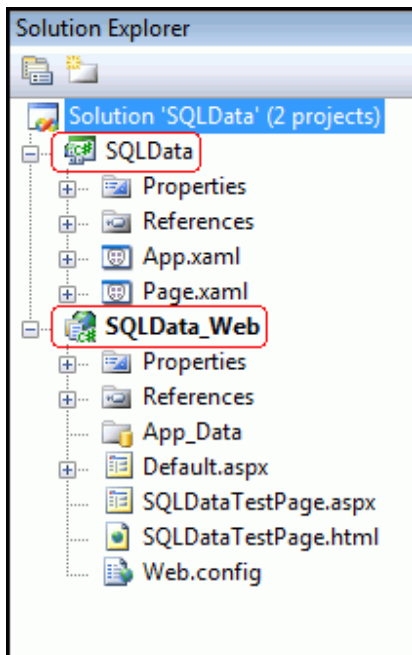
はじめに

最初に、SQLData という名前のプロジェクトを作成しますが、WCF Web サービス (データベースへの接続のため) を作成できる Silverlight プロジェクトと Server プロジェクトの両方が必要であるため、*Web アプリケーション プロジェクト* を選択してください。



2つのプロジェクトの確認

Visual Studio 2008 は、1つのソリューションの下に2つのプロジェクトを作成します。



ソリューションと最初のプロジェクトには、SQLData という名前が付いています。その最初のプロジェクトは Silverlight アプリケーションで、前のチュートリアルで確認したのと同じファイルがあります。

2 番目のプロジェクトの SQLData_Web は、Silverlight プロジェクトのテスト環境として作成され、3 つの潜在的なエントリ ポイントがあります。

- Default.aspx
- SQLDataTestPage.aspx
- SQLDataTestPage.html

SQLDataTestPage.aspx は、具体的に Silverlight コントロールをテストするために設計され、ざっと確認してみると、ソースが Silverlight プロジェクトで作成される .xap ファイル (発音はザップ ファイル) である AJAX ScriptManager および ASP:Silverlight コントロールが含まれていることがわかります。

```
<form id="form1" runat="server" style="height:100%;">
  <asp:ScriptManager ID="ScriptManager1" runat="server"/>
  <div style="height:100%;">
    <asp:Silverlight ID="Xaml1" runat="server" Source="~/ClientBin/SQLData.xap"
    Version="2.0" Width="100%" Height="100%" />
  </div>
</form>
```

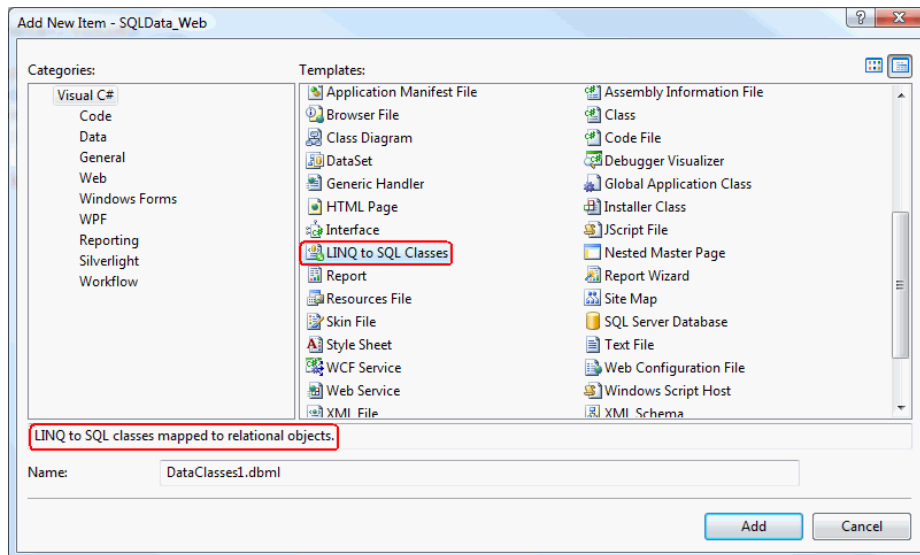
LINQ to SQL

LINQ は、VB 9 と C# 3 の両方に対する非常に強力な追加で、Silverlight および前進する他の .NET テクノロジーのデータ取得の中心的技法になる可能性があります。

