

```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.IO;
using ChartDirector;
using WFET.it.sipeaa.www;

namespace WFET
{
    /// <summary>
    /// Web form application for the web service ET (using the component ET)
    /// The component is described at http://www.isci.it/tools
    /// </summary>
    /// <author>
    /// Marcello Donatelli
    /// m.donatelli@isci.it
    /// </author>>
    public class WebForm1 : System.Web.UI.Page
    {
        .....
        .....
        .....

        private void btnEstimate_Click(object sender, System.EventArgs e)
        {
            lblResult.Text = "0";
            lblUnits.Text = " ";
            DataGrid1.Visible = false;
            WebChartViewer1.Visible = false;
            WebChartViewer2.Visible = false;
            // refresh proper inputs
            HideTextBoxes();
            MakeVisible();
            //read inputs
            double maxAT = double.Parse(txtMaxAirTemperature.Text);

```

```

double minAT = double.Parse(txtMinAirTemperature.Text);
double gSRad = double.Parse(txtGlobalSolarRadiation.Text);
double pSRad = double.Parse(txtPotentialSolarRadiation.Text);
double maxRH = double.Parse(txtMaxRelativeAirHumidity.Text);
double minRH = double.Parse(txtMinRelativeAirHumidity.Text);
double wSpeed = double.Parse(txtWindSpeed.Text);
double wMeasH = double.Parse(txtWindMeasurementHeight.Text);
double cSkyT = double.Parse(txtClearSkyTransmissivity.Text);
double sHar = double.Parse(txtSlopeHargreaves.Text);
double iHar = double.Parse(txtInterceptHargreaves.Text);
//load hourly inputs
LoadDataSet();
WFET.hourlyData hData = new WFET.hourlyData();
hData = hourlyData1;
//the data for the line chart
double[] dataT = new double[24];
double[] dataH = new double[24];
double[] dataW = new double[24];
double[] dataR = new double[24];
double[] dataP = new double[24];
double[] dataX = new double[24];
int i = 0;
for (i=0; i<24; i++)
{
    dataT[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyAirTemperature"];
    dataH[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyRelativeAirHumidity"];
    dataR[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyGlobalSolarRadiation"];
    dataW[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyWindSpeed"];
    dataP[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyPotentialSolarRadiation"];
    dataX[i] = i; // hour, 0-23
}
//create instance of the web service
ISCI_CLIMA_ServiceET et = new ISCI_CLIMA_ServiceET();
//call methods of the service
if (rblOptions.Items.FindByValue("saturationVaporPressure").Selected == true)
{
    lblResult.Text = Math.Round(et.SaturatedVaporPressure(maxAT), 3).ToString();
    lblUnits.Text = "KPa";
}
if (rblOptions.Items.FindByValue("slopeSaturationVaporPressure").Selected == true)
{
    lblResult.Text = Math.Round(et.SlopeSaturatedVaporPressure(maxAT, minAT), 3).ToString();
}

```

```

        lblUnits.Text = "KPa C-1";
    }
    if (rblOptions.Items.FindByValue("actualVaporPressure").Selected == true)
    {
        lblResult.Text = Math.Round(et.ActualVaporPressure(maxAT, minAT, maxRH, minRH), 3).ToString();
        lblUnits.Text = "KPa";
    }
    if (rblOptions.Items.FindByValue("vaporPressureDeficit").Selected == true)
    {
        lblResult.Text = Math.Round(et.VaporPressureDeficit(maxAT, minAT, maxRH, minRH), 3).ToString();
        lblUnits.Text = "KPa";
    }
    if (rblOptions.Items.FindByValue("netRadiation").Selected == true)
    {
        lblResult.Text = Math.Round(et.NetRadiation(maxAT, minAT, pSRad, gSRad, cSkyT), 3).ToString();
        lblUnits.Text = "MJ m-2";
    }
    if (rblOptions.Items.FindByValue("isothermalLongWaveNetRadiation").Selected == true)
    {
        lblResult.Text = Math.Round(et.IsothermalLongWaveNetRadiation(maxAT, minAT, pSRad, gSRad, cSkyT,
        wMeasH), 3).ToString();
        lblUnits.Text = "MJ m-2";
    }
}

if (rblOptions.Items.FindByValue("referenceETHargreaves").Selected == true)
{
    lblResult.Text = Math.Round(et.ReferenceETHargreaves(maxAT, minAT, pSRad, sHar,
    iHar), 3).ToString();
    lblUnits.Text = "mm day-1";
}

if (rblOptions.Items.FindByValue("referenceETPenmanMonteith").Selected == true)
{
    lblResult.Text = Math.Round(et.ReferenceETPenmanMonteith(maxAT, minAT, pSRad, gSRad, cSkyT,
    wSpeed, wMeasH, maxRH, minRH), 3).ToString();
    lblUnits.Text = "mm day-1";
}

if (rblOptions.Items.FindByValue("referenceETPenmanMonteithHourly").Selected == true)
{
    GraphMetData();
    double[] refPMh = et.ReferenceETPenmanMonteithHourly(dataT, dataP, dataR, cSkyT, dataW, wMeasH,
    dataH, pSRad);
}

