Guillermo "Guille" Som

Acceso asistido a datos con ADO.NET 2.0 y Visual Studio 2005

Como ya adelantamos en el artículo anterior, en éste veremos cómo utilizar los asistentes de Visual Studio 2005 relacionados con el acceso a datos. En particular, veremos los pasos necesarios para crear formularios tanto para introducir datos como para mostrar el resultado de la llamada a un procedimiento almacenado de SQL Server 2005.

» ¿Asistentes? ¡Sí, gracias!

Es evidente que quien se acerca por primera vez a Visual Studio 2005 y quiere hacer cualquier tipo de aplicación, particularmente una de acceso a datos, y se encuentra que existen asistentes (*wizards*) para ayudarle a crear la aplicación, no dudará en usarlos. Además, quedará totalmente impresionado por la "belleza" del formulario resultante: botones al estilo de Office 2003, un navegador de registros, unos bonitos botones para añadir, eliminar y guardar los datos (ver figura 1). Vamos, ¡que no hay quien se resista!

Pero debemos recordar que asistente significa eso: que nos asiste, lo cual no significa que haga milagros. El asistente de acceso a datos nos va ayudando en la creación de todo lo necesario para que nuestro proyecto sepa encontrar los datos que queremos usar, para finalmente prepararlo todo y dejarlo listo y operativo. Y como veremos en un momento, lo único que tenemos que hacer para lograr esto es pulsar varias veces el botón "Siguiente". Qué bien, ¿no? Pues sí, muy bien. Pero (lamentablemente, siempre hay un pero) el código generado por el asistente, pues... en fin... que precisamente para el que se inicia no es el idóneo. Al menos si lo queremos personalizar posteriormente.

Llegados a este punto, debemos recordar que los asistentes nos ayudan a hacer las cosas que no sabemos hacer. Bueno, esto no es del todo cierto, ya que también podemos usarlos para que nos faciliten las tareas repetitivas y nos ayuden a generar el código necesario para lo que en ese preciso momento necesitamos. Pero por regla general, los usamos cuando no tenemos demasiado claro qué es lo que hay que hacer. Y esto es lo que pasa con el asistente de acceso a datos de Visual Studio 2005; que sí, nos genera una aplicación en la que solo necesitamos unos pocos clics de ratón, pero casi seguro que ésta no tiene toda la funcionalidad que esperaríamos que tuviera. Por ejemplo, si miramos la barra de botones de la aplicación mostrada en la figura 1, podemos pensar que el botón "Guardar" (el que tiene el icono del disquete) sirve para guardar las modificaciones que realicemos. En este caso, ese botón sirve para que los datos se almacenen físicamente en la base de datos; es decir, trabajamos con los datos en modo desconectado, y cuando ya queremos que se guarden en la base de datos, es cuando tendremos que pulsar en ese botón. Sabiendo esto, nos podemos llegar a plantear dos preguntas, la primera de las cuales sería: si ese botón sólo sirve para guardar los datos en la base de datos, ¿cada vez que añadimos un nuevo registro tenemos que conectarnos para guardarlo?

La respuesta es no, ese botón solo hay que pulsarlo cuando ya hayamos terminado de añadir todos los datos que tengamos que añadir. Entonces es cuando entra en juego la segunda pregunta: ya que si eso es así, ¿cómo se guardan cada uno de los datos que introducimos en modo desconectado? La respuesta es: automáticamente, sí, ¡no tenemos que hacer nada para que se guarden!

Si el lector ha leído el artículo anterior de la serie (**dotNetManía** nº 25), recordará que una de las cosas

Guillermo "Guille" Som
es Microsoft MVP de Visual Basic desde 1997. Es redactor de dotNetManía, miembro de *Ineta Speakers Bureau Latin America*, mentor de *Solid Quality Learning* Iberoamérica y autor del libro *Manual Imprescindible de Visual Basic .NET*.
http://www.elguille.info

