

Enlace a datos. Control de cuadrícula

Indice

Nº-12 Enlace a datos. Control de cuadrícula.....	1
1. Introducción.....	1
1.1 Tipos de enlace de datos de ASP.NET.....	1
1.2 Cómo funciona el enlace a datos.....	2
2. Enlace sencillo.....	2
2.1 Enlace con propiedades.....	3
2.2 Problemas con el enlace sencillo de datos.....	5
2.3 Utilizar código en lugar de enlace sencillo de datos.....	6
3. Enlace datos repetitivos.....	6
3.1 Enlace de datos con un control de lista.....	7
3.2 Enlace con Colecciones fuertemente tipificadas.....	8
3.3 Utilizar la propiedad "DataValueField".....	11
3.4 Enlace de datos con ADO.NET.....	12
3.5 Crear un editor de registros.....	14
4. Controles con origen de datos.....	22
4.1 Ciclo de vida del enlace de datos.....	23
4.2 SQLDataSource.....	23
4.3 Seleccionar registros.....	24
4.4 Detalles de la ejecución de los orígenes de datos.....	30
4.5 Comandos parametrizados.....	30
4.6 Control de errores.....	37
4.7 Actualizar registros.....	37
4.8 Asistente de SQLDataSource para generar los comandos.....	40
5. Controles de datos.....	43
6. GridView.....	44
6.1 Generación automática de columnas.....	44
6.2 Definir columnas.....	51
6.3 Generación de columnas.....	52
6.4 Dar formato a la cuadrícula.....	55
6.5 Seleccionar una fila.....	63
6.6 Editar datos en la cuadrícula.....	75
7. Ordenar y paginar la cuadrícula.....	84
7.1 Ordenación.....	84
7.2 Ordenar y seleccionar.....	88
7.3 Paginación.....	91
8. Utilizar plantillas.....	93
Utilizar varias plantillas.....	98
8.1 Edición con una plantilla.....	100
9. Controles "DetailsView" y "FromView".....	105
9.1 DetailsVew.....	105
10. FormView.....	110
Ejercicios.....	115
Ejercicio 1.....	115
Ejercicio 2.....	115
Ejercicio3.....	117
Ejercicio 4.....	118

Nº-12 Enlace a datos. Control de cuadrícula

1. Introducción

En el capítulo anterior vimos como utilizar ADO.NET para recuperar información de bases de datos. Cómo almacenarlos en DataSet y como aplicar cambios utilizando comandos. Estas técnicas son flexibles y potentes pero presentan algún inconveniente. Por ejemplo, podemos utilizar un DataReader o un Dataset para realizar una consulta, darle formato y ponerlo en una tabla HTML de una página web. Sin embargo, hay que escribir mucho código repetitivo para realizar todo esto: consultar, dar formato y mostrar los datos.

Repetir el código no es un problema pero ya sabes que según Murphy no está exento de errores y siempre es más incómodo de trabajar. ASP.NET nos va a ayudar evitando varios de estos pasos proporcionando unos controles que nos van a realizar el trabajo duro por nosotros, creando salidas HTML perfectas con todo el formato que queramos. Estos son los controles enlazados a datos que es el tema de nuestro capítulo, comencemos ...

La tarea principal que tendremos que realizar en los controles enlazados a datos será la de elegir el control adecuado para que nos realice la consulta y que muestre los datos con un determinado formato. La idea es muy parecida a cuando lo realizábamos en el capítulo anterior pero mucho más sencilla porque los controles tienen mucha funcionalidad ya incorporada. Por ejemplo, podemos presentar un resultado en pantalla y si el usuario lo modifica lo podemos actualizar en el momento, al estar enlazado el control con el origen de datos. Antes teníamos que construir una SQL con la actualización y ejecutarla.

Los enlaces a datos no se mantienen como una aplicación Windows ya que es imposible mantener una conexión abierta continuamente a través de Internet. El enlace de datos de ASP.NET solo funciona en un sentido, desde el origen hasta nuestra página web: desde una base de datos hasta nuestro control. Si el usuario modifica los datos en un control enlazado, nuestro programa actualizará el registro en la base de datos. Los controles que veremos más adelante y que utilizan esta técnica del enlace a datos nos proporcionarán gran potencia y flexibilidad. Antes de ver estos controles avanzados debemos conocer como funciona este enlace de datos a controles ASP.NET

1.1 Tipos de enlace de datos de ASP.NET

Tenemos dos tipos de enlaces de datos ASP.NET: enlace a un valor individual o a una repetición de datos:

- Enlace a un valor individual. Utilizaremos este tipo de enlace cuando queramos añadir información en cualquier parte de la página. Podemos poner esta información por ejemplo, en una propiedad de un control como un texto dentro de una etiqueta HTML. Nos permite obtener una variable, una propiedad o una expresión e insertarla dinámicamente en la página.
- Enlace a valor repetitivo. En este caso nos permite mostrar una tabla entera. Al contrario que el caso anterior para poder hacer esto necesitaremos de algún control que nos permita realizar la presentación de estos datos, que como te imaginas, será del tipo de una tabla. Pero también puede ser una lista o un grupo de casillas de verificación. Y también al contrario que el caso anterior, donde el origen de datos es una base de datos en este caso, el origen de datos puede ser una matriz o una colección.

1.2 Cómo funciona el enlace a datos.

Dependiendo de la forma de trabajo elijamos (las dos anteriores) nuestro enlace funcionará de forma distinta. Para un valor sencillo insertaremos una expresión de enlace a datos dentro del código de la página “.aspx” , no en la página de código detrás (.vb), luego volveremos a mezclar el código HTML con el ASP.NET. En el segundo caso, en el de un dato repetitivo, utilizaremos un control especial y con sus propiedades realizaremos el enlace a los datos.

Una vez que hemos hecho este enlace lo activaremos mediante el método "Databind()" que es una parte fundamental de todo este proceso. Podemos enlazar un control sencillo, un control complejo o la página entera y con Databind realimentaremos ese enlace obteniendo datos actualizados. Normalmente llamaremos a este método en el "Page_Load" si no lo ponemos ASP.NET no realizará el enlace a los datos y los controles estarán vacíos. Vamos a trabajar ya con estos dos métodos.

2. Enlace sencillo

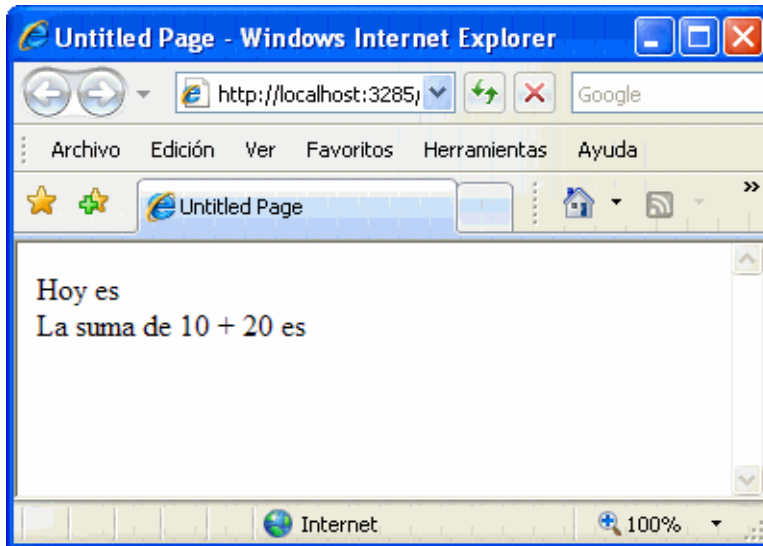
El enlace sencillo se parece a cuando empezamos a practicar con ASP.NET y escribíamos texto dinámico. Por ejemplo:

```
<%# expresion %>
```

Veamos un sencillo ejemplo para ver como se puede escribir de forma dinámica en la página. Crea una página sencilla y pon este código:

```
<body>
  <form id="form1" runat="server">
    <div>
      Hoy es <%#Now%><br />
      La suma de 10 + 20 es <%#10 + 20%><br />
    </div>
  </form>
</body>
```

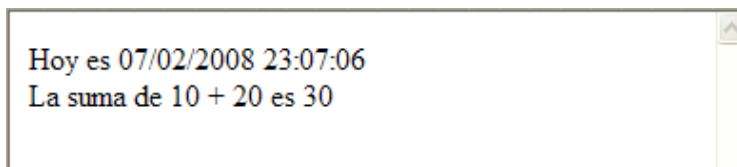
Ejecuta la página y el resultado es:



Vaya, no funciona como quisiéramos, nos debe faltar algo. Hemos puesto un código dinámico pero no parece que se ponga en marcha. Veamos porqué, antes os he dicho que podemos hacer un "DataBind" en un control sencillo, en un control enlazado a datos o en la página. ¿La página? Pues sí, si hacemos un "DataBind" de la página lo que hacemos es que se ejecute todo el código que hay enlazado o asociado con el HTML o los controles. Pon ahora lo siguiente en el evento Load de la página:

```
Protected Sub Page_Load(ByVal sender As Object,  
    Me.DataBind()  
End Sub
```

Lo que estamos haciendo es que la página se enlace a los datos, que significa que ejecutará todo el código que haya enlazado bien al HTML como en nuestro ejemplo, bien a controles mas complejos. Si ejecutamos la página:



ASP.NET sustituye las secciones de código que hemos puesto por su ejecución. Estas secciones de código se encierran en los caracteres `<%# %>`. Cuando ASP.NET se encuentre esas etiquetas ejecutará el código que haya dentro si se le solicita. Podemos mostrar la ejecución de cualquier método de cualquier objeto, consiguiendo mezclar el HTML y el código ASP.NET para enlazar los resultados. Es una forma distinta ya que siempre habíamos trabajado con los eventos desde nuestra página de .vb pero en ocasiones esto nos dará una libertad tremenda para poder modificar o presentar datos.

2.1 Enlace con propiedades

Antes hemos mostrado información estática dentro del código HTML. Pero también podremos hacer estas operaciones para manipular objetos o sus propiedades, simplemente debemos saber donde poner el código de enlace en el código HTML.

Enlace a datos. Control de cuadrícula

Veamos este ejemplo donde definiremos una variable llamada URL y la utilizaremos para apuntar a un gráfico en nuestro directorio de aplicación. Crea una página, en la página de código .vb crea una variable "URL" de tipo string y en el evento Load de la página le asignas un gráfico que exista y después llamas al método "DataBind" para que ASP.NET enlace todo el código, lo ejecute y escriba el resultado.

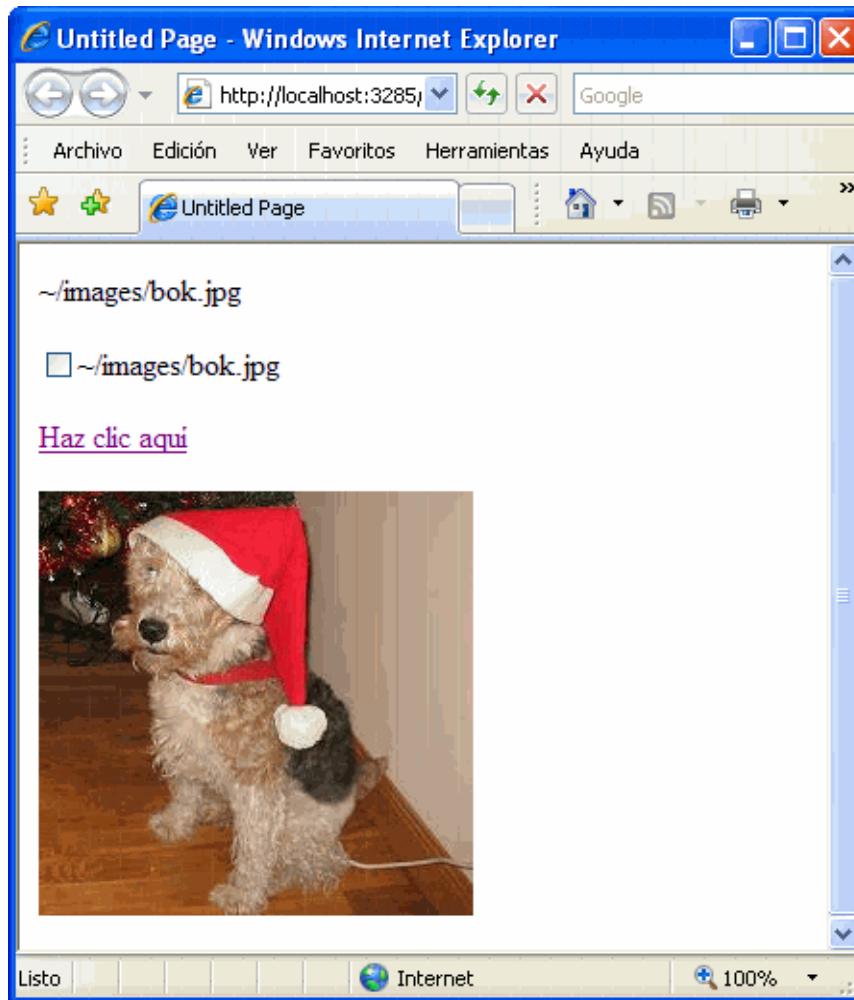
```
Protected URL As String
```

```
Protected Sub Page_Load(ByVal sender As Object, ByVal  
    URL = "~/images/bok.jpg"  
    Me.DataBind()  
End Sub
```

Ahora podemos crear los siguientes controles en nuestra página aspx:

```
<body>  
    <form id="form1" runat="server">  
        <div>  
            <asp:Label ID="lb_dinamica" runat="server"><%# URL %></asp:Label>  
            <br /><br />  
            <asp:Checkbox ID="chk_dinamico" Text="<%# URL %>" runat="server" />  
            <br /><br />  
            <asp:HyperLink ID="lnk_dinamico" Text="Haz clic aqui" NavigateUrl="<%# URL %>" runat="server" />  
            <br /><br />  
            <asp:Image ID="img_dinamica" ImageUrl="<%# URL %>" runat="server" />  
        </div>  
    </form>  
</body>
```

Si la ejecutamos, se han establecido las propiedades adecuadas:



La idea que mi perro y yo queremos transmitir es que además de poder poner código variable donde queramos, esas variables también pueden ser propiedades de los controles. El DataBind modificará las etiquetas para poner el código ejecutado, por ejemplo:

```
<asp:Image ID="img_dinamica" ImageUrl="<%# URL %>" runat="server" />
```

Si abrimos el código HTML enviado al navegador vemos que ha puesto:

```

```

Ha ejecutado el código que hay dentro de `<%# %>` y que en este caso no es mas que escribir la variable llamada "URL", que como contiene la dirección de una imagen, queda asignada al control.

2.2 Problemas con el enlace sencillo de datos

Esta técnica nos plantea dos problemas

- Primero, tenemos código dentro de nuestras páginas de interfaz del usuario. Con esto desaparece una de las ventajas de ASP.NET que era que todo el código podía estar en una página independiente (.vb). Esto supone la ruptura con esta idea pero si se asume no pasa nada, sólo que al editar las páginas debemos recordar que tenemos este tipo de código dentro de ellas.

