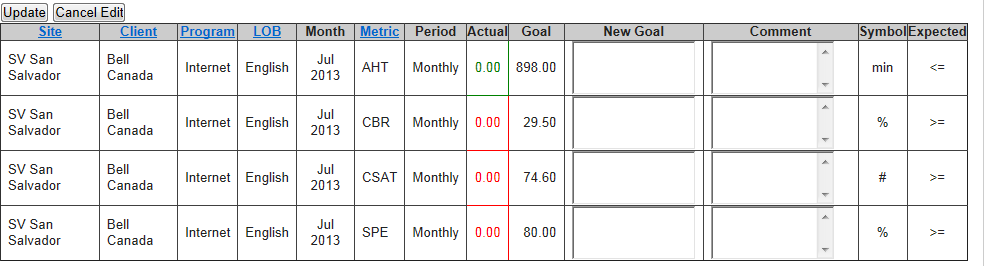
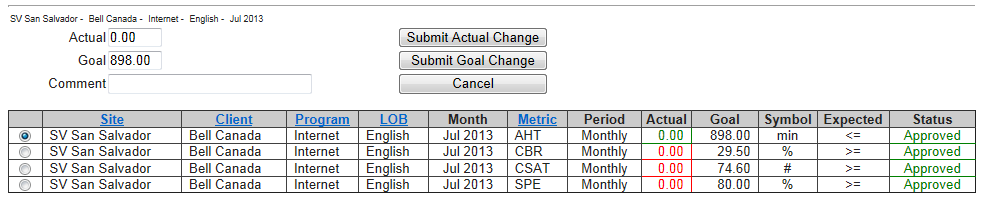


Edit goals in group



Select a row with checkbox to update both, actual value and goal



<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="ucCVPassRate\_View.ascx.cs"

Inherits="DataMaps.ucCVPassRate\_View" %>

<%@ Register Assembly="System.Web" Namespace="System.Web.UI" TagPrefix="asp" %>

<%@ Register Assembly="DataMapsClassLibrary" Namespace="DataMaps" TagPrefix="cc1" %>

<script src="../scripts/scripts.js" type="text/javascript"></script>

<%@ Register Src="~/usercontrols/ucCVPassRate\_Filter.ascx" TagName="MetricFilter"

TagPrefix="uc2" %>

<script type="text/javascript">

function CheckComment(opcion) {

var text = document.getElementById('<%=txComment.ClientID%>').value;

var label="";

if (opcion == 1) {

labels = '<%=lbCheck0.ClientID%>';

}

else{

labels = '<%=lbCheck1.ClientID%>';

}

if (text.trim() == "") {

document.getElementById(labels).innerText = "Provide specific justification for the request";

Blink(labels);

return false;

}

else {

return true;

}

}

function Esconder() {

var text = document.getElementById('<%=plDetails.ClientID%>');

text.innerHTML = "";

return true;

}

String.prototype.trim = function () {

return this.replace(/^\s\*/, "").replace(/\s\*$/, "");

}

function ValidarActuals() {

var valor = document.getElementById('<%=txActual.ClientID%>').value;

var valor1 = document.getElementById('<%=ActualValue.ClientID%>').value;

var labels = '<%=lbCheck0.ClientID%>'

if (!IsNumeric(valor, false)) {

document.getElementById(labels).innerText = " It is not a valid number";

Blink(labels);

return false;

}

else {

if (valor.trim() == valor1.trim()) {

document.getElementById(labels).innerText = " You haven't changed the value";

Blink(labels);

return false;

}

else {

document.getElementById(labels).innerText = "";

}

}

var opcion = document.getElementById('<%=GoApprover.ClientID%>').value;

if (opcion =="False") return CheckComment(1);

}

function ValidarGoals() {

var goals = document.getElementById('<%=txGoal.ClientID%>').value;

var goals1 = document.getElementById('<%=GoalValue.ClientID%>').value;

var labels = '<%=lbCheck1.ClientID%>'

if (!IsNumeric(goals,false)) {

document.getElementById(labels).innerText = " It is not a valid number";

Blink(labels);

return false;

}

else {

if (goals.trim() == goals1.trim()) {

document.getElementById(labels).innerText = " You haven't changed the value";

Blink(labels);

return false;

}

else {

document.getElementById(labels).innerText = "";

}

}

return CheckComment(2);

}

function lostFocus(x) {

var sText = document.getElementById(x).value;

if (sText == "") return;

sText = sText.replace('Not Valid Number ', '');

if (!IsNumeric(sText,false)) {

document.getElementById(x).style.background = 'white';

document.getElementById(x).innerText = 'Not Valid Number ' + sText;

document.getElementById(x).style.color = 'red';

}

else {

document.getElementById(x).style.background = 'white';

document.getElementById(x).style.color = 'green';

}

}

</script>

<uc2:MetricFilter ID="MetricFilter1" runat="server" OnddlRegionSelectedIndexChanged="MetricFilter1\_ddlRegionSelectedIndexChanged"

OnddlSiteSelectedIndexChanged="MetricFilter1\_ddlSiteSelectedIndexChanged" OnddlClientSelectedIndexChanged="MetricFilter1\_ddlClientSelectedIndexChanged"

OnddlProgramSelectedIndexChanged="MetricFilter1\_ddlProgramSelectedIndexChanged"

OnddlLOBSelectedIndexChanged="MetricFilter1\_ddlLOBSelectedIndexChanged" OnddlMesSelectedIndexChanged="MetricFilter1\_ddlMesSelectedIndexChanged"

OnddlMetricSelectedIndexChanged="MetricFilter1\_ddlMetricSelectedIndexChanged" OnExcelClick="MetricFilter1\_ExcelClick"

OnddlStatusSelectedIndexChanged="MetricFilter1\_ddlStatusSelectedIndexChanged" />

<asp:ScriptManagerProxy ID="ScriptManagerProxy1" runat="server">

</asp:ScriptManagerProxy>

<asp:UpdatePanel ID="UpdatePanel1" runat="server">

<ContentTemplate>

<asp:Panel ID="plDetails" runat="server" Width="100%" Visible="False">

<hr />

<table style="width: 100%;">

<tr>

<td>

</td>

</tr>

<tr>

<td colspan="3">

<asp:Label ID="lblSite" runat="server" Font-Size="XX-Small" />

&nbsp;<asp:Label ID="lblClient" runat="server" Font-Size="XX-Small" />

&nbsp;<asp:Label ID="lblProgram" runat="server" Font-Size="XX-Small" />

&nbsp;<asp:Label ID="lblLOB" runat="server" Font-Size="XX-Small" />

&nbsp;<asp:Label ID="lblMetric" runat="server" Font-Size="XX-Small" />

</td>

</tr>

<tr>

<td style="width: 10%;" align="right">

<asp:Label ID="Label1" runat="server" CssClass="label" Font-Size="Small" Text="Actual" />

</td>

<td style="width: 30%;">

<asp:TextBox ID="txActual" runat="server" Font-Size="Small" Width="50px" />

<asp:Label ID="lbCheck0" class="super" runat="server" ForeColor="Red" Font-Size="Small"></asp:Label>