```

```

        double[] VPD = et.VaporPressureDeficitHourly(dataT, dataP, dataR, cSkyT, dataW, wMeasH, dataH,
        pSRad);
        GraphHourlyEstimates(refPMh, VPD);
        DataGrid1.Visible = true;
        WebChartViewer1.Visible = true;
        WebChartViewer2.Visible = true;
    }
}

private void GraphMetData()
{
    LoadDataSet();
    WFET.hourlyData hData = new WFET.hourlyData();
    hData = hourlyData1;
    //the data for the line chart
    double[] dataT = new double[24];
    double[] dataH = new double[24];
    double[] dataW = new double[24];
    double[] dataR = new double[24];
    double[] dataP = new double[24];
    double[] dataX = new double[24];
    int i = 0;
    for (i=0; i<24; i++)
    {
        dataT[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyAirTemperature"];
        dataH[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyRelativeAirHumidity"];
        dataR[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyGlobalSolarRadiation"];
        dataW[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyWindSpeed"];
        dataP[i] = (double)hData.Tables["hourlyMet"].Rows[i]["hourlyPotentialSolarRadiation"];
        dataX[i] = i; // hour, 0-23
    }
    //Create a XYChart object of size 250 x 250 pixels
    XYChart c = new XYChart(325, 266);

    //Set the plotarea and add grids
    c.setPlotArea(50, 45, 220, 165, -1, -1, 0xc0c0c0, 0xc0c0c0, -1);

    //Add titles to the top and bottom of the chart using 7.5pt Arial font.
    //The text is white 0xffffffff on a deep blue 0x31319C background.
    c.addTitle2(Chart.Top, "HOURLY INPUTS", "arial.ttf", 7.5,
        0xffffffff, 0x31319c);
}

```

```

//Use 7.5 pts Arial as the x axis label font
c.xAxis().setLabelStyle("", 7.5);
c.xAxis().setTitle("hour", "arial.ttf", 10);

//set labels
c.xAxis().setLinearScale(0,24,5);

//Add a legend box at (top of the chart) using horizontal layout and 8
//pts Arial font Set the background and border color to Transparent.
c.addLegend(1, 10, false, "arialnarrow.ttf", 7).setBackground(Chart.Transparent);

//Add layers to the chart
Layer layer1 = c.addLineLayer();
Layer layer2 = c.addLineLayer();
Layer layer3 = c.addLineLayer();
Layer layer4 = c.addLineLayer();
Layer layer5 = c.addLineLayer();
//Add lines. Plot the points with a 7 or 8 pixel square symbol
layer1.addDataSet(dataT, 0xc0c0c0, "C").setDataSymbol(Chart.SquareSymbol, 7);
layer2.addDataSet(dataH, 0xc00000, "%").setDataSymbol(Chart.CircleSymbol, 7);
layer3.addDataSet(dataR, 0x31319c, "MJ m-2").setDataSymbol(Chart.DiamondSymbol, 8);
layer5.addDataSet(dataP, 0xc09090, "MJ m-2").setDataSymbol(Chart.DiamondSymbol, 8);
// use axis2 for two lines
layer3.setUseYAxis2();
layer5.setUseYAxis2();
layer4.addDataSet(dataW, 0x00C000, "m s-1").setDataSymbol(Chart.TriangleSymbol, 8);
layer4.setUseYAxis2();

//Add a title to the secondary (right) y axis
c.yAxis().setTitle("AirTemp & AirRelHum");
c.yAxis2().setTitle("SolarRad & Wind");

//output the chart
WebChartViewer1.Image = c.makeWebImage(1);

//include tool tip for the chart
WebChartViewer1.ImageMap = c.getHTMLImageMap("clickable", "",
    "title='ET hourly input data'");
}

private void GraphHourlyEstimates(double[] ETOut, double[] VPDOut)
{

