TITULO

# DESARROLLO DE APLICACIONES I

## Sesión Nro. 12

# Objetivos de la sesión:

**Al culminar la presente podrás:**

- **Implementar el mantenimiento de una tabla desde formularios ASP. Net reutilizando las capas funcionales ADO, BL y el proyecto de entidades de negocio BE.**
- **Interacción con ModalPopup AJAX para agilizar las ventanas de inserción y actualización de registros.**
- **Implementar una paginación local para mejor presentación de los registros en la vista de listado.**

# Temario

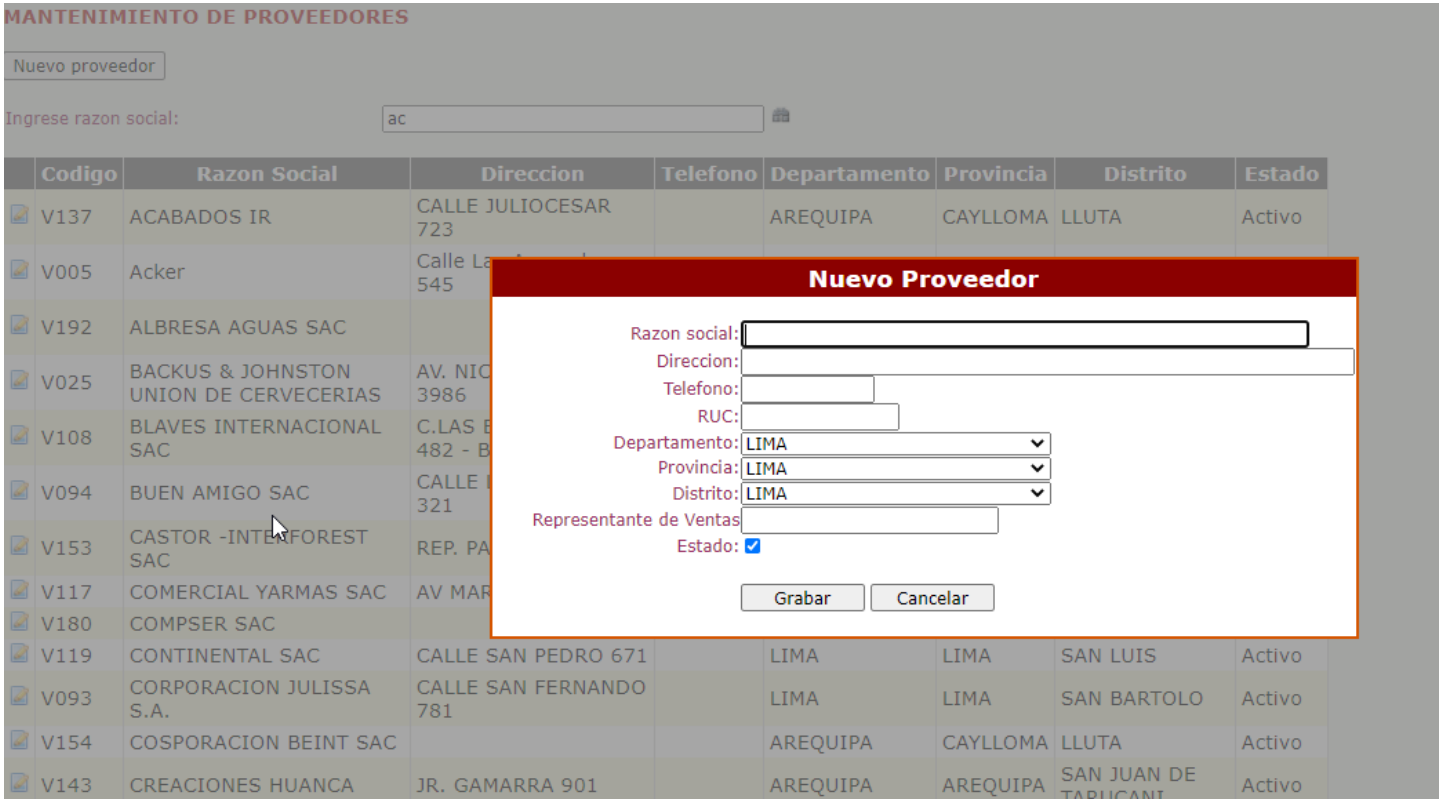
1. Adaptación de la plantilla de mantenimiento
2. Codificación de los principales eventos
3. Paginación del control GridView