</td>

<td>

<asp:Button ID="btnActual" runat="server" Text="Submit Actual Change" OnClientClick="return ValidarActuals();"

OnClick="btnActual\_Click" Font-Size="Small" Width="150px" />

</td>

</tr>

<tr>

<td align="right">

<asp:Label ID="Label2" runat="server" CssClass="label" Font-Size="Small" Text="Goal" />

</td>

<td>

<asp:TextBox ID="txGoal" runat="server" Font-Size="Small" Width="50px" />

<asp:Label ID="lbCheck1" runat="server" ForeColor="Red" Font-Size="Small" />

</td>

<td>

<asp:Button ID="btnGoal" runat="server" Text="Submit Goal Change" Font-Size="Small"

OnClientClick="return ValidarGoals();" OnClick="btnGoal\_Click" Width="150px" />

</td>

</tr>

<tr>

<td align="right">

<asp:Label ID="lblComment" runat="server" CssClass="label" Font-Size="Small" Text="Comment" />

</td>

<td>

<asp:TextBox ID="txComment" runat="server" Font-Size="Small" Width="200px" />

</td>

<td>

<asp:Button ID="btnCancelChange" runat="server" Font-Size="Small" OnClick="btnCancelChange\_Click"

Text="Cancel" Width="150px" />

<asp:Button ID="btnEsconder" Text="Delete" runat="server" CommandName="Delete" Visible="False"

OnClick="btnEsconder\_Click" />

</td>

</tr>

</table>

</asp:Panel>

<asp:Button ID="Button2" runat="server" OnClick="Button2\_Click" Text="Edit Actual Values"

Visible="False" />

<asp:Button ID="Button1" runat="server" Text="Edit Goal Values" Visible="False" OnClick="Button1\_Click" />

<asp:Button ID="Button3" runat="server" OnClick="Button3\_Click" Text="Cancel Edit"

Visible="False" />

&nbsp;<asp:Label ID="lbWait" runat="server" Font-Size="Small" ForeColor="Red" />

<br />

</ContentTemplate>

</asp:UpdatePanel>

<asp:UpdatePanel ID="UpdatePanel2" runat="server">

<ContentTemplate>

<asp:GridView ID="GVMetrics" runat="server" AllowPaging="True" AutoGenerateColumns="False"

DataKeyNames="MetricDate,MetricTimePeriodID,MetricConfigurationID,Version\_ID,LastUpdt\_DT,LastUpdtBy\_ID,Status"

EmptyDataText="No records found." OnDataBound="GVMetrics\_DataBound" OnPageIndexChanging="GVMetrics\_PageIndexChanging"

OnRowCancelingEdit="GVMetrics\_RowCancelingEdit" OnRowCreated="GridView1\_RowCreated"

AllowSorting="True" OnRowDataBound="GVMetrics\_RowDataBound" OnSelectedIndexChanging="GVMetrics\_SelectedIndexChanging"

Width="100%" PageSize="15" CssClass="GridView" OnSorting="GVMetrics\_Sorting">

<Columns>

<asp:TemplateField>

<ItemTemplate>

<cc1:GridViewRowSelector ID="GridViewRowSelector1" runat="server" />

</ItemTemplate>

<ItemStyle HorizontalAlign="Center"></ItemStyle>

</asp:TemplateField>

<asp:BoundField DataField="Site" HeaderText="Site" SortExpression="Site">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Client" HeaderText="Client" SortExpression="Client">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Program" HeaderText="Program" SortExpression="Program">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="LOB" HeaderText="LOB" SortExpression="LOB">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Month" ItemStyle-Font-Size="Small" HeaderStyle-Font-Size="Small"

HeaderText="Month">

<HeaderStyle Font-Size="Small" />

<ItemStyle HorizontalAlign="Center" Font-Size="Small" />

</asp:BoundField>

<asp:BoundField DataField="Metric" HeaderText="Metric" SortExpression="Metric">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Period" HeaderStyle-Font-Size="Small" HeaderText="Period">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Center" Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Actual" HeaderStyle-Font-Size="Small" HeaderText="Actual" DataFormatString="{0:n2}">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Right" Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:TemplateField HeaderStyle-Font-Size="Small" ItemStyle-HorizontalAlign="Right"

ItemStyle-Font-Size="Small" HeaderText="New Actual Value" Visible="False">

<ItemTemplate>

<asp:TextBox ID="txtNewValue" aligment="Right" onblur="lostFocus(this.id)" runat="server"

Width="140px" />

</ItemTemplate>

<HeaderStyle Font-Size="Small" />

<ItemStyle Font-Size="Small" HorizontalAlign="Center" Width="140px" />

</asp:TemplateField>

<asp:BoundField DataField="Goal" HeaderStyle-Font-Size="Small" HeaderText="Goal" DataFormatString="{0:n2}">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Right" Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:TemplateField HeaderStyle-Font-Size="Small" ItemStyle-HorizontalAlign="Right"

ItemStyle-Font-Size="Small" HeaderText="New Goal" Visible="False">

<ItemTemplate>

<asp:TextBox ID="txtNewGoal" aligment="Right" onblur="lostFocus(this.id)" runat="server"

Width="120px"

height="50px" />

</ItemTemplate>

<HeaderStyle Font-Size="Small" />

<ItemStyle Font-Size="Small" HorizontalAlign="Center" Width="50px"/>

</asp:TemplateField>

<asp:TemplateField HeaderStyle-Font-Size="Small" ItemStyle-HorizontalAlign="Left"

ItemStyle-Font-Size="Small" HeaderText="Comment" Visible="False">

<ItemTemplate>

<asp:TextBox ID="txtCommentGoal" aligment="Left" runat="server" TextMode = "Multiline"

Width="120px"

height="50px" />

</ItemTemplate>

<HeaderStyle Font-Size="Small" />

<ItemStyle Font-Size="Small" HorizontalAlign="Left" Width="140px" />

</asp:TemplateField>

<asp:BoundField DataField="Symbol\_Code" HeaderStyle-Font-Size="Small" HeaderText="Symbol">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Center" Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Expected" HeaderStyle-Font-Size="Small" HeaderText="Expected">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Center" Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Status" HeaderStyle-Font-Size="Small" ItemStyle-Font-Size="Small"

ItemStyle-HorizontalAlign="Center" HeaderText="Status">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Center" Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:TemplateField ItemStyle-HorizontalAlign="Center">

<ItemTemplate>

<asp:ImageButton ID="btnCancel" runat="server" AlternateText="Cancel" CommandName="Cancel"

ImageUrl="~/images/remove\_24.png" ItemStyle-HorizontalAlign="Center" />

</ItemTemplate>

<ItemStyle HorizontalAlign="Center"></ItemStyle>

</asp:TemplateField>

<asp:BoundField DataField="LastUpdt\_DT" HeaderStyle-Font-Size="Small" ItemStyle-Font-Size="Small"