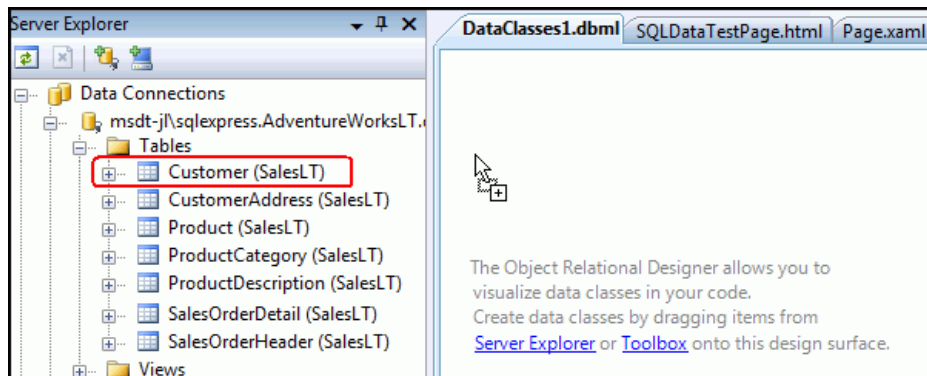
LINQ の手始めとしては、<http://tinyurl.com/28q63z> で利用可能な ScottGu の優れたチュートリアルである [ScottGu のチュートリアル](#) を使用することです。また、LINQ については多くの本があります。

このチュートリアルでは、解析を行う簡単な LINQ クエリを記述しています。

最初にサーバー プロジェクトで右クリックし、[Add (追加)] を選択して LinqToSql Classes テンプレートを選択します。ウィンドウの下にある説明に「リレーショナル オブジェクトにマップされた LINQ to SQL クラス」とあることに注意します。

