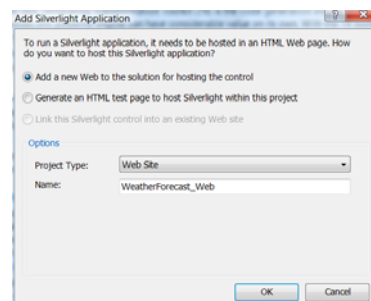
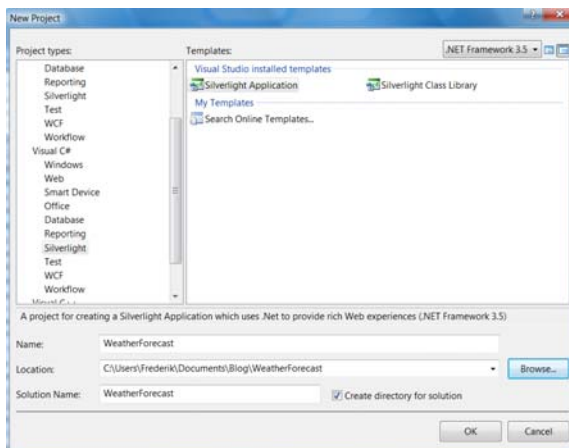


Weather forecast (part 1)

I will create a small application that offers the weather forecast for a certain city in the USA. I will consume two webservices for this. The first service will give me an overview of all the cities with their state and zip code. The second service offers the weather forecast for that zip code.

Step 1: creating the project

- Open Visual Studio 2008 and choose File → New Project → Visual C# → Silverlight → Silverlight Application.
- Give your application a name and a location and click ok.
- Visual Studio will ask if you like to create a new web project to host the Silverlight content. Just click ok, we will use this web project to call a webservice.



Step 2: use the zipcode webservice

The website <http://www.websvcicx.net/WCF/default.aspx> has a variety of services. For example the service <http://www.websvcicx.net/uszip.asmx> offers a GetInfoByCity method which returns all the cities with that name plus their zip code and state.

The problem with this service is that it returns a collection of System.Xml.XNode elements.

Silverlight can't work with this type, so we have to consume the service in the web project in a WCF service and consume this WCF service in the Silverlight project.

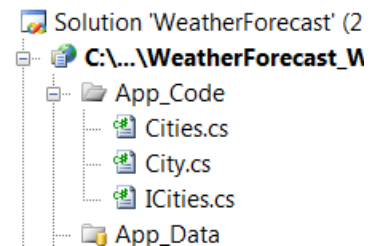
- First we will change the port number of the web project from dynamic to static. Select the web project and set the property "use dynamic ports" in the property panel from true to false. Select the project again and set the port number to 8081.

Always Start W True	
Full Path	C:\Users\Frederik\Documents\Blog\WeatherForecast_Web
Opened URL	file:///C:/Users/Frederik/My Documents/Blog/WeatherForecast_Web/WeatherForecast_Web.silverlight
Port number	8081
Use dynamic ports	False
Virtual path	/WeatherForecast_Web

- Add a new class City.cs to the web project (allow to place it in the App_Code folder) and insert the following code:

```
// add this namespace, so you can use DataContract and DataMember attributes
using System.Runtime.Serialization;

[DataContract]
public class City
{
    [DataMember]
    public string ZipCode { get; set; }
    [DataMember]
    public string Name {get; set;}
    [DataMember]
    public string State { get; set; }
}
```



- Right click the web project and add a new WCF Service with the name Cities.svc. You will see that Visual Studio has created a class Cities.cs and an interface ICities.cs for you.

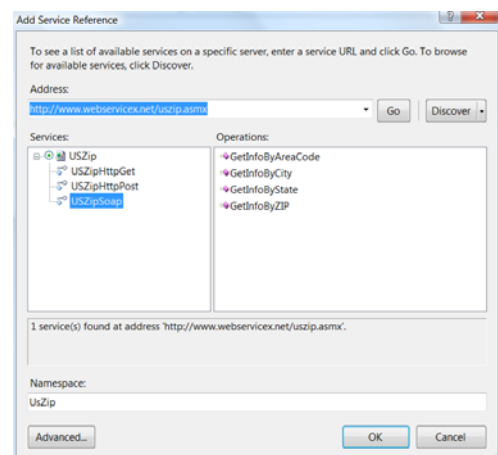
- Open the file ICities.cs and change the code to the following:

```
// add this namespace, so you can use the List<> class
using System.Collections.Generic;

[ServiceContract]
public interface ICities
{
    [OperationContract]
    List<City> getCitiesByName (string name) ;
}
```

Open the file Cities.cs. In this class, we will call the webservice and return a list of City objects.

- Right click the web project and choose Add Service Reference.
- Fill in the address: <http://www.webservicex.net/uszip.asmx> and click go
- Fill in UsZip for the namespace.
- Go to the web.config file and change the attribute binding under the service node from wsHttpBinding to basicHttpBinding
- Change the code in Cities.cs and change it to the following and build your web project



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using UsZip;

public class Cities : ICities
{
    public List<City> getCitiesByName(string name)
    {
        // creating the variables
```

```

USZipSoapClient proxy;
List<City> cityList = new List<City>();
BasicHttpBinding binding = new BasicHttpBinding();
EndpointAddress endpoint = new
EndpointAddress("http://www.webservices.net/uszip.asmx");
// create a new soap client for the binding and the endpoint
proxy = new USZipSoapClient(binding, endpoint);
// get all the information as an XmlNode
System.Xml.XmlNode node = proxy.GetInfoByCity(name);
// loop through all the childnodes and create a new instance of City for each
childnode
foreach (System.Xml.XmlNode child in node.ChildNodes)
{
    // add every instance to the list
    cityList.Add(new City { Name = child.FirstChild.InnerText, ZipCode =
    child.ChildNodes[2].InnerText, State = child.ChildNodes[1].InnerText});
}
// return the list
return cityList;
}
}
}

```

Step 3: test the webservice in the Silverlight project

- Right click the Silverlight project and choose “Add Service Reference”. Click the discover button. Visual Studio should normally find the service we created in the web project. Insert UsZip for the namespace and click ok.
- Go to the Page.xaml.cs file and insert the following code after the InitializeComponent method in the constructor