ItemStyle-HorizontalAlign="Center" HeaderText="Last Update" DataFormatString="{0:d}">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Center" Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="LastUpdtBy\_ID" HeaderStyle-Font-Size="Small" ItemStyle-Font-Size="Small"

ItemStyle-HorizontalAlign="Center" HeaderText="User">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Center" Font-Size="Small"></ItemStyle>

</asp:BoundField>

<asp:BoundField DataField="Comment1" HeaderStyle-Font-Size="Small" ItemStyle-Font-Size="Small"

ItemStyle-HorizontalAlign="Center" HeaderText="Comment">

<HeaderStyle Font-Size="Small"></HeaderStyle>

<ItemStyle HorizontalAlign="Center" Font-Size="Small"></ItemStyle>

</asp:BoundField>

</Columns>

<HeaderStyle CssClass="HeaderStyle" BackColor="#CCCCCC" />

<RowStyle CssClass="RowStyle" />

<SelectedRowStyle CssClass="SelectedRowStyle" />

<AlternatingRowStyle CssClass="AlternatingRowStyle" />

</asp:GridView>

<asp:HiddenField ID="OpenPanel" runat="server" Value="False" />

<asp:HiddenField ID="ActualValue" runat="server" Value="False" />

<asp:HiddenField ID="GoalValue" runat="server" Value="False" />

<asp:HiddenField ID="Outset" runat="server" Value="True" />

<asp:HiddenField ID="ChangeActuals1" runat="server" Value="False" />

<asp:HiddenField ID="ChangeOutside1" runat="server" Value="False" />

<asp:HiddenField ID="ChangeGoals1" runat="server" Value="False" />

<asp:HiddenField ID="GoApprover" runat="server" Value="False" />

<br />

<asp:Label ID="lbWait0" runat="server" Font-Size="Small" ForeColor="Red" />

</ContentTemplate>

</asp:UpdatePanel>

using System;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Net.Mail;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using Microsoft.ApplicationBlocks.Data;

namespace DataMaps

{

public partial class ucCVPassRate\_View : System.Web.UI.UserControl

{

static int period\_cell = 7;

static int val\_cell = 8;

static int edit\_cell = 9;

static int goal\_cell = 10;

static int edit2\_cell = 11;

static int edit3\_comment\_cell = 12;

//static int Symbol\_cell = 13;

static int expect\_cell = 14;

static int status\_cell = 15; // and image - flag

static int cancel\_cell = 16;

static int update\_date\_cell = 17;

static int user\_cell = 18;

static int comment\_cell = 19;

//static int UntilMonth = 1;

private static bool SetProduction = true;

private static int ChangeWindowClosingDay = 15;

private bool ShowCancelColumn = false;

private static string connect = AppSettings.sqlConnDWReference;

#region Inicio

// Set the variable to TRUE to read the roles in production

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public bool ChangeActuals { get; set; }

public bool ChangeActualsOutsideWindow { get; set; }

public bool ChangeGoals { get; set; }

// CHECK FROM DATATABASE TYPE OF USER AUTHORIZATION

//=================================================

protected void Check\_Authorization()

{

bool inicio = System.Convert.ToBoolean(Outset.Value);

if (!inicio)

{

ChangeActuals = System.Convert.ToBoolean(ChangeActuals1.Value);

ChangeActualsOutsideWindow = System.Convert.ToBoolean(ChangeOutside1.Value);

ChangeGoals = System.Convert.ToBoolean(ChangeGoals1.Value);

return;

}

ChangeActuals = true;

ChangeActualsOutsideWindow = true;

ChangeGoals = true;

Outset.Value = "False";

if (Roles.Enabled)

{

string[] roles = Roles.GetRolesForUser(HttpContext.Current.User.Identity.Name);

ChangeActuals = roles.Contains("RefMap.CVCapture.ChangeActuals");

ChangeActualsOutsideWindow = roles.Contains("RefMap.CVCapture.ActualsWindow");

ChangeGoals = roles.Contains("RefMap.CVCapture.ChangeGoals");

}

ChangeActuals1.Value = System.Convert.ToString(ChangeActuals);

ChangeOutside1.Value = System.Convert.ToString(ChangeActualsOutsideWindow);

ChangeGoals1.Value = System.Convert.ToString(ChangeGoals);

}

// PAGE LOAD

//==========

protected void Page\_Load(object sender, EventArgs e)

{

MetricFilter1.ProgramCode\_Show = false;

MetricFilter1.CVPassRate\_Client = true;

Check\_Authorization();

if (IsPostBack) return;

string machine = System.Environment.MachineName.ToString();

if (machine == "WWS10-001-STAGE")

{

SetProduction = false;

lbWait.Text = "Staging Test";

}

plDetails.Visible = false;

MetricFilter1.LoadDDLs();

}

#endregion

#region Show Options

protected void ShowColumns(bool option)

{

GVMetrics.Columns[period\_cell].Visible = option; // period

GVMetrics.Columns[status\_cell].Visible = option; // status

GVMetrics.Columns[update\_date\_cell].Visible = !option; // update date

GVMetrics.Columns[user\_cell].Visible = !option; // user

GVMetrics.Columns[comment\_cell].Visible = !option; // comment

// don't show actuals if in future

DateTime dtCurrent = DateTime.Now;

DateTime dtSelected = new DateTime(MetricFilter1.SelectedYear, MetricFilter1.SelectedMonth, 1);

int result = DateTime.Compare(dtCurrent, dtSelected);

if (result <= 0)

GVMetrics.Columns[val\_cell].Visible = false;

else

GVMetrics.Columns[val\_cell].Visible = true;

}

// VIEW OPTIONS IN GRID AND TABLE

//===============================

private void View\_Options()

{

ShowCancelColumn = false;

btnActual.Enabled = false;

txActual.Enabled = false;

btnGoal.Enabled = false;

txGoal.Enabled = false;

Button2.Visible = false;

GVMetrics.Columns[0].Visible = false;

//ALLOW IF USER HAS PREVILEGES

if (ChangeActuals && InWindow())

{

btnActual.Enabled = true;

txActual.Enabled = true;

GVMetrics.Columns[0].Visible = true;

Button2.Visible = true;

}

if (ChangeActualsOutsideWindow && InExtendedWindow())

{

btnActual.Enabled = true;

txActual.Enabled = true;

GVMetrics.Columns[0].Visible = true;

}

if (ChangeGoals && InExtendedWindow())

{

btnGoal.Enabled = true;

txGoal.Enabled = true;

GVMetrics.Columns[0].Visible = true;

}

if (Button2.Visible)

{

GoApprover.Value = "True";

}

else

{

if (ChangeActualsOutsideWindow && InExtendedWindow())

{

GoApprover.Value = "False";

}

else

{

GoApprover.Value = "True";

}

}

}

protected void View\_Options\_Goal()

{

if (!ChangeGoals)

{

Button1.Visible = false;

return;

}

if (InExtendedWindow())

{

Button1.Visible = true;

return;

}

try

{

DateTime dtCurrent = DateTime.Now;

