

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Web.UI.WebControls;
using CustomersData;
using CustomersBus;

namespace Dashboard
{
    /// <summary>
    /// Author: Doug Streitenberger
    /// Purpose: This web form presents a GridView control of the Customers Table
    /// supporting table maintenance.
    /// A Sortable, Pageable GridView control is used.
    /// Includes functionality to add, update, and delete Customer records.

    /// Note: Page errors are handled by the InsertDefect method of
    /// the CustDefects class within the CustomersData Namespace.
    /// This method inserts an entry to table tblDefects and
    /// includes the page on which the error occurred along with
    /// the error description.
    /// </summary>
    public partial class DashboardDataAdmin : System.Web.UI.Page
    {
        /// <summary>
        /// Set up default sorting parameters for the Customer GridView control
        /// Current sort values (both field and direction) are stored
        /// in attributes of the gvCustomersAdmin GridView control.
        /// </summary>
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                gvCustomersAdmin.Attributes.Add("SortDir", "ASC");
                gvCustomersAdmin.Attributes.Add("SortVal", "CustomerID");
                BindData();
            }
        }

        /// <summary>
        /// A DataTable containing all Customer records is returned
        /// from CustomersEnum method of the CustData class
        /// within the CustomersData Namespace. This DataTable is then Converted to
        /// a DataView for sorting based upon the current sorting
        /// attributes held within the gvCustomersAdmin GridView control.
        /// </summary>
        private void BindData()
        {
            try
            {

```

```

        DataTable dt = CustData.CustomersEnum();
        DataView dv = new DataView(dt);
        dv.Sort = gvCustomersAdmin.Attributes["SortVal"] + " " + gvCustomersAdmin.Attributes["SortDir"];
        gvCustomersAdmin.DataSource = dv;
        gvCustomersAdmin.DataBind();
    }
    catch (Exception ex)
    {
        CustDefects objDefects = new CustDefects();
        objDefects.InsertDefect(Request.ServerVariables["SCRIPT_NAME"], ex.ToString());
        objDefects = null;
    }
}

/// <summary>
/// For each Customer record in the Customer GridView control,
/// add an attribute to the delete button of each record
/// which is used to confirm whether the user wants to delete this
/// customer record.
/// </summary>
protected void gvCustomersAdmin_OnRowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow
        && gvCustomersAdmin.EditIndex == -1)
    {
        ImageButton btnDel = ((ImageButton)e.Row.FindControl("btnDelete"));
        string sConfirm = "return confirm('Are you sure you want to delete this Customer?')";
        btnDel.Attributes.Add("onclick", sConfirm);
    }
}

/// <summary>
/// A method, 'GetSortDirection' of the internal DashboardCommon class is used to
/// set the sort order attributes of the gvCustomersAdmin GridView control based upon
/// the current sort order attributes of the GridView control.
/// BindData is then called to execute the new sorting.
/// </summary>
protected void gvCustomersAdmin_OnSorting(object sender, GridViewSortEventArgs SortArgs)
{
    try
    {
        DashboardCommon objCommon = new DashboardCommon();
        string strSortDirection = objCommon.GetSortDirection(gvCustomersAdmin.Attributes["SortDir"],
            gvCustomersAdmin.Attributes["SortVal"], SortArgs.SortExpression.ToString());
        objCommon = null;
        gvCustomersAdmin.Attributes["SortVal"] = SortArgs.SortExpression.ToString();
        gvCustomersAdmin.Attributes["SortDir"] = strSortDirection;
        BindData();
    }
    catch (Exception ex)

```

```

    {
        CustDefects objDefects = new CustDefects();
        objDefects.InsertDefect(Request.ServerVariables["SCRIPT_NAME"], ex.ToString());
        objDefects = null;
    }
}

/// <summary>
/// Set the Current Paging Index to the requested page.
/// </summary>
protected void gvCustomersAdmin_OnPageIndexChanging(object sender, GridViewPageEventArgs e)
{
    gvCustomersAdmin.PageIndex = e.NewPageIndex;
    BindData();
}

/// <summary>
/// On edit of Customer GridView control,
/// set the GridView edit item index for the record being edited,
/// hide the page footer.
/// </summary>
protected void gvCustomersAdmin_OnRowEditing(object sender, GridViewEditEventArgs e)
{
    gvCustomersAdmin.EditIndex = e.NewEditIndex;
    gvCustomersAdmin.ShowFooter = false;
    BindData();
}

/// <summary>
/// On Cancel Edit, set the GridView edit item index
/// to display normal page view (no edits)
/// show the page footer
/// </summary>
protected void gvCustomersAdmin_OnRowCancelingEdit(object sender, GridViewCancelEventArgs e)
{
    gvCustomersAdmin.EditIndex = -1;
    gvCustomersAdmin.ShowFooter = true;
    BindData();
}

