ESPECIFICACION FUNCIONAL

Guía Práctica - Semana 07

Competencia:

El desarrollo de esta guía práctica te permitirá la creación de reportes en formato Excel y XML

Descripción:

La presente especificación funcional se basa en la solución **DemoVentas_3Capas**, y se compone de lo siguiente

a) Parte 1: Un laboratorio guiado para completar el ejercicio iniciado en clase con respecto a la generación de archivos Excel. Se empleará un proceso asíncrono para la generación del listado de facturas.
b) Parte 2: Un laboratorio guiado para la generación y consumo de archivos XML en aplicaciones Windows (que también se podría emplear en aplicaciones WEB)

PARTE 1: Generando un reporte Excel de Facturas (simulando una carga extensa de datos con los controles ProgressBar y BackGroundWorker)

Paso 1:

Recupera tu solución **DemoVentas_3Capas**. En le capa de datos **ProyVentas_ADO** dentro de la clase **FacturaADO** agregaremos el siguiente método:

```
public DataTable ListarFactura()
{

    cnx.ConnectionString = MiConexion.GetCnx();
    cmd.Connection = cnx;
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "usp_ListarFactura";
    try
    {
        DataSet dts = new DataSet();
        SqlDataAdapter ada = new SqlDataAdapter(cmd);
        ada.Fill(dts, "Facturas");
        return dts.Tables["Facturas"];
    }
    catch (SqlException ex)
    {
        throw new Exception(ex.Message);
    }

}
```

Este método devolverá en un objeto DataTable el resultado del SP **usp_ListarFactura**, que contiene toda la facturación encapsulada en una vista (puede revisar el store y la vista subyacente **vw_VistaFacturas**)

Paso 2:

En el proyecto **ProyVentas_BL**, dentro de la clase **FacturaBL** agregue el método que invoque al método implementado en la parte 1, así:

```
public DataTable ListarFactura()
{
    return objFacturaADO.ListarFactura();
}
```

Nota: recuerde que esta clase ya se creó y ya cuenta con la declaración de la instancia **objFacturaADO**.

Paso 3:

Ubíquese en la capa de presentación, es decir, el proyecto **ProyVentas_GUI**. En el formulario **frmListadosExcel** implementaremos la generación del listado en Excel de la facturación. Empezaremos por el evento clic del botón **btnListadoFacturas**, tal como se muestra a continuación:

```
private void btnListadoFacturas_Click(object sender, EventArgs e)
{
    try
    {
        // Mostramos controles asociados
        MostrarControles(true);
        // Se establece el inicio del trabajo asincrono del control BackGroundWorker bkgDatos
        bkgDatos.RunWorkerAsync();

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Como se aprecia, tras invocar al procedimiento **MostrarControles** con el parámetro en true (para poder mostrar el gif, label y barra de progreso), se invoca al método **RunWorkerAsync** del control **bkgDatos** y así iniciar de manera asíncrona el proceso de emisión del listado.

Paso 4:

En el evento **DoWork** del control **bkgDatos** codifique lo siguiente:

```
private void bkgDatos_DoWork(object sender, DoWorkEventArgs e)
{
    // En este evento se establece que trabajo va a realizar el control BackgroundWorker  bkgDatos....
    try
    {
        // El trabajo es obtener en un datatable las facturas mediante el metodo ListarFactura....
        DataTable dtFacturas = new DataTable();
        dtFacturas = objFacturaBL.ListarFactura();
        // Simulamos un efecto de carga con este bucle...
        // OJO: No emplee este bucle en cargas que de por si son extensas.
        // En casos normales seria solo : bkgDatos.ReportProgress(100)
        // Bucle de simulacion:
        for (int i = 0; i < 500; i++)
        {
            // Manejamos el reporte de progreso del trabajo, lo cual incide en el evento ProgressChanged  que viene luego....
            bkgDatos.ReportProgress(i / 5);
            System.Threading.Thread.Sleep(5);
        }

        // Se devuelve el resultado del trabajo del control bkgDatos...
        e.Result = dtFacturas;

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Los comentarios incluidos nos exoneran de mayor comentario al respecto. Tome en cuenta el tema de cuánto tiempo se podrá tardar en hacer el trabajo, dependiendo de la cantidad de data a procesar en el reporte. Como el proceso es asíncrono, si la creación del reporte lleva regular tiempo, podremos hacer cualquier otra operación con el sistema mientras el trabajo se va realizando en "segundo plano", lo que caracteriza al control **BackGroundWorker**.