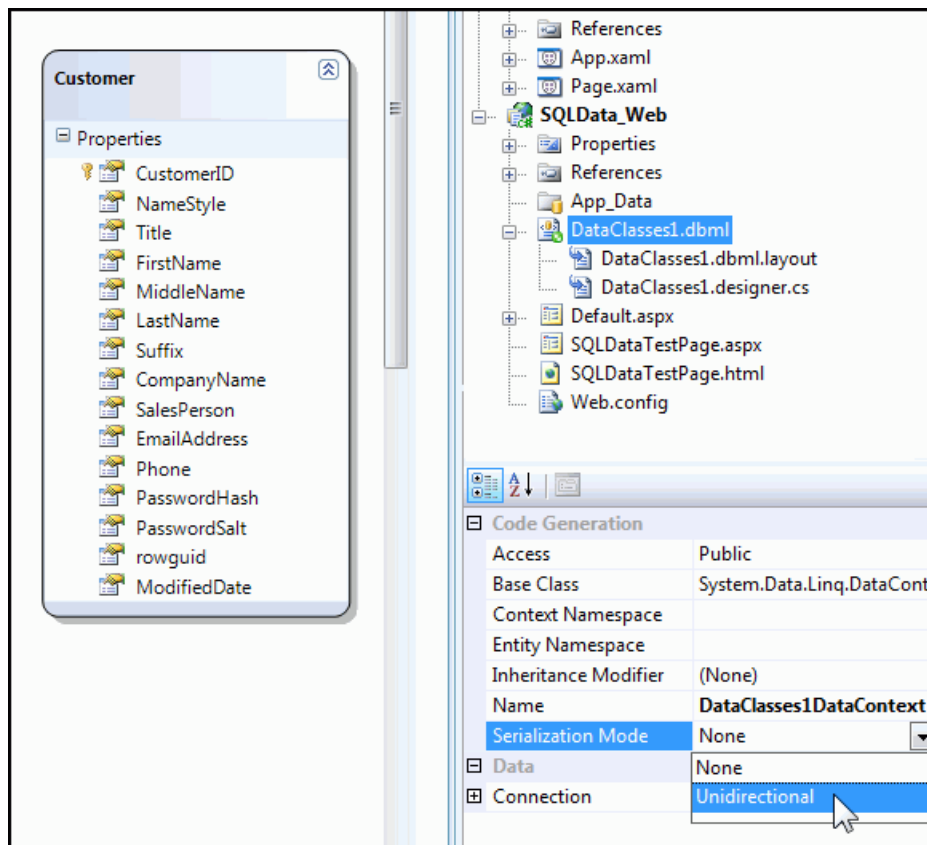


[Object Relational Designer (オブジェクト リレーショナル デザイナ)] ウィンドウが開いたら、サーバー エクスプローラを開き、AdventureWorks データベース (SQL Server とともにインストール済み、または Microsoft.com で利用可能) に移動します。表を展開して表示し、Customer テーブルを DataClasses1.dbml デザイナ ワークスペースにドラッグします。



結果の LINQ クラスをシリアル化可能にする

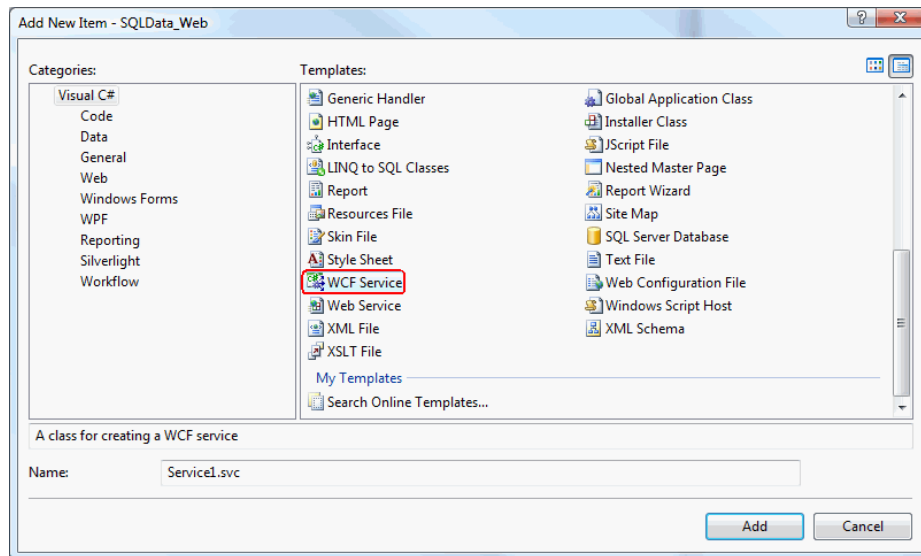
Customer テーブルに対応して LINQ クラスが生成される一方で、既定ではそのクラスはシリアル化可能ではなく、変更の必要がある Web サービスで利用可能になります。デザイン サーフエイスをクリックし、クラス全体のプロパティを開いて、シリアル化モードを [None (なし)] から [Unidirectional (一方向)] に設定します。



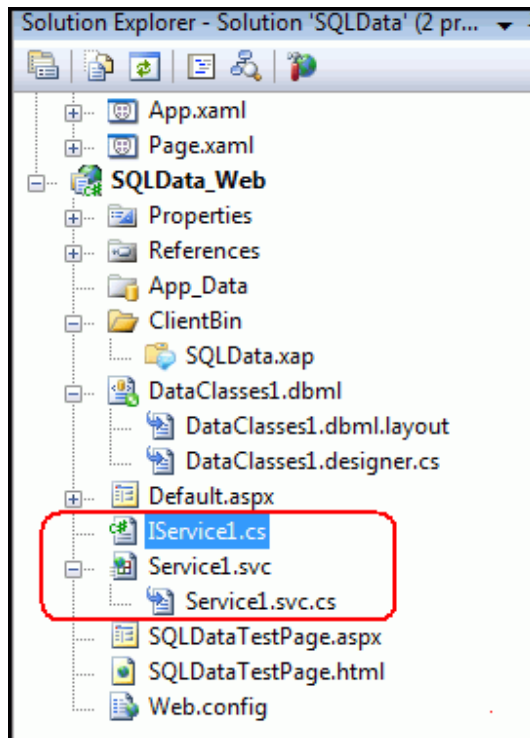
Web サービスの作成

Web サービス (および IntelliSense) が Customer クラスとそのメンバを識別できるように、まず LINQ クラス (クエリではなく) を作成しました。すべて整った状態で、Visua Studio 2008 に Web サービスを作成させます。