/// <summary>
/// Handles Update of gvCustomersAdmin GridView line items
/// </summary>
protected void gvCustomersAdmin_OnRowUpdating(object sender, GridViewUpdateEventArgs e)
{
    try
    {
        // Get all field values to update this Customer record
        // There is a custom validator within each record line item of the
        // GridView used for alert messages in the case of invalid data entry.
    }
}

```

```

string sCustomerID = ((Label)gvCustomersAdmin.Rows[e.RowIndex].FindControl("lblCurrCustomerID")).Text;
string sCompanyName = ((TextBox)gvCustomersAdmin.Rows[e.RowIndex].FindControl("tbxCompanyName")).Text;
string sContactName = ((TextBox)gvCustomersAdmin.Rows[e.RowIndex].FindControl("tbxContactName")).Text;
string sPhone = ((TextBox)gvCustomersAdmin.Rows[e.RowIndex].FindControl("tbxPhone")).Text;
string sFax = ((TextBox)gvCustomersAdmin.Rows[e.RowIndex].FindControl("tbxFax")).Text;
    CustomValidator UpdateValidator =
        ((CustomValidator)gvCustomersAdmin.Rows[e.RowIndex].FindControl("CustValEditRecord"));

// Ensure the data entered is valid prior to attempting to Update Record
if (ValidCustomerData(sCustomerID, sCompanyName.Length, sContactName.Length,
    sPhone.Length, sFax.Length, UpdateValidator))
{
    // Execute the method to update this customer information
    // Use CustomerUpdate method of the CustData class
    // within the CustomersData Namespace.
    // The method uses both oldCustomer object (before update)
    // and customer object (after update) objects
    // to control data concurrency and update
    string sOldCompanyName =
        ((Label)gvCustomersAdmin.Rows[e.RowIndex].FindControl("lblCurrCompanyName")).Text;
    string sOldContractName =
        ((Label)gvCustomersAdmin.Rows[e.RowIndex].FindControl("lblCurrContractName")).Text;
    string sOldPhone = ((Label)gvCustomersAdmin.Rows[e.RowIndex].FindControl("lblCurrPhone")).Text;
    string sOldFax = ((Label)gvCustomersAdmin.Rows[e.RowIndex].FindControl("lblCurrFax")).Text;
    Customer oldCustomer =
        new Customer(sCustomerID, sOldCompanyName, sOldContractName, sOldPhone, sOldFax);

    Customer customer = new Customer(sCustomerID, sCompanyName, sContactName, sPhone, sFax);
    int iResult = CustData.CustomerUpdate(oldCustomer, customer);
    customer = null;

    // If no record was updated, show alert for possible data concurrency error.
    if (iResult == 99)
    {
        string sMessage = "Possible Data Concurrency Error. Another user has updated this record. " +
            "Please try again.";
        string script = "<script language=javascript>alert('" + sMessage + "');</script>";
        ClientScript.RegisterClientScriptBlock(this.GetType(), "AlertMessage", script);
    }

    // Set the GridView edit item index to display normal page view (no edits)
    // Show GridView footer
    gvCustomersAdmin.EditIndex = -1;
    gvCustomersAdmin.ShowFooter = true;
    BindData();
}
}
catch (Exception ex)
{
    CustDefects objDefects = new CustDefects();

```

```

        objDefects.InsertDefect(Request.ServerVariables["SCRIPT_NAME"], ex.ToString());
        objDefects = null;
    }
}

/// <summary>
/// Handles Delete and Add of Customers GridView line items
/// </summary>
protected void gvCustomersAdmin_OnRowCommand(object sender, GridViewCommandEventArgs e)
{
    try
    {
        if (e.CommandName.ToUpper() == "CUSTOMERDELETE" && gvCustomersAdmin.EditIndex == -1)
        {
            // Get the CustomerID so we know which record to delete
            // Execute the method to delete this customer record
            // Use CustomerDelete method of the CustData class
            // within the CustomersData Namespace.
            string sCustomerID = e.CommandArgument.ToString(); ;

            // Do Delete Stuff
            CustData.CustomerDelete(sCustomerID);

            // Refresh the GridView.
            BindData();
        }

        if (e.CommandName.ToUpper() == "CUSTOMERADD")
        {
            // Get all field values to add this Customer record
            // There is a custom validator within the footer of the
            // GridView control used for alert messages in the case of invalid data entry.
            string sCustomerID = ((TextBox) gvCustomersAdmin.FooterRow.FindControl("tbxNewCustomerID")).Text;
            string sCompanyName = ((TextBox) gvCustomersAdmin.FooterRow.FindControl("tbxNewCompanyName")).Text;
            string sContactName = ((TextBox) gvCustomersAdmin.FooterRow.FindControl("tbxNewContactName")).Text;
            string sPhone = ((TextBox) gvCustomersAdmin.FooterRow.FindControl("tbxNewPhone")).Text;
            string sFax = ((TextBox) gvCustomersAdmin.FooterRow.FindControl("tbxNewFax")).Text;
            CustomValidator AddValidator =
                ((CustomValidator) gvCustomersAdmin.FooterRow.FindControl("CustValAddRecord"));