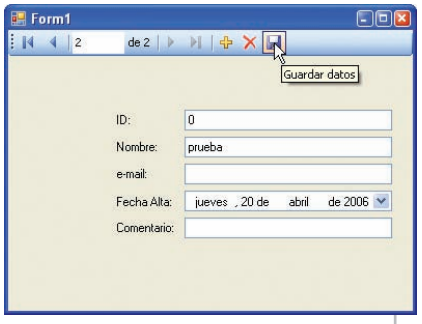

Figura 1. Guardando en la base de datos los datos introducidos

que allí comentaba era que a mí particularmente no me gusta este tipo de automatizaciones; por lo tanto, tantas admiraciones por mi parte hay que tomárselas con calma. Y no es porque el que estas cosas se hagan de forma automática, es decir, sin participación del programador, esté mal. No, lo que ocurre es que si son automáticas, no controlamos lo que pasa, o al menos las cosas pasan sin que nosotros en un principio nos enteremos de que están pasando. Y esto, sinceramente, para una aplicación creada para salir del paso está muy bien, pero si lo que queremos hacer es algo más, digamos, profesional, no deberían bastarnos tales automatizaciones.

Veamos uno de los problemas que pueden surgir con este tipo de automatizaciones.

Supongamos que el lector tiene la costumbre de crear sus tablas con un índice autonumérico; es decir, queremos mantener un campo en la tabla cuyo valor se genere automáticamente para cada nuevo registro que creemos. Ese tipo de campo tiene la cualidad de que sus valores son únicos; es decir, no puede haber dos registros en la tabla con el mismo valor.

Utilizando el asistente de acceso a datos, creamos un formulario como el de la figura 1. Es posible que queramos saber qué código es el que se ejecuta cuando el usuario pulsa en "Guardar"; lo lógico en estos casos es que hagamos doble clic en el botón correspondiente de la barra de botones para que Visual Studio nos muestre el código, que sería como el que podemos ver en el fuente 1. No es necesario que describamos el código ahora; lo que interesa es que pulsando en ese botón veremos el código que se utilizará.

```
private void pruebaBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    this.Validate();
    this.pruebaBindingSource.EndEdit();
    this.pruebaTableAdapter.Update(this.pruebaGuilleCSDataSet.Prueba);
}
```

Fuente 1. El código del método de guardar los datos

Como somos curiosos, es posible que también nos interese ver el código que hay detrás del botón "Nuevo" (el que se ejecuta al pulsar en el botón con el icono de la cruz). Como no vemos el código en el fichero del formulario, hacemos lo mismo de antes: nos vamos al formulario en modo de diseño, hacemos doble clic y... no se nos muestra nada. Bueno, sí, el método pero vacío, tal como lo vemos en el fuente 2.

```
private void bindingNavigatorAddNewItem_Click(object sender, EventArgs e)
{
}
```

Fuente 2. El método vacío de añadir nuevo registro

Y como vemos que está en blanco, podríamos pensar que hay que rellenarlo de código. Para ello, nos armamos de valor y buscamos la forma de hacer que eso funcione como queremos. En esa búsqueda encontramos que en la tabla `Prueba` del `DataSet` autogenerado existe un método llamado `AddPruebaRow` con dos sobrecargas, una en la que se pueden especificar los datos individuales de cada una de las columnas de la tabla. Por tanto, podríamos pensar que si usamos el código mostrado en el fuente 3, ya tenemos la forma de añadir un nuevo registro a la tabla.

> Dejemos que trabajen los asistentes y así nos evitaremos algunos problemas

```
private void bindingNavigatorAddNewItem_Click(object sender, EventArgs e)
{
    pruebaGuilleCSDataSet.Prueba.AddPruebaRow("nuevo", "", DateTime.Now , "");
}
```

Fuente 3. El método "supuestamente" para añadir nuevos registros

Ahora que estamos seguros de que ya tenemos el código para añadir nuevos registros que el asistente "se olvidó" de añadir, ejecutamos la aplicación (porque nuestra curiosidad nos hizo examinar el código antes de probar si ya funcionaba). Pulsamos en el botón de añadir, y todo bien, al menos aparentemente, ya que se muestran los datos que hemos indicado en los argumentos del método `AddPruebaRow`. Rellenamos los campos que están vacíos, pulsamos en el botón "Guardar", y aparentemente todo va bien. Pero si nos fijamos bien en la figura 1, seguramente detectaremos algo que no es normal; ahora veremos de qué se trata.

