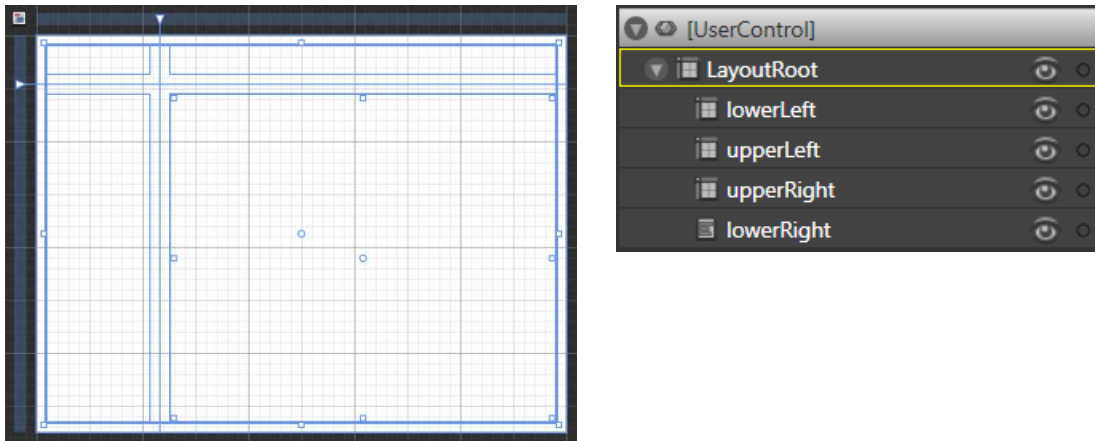


# Weather forecast ( part 2 )

In the first part of this tutorial, I have consumed two webservice and tested them in a Silverlight project. In the second part, I will create a user interface and use some styling that should make the user interface more attractive.

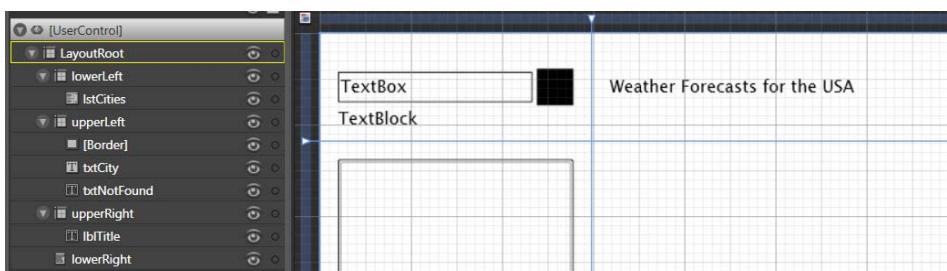
## Step 1: create the user interface

- Open the project with Expression Blend 2.5 March Preview.
- Set the dimensions of page.xaml to 1024 x 768.
- Double click on the LayoutRoot grid to activate it. You see a blue bar around the grid. Hover this bar with your mouse and split the grid into four parts.
- Insert a grid in every part, except in the bottom-right corner where you use a stackpanel. Make sure every container ( grid or stackpanel ) has a 8px margin on every side and that they stretch horizontal and vertical. Also give the containers a name.



Now, we will add some controls in each grid (Make sure you set the margins and the alignment for the controls):

- In the grid upperLeft, we need controls to search for a city:
  - A textbox ( txtCity ) for the name of the city
  - A textblock ( txtNotFound ) that can inform the user is the city isn't found
  - A border with a canvas in it that we will use as a button ( it is also possible to use a button and create a new control template. But sometimes, it is easier to start with a new canvas )
- In the grid upperRight, we put a textblock ( lblTitle ) to show the title for the application.
- In the grid lowerRight, we place a listbox ( lstCities ) that shall contain the result of our search.



## Step 2: design the user interface

In this step, we will add some markup to the controls. We will start with creating some resources. A resource is something (a brush, a font, ...) that you can reuse in your application. You can create these resources either in xaml or with Expression Blend.