テスト プロジェクトを右クリックし、[Add New (新規追加)] を選択して、テンプレートから WCF サービスを選択します。



結果として、WCF Web サービスのためのサービス コントラクトが保存された 3 つの新しいファイルが作成されます。



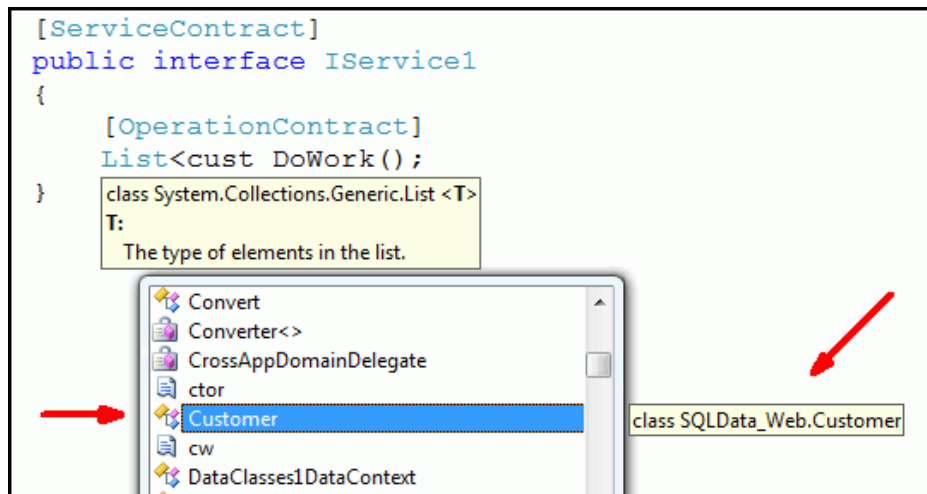
.NET 2 プロキシ ベース Web サービスに慣れている場合は、少し調整を行ってください。WCF は、基本の XML を詳細に表示し、より多くのコントラクト ベースのプログラミング モデルを提供します。WCF の詳細については、<http://tinyurl.com/ymxjqk> を参照してください。

Visual Studio 2008 で作成されたコントラクトを含む最初のファイル、IService1.cs を開きます。

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

自分の Web サービスで提供したいコントラクトとは関係なく、この「ダミー」コントラクトに置き換えることができます。このチュートリアルの目的として、Web サービスが、顧客の姓の始めの文字を表す文字列を指定する Customer オブジェクトの一覧を返すコントラクトを希望します。

したがって、メソッドの機能を、無効を返すことから List<Customer> を返すことに変更します。ただし、戻り値の型の変更を開始すると同時に、IntelliSense は助けとなるポップアップを表示できるようになります。これは特に、これまでに定義した LINQ クラスでこの型を作成したためです。



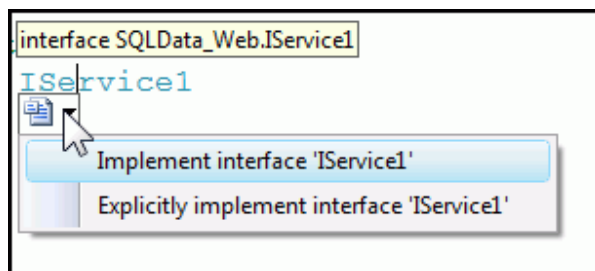
バーを指定する foo のセットを返すメソッドの規約は、GetFoosByBar という名前を付けることです。したがって、ここで GetCustomersByLastName という名前を付けます。

```

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