Paso 5:

Para ir mostrando el avance del proceso en el control **prgBar** codifique el evento **ProgressChanged** del control **bkgDatos**, como se muestra a continuación**:**

```
private void bkgDatos_ProgressChanged(object sender, ProgressChangedEventArgs e)
{
    // Se va cargando la barra de progreso con el porcentaje de avance del trabajo
    // generado por el evento DoWork
    prgBar.Value = e.ProgressPercentage;
}
```

Paso 6:

Para terminar el proceso de creación del listado Excel de las facturas codifique el evento **RunWorkerCompleted** del control **bkgDatos**, tal como se aprecia:

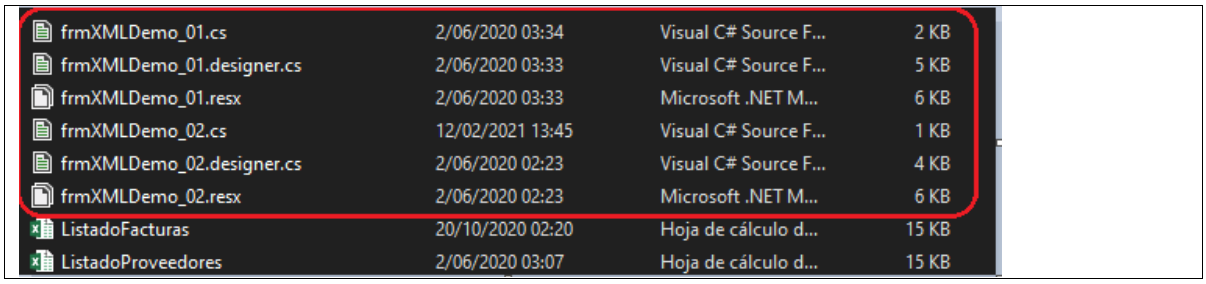
```
private void bkgDatos_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
{
    // En este evento se codifica lo necesario tras completar el trabajo asignado al  control BackGroundWorker bkgDatos
    try
    {
        // Ruta del archivo plantilla del reporte
        String rutaarchivo = @"C:\MisExcel\ListadoFacturas.xlsx";
        // Se obtiene el resultado del trabajo realizado , es decir, el datatable con la informacion de las
        // facturas (e.Result)
        DataTable dtFacturas = (DataTable)e.Result;
        // Construimos el reporte en Excel en base al archivo plantilla
        Int16 fila1 = 5;
        using (var pck = new ExcelPackage(new FileInfo(rutaarchivo)))
        {
            ExcelWorksheet ws = pck.Workbook.Worksheets["Hoja1"];
            //LLenamos el Excel con las facturas, recorriendo cada fila del datatable dtFacturas
            foreach (DataRow drFactura in dtFacturas.Rows)
            {
                ws.Cells[fila1, 1].Value = drFactura["Num_fac"].ToString();
                ws.Cells[fila1, 2].Value = drFactura["Fec_fac"].ToString();
                ws.Cells[fila1, 3].Value = drFactura["Fec_can"].ToString();
                ws.Cells[fila1, 4].Value = drFactura["Raz_Soc_cli"].ToString();
                ws.Cells[fila1, 5].Value = drFactura["Departamento"].ToString() + "-" +
                                           drFactura["Provincia"].ToString() + "-" +
                                           drFactura["Distrito"].ToString();
                ws.Cells[fila1, 6].Value = drFactura["Total"].ToString();
                ws.Cells[fila1, 7].Value = drFactura["Nom_ven"].ToString() + " " + drFactura["Ape_ven"].ToString();
                ws.Cells[fila1, 8].Value = drFactura["Estado"].ToString();
                fila1 += 1;
            }
            //Modificamos el ancho de las columnas
            ws.Column(1).Width = 30;
            ws.Column(2).Width = 50;
            ws.Column(3).Width = 90;
            ws.Column(4).Width = 40;
            ws.Column(5).Width = 45;
            ws.Column(6).Width = 45;
            // Creamos el nombre asociado al usuario logueado...
            String filename = "ListadoFacturas_" + clsCredenciales.Usuario + ".xlsx";
            // Creamos el nuevo archivo...
            FileStream fs = new FileStream(@"C:\MisExcel\" + filename, FileMode.Create);
            pck.SaveAs(fs);
            // Destruimos las instancias....
            pck.Dispose();
            fs.Dispose();
            // Enviamos el mensaje al usuario de conformidad de creacion del archivo Excel....
            MessageBox.Show("El archivo :" + filename + " se ha generado con exito.", "Mensaje",
                                        MessageBoxButtons.OK, MessageBoxIcon.Information);
            // Ocultamos controles asociados...
            MostrarControles(false);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
```

El trabajo final es la elaboración del reporte, ya obtenida la data. Con esto se debe generar el listado con un archivo cuyo nombre estará asociado al usuario que inicio la sesión. Ejecute la aplicación desde el formulario **frmLogin** y verifique.