Si ahora volvemos a pulsar en el botón "Nuevo" para crear un nuevo registro y posteriormente pulsamos en el botón de guardar... esta vez la cosa no irá tan bien como antes. Lo que ocurrirá es que recibiremos el error que vemos en la figura 2, indicándonos que

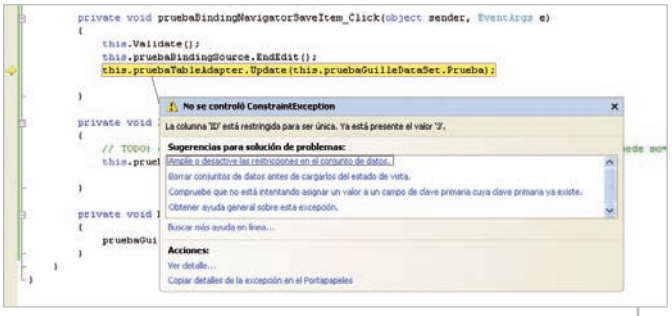

Figura 2. Error al crear un registro con un ID que ya existe

el ID debe ser único y que ya existe un registro con ese mismo ID (3).

¿Qué es lo que está ocurriendo? Pues muy claro, que estamos duplicando código, y por tanto estamos creando dos registros cada vez que pulsamos en el botón "Nuevo". Si nos fijamos en la figura 1, veremos que en la caja de texto que hay en la barra de botones se muestra el número 2 cuando en realidad sólo habíamos creado un registro. La primera vez, cuando la base de datos estaba vacía, no había problemas, ya que se crean dos registros, pero al no existir ninguno, esa duplicidad la gestiona bien, pero si ya tenemos registros, sí que recibiremos ese error.

¿Qué debemos sacar en claro de todo esto? Que si usamos el asistente, dejemos al asistente hacer su trabajo y adaptémonos a lo que ha hecho. Si lo que buscamos es escribir nosotros el código, entonces lo mejor es no usar el asistente y escribir todo el código prácticamente de forma manual. De cómo escribir nuestro código "a mano" nos ocuparemos en el próximo artículo; en éste nos vamos a centrar en los asistentes, pero... dejemos que trabajen ellos y así nos evitaremos algunos problemas.

Asistentes para introducir datos y crear consultas

A continuación, vamos a usar el asistente de acceso a datos de Visual Studio 2005 para crear una aplicación como la que hemos estado comentando. Después crearemos un procedimiento almacenado (*stored procedure*) de SQL Server en el que se requiere un parámetro de fecha, y usaremos el asistente para que cree un formulario en el que podamos realizar consultas basándonos en ese procedimiento almacenado. Empecemos por el formulario de introducción de datos.

Crear un formulario de acceso a datos de forma asistida

Para seguir los pasos que vamos a indicar ahora, necesitaremos tener una base de datos llamada `PruebaGuille` en la instancia local de `SQLEXPRESS`. Esa base de datos tendrá una tabla llamada `Prueba`, y la creación de la base de datos y la tabla la podemos efectuar por medio de la utilidad aquí descrita y que puede descargar con el material de apoyo de este artículo.

> El asistente nos genera una aplicación con solo unos pocos clics de ratón, pero casi seguro que ésta no tiene toda la funcionalidad que esperaríamos que tuviera

Lo primero que haremos es crear un nuevo proyecto del tipo "Formulario de Windows". Una vez que tengamos el formulario en modo de diseño, seleccionamos del menú "Datos" la opción "Agregar nuevo origen de datos". Esto hará que se muestre el asistente de acceso a datos.

Empezaremos por indicarle al asistente el origen de los datos que queremos usar. Se nos dará a elegir entre tres opciones: "Base de datos", "Servicio Web" y "Objeto"; en nuestro caso, será una base de datos.

