Conciencia Tecnológica No. 41, Enero-Junio 2011

Acceso a Datos con DataSets en Visual Web Developer 2008/2010

Nota de Divulgación

Ing. Jorge Alberto Rivera Guerra
WebMaster del ITJ
Departamento de Sistemas y Computación
Instituto Tecnológico de Jiquilpan, Carr. Nac. S/N Km 202
Jiquilpan, Michoacán, C.P: 59510, Tel y Fax 01(353)5330574, rsimc@hotmail.com

Resumen

Para tener acceso a una base de datos almacenada en un servidor SQL utilizando Visual Studio .Net 2008 o 2010 con DataSets, existen diversas maneras de realizarlo, la mayor parte de ellas basadas en programación compleja. En este artículo se describe una manera de realizarlo. Solo basta con agregar la carpeta ASP.NET App_Code, añadir un nuevo elemento del tipo DataSet, anexar la o las tablas de la bases de datos, crear las consultas SQL, diseñar los Webforms y escribir el código para llamar a las consultas para el usuario.

Palabras clave: DataSets, SQL, WebForms, Llave primaria, ASP.NET

Introducción

El acceso a datos en la web se ha convertido hoy en día en una necesidad dado el volumen de información que se maneja dentro de cualquier organización, es por ello que se considera importante que dentro de la comunidad de programadores se cuente con las más versátiles y poderosas herramientas que simplifiquen nuestra labor.

Los accesos simplificados a datos que se pueden crear automáticamente con la utilización de formularios y otros controles, pueden ser útiles en algunos casos, pero existen otros en los que se necesita manipular la información; como por ejemplo un sitio web de comercio electrónico, en el cual se utiliza un carrito de compras el cual es programado de manera interna para llevar el control de los productos que ha seleccionado el cliente y es aquí donde se utiliza el acceso a los datos con DataSets, ofreciendo una respuesta rápida al cliente, considerando que todo el procesamiento es llevado a cabo en el servidor que aloja el sitio web y después enviado al navegador del cliente en HTML.

A. DataSet

En [1] el DataSet es una representación de datos residente en memoria que proporciona un modelo de programación relacional coherente independientemente del origen de datos que contiene. Un DataSet representa un conjunto completo de datos, incluyendo las tablas que contienen, ordenan y restringen los datos, así como las relaciones entre las tablas.

Existen varias maneras de trabajar con un DataSet [1], que se pueden aplicar de forma independiente o conjuntamente:

- Crear mediante programación una DataTable, DataRelation y una Constraint en un DataSet y rellenar las tablas con datos.
- Llenar el DataSet con tablas de datos de un origen de datos relacional existente mediante DataAdapter.
- Cargar y hacer persistente el contenido de DataSet mediante XML.

En [1] se incluyen funcionalidades parar transportar un DataSet con establecimiento inflexible de tipos mediante un servicio Web XML. El diseño del DataSet lo convierte en idóneo para el transporte de datos mediante servicios Web XML.

Los DataSets son potencialmente utilizados para generar reportes como los menciona el programador John Charles Olamendy. [6]

Este artículo está organizado de la siguiente manera, en la sección II se mencionan los trabajos existente relacionados con el acceso a datos utilizando DataSets, en la sección III se muestra la metodología propuesta para el acceso a datos con DataSets utilizando DataTable y TableAdapter; finalmente en la sección IV se proporcionan las conclusiones.

Trabajos existentes

En esta sección se analizan las propuestas de algunos programadores web [2], [3], [5], [6] y [7] , tomando en cuenta las maneras de llenado de un DataSet [1]:

A. Crear mediante programación una DataTable.

Hernández et al. propone una solución de utilización de un DataSet mediante programación:

```
Private Sub BuildDataSet()
Dim customerOrders As New Data.DataSet("CustomerOrders")
Dim customersAsData.DataTable=customerOrders.Tables.Add("Customers")
 Dim orders As Data.DataTable = customerOrders.Tables.Add("Orders")
Dim row As Data.DataRow
   With customers
      .Columns.Add("CustomerID", Type.GetType("System.Int32″))
      .Columns.Add("FirstName", Type.GetType("System.String"))
      .Columns.Add("LastName", Type.GetType("System.String"))
    .Columns.Add("Phone", Type.GetType ("System.String"))
     .Columns.Add("Email", Type.GetType("System.String"))
   End With
   With orders
      .Columns.Add("CustomerID", Type.GetType("System.Int32″))
      .Columns.Add("OrderID", Type.GetType("System.Int32″))
      .Columns.Add("OrderAmount", Type.GetType("System.Double"))
      .Columns.Add("OrderDate", Type.GetType("System.DateTime"))
   End With

customerOrders.Relations.Add("Customers_Orders", _
  customerOrders.Tables("Customers").Columns("CustomerID"), _
  customerOrders.Tables("Orders").Columns("CustomerID"))
   row = customers.NewRow()
   row("CustomerID") = 1
   row("FirstName") = "Miriam"
   row("LastName") = "McCarthy"
   row("Phone") = "555-1212″
   row("Email") = "tweety@hotmail.com"
   customers.Rows.Add(row)

   row = orders.NewRow()
   row("CustomerID") = 1
   row("OrderID") = 22
   row("OrderAmount") = 0
   row("OrderDate") = #11/10/1997#
   orders.Rows.Add(row)

   Console.WriteLine(customerOrders.GetXml())
   Console.ReadLine()
End Sub
```

El código propuesto por [2] y [5] tiene la deficiencia de crear la DataTable dinámicamente y tardaría en procesar la consulta el servidor Web y no asociada a una base de datos SQL provocando la pérdida de datos en un momento crítico. Todo lo anterior sin considerar la presentación de los datos al usuario.

B. Llenar el DataSet con tablas de datos de un origen de datos relacional existente.

Sánchez et al. realizan el llenado del DataSet con tablas de datos de un origen de datos relacional existente mediante DataAdapter:

```
'Tipificado ---- dsCustomers.xsd
dsTDatos = New dsCustomers
'Crear el Adaptador
Dim daTDatos As New SqlDataAdapter("Select * from ClientesTyped", cn)
'Llenar el DataSet, creando el DataTable ClientesTyped
  datDatos.Fill(dsTDatos, "ClientesTyped")
'Origen del Grid Typed
dgDatosT.DataSource = dsTDatos.Tables("ClientesTyped")
With dgDatosT  'Opciones del Grid.
    .CaptionText = "DataSet Tipificado"
    .CaptionForeColor = Color.Orange
     .ReadOnly = True
End With
```

Esta propuesta utiliza un DataSet previamente creado, el cual está enlazado a un origen de datos SQL y presenta una mejoría notable, pero todavía sigue siendo complejo y teniendo 9 líneas de código.

No se considerará la carga de datos persistentes mediante XML, debido a que no corresponde al tema que se está tratando.

Metodología Propuesta

Como se ha observado en las propuestas [2], [3], [5], [6], [7] el código utilizado es complejo; por lo cual se propone la siguiente manera de llevarlo a cabo:

```
Dim TA As New dsClientesTableAdapters.ClientesTableAdapter

 TA.AltaCliente(txtNombre.Text, txtA_Paterno.Text, txtA_Materno.Text, txtDirección.Text, txtemail.Text)
```

Dónde:
dsClientes= Es el DataSet
TA= Es un objeto que se crea para generar una conexión al DataSet.

Los demás campos son cajas de texto que se encuentran dentro de un WebForm.

Como se puede observar solamente bastan 2 líneas de código para poder llevar a cabo la inserción de los datos de las cajas de texto que se encuentran en el WebForm a la base de datos. Y el tiempo que enviársela al servidor SQL sería mucho más rápido que los propuestos por [3], [4], [6] y [7], basta solamente con crear el objeto y utilizar la consulta almacenada en un procedimiento SQL.