Enlace a datos. Control de cuadrícula

- El segundo problema o inconveniente es la fragmentación del código, podemos repetir código en dos sitios que afecten al mismo control sin darnos cuenta. Por ejemplo, creamos un control en nuestra página y le asociamos una consulta. Pero en una revisión ponemos código en algún evento que sobrescribe el comportamiento con el que habíamos diseñado ese enlace. Esto puede ser accidental o no, esto puede ser un inconveniente además si esas páginas las modifican varias personas. Ya que al editarlas deberían explorar si hay algo de código incorporado en los controles en el HTML.

Pero para muchos programadores esta técnica permite una flexibilidad total. Por ejemplo, es algo que personalmente utilizo porque el antiguo ASP era así, todo incluido en la misma página. Y uno se acostumbra a la flexibilidad aun teniendo en cuenta los inconvenientes anteriores.

2.3 Utilizar código en lugar de enlace sencillo de datos

Si finalmente decidimos no utilizar el enlace de esa forma, podemos realizar el mismo proceso por código. Por ejemplo podemos utilizar un controlador de eventos para mostrar la misma salida que el primer ejemplo.

```
Sub page_load

    lb_dinamico.text = " Hoy es " & now & "<br />"

    lb_dinamico.text &= "La suma de 10 + 20 es " & 10 + 20 & "<br />"

End sub
```

La idea es que todo lo que se haga con enlaces sencillos podemos hacerlo con código. El enlace de expresiones está bien para incluir pequeñas parte de código que pueden hacernos complicada nuestra página.

3. Enlace datos repetitivos

Así como el anterior enlace es opcional ya que podemos reproducirlo con código en la página .vb los enlaces a datos repetitivos son muy útiles y se utilizarán muy a menudo en nuestras páginas.

Estos enlaces se asocian a controles especiales que al quedar vinculados producirán una salida con los distintos valores del enlace a los datos originales. Para utilizar un enlace de este tipo enlazaremos un control con un origen de datos, como un campo de una tabla y la llamada al método Databind() realizará el proceso de llenar los datos. Esto nos ahorra mucho trabajo porque automáticamente se rellenará con ese origen de datos sin tener que hacer código ni bucles para rellenar los datos. Este tipo de enlaces nos simplificará la vida aportándonos formatos avanzados y plantillas que configurarán automáticamente la forma de mostrar los datos.

Para utilizar estos enlaces de datos necesitaremos los controles de ASP.NET adecuados que los soporten. Por suerte ASP.NET proporciona unos cuantos que van a cubrir de sobra todas nuestras necesidades:

- Cuadros de lista (ListBox), cuadros desplegables (DropDownList), Listas de casillas de verificación (CheckBoxList) y Listas de botones de opción (RadioButtonList). Proporcionan una lista de un solo campo de información. Por ejemplo la lista de los autores o la lista de los libros.
- HtmlSelect. Representa al "<select>" de HTML y funciona igual que el control Web ListBox. Se utiliza solo para compatibilidad.
- Cuadrícula (GridView), vista de detalle (DetailView), FormView y ListView. Son los controles

enriquecidos mas avanzados y que van a proporcionar funciones muy avanzadas para el tratamiento de datos. Son los mas versátiles y potentes y los veremos en la siguiente sección

3.1 Enlace de datos con un control de lista

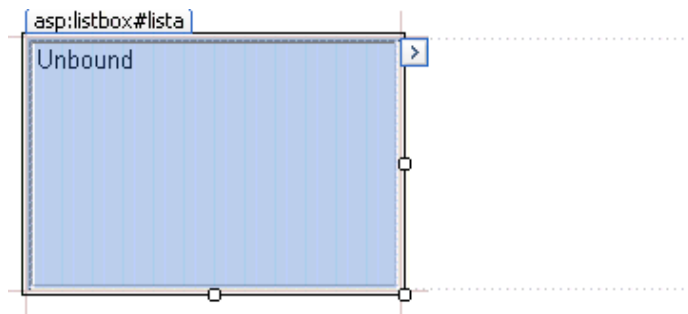
Esta es el modo mas sencillo de enlace a datos y necesitaremos sólo tres pasos para realizarlo:

1. Crearemos y rellenaremos un objeto de datos. Por ejemplo una matriz, un fichero de texto, una consulta de una base de datos en forma de DataReader, DataSet, ...
2. Enlace de los datos al tipo de control adecuado. Estableceremos las propiedades adecuadas, incluyendo "DataSource".
3. Activar el enlace mediante un método "DataBind".

El proceso es el mismo para un cuadro de lista, un cuadro desplegable, una lista de casillas de verificación y una lista de botones de opción. y además estos controles proporcionan las mismas propiedades y funcionan de la misma forma. La única diferencia es la presentación en la página web que, evidentemente, es distinta.

Ejemplo

Veamos un sencillo ejemplo utilizando una matriz. Creamos una página con un cuadro de lista:



Primero importaremos el espacio de nombres "System.collections" en nuestro código y en el evento load de la página crearemos nuestra matriz. Después estableceremos el enlace y finalmente lo activaremos:

Enlace a datos. Control de cuadrícula

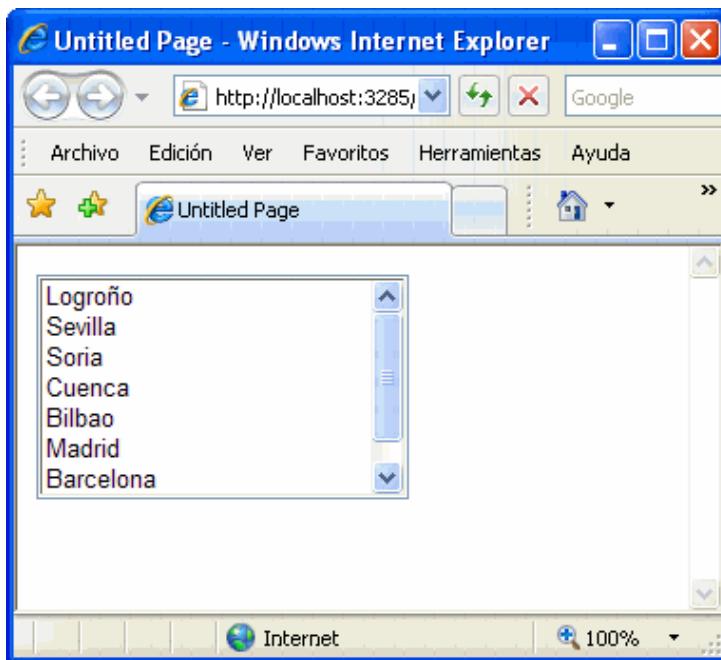
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    'Paso 1: creamos un origen de datos
    Dim ciudades As New ArrayList

    ciudades.Add("Logroño")
    ciudades.Add("Sevilla")
    ciudades.Add("Soria")
    ciudades.Add("Cuenca")
    ciudades.Add("Bilbao")
    ciudades.Add("Madrid")
    ciudades.Add("Barcelona")
    ciudades.Add("Valencia")

    '2. Establecemos el enlace con un cuadro de lista de la página
    lista.DataSource = ciudades

    '3 Activamos el enlace
    lista.DataBind()
End Sub
```

El resultado será:



Esta técnica nos ahorrará muchas líneas de código. En nuestro ejemplo no porque hemos tenido que crear la matriz pero en condiciones normales si que será muy útil.

3.2 Enlace con Colecciones fuertemente tipificadas

Se dice que es fuertemente tipificada cuando tenemos que definir los tipos de datos exactos. En ese momento los distintos tipos no se puede intercambiar. Un cuadro de lista no cumple esto porque le podemos asignar muchos tipos de datos distintos. Si recuerdas cuando vimos las colecciones, había un espacio de nombres donde teníamos muchas colecciones definidas (colores, tipos de letras, estilos de contornos, ...) llamado

Enlace a datos. Control de cuadrícula

"System.Collection.Generic".

Cuando utilizamos colecciones genéricas elegiremos el tipo de elemento y el objeto colección quedará enlazado. Esto significa que si intentamos añadir otro tipo de objeto que no sea de esa colección se producirá un error. Ahora entiendes lo de fuertemente tipificado: es de un tipo de datos u objeto y solo de ese tipo. Para poder utilizar estas colecciones añadiremos el espacio de nombres correspondiente:

```
Imports System.Collections.Generic
```

El enlace genérico de una clase ArrayList se llama "List". Así se crearía un objeto de colección "List" que almacene sólo cadenas de caracteres o "strings":

```
Dim ciudad as New List (of String)()

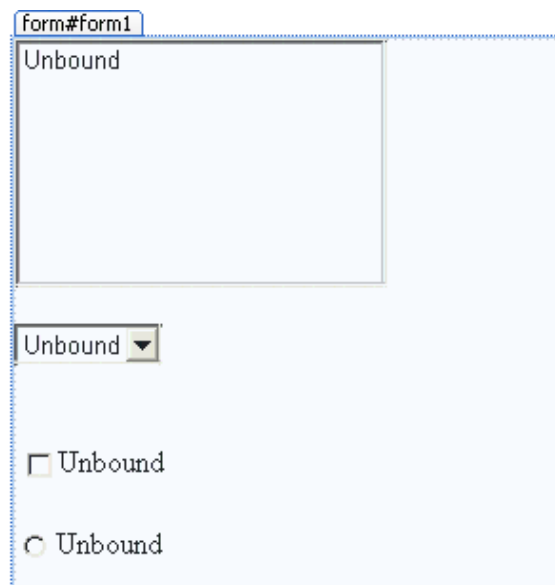
ciudades.Add ("Logroño")

ciudades.Add ("Soria")
```

La única diferencia es que necesitaremos especificar el tipo de datos cuando declaramos el objeto "List":

Ejemplo

Veamos como hacemos un enlace múltiple partiendo de nuestra lista. Lo haremos con varios controles distintos para que veas el resultado y que compruebes que todos utilizan la misma sintaxis. La página será de esta forma:



Y ahora enlazamos todos en el código:

Enlace a datos. Control de cuadrícula

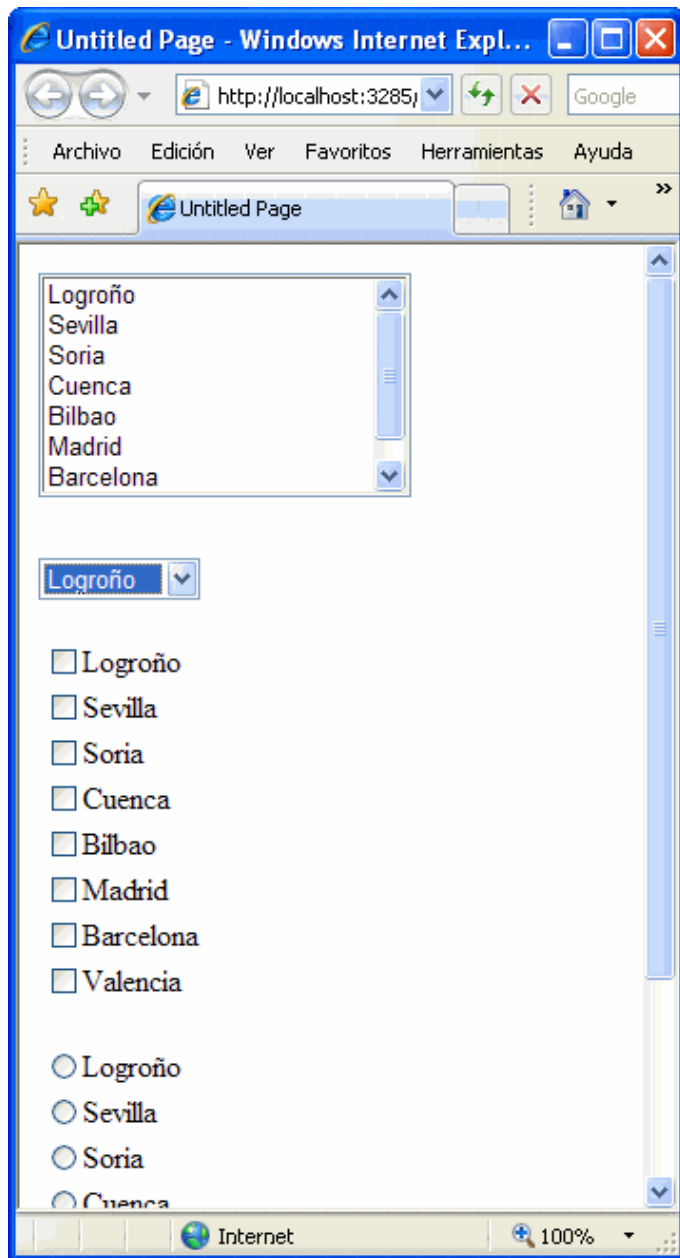
```
'Paso 1: creamos un origen de datos
Dim ciudades As New ArrayList

ciudades.Add("Logroño")
ciudades.Add("Sevilla")
ciudades.Add("Soria")
ciudades.Add("Cuenca")
ciudades.Add("Bilbao")
ciudades.Add("Madrid")
ciudades.Add("Barcelona")
ciudades.Add("Valencia")

'2. Establecemos el enlace con un cuadro de lista de la página
lista.DataSource = ciudades
cuadro.DataSource = ciudades
Radios.DataSource = ciudades
checks.DataSource = ciudades

'3 Activamos el enlace
Me.DataBind()
```

Para producir este resultado:



Como has visto no hemos utilizado un "databind" para cada uno de ellos sino que al final hemos puesto un "me.databind()". "me" ya sabes que significa la página actual, así que lo que produce es que se enlacen todos los orígenes de datos definidos.

3.3 Utilizar la propiedad "DataValueField"

Todos los controles de lista que permiten en enlace de datos permiten también la propiedad "DataValueField" que nos va a permitir añadir información a cada elemento. Esto es muy útil que lo sepas porque vamos a querer casi siempre que nos devuelva un valor. Por ejemplo, en lugar de la ciudad elegida nos interesa más que nos devuelva el índice de la ciudad en la base de datos. Verás como si nos hace falta esta información extra!. La forma de poner este dato es muy sencilla y sería esta:

```
lista.DataTextField="Valor"
```

Enlace a datos. Control de cuadrícula

```
lista.DataValueField="clave"
```

El control parece el mismo, pero si vemos el código HTML que genera verás como en la página se ve esto:

```
<select name="cuadro" id="cuadro">

  <option value="1">Logroño</option>

  <option value="2">Sevilla</option>

  <option value="3">Soria</option>

  <option value="4">Cuenca</option>

  <option value="5">Bilbao</option>

  <option value="6">Madrid</option>

  <option value="7">Barcelona</option>

  <option value="8">Valencia</option>

</select>
```

Que es un valor asociado a la ciudad elegida. Luego podremos recuperar la información mediante la propiedad "SelectedItem". Por ejemplo:

```
lb_mensaje.Text = "Has seleccionado" & lista.SelectedItem.Text

lb_mensaje.Text &= "Con el valor: " & lista.SelectedItem.Value
```

Quédate con la idea porque esto lo vamos a utilizar mucho mas de lo que piensas

3.4 Enlace de datos con ADO.NET

Bien, visto el enlace con matrices ahora veremos como enlazar los controles con bases de datos. Esta será una de las operaciones que mas realicemos en nuestras páginas. Ahora iremos viendo todo este proceso de forma manual pero verás con estos conocimientos aplicados a los controles de datos que veremos luego como empezamos a poder crear ya páginas definitivas.

Los pasos para enlazar nuestros controles con un origen de datos externo, como una base de datos, siguen siendo los mismos: primero creamos el origen de datos bien en un DataReader o un DataSet. El segundo paso es rellenar el control con este origen de datos y finalmente realizar el enlace "físico".

Recuerda que el DataReader es mas rápido y consume menos recursos pero solo lo podemos recorrer desde el principio hasta el final, no podemos movernos por él, pero esto no es importante para un enlace de datos de este tipo. Vamos con un ejemplo, crearemos un dataset para enlazarlo con una lista. El dataset lo vamos a construir a mano, es decir, vamos a añadir una tabla y dentro de ella vamos a añadir filas a mano. Para que veas que no siempre se necesitan bases de datos. Los pasos que tendremos que realizar son:

1. Crear el DataSet
2. Crear una tabla "DataTable" y añadirla a la colección
3. Definir la estructura de la tabla añadiendo objetos "DataColumn" a la colección "DataTable.Columns"

4. Proporcionar los datos.

Creamos una página de ejemplo con un cuadro de lista y luego creamos el dataset, lo rellenamos y lo enlazamos:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    'Definimos un DataSet
    Dim datos As New DataSet()
    datos.Tables.Add("usuarios")

    'Definimos dos columnas
    datos.Tables("usuarios").Columns.Add("Nombre")
    datos.Tables("usuarios").Columns.Add("Ciudad")

    'Añadimos unos datos de ejemplo
    Dim fila As DataRow = datos.Tables("usuarios").NewRow()
    fila("Nombre") = "Jose"
    fila("Ciudad") = "Logroño"
    datos.Tables("Usuarios").Rows.Add(fila)

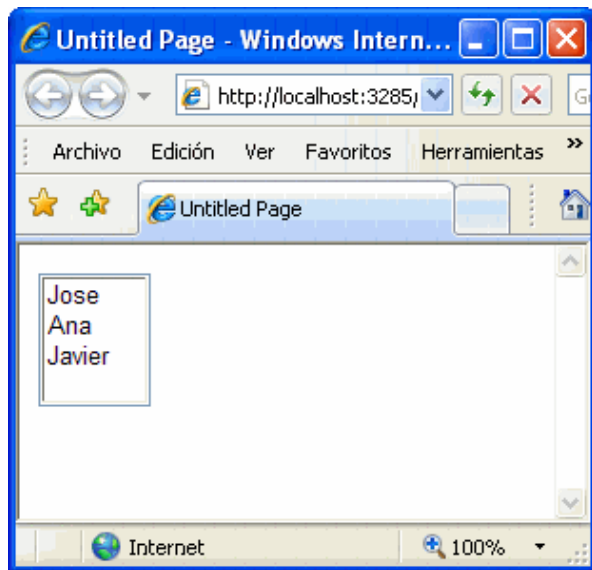
    fila = (datos.Tables("usuarios").NewRow())
    fila("Nombre") = "Ana"
    fila("Ciudad") = "Soria"
    datos.Tables("Usuarios").Rows.Add(fila)

    fila = (datos.Tables("usuarios").NewRow())
    fila("Nombre") = "Javier"
    fila("Ciudad") = "Madrid"
    datos.Tables("Usuarios").Rows.Add(fila)

    Lista.DataSource = datos.Tables("usuarios")
    Lista.DataTextField = "Nombre"

    Me.DataBind()
End Sub
```

El resultado será el cuadro de lista con las tres filas:



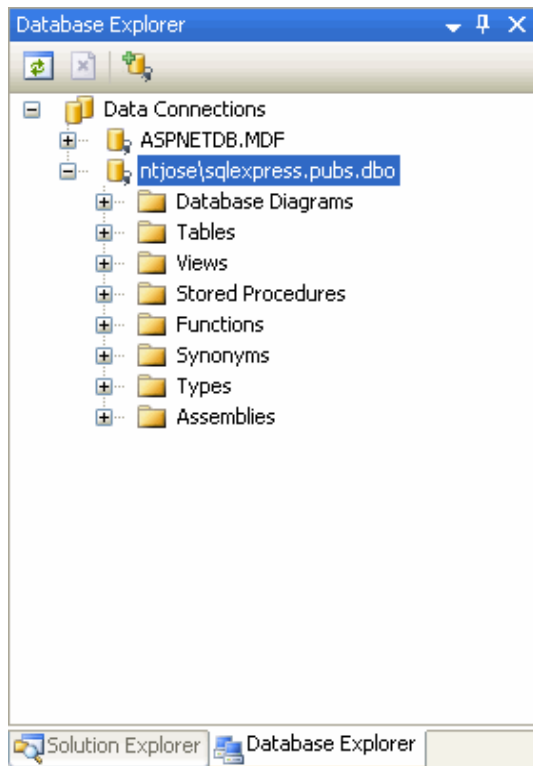
3.5 Crear un editor de registros

Este va a ser un ejemplo de cómo podemos trabajar con los enlaces en una aplicación completa de ASP.NET. Mostraremos al usuario unos registros y podrá editar y modificar uno de ellos. Como te he dicho antes, paciencia, luego veremos como se hace todo de forma automática pero de momento hay que aprender a realizarlo manualmente.

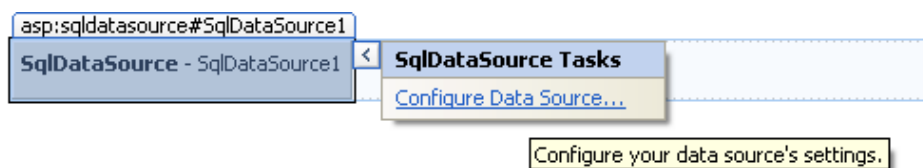
De momento tendremos nuestra cadena de conexión en nuestro ejemplo que tendremos que guardar en el fichero web.config. Podemos escribirla a mano o hacer un pequeño truco para que nos la ponga automáticamente. Sigue estos pasos...

1. Tenemos nuestra conexión a la base de datos en el explorador de bases de datos:

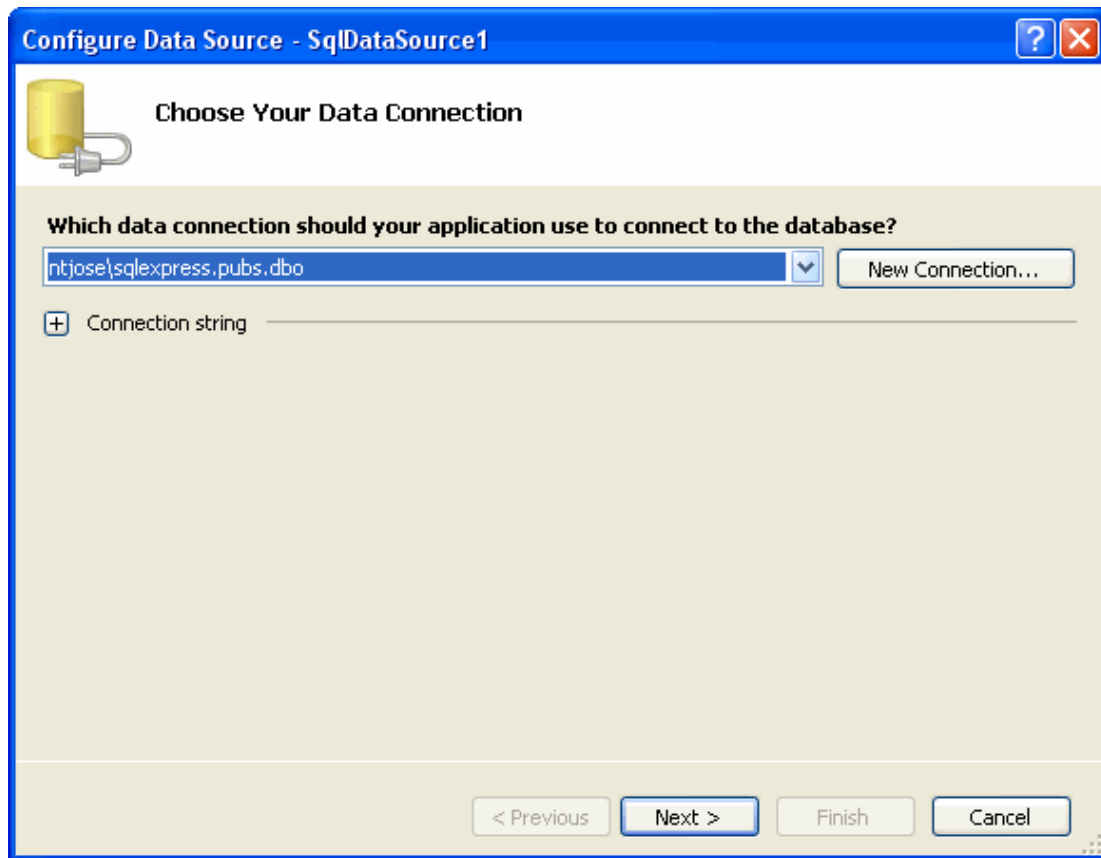
Enlace a datos. Control de cuadrícula



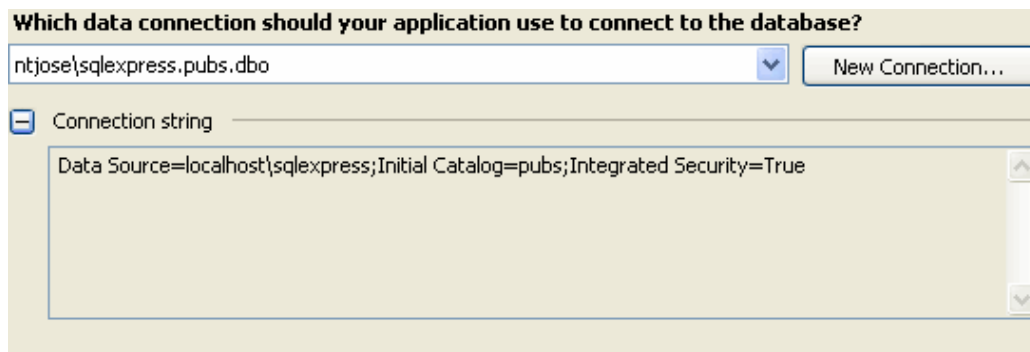
2. Creamos una página y añadimos un control de SQLDataSource:



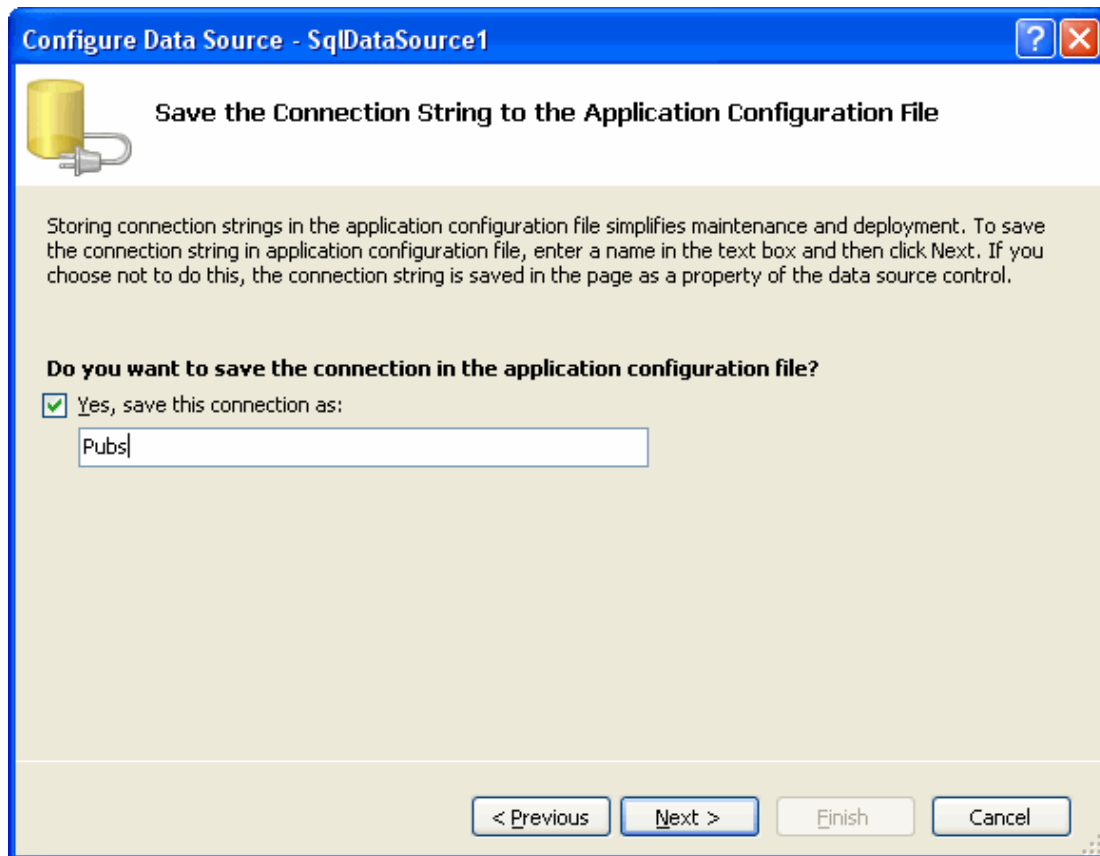
3. Hacemos clic en Configurar y pulsamos en el desplegable y verás nuestra conexión de la base de datos del servidor SQL Server 2005 Express. Ojo esto todavía no es una cadena de conexión pero como detecta que lo tenemos configurado nos lo selecciona para poder crear la cadena de conexión sobre ese servidor:



4. Si miras en los detalles de como sería la conexión con "Connection String" verás:



5. Va todo bien, ahora pulsamos abajo en "Next":



Donde nos pide que si queremos guardar la cadena conexión en el fichero Web.config. Le marcamos que si y le ponemos un nombre a nuestra cadena de conexión, por ejemplo "Pubs". Pulsamos en siguiente.

6. Vemos ahora que nos pide que consulta asociamos:

Seleccionamos por ejemplo los campos "au_id" y "au_fname" y pulsamos en continuar...