*Note: I've used <http://kuler.adobe.com> to find a nice color scheme.*

In Xaml, open the file App.xaml and add this code between the <Application.Resources> elements:

```
<SolidColorBrush x:Key="SandBrush" Color="#FFE6E2AF" />
<SolidColorBrush x:Key="GrayBrush" Color="#FFA7A37E" />
<SolidColorBrush x:Key="WhiteBrush" Color="#FFFECCA" />
<SolidColorBrush x:Key="LightBlue" Color="#FF046380" />
<SolidColorBrush x:Key="DarkBlue" Color="#FF002F2F" />
<FontFamily x:Key="TrebuchetFont" >Trebuchet MS</FontFamily>
```

Or, with Expression Blend: draw a rectangle somewhere and give it a color (#FFE62AF for example). Go to the brushes panel and click the little square near the color. A dialog will appear. Give the resource a name and choose Application as target (you can reuse this resource in your entire project). Repeat these steps for all the resources (works also with fonts) you want. Finally, remove the rectangle.

### 2.1 set a gradient as background

- Select the grid layoutRoot and click the Gradient Brush button in the brushed panel.
- Click on the black handler and the color to #FF046380. Do the same with the white handler and set the color to #FFFECCA.
- You can change the direction and length of the gradient with the Brush transform tool in the toolbox.



### 2.2 change the font and color of the textblocks

- Select lblTitle and click the Brush resource in the brushes panel. Select SandBrush as color.
- Go to the font panel, click the little square near the font combobox and choose Local Resource → TrebuchetFont
- Set the size to 48 and the font to bold
- Do the same for txtNotFound, but set the font size to 10

### 2.3 Adjust the textbox

- Select txtCities in the upperLeft grid
- Set the background to a SandBrush, the foreground and border to DarkBlue. Give the border a thickness of 4px on each side.
- Set the font of the textbox to TrebuchetFont and the size to 14.

### 2.4 Create btnSearch

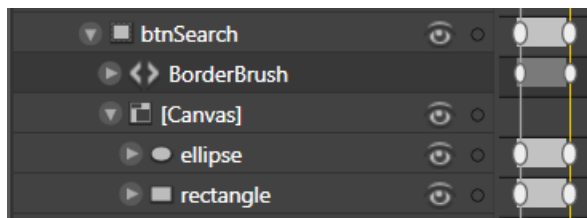
- Select the border, name it btnSearch and add a canvas to the border.
- Set the borderbrush of the border to #FFE6E2AF and give it a borderthickness of 2px on each side. Set the cornerRadius to 4 and set the property cursor to hand.

- Set the background of the border to a color with Alpha value 0. You won't see a background color, but you will be able to click in the middle of the border and trigger the LeftMouseDown event ( comes later... )
- Double click the border to activate it and put a canvas in it. Draw an ellipse with a stroke and no fill in it plus a rectangle. Set the comerradius of the rectangle to 4 and rotate it by dragging the handles around the rectangle. Set the color of the ellipse and rectangle to #FFE6E2AF.



*Note: we don't use the resource SandBrush here, because I want to animate the colors of this button. This doesn't works directly with resource.*

- Click the + button at the top of the Objects and Timelines panel and choose to create a new storyboard. Name the storyboard overSearchButton.
- Click the Record Keyframe button for the border, ellipse and rectangle.
- Move the playhead to 0.5 seconds and change the colors to #FF002F2F ( changing a value automatically adds a new keyframe ).



- Right click the name of your storyboard on top of the objects and timelines panel. Choose duplicate, right click again and rename it to outSearchButton. Finally, right click again and choose reverse. Now, you have created a second storyboard with the reserve animation of the first

We will start this storyboard form code. Open the project in Visual Studio and open the file page.xaml.cs. Remove all the coded we have used in the first part to test the web services ( but keep the using statements ) and add the following code:

```
public partial class Page : UserControl
{
    public Page()
    {
        InitializeComponent();

        // eventlisteners for the MouseEnter and MouseLeave event of the searchbutton
        btnSearch.MouseEnter += new MouseEventHandler(btnSearch_MouseEnter);
        btnSearch.MouseLeave += new MouseEventHandler(btnSearch_MouseLeave);
    }

    void btnSearch_MouseLeave(object sender, MouseEventArgs e)
    {
        // starts the storyboard
        outSearchButton.Begin();
    }

    void btnSearch_MouseEnter(object sender, MouseEventArgs e)
    {
        // starts the storyboard
        overSearchButton.Begin();
    }
}
```

