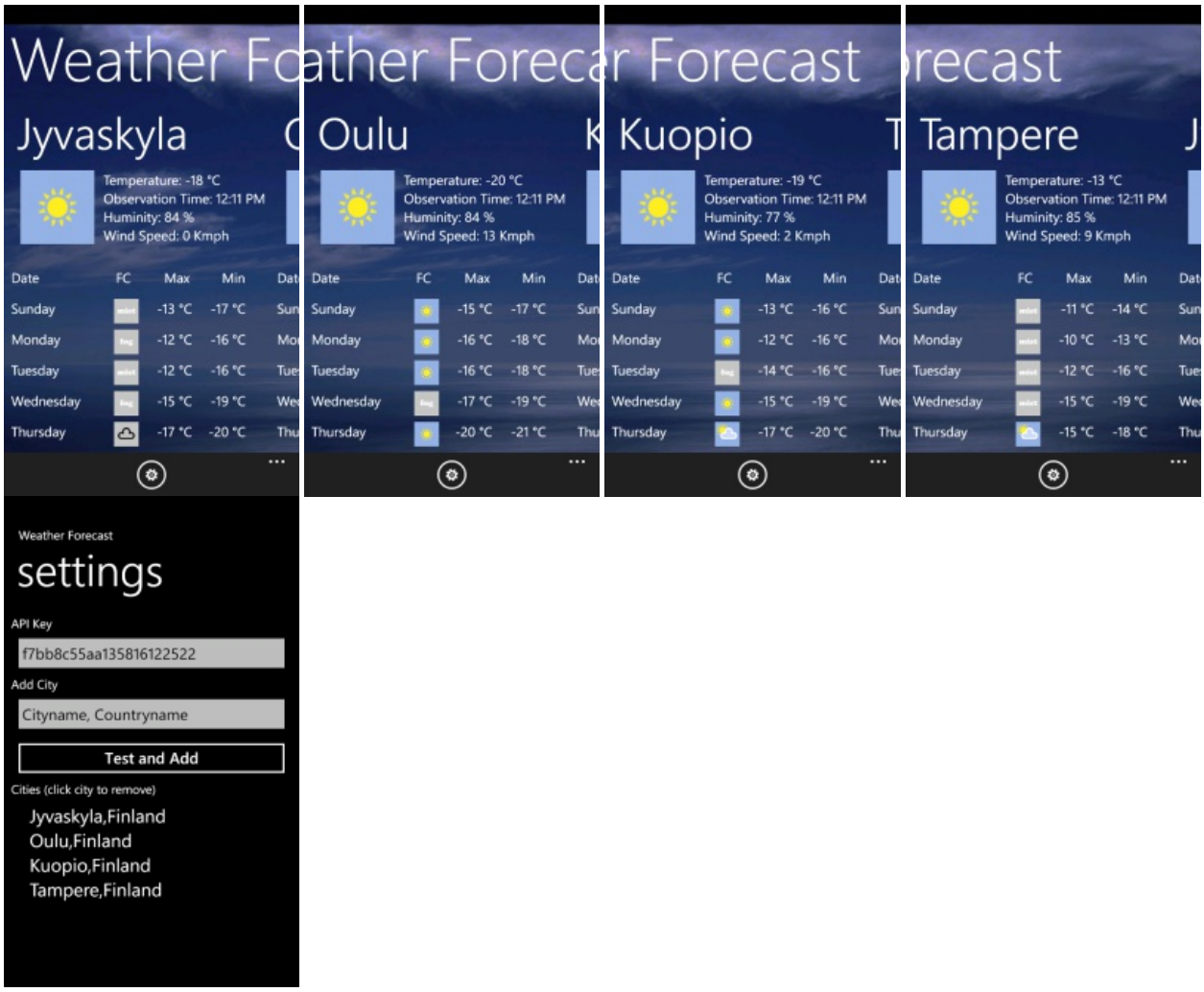
Weather in Windows Phone 7

This article explains how to load XML based data to show weather forecast in your favorite cities around the world.