7. Finalmente pulsamos en Terminar y tendremos nuestra cadena de conexión configurada y además almacenada en el fichero de configuración web.config. Que si lo abres verás:

```
<connectionStrings>
  <add name="Pubs" connectionString="Data Source=localhost\\sqlexpress;Initial Catalog=Pubs;
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Perfecto, en lugar de editarla a mano nos hemos aprovechado de este control para crear la cadena en el fichero de configuración. Ahora sigamos con nuestro editor de registro.

Ahora vamos a escribir el código necesario para crear una consulta y añadirla a nuestro cuadro desplegable:

Unbound ▼

1

En la página hemos borrado el control anterior ya que solo nos ha servido de momento para que nos crease la consulta. Escribimos ahora el código para hacer la consulta y enlazarla al control:

Enlace a datos. Control de cuadrícula

```
Protected Sub form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles form1_Load
    If Not Me.IsPostBack Then
        'Limpiamos el cuadro de lista:
        lista_autores.Items.Clear()
        'Definimos la SQL:
        Dim sql As String = "select au_id,au_fname from authors"
        'Definimos los objeto ADO.NET:
        Dim conexion As New SqlConnection(cadena_conexion)
        Dim comando As New SqlCommand(sql, conexion)
        Dim resultado As SqlDataReader

        'Abrimos la base de datos y leemos:
        conexion.Open()
        resultado = comando.ExecuteReader()

        'Enlazamos automáticamente con la lista desplegable:
        lista_autores.DataSource = resultado
        lista_autores.DataTextField = "au_fname"
        lista_autores.DataValueField = "au_id"

        'Activamos el enlace
        Me.DataBind()

        resultado.Close()
    End If
End Sub
```

Ejecutamos la página y:



Fíjate que hemos puesto dos asignaciones en la lista:

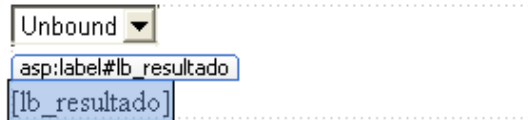
```
'Enlazamos automáticamente con la lista desplegable:
lista_autores.DataSource = resultado
lista_autores.DataTextField = "au_fname"
lista_autores.DataValueField = "au_id"
```

Por un lado hemos puesto el campo que queremos que nos muestre: "au_fname" y por otro el dato que nos va a devolver cuando seleccionemos uno de la lista: "au_id" que es precisamente el índice de ese registro. Si vemos la página HTML generada lo veremos mejor:

Enlace a datos. Control de cuadrícula

```
<select name="lista_autores" id="lista_autores">
<option value="409-56-7008">Abraham</option>
<option value="648-92-1872">Reginald</option>
<option value="238-95-7766">Cheryl</option>
<option value="722-51-5454">Michel</option>
```

Ahora haremos lo necesario para que cuando se seleccione un elemento nos muestre los libros de este autor. De momento ponemos en la página un "label" para mostrar el resultado:



Y ahora escribiremos el resultado. Tenemos que parametrizar la consulta para que nos devuelva solo los libros de este autor así que tendremos que crear una sentencia SQL del tipo:

```
select * from titles where au_id=@au_id
```

Ese valor de comparación "@au_id" es el que tendremos que extraer de la lista desplegable y ponerlo en la consulta de los libros. Así que tendremos que hacer todo el proceso completo de consulta pero añadiendo al comando un parámetro:

```
Protected Sub lista_autores_SelectedIndexChanged(ByVal sender As Object, ByVal e As S
    'Creamos un comando para buscar el registro que coincida
    Dim Seleccion As String = "Select * from authors where au_id=@au_id"

    'Creamos el comando de conexión...
    'Definimos los objeto ADO.NET:
    Dim conexion As New SqlConnection(cadena_conexion)
    Dim productos As New SqlCommand(Seleccion, conexion)

    productos.Parameters.AddWithValue("@au_id", lista_autores.SelectedItem.Value)

    'Abrimos la base de datos y leemos:
    Using (conexion)
        conexion.Open()
        Dim resultado As SqlDataReader = productos.ExecuteReader()
        resultado.Read()

        'Actualizamos la pantalla
        lb_nombre.Text = resultado("au_lname")
        lb_apellidos.Text = resultado("au_fname")
        txt_telefono.Text = resultado("phone")

        resultado.close()
    End Using
End Sub
```

En la pantalla hemos añadido los controles para poder escribir los datos, son los controles que has visto justo debajo de hacer la consulta. Hemos dejado uno editable en un cuadro de texto para poder modificar su valor:

Enlace a datos. Control de cuadrícula

Consulta1: con...lexpress.pubs) Tema_12/Edito...gistros2.aspx Tema_12/Edito...stros2.

Sin enlazar ▼

Nombre: [lb_nombre]	Telefono: <input type="text"/>
Apellidos: [lb_apellidos]	<input type="button" value="Actualizar"/>

El resultado si la ejecutamos y seleccionamos un elementos es:

Untitled Page - Microsoft Internet Explorer proporcionado por Metzeler ...

http://localhost:32364/Wel Google

Google G Configuración

Untitled Page

Innes ▼

Nombre: del Castillo	Telefono: 615 996-8275
Apellidos: Innes	<input type="button" value="Actualizar"/>

Listo Intranet local 100%

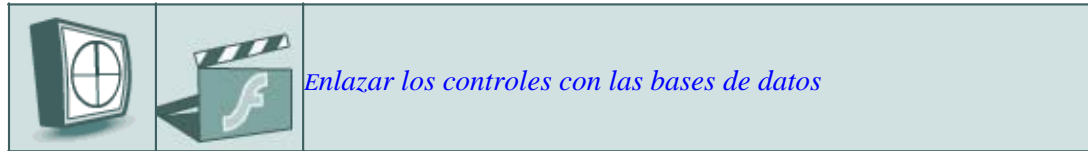
Y ahora el código para el evento clic y así podemos actualizar el campo del precio de los libros.

```
Protected Sub Btnn_actualizar_Click(ByVal sender As Object, ByVal e As System.EventArgs) Han
    'Ahora actualizamos el teléfono del autor
    Dim Sql = "Update authors set phone=@phone where au_id=@au_id"
    Dim conexion As New SqlConnection(cadena_conexion)
    Dim comando As New SqlCommand(Sql, conexion)

    comando.Parameters.AddWithValue("@phone", CType(txt_telefono.Text, String))
    comando.Parameters.AddWithValue("@au_id", lista_autores.SelectedItem.Value)

    'Ejecutamos la actualización
    Using conexion
        conexion.Open()
        comando.ExecuteNonQuery()
    End Using
End Sub
```

Podríamos mejorar mucho esto poniendo controles de validación, pero la idea es ver los pasos de las consultas, enlaces y como hemos vuelto a poner los parámetros en la sentencia SQL que nos ayudarán mucho en la construcción correcta de las sentencias SQL.



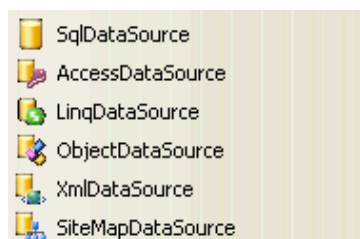
4. Controles con origen de datos

En el capítulo anterior vimos como conectarnos a bases de datos, ejecutar consultas, recorrer los registros y muchas acciones mas con las tablas. Ahora vamos a realizar el enlace de estas tablas con los controles especiales para almacenar estos orígenes de datos. Aquí entran en acción los controles enlazados a datos que verás son una forma sencilla de mostrar los datos con muchas propiedades para darles estilos y un gran control sobre sus eventos.

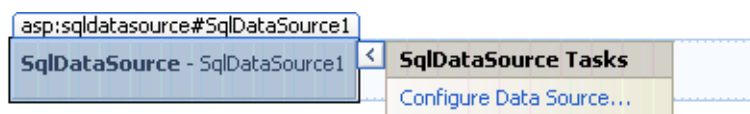
Los controles con origen de datos son todos los que implementan la interface "IDataSource" que son...

- **SqlDataSource.** Que nos permite conectarnos a cualquier origen de datos que sea proveedor de datos de ASP.NET: SQL Server, Oracle, OLE DB y ODBC. Nunca mas escribiremos el código para acceder a los datos
- **AccessDataSource.** Este origen de datos permite leer y escribir datos en bases de datos de Access. Como comentario ya habitual mío, recordarte que Access sólo debe utilizarse para pruebas y sitios web muy pequeños, para cualquier intento serio de página debe hacerse con SQL Server.
- **ObjectDataSource.** Permite conectarnos a clases de datos personalizadas.
- **XMLDataSource.** Permite conectarse a ficheros XML
- **SiteMapDataSource.** Permite conectarnos a un fichero ".sitemap" con la estructura de navegación de nuestro sitio web.

Que si te fijas en los controles disponibles dentro de la sección Data son:



Vamos a crear una página nueva y ponemos un control del tipo "SqlDataSource":



Este control no crea ningún objeto en la página, es decir, añadimos la capacidad a la página para que otros controles lo utilicen. Por tanto no será visible, digamos que define un enlace a una base de datos y luego otros controles utilizarán esa definición.

4.1 Ciclo de vida del enlace de datos

Los controles de origen de datos realizan dos tareas fundamentales:

- Pueden recuperar datos y proporcionarlos a un control enlazado. De esta forma el control se rellena automáticamente sin tener que llamar a "DataBind()"
- Pueden actualizar el origen de datos. Podemos utilizar controles que nos permitan editar filas y actualizar sus datos.

Pero antes de utilizarlos tenemos que entender un poco el ciclo de vida de la página. La secuencia es esta:

1. Se crea el objeto "page", basado en el fichero .aspx
2. Comienza el ciclo de vida de la página y se activan los eventos "Page.Init" y "Page.Load"
3. Se ejecutan los demás eventos de controles
4. Si el usuario ha aplicado algún cambio el control de origen de datos realiza ahora las tareas de actualización. Si se actualiza una fila se disparan los eventos "Updating" y "Updated". Si se inserta una fila se disparan "Inserting" e "Inserted" y si se borra una fila "Deleting" y "Deleted"
5. Se activa el evento "Page.PreRendered"
6. El origen de datos realiza las consultas e inserta los datos en los controles enlazados. Esto sucede la primera vez que se solicita la página y en cada "postback" asegurando que se tienen los últimos datos de la base de datos. Se activan los eventos "Selecting" y "Selected"
7. Se termina de "renderizar" o construir la página

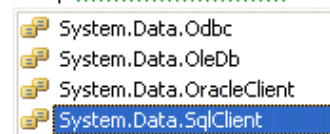
4.2 SQLDataSource

Representa una conexión a una base de datos que utiliza un proveedor ADO.NET. SqlDataSource necesita ahora una forma genérica para poder crear los objetos que necesita: Connection, Command y Datareader, que los conoces de sobra. La única forma posible es creando los objetos específicos según el proveedor de datos:

- System.Data.SqlClient
- System.Data.OracleClient
- System.Data.OleDb
- System.Data.Odbc

Esos son los cuatro tipos de proveedores que podemos utilizar con SqlDataSource, por ejemplo:

```
<asp:SqlDataSource ID="SqlDataSource1" ProviderName=| runat="server"></asp:
```



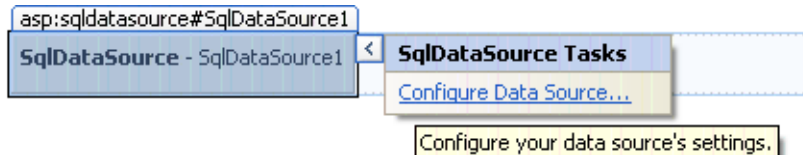
Al intentar editar ese control en el IDE y poner el proveedor con la propiedad "ProviderName" nos muestra ya los cuatro tipos de origen de daos que podemos asignarle a este objeto. Aunque podemos omitirlo porque por defecto se asume que el origen es de tipo "SQLClient".

El siguiente paso será indicarle la cadena de conexión, imprescindible para poder hacer la conexión. Podemos ponerla aquí directamente o cargarla de nuestro "web.config" como ya hicimos ejemplos atrás. La cadena de conexión se incluiría de esta forma:

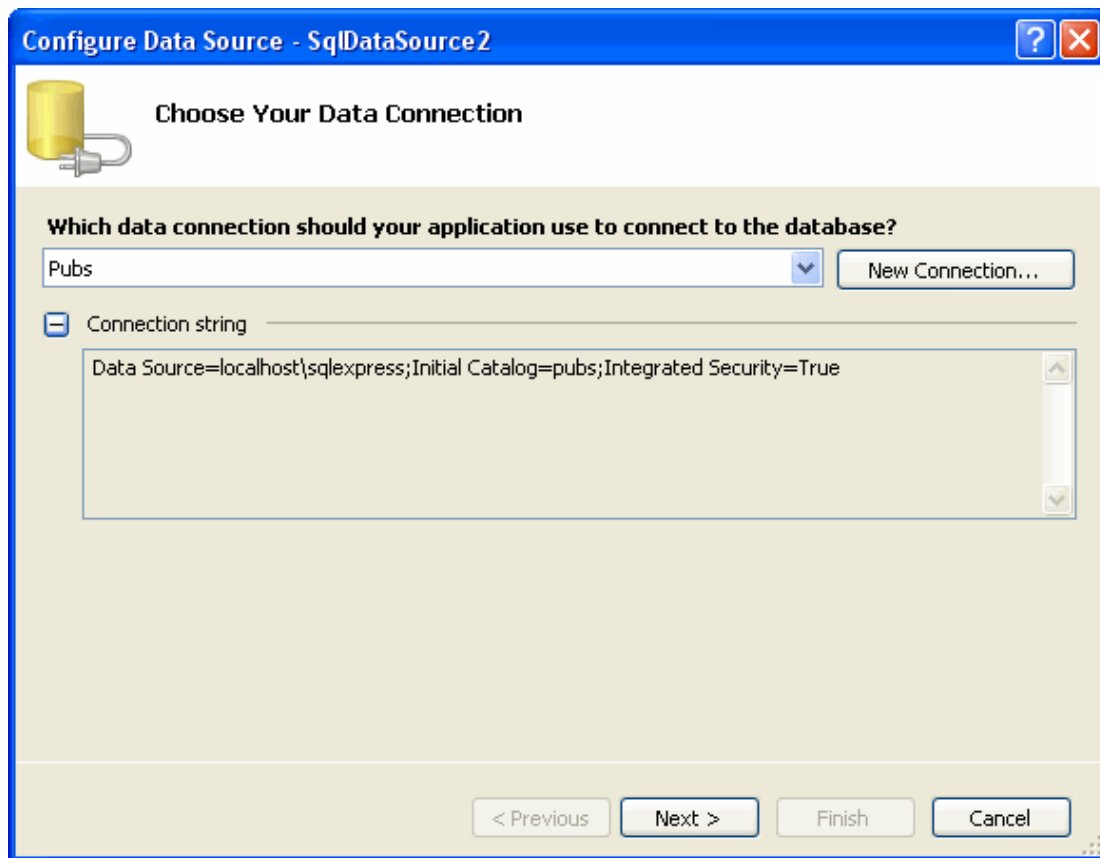
Enlace a datos. Control de cuadrícula