PARTE 2: Creación de reportes con formato XML

Paso 1:

Ubíquese en la capa de presentación, proyecto **ProyVentas_GUI** y proceda a copiar y pegar los 6 archivos correspondientes a los formularios **frmXMLDemo_01 y frmXMLDemo_02**, adjuntos en la plantilla de la sesión 7:



Paso 2:

Copiados los formularios procederemos a implementar el código de los botones rotulados como **Generar XML Proveedores (btnGenerarXMLProveedores)** y **Generar XML Facturas (btnGenerarXMLFacturacion).** Empezaremos con el primero de los nombrados. En el evento clic de este botón codifique lo siguiente (observe que la plantilla tiene todo el esquema listo con el comentario "Codifique"):

```
private void btnGenerarXMLProveedores_Click(object sender, EventArgs e)
{
    try
    {
        // Se creara un documento XML con todos los proveedores desde un objeto DataTable
        dtb = objProveedorBL.ListarProveedor();
        //Creación del archivo XML con un Datatable (no olvide crear la carpeta XML o la de su preferencia)
        dtb.WriteXml(@"C:\XML\ListadoProveedores.xml");
        // Ahora escribimos el XML en un StringWriter  y lo mostramos en la caja de texto
        dtb.WriteXml(obj1XML);
        txtXML.Text = obj1XML.ToString();

        // Enviamos mensaje de conformidad..
        MessageBox.Show("El archivo XML de proveedores ha sido generado con exito.", "Mensaje",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Como se aprecia, mediante un objeto **DataTable** llamado **dtb** se genera el archivo XML empleando el método **WriteXML** sobre una ruta y nombre de archivo dados, con el contenido del listado de proveedores. Así mismo, y solo para conseguir el efecto visual el código XML se carga a una instancia de **StringWriter** llamada **obj1XML** (declarada al inicio del formulario) y dicha instancia luego se muestra en el textbox **txtXML.**

Nota: Asegúrate que la ruta exista "C:\XML\". Puedes establecer otra ruta si así lo decides.

Paso 3:

Para la generación del listado XML con las facturas, procederemos de manera similar que en la parte 2, en el evento clic del botón **btnGenerarXMLFacturacion** así:

```
private void btnGenerarXMLFacturacion_Click(object sender, EventArgs e)
{
    try
    {
        // Se creara un documento XML con todos las facturas desde un objeto DataTable
        dtb = objFacturaBL.ListarFactura();
        //Creación del archivo XML con un datatable
        dtb.WriteXml(@"C:\XML\ListadoFacturas.xml");
        // Ahora escribimos el XML en un StringWriter  y lo mostramos en la caja de texto
        dtb.WriteXml(obj1XML);
        txtXML.Text = obj1XML.ToString();
        // Enviamos mensaje de conformidad..
        MessageBox.Show("El archivo XML de facturacion ha sido generado con exito.", "Mensaje",
            MessageBoxButtons.OK, MessageBoxIcon.Information);

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

}
```

Pruebe ahora ejecutando desde el formulario frmLogin, no sin antes habilitar la correspondiente opción del menú de listados rotulada como Listados XML.

Paso 4:

Finalmente puede comprobar que los archivos XML pueden servir también como fuente u origen de datos para otro proceso. En el botón rotulado como **Consumir XML** del formulario **frmXMLDemo_01**, se mostrará como cuadro de dialogo al formulario **frmXMLDemo_02**. Revise el código del evento Load de este último (ya está codificado en la plantilla) y vera en el evento clic del botón **btnProveedores** como mediante una instancia de la clase **DataSet** llamada **dts** e invocando al método **ReadXML** se consume el archivo ListadoProveedores.xml (que ya debió ser generado en las pruebas respectivas) y se establece como origen de datos (**DataSource**) para el control **DataGridView** llamado **dtgDatos.**

```
private void btnProveedores_Click(object sender, EventArgs e)
{
    try
    {
        // Para la lectura del XML lo hacemos con un DataSet
        // Asegurese que la ruta sea la correcta...
        DataSet dts = new DataSet();
        dts.ReadXml(@"C:\XML\ListadoProveedores.xml");
        dtgDatos.DataSource = dts.Tables[0];
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}
}
```

Tras comprobar la correcta ejecución de su producto, grabe su solución y proceda según lo indicado por su instructor.