A continuación se detalla el escenario necesario que se debe contar en visual web developer 2008/2010:

A. Estructura de la Base de Datos bd.mdf

La estructura de la base de datos está integrada solamente por la Tabla Clientes y se muestra en la siguiente figura:

Nombre de columna	Tipo de datos	Permitir val...
id	int	☐
Nombre	nvarchar(50)	☑
A_Paterno	nvarchar(50)	☑
A_Materno	nvarchar(50)	☐
Dirección	nvarchar(50)	☑
email	nvarchar(50)	☑

Fig 1. Estructura de la Base de Datos BD.mdf

Es importante mencionar que se debe establecer una Llave primaria en la tabla, para poder llevar a cabo inserciones, de lo contrario solamente nos permitirá hacer consultas a los registros. Para nuestro caso la Llave primaria es el campo "id" de la Tabla Clientes.

B. Carpeta App_Code

Para agregar la carpeta App_Code a nuestro sitio web solo basta con presionar botón derecho sobre la raíz del sitio y elegir "Agregar carpeta ASP.NET" y en el menú contextual seleccionar "App_Code". Después de esos pasos tendremos la siguiente estructura:

Fig 2. Carpetas y archivos del sitio web

C. Agregar el DataSet

La carpeta App_Code es la contenedora de DataSets y otros tipos de contenedores de clases y archivos. Dentro de ésta vamos a agregar nuestro DataSet "dsClientes.xsd" que es el corazón de nuestra aplicación. Para esto hacemos clic derecho sobre la carpeta "App_Code" y elegimos agregar nuevo elemento; dicho elemento será del tipo "DataSet" y le asignaremos el nombre "dsClientes.xsd". Una vez hecho lo anterior vamos a abrir nuestro DataSet con doble clic y entonces estamos listos para añadir la tabla clientes de la base de datos, para esto damos clic en "Explorador de servidores", pulsamos sobre el signo "+" de "BD.mdf" para desplegar la estructura de misma y expandimos la vista "Tablas" en donde tendremos nuestra tabla "Clientes", solo basta con arrastrar y soltar la tabla sobre el área de trabajo del DataSet y tendremos entonces nuestro DataSet creado.

En la figura 3 podemos observar cómo es que finalmente queda nuestro DataSet.

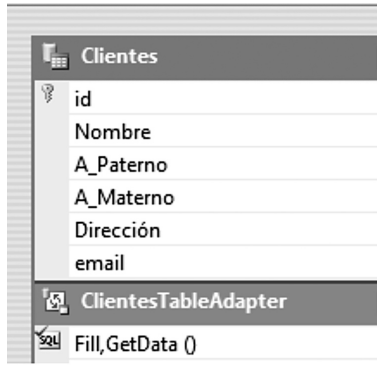


Fig 3. Generación del DataSet inicial.

En seguida se procede a generar las consultas que se necesitarán, para nuestro caso solamente ejemplificaremos como generar la consulta de inserciones de datos en la base de datos.

D. Agregar la Consulta de Inserción

Pulsamos botón derecho sobre el nombre "ClientesTableAdapter" y elegimos "Agregar Consulta"; es importante mencionar que si no aparece, debemos primero hacer clic izquierdo, para seleccionar la etiqueta y de esta manera se observe en color azul, y ahora si hacer clic derecho para elegir "Agregar Consulta". Una vez realizado lo anterior seleccionamos "Usar instrucciones SQL" y presionamos "Siguiente >", estando en este paso podemos ver en la figura 4 que se puede agregar consultas de inserción, actualización, borrado, etc.

En nuestro caso seleccionamos "INSERT" y presionamos el botón "Siguiente >" y automáticamente el asistente genera la instrucción SQL; para finalizar le damos el Nombre a la función "AltaCliente",

presionamos el botón “Siguiente >” y “Finalizar”, ver figura 5.

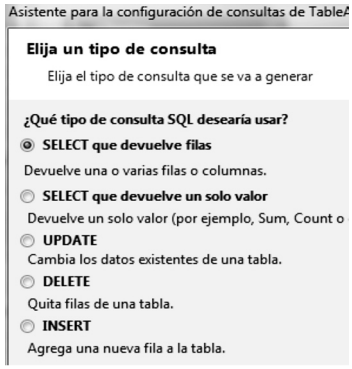


Fig 4. Asistente para la Configuración de Consultas

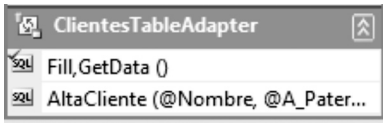


Fig 5. Consulta de inserción generada.