# Tema Nro. 1:
# Adaptación de la plantilla de mantenimiento

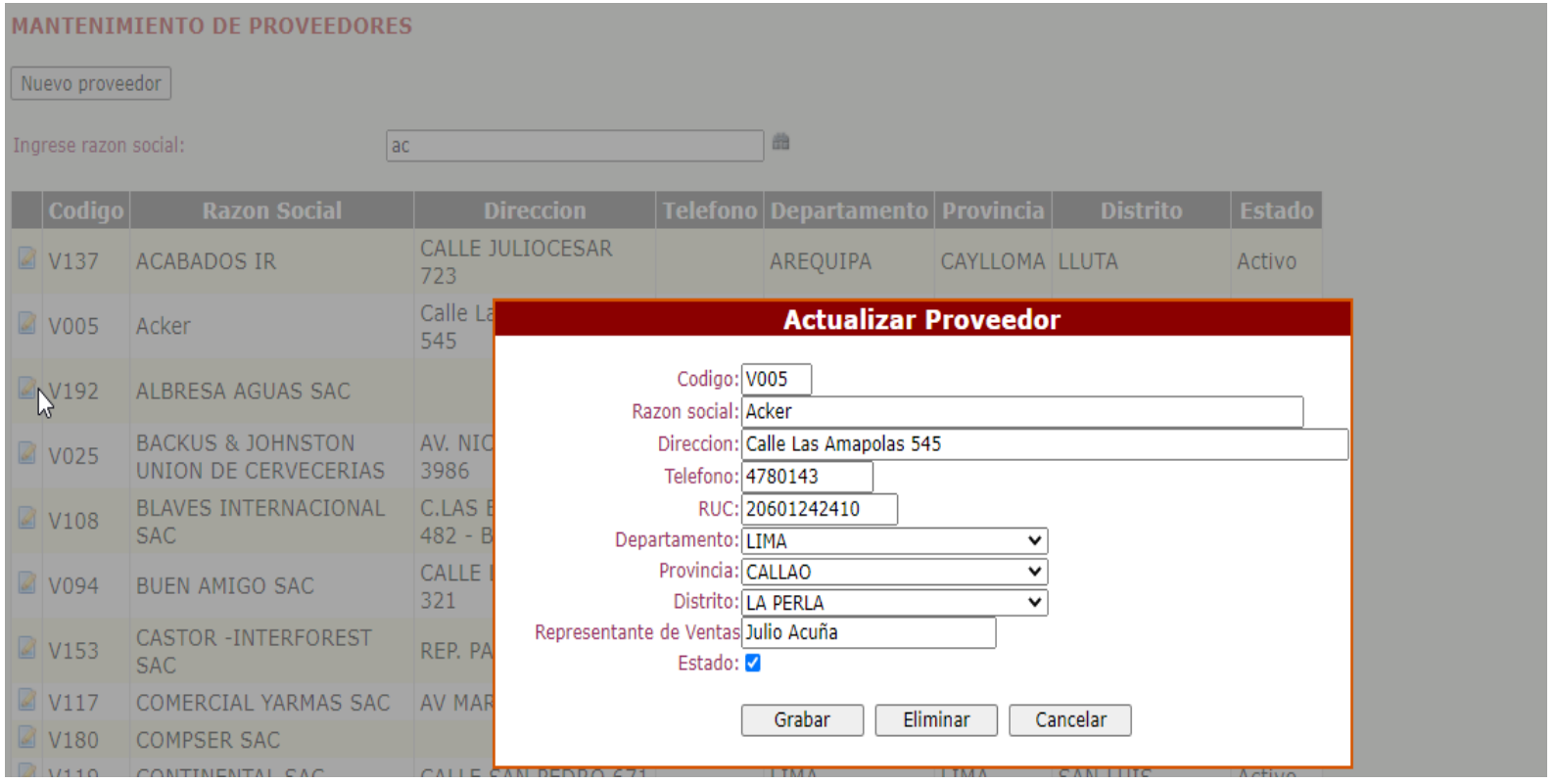
# 1.1 Adaptar el diseño de la plantilla

- Como parte de la implementación del mantenimiento de la tabla de proveedores (basándonos en lo hecho en la capa de presentación Windows) en la capa de presentación WEB, se deberá adaptar la plantilla proporcionada por su instructor, de tal manera que se tenga el siguiente diseño final

# 1.2 Diseño final WebMantProveedores: Nuevo Proveedor



# 1.2 Diseño final WebMantProveedores: Actualizar Proveedor



# Tema Nro. 2:
# Codificación de los principales eventos

# 2.1 Evento Load y la paginación del GridView

- En este parte se cargara el GridView con los proveedores manejando filtrado y paginación

```
public partial class WebMantProveedores : System.Web.UI.Page
{
    // Instancias...
    ProveedorBE objProveedorBE = new ProveedorBE();
    ProveedorBL objProveedorBL = new ProveedorBL();
    UbigeoBL objUbigeoBL = new UbigeoBL();
    DataView dtv;

    protected void Page_Load(object sender, EventArgs e)
    {
        try
        {
            if (Page.IsPostBack == false)
            {