Introduction

In this article, I will show how to create Windows Phone application which loads Weather Forecast from World Weather Online service. There is already nice Weather Online article here on Nokia Developer Wiki, but I want to make it more deeply to store my favorite cities to phone and use panorama view to show weather forecasts.

Application is designed to run in Panorama Type in WP7. First application loads stored cities and Weather Online ApiKey from device and then it fetches all the weather forecasts. Each forecast is shown in own Panorama View. User can add ApiKey and favourite cities with Settings Page.



Weather Online API Key

You will need your own api key from Weather Online. Read me about getting api key from Weather Online. It is nicely documented there in Basic Idea -section.

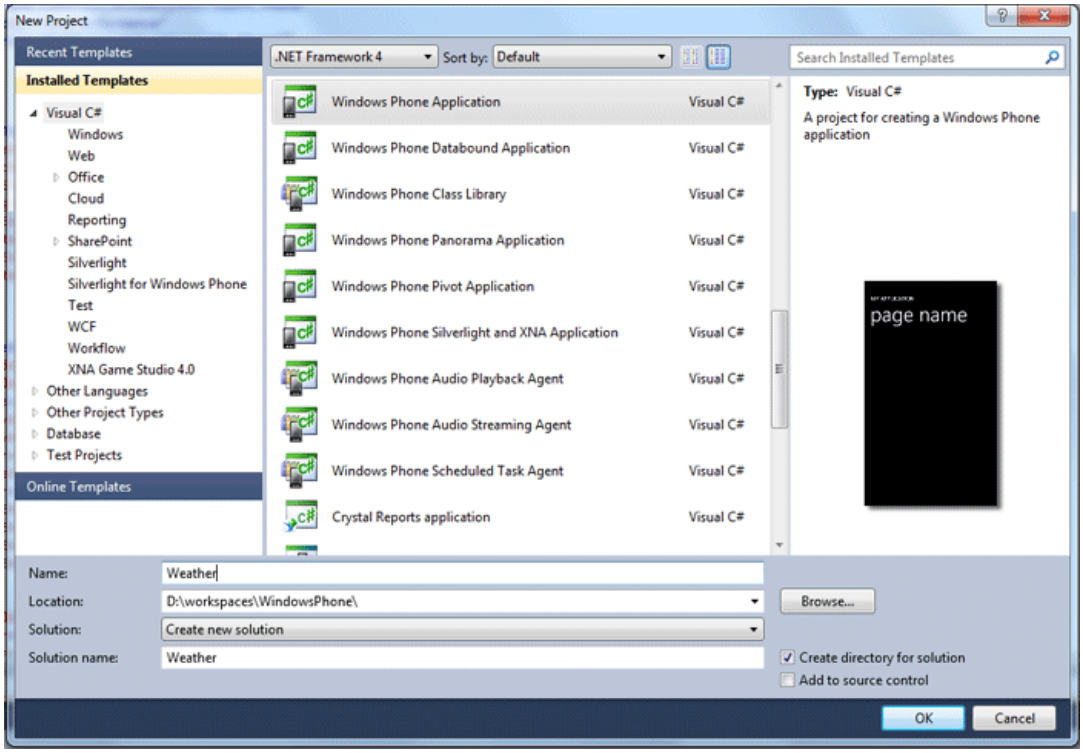
Windows Phone 7.1 SDK

To develop applications for Windows Phone 7 devices, you need to install Windows Phone 7.1 SDK. You can download the latest SDK for Windows Phone here.

Windows Phone Application

To start creating a new Windows Phone Application, start Microsoft Visual studio then create a new Project and select Windows Phone Application Template. There is a Panorama Template also available but select Windows Phone Application Template. In this way we keep our application as clean as possible and we can do later those panoramas with coding.

(C) Copyright Nokia 2014. All rights reserved.