El siguiente paso es elegir una de las conexiones que pudiéramos tener o bien crear una nueva (figura 3). Nosotros elegiremos una nueva, lo que resultará en una nueva ventana en la que podremos indicar la base de datos que queremos usar. En nuestro caso, será una base de datos de SQL Server. Si en el origen de datos no tuviésemos a la vista Microsoft SQL Server (*SqlClient*), tendremos que pulsar en el botón "Cambiar" y aparecerá una nueva ventana en la que podremos indicar el tipo de base de datos a usar. Si normalmente vamos a usar conexiones de SQL Server, podemos marcar la opción "Utilizar siempre esta selección", lo que hará que de forma predeterminada se muestre la ventana de conexión para el tipo de base de datos que hayamos seleccionado, pero en cualquier

Una utilidad para crear la base de datos y tabla de ejemplo

Para utilizar la base de datos que estamos usando en este artículo (u otras bases de datos con una tabla que tenga esa misma estructura), en el código que acompaña a este artículo se adjunta una pequeña utilidad que nos permite crear tanto la base de datos como la tabla y el procedimiento almacenado usados en los ejemplos. Esa base de datos la podemos crear en cualquiera de las instancias de SQL Server que tengamos instaladas en nuestro equipo. Si sólo disponemos de SQL Server 2005 Express Edition, la instancia tendrá el formato `nombreEquipo\SQLEXPRESS` o `(local)\SQLEXPRESS`.

momento podremos seleccionar otro tipo simplemente pulsando en el botón "Cambiar".

Una vez que tenemos seleccionado el origen de datos, que en nuestro caso será de SQL Server, tenemos que indicar la instancia de SQL Server en la que está la base de datos, y una vez hecho esto, podemos seleccionar la base de datos a usar. Para comprobar que todo funciona, podemos pulsar en el botón "Probar conexión" para estar seguros de que todo va bien. El nombre de la conexión lo genera el asistente de forma automática, tal como vemos en la figura 3.

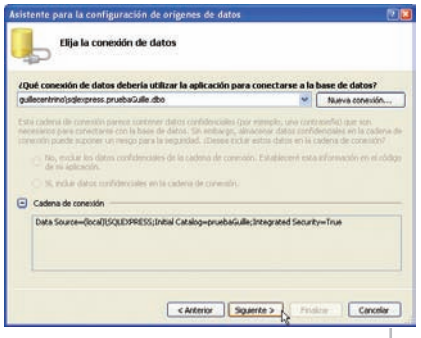

Figura 3. Seleccionar una conexión existente o crear una nueva

El siguiente paso es guardar la cadena de conexión para conectarse a la base de datos; ese nombre sí lo podemos especificar, y se almacenará en la configuración de la aplicación.

A continuación, seleccionamos la fuente de datos; en este caso será de la tabla `Prueba`, y podemos seleccionar los campos que en realidad nos interesen, ya que no es imprescindible seleccionarlos todos, como es nuestro caso (figura 4).

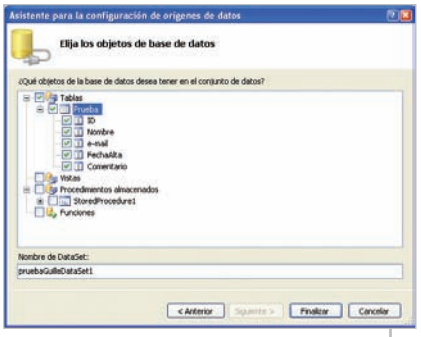

Figura 4. Selección de la tabla o tablas a usar con esta conexión

Una vez que hemos hecho la selección de la tabla (o tablas, si tuviésemos más), pulsamos en "Finalizar". Y veremos que ahora tendremos una nueva ventana acoplada con el cuadro de herramientas y el explorador de servidores. En esa nueva ventana tendremos los orígenes de datos que hayamos agregado en nuestro proyecto.