```
<asp:SqlDataSource ConnectionString="<%$ ConnectionStrings:Pubs %>" ... />
```

De todas formas será un poco más fácil si lo hacemos con el IDE, así que así sería... hacemos clic en:



Que nos permitirá seleccionar la cadena de conexión, en este caso ya tenemos la que creamos antes así que la seleccionamos:



En el cuadro desplegable nos mostrará todas las configuraciones de conexiones a bases de datos que tengamos en el fichero "web.config". Además podemos ver los detalles de esta cadena de conexión en el recuadro de abajo si pulsamos en el recuadro que tiene al lado con un "+". Como ves aprendemos como se hacen las cosas a mano y luego vemos los atajos con el IDE. Mejor así porque aprenderemos porqué se hacen estos pasos. Sigamos con nuestro ejemplo...

El siguiente paso será seleccionar los registros....

4.3 Seleccionar registros

Podemos utilizar los controles SqlDataSource para recuperar datos de una consulta. Y además añadir comandos para "Insertar, actualizar y borrar datos". Para estas acciones de proporcionan cuatros propiedades:

Enlace a datos. Control de cuadrícula

SelectCommand, InsertCommand, UpdateCommand y DeleteCommand que mediante unas cadenas de caracteres indicarán los comandos de cada caso.

Especificaremos solo los comandos que queremos ejecutar, por ejemplo, para una consulta sólo definiremos el de "SelectCommand". Veámoslo en la pantalla anterior. Pulsamos siguiente y si es la primera vez que utilizamos la conexión nos ofrecerá la posibilidad de guardarla en el fichero de configuración "web.config". Pulsamos siguiente y:

Configure Data Source - SqlDataSource1

Configure the Select Statement

How would you like to retrieve data from your database?

☒ Specify a custom SQL statement or stored procedure

☐ Specify columns from a table or view

Name: authors

Columns:

<input type="checkbox"/> *	<input type="checkbox"/> city
<input type="checkbox"/> au_id	<input type="checkbox"/> state
<input type="checkbox"/> au_lname	<input type="checkbox"/> zip
<input type="checkbox"/> au_fname	<input type="checkbox"/> contract
<input type="checkbox"/> phone	
<input type="checkbox"/> address	

☐ Return only unique rows

WHERE...

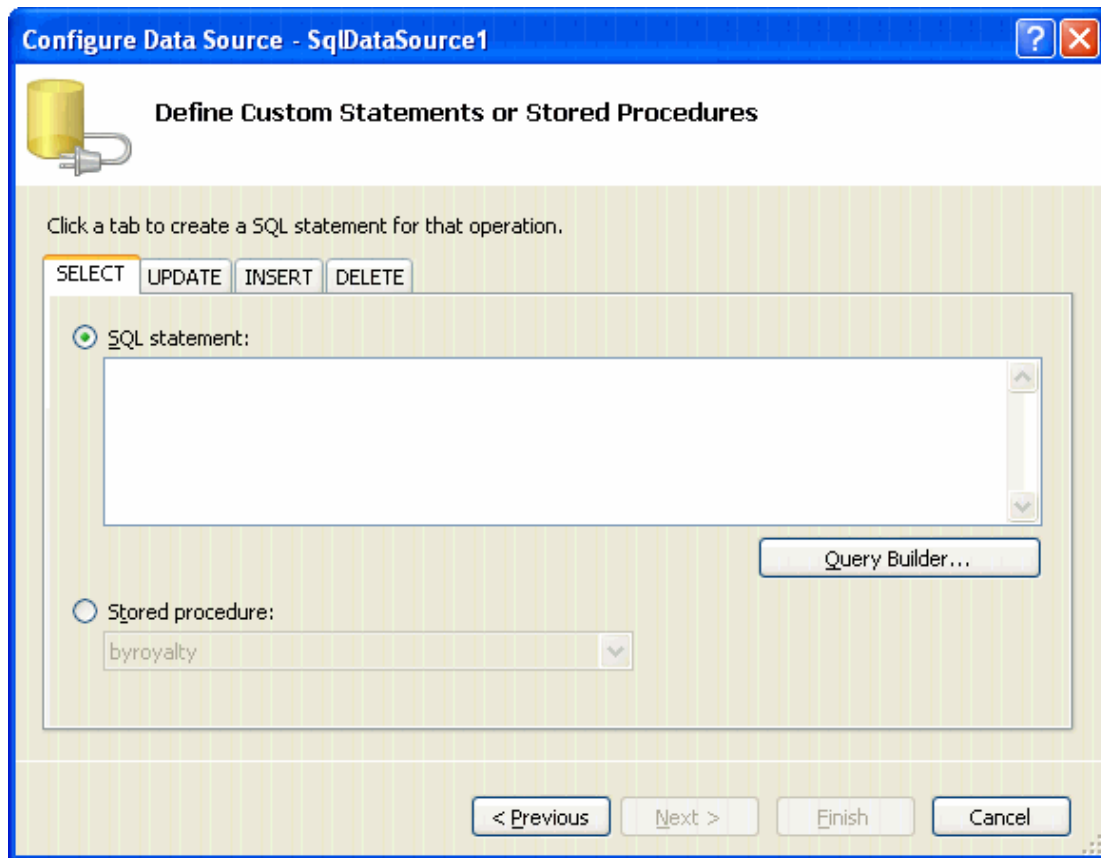
ORDER BY...

Advanced...

SELECT statement:

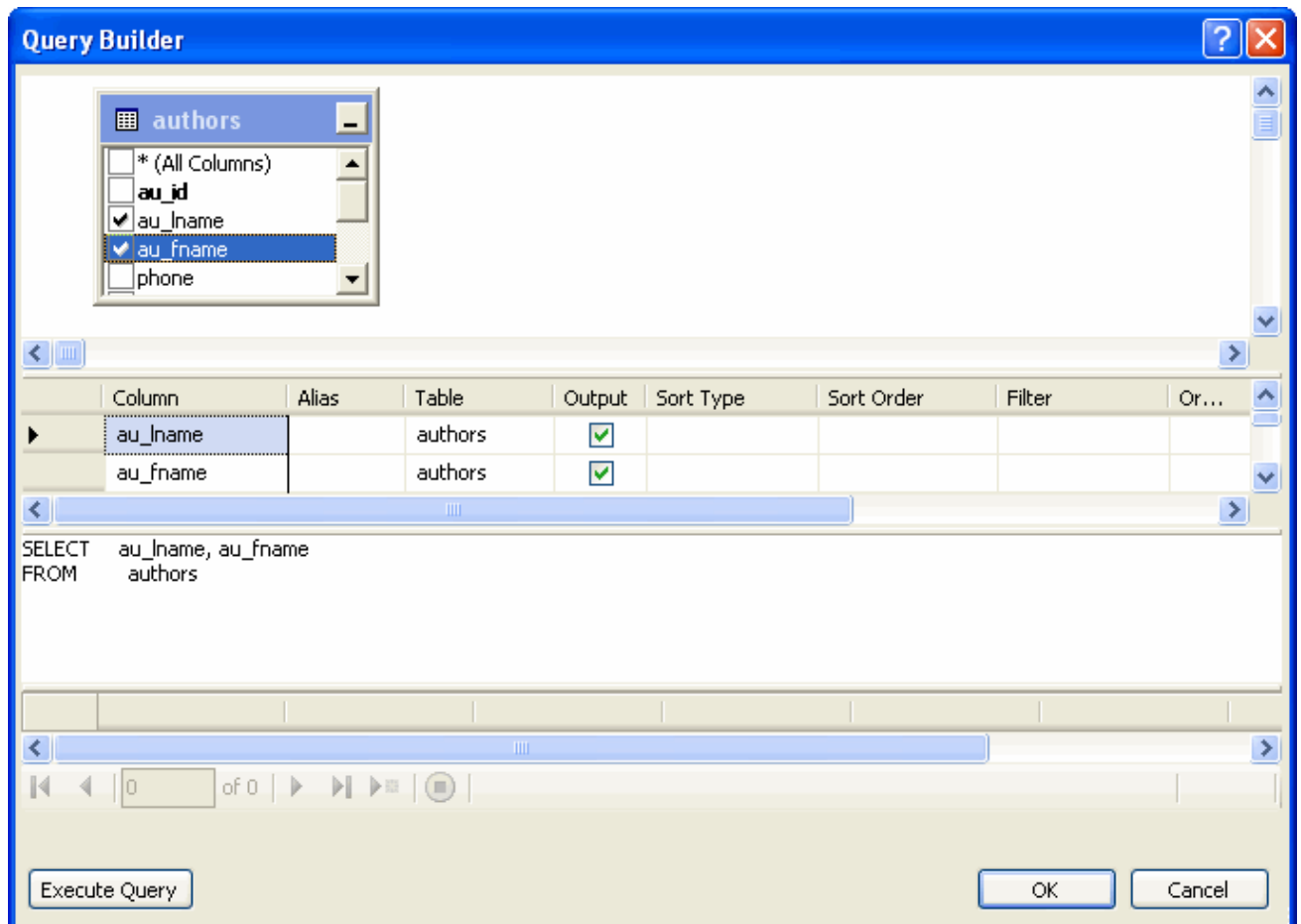
< Previous Next > Finish Cancel

Podemos indicar unas columnas de alguna tabla o la primera opción para especificar una instrucción SQL que es lo que hemos comentado antes, pulsamos siguiente:

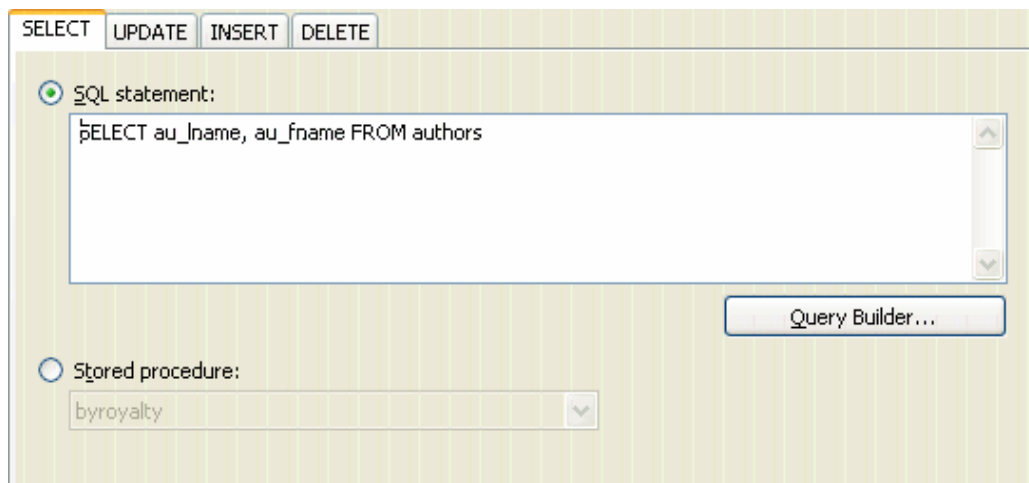


Y como ves nos permite definir los comandos para las cuatro acciones que hemos comentado antes. Todo esto que estamos viendo es una forma gráfica muy sencilla de rellenar los parámetros de este control que luego veremos en el editor.

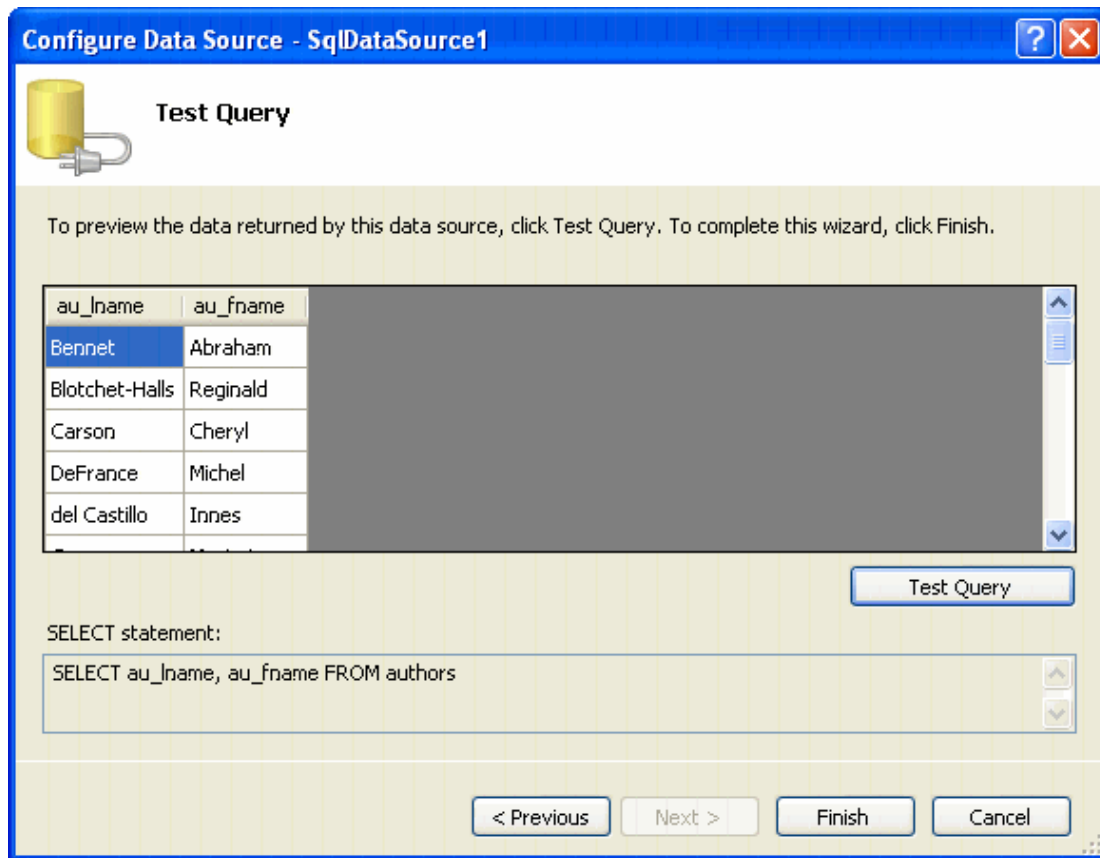
Como ves disponemos además de un generador de consultas que utilizaremos para seleccionar un par de campos de prueba:



Podemos ejecutarla para comprobar que va bien y por fin aceptarla:



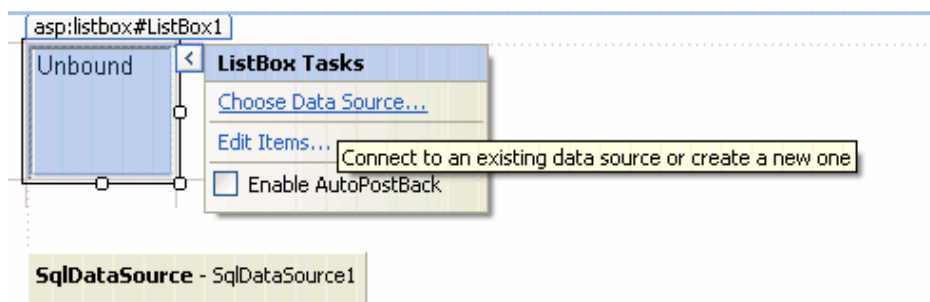
Pulsamos en siguiente:



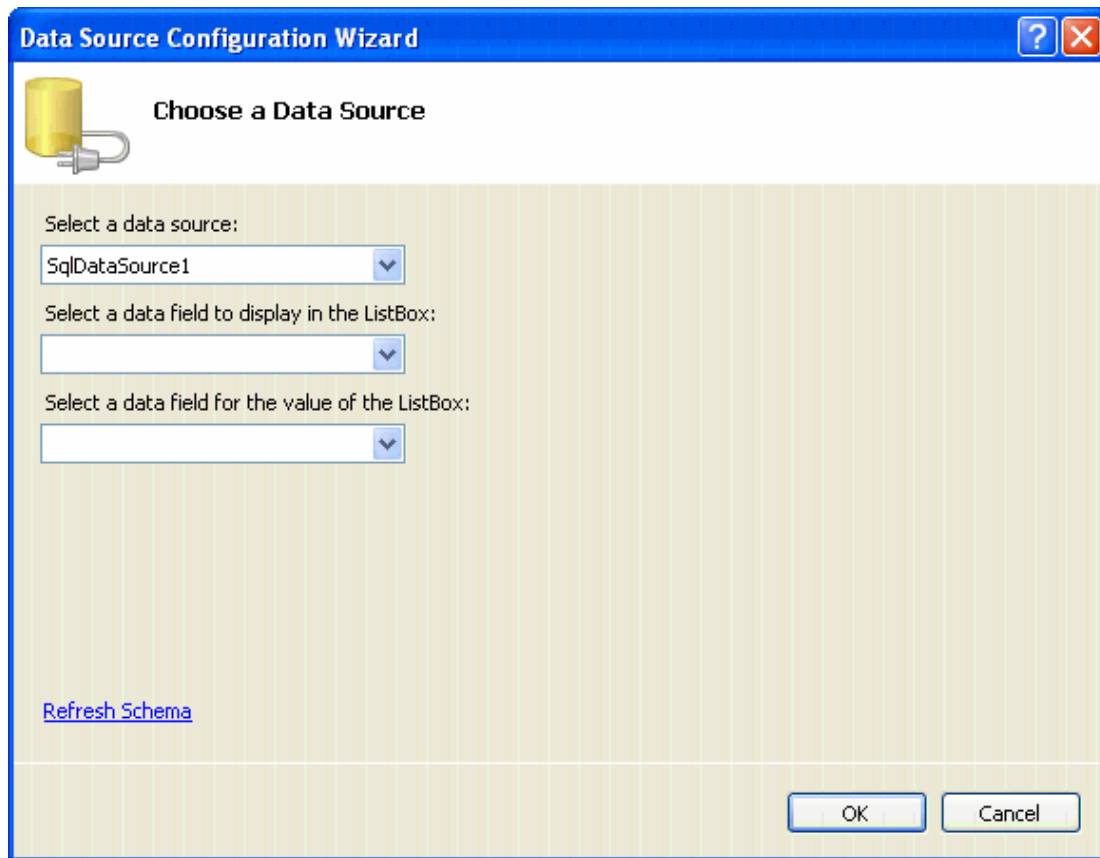
Y podemos hacer una comprobación final del resultado que obtendremos con esa consulta... Finalmente pulsamos "finish" y vemos el código generado:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= ConnectionStrings:Pubs %> %>"
    ProviderName="System.Data.SqlClient"
    SelectCommand="SELECT au_lname, au_fname FROM authors"></asp:SqlDataSource>
```

