How do I Display Weather Information?

Charles M. Rich – September 23, 2010

Table of Contents

Introduction

When you get into the www.RichEveryday.Net website and then go to the page which displays photos, the current Orlando weather conditions are displayed. This information is obtained from the National Weather Service from a web service which they provide. Here is how it works.

This is an example of weather information displayed:

HTML for Weather Display

Here is the HTML for this segment:

```
<table>
  <tr align="center">
    <td>
      <asp:Label ID="WeatherCityLabel" runat="server" Text="" Font-
Bold="true">
      </asp:Label>
      <b> Weather: </b>
    </td>
    <td>
      <asp:Image ID="WeatherIcon" runat="server" Height="55px" Width="58px"
/>
    </td>
    <td>
      <asp:Label ID="WeatherDescriptionLabel" runat="server" Text="" Font-
Bold="true">
      </asp:Label>
    </td>
    <td>
      <asp:Label ID="WeatherTemperatureLabel" runat="server" Text="" Font-
Bold="true">
      </asp:Label>
      <b> degrees.</b>
    </td>
    <td>
```

```
      <b>Humidity </b>
      <asp:Label ID="WeatherHumidityLabel" runat="server" Text="" Font-
Bold="true">
      </asp:Label>
      <b>% </b>
    </td>
  </tr>
  <tr>
    <td colspan="5" align="center">
      As of: 
      <asp:Label ID="AsOfLabel" runat="server" Text=""></asp:Label>
       
      <asp:UpdateProgress ID="UpdateProgress1" runat="server"
           AssociatedUpdatePanelID="MainUpdatePanel" >
        <ProgressTemplate>
          <image src="../Images/progressindicator.gif"></image>
            Processing .....
        </ProgressTemplate>
      </asp:UpdateProgress>
    </td>
  </tr>
```