```

```

//the data for the line chart
double[] dataET = new double[24];
double[] dataVPD = new double[24];
double[] dataX = new double[24];
double cumET = 0;
int i = 0;
for (i=0; i<24; i++)
{
    dataET[i] = ETOut[i];
    dataVPD[i] = VPDOut[i];
    dataX[i] = i; // hour, 0-23
    cumET = cumET + dataET[i];
}
lblResult.Text = Math.Round(cumET,2).ToString();
lblUnits.Text = "mm day-1";

//Create a XYChart object of size 250 x 250 pixels
XYChart c = new XYChart(325, 266);

//Set the plotarea and add grids
c.setPlotArea(50, 45, 220, 165, -1, -1, 0xc0c0c0, 0xc0c0c0, -1);

//Add titles to the top and bottom of the chart using 7.5pt Arial font.
//The text is white 0xffffffff on a deep blue 0x31319c background.
c.addTitle2(Chart.Top, "HOURLY ESTIMATES", "arial.ttf", 7.5,
    0xffffffff, 0x31319c);

//Add text in the plot area (top left corner of plotarea)
string cumulatedET = "daily total ETo = " + Math.Round(cumET,2).ToString();
c.addText(55, 50, cumulatedET, "arial.ttf", 8, 0xc00000).setAlignment(Chart.TopLeft);

//Use 7.5 pts Arial as the x axis label font
c.xAxis().setLabelStyle("", 7.5);
c.xAxis().setTitle("hour", "arial.ttf", 10);

//set labels
c.xAxis().setLinearScale(0,24,5);

//Add a legend box at (top of the chart) using horizontal layout and 8
//pts Arial font Set the background and border color to Transparent.
c.addLegend(50, 10, false, "", 8).setBackground(Chart.Transparent);

```

```

//Add a title to the secondary (right) y axis
c.yAxis().setTitle("ETo (mm h-1)");
c.yAxis2().setTitle("VPD (KPa)");

//Add layers to the chart
Layer layer1 = c.addLineLayer();
Layer layer2 = c.addLineLayer();
//Add lines. Plot the points with a 7 or 8 pixel square symbol
layer1.addDataSet(dataET, 0xc00000, "ET PenmanMonteith").setDataSymbol(Chart.CircleSymbol, 7);
layer2.addDataSet(dataVPD, 0x31319c, "VPD").setDataSymbol(Chart.DiamondSymbol, 8);
layer2.setUseYAxis2();

//output the chart
WebChartViewer2.Image = c.makeWebImage(1);

//include tool tip for the chart
WebChartViewer2.ImageMap = c.getHTMLImageMap("clickable", "",
        "title='ET hourly estimates'");
}

public void LoadDataSet()
{
    try
    {
        OleDbDataAdapter1.Fill(hourlyData1);
        DataGrid1.DataBind();
    }
    catch (System.Exception eFillDataSet)
    {
        throw eFillDataSet;
    }
}

#region Handling text boxes visibility according to method
private void HideTextBoxes()
{
    txtMaxAirTemperature.Visible = false;
    txtMinAirTemperature.Visible = false;
    txtGlobalSolarRadiation.Visible = false;
    txtPotentialSolarRadiation.Visible = false;
    txtMaxRelativeAirHumidity.Visible = false;
}

```

```

        txtMinRelativeAirHumidity.Visible = false;
txtWindSpeed.Visible = false;
        txtWindMeasurementHeight.Visible = false;
        txtClearSkyTransmissivity.Visible = false;
        txtSlopeHargreaves.Visible = false;
        txtInterceptHargreaves.Visible = false;
        DataGrid1.Visible = false;
        WebChartViewer1.Visible = false;
        WebChartViewer2.Visible = false;
    }

private void MakeVisible()
{
    if (rblOptions.Items.FindByValue("slopeSaturationVaporPressure").Selected == true)
    {
        txtMaxAirTemperature.Visible = true;
        txtMinAirTemperature.Visible = true;
    }
    if (rblOptions.Items.FindByValue("saturationVaporPressure").Selected == true)
    {
        txtMaxAirTemperature.Visible = true;
    }
    if (rblOptions.Items.FindByValue("actualVaporPressure").Selected == true)
    {
        txtMaxAirTemperature.Visible = true;
        txtMinAirTemperature.Visible = true;
        txtMaxRelativeAirHumidity.Visible = true;
        txtMinRelativeAirHumidity.Visible = true;
    }
    if (rblOptions.Items.FindByValue("vaporPressureDeficit").Selected == true)
    {
        txtMaxAirTemperature.Visible = true;
        txtMinAirTemperature.Visible = true;
        txtMaxRelativeAirHumidity.Visible = true;
        txtMinRelativeAirHumidity.Visible = true;
    }
    if (rblOptions.Items.FindByValue("isothermalLongWaveNetRadiation").Selected == true)
    {
        txtMaxAirTemperature.Visible = true;
        txtMinAirTemperature.Visible = true;
        txtGlobalSolarRadiation.Visible = true;
    }
}