                CargarDatos("");
            }
        }
        catch (Exception ex)
        {
            lblMensajePopup.Text = "Error : " + ex.Message;
            PopMensaje.Show();
        }

    }
    private void CargarDatos(String strFiltro)
    {
        // Creamos la vista en memoria y cargamos los datos
        dtv = new DataView(objProveedorBL.ListarProveedor());
        dtv.RowFilter = "raz_soc_prv like '%" + strFiltro + "%'";
        grvProveedores.DataSource = dtv;
        grvProveedores.DataBind();
        if (grvProveedores.Rows.Count == 0)
        {
            lblMensajePopup.Text = "No existen registros con el criterio ingresado.";
            PopMensaje.Show();
        }

    }
```

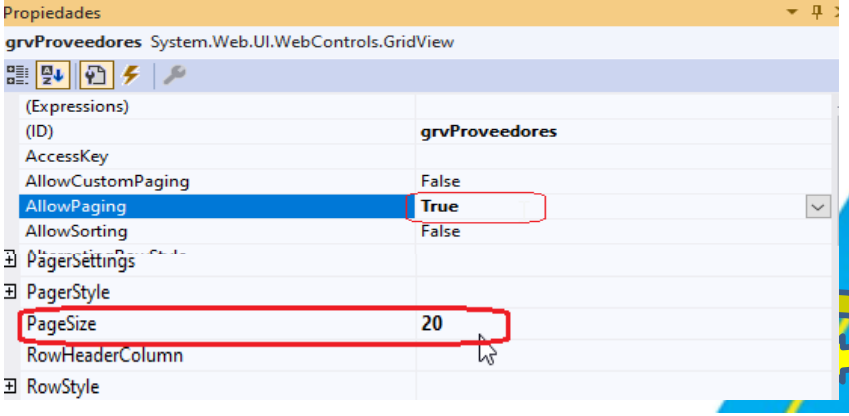
```
protected void btnFiltrar_Click(object sender, ImageClickEventArgs e)
{
    try
    {
        CargarDatos(txtFiltro.Text);

    }
    catch (Exception ex)
    {

        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }
}


protected void grvProveedores_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    grvProveedores.PageIndex = e.NewPageIndex;
    CargarDatos(txtFiltro.Text);
}
```