### Step 3: fill up the listbox with cities

Add the following code to the file page.xaml.cs. This code will call the webservice UsZip when btnSearch is clicked and set the itemsSource of the listbox to the result. If there were no cities found, the user will see a warning.

```
public Page()
{
    InitializeComponent();

    // eventlisteners for the MouseEnter and MouseLeave event of the searchbutton
    btnSearch.MouseEnter += new MouseEventHandler(btnSearch_MouseEnter);
    btnSearch.MouseLeave += new MouseEventHandler(btnSearch_MouseLeave);
    // eventlisteners for a click on searchbutton and a change in txtCity
    btnSearch.MouseLeftButtonDown += new
    MouseButtonEventHandler(btnSearch_MouseLeftButtonDown);
    txtCity.TextChanged += new TextChangedEventHandler(txtCity_TextChanged);
}

void txtCity_TextChanged(object sender, TextChangedEventArgs e)
{
    // set the warning to blanco
    txtNotFound.Text = "";
}

void btnSearch_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    // creating a new basicHttpBinding for the cityservice
    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);

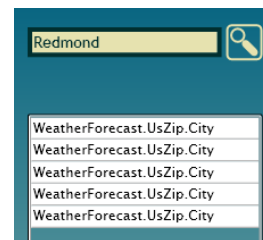
    // get all the cities with a certain name
    client.getCitiesByNameAsync(txtCity.Text);
}

void client_getCitiesByNameCompleted(object sender, getCitiesByNameCompletedEventArgs e)
{
    // check if cities were found
    if (e.Result.Length == 0)
        // no cities found
        txtNotFound.Text = "no cities were found";
    else
        // set the itemssource of the listbox to the result of the webservice
        lstCities.ItemsSource = e.Result;
}
```

If you test your code, this should be the result. Now we will change the itemtemplate of the listbox, so we see the name of the city, the zip code and the state.

Go to page.xaml and find the listbox element lstCities. We will define a new template for all the items in the listbox. In the datatemplate element, we define how each element will look like.

I start with a grid, that contains a rectangle with an opacity brush ( giving the rectangle a "glass" effect ). The grid also contains some textblocks. The text property of each textblock is bound to a property of the City class ( remember, the itemsSource of the listbox is a collection of City objects ) Make the following modifications to the listboxelement:

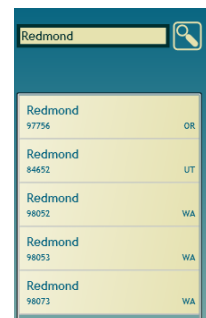


```

<ListBox x:Name="lstCities"...>
<ListBox.ItemTemplate>
  <DataTemplate>
    <Grid Width="220" Height="48" HorizontalAlignment="Stretch"
      VerticalAlignment="Stretch">
      <Rectangle HorizontalAlignment="Stretch" Fill="{StaticResource SandBrush}"
        Margin="0,0,0,0" VerticalAlignment="Stretch">
        <Rectangle.OpacityMask>
          <LinearGradientBrush EndPoint="0.985000014305115,0.90200001001358"
            StartPoint="0.0270000007003546,0.0869999974966049">
            <GradientStop Color="#7F000000"/>
            <GradientStop Color="#FFFFFF" Offset="1"/>
          </LinearGradientBrush>
        </Rectangle.OpacityMask>
      </Rectangle>
      <TextBlock Height="Auto" Margin="8,8,8,0" VerticalAlignment="Top"
        Text="{Binding Name}" TextWrapping="Wrap" FontFamily="{StaticResource TrebuchetFont}"
        TextAlignment="Left" HorizontalAlignment="Stretch" FontSize="14" Foreground="{StaticResource
        LightBlue}"/>
      <TextBlock Height="Auto" HorizontalAlignment="Left" Margin="8,0,8,8"
        VerticalAlignment="Bottom" FontFamily="{StaticResource TrebuchetFont}" Text="{Binding
        ZipCode}" TextAlignment="Left" TextWrapping="Wrap" FontSize="10" Foreground="{StaticResource
        LightBlue}" Width="150"/>
      <TextBlock Height="Auto" HorizontalAlignment="Right" Margin="8,0,48,8"
        VerticalAlignment="Bottom" FontFamily="{StaticResource TrebuchetFont}" Text="{Binding State}"
        TextAlignment="Right" TextWrapping="Wrap" FontSize="10" Foreground="{StaticResource
        LightBlue}" Width="50"/>
    </Grid>
  </DataTemplate>
</ListBox.ItemTemplate>
</ListBox>