```

```

        txtPotentialSolarRadiation.Visible = true;
        txtClearSkyTransmissivity.Visible = true;
        txtWindMeasurementHeight.Visible = true;
    }
    if (rblOptions.Items.FindByValue("netRadiation").Selected == true)
    {
        txtMaxAirTemperature.Visible = true;
        txtMinAirTemperature.Visible = true;
        txtGlobalSolarRadiation.Visible = true;
        txtPotentialSolarRadiation.Visible = true;
        txtClearSkyTransmissivity.Visible = true;
    }
    if (rblOptions.Items.FindByValue("referenceETPenmanMonteith").Selected == true)
    {
        txtMaxAirTemperature.Visible = true;
        txtMinAirTemperature.Visible = true;
        txtGlobalSolarRadiation.Visible = true;
        txtPotentialSolarRadiation.Visible = true;
        txtMaxRelativeAirHumidity.Visible = true;
        txtMinRelativeAirHumidity.Visible = true;
        txtWindSpeed.Visible = true;
        txtWindMeasurementHeight.Visible = true;
        txtClearSkyTransmissivity.Visible = true;
    }
    if (rblOptions.Items.FindByValue("referenceETPenmanMonteithHourly").Selected == true)
    {
        txtWindMeasurementHeight.Visible = true;
        txtClearSkyTransmissivity.Visible = true;
        LoadDataSet();
        DataGrid1.Visible = true;
        GraphMetData();
        WebChartViewer1.Visible = true;
    }
    if (rblOptions.Items.FindByValue("referenceETHargreaves").Selected == true)
    {
        txtMaxAirTemperature.Visible = true;
        txtMinAirTemperature.Visible = true;
        txtPotentialSolarRadiation.Visible = true;
        txtSlopeHargreaves.Visible = true;
        txtInterceptHargreaves.Visible = true;
    }
}

```

```

private void btnViewInputsRequired_Click(object sender, System.EventArgs e)
{
    HideTextBoxes();
    MakeVisible();
}
#endregion

private void DataGrid1_EditCommand(object source, System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
    WebChartViewer1.Visible = false;
    WebChartViewer2.Visible = false;
    DataGrid1.EditItemIndex = e.Item.ItemIndex;
    LoadDataSet();
    DataGrid1.DataBind();
}

private void DataGrid1_UpdateCommand(object source, System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
    string T, R, P, H, W;

    // Gets the value of the key field of the row being updated
    string key = DataGrid1.DataKeys[e.Item.ItemIndex].ToString();

    // The first column -- Cells(0) -- contains the Update and Cancel buttons.
    System.Web.UI.WebControls.TextBox tb;

    // Gets the value the TextBox control in the 2nd column
    tb = (System.Web.UI.WebControls.TextBox) (e.Item.Cells[1].Controls[0]);
    T = tb.Text;

    // Gets the value the TextBox control in the 3th column
    tb = (System.Web.UI.WebControls.TextBox) (e.Item.Cells[2].Controls[0]);
    R = tb.Text;

    // Gets the value the TextBox control in the 4th column
    tb = (System.Web.UI.WebControls.TextBox) (e.Item.Cells[3].Controls[0]);
    P = tb.Text;

    // Gets the value the TextBox control in the 5th column
    tb = (System.Web.UI.WebControls.TextBox) (e.Item.Cells[4].Controls[0]);
}

```

```
H = tb.Text;

// Gets the value the TextBox control in the 6th column
tb = (System.Web.UI.WebControls.TextBox) (e.Item.Cells[5].Controls[0]);
W = tb.Text;

LoadDataSet();

// Finds the row in the dataset table that matches the
// one the user updated in the grid.
hourlyData.hourlyMetRow r;
r = hourlyData1.hourlyMet.FindByDayHour(int.Parse(key));

// Updates the dataset table.
r.hourlyAirTemperature = double.Parse(T);
r.hourlyGlobalSolarRadiation = double.Parse(R);
r.hourlyPotentialSolarRadiation = double.Parse(P);
r.hourlyRelativeAirHumidity = double.Parse(H);
r.hourlyWindSpeed = double.Parse(W);

// Calls a SQL statement to update the database from the dataset
oleDbDataAdapter1.Update(hourlyData1);

// Takes the DataGrid row out of editing mode
DataGrid1.EditItemIndex = -1;

// Refreshes the grid
DataGrid1.DataBind();
}
}
}
```