**No olvide definir las propiedades de paginación del Gridview**



# 2.2 El evento clic del botón Nuevo

- Para insertar un nuevo proveedor debemos preparar el modal popup PopMant, limpiando los controles y cargando el ubigeo.

```
protected void btnNuevo_Click(object sender, EventArgs e)
{
    try
    {
        // Limpiamos todos los controles del panel1 asociados al modalpopup
        // de insercion (PopMan01)
        lblMensaje1.Text = "";
        txtRS1.Text = "";
        txtDir1.Text = "";
        txtRUC1.Text = "";
        txtTel1.Text = "";
        txtRepVen1.Text = "";
        // Activo por defecto al insertar
        chkEstado1.Checked = true;
        // Cargamos el ubigeo
        CargarUbigeo(1,"14", "01", "01"); // Lima, Lima , Lima por defecto
        // Mostramos el PopMant
        txtRS1.Focus();
        PopMan01.Show();

    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }
}
```

```
private void CargarUbigeo(Int16 intAccion,String IdDepa, String IdProv, String IdDist)
{
    // Combos de ubigeo para el CRUD del proveedor
    if (intAccion == 1) // Combos de insercion
    {
        cboDepartamento1.DataSource = objUbigeoBL.Ubigeo_Departamentos();
        cboDepartamento1.DataValueField = "IdDepa";
        cboDepartamento1.DataTextField = "Departamento";
        cboDepartamento1.DataBind();
        cboDepartamento1.SelectedValue = IdDepa;
        cboProvincia1.DataSource = objUbigeoBL.Ubigeo_ProvinciasDepartamento(IdDepa);
        cboProvincia1.DataValueField = "IdProv";
        cboProvincia1.DataTextField = "Provincia";
        cboProvincia1.DataBind();
        cboProvincia1.SelectedValue = IdProv;
        cboDistrito1.DataSource = objUbigeoBL.Ubigeo_DistritosProvinciaDepartamento(IdDepa, IdProv);
        cboDistrito1.DataValueField = "IdDist";
        cboDistrito1.DataTextField = "Distrito";
        cboDistrito1.DataBind();
        cboDistrito1.SelectedValue = IdDist;
    }
    else if (intAccion == 2) // Combos de actualizacion
    {
        cboDepartamento2.DataSource = objUbigeoBL.Ubigeo_Departamentos();
        cboDepartamento2.DataValueField = "IdDepa";
        cboDepartamento2.DataTextField = "Departamento";
        cboDepartamento2.DataBind();
        cboDepartamento2.SelectedValue = IdDepa;
        cboProvincia2.DataSource = objUbigeoBL.Ubigeo_ProvinciasDepartamento(IdDepa);
        cboProvincia2.DataValueField = "IdProv";
        cboProvincia2.DataTextField = "Provincia";
        cboProvincia2.DataBind();
        cboProvincia2.SelectedValue = IdProv;
        cboDistrito2.DataSource = objUbigeoBL.Ubigeo_DistritosProvinciaDepartamento(IdDepa, IdProv);
        cboDistrito2.DataValueField = "IdDist";
        cboDistrito2.DataTextField = "Distrito";
        cboDistrito2.DataBind();
        cboDistrito2.SelectedValue = IdDist;

    }
}
```

# No se olvide….

- No olvide programar el evento SelectedIndexChanged de los combos cboDepartamento1 y cboProvincia1 para el efecto cascada del ubigeo….

```
protected void cboDepartamento1_SelectedIndexChanged(object sender, EventArgs e)
{
    CargarUbigeo(1,cboDepartamento1.SelectedValue.ToString(), "01", "01");
    //Para que al postear el evento SelectedIndexChanged se mantenga abierto el PopMan01
    PopMan01.Show();
}

protected void cboProvincia1_SelectedIndexChanged(object sender, EventArgs e)
{
    CargarUbigeo(1,cboDepartamento1.SelectedValue.ToString(),
                   cboProvincia1.SelectedValue.ToString(), "01");
    //Para que al postear el evento SelectedIndexChanged se mantenga abierto el PopMant
    PopMan01.Show();
}
```