```

インターフェイスでコントラクトを変更したため、.cs ファイルでの実装の変更を確認する必要があります。しかし、それは大変な作業です。cs ファイルを開くとき、インターフェイスをクリックすれば、スマート タグが表示されます。タグを開くと、実装スケルトンの作成が表示されます。



DoWork メソッドはそのままにして、GetCustomersByLastName に LINQ クエリを入れます。

```

public class Service1 : IService1
{
    #region IService1 Members
    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();
}
#endregion
}
```

(スマート タグに実装スケルトンを作成をさせたとき、領域コメントが入力されています。)

LINQ 構文

最後に LINQ について説明します。LINQ の学習で難しい点は、[Anders](#) の記事を参照し、説明を読んでみます。彼は第一人者です。Anders の記事を参照できない場合は、[Scott のチュートリアル](#)、[C# 3.0 の本](#)の LINQ の章、または LINQ に関する優れた本が多数ありますので、そちらを参照してください。

1 つの LINQ ステートメントを取り上げてみましょう。まず、新しい var インターフェイス変数を使用しますが、驚くことにこれは型 safe です (この型を暗示しています。決して型がないわけではありません)。LINQ クエリの結果を割り当てますが、これは enumerable 型のオブジェクトとなります。

クエリ構文は、Select ステートメントが最後に配置されること以外は、SQL 構文に似ています。そこで、英語では、「これまでにお話したデータベースに接続し、その接続に db という名前を付けます」となります。そのデータベースを開き、customers という名前のテーブルを見つけ、文字列 lastName に保持されている文字で LastName フィールドが始まる各レコードを探します。一致するレコードをすべて提示してください。これらすべてのレコードをオブジェクト matchingCustomers に割り当てますが、(a) タイプ enumerable になることがわかっている、(b) そこで ToList() を呼び出すと List<Customer> が返されることがわかっている、よい方法といえます。

バインディングに注意

Web.config ファイルでわかるように、WCF は wsHttpBinding を既定のバインディングとして使用します。

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

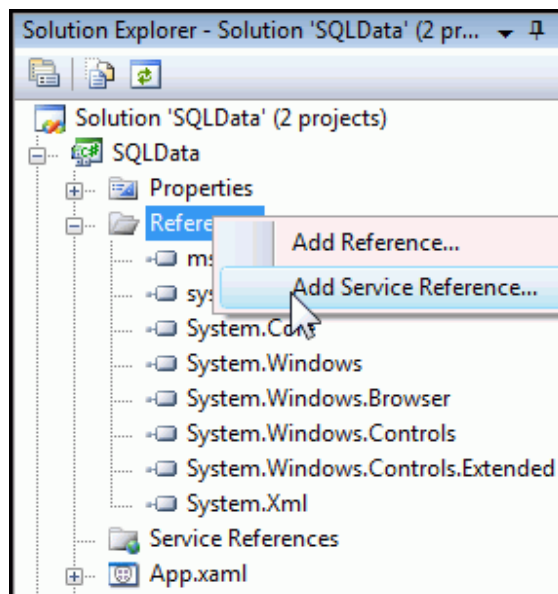
ただし Silverlight は基本的なバインディング (SOAP 1.1 など) のみをサポートするので、バインディングを変更する必要があります。