DateTime dtSelected = new DateTime(MetricFilter1.SelectedYear, MetricFilter1.SelectedMonth, 1);

int result = DateTime.Compare(dtCurrent, dtSelected);

if (result <= 0)

{

Button1.Visible = true;

return;

}

}

catch

{

}

}

public static bool IsNumber(string theValue)

{

try

{

Convert.ToInt32(theValue);

return true;

}

catch

{

try

{

Convert.ToDecimal(theValue);

return true;

}

catch

{

return false;

}

}

}

/\*

protected bool CheckToday()

{

int anio = System.DateTime.Today.Year \* 100 + System.DateTime.Today.Month;

int ss = MetricFilter1.ddlMesSelectedIndex;

if (anio == ss)

return true;

else

return false;

}

protected bool CheckPrevious()

{

int mes = System.DateTime.Today.Month - 1;

int anio = System.DateTime.Today.Year;

if (mes == 0)

{

mes = 12;

anio--;

}

anio = anio \* 100 + mes;

int ss = MetricFilter1.ddlMesSelectedIndex;

if (anio == ss && DateTime.Now.Day <= ChangeWindowClosingDay)

return true;

else

return false;

}

\*/

protected void CleanData()

{

lblSite.Text = string.Empty;

lblClient.Text = string.Empty;

lblProgram.Text = string.Empty;

lblLOB.Text = string.Empty;

lblMetric.Text = string.Empty;

txGoal.Text = string.Empty;

txActual.Text = string.Empty;

txComment.Text = string.Empty;

}

private bool InWindow()

{

//update values from current month, or from previous month if in the first 15 days of current month

int selectedMonth = MetricFilter1.ddlMesSelectedIndex;

if (selectedMonth == 0) return false;

DateTime dtCurrent = DateTime.Now;

DateTime dtSelected = new DateTime(MetricFilter1.SelectedYear, MetricFilter1.SelectedMonth, 1);

if (dtSelected.Year == dtCurrent.Year)

{

if (dtSelected.Month == dtCurrent.Month)

{

return true;

}

if (dtSelected.Month == dtCurrent.AddMonths(-1).Month && dtCurrent.Day <= ChangeWindowClosingDay)

{

return true;

}

}

if (dtSelected.Year == dtCurrent.AddYears(-1).Year)

{

if (dtSelected.Month == dtCurrent.AddMonths(-1).Month && dtCurrent.Day <= ChangeWindowClosingDay)

{

return true;

}

}

return false;

}

private bool InExtendedWindow()

{

//update values from current or previous month, no matter which day in current month

int selectedMonth = MetricFilter1.ddlMesSelectedIndex;

if (selectedMonth == 0) return false;

DateTime dtCurrent = DateTime.Now;

DateTime dtSelected = new DateTime(MetricFilter1.SelectedYear, MetricFilter1.SelectedMonth, 1);

if (dtSelected.Year == dtCurrent.Year)

{

if (dtSelected.Month == dtCurrent.Month)

{

return true;

}

if (dtSelected.Month == dtCurrent.AddMonths(-1).Month)

{

return true;

}

}

if (dtSelected.Year == dtCurrent.AddYears(-1).Year)

{

if (dtSelected.Month == dtCurrent.AddMonths(-1).Month)

{

return true;

}

}

return false;

}

protected string GetUser()

{

//string userName = HttpContext.Current.User.Identity.Name;

string userName = "130578";

try

{

userName = System.Web.Security.Membership.GetUser(HttpContext.Current.User.Identity.Name).UserName.ToString();

}

catch

{

userName = "130578";

}

return userName;

}

protected bool IsNumeric(string numero)

{

try

{

Decimal result = Decimal.Parse(numero);

}

catch

{

return false;

}

return true;

}

protected bool Validate\_Values()

{

if (txActual.Enabled)

{

if (!IsNumeric(txActual.Text))

{

lbCheck0.Text = "Type a Valid Actual Value";

txActual.Focus();

return false;

}

}

if (txGoal.Enabled)

{

if (!IsNumeric(txGoal.Text))

{

lbCheck1.Text = "Type a Valid Goal";

txGoal.Focus();

return false;

}

}

return true;

}

private int GVMetricsSelectedIndex

{

get

{

if (string.IsNullOrEmpty(Request.Form["MetricsGroup"]))

return -1;

else

return Convert.ToInt32(Request.Form["MetricsGroup"]);

}

}

#endregion

#region Update Process

public void UpdateMetric(int Option, int RowIndex)

{

string actual;

string goal;

actual = null;

goal = null;

if (Option == 1 || Option == 2)

{

if (txActual.Text.Trim() != "") actual = txActual.Text.Trim(); //Decimal.Parse(txActual.Text);

if (txGoal.Text.Trim() != "") goal = txGoal.Text.Trim(); Decimal.Parse(txGoal.Text);

// option 1 actual values update

// option 2 goal

// option 3 actual values permission

if (Option == 1 && !InWindow())

{

Option = 3;

}

}

string comment = txComment.Text;

string userName = GetUser();

//String strSQL = "sp\_CVPassRate\_Update\_Metrics";

String strSQL = "sp\_CVPassRate\_Update\_Metrics2";

SqlParameter[] parameters = new SqlParameter[10];

parameters[0] = new SqlParameter("@MetricDate", GVMetrics.DataKeys[RowIndex].Values["MetricDate"]);

parameters[1] = new SqlParameter("@MetricTimePeriodID", GVMetrics.DataKeys[RowIndex].Values["MetricTimePeriodID"]);

parameters[2] = new SqlParameter("@MetricConfigurationID", GVMetrics.DataKeys[RowIndex].Values["MetricConfigurationID"]);

parameters[3] = new SqlParameter("@Version\_ID", GVMetrics.DataKeys[RowIndex].Values["Version\_ID"]);

parameters[4] = new SqlParameter("@Option", Option);

parameters[5] = new SqlParameter("@Actual", actual);

parameters[6] = new SqlParameter("@Goal", goal);

parameters[7] = new SqlParameter("@Comment", comment);

parameters[8] = new SqlParameter("@User", userName);

parameters[9] = new SqlParameter("@LastUpdt\_DT", GVMetrics.DataKeys[RowIndex].Values["LastUpdt\_DT"]);

SqlHelper.ExecuteNonQuery(AppSettings.sqlConnDWReference, CommandType.StoredProcedure, strSQL, parameters);

/\*DateTime uno1 = (DateTime)GVMetrics.DataKeys[RowIndex].Values["MetricDate"];

int uno = (int)GVMetrics.DataKeys[RowIndex].Values["MetricTimePeriodID"];

uno = (int)GVMetrics.DataKeys[RowIndex].Values["MetricConfigurationID"];

uno = (int)GVMetrics.DataKeys[RowIndex].Values["Version\_ID"];\*/

Load\_Data();

if (Option > 3 || Option == 1) return;

string place = "There is a request to change goal for \n\n";

if (Option == 3) place = "There is a request to change actual value for \n\n";

int i = RowIndex;

place = place + "Site: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[1].Text.Trim().ToString()) + "\n";

place = place + "Client: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[2].Text.Trim().ToString()) + "\n";

place = place + "Program: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[3].Text.Trim().ToString()) + "\n";

place = place + "LOB: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[4].Text.Trim().ToString()) + "\n";

place = place + "Metric: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[6].Text.Trim().ToString()) + "\n";

place = place + "Period: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[5].Text.Trim().ToString()) + "\n";