In this example I have used C# as code behind language.

XML Parsing in Windows Phone 7

There are many different ways to load and parse XML data in Windows Phone 7. In this example I will use XML Deserialization to load XML document and pass the data to a deserializer. You will need to add references to `System.Xml.Serialization` and `System.Xml.Linq` to your project. Right click "References" in your project Solutions Explorer and select "Add new Reference..."

Loading and displaying many images from net

There can be a small issues when you try to load lot of images from the web at the same time in Windows Phone. You can find a few assemblys from the net to fix this. One good one is PhonePerformance assembly (you can download compiled PhonePerformance assembly with source codes here). You have to download and unzip it. Copy PhonePerformance.dll to your project and add a new reference to it with Solution Explorer.

You might get warning with to unblock PhonePerformance.dll assembly (because it is downloaded from web). Close your Visual Studio. Open Windows Explorer and browse your PhonePerformance.dll file. Open file properties and Unblock.

Classes behind the Application

To create a New Classes in to your project:

1. Right click your project in Solutions Explorer
2. Select Add and then Class

Forecast.cs

Weather Online offers lots of different weather information. In this example I only use those what are shown in `Forecast` Class (check available data and modify your own needs). This class is used when weather data is loaded from Weather Online service.

```
namespace Weather
{
    public class Forecast
    {
        public string query { get; set; } // cityname, countryname
        public string observation_time { get; set; }
        public string date { get; set; }
        public string temp_C { get; set; }
        public string tempMaxC { get; set; }
```


(C) Copyright Nokia 2014. All rights reserved.

```
        public string tempMinC { get; set; }
        public string weatherIconUrl { get; set; }
        public string windspeedKmph { get; set; }
        public string humidity { get; set; }
    }
}
```

PanoramaItemObject.cs

I will bind all the weather forecast information's to Panorama View. Each view has basic information about current weather and five day forecast. Above `Forecast` Class will be used here in forecasts List.

```
using System.Collections.Generic;

namespace Weather
{
    public class PanoramaItemObject
    {
        public string observation_time { get; set; }
        public string date { get; set; }
        public string temperature { get; set; }
        public string huminity { get; set; }
        public string windspeed { get; set; }
        public string weatherIconUrl { get; set; }
        // five day's forecast
        public List<Forecast> forecasts { get; set; }
    }
}
```

Panorama View

Design (MainPage.xaml)

This application generates as many Panorama Views as there are Cities stored in Settings Page Cities List. First Panorama Title and Background is modified:

```
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <controls:Panorama Title="Weather Forecast" x:Name="Panorama">
            <controls:Panorama.TitleTemplate>
                <DataTemplate>
                    <TextBlock Text="{Binding Content, RelativeSource={RelativeSource
TemplatedParent}}"
                               Foreground="White"
                               FontSize="100"
                               Margin="0,60,0,0"/>
                </DataTemplate>
            </controls:Panorama.TitleTemplate>
            <controls:Panorama.Background>
                <ImageBrush ImageSource="Images/Background3.jpg"/>
            </controls:Panorama.Background>
        </controls:Panorama>
    </Grid>
```

You can add your own Panorama background image to your own project. First create a new folder for images to Solutions Explorer and then copy your image there in Windows. Finally add your image to project by selecting image folder and Add

(C) Copyright Nokia 2014. All rights reserved.

Existing Item... Use a properly sized image in your application. Background images should be between 480 x 800 pixels and 1024 x 800 pixels.

I will use two templates to show weather forecast in this Panorama View. Those templates are stored in `App.xaml` file.

Design (App.xaml)

Each Panorama View's are equal and those are showing weather forecast from specific city. I have used `DataTemplates` to show current weather forecast and five day's forecast. `App.xaml` file has `Application.Resources` element where you can store your DataTemplates.