**No olvide tampoco asignar el valor true a la propiedad AutoPostback de cada combo**

# 2.3 El evento clic del botón Grabar1

- Para grabar el nuevo proveedor debemos configurar la entidad de negocio y llamar al método InsertarProveedor

```
protected void btnGrabar1_Click(object sender, EventArgs e)
{
    try
    {
        // Puede hacer las validaciones por codigo (o agregar RequiredFieldValidator)
        // Como ejemplo solo validaremos la Razon Social
        if (txtRS1.Text.Trim() == "")
        {
            throw new Exception("La razon social es obligatoria.");
        }

        // Tras las validaciones se configura la instancia de objProveedorBE y se invoca al
        // metodo Insertar del objProveedorBL
        objProveedorBE.Raz_soc_prv = txtRS1.Text;
        objProveedorBE.Dir_prv = txtDir1.Text;
        objProveedorBE.Ruc_prv = txtRUC1.Text;
        objProveedorBE.Tel_prv = txtTel1.Text;
        objProveedorBE.Rep_ven = txtRepVen1.Text;
        objProveedorBE.Id_Ubigeo = cboDepartamento1.SelectedValue +
                                   cboProvincia1.SelectedValue + cboDistrito1.SelectedValue;
        objProveedorBE.Usu_Registro = "jleon";
        objProveedorBE.Est_prv = Convert.ToInt16(chkEstado1.Checked);
        // Insertamos el registro...
        if (objProveedorBL.InsertarProveedor(objProveedorBE) == true)
        {
            // Refrescamos el listado
            CargarDatos(txtFiltro .Text );
        }
        else
        {
            throw new Exception("No se inserto el registro. Contacte con IT.");
        }
    }
    catch (Exception ex)
    {
        lblMensaje1.Text = ex.Message;
        PopMan01.Show();// para mantener el popup abierto tras el clic en Grabar1
    }
}
```

# 2.4 El evento RowCommand del GridView

- Para actualizar un proveedor debemos seleccionarlo desde el grid, haciendo clic en el el ColumnButton “Editar” y mostrando en el PopMan02 el proveedor seleccionado

```
protected void grvProveedores_RowCommand(object sender, GridViewCommandEventArgs e)
{
    try
    {
        // Se obtiene en que fila del grid se ha seleccionado el boton
        Int16 fila = Convert.ToInt16(e.CommandArgument);
        // Validamos si el boton tiene el nombre Editar
        if (e.CommandName == "Editar")
        {
            // Limpiamos el lblMensaje2
            lblMensaje2.Text = "";
            // Obtenemos el codigo del proveedor de la celda 1 de la fila seleccionada
            String strCod = grvProveedores.Rows[fila].Cells[1].Text;
            // Obtenemos la instancia del proveedor a actualizar
            objProveedorBE = objProveedorBL.ConsultarProveedor(strCod);
            // Mostramos el proveedor en los controles del panel del PopMan02 (actualizacion del  registro)
            lblCod.Text = objProveedorBE.Cod_prv;
            txtRS2.Text = objProveedorBE.Raz_soc_prv;
            txtDir2.Text = objProveedorBE.Dir_prv;
            txtRUC2.Text = objProveedorBE.Ruc_prv;
            txtTel2.Text = objProveedorBE.Tel_prv;
            txtRepVen2.Text = objProveedorBE.Rep_ven;
            chkEstado2.Checked = Convert.ToBoolean(objProveedorBE.Est_prv);
            String Id_Ubigeo = objProveedorBE.Id_Ubigeo;
            //Manejamos el ubigeo para el panel de PopMan02
            CargarUbigeo(2,Id_Ubigeo.Substring(0, 2), Id_Ubigeo.Substring(2, 2),
                                                      Id_Ubigeo.Substring(4, 2));
            //Mostramos el PopMan02
            PopMan02.Show();

        }
    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }
}
```