```

Because Expression Blend doesn't support editing control templates for Silverlight applications for the moment, the best way to create this xaml is drawing your grid and all the elements somewhere in the scratch area of your workspace. Copy the xaml into the datatemplate and make the bindings by typing the code. You will switch a lot between Blend and Visual Studio for this.



Test your code and the listbox should look like this:

The only thing we have to solve is the border around the listbox and the horizontal scrollbar. Because a Silverlight listbox doesn't have a border property, we have to modify the template for this. Go to the <UserControl.Resources> element and add this style:

```

<Style x:Key="listboxStyle" TargetType="ListBox">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate>
        <Grid>
          <ScrollViewer Cursor="Hand" x:Name="ELEMENT_SV"
            VerticalScrollBarVisibility="Auto"
            HorizontalScrollBarVisibility="Hidden">
            <ItemsPresenter>
            </ItemsPresenter>
          </ScrollViewer>
        </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>

```

This creates a new template for the listbox. Normally a listbox has a border with a scrollviewer and an items presenter. I've replaced the border with a grid, so the big border around the listbox should be gone if you set this style to your listbox element:

```

<ListBox Style="{StaticResource listboxStyle}" ...

```

## Step 4: getting the weather forecast for a certain zip code or city

I add an eventlistener for the SelectionChanged event. In the eventhandler, I first check if a zip code was selected and remove all the elements in the stackpanel lowerRight. I make a request for the weather forecast for the selected zip code.

When the service responds, I check if the result is null. In that case I will make a new request for a weather forecast for the city I've entered. If the result is null again, I show a message that there can no forecasts be found. This is the code:

```
WeatherForecastSoapClient proxy;
void lstCities_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    // set the warning to blanco
    txtNotFound.Text = "";

    // check if an item was selected
    if (lstCities.SelectedIndex > -1)
    {
        // clear every element in the stackpanel
        lowerRight.Children.Clear();

        // get the weather forecast for the selected zip code
        Binding binding = new BasicHttpBinding();
        EndpointAddress endpoint = new
        EndpointAddress("http://www.websvcex.net/WeatherForecast.asmx");
        proxy = new WeatherForecastSoapClient(binding, endpoint);
        proxy.GetWeatherByZipCodeCompleted += new
        System.EventHandler<GetWeatherByZipCodeCompletedEventArgs>(proxy_GetWeatherByZipCodeCompleted)
        ;
        // get the selected zip code
        City selectedCity = lstCities.SelectedItem as City;
        // download the forecast
        proxy.GetWeatherByZipCodeAsync(selectedCity.ZipCode);
    }
}

// is executed when the weather forecast for a certain zip code is downloaded
void proxy_GetWeatherByZipCodeCompleted(object sender, GetWeatherByZipCodeCompletedEventArgs
e)
{
    WeatherForecasts forecast = e.Result;
    // test if a forecast was found for the zip code
    if (forecast.Details != null)
    {
        //add code here later
    }
    else
    {
        // no data found for the zip code: try again with the cityname
        proxy.GetWeatherByPlaceNameCompleted += new
        System.EventHandler<GetWeatherByPlaceNameCompletedEventArgs>(proxy_GetWeatherByPlaceName
        eCompleted);
        proxy.GetWeatherByPlaceNameAsync(txtCity.Text);
    }
}

// is executed when the forecast for a certain placename is downloaded
void proxy_GetWeatherByPlaceNameCompleted(object sender,
GetWeatherByPlaceNameCompletedEventArgs e)
{
    WeatherForecasts forecast = e.Result;
    // test if a forecast was found for the cityname
    if (forecast != null)
    {
        // add code here later
    }
    else
    {
        // no forecasts were found for the city
        txtNotFound.Text = "no forecasts were found";
    }
}
```