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

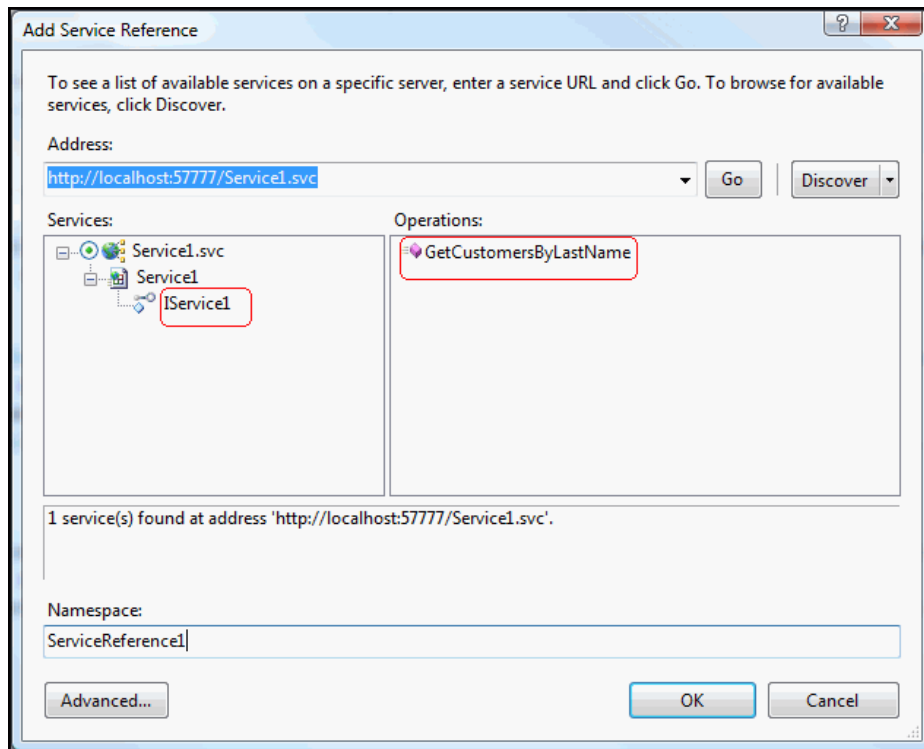
これで、サービスを開始する準備ができました。

Silverlight アプリケーションの作成

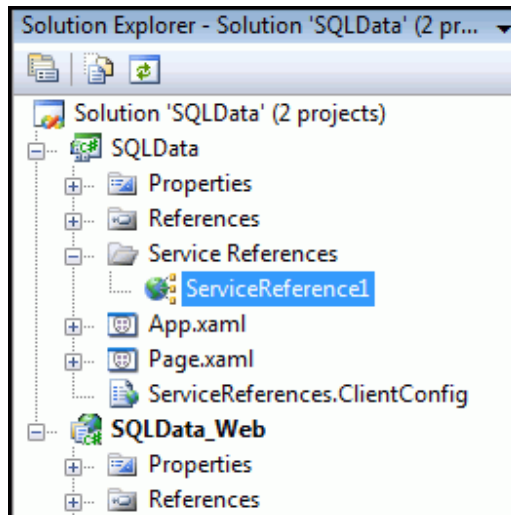
次の手順では、この Web サービスと対話する Silverlight アプリケーションを作成します。これを作成するには、Silverlight プロジェクトで [References (参照)] を右クリックし、[Add Service Reference (サービス参照の追加)] を選択します。



[Add Service Reference (サービス参照の追加)] が開いたら、[Discover (検出)] をクリックし、*ソリューションのサービス*を選択します。作成したサービスが見つかります。[OK] をクリックする前に [Service (サービス)] をクリックし、作成した操作 (GetCustoemrByLastName) が見つかることを確認します。