# No se olvide….

- No olvide programar el evento SelectedIndexChanged de los combos cboDepartamento2 y cboProvincia2 para el efecto cascada del ubigeo….

```
protected void cboDepartamento2_SelectedIndexChanged(object sender, EventArgs e)
{
    CargarUbigeo(2,cboDepartamento2.SelectedValue.ToString(), "01", "01");
    //Para que al postear el evento SelectedIndexChanged se mantenga abierto el PopMan02
    PopMan02.Show();
}

protected void cboProvincia2_SelectedIndexChanged(object sender, EventArgs e)
{
    CargarUbigeo(2,cboDepartamento2.SelectedValue.ToString(),
                   cboProvincia2.SelectedValue.ToString(), "01");
    //Para que al postear el evento SelectedIndexChanged se mantenga abierto el PopMan02
    PopMan02.Show();
}
```

**No olvide tampoco asignar el valor true a la propiedad AutoPostback de cada combo**

# 2.5 El evento clic del botón btnGrabar2

- Si queremos actualizar los datos del proveedor debemos codificar el botón btnGrabar2 del PopMan02

```
protected void btnGrabar2_Click(object sender, EventArgs e)
{
    try
    {
        // Validamos la razon social
        if (txtRS2.Text == "")
        {
            throw new Exception("La razon social es obligatoria.");

        }
        // Tras las validaciones se configura la instancia de objProveedorBE y se invoca al metodo Actualizar
        // del objProveedorBL
        objProveedorBE.Cod_prv = lblCod.Text;
        objProveedorBE.Raz_soc_prv = txtRS2.Text;
        objProveedorBE.Dir_prv = txtDir2.Text;
        objProveedorBE.Ruc_prv = txtRUC2.Text;
        objProveedorBE.Tel_prv = txtTel2.Text;
        objProveedorBE.Rep_ven = txtRepVen2.Text;
        objProveedorBE.Id_Ubigeo = cboDepartamento2.SelectedValue + cboProvincia2.SelectedValue + cboDistrito2.SelectedValue;
        objProveedorBE.Usu_Ult_Mod = "cmontoya";
        objProveedorBE.Est_prv = Convert.ToInt16(chkEstado2.Checked);
        // Actualizamos...
        if (objProveedorBL.ActualizarProveedor(objProveedorBE) == true)
        {
            CargarDatos(txtFiltro.Text);
        }
        else
        {
            throw new Exception("No se actualizo el registro. Contacte con IT.");
        }
    }
    catch (Exception ex)
    {
        lblMensaje2.Text = ex.Message;
        PopMan02.Show();
    }
}
```

# 2.7 Eventos adicionales

- Por ultimo el evento clic del botón btnFiltrar para aplicar el filtro por razón social  y el evento RowDataBound del GridView para mostrar de color rojo los proveedores inactivos

```
protected void grvProveedores_RowDataBound(object sender, GridViewRowEventArgs e)
{
    try
    {
        // El evento recorre cada fila del grid cada vez que este se enlaza a su origen de datos
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            // Obtenemos el valor del campo  Estado de cada fila (celda 8 )
            String estado = e.Row.Cells[8].Text;
            // Si esta inactivo el color de la fuente (ForeColor) de la fila estara en rojo
            if (estado == "Inactivo")
            {
                e.Row.ForeColor = System.Drawing.Color.Red;
            }
        }
    }
    catch (Exception ex)
    {
        lblMensajePopup.Text = ex.Message;
        PopMensaje.Show();
    }
}
```