SendEmail(place);

}

#endregion

#region Retrieve Data into Grid

// LOAD DATA

//==========

protected void Load\_Data()

{

Button3.Visible = false;

ShowCancelColumn = false;

CleanData();

plDetails.Visible = false;

OpenPanel.Value = "False";

Session["Data"] = "";

Button2.Text = "Edit Actual Values";

Button1.Text = "Edit Goal Values";

lbCheck0.Text = "";

lbCheck1.Text = "";

lbWait.Text = "";

Button2.Visible = false;

Button1.Visible = false;

int SiteID = System.Convert.ToInt32(MetricFilter1.ddlSiteSelectedValue);

int ClientID = System.Convert.ToInt32(MetricFilter1.ddlClientSelectedValue);

if (SiteID == 0 && ClientID == 0)

{

GVMetrics.Visible = false;

MetricFilter1.ExcelButton = false;

return;

}

GVMetrics.Visible = true;

DataSet ds = RetrieveData();

if (MetricFilter1.GetStatus != "Rejections")

{

ShowColumns(true);

View\_Options();

View\_Options\_Goal();

}

else

{

ShowColumns(false);

GVMetrics.Columns[0].Visible = false;

Button1.Visible = false;

Button2.Visible = false;

Button3.Visible = false;

}

if (ds.Tables[0].Rows.Count > 0)

{

MetricFilter1.ExcelButton = true;

}

else

{

MetricFilter1.ExcelButton = false;

}

GVMetrics.Columns[edit\_cell].Visible = false;

GVMetrics.Columns[edit2\_cell].Visible = false;

GVMetrics.Columns[edit3\_comment\_cell].Visible = false;

Session["Data"] = ds;

ShowCancelColumn = false;

GVMetrics.SelectedIndex = -1;

GVMetrics.DataSource = ds;

GVMetrics.DataBind();

GVMetrics.Columns[cancel\_cell].Visible = ShowCancelColumn;

}

protected DataSet RetrieveData()

{

int RegionID = System.Convert.ToInt32(MetricFilter1.ddlRegionSelectedValue);

int SiteID = System.Convert.ToInt32(MetricFilter1.ddlSiteSelectedValue);

int ClientID = System.Convert.ToInt32(MetricFilter1.ddlClientSelectedValue);

int ProgramID = System.Convert.ToInt32(MetricFilter1.ddlProgramSelectedValue);

int LOB = System.Convert.ToInt32(MetricFilter1.ddlLOBSelectedValue);

int Metric = System.Convert.ToInt32(MetricFilter1.ddlMetricSelectedValue);

SqlParameter[] parameters = new SqlParameter[8];

parameters[0] = new SqlParameter("@Date1", MetricFilter1.ddlMesSelectedValue);

parameters[1] = new SqlParameter("@RegionID", RegionID);

parameters[2] = new SqlParameter("@SiteID", SiteID);

parameters[3] = new SqlParameter("@ClientID", ClientID);

parameters[4] = new SqlParameter("@ProgramID", ProgramID);

parameters[5] = new SqlParameter("@LOB", LOB);

parameters[6] = new SqlParameter("@MetricName\_ID", Metric);

parameters[7] = new SqlParameter("@Date3", MetricFilter1.ddlMesSelectedIndex);

DataSet ds;

if (MetricFilter1.GetStatus != "Rejections")

{

ds = SqlHelper.ExecuteDataset(connect, CommandType.StoredProcedure, "sp\_CVPassRate\_GetMetrics", parameters);

}

else

{

ds = SqlHelper.ExecuteDataset(connect, CommandType.StoredProcedure, "sp\_CVPassRate\_GetMetrics\_Rejected", parameters);

}

return ds;

}

#endregion

#region SelectedIndexChanged

protected void MetricFilter1\_ddlSiteSelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void MetricFilter1\_ddlClientSelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void MetricFilter1\_ddlStatusSelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void MetricFilter1\_ddlMetricSelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void MetricFilter1\_ddlProgramSelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void MetricFilter1\_ddlLOBSelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void MetricFilter1\_ddlMesSelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void MetricFilter1\_ddlRegionSelectedIndexChanged(object sender, EventArgs e)

{

Load\_Data();

}

protected void MetricFilter1\_ExcelClick(object sender, EventArgs e)

{

ExportToExcel();

}

protected void ExportToExcel()

{

HttpResponse response = HttpContext.Current.Response;

response.Clear();

response.Charset = "";

// set the response mime type for excel

response.ContentType = "application/vnd.ms-excel";

response.AddHeader("Content-Disposition", "attachment;filename=CVPASS\_Rate.xls");

StringWriter StringWriter = new System.IO.StringWriter();

HtmlTextWriter HtmlTextWriter = new HtmlTextWriter(StringWriter);

DataGrid dg = new DataGrid();

dg.DataSource = Session["Data"];

dg.DataBind();

dg.RenderControl(HtmlTextWriter);

try

{

Response.Write(StringWriter.ToString());

//Response.End();

//HttpContext.Current.ApplicationInstance.CompleteRequest();

}

catch (Exception)

{

}

}

#endregion

#region GridView Behavior

protected void SetValues(GridViewRowEventArgs e, int option)

{

string sites = e.Row.Cells[1].Text;

string clients = e.Row.Cells[2].Text;

string programs = e.Row.Cells[3].Text;

string LOB = e.Row.Cells[4].Text;

string metrics = e.Row.Cells[5].Text;

string values = e.Row.Cells[val\_cell].Text;

string goals = e.Row.Cells[goal\_cell].Text;

string expected = e.Row.Cells[expect\_cell].Text;

try

{

if (expected != "")

{

double diff = System.Convert.ToDouble(values) - System.Convert.ToDouble(goals);

double percent = diff / System.Convert.ToDouble(goals) \* 100;

if (percent < 0 && percent > -5.0)

{

e.Row.Cells[val\_cell].ForeColor = System.Drawing.Color.Goldenrod;

}

else

{

if (expected == "&gt;=")

{

if (diff < 0)

{

//e.Row.Cells[val\_cell].BackColor = System.Drawing.Color.Red;

e.Row.Cells[val\_cell].ForeColor = System.Drawing.Color.Red;

}

else

{

e.Row.Cells[val\_cell].ForeColor = System.Drawing.Color.Green;

}

}

else

{

if (diff > 0)

{

e.Row.Cells[val\_cell].ForeColor = System.Drawing.Color.Red;

}

else

{

e.Row.Cells[val\_cell].ForeColor = System.Drawing.Color.Green;