De manera similar se pueden crear las consultas restantes para obtener una estructura de nuestro adaptador como se muestra en siguiente figura:

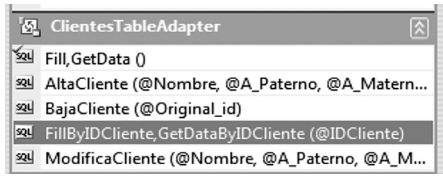
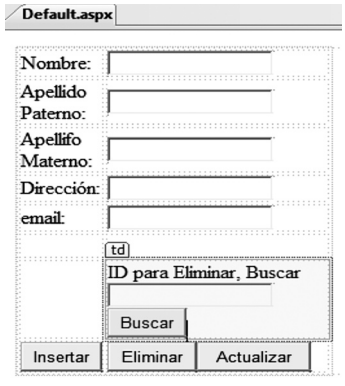


Fig 6. DataSet terminado

Por otra parte lo que hace falta es el formulario Web.

E. Creación del formulario utilizando WebForms

Utilizaremos la página “Default.aspx” que se genera cuando abrimos un nuevo sitio web en el Microsoft Web Developer 2008 y agregamos los controles para



que tenga la siguiente interfaz:

Fig 7. Interfaz de usuario

Los nombres de los controles son los siguientes: txtNombre, txtA_Paterno, txtA_Materno, txtDirección, txtemail, txtID, btnBuscar, btnInsertar, btnEliminar, btnEliminar y btnActualizar.

Conclusiones

En este artículo se describe una nueva metodología para el acceso a datos, con la cual el programador web verá reflejada la potencia de acceso a datos de una manera simplificada. Ya que las clases y objetos que se generan automáticamente dentro de los DataSets son de gran ayuda y sobre todo la velocidad que se tiene para programar utilizando la ayuda IntelliSense que nos muestra los métodos de los objetos y campos de las bases de datos de cada uno de los DataSets.

Con la reducción de código necesario para la utilización de DataSets que se propone, el usuario del sitio web verá reflejado un mejor rendimiento de la aplicación, debido a que el código que tiene que procesar el servidor web es el indispensable. Por otro lado las consultas se encuentran almacenadas en el servidor SQL y ello tiene la ventaja de que se procesan de una manera segura y rápida. Generando con lo anterior una mejor respuesta de acceso a datos al cliente que visita al sitio web. Así mismo las consultas pueden ser reutilizables para otras aplicaciones que utilicen las mismas bases de datos.

Referencias

[1] MSDN. Microsoft Developer Network.[En línea] Citado el: 8 de Marzo de 2011. http://msdn.microsoft.com/es-es/library/ss7fbaez(v=vs.80).aspx.

[2] Hernández, Gustavo Benavides. LSCV SYSTEMS. [En línea], 13 de Febrero de 2008.Citado el: 8 de Marzo de 2011. http://www.lscvsystems.com/blog/?p=16.

[3] Sánchez, Ing. Fernando Luque. El Guille. [En línea] 14 de Julio de 2005. Citado el: 8 de Marzo de 2011. http://www.elguille.info/colabora/ NET2005/FernandoLuque_DataSetsI.htm.

[4] Scott Michel, “Creating Data Access Layer”, [En línea] ASP.NET home page: http://www.asp.net/ data-access/tutorials/creating a-data-access-layer-vb

[5] ImforIT [En líena], home page: http://www.informit.com/articles/article.aspx?p=27002

[6] John Charles Olamendy, "Reporting using Datasets", [En línea] C# Corner home page: http://www.c-sharpcorner.com/UploadFile/john_charles/ReportWizardinVisualStudio200805082008 - 144910PM/ReportWizardinVisualStudio2008.aspx

[7] Roger Jennings, "Visual Basic 2005 Database Programming (2008)", Wiley Publishing Inc, 2006

Artículo recibido: 22 de noviembre de 2010

Aceptado para publicación: 13 de abril de 2011