Lo siguiente será usar ese origen de datos. Para ello, seleccionamos la tabla a usar y la arrastramos hasta el formulario. De forma predeterminada, se creará un `DataGridView` para trabajar con esos datos, pero si lo que nos interesa es obtener cajas de texto para cada uno de los campos de la tabla tal y como hemos visto anteriormente, tendremos que pulsar en la lista desplegable que hay junto al nombre de la tabla y seleccionar "Detalles" (figura 5). Una vez hecho ese cambio, arrastramos la tabla al formulario y veremos que se añaden los controles correspondientes a cada uno de los campos (o columnas) que seleccionamos anteriormente, además de la barra de botones para la navegación entre los registros y la creación y eliminación de registros. El aspecto del formulario será el que se muestra en la figura 6.

Si queremos cambiar el aspecto de los controles, podemos hacerlo sin mayor problema. Por ejemplo, podríamos hacer *MultiLine* la caja de texto para el "Comentario" y cambiarle el tamaño. También podemos anclar los controles para que se ajusten al nuevo tamaño del formulario, etc.

Si ahora ejecutamos el programa (sin añadir ni una sola línea de código como hicimos antes), veremos que

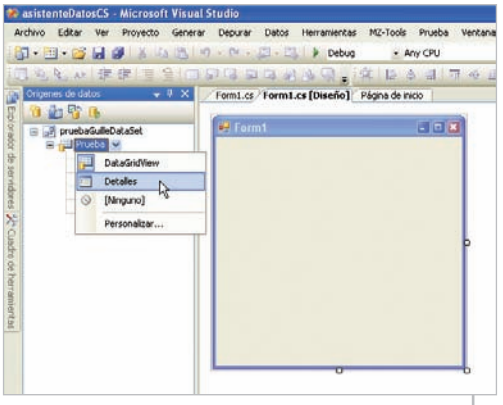

Figura 5. Selección de la forma en la que se mostrarán los datos

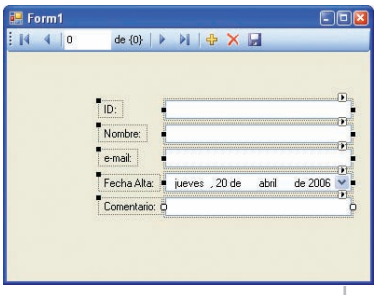

Figura 6. El asistente añade las etiquetas y controles adecuados

todo funciona bien, y podremos empezar a añadir nuevos datos a la tabla (figura 7).

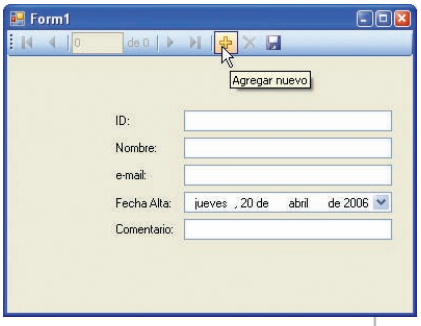

Figura 7. Sin escribir nada, la aplicación estará totalmente operativa

El único código que existe es el del `DataSet` "tipado" que se ha creado, además de un poco de código en el propio formulario, que es el que se encarga de llenar los datos al cargarse el formulario (evento `Form_Load`) y el código del botón "Guardar"; el resto de la funcionalidad es totalmente transparente para nosotros.

> **NOTA**
>
> Al utilizar el asistente para la conexión a datos, se habrá creado un **DataSet** "tipado" que es el que se usará para toda la gestión de esos datos. Y es precisamente el que se utiliza a la hora de arrastrar el origen de datos al formulario. El nombre que le da el asistente de Visual Studio 2005 está formado por el nombre de la base de datos y la palabra "*DataSet*"; en nuestro ejemplo, se llama **pruebaGuilleDataSet**. Si añadimos nuevos orígenes de datos desde la misma base de datos, la nomenclatura será la misma, pero con un número al final del nombre.