            // Ensure the data entered is valid prior to attempting to Insert Record
            if (ValidCustomerData(sCustomerID, sCompanyName.Length, sContactName.Length,
                sPhone.Length, sFax.Length, AddValidator))
            {
                // Instantiate a Customer object to hold this customer record information
                // Execute the method to insert this customer record
                // Use CustomerAdd method of the CustData class
                // within the CustomersData Namespace.
                Customer customer = new Customer(sCustomerID, sCompanyName, sContactName, sPhone, sFax);
                CustData.CustomerAdd(customer);
            }
        }
    }
}

```

```

        customer = null;

        // Refresh the GridView.
        BindData();
    }
}
}
catch (Exception ex)
{
    CustDefects objDefects = new CustDefects();
    objDefects.InsertDefect(Request.ServerVariables["SCRIPT_NAME"], ex.ToString());
    objDefects = null;
}
}

/// <summary>
/// Checks for valid data entry for both insert and update of customer record.
/// </summary>
/// <param name="sCustomerID">Customer ID for record insert</param>
/// <param name="iCompanyNameLen">Length of Company Name As Entered</param>
/// <param name="iContactNameLen">Length of Contact Name As Entered</param>
/// <param name="iPhoneLen">Len of Phone As Entered</param>
/// <param name="iFaxLen">Len of Fax As Entered</param>
/// <param name="thisValidator">Custom Validator to return validation error messages</param>
/// <returns>Boolean bValid - whether record entry is valid</returns>
private bool ValidCustomerData(string sCustomerID, int iCompanyNameLen, int iContactNameLen,
    int iPhoneLen, int iFaxLen, CustomValidator thisValidator)
{
    bool bValid = true;

    // Check CustomerID if this is an added record.
    // CustomerID cannot be null value or exceed 5 characters in length.
    if (thisValidator.ID.ToString().ToUpper() == "CUSTVALADDRECORD")
    {
        // If CustomerID is not entered (null)
        int iCustomerIDLen = sCustomerID.Length;
        if (iCustomerIDLen == 0)
        {
            // Set the custom add validator to false
            // and assign to the custom validator the message to display.
            // return false;
            bValid = false;
            thisValidator.IsValid = bValid;
            thisValidator.Text = "Enter a Customer ID";
            return bValid;
        }
        // On Adding a Customer record, connect to the database
        // and determine if this CustomerID already exists. Must be unique value.
        // Uses UniqueCustomerID method of the CustData class
        // within the CustomersData Namespace.
    }
}

```

```
bool bCustIDExists = CustData.UniqueCustomerID(sCustomerID);
if (bCustIDExists)
{
    bValid = false;
    thisValidator.IsValid = bValid;
    thisValidator.Text = "There is already a Customer ID record with this Customer ID";
    return bValid;
}
}
// Company Name Required Field (not null)
// If there is no entry in the Company Name field,
// Set the custom add/update validator to false
// and assign to the custom validator the message to display.
// return false;
if (iCompanyNameLen == 0)
{
    bValid = false;
    thisValidator.IsValid = bValid;
    thisValidator.Text = "Enter a Company Name";
    return bValid;
}
// The remaining data validation is not a necessity as length of entries
// should be controlled through the maxlength attribute of individual TextBoxs.
// However, I like to add an additional layer of data validation to ensure
// date integrity.
if (iCompanyNameLen > 40)
{
    bValid = false;
    thisValidator.IsValid = bValid;
    thisValidator.Text = "Company Name Limited to 40 Characters. Current Length: " + iCompanyNameLen;
    return bValid;
}
if (iContactNameLen > 30)
{
    bValid = false;
    thisValidator.IsValid = bValid;
    thisValidator.Text = "Contact Name Limited to 30 Characters. Current Length: " + iContactNameLen;
    return bValid;
}
if (iPhoneLen > 24)
{
    bValid = false;
    thisValidator.IsValid = bValid;
    thisValidator.Text = "Phone Limited to 24 Characters. Current Length: " + iPhoneLen;
    return bValid;
}
if (iFaxLen > 24)
{
    bValid = false;
    thisValidator.IsValid = bValid;
```

```
        thisValidator.Text = "Fax Limited to 24 Characters. Current Length: " + iFaxLen;
        return bValid;
    }
    return bValid;
}
}
```