If the response is different from null, I will call a method called createWeatherByDay and pass the entire forecast to this method. In the method, I will loop through the forecast and check if the forecast for that day is different from null. In the next step, I will create a new element for those days.

```
void createWeatherByDay(WeatherForecasts forecast)
{
    foreach (WeatherData data in forecast.Details)
    {
        // create a new new WeatherByDay element for each day in the weatherforecast
        if (data.Day != null)
        {
            ...
        }
    }
}
```

## Step 5: create a WeatherByDay user control

Go back to Expression Blend, right click your Silverlight project and choose "Add New Item...". Give your new user control a name: WeatherByDay.xaml. Also make sure that the checkbox "Include code file" is checked.

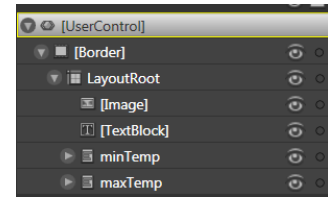
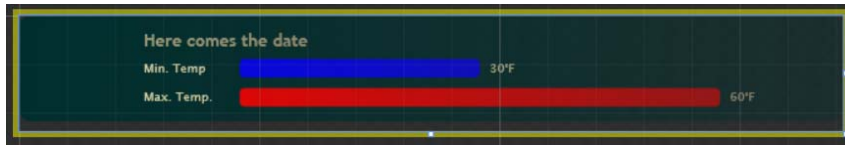
- Set the size of the user control to 686 x95
- Group the LayoutRoot into a border by right clicking the LayoutRoot and choose Group into → Border. Check if the width and height for the grid and border are still set to auto.
- Set the background of the LayoutRoot to nothing.
- Set the bottom margin of the border to 8px, give it a DarkBlue border of 4px on each side.
- Set the cornerRadius of the border to 8 on each side, and set the background to DarkBlue.
- Create an opacity mask for the border:
  - Select the opacity mask property and click the Gradient Brush button
  - Set the alpha value of the white color to 40 and use the Brush Transform from to toolbox to rotate and move the gradient.



We will now add some components to the LayoutRoot:

- An image on the left side of the grid
- A textblock near the image that will show the date of the forecast
- A stackpanel with a textblock, a rectangle ( rectMin ) and a second textblock ( txtMin ) for the minimum temperature
- A stackpanel with a textblock, a rectangle ( rectMax ) and a second textblock ( txtMax ) for the maximum temperature

Your interface could look like this:



Or, as we say in xaml:

```
<Border Height="Auto" Width="Auto" Margin="0,0,0,8" BorderBrush="{StaticResource DarkBlue}"
BorderThickness="4,4,4,4" CornerRadius="8,8,8,8" Background="{StaticResource DarkBlue}">
    <Border.OpacityMask>
        <LinearGradientBrush EndPoint="0.991999983787537,0.933000028133392"
StartPoint="0.021999998807907,0.105999998748302">
            <GradientStop Color="#FF000000"/>
            <GradientStop Color="#66FFFFFF" Offset="1"/>
        </LinearGradientBrush>
    </Border.OpacityMask>

    <Grid x:Name="LayoutRoot" Height="Auto" Width="Auto">
        <Image HorizontalAlignment="Left" Margin="8,8,0,8" Width="Auto"/>
        <TextBlock Height="Auto" HorizontalAlignment="Left" Margin="100,8,0,0"
VerticalAlignment="Top" Width="Auto" TextWrapping="Wrap" FontFamily="{StaticResource
TrebuchetFont}" FontSize="14" Foreground="{StaticResource GrayBrush}" FontWeight="Bold"
Text="Here comes the date"/>
        <StackPanel HorizontalAlignment="Stretch" Margin="100,0,42,32"
VerticalAlignment="Bottom" x:Name="minTemp" Orientation="Horizontal">
            <TextBlock Text="Min. Temp" TextWrapping="Wrap" FontSize="10"
Foreground="{StaticResource SandBrush}" FontWeight="Normal" VerticalAlignment="Center"
HorizontalAlignment="Left" Width="64"/>
            <Rectangle Fill="#FF0000FF" HorizontalAlignment="Left" Margin="16,0,8,0"
VerticalAlignment="Stretch" Height="Auto" Width="200" x:Name="rectMin" RadiusX="4"
RadiusY="4"/>
            <TextBlock Text="30°F" TextWrapping="Wrap" FontSize="10"
Foreground="{StaticResource SandBrush}" FontWeight="Normal" Width="Auto" x:Name="txtMin"/>
        </StackPanel>
        <StackPanel HorizontalAlignment="Stretch" Margin="100,0,42,8"
VerticalAlignment="Bottom" x:Name="maxTemp" Orientation="Horizontal">
            <TextBlock FontSize="10" FontWeight="Normal" Foreground="{StaticResource
SandBrush}" Text="Max. Temp." TextWrapping="Wrap" VerticalAlignment="Center"
HorizontalAlignment="Left" Width="64"/>
            <Rectangle Height="Auto" Width="400" Fill="#FFFFFF0000"
HorizontalAlignment="Left" Margin="16,0,8,0" x:Name="rectMax" RadiusX="4" RadiusY="4"/>
            <TextBlock FontSize="10" FontWeight="Normal" Foreground="{StaticResource
SandBrush}" Text="60°F" TextWrapping="Wrap" Width="Auto" x:Name="txtMax"/>
        </StackPanel>
    </Grid>
</Border>
```

Go to Visual Studio, open the file WeatherByDay.xaml.cs and make these modifications:

```
namespace WeatherForecast
{
    public partial class WeatherByDay : UserControl
    {
        public WeatherByDay(WeatherForecast.Forecast.WeatherData data)
        {
            // Required to initialize variables
            InitializeComponent();

            // set the datacontext of the control to the received data
            this.DataContext = data;
            // add the ° F symbol after the temperatures
            txtMin.Text += " ° F";
            txtMax.Text += " ° F";

            // set the length of the rectangles: one degree = 5 pixels
            rectMin.Width = Convert.ToDouble(data.MinTemperatureF) * 5;
            rectMax.Width = Convert.ToDouble(data.MaxTemperatureF) * 5;
        }
    }
}
```

The constructor requires an argument of the type WeatherData. This type contains all the information we need for one day.

I set the DataContext of the control to the received WeatherData. Now, I can use databinding in xaml to fill in the textblocks.

This code also adds the ° F symbol after the temperatures and sets the length of the rectangles.

Finally, you have to make some modifications in xaml:

- Set the width of both rectangles ( rectMin and rectMax ) to 0
- Bind the text of the textblocks txtMin and txtMax to their value: `{Binding MaxTemperatureF}` or `{Binding MinTemperatureF}`
- Bind the source of the image to the property WeatherImage: `<Image Source="{Binding WeatherImage}" ...`
- Bind the text of the textblock with the date to the property Day: `Text="{Binding Day}"`

Go back to the file page.xaml.cs and navigate to the method createWeatherByDay(). Make sure your method creates a new WeatherByDayElement for each day and adds it to the stackpanel.

```
void createWeatherByDay(WeatherForecasts forecast)
{
    foreach (WeatherData data in forecast.Details)
    {
        // create a new new WeatherByDay element for each day in the weatherforecast
        if (data.Day != null)
        {
            // create a new instance of the WeatherByDay control
            WeatherByDay day = new WeatherByDay(data);
            // add this control to the stackpanel
            lowerRight.Children.Add(day);
        }
    }
}
```

Hit F5 to test your project, and you should be able to see the weather forecast for every city in the USA.