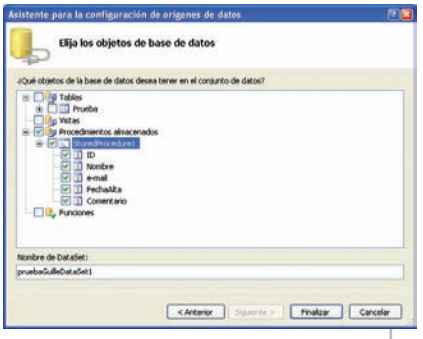

Figura 8. Selección del procedimiento almacenado a utilizar

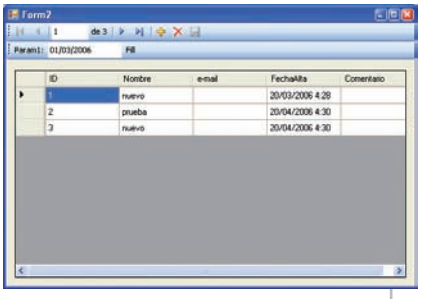

Figura 9. El DataGridView que mostrará los datos del procedimiento almacenado

Crear un formulario de consulta con el asistente

Para finalizar este artículo, vamos a utilizar el asistente para crear un nuevo **DataSet**, el cual usaremos para acceder a un procedimiento almacenado de la base de datos. Ese procedimiento almacenado tiene el código mostrado en el fuente 4. Como podemos ver, recibe un parámetro que es la fecha a partir de la que queremos mostrar los datos que tengamos en la tabla **Prueba**.

```
ALTER PROCEDURE [dbo].[StoredProcedure1]
@Param1 datetime
AS
SELECT * FROM Prueba
WHERE FechaAlta >= @Param1
```

Fuente 4. El procedimiento almacenado que usaremos para la consulta

Los pasos que tendremos que dar serán casi los mismos que hemos visto anteriormente; al menos hasta llegar a la selección de la conexión, donde podemos usar la que ya tenemos o bien crear una nueva. En nuestro caso, seleccionaremos de la lista desplegable la cadena de conexión que ya tenemos, tal como vimos en la figura 3.

Y finalmente, tal como vemos en la figura 8, seleccionamos el procedimiento almacenado y pulsamos en "Finalizar".

La información a mostrar estará en otro formulario; por tanto, debemos añadir un nuevo formulario al proyecto y ese será el que usemos para arrastrar el origen de datos al formulario. En este caso, lo dejaremos como **DataGridView** (el valor predeterminado), y el aspecto del formulario será el mostrado en la figura 9.

Si este **DataGridView** sólo lo queremos usar para mostrar los datos, sin permitir que se modifiquen los registros existentes ni que se añadan nuevos, debemos seleccionar las opciones correspondientes en las propiedades del control, que con Visual Studio 2005 son más fáciles de asignar.

> **NOTA**
>
> Si al mostrar la ventana con los orígenes de datos no se muestran los dos *DataSet*, debemos pulsar en el botón "Actualizar" para que se refresque la información de la ventana.

Si nos fijamos en la mencionada figura, veremos que hay una casilla en la que escribiremos la fecha que queremos buscar y al pulsar en el botón "Fill" (cuyo texto podemos cambiar) se rellenará el **DataGridView** con los datos correspondientes.

Por último, para que este nuevo formulario se pueda mostrar, podemos añadir un nuevo botón a la barra de botones del formulario de inicio, para que al pulsarlo se lance el formulario de datos. El código para ello es el que podemos ver en el fuente 5.

```
private void tsbConsulta_Click(object sender, EventArgs e)
{
    Form2 f2 = new Form2();
    f2.Show();
}
```

Fuente 5. El código para mostrar el formulario de consultas

Conclusiones

Como hemos podido observar, el uso de los asistentes de acceso a datos nos permite crear aplicaciones de una forma bastante rápida y prácticamente sin necesidad de escribir ni una sola línea de código. De hecho, hemos visto que por culpa de la duplicidad de código, podemos hacer que el resultado final no sea el esperado.

En próximos artículos de esta sección seguiremos con el tratamiento de datos usando ADO.NET 2.0, pero prácticamente nos dedicaremos a ver más código y menos capturas de asistentes. En el siguiente, veremos cómo hacer lo que acabamos de hacer en este artículo, pero todo a base de código "puro y duro", que es lo que a algunos nos gusta, aunque para ello tengamos que trabajar más. Como veremos, no será tan complicado; al menos una vez que sepamos los pasos que tenemos que dar y las clases de ADO.NET 2.0 que tenemos que usar.