[OK] をクリックし、プロジェクトにサービスを追加します。この参照で Web サービス (およびそのメソッド) にアクセスします。



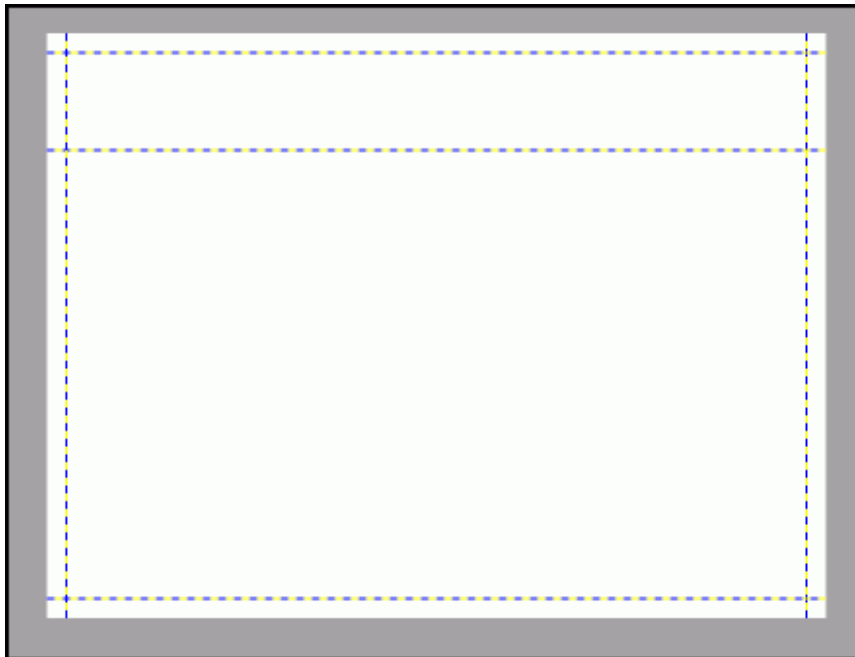
XAML の作成

Page.xaml で、ユーザーの姓を入力する上端行と結果を表示する最下行で構成される、非常に簡単な UI を作成します。初めに、グリッドの行と列をレイアウトします。

```
<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>
```

ShowGridLines を true に設定しましたが、予想どおりの結果を得られるように、3 番目の行と 2 番目の列はスター サイズ指定を使用し、残りのスペースすべてを利用するようにしています。

グリッドの周囲には小さい余白と、小さい上端行と大きい最下行の 2 行があります。



上端行へのコントロールの配置

上端行に、Textblock (プロンプト用) と TextBox (入力用) およびボタンを 1 つ配置したいと思います。最も簡単な方法は、スタック パネルを使用する方法で、結果と区別するため境界線で囲むことです。

```
<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>
```

最後に、DataGrid をツールボックスから XAML にドラッグします。

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

接頭辞 my が付けられ、新しい名前空間がサポートを宣言されていることがわかります。

```
xmlns:my="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
```

検索ボタンのイベント ハンドラを記述する

ユーザーが検索ボタンをクリックしたとき、テキスト ボックスのテキストを取り出し、そのテキストを Web サービスに渡して、顧客のコレクションを返すようにしたいと思います。スケルトン イベント処理コードを page.xaml.cs に設定しましょう。

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

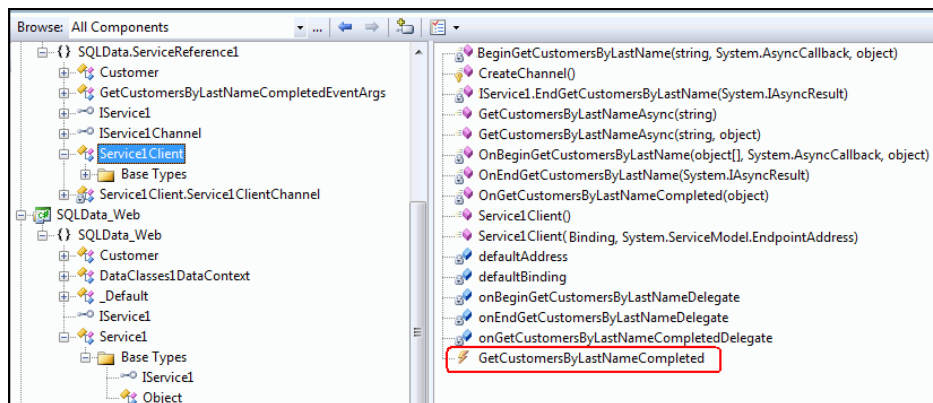
void Page_Loaded(object sender, RoutedEventArgs e)
{
  Search.Click += new RoutedEventHandler(Search_Click);
}

void Search_Click(object sender, RoutedEventArgs e)
{
}
```

サービスの非同期呼び出し

Silverlight から Web サービスを呼び出す唯一の方法は、非同期呼び出しです (ブラウザで実行され、ブロックするわけにはいかないため、当然です)。

最初の作業として、Web サービスの Service1Client メンバへの参照を取得します。これは、オブジェクト ブラウザで確認できます。それには、このオブジェクトが必要な非同期メソッドを持つことを確認します。



(スペースの関係から、図は若干簡略化されています)

Service1Client をローカル オブジェクト webService に割り当てます。