}

}

}

}

}

catch { }

string flag = e.Row.Cells[status\_cell].Text;

int ss = e.Row.RowIndex;

string rowClick = " LoadDataToUpdate('" + sites + "','" + clients + "','" + programs + "','" + LOB + "','" + metrics + "','" + values + "','" + goals + "','" + ss + "'); ";

rowClick = " alert('" + LOB + "'); ";

flag = flag.Trim();

if (flag == "Pending Goal" || flag == "Pending Actual")

{

// in process

e.Row.Cells[status\_cell].Text = flag;

e.Row.ToolTip = "Authorization in Process";

//rowClick = "alert('Authorization in Process'); return false;";

//e.Row.Cells[flag\_cell].BackColor = System.Drawing.Color.Yellow;

e.Row.Cells[status\_cell].ForeColor = System.Drawing.Color.Goldenrod;

//e.Row.Cells[flag\_cell].Font.Bold = true;

//e.Row.Cells[0].Text = "";

// if it is the current month, you may update actual values, even with a pending goal

if (MetricFilter1.ddlMesSelectedIndex > 0) e.Row.Cells[0].Enabled = false;

}

else

if (flag == "Rejected")

{

//rejected

e.Row.Cells[status\_cell].Text = "Rejected";

e.Row.ToolTip = "Request Rejected";

e.Row.Cells[status\_cell].ForeColor = System.Drawing.Color.Red;

//e.Row.Cells[flag\_cell].ForeColor = System.Drawing.Color.White;

//e.Row.Cells[flag\_cell].Font.Bold = true;

//e.Row.Cells[0].Enabled = false;

// rowclick

}

else

if (flag == "Approved" || flag == "Active")

{

// approved

//e.Row.Cells[flag\_cell].Text = "Approved";

//e.Row.ToolTip = "";

//rowClick = "alert('Request Approved'); return false;";

e.Row.Cells[status\_cell].ForeColor = System.Drawing.Color.Green;

//e.Row.Cells[flag\_cell].ForeColor = System.Drawing.Color.White;

//e.Row.Cells[flag\_cell].Font.Bold = true;

// e.Row.Cells[0].Text = "";

}

else

{

e.Row.Attributes.Add("onmouseover", "this.previous\_color=this.style.backgroundColor; if (this.style.backgroundColor!='00CCCC') { this.previous\_color=this.style.backgroundColor;this.style.backgroundColor='#C1DAD7'} ");

e.Row.Attributes.Add("onmouseout", " if (this.style.backgroundColor!='00CCCC') this.style.backgroundColor=this.previous\_color;");

if (Button2.Visible || GVMetrics.Columns[0].Visible) e.Row.ToolTip = "Click radio button to make updates";

// rowclick

}

//if (option == 1) e.Row.Attributes.Add("onclick", rowClick);

}

protected void GVMetrics\_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)

{

GridViewRow row = GVMetrics.Rows[e.NewSelectedIndex];

lblSite.Text = HttpUtility.HtmlDecode(row.Cells[1].Text) + " - ";

lblClient.Text = HttpUtility.HtmlDecode(row.Cells[2].Text) + " - ";

lblProgram.Text = HttpUtility.HtmlDecode(row.Cells[3].Text) + " - ";

lblLOB.Text = HttpUtility.HtmlDecode(row.Cells[4].Text) + " - ";

lblMetric.Text = HttpUtility.HtmlDecode(row.Cells[5].Text);

txActual.Text = HttpUtility.HtmlDecode(row.Cells[val\_cell].Text.Trim());

txGoal.Text = HttpUtility.HtmlDecode(row.Cells[goal\_cell].Text.Trim());

if (!IsNumeric(txActual.Text)) txActual.Text = "";

if (!IsNumeric(txGoal.Text)) txGoal.Text = "";

ActualValue.Value = txActual.Text;

GoalValue.Value = txGoal.Text;

lbCheck0.Text = "";

lbCheck1.Text = "";

// SHOW PANEL

plDetails.Visible = true;

OpenPanel.Value = "True";

GVMetrics.Columns[edit\_cell].Visible = false;

GVMetrics.Columns[edit2\_cell].Visible = false;

GVMetrics.Columns[edit3\_comment\_cell].Visible = false;

Button2.Visible = false;

Button1.Visible = false;

//\*\* 15

// check if pending goal to set the values

if (MetricFilter1.ddlMesSelectedIndex == 0 && ChangeGoals)

{

btnGoal.Enabled = ChangeGoals;

txGoal.Enabled = ChangeGoals;

if (row.Cells[10].Text == "Pending Goal")

{

btnGoal.Enabled = false;

txGoal.Enabled = false;

}

}

}

protected void GVMetrics\_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)

{

UpdateMetric(4, e.RowIndex);

}

protected void GVMetrics\_PageIndexChanging(object sender, GridViewPageEventArgs e)

{

GVMetrics.PageIndex = e.NewPageIndex;

if (OpenPanel.Value == "False")

{

ShowCancelColumn = false;

GVMetrics.DataSource = Session["Data"];

GVMetrics.DataBind();

GVMetrics.Columns[cancel\_cell].Visible = ShowCancelColumn;

plDetails.Visible = false;

}

else

{

Load\_Data();

}

}

// protected void GVMetrics\_DataBound(object sender, GridViewRowEventArgs e)

protected void GVMetrics\_DataBound(object sender, EventArgs e)

{

//GVClickable(GVMetrics, e.Row);

//GVMetrics.Columns[cancel\_cell].Visible = ShowCancelColumn;

}

protected void GVMetrics\_RowDataBound(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.DataRow)

{

SetValues(e, 1);

bool seguir = ChangeActuals || ChangeActualsOutsideWindow || ChangeGoals;

if (!seguir) return;

ImageButton btnCancel = (ImageButton)e.Row.FindControl("btnCancel");

string userName = GetUser();

string LastUpdtBy\_ID = (string)GVMetrics.DataKeys[e.Row.RowIndex].Values["LastUpdtBy\_ID"];

btnCancel.Visible = false;

if (userName == LastUpdtBy\_ID)

{

string Status = (string)GVMetrics.DataKeys[e.Row.RowIndex].Values["Status"];

if (Status == "Pending Actual" && (ChangeActuals || ChangeActualsOutsideWindow))

{

btnCancel.Visible = true;

}

if (Status == "Pending Goal" && ChangeGoals)

{

btnCancel.Visible = true;

}

}

if (btnCancel.Visible) ShowCancelColumn = true;

}

}

protected void GridView1\_RowCreated(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.DataRow)

{

SetValues(e, 2);

}

//if (e.Row.RowType == DataControlRowType.DataRow)

//{

// // Grab a reference to the Literal control

// Literal output = (Literal)e.Row.FindControl("RadioButtonMarkup");

// // Output the markup except for the "checked" attribute

// output.Text = string.Format("<input type=\"radio\" name=\"MetricsGroup\" id=\"RowSelector{0}\" value=\"{0}\"", e.Row.RowIndex);

// // See if we need to add the "checked" attribute

// if (GVMetricsSelectedIndex == e.Row.RowIndex)

// output.Text += " checked=\"checked\"";

// output.Text += "onclick=\"" + Page.ClientScript.GetPostBackEventReference(GVMetrics, "Select$" + e.Row.RowIndex.ToString(), true) + "\"";