The below template is used to show five day forecast. Each row is showing Date, Image and maximum and minimum temperature.

```
<DataTemplate x:Key="ForecastsDataTemplate">
            <StackPanel Height="40" Orientation="Horizontal" Margin="0,10,0,0">
                <TextBlock Text="{Binding date}" FontSize="22" TextAlignment="Left"
Width="150"/>
                <TextBlock Text="   " FontSize="20"/>
                <Image delay:LowProfileImageLoader.UriSource="{Binding weatherIconUrl}"
Width="40" Height="40"/>
                <TextBlock Text="   " FontSize="20"/>
                <TextBlock Text="{Binding tempMaxC, StringFormat='\{0\} °C'}"
FontSize="22" TextAlignment="Right" Width="70"/>
                <TextBlock Text="   " FontSize="20"/>
                <TextBlock Text="{Binding tempMinC, StringFormat='\{0\} °C'}"
FontSize="22" TextAlignment="Right" Width="70"/>
            </StackPanel>
        </DataTemplate>
```

The "main" template for the Panorama View is described below. There is 150px space for the weather forecast image in the left and current forecast is shown in the right. At the bottom there are `StackPanel` for the five day's forecast. First there are some "header" `TextBlocks` and then one `ListBox` for the forecasts. You can use other templates with `ItemTemplate` attribute in `ListBox`.

```
<DataTemplate x:Key="ForecastTemplate">
            <Grid x:Name="ContentPanel" Grid.Row="0" Margin="0,-10,0,0">
                <Grid Height="150" VerticalAlignment="Top">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="150"/>
                        <ColumnDefinition Width="*"/>
                    </Grid.ColumnDefinitions>
```


(C) Copyright Nokia 2014. All rights reserved.

```
                    <Image delay:LowProfileImageLoader.UriSource="{Binding
weatherIconUrl}" Width="120" Height="120" Grid.Column="0" VerticalAlignment="Top"/>
                    <StackPanel Grid.Column="1" Height="200" VerticalAlignment="Top">
                        <TextBlock Text="{Binding temperature}" FontSize="22"/>
                        <TextBlock Text="{Binding observation_time}" FontSize="22"/>
                        <TextBlock Text="{Binding huminity}" FontSize="22"/>
                        <TextBlock Text="{Binding windspeed}" FontSize="22"/>
                    </StackPanel>
                </Grid>
                <Grid Height="300" VerticalAlignment="Bottom">
                    <StackPanel Grid.Column="1" VerticalAlignment="Top"
Margin="0,0,0,0">
                        <StackPanel Grid.Row="4" Height="40" Orientation="Horizontal"
Margin="0,0,0,0">
                            <TextBlock Text="Date" FontSize="22" TextAlignment="Left"
Width="170"/>
                            <TextBlock Text="FC" FontSize="22" TextAlignment="Left"
Width="60"/>
                            <TextBlock Text="Max" FontSize="22" TextAlignment="Right"
Width="60"/>
                            <TextBlock Text="Min" FontSize="22" TextAlignment="Right"
Width="90"/>
                        </StackPanel>
                        <ListBox ItemTemplate="{StaticResource ForecastsDataTemplate}"
ItemsSource="{Binding forecasts}"/>
                    </StackPanel>
                </Grid>
            </Grid>
        </DataTemplate>
```

Remember add `namespace` for PhonePerformance.dll to load Images in this example

```
xmlns:delay="clr-namespace:Delay;assembly=PhonePerformance"
```

All the data is bind to UI from `MainPage.xaml.cs` Class.

Programming (MainPage.xaml.cs)

All the XML loading process is happening here in the `MainPage.xml.cs` file. All stored cities are first loaded to one `ObservableCollection<String> queries` and API key are stored to one `string apikey`. `LoadForecast` method will be called as many time as there are cities stored to `IsolatedStorageSettings`.