```
void Search_Click(object sender, RoutedEventArgs e)
{
    ServiceReference1.Service1Client webService =
    new SQLData.ServiceReference1.Service1Client();
}
```

次に、WebService を使用して、GetCustomersByLastNameCompleted イベントが呼び出されたときに呼び出されるメソッドのイベント ハンドラを設定します。

```
webService.GetCustomersByLastNameCompleted +=
new EventHandler<SQLData.ServiceReference1.
GetCustomersByLastNameCompletedEventArgs>
(webService_GetCustomersByLastNameCompleted);
```

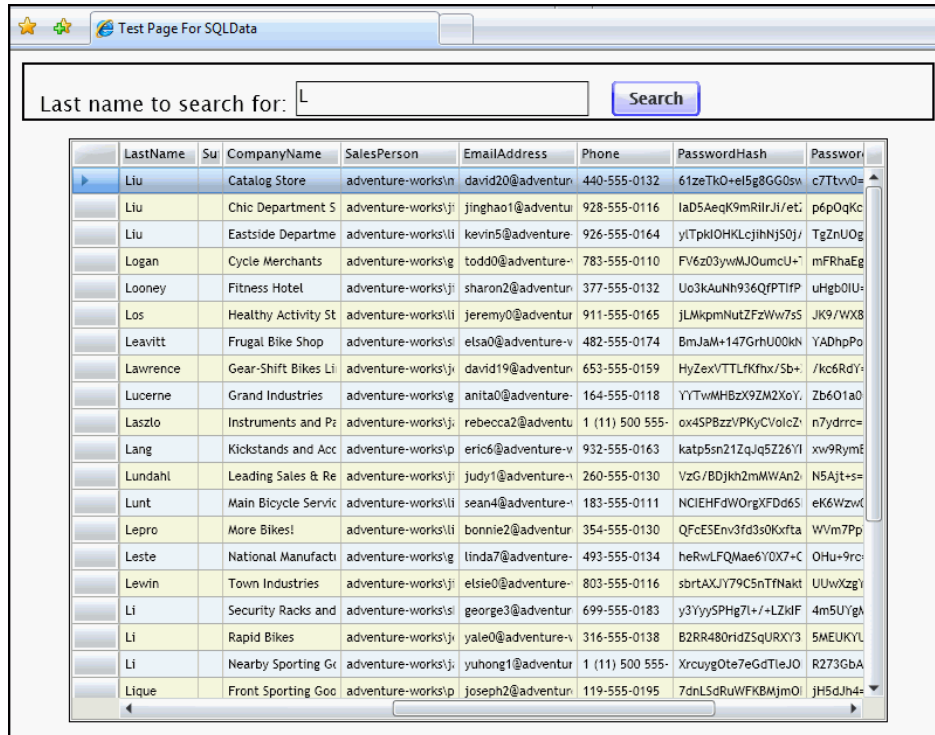
最後に、非同期呼び出しを行います。

```
webService.GetCustomersByLastNameAsync(LastName.Text);
}
```

サービスが完了したら、GetCustomersByLastNameCompleted イベントが発生し、メソッドが呼び出されます。注意深く構築されたこの Customers の一覧は、DataGrid の ItemSource プロパティに割り当てる e.Result に格納されます。これで、すべてのバインディングに、バインド先のソースが設定されました。

```
void webService_GetCustomersByLastNameCompleted(
```

```
object sender,  
SQLData.ServiceReference1.GetCustomersByLastNameCompletedEventArgs e)  
{  
  
    theDataGrid.ItemsSource = e.Result;  
}
```



簡単ですね。

方法がわかれば、スタイルの使用や希望どおりの外観を実現するのも、長くても数時間で出来ます。