Ahora que ya tenemos nuestro origen de datos configurado podemos añadir a la página de ejemplo un control de lista para enlazarlo con este origen de datos, sería de la forma. Ponemos un cuadro de lista en la página:

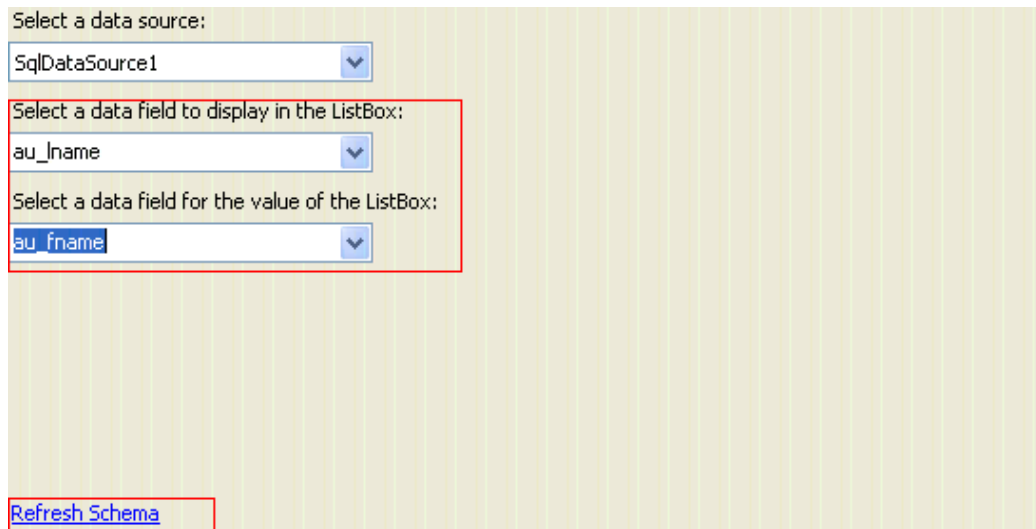


Y seleccionamos ahora "Choose Data Source" para indicarle:



The image shows a Windows-style dialog box titled "Data Source Configuration Wizard". It has a blue title bar with a question mark icon and a close button. The main area has a light yellow background. At the top left, there is a yellow cylinder icon with a plug. The title "Choose a Data Source" is centered. Below it, there are three dropdown menus. The first is labeled "Select a data source:" and contains "SqlDataSource1". The second is labeled "Select a data field to display in the ListBox:" and is empty. The third is labeled "Select a data field for the value of the ListBox:" and is empty. At the bottom left, there is a blue link labeled "Refresh Schema". At the bottom right, there are two buttons: "OK" and "Cancel".

Indicamos el origen de datos que hemos puesto: "SqlDataSource1". Ahora debemos decirle los campos que queremos enlazar, para que aparezcan en las listas pulsaremos en "refresh schema" para que haga la consulta y sepa las columnas que hay y nos permite seleccionarlas:



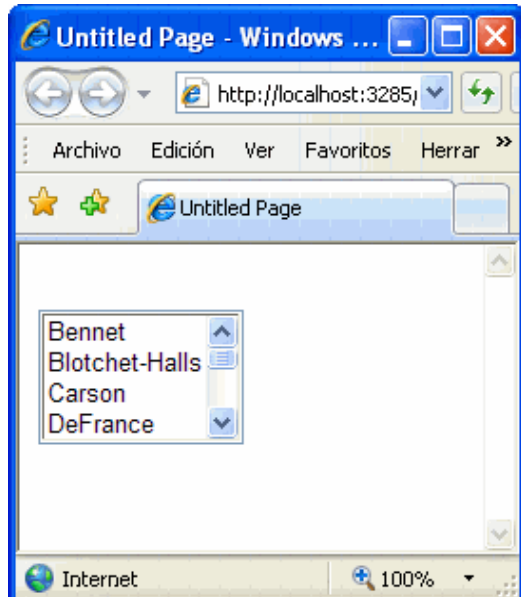
This image is a close-up of the "Data Source Configuration Wizard" dialog box. A red rectangular box highlights the three dropdown menus. The first dropdown, "Select a data source:", shows "SqlDataSource1". The second dropdown, "Select a data field to display in the ListBox:", shows "au_lname". The third dropdown, "Select a data field for the value of the ListBox:", shows "au_fname". Below the dropdowns, the "Refresh Schema" link is also highlighted with a red rectangular box.

Pulsamos en "Ok" para terminar. Ahora fíjate en las propiedades que nos ha generado en la página:

```
<asp:ListBox ID="ListBox1" runat="server" DataSourceID="SqlDataSource1"
    DataTextField="au_lname" DataValueField="au_fname"></asp:ListBox>
```

Enlace a datos. Control de cuadrícula

Como ves le ha indicado el origen de datos y con las propiedades "DataTextField" y "DataValueField" le indicamos el campo que queremos mostrar y el valor que queremos que nos devuelva al seleccionar el elemento. Si ejecutamos la página veremos que nos ha rellenado el cuadro de lista con lo que le hemos ido indicando en estos pasos:



La ventaja de trabajar de esta forma es que no necesitamos escribir código para realizar operaciones sencillas con bases de datos. Hemos hecho una conexión, una consulta y un enlace automático a un control de lista sin tener que escribir código en la página, sólo parámetros de los controles.

4.4 Detalles de la ejecución de los orígenes de datos

Antes dijimos que podíamos enlazar una DataReader o un Dataset... entonces cual debemos utilizar? Este control utiliza por defecto el Dataset por medio de la propiedad DataSourceMode establecida a SqlDataSourceMode.DataSet. La otra opción del DataReader sería con el valor SqlDataSourceMode.DataReader. El Dataset es mas versátil porque permite mas operaciones: ordenación, paginación, filtros, ... sin embargo el DataReader es mas eficaz en consultas extensas porque consume menos recursos.

Mas adelante veremos los controles avanzados que permitirán realizar todo esto de una forma mas sencilla todavía, aunque la base será la misma.

Esto que estamos viendo va a ser de fundamental importancia para cuando trabajemos con controles enlazados a datos, ya que les vamos a proporcionar a los controles las operaciones necesarios.

4.5 Comandos parametrizados

En el ejemplo anterior SQLDataSource recuperaba una serie de elementos de una tabla gracias a que le proporcionábamos en comando que debía ejecutar:

Enlace a datos. Control de cuadrícula

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= $ ConnectionStrings:Pubs %>"
    ProviderName="System.Data.SqlClient"
    SelectCommand="SELECT au_lname, au_fname FROM authors"></asp:SqlDataSource>
```

Los orígenes de datos como ves además de proporcionarnos un enlace a la base de datos pueden contener una serie de comandos para que cuando los soliciten los controles puedan ejecutarlos sin necesidad de añadir código ASP.NET. Por tanto la idea es proporcionar las instrucciones de SQL necesarias para todas las operaciones: select, insert, update y delete.

El ejemplo que hicimos hace un rato, donde actualizábamos a mano el teléfono de una fila de la tabla de autores era bastante bueno, seleccionábamos un registro y editábamos sus datos para así poder modificarlos en la base de datos mediante un comando SQL.

Vamos a ampliar nuestro ejemplo para que nos muestre en un control los datos del autor seleccionado. Primero crearemos un segundo enlace a datos a la misma tabla:

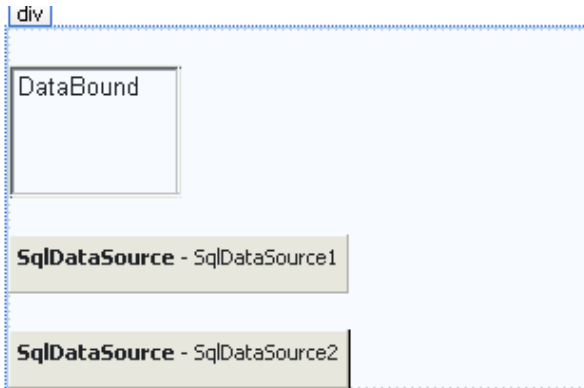
```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%= $ ConnectionStrings:Pubs %>"
    SelectCommand="SELECT * FROM [authors]"></asp:SqlDataSource>
```

Lo puedes crear con el asistente seleccionado como campos "*" en la tabla de autores para que nos extraiga todas las columnas:

Pero tenemos un problema ya que tenemos que proporcionar a este segundo origen de datos el valor del autor seleccionado en el cuadro de lista. Definiremos pues un parámetro "@au_id" que defina el identificador del

Enlace a datos. Control de cuadrícula

autor a recuperar. ¿cómo rellenamos esa información? Para rellenarla proporcionaremos un parámetro con "<selectparameters>" dentro de nuestro "SqlDataSource" así le decimos a este que busque ese dato que lo tendremos dentro de un control. Así que añadimos otro SqlDataSource:



En el cuadro de lista he hecho una pequeña modificación para que al seleccionarlo nos devuelva el campo "au_id" que es el índice del registro:

```
<asp:ListBox ID="lista_autores" runat="server" DataSourceID="SqlDataSource1"
    DataTextField="au_lname" DataValueField="au_id" AutoPostBack="True"></asp:ListBox>
<br />
```

Por supuesto con el "autopostback" activado para que se recargue la página. En el primer origen de datos también he cambiado la consulta SQL para que recupere todos los campos, ya que si dejo el ejemplo anterior solo me traía el nombre y apellidos:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= $ ConnectionStrings:Pubs %>"
    ProviderName="System.Data.SqlClient"
    SelectCommand="SELECT * FROM authors"></asp:SqlDataSource>

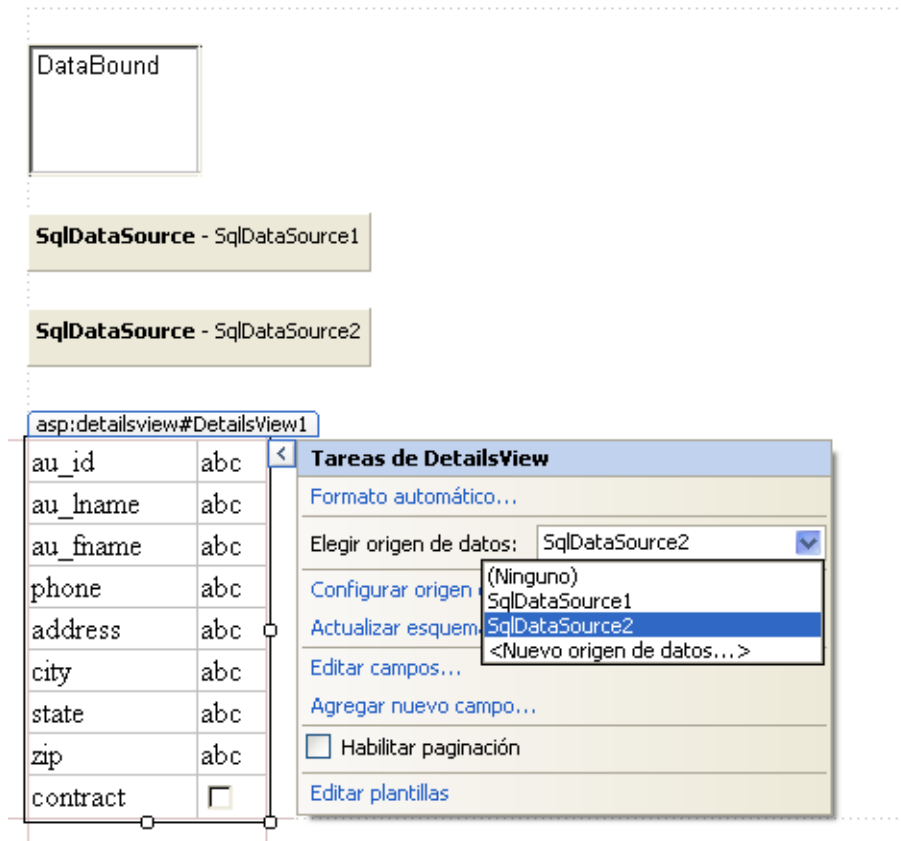
<br />
```

Ahora pondremos un parámetro al segundo Datasource, de momento por código, luego veremos como hacerlo de forma automática:

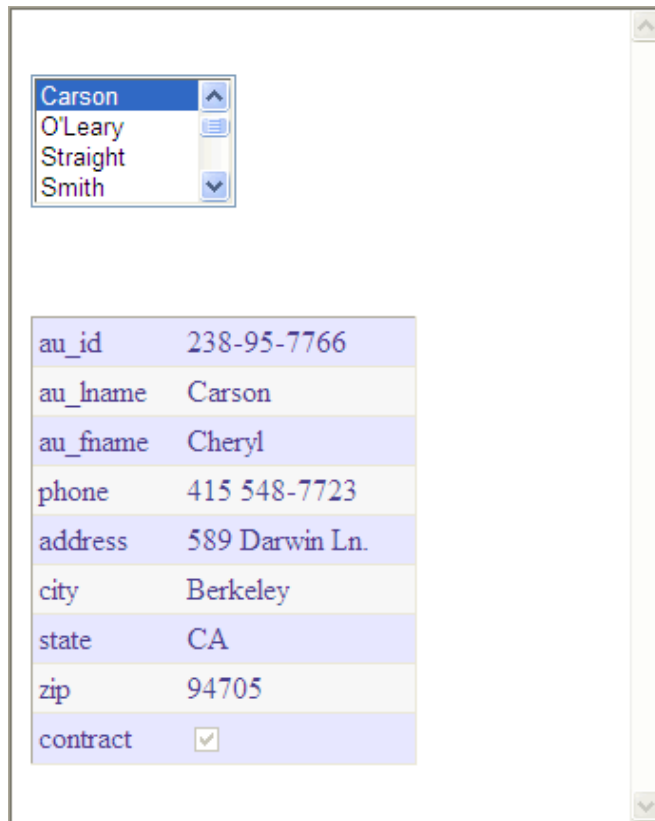
```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%= $ ConnectionStrings:Pubs %>"
    SelectCommand="SELECT * FROM [authors] WHERE ([au_id] = @au_id)"
    <SelectParameters>
        <asp:ControlParameter ControlID="lista_autores" Name="au_id"
            PropertyName="SelectedValue" Type="String" />
    </SelectParameters>
</asp:SqlDataSource>
```

y finalmente incluimos un control de tipo "ViewDetails" a la página y le indicamos el origen de datos es el segundo que hemos insertado:

Enlace a datos. Control de cuadrícula

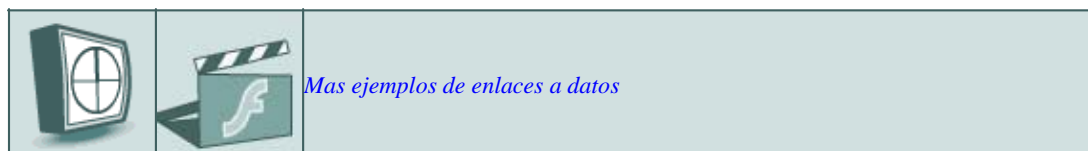


Ejecutamos la página y verás como al seleccionar un elemento de arriba se nos actualiza debajo los datos con un control muy interesante, y todo sin escribir ni una línea de código:



au_id	238-95-7766
au_lname	Carson
au_fname	Cheryl
phone	415 548-7723
address	589 Darwin Ln.
city	Berkeley
state	CA
zip	94705
contract	<input checked="" type="checkbox"/>

Podemos definir tantos parámetros como queramos. Y los enlazaríamos cada uno con su cuadro de texto pero no parece lo mejor porque tendríamos que hacer un montón de consultas para rellenar cada control. La solución es utilizar uno de los controles avanzados para mostrar información, como el que hemos utilizado de "DetailsView". Enseguida veremos la cuadrícula GridView, FormView y alguno mas que nos harán esta tarea trivial.



Mas parámetros

Hemos utilizado un parámetro muy interesante que nos ha permitido encadenar la propiedad de un control con un parámetro para nuestra consulta SQL. Esto nos será muy útil para las consultas y cuando es de este tipo se llama parámetro de control o "control parameter", pero podemos utilizar otro tipo de parámetro como estos:

Origen	Etiqueta	Descripción
Propiedad Control	<asp:ControlParameter>	Propiedad de otro control de la página
Valor de un Querystring	<asp:QueryStringParameter>	Valor de la cadena "querystring"
Valor de estado de sesión	<asp:SessionParameter>	Valor almacenada en la sesión del usuario
Cookie	<asp:CookieParameter>	Valor de una cookie
Valor de un perfil	<asp:ProfileParameter>	Valor del perfil de usuario, que veremos en el último capítulo

Enlace a datos. Control de cuadrícula

Variable de formulario	<asp:FormParamater>	Valor de un control de tipo "input" de un formulario. Lo utilizaremos si hemos deshabilitado la vista de estado
------------------------	---------------------	---

Estate atento a este tipo de parámetros porque son muy interesantes y cuando nivel con ASP verás como te acuerdas de ellos. En el editor gráfico podemos acceder a esta opción desde la pantalla de la consulta cuando configuramos la conexión y le indicamos una selección de registros.

Configurar origen de datos - SqlDataSource1

Configurar la instrucción Select

¿Cómo desea recuperar los datos de la base de datos?

☐ Especificar una instrucción SQL o un procedimiento almacenado personalizado

☒ Especificar columnas de una tabla o vista

Nombre:
Categorías

Columnas:

- ☐ *
- ☐ IdCategoría
- ☐ NombreCategoría
- ☐ Descripción

☐ Devolver sólo filas únicas

WHERE...

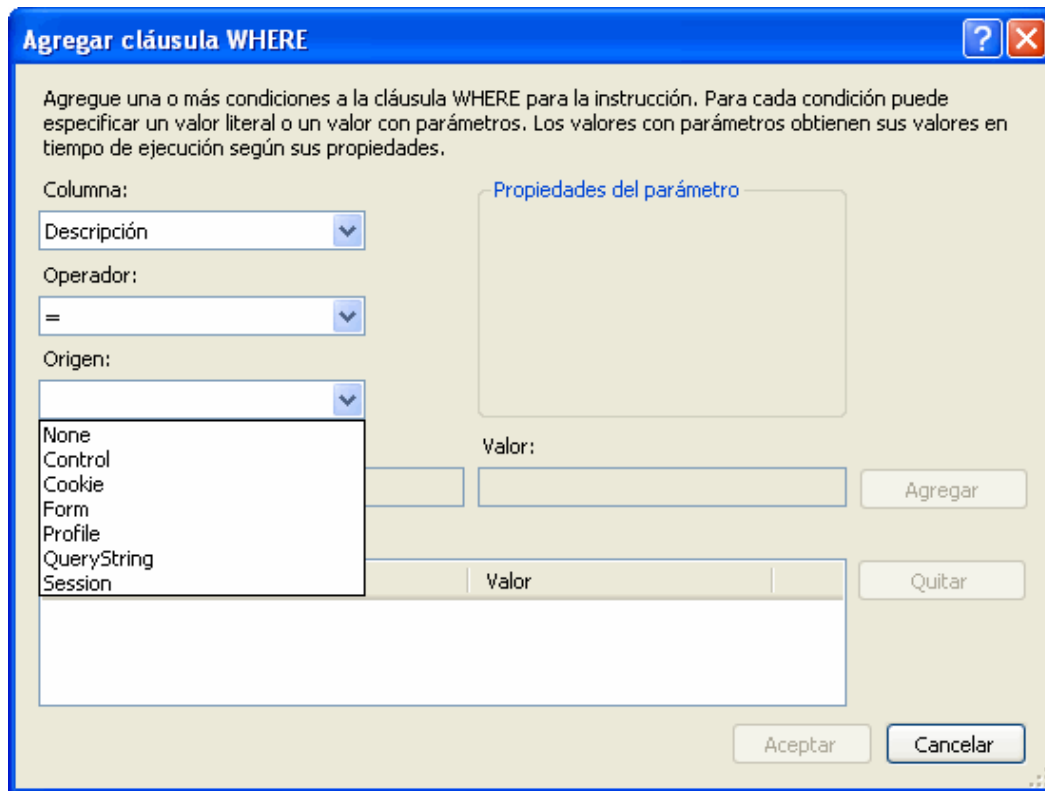
ORDER BY...

Avanzadas...

Instrucción SELECT:

< Anterior Siguiente > Finalizar Cancelar

Y luego pulsando las restricciones "Where":



Mas adelante en este capítulo veremos como hacerlo funcionar pero puedes ver las opciones en el cuadro desplegable que son las que te he enumerado antes.

Establecer valores de parámetros en el código

Antes hemos puesto un parámetro a la sentencia SQL, si recuerdas:

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%= $ ConnectionStrings:Pubs %>"
    SelectCommand="SELECT * FROM [authors] WHERE ([au_id] = @au_id)">
    <SelectParameters>
        <asp:ControlParameter ControlID="lista_autores" Name="au_id"
            PropertyName="SelectedValue" Type="String" />
    </SelectParameters>
</asp:SqlDataSource>
```

Pero ahora necesitamos modificar la sentencia SQL, porque no quiero poner a todos los autores en la lista desplegable, sino solo los de determinada ciudad. Necesitamos añadir un valor a esa sentencia SQL para añadir la restricción así que por un lado pondremos ese nuevo parámetro "@city" y por otro en la parte de abajo donde los "SeletParameters" le pondremos el valor de éste que será por ejemplo una constante:

Enlace a datos. Control de cuadrícula

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%= ConnectionStrings:Puhs %>"
    SelectCommand="SELECT * FROM [authors] WHERE ([au_id] = @au_id) and (city=@city)">
    <SelectParameters>
        <asp:ControlParameter ControlID="lista_autores" Name="au_id"
            PropertyName="SelectedValue" Type="String" />
        <asp:Parameter Name="fechas" DefaultValue="Logroño" />
    </SelectParameters>
</asp:SqlDataSource>
```

Con lo que le estamos proporcionando otro valor.

4.6 Control de errores

Como siempre en la programación nada está exento de errores así que de alguna forma tenemos que prever que pueda fallar algo en la consulta a la base de datos. ¿que fallos se pueden dar en la consulta? Pues muy fácil, que la base de datos no esté en ese momento operativa o que el servidor de base de datos esté sufriendo algún tipo de mantenimiento. Es lo que decimos los informáticos en las averías.

La idea es que controlemos el evento del origen de datos justo después de producirse, sin dar tiempo a que se enlacen los controles y produzcan el error en la pantalla. Veamos aquí como lo hacemos para el método "Selected" pero podríamos hacerlo para todas las demás opciones de insertar, borrar o modificar. O incluso crear un controlador de errores común para todos. Fíjate en el parámetro de entrada de este controlador de eventos:

```
Protected Sub autores_Selected (Byval sender as Object, _
    ByVal e as SqlDataSourceStatusEventArgs) Handles autores.selected

    if e.exception isnot Nothing then

        label_error.Text="Ha habido un problema en la consulta de la tabla de autores..."

        e.exceptionHandled=True

    end if

End Sub
```

Hemos accedido a la excepción a través de la propiedad "SqlDataSourceStatusEventArgs.Exception". Si queremos prevenir errores en el futuro simplemente pondremos la propiedad "SqlDataSourceStatusEventArgs.ExceptionHandled" a "true", como hemos hecho en el código.

4.7 Actualizar registros

La consulta de los datos en nuestras bases de datos como acabamos de ver es una parte de los procesos que podemos hacer con ellas. Ahora trataremos la actualización de registros que requiere de un poco mas de lógica. Muy sencillo, hay muchos controles que son solo para mostrar datos como las lista o las listas desplegables y no nos permiten la edición. Para esto tenemos unos controles de acceso a datos realmente potentes y que veremos mas adelante. Antes de poder hacer estas operaciones de modificación de los datos tendremos que crear nuestros comandos SQL para realizarlos.

Enlace a datos. Control de cuadrícula

Te recuerdo que todo esto que estamos viendo es para poder realizar las operaciones habituales de las bases de datos sin escribir código. Ya que le estamos diciendo al origen de datos las sentencias que debe ejecutar según las acciones y además con parámetros: consulta los registros que contengan el dato seleccionado en un control de cuadro de lista. Podríamos hacer todo a mano como vimos en el capítulo anterior pero normalmente no hará falta y combinaremos esta teoría con los controles avanzados de datos que veremos enseguida. Pero para comprender como funcionan tenemos que ver todos estos detalles, luego verás la sencillez al poner todo junto.

Para poder realizar estas operaciones debemos pues definir los comandos necesarios mediante: "InsertCommand" para insertar, "DeleteCommand" para borrar y "UpdateCommand" para actualizar datos. Por supuesto pondremos los comandos que necesitemos porque es posible que en algunas ocasiones no queremos mas que actualizar datos. La definición de los comandos será igual que para la instrucción "Select" de consulta de antes, por ejemplo:

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%$ ConnectionStrings:Puhs %>"
    SelectCommand="SELECT * FROM [authors] WHERE ([au_id] = @au_id) and (city=@city)">
    <SelectParameters>
        <asp:ControlParameter ControlID="lista_autores" Name="au_id"
            PropertyName="SelectedValue" Type="String" />
        <asp:Parameter Name="fechas" DefaultValue="Logroño" />
    </SelectParameters>
</asp:SqlDataSource>
```

Es un ejemplo de como poner todos los comandos que vamos a querer que nos ejecute el control. Ahora nos toca la parte del control que actualizará los datos. Como hemos dicho no todos permiten la actualización. Veamos un ejemplo con uno que se llama "DetailsView" y que veremos enseguida. Este control tiene opciones que permiten editar y actualizar los datos y se lo indicaremos de esta forma en la definición:

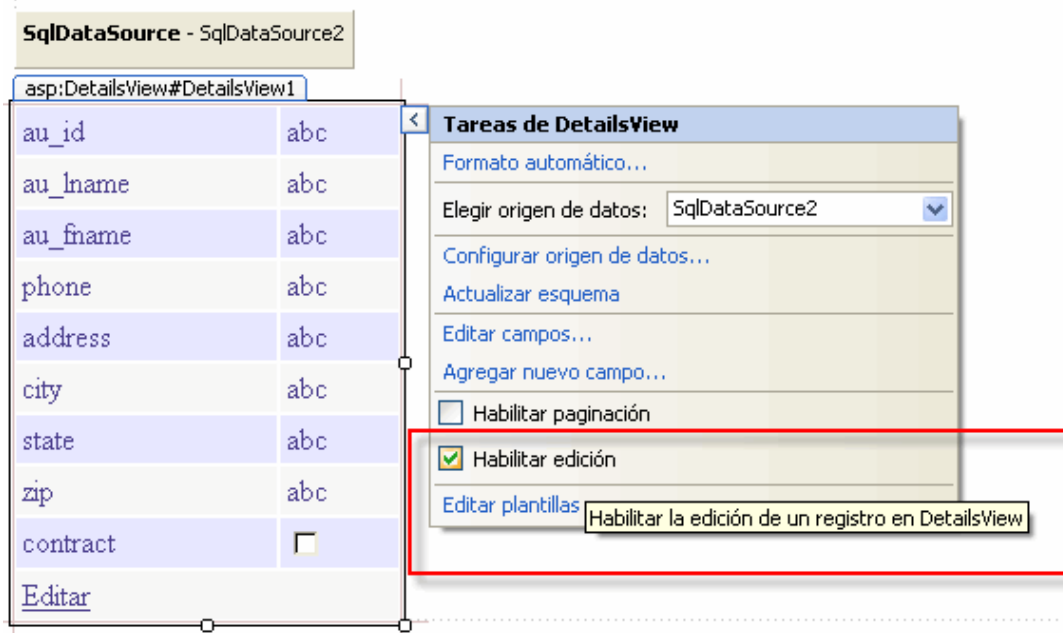
```
<asp:DetailsView id="DetailsView1" runat="server" DataSourceID="autores"