Code-Behind for the Weather Display

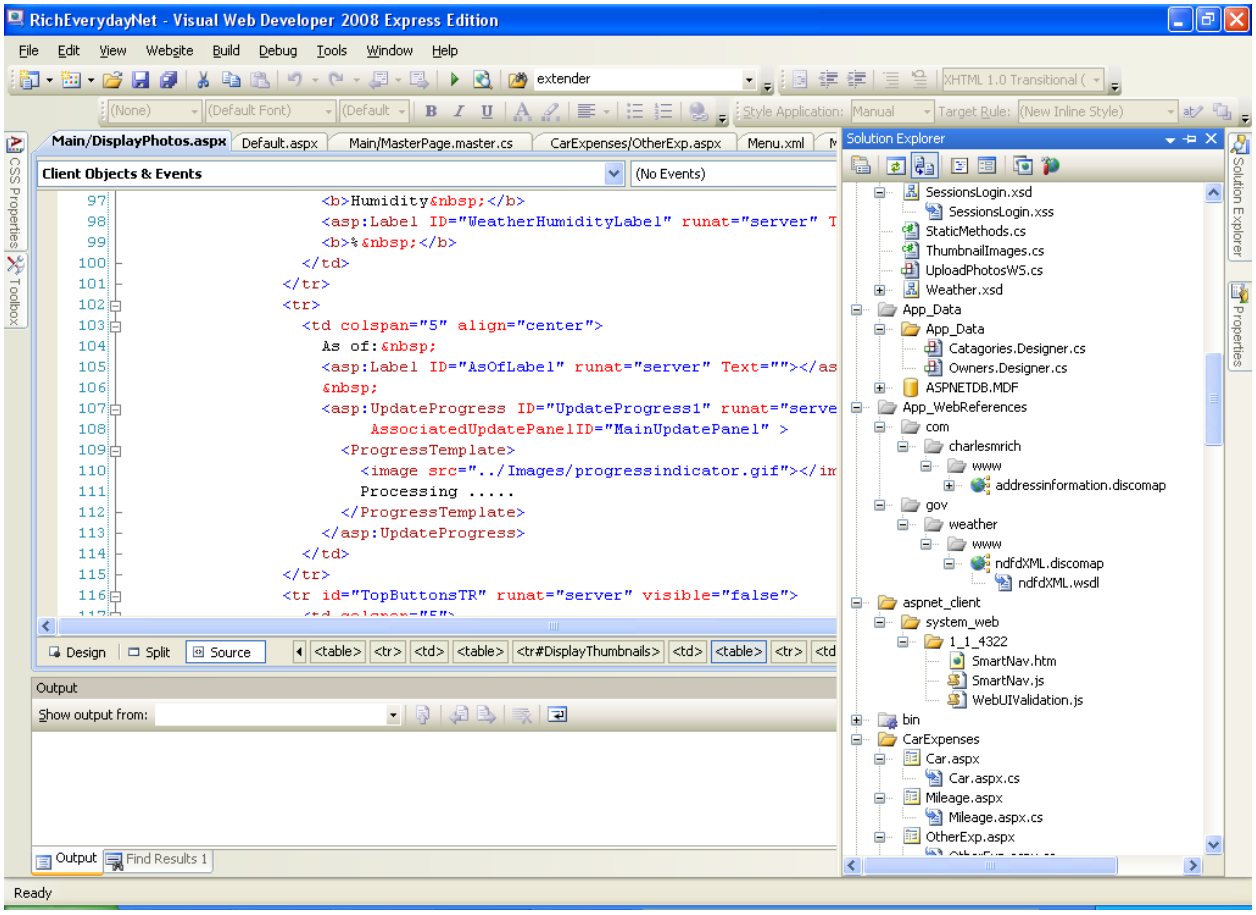
Here is the code behind in DIsplayPhotos.aspx.cs which fills in these items:

```
GetWeatherInfo getWeatherInfo = new GetWeatherInfo();
WeatherIcon.ImageUrl = getWeatherInfo.WeatherIcon;
WeatherCityLabel.Text = getWeatherInfo.City;
WeatherDescriptionLabel.Text = getWeatherInfo.WeatherConditions;
WeatherTemperatureLabel.Text = getWeatherInfo.Temperature;
WeatherHumidityLabel.Text = getWeatherInfo.RelativeHumidity;
AsOfLabel.Text = getWeatherInfo.Updated;
```

As you can see, the GetWeatherInfo class does a lot of work. We will get to that later.

Web Reference

The National Weather Service, web service, is contacted to request this weather information. The web service access is displayed in the Visual Studio Solution Explorer like this:



Notice at the right you see the App_WebReferences folder followed by gov, weather, www and finally the WSDL XML which defines all of the fields I could be using.