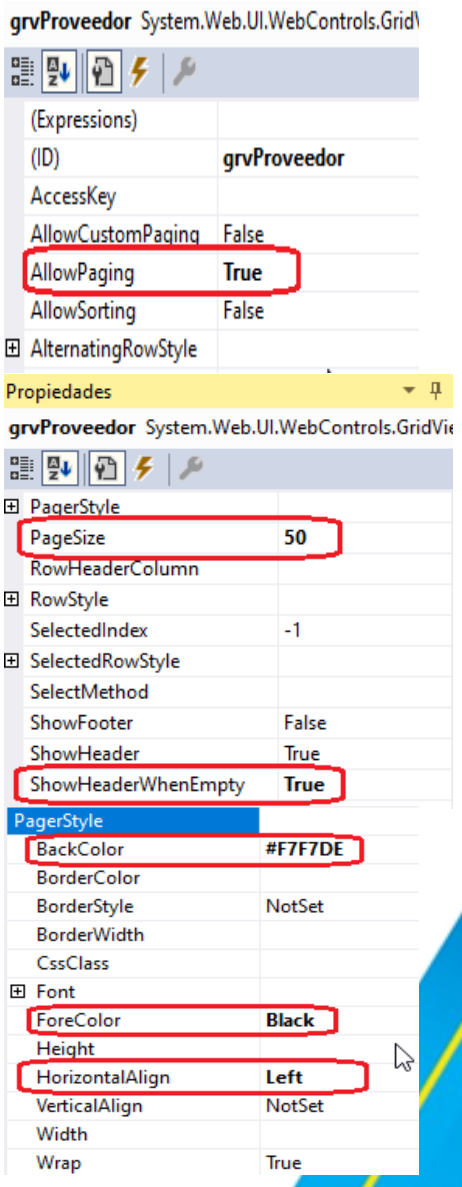
# Tema Nro. 3:
# La paginación de datos en un GridView

# 3.1 Paginando en un GridView

- Cuando hay muchos registros que mostrar en un GridView lo mas recomendable es paginarlos, para que se muestren dichos registros en paginas de 20 en 20 (recomendable) o de 50 en 50.
- Hay 2 formas de hacer esto: paginando el GridView o paginando la consulta desde un store procedure.
- En el mantenimiento de Proveedores emplearemos la primera manera, que si bien es cierto no es la mas eficiente, para volúmenes de datos menores a 1000 es aceptable.
- Veremos en un próximo ejemplo la paginación desde Store Procedure.

# 3.2 Propiedades del GirdView para paginar

- Si desea que el GridView permita paginación debe asignar el valor True a la propiedad AllowPage
- La definición del tamaño de pagina lo define en la propiedad PageSize y si desea que la cabecera del GridView este siempre visible así no hayan registros que mostrar lo hace colocando en True la propiedad ShowHeaderWhenEmpty.
- Por ultimo el estilo de la paginación lo define en el grupo de propiedades PageStyle



# 3.3 El evento PageIndexChanging

- Para pasar de pagina en pagina en los registros del GridView debe programar el evento PageIndexChanging.
- A continuación mostramos nuevamente el código de dicho evento para nuestro ejemplo del mantenimiento de proveedores:

```
protected void grvProveedores_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    grvProveedores.PageIndex = e.NewPageIndex;
    CargarDatos(txtFiltro.Text);
}
```

# LABORATORIO

Implemente el mantenimiento de la tabla Tb_Proveedor de la base de datos VentasLeon. Tome como referencia el diseño del mantenimiento hecho en Windows para dicha tabla, y oriéntelo a WEB en función de lo aprendido con el empleo de controles de la ToolKit AJAX.

# Conclusiones de la sesión:

- Las capas son reutilizables y nos permiten independizar la capa de presentación.
- Es importante interactuar con AJAX (u otra herramienta)  que consolide una interface amigable para el mantenimiento de una tabla.
- Es importante paginar para poder mostrar de manera amigable y en secuencia de paginas el listado de los registros.