// // Add the closing tag

// output.Text += " />";

//}

}

#endregion

#region Buttons

protected void btnActual\_Click(object sender, EventArgs e)

{

txGoal.Text = GoalValue.Value;

UpdateMetric(1, GVMetrics.SelectedRow.RowIndex);

}

protected void btnGoal\_Click(object sender, EventArgs e)

{

txActual.Text = ActualValue.Value;

UpdateMetric(2, GVMetrics.SelectedRow.RowIndex);

}

protected void btnCancelChange\_Click(object sender, EventArgs e)

{

plDetails.Visible = false;

Button2.Visible = (ChangeActuals && InWindow());

Button1.Visible = Button2.Visible;

}

protected void btnEsconder\_Click(object sender, EventArgs e)

{

GVMetrics.Columns[edit\_cell].Visible = !GVMetrics.Columns[edit\_cell].Visible;

GVMetrics.Columns[edit2\_cell].Visible = !GVMetrics.Columns[edit2\_cell].Visible;

GVMetrics.Columns[edit3\_comment\_cell].Visible = !GVMetrics.Columns[edit3\_comment\_cell].Visible;

//ExportToExcel();

}

#endregion

#region Update Multiple Values

// UPDATE ACTUAL VALUES

protected void Button2\_Click(object sender, EventArgs e)

{

if (Button2.Text == "Edit Actual Values")

{

Button2.Text = "Update";

Button1.Visible = false;

Button3.Visible = true;

GVMetrics.Columns[0].Visible = false;

GVMetrics.Columns[edit\_cell].Visible = true;

GVMetrics.Columns[status\_cell].Visible = false;

}

else

{

if (CheckValues())

{

string emails = "";

for (int RowIndex = 0; RowIndex < GVMetrics.Rows.Count; RowIndex++)

{

emails = UpdateAllValues(RowIndex, 1);

}

Load\_Data();

//email

if (emails != "")

{

emails = "There is a request to change several actual values for \n\n" + emails;

SendEmail(emails);

}

}

}

}

// UPDATE MULTIPLE GOALS

protected void Button1\_Click(object sender, EventArgs e)

{

if (Button1.Text == "Edit Goal Values")

{

Button1.Text = "Update";

Button2.Visible = false;

Button3.Visible = true;

GVMetrics.Columns[0].Visible = false;

GVMetrics.Columns[edit2\_cell].Visible = true;

GVMetrics.Columns[edit3\_comment\_cell].Visible = true;

for (int RowIndex = 0; RowIndex < GVMetrics.Rows.Count; RowIndex++)

{

ImageButton btnCancel = (ImageButton)GVMetrics.Rows[RowIndex].FindControl("btnCancel");

TextBox numero = (TextBox)GVMetrics.Rows[RowIndex].FindControl("txtNewGoal");

TextBox comenta = (TextBox)GVMetrics.Rows[RowIndex].FindControl("txtCommentGoal");

numero.Enabled = !btnCancel.Visible;

comenta.Enabled = !btnCancel.Visible;

if (btnCancel.Visible)

{

numero.BackColor = System.Drawing.Color.LightGray;

comenta.BackColor = System.Drawing.Color.LightGray;

}

else

{

numero.BackColor = System.Drawing.Color.White;

comenta.BackColor = System.Drawing.Color.White;

}

}

GVMetrics.Columns[status\_cell].Visible = false;

GVMetrics.Columns[comment\_cell].Visible = false;

}

else

{

string emails = "";

if (CheckValues())

{

for (int RowIndex = 0; RowIndex < GVMetrics.Rows.Count; RowIndex++)

{

emails = UpdateAllValues(RowIndex, 2);

}

Load\_Data();

//email

if (emails != "")

{

emails = "There is a request to change several goals for \n\n" + emails;

SendEmail(emails);

}

}

}

}

protected bool CheckValues()

{

bool PassCheck = true;

string TxWhich = "txtNewValue";

if (Button1.Visible) TxWhich = "txtNewGoal";

for (int j = 0; j < GVMetrics.Rows.Count; j++)

{

TextBox numero = (TextBox)GVMetrics.Rows[j].FindControl(TxWhich);

string numero2 = numero.Text;

if (numero2 != "")

{

numero2 = numero2.Replace("Invalid number ", "");

if (!IsNumber(numero2))

{

numero.Text = "Invalid number " + numero2;

numero.ForeColor = System.Drawing.Color.Red;

PassCheck = false;

}

else

{

// applies comment for goals only

if (Button1.Visible)

{

TextBox Comentario = (TextBox)GVMetrics.Rows[j].FindControl("txtCommentGoal");

string lugar = Comentario.Text;

if (lugar.Trim() == "")

{

lugar = System.Convert.ToString(j + 1);

lbWait.Text = "Please type comment in row " + lugar;

numero.ForeColor = System.Drawing.Color.Red;

PassCheck = false;

}

else

numero.ForeColor = System.Drawing.Color.Green;

}

}

}

}

return PassCheck;

}

//ACTUALIZE EN FORMA MASIVA

protected string UpdateAllValues(int RowIndex, int opcion)

{

TextBox numero;

string emails = "";

if (opcion == 1)

numero = (TextBox)GVMetrics.Rows[RowIndex].FindControl("txtNewValue");

else

numero = (TextBox)GVMetrics.Rows[RowIndex].FindControl("txtNewGoal");

numero.ForeColor = System.Drawing.Color.Black;

string numero2 = numero.Text;

if (numero2 != "")

{

string userName = GetUser();

//String strSQL = "sp\_CVPassRate\_Update\_Metrics";

String strSQL = "sp\_CVPassRate\_Update\_Metrics2";

string comentario = "";

if (opcion == 2)

{

TextBox Comentario = (TextBox)GVMetrics.Rows[RowIndex].FindControl("txtCommentGoal");

comentario = Comentario.Text;

comentario = comentario.Trim();

}

//string goal = goals2; // System.Convert.ToDouble(goals2);

SqlParameter[] parameters = new SqlParameter[10];

parameters[0] = new SqlParameter("@MetricDate", GVMetrics.DataKeys[RowIndex].Values["MetricDate"]);

parameters[1] = new SqlParameter("@MetricTimePeriodID", GVMetrics.DataKeys[RowIndex].Values["MetricTimePeriodID"]);

parameters[2] = new SqlParameter("@MetricConfigurationID", GVMetrics.DataKeys[RowIndex].Values["MetricConfigurationID"]);

parameters[3] = new SqlParameter("@Version\_ID", GVMetrics.DataKeys[RowIndex].Values["Version\_ID"]);

parameters[4] = new SqlParameter("@Option", opcion);

double valor;

double goals;

string valores;

if (opcion == 1)

{

valor = System.Convert.ToDouble(numero2); // correcto

parameters[5] = new SqlParameter("@Actual", valor);

try

{

valores = GVMetrics.Rows[RowIndex].Cells[goal\_cell].Text;

valor = System.Convert.ToDouble(valores);

parameters[6] = new SqlParameter("@Goal", valor);