The GetWeatherInfo.cs Class

Here is GetWeatherInfo.cs which obtains and formats weather information from the National Weather Service:

```
//
// Programmer:      Charles M. Rich
// Project:         RichEveryday.net
// Description:     This class obtains weather information which is displayed
//                  on the DisplayPhotos.aspx page. It is intended not to
//                  have this class slow
//                  down redering of the DisplayPhotos.aspx page. Since the
//                  response from
//                  the National Weather Service web service could be slow,
//                  the most recent
//                  weather data if first obtained from the database. If the
//                  database
//                  weather data is more than an hour old, then a thread is
//                  started and
//                  the National Weather Service web service is accessed in
//                  the thread to
//                  obtain more current information. When a response is
//                  received from the
//                  web service, then the database is updated. The next time
//                  the user
//                  does anything to cause the DisplayPhotos.aspx page to
//                  refresh, the new weather data will be displayed.
//
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Xml;
using System.Xml.XPath;
using System.IO;
using System.Text;
using System.Threading;
using System.Data;

/// <summary>
/// Get Weather Info for the given Latitude and Longitude
/// </summary>
public class GetWeatherInfo
{
    private string temperature;
    private string relativeHumidity;
    private string weatherConditions;
    private string weatherIcon = string.Empty;
    private string city;
    private DateTime updated;

    private static DateTime lastWeatherQuery = DateTime.MinValue;

    private decimal latitude;           // (decimal)28.51;
```

```
    private decimal longitude;          // (decimal)-81.38;

    /// <summary>
    ///
    /// </summary>
    /// <param name="latitude"></param>
    /// <param name="longitude"></param>
      public GetWeatherInfo()
      {
        StaticMethods.GetLatitudeLongitude(out latitude, out longitude, out
city);
        GetWeatherData();
    }

    /// <summary>
    /// Get weather data from the database and if it is older than an hour,
    /// request new weather data from the National Weather Service.
    /// </summary>
    private void GetWeatherData()
    {
        WeatherTableAdapters.WeatherTableAdapter ta =
            new WeatherTableAdapters.WeatherTableAdapter();
        DataTable dt = ta.GetFromWeather();
        if ((dt != null) && (dt.Rows.Count > 0))
        {
            DataRow dr = dt.Rows[0];
            temperature = Convert.ToString(dr["Temperature"]);
            relativeHumidity = Convert.ToString(dr["Humidity"]);
            weatherConditions = Convert.ToString(dr["Conditions"]);
            weatherIcon = Convert.ToString(dr["Icon"]);
            city = Convert.ToString(dr["City"]);
            updated = (DateTime)dr["Updated"];
        }

        TimeSpan ts = DateTime.Now - updated;
        if (ts.Hours >= 1)
            GetLatestWeather();
    }

    /// <summary>
    /// Get weather from the National Weather Service in a thread
    /// </summary>
    private void GetLatestWeather()
    {
        Thread thread = new Thread(new ThreadStart(FetchWeather));
        thread.Start();
    }

    /// <summary>
    /// Fetch weather info from the National Weather Service web service.
    /// </summary>
    private void FetchWeather()
    {
        string xmlResult = "";

        gov.weather.www.ndfdXML weatherFetcher = new
gov.weather.www.ndfdXML();
```

```
        gov.weather.www.weatherParametersType weatherParameters =
            new gov.weather.www.weatherParametersType();

        DateTime startTime = DateTime.Now.AddHours((double)-1);
        DateTime endTime = DateTime.Now;

        weatherParameters.icons = true;
        weatherParameters.rh = true;
        weatherParameters.temp = true;
        weatherParameters.wx = true;

        try
        {
            xmlResult = weatherFetcher.NDFDgen(latitude, longitude,
gov.weather.www.productType.timeseries,
                startTime, endTime, weatherParameters);
        }
        catch (Exception ex)
        {
            string message = ex.Message;
            string helpLink = ex.HelpLink;
        }

        byte[] byteArray = new byte[xmlResult.Length];
        ASCIIEncoding encoding = new ASCIIEncoding();
        byteArray = encoding.GetBytes(xmlResult);
        MemoryStream memoryStream = new MemoryStream(byteArray, 0,
byteArray.Length);
        XmlReader reader = new XmlTextReader(memoryStream);

        XmlDocument doc = new XmlDocument();
        doc.Load(reader);
        XPathNavigator nav = doc.CreateNavigator();

        XPathNavigator tempNav =
nav.SelectSingleNode("//dwml/data/parameters/temperature/value");
        temperature = tempNav.Value ;

        XPathNavigator relHumNav =