```

using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

using System.ServiceModel;
using System.ServiceModel.Channels;
using WeatherForecast.UsZip;

namespace WeatherForecast
{
    public partial class Page : UserControl
    {
        public Page()
        {
            InitializeComponent();
            // creating a new basicHttpBinding
            Binding binding = new BasicHttpBinding();
            EndpointAddress endPoint = new
            EndpointAddress("http://localhost:8081/WeatherForecast_Web/Cities.svc");
            CitiesClient client = new CitiesClient(binding, endPoint);
            // eventhandler that will be executed when all the cities are downloaded
            client.getCitiesByNameCompleted += new
            System.EventHandler<getCitiesByNameCompletedEventArgs>(client_getCitiesByNameCompleted);
            // get all the cities with the name "Redmond" ( for example )
            client.getCitiesByNameAsync("Redmond");
        }

        void client_getCitiesByNameCompleted(object sender,
        getCitiesByNameCompletedEventArgs e)
        {
        }
    }
}

```

- Place a breakpoint at the `client_getCitiesByStateCompleted` method and run the project. If Visual Studio asks you to modify `web.config` to enable debugging, check the modify option (first radiobutton) and click ok
- When the breakpoint is reached, you can check the content of `e.Results` in the watch panel and should see this:

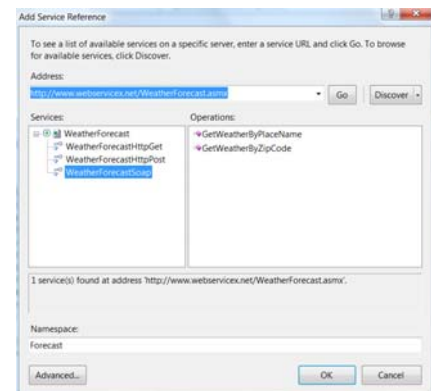
Name	Value
e.Result	{WeatherForecast.UsZip.City(5)}
[0]	{WeatherForecast.UsZip.City}
[1]	{WeatherForecast.UsZip.City}
[2]	{WeatherForecast.UsZip.City}
Name	"Redmond"
NameField	"Redmond"
PropertyChanged	null
State	"WA"
StateField	"WA"
ZipCode	"98052"
ZipCodeField	"98052"
[3]	{WeatherForecast.UsZip.City}
[4]	{WeatherForecast.UsZip.City}

- Stop the execution of your program.

Step 4: consume the weather forecast webservice

We will use the <http://www.webservices.net/WeatherForecast.asmx> service which offers the `GetWeatherByZipCode(string zipCode)` method. This method returns an object of the type `WeatherForecasts`. Once we have obtained this object, we can use it to get all the data we need. This time, we can add the service directly in the Silverlight project because the service returns an object and not `System.Xml.Xmlnode`.

- Right click the Silverlight project and choose "Add Service Reference".
- Fill in <http://www.webservices.net/WeatherForecast.asmx> in the address field and click go.
- Fill in "Forecast" as the namespace for the service.



We will test the service the same way we did for the `UsZip` service. Comment out all the code we used to test the `UsZip` service and replace it with the following code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

using System.ServiceModel;
using System.ServiceModel.Channels;
using ServerDiagnostics.CityService;
using ServerDiagnostics.Forecast;

namespace ServerDiagnostics
{
    public partial class Page : UserControl
    {
        public Page()
        {
            InitializeComponent();
            WeatherForecastSoapClient proxy;
```

```

        Binding binding = new BasicHttpBinding();
        EndpointAddress endpoint = new
        EndpointAddress("http://www.websvcicex.net/WeatherForecast.asmx");
        proxy = new WeatherForecastSoapClient(binding, endpoint);
        proxy.GetWeatherByZipCodeCompleted += new
        System.EventHandler<GetWeatherByZipCodeCompletedEventArgs>(proxy_GetWeatherByZipCodeCompleted)
        ;

        proxy.GetWeatherByZipCodeAsync("98052");
    }

    void proxy_GetWeatherByZipCodeCompleted(object sender,
    GetWeatherByZipCodeCompletedEventArgs e)
    {
        }
    }
}

```

Place a breakpoint at the proxy_GetWeatherByZipCodeCompleted method and run the project. When the breakpoint is reached, check the content of e.Result. You should see it is an object of the type WeatherForecasts. This object has a property Details which is a collection of WeatherData objects. This object contains the minimum and maximum temperature for a certain day and a picture that represents the weather.

Name	Value
e.Result	{WeatherForecast.Forecast.WeatherForecasts}
AllocationFactor	0.007734
allocationFactorField	0.007734
Details	{WeatherForecast.Forecast.WeatherData[7]}
[0]	{WeatherForecast.Forecast.WeatherData}
Day	"Monday, March 17, 2008"
dayField	"Monday, March 17, 2008"
MaxTemperatureC	"10"
maxTemperatureCField	"10"
MaxTemperatureF	"50"
maxTemperatureFField	"50"
MinTemperatureC	"3"
minTemperatureCField	"3"
MinTemperatureF	"37"
minTemperatureFField	"37"
PropertyChanged	...

Coming up next...

In the first part of this tutorial, we have consumed two webservices and tested them. In the second part, I shall create the user interface for the application and edit the style + templates of the controls.