}

catch

{

parameters[6] = new SqlParameter("@Goal", null);

}

}

else

{

try

{

valores = GVMetrics.Rows[RowIndex].Cells[val\_cell].Text;

valor = System.Convert.ToDouble(valores);

parameters[5] = new SqlParameter("@Actual", valor);

}

catch

{

parameters[5] = new SqlParameter("@Actual", null);

}

goals = System.Convert.ToDouble(numero2); // correcto

parameters[6] = new SqlParameter("@Goal", goals);

}

parameters[7] = new SqlParameter("@Comment", comentario);

parameters[8] = new SqlParameter("@User", userName);

parameters[9] = new SqlParameter("@LastUpdt\_DT", GVMetrics.DataKeys[RowIndex].Values["LastUpdt\_DT"]);

SqlHelper.ExecuteNonQuery(AppSettings.sqlConnDWReference, CommandType.StoredProcedure, strSQL, parameters);

// NO NECESITA AUTORIZACION PARA ACTUALS DEL MES

if (opcion == 2)

{

int i = RowIndex;

emails = emails + "Site: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[1].Text.Trim().ToString()) + "\n";

emails = emails + "Client: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[2].Text.Trim().ToString()) + "\n";

//place = place + "Program: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[3].Text.Trim().ToString()) + "\n";

//place = place + "LOB: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[4].Text.Trim().ToString()) + "\n";

//place = place + "Metric: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[6].Text.Trim().ToString()) + "\n";

emails = emails + "Period: " + HttpUtility.HtmlDecode(GVMetrics.Rows[i].Cells[5].Text.Trim().ToString()) + "\n\n";

}

}

return emails;

}

protected void Button3\_Click(object sender, EventArgs e)

{

for (int RowIndex = 0; RowIndex < GVMetrics.Rows.Count; RowIndex++)

{

TextBox numero = (TextBox)GVMetrics.Rows[RowIndex].FindControl("txtNewValue");

numero.ForeColor = System.Drawing.Color.Black;

}

Button1.Text = "Edit Goal Values";

Button2.Text = "Edit Actual Values";

View\_Options();

View\_Options\_Goal();

Button3.Visible = false;

GVMetrics.Columns[0].Visible = true;

GVMetrics.Columns[edit\_cell].Visible = false;

GVMetrics.Columns[edit2\_cell].Visible = false;

GVMetrics.Columns[edit3\_comment\_cell].Visible = false;

GVMetrics.Columns[status\_cell].Visible = true;

}

#endregion

#region Sort

private string ConvertSortDirectionToSql(SortDirection sortDirection)

{

string newSortDirection = String.Empty;

switch (sortDirection)

{

case SortDirection.Ascending:

newSortDirection = "ASC";

break;

case SortDirection.Descending:

newSortDirection = "DESC";

break;

}

return newSortDirection;

}

protected void GVMetrics\_Sorting(object sender, GridViewSortEventArgs e)

{

DataSet ds = RetrieveData();

if (ds.Tables[0].Rows.Count > 0)

{

DataView dataView = new DataView(ds.Tables[0]);

dataView.Sort = e.SortExpression + " " + ConvertSortDirectionToSql(e.SortDirection);

Session["Data"] = dataView;

GVMetrics.DataSource = dataView;

GVMetrics.DataBind();

}

//}

}

#endregion

#region Email

protected void SendEmail(string email)

{

string machine = System.Environment.MachineName.ToString();

//if (machine == "WWS10-001-STAGE" || machine == "LAPTOP\_RMARTIN") return;

int steps = 0;

string SendTo = "";

try

{

MailMessage mail = new MailMessage();

SmtpClient SmtpServer = new SmtpClient("mailedc01");

steps = 1;

string userName = GetUser();

string From = "CVMetric\_ChangeRequest\_No\_Reply@stream.com";

steps = 2;

//try

//{

// From = System.Web.Security.Membership.GetUser(userName).Email.ToString();

// steps = 2;

//}

//catch

//{

// From = "CVMetric\_ChangeRequest\_No\_Reply@stream.com";

//}

mail.From = new MailAddress(From);

steps = 3;

SendTo = GetUsers();

if (SendTo == "")

{

lbWait.Text = "Error sending the email. The list for approvers is empty.";

return;

}

steps = 4;

// take the last ','

int largo = SendTo.Length;

SendTo = SendTo.Substring(0, largo - 1);

steps = 5;

mail.To.Add(SendTo);

steps = 6;

//sp\_RefMap\_GetEmployeeName

string Nombre = GetUserName();

steps = 7;

string place = "";

try

{

int pos1 = email.IndexOf("Site");

int pos2 = email.IndexOf("Client");

string sitio = email.Substring(pos1, (pos2 - pos1 - 1));

place = " for " + sitio;

}

catch

{

}

if (SendTo.Length > 40)

mail.Subject = "xStream Data Maps, Request to Approve " + place;

else

mail.Subject = "xStream Data Maps, Request to Approve. ";

if (machine == "WWS10-001-STAGE" || machine == "LAPTOP\_RMARTIN")

{

mail.Subject = "Test Email: " + mail.Subject;

}

mail.Body = email + "\nFrom " + Nombre + "\n\nClick this link to access the tool: http://referencesystem.stream.com/ ";

SmtpServer.Port = 25;

//SmtpServer.Credentials = new System.Net.NetworkCredential("90980", "SAL@Si0920");

//SmtpServer.EnableSsl = true;

steps = 10;

SmtpServer.Send(mail);

}

catch

{

lbWait.Text = "Error sending the email in step " + steps.ToString() + " , report it to Tomas.Dale@Stream.com " + SendTo;

}

}

protected string GetUserName()

{

if (Session["Employee"] != null)

{

return Session["Employee"].ToString();

}

string From = GetUser();

String strSQL = "sp\_RefMap\_GetEmployeeName";

SqlParameter[] parameters = new SqlParameter[1];

parameters[0] = new SqlParameter("@LoginID", SqlDbType.Int);

parameters[0].Value = System.Convert.ToInt32(From);

object names = SqlHelper.ExecuteScalar(connect, CommandType.StoredProcedure, strSQL, parameters);

Session["Employee"] = names.ToString();

return names.ToString();

}

// GET USERS FOR EMAIL

protected string GetUsers()

{

if (Cache["UserList"] != null)

{

return Cache["UserList"].ToString();

}

string listado = "";

try

{

string correo1, user1;

string[] users = System.Web.Security.Roles.GetUsersInRole("RefMap.CVApprove.Change");

for (int i = 0; i < users.Length; i++)

{

user1 = users[i];

correo1 = System.Web.Security.Membership.GetUser(user1).Email.ToString();

listado = listado + correo1 + ",";

}

}

catch

{

listado = "Robert.Martin@Stream.com,";

}

if (listado.Length > 40)

{

Cache.Insert("UserList", listado, null, System.Web.Caching.Cache.NoAbsoluteExpiration, new TimeSpan(0, 15, 0));

}

return listado;

}

#endregion

}

}