nav.SelectSingleNode("//dwml/data/parameters/humidity/value");
        relativeHumidity = relHumNav.Value;

        XPathNavigator iconNav =
nav.SelectSingleNode("//dwml/data/parameters/conditions-icon/icon-link");
        weatherIcon = iconNav.Value;

        XPathNavigator coverageNav =
            nav.SelectSingleNode("//dwml/data/parameters/weather/weather-
conditions/value");
        if (coverageNav == null)
        {
            weatherConditions = "";
        }
        else
        {
            coverageNav.MoveToFirstAttribute();
            string sCoverage = coverageNav.Value;
```

```
            coverageNav.MoveToNextAttribute();
            string sIntensity = coverageNav.Value;
            coverageNav.MoveToNextAttribute();
            string sWeatherType = coverageNav.Value;

            switch (sCoverage)
            {
                case "chance":
                    sCoverage = "chance of";
                    break;
                default:
                    //sCoverage = "chance of " + sCoverage;
                    break;
            }

            char[] firstChars = sCoverage.ToCharArray();
            weatherConditions = firstChars[0].ToString().ToUpper() +
sCoverage.Substring(1) +
                " " + sWeatherType + ".";

            if (coverageNav.MoveToFirstAttribute())
            {
                while (coverageNav.MoveToNextAttribute())
                {
                    string name = reader.Name;
                    switch (name)
                    {
                        case "coverage":
                            sCoverage = reader.Value;
                            break;
                        case "intensity":
                            sIntensity = reader.Value;
                            break;
                        case "weathertype":
                            sWeatherType = reader.Value;
                            break;
                    }
                }
            }
        }
        lastWeatherQuery = DateTime.Now;

        WeatherTableAdapters.WeatherTableAdapter ta =
            new WeatherTableAdapters.WeatherTableAdapter();
        int iTemperature = Convert.ToInt32(temperature);
        int iHumidity = Convert.ToInt32(relativeHumidity);
        int result = ta.UpdateQuery(iTemperature, iHumidity,
weatherConditions,
            weatherIcon, lastWeatherQuery, 1);
      }

    /// <summary>
    /// Get the Longitude value
    /// </summary>
    public decimal Longitude
    {
        get {return(longitude);}
```

```
    }

    /// <summary>
    /// Get the Latitude value
    /// </summary>
    public decimal Latitude
    {
        get { return (latitude); }
    }

    /// <summary>
    /// Get the current temperature
    /// </summary>
    public string Temperature
    {
        get { return (temperature); }
    }

    /// <summary>
    /// Get the relative humidity
    /// </summary>
    public string RelativeHumidity
    {
        get { return (relativeHumidity); }
    }

    /// <summary>
    /// Get the weather icon
    /// </summary>
    public string WeatherIcon
    {
        get { return (weatherIcon); }
    }

    /// <summary>
    /// Get the weather conditions retrieved from the National Weather
Service
    /// </summary>
    public string WeatherConditions
    {
        get { return (weatherConditions); }
    }

    /// <summary>
    /// Get the city name which we are getting the weather for
    /// </summary>
    public string City
    {
        get { return (city); }
    }

    /// <summary>
    /// When data was last updated
    /// </summary>
    public string Updated
    {
        get
```

```
        {
            string lastUpdate = updated.ToString("dd-MMM-yyyy hh:mm tt");
            return (lastUpdate);
        }
    }
}
```

Conclusion

My wife said "You just get the weather and display in on the screen, don't you". That is basically what you do, but there are several steps into making it all work. If the National Weather Service was using a standard web service where I could just asks for the particular fields that I need, that would have been easier. Instead, they return the information to me in XML format and then I have to extract the fields that I want from that. At least they minimize the amount of data over the internet by letting me request the fields that I want. Here is the code that does this:

```
gov.weather.www.weatherParametersType weatherParameters =
  new gov.weather.www.weatherParametersType();

weatherParameters.icons = true;
weatherParameters.rh = true;
weatherParameters.temp = true;
weatherParameters.wx = true;

xmlResult = weatherFetcher.NDFDgen(latitude, longitude,
gov.weather.www.productType.timeseries,
startTime, endTime, weatherParameters);
```

I tried to write the code as if I expected a lot of hits on my webpage so I don't try to hit the National Weather Service website too often. By saving current data in my database, requests less than an hour old use existing data. If the data is old, current weather information is obtained.