Needed Class variables

```
private ObservableCollection<String> queries = new ObservableCollection<String>();
private int query;

private string weatherURL = "http://free.worldweatheronline.com/feed/weather.ashx?q=";
private string apiKey;

private IsolatedStorageSettings appSettings;
const string QueriesSettingsKey = "QueriesKey";
const string APISettingsKey = "APIKey";
```

In `MainPage` Constructor we first get instance from application settings and then check if there are network connection available

(C) Copyright Nokia 2014. All rights reserved.

```
// Constructor
public MainPage()
{
            InitializeComponent();

            // get settings for this application
            appSettings = IsolatedStorageSettings.ApplicationSettings;

            // is there network connection available
            if (!System.Net.NetworkInformation.NetworkInterface.GetIsNetworkAvailable())
            {
                MessageBox.Show("There is no network connection available!");
                return;
            }
}
```

Each time when this Main Page is shown, `OnNavigatedTo` method will be called. Here we load all the cities (queries) and api key from Isolated Storage. After that Panorama Views will be removed and new forecasts are loaded and shown.

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
            if (appSettings.Contains(QueriesSettingsKey))
            {
                queries = (ObservableCollection<String>)appSettings[QueriesSettingsKey];
            }

            if (appSettings.Contains(APISettingsKey))
            {
                apiKey = (string)appSettings[APISettingsKey];
            }
            else
            {
                apiKey = "";
            }

            // delete old Panorama Items
            Panorama.Items.Clear();

            // start loading weather forecast
            query = 0;
            if (queries.Count() > 0 && apiKey != "") LoadForecast();
}
```

`LoadForecast` method will use `WebClient` Class to load XML data asynchronously from Weather Online Server.

```
private void LoadForecast()
{
            WebClient downloader = new WebClient();
            Uri uri = new Uri(weatherURL + queries.ElementAt(query) +
"&format=xml&num_of_days=5&key=" + apiKey, UriKind.Absolute);
            downloader.DownloadStringCompleted += new
DownloadStringCompletedEventHandler(ForecastDownloaded);
            downloader.DownloadStringAsync(uri);
}
```


(C) Copyright Nokia 2014. All rights reserved.

`ForecastDownloaded` method will be called when a new forecast is loaded from Weather Online Server. There we first parse current weather `Forecast` Object and then all the five day's `Forecast` Object's from the XML data. Finally a new `PanoramaItem` will be created with `AddPanoramaItem` method.

```
private void ForecastDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
            if (e.Result == null || e.Error != null)
            {
                MessageBox.Show("Cannot load Weather Forecast!");
            }
            else
            {
                XDocument document = XDocument.Parse(e.Result);
                var data1 = from query in document.Descendants("current_condition")
                            select new Forecast
                            {
                                observation_time = (string)
query.Element("observation_time"),
                                temp_C = (string)query.Element("temp_C"),
                                weatherIconUrl = (string)query.Element("weatherIconUrl"),
                                humidity = (string)query.Element("humidity"),
                                windspeedKmph = (string)query.Element("windspeedKmph")
                            };

                Forecast forecast = data1.ToList<Forecast>()[0];

                var data2 = from query in document.Descendants("weather")
                            select new Forecast
                            {
                                date = (string)query.Element("date"),
                                tempMaxC = (string)query.Element("tempMaxC"),
                                tempMinC = (string)query.Element("tempMinC"),
                                weatherIconUrl =
(string)query.Element("weatherIconUrl"),
                            };

                List<Forecast> forecasts = data2.ToList<Forecast>();

                for (int i = 0; i < forecasts.Count(); i++)
                {
                    forecasts[i].date =
DateTime.Parse(forecasts[i].date).ToString("dddd");
                }


                AddPanoramaItem(forecast,forecasts);
            }
}
```

`AddPanoramaItem` method will take current weather forecast object and five day forecast List as a parameter. First `PanoramaItemObject` will be created to bind the weather data to UI templates. Next "real" Panorama Item will be created with city name as Panorama header text and `ForecastTemplate` will be used behind the Panorama Item. After that next forecast will be loaded.

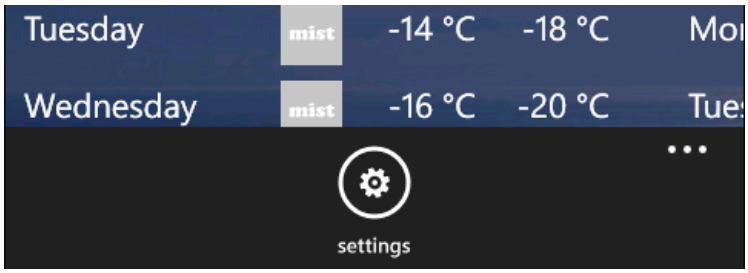
(C) Copyright Nokia 2014. All rights reserved.

```
private void AddPanoramaItem(Forecast forecast, List<Forecast> forecasts)
{
            // create object to bind the data to UI
            PanoramaItemObject pio = new PanoramaItemObject();
            pio.temperature = "Temperature: " + forecast.temp_C + " °C";
            pio.observation_time = "Observ. Time: " + forecast.observation_time;
            pio.windspeed = "Wind Speed: " + forecast.windspeedKmph + " Kmph";
            pio.huminity = "Huminity: " + forecast.humidity + " %";
            pio.weatherIconUrl = forecast.weatherIconUrl;
            pio.forecasts = forecasts;

            // create PanoramaItem
            PanoramaItem panoramaItem = new PanoramaItem();
            panoramaItem.Header = queries[query];
            // modify header to show only city (not the country)
            int index = queries[query].IndexOf(",");
            if (index != -1) panoramaItem.Header = queries[query].Substring(0,
queries[query].IndexOf(","));
            else panoramaItem.Header = queries[query];
            // use ForecastTemplate in Panorama Item
            panoramaItem.ContentTemplate =
(DataTemplate)Application.Current.Resources["ForecastTemplate"];
            panoramaItem.Content = pio;
            // add Panorama Item to Panorama
            Panorama.Items.Add(panoramaItem);
            // query next city forecast
            query++;
            if (query < queries.Count()) LoadForecast();
}
```

Settings

This application settings are stored to Isolated Storage. Settings Page is opened when user clicks Settings icon or text in Application Bar.



Design (Main.xaml)

You can show icon and text in Application Bar. Copy your image to images folder in windows and set it build action to Content in Properties Panel. Add event handling in your icon and text.

```
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton
IconUri="/Images/appbar.feature.settings.rest.png" Text="Settings"
Click="Settings_Click"/>
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="Settings" Click="Settings_Click"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
```


(C) Copyright Nokia 2014. All rights reserved.

```
    </phone:PhoneApplicationPage.ApplicationBar>
```

Programming (Main.xaml.cs)

A Settings Page will be opened when user click settings in Main Page.

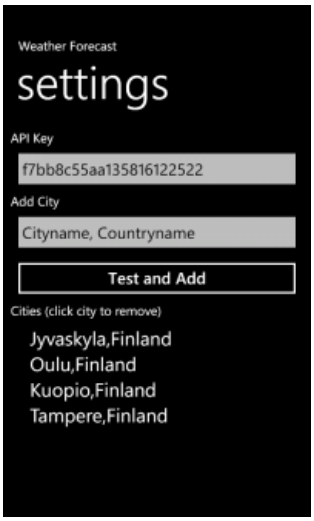
```
private void Settings_Click(object sender, EventArgs e)
{
            this.NavigationService.Navigate(new Uri("/SettingsPage.xaml",
UriKind.Relative));
}
```

Settings Page

To create a new Page to your project, right click your project in Solution Explorer and select Add, New Item... Select Windows Phone Portrait Page and name it to `SettingsPage.xaml`.

Design (SettingsPage.xaml)

Settings Page is normal Portrait Page in this application. Application and Page title are show at the top of the page.



User can add or modify API key and add more cities to `CitiesList`. Added cities can be removed by clicking the city name in the List (confirmation will be asked via `MessageBox`).

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel Orientation="Vertical">
                <TextBlock Text="API Key"/>
                <TextBox x:Name="APIKey" Text=""/>
                <TextBlock Text="Add City"/>
                <TextBox x:Name="NewCityName" Text="Cityname, Countryname"/>
                <Button Content="Test and Add" Click="Test_Click"/>
                <TextBlock Text="Cities (click city to remove)"/>
                <ListBox x:Name="CitiesList" VerticalAlignment="Top" FontSize="30"
ItemsSource="{Binding queries}"
                              Height="280" Margin="30,10,0,0"
SelectionChanged="CitiesList_SelectionChanged"/>
            </StackPanel>
        </Grid>
```

Programming (SettingsPage.xaml.cs)

Here we first have to test is there Weather Forecast available to the city which user is adding to settings. If city works with

(C) Copyright Nokia 2014. All rights reserved.

submitted API key, both are added to Class variables. Settings are stored when user navigates back to Panorama View.

Needed Class variables are the same which are listed in `Main.xaml.cs` Class and settings instance are loaded same way.

Each time when this Settings Page is shown, `OnNavigatedTo` method will be called. Here we load all the cities (queries) and api key from Isolated Storage.

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
            if (appSettings.Contains(QueriesSettingsKey))
            {
                queries = (ObservableCollection<String>)appSettings[QueriesSettingsKey];
            }
            if (appSettings.Contains(APISettingsKey))
            {
                apiKey = (string)appSettings[APISettingsKey];
                APIKey.Text = apiKey;
            }
            // add cites to CitiesList
            CitiesList.ItemsSource = queries;
}
```

`OnNavigatedFrom` will be called when user navigates back to Panorama View. All modifications (cities and apikey) will be saved to Isolated Storage.

```
protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
{
            // add queries to isolated storage
            appSettings.Remove(QueriesSettingsKey);
            appSettings.Add(QueriesSettingsKey, queries);

            // add apikey to isolated storage
            appSettings.Remove(APISettingsKey);
            appSettings.Add(APISettingsKey, apiKey);
}
```

User can test a new city (or api key) here also. A new `WebClient` Object will be created and a new weather forecast will be loaded. If there are no errors, then this city and api key works (a new city is stored to `CitiesList`).

```
private void ForecastDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
            if (e.Result == null || e.Error != null)
            {
                MessageBox.Show("Cannot load Weather Forecast!");
            }
            else
            {
                XDocument document = XDocument.Parse(e.Result);
                XElement xmlRoot = document.Root;

                if (xmlRoot.Descendants("error").Count() > 0)
                {
                    MessageBox.Show("There is no weather forecast available for " +
query + " or your apikey is wrong!");
                    NewCityName.Text = query;
                }
```


(C) Copyright Nokia 2014. All rights reserved.

```
                else
                {
                    queries.Add(query);
                    NewCityName.Text = "Cityname,Countryname";
                }
            }
}
```

User can remove cities from `CitiesList` by clicking city name in the List. Remember **not** to try remove city from the list, but remove it from `ObservableCollection<String> queries`.

```
private void CitiesList_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
            int selectedIndex = (sender as ListBox).SelectedIndex;
            if (selectedIndex == -1) return;
            MessageBoxResult m = MessageBox.Show("Do you want to delete " +
queries[selectedIndex] + " from the list?","Delete City?", MessageBoxButton.OKCancel);
            if (m == MessageBoxResult.OK)
            {
                queries.RemoveAt(selectedIndex);
            }
}
```

Summary

I have written a few XML based articles here in Nokia Developer Wiki - Hope you find these articles useful and it helps you work with XML in WP7.

You can download source code here: File:PTM Weather.zip.

(C) Copyright Nokia 2014. All rights reserved.