    AutoGenerationEditButton="true" AutoGenerateDeleteButton="true" />
```

Recuerda, estos comandos solo los podrán ejecutar los controles especiales que permiten la edición de datos y que veremos enseguida. Al poner las dos últimas propiedades que ves en esa declaración el control permitirá la edición de los datos y según el botón que se pulse: "Actualizar" o "Borrar" ejecutará la instrucción que le hemos indicado antes. Extrae los datos del control y los hace encajar con los parámetros de la sentencia SQL, construyendo el comando que ejecutará, todo ello sin escribir ni una línea de código ASP.NET en la página de código detrás ".vb". Por supuesto en este ejemplo podríamos poner también el comando de insertar "InsertCommand" y en el control que aparezca el enlace para esto poniendo "AutoGenerateInsertButton". Aunque no te preocupes que con el IDE haremos esto con dos clics, la idea es que comprendas que se pueden asociar los comandos que queremos con el control de origen de datos que luego si tenemos un control avanzado asociado a esta conexión podrá ejecutarlos de forma automática. Consiguiendo de una forma sencilla la edición de los datos.

De todas formas verás que fácil es a partir de ahora porque en el ISE podremos indicar todas estas propiedades, fíjate que complicado sería decirle que incluya lo necesario para la actualización:

Enlace a datos. Control de cuadrícula



Que al ejecutar la página nos permitirá realizar las acciones si hemos puesto las sentencias SQL adecuadas:



au_id	238-95-7766
au_lname	Carson
au_fname	Cheryl
phone	415 548-7723
address	589 Darwin Ln.
city	Berkeley
state	CA
zip	94705
contract	<input checked="" type="checkbox"/>
Actualizar Cancelar	

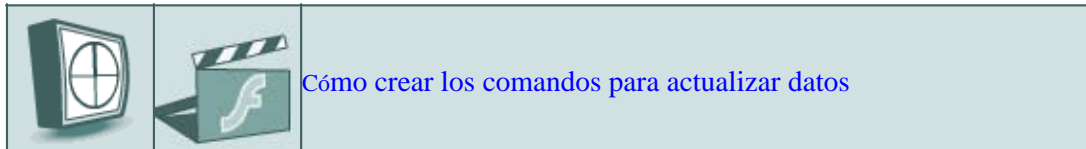
Un detalle mas...

Enlace a datos. Control de cuadrícula

En el capítulo anterior vimos que cuando se inserta, borra o actualiza una fila de una tabla nos devolvía un valor diciendo el número de filas afectadas en la ejecución de la sentencia SQL. De esta forma sabíamos que se había ejecutado correctamente. Aquí no disponemos de código ASP.NET para hacer esto pero si podemos consultarlo en un evento para saber si la sentencia ha afectado a alguna línea, que es el síntoma de que todo ha ido bien. Bien, pues SqlDataSource nos tiene una excepción para indicarnos que no se ha realizado la instrucción así que lo que haremos será ver ese valor devuelto que os acabo de comentar. Si es 0 mal y si es 1 o mas de uno es que ha ido bien y nos lo indicará la propiedad "AffectedRows":

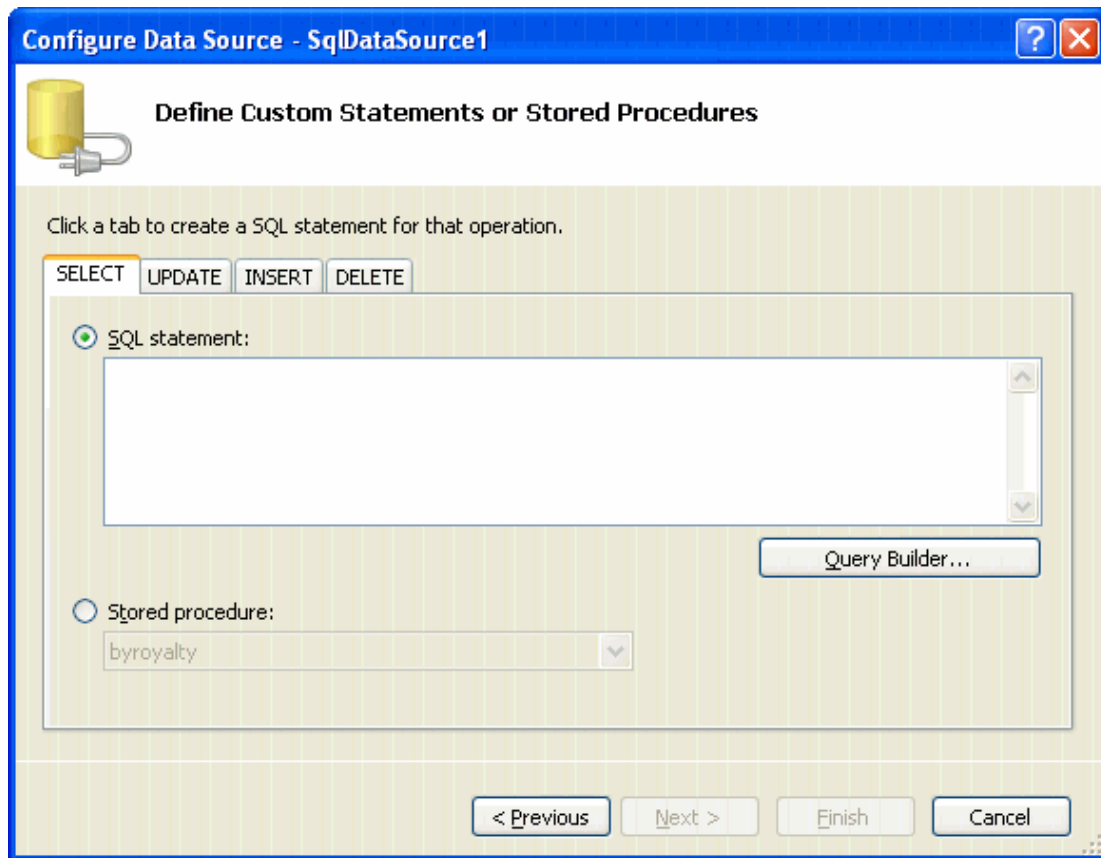
```
Private Sub autores_Updated (Byval Sender as Object, Byval e as SqlDataSourceStatusEventArgs) Handles autores.updated
    if e.AffectedRows=0 Then
        lb_info.Text="No se han realizado los cambios. Se ha producido un error de_
        concurrencia o el comando no está escrito correctamente.
    else
        lb_info.Text="Datos actualizados correctamente"
    end if
End Sub
```

Y vista toda esta teoría, vamos con los controles avanzados...



4.8 Asistente de SqlDataSource para generar los comandos

El control SqlDataSource tiene una opción en las que nos genera de forma automática los comandos. Habitualmente los pondremos a mano en la vista HTML del control o en la pantalla que ya conocemos de:



Que es la que nos pondrá las sentencias SQL ya conocidas y que además si recuerdas nos permite poner los parámetros que hemos visto antes. Pero tenemos una pequeña ayuda y es la siguiente. Si en la primera pantalla de este asistente marcamos que queremos recuperar todas las columnas de la tabla indicando "*" que significa todas las columnas:

Configurar origen de datos - SqlDataSource1

Configurar la instrucción Select

¿Cómo desea recuperar los datos de la base de datos?

☐ Especificar una instrucción SQL o un procedimiento almacenado personalizado

☒ Especificar columnas de una tabla o vista

Nombre:

Categorías

Columnas:

☒ *

☐ IdCategoría

☐ NombreCategoría

☐ Descripción

☐ Devolver sólo filas únicas

WHERE...

ORDER BY...

Avanzadas...

Instrucción SELECT:

SELECT * FROM [Categorías]

< Anterior Siguiete > Finalizar Cancelar

Podremos pulsar la opción "Avanzadas":

Opciones de generación SQL avanzadas

Se pueden generar instrucciones adicionales Insert, Update y Delete para actualizar el origen de datos.

☒ **Generar instrucciones Insert, Update y Delete**

Genera instrucciones INSERT, UPDATE, y DELETE basadas en la instrucción SELECT. Para que esta opción esté habilitada, debe tener todos los campos de la clave principal seleccionados.

☐ **Usar concurrencia optimista**

Modifica instrucciones UPDATE y DELETE para detectar si la base de datos ha cambiado desde que se cargó el registro en el DataSet. Esto ayuda a prevenir conflictos de concurrencia.

Aceptar Cancelar

Que excepcionalmente nos generará los comandos automáticamente. Esto solo funciona cuando seleccionamos todas las columnas así que tiene una utilidad limitada pero el resultado desde luego es que nos tienen preparado el control de origen de datos para asociarlo con un control que permita todas estas operaciones y que veremos ahora mismo...

Enlace a datos. Control de cuadrícula

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= ConnectionStrings:Pub >%"
    DeleteCommand="DELETE FROM [authors] WHERE [au_id] = @au_id"
    InsertCommand="INSERT INTO [authors] ([au_id], [au_lname], [au_fname], [phone], [address], [city],
    SelectCommand="SELECT * FROM [authors]"
    UpdateCommand="UPDATE [authors] SET [au_lname] = @au_lname, [au_fname] = @au_fname, [phone] = @pho:
    <DeleteParameters>
        <asp:Parameter Name="au_id" Type="String" />
    </DeleteParameters>
    <UpdateParameters>
        <asp:Parameter Name="au_lname" Type="String" />
        <asp:Parameter Name="au_fname" Type="String" />
        <asp:Parameter Name="phone" Type="String" />
        <asp:Parameter Name="address" Type="String" />
        <asp:Parameter Name="city" Type="String" />
        <asp:Parameter Name="state" Type="String" />
        <asp:Parameter Name="zip" Type="String" />
        <asp:Parameter Name="contract" Type="Boolean" />
        <asp:Parameter Name="au_id" Type="String" />
    </UpdateParameters>
    <InsertParameters>
        <asp:Parameter Name="au_id" Type="String" />
        <asp:Parameter Name="au_lname" Type="String" />
        <asp:Parameter Name="au_fname" Type="String" />
        <asp:Parameter Name="phone" Type="String" />
        <asp:Parameter Name="address" Type="String" />
        <asp:Parameter Name="city" Type="String" />
        <asp:Parameter Name="state" Type="String" />
        <asp:Parameter Name="zip" Type="String" />
        <asp:Parameter Name="contract" Type="Boolean" />
    </InsertParameters>
</asp:SqlDataSource>
```

¡Que maravilla! Nos ha puesto todos los comandos y todos los valores de los campos... desde luego que es una gran ayuda para nuestros comienzos. Por ejemplo podremos eliminar las líneas de los datos que no queremos modificar, por ejemplo el índice "au_id" y funcionará a la perfección adosándolo a un control de vista de detalles como el que pusimos antes.

5. Controles de datos

Una vez que ya tenemos nuestros datos vamos a ver cómo enlazarlos con los controles adecuados. ASP.NET no ha diseñado los controles enlazados a datos de forma general, sino que les ha dado determinadas funcionalidades dependiendo del uso que se les va a dar.

En este capítulo daremos un gran paso en los controles enlazados porque veremos los mas potentes: GridView, DetailsView y FormView. Que nos van a permitir ver tablas de datos enteras. Estos controles de datos enriquecidos tienen alguna diferencia respecto a los controles sencillos anteriores ya que tienen mas posibilidades al mostrarnos mas de un campo a la vez, habitualmente con el aspecto de una tabla. Y, lo mejor de todo, nos van a permitir editar registros directamente, ordenar, paginas resultados. Estos controles son:

- GridView. Es una cuadrícula muy versátil para mostrar grandes tablas, este es sin duda el mejor y mas completo control de ASP.NET
- DetailsView. Muestra un registro a la vez en una tabla. También puede editar los datos para modificarse.
- FormView. AL igual que el anterior, permite visualiza y modificar un registro. La diferencia es que este está basado en plantillas que nos van a permitir combinar campos en un entorno mas flexible.
- ListView. Parecido al " gridview " permite mostrar varias filas la vez. Paso lo mismo que en la caso anterior, la diferencia está en que este control permite plantillas. Este control no lo veremos par a poder centrarnos en los otros que son los

importantes. Éste se compone de una combinación de los otros así que es fácil utilizarlo.

6. GridView

Este control es una versátil cuadrícula que nos va a permitir mostrar en forma de tabla datos enlazados a tablas de bases de datos. Cada columna de nuestra tabla de la base de datos tendrá su correspondencia con las columnas en la cuadrícula. Es sin duda, el mas potente y versátil incluyendo métodos para seleccionar, editar, paginar y editar filas y además el único de momento que nos mostrará mas de una fila en pantalla.

6.1 Generación automática de columnas

Como ya habrás imaginado,. Este control dispone de la propiedad `DataSource` para enlazarse con el origen de los datos que queremos mostrar y que enlazaremos con el método ya conocido de `DataBind` . Sin embargo no dispone de propiedades como `DataTextField` o `DataValueField` para poder elegir las columnas que queremos mostrar. Esto es porque el `GridView` genera de forma automática una columna para cada campo cuando la propiedad `AutoGenerateColumns` está a `true` que es la opción predeterminada.

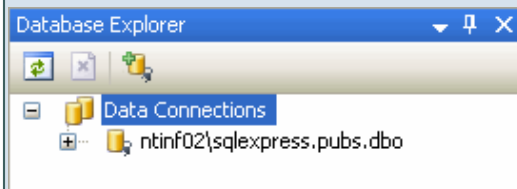
Para crear una cuadrícula sencilla, con una columna por campo solo necesitaremos:

```
<asp:GridView ID="GridView1" runat="server" />
```

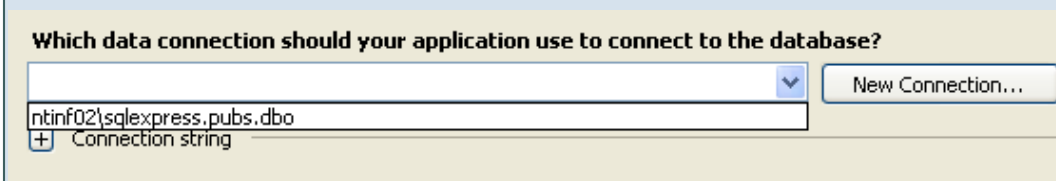
Una vez puesta en la página solo debemos rellenar los datos como ya hemos hecho en otras ocasiones. Asegúrate de que tienes una cadena con el origen de datos definida en el web.config:

```
<connectionStrings>
  <add name="Conexion Pubs" connectionString="Data Source=localhost\\sqlexpress;
    Initial Catalog=pubs;Integrated Security=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Nota Te recuerdo que si quieres crear la cadena de forma automática en ese fichero, haz lo siguiente. Añade primero en el administrador de conexiones (solapa junto al explorador de soluciones o proyectos), una conexión al servidor SQL Server que tu quieras, por ejemplo el SQL Server Express Local:




Luego en una página en blanco añade un control de tipo `SqlDataSource` y pulsa en su opción de "Configurar Data Source". Si pulsas en el desplegable te aparecerá el servidor de bbdd que has añadido antes:



Y al pulsar en siguiente nos dice:

Configure Data Source - SqlDataSource1



Save the Connection String to the Application Configuration File

Storing connection strings in the application configuration file simplifies maintenance and deployment. To save the connection string in application configuration file, enter a name in the text box and then click Next. If you choose not to do this, the connection string is saved in the page as a property of the data source control.

Do you want to save the connection in the application configuration file?

☒ Yes, save this connection as:

Conexion_Pubs

< Previous

Next >

Finish

Cancel

Que es que si queremos grabar la configuración en el web.config. Le decimos que Si lo grabe y perfecto, hemos añadido la configuración en ese servidor de forma automática.

Sigamos con nuestro ejemplo. Queríamos poner un GridView en una pagina nueva:

tama_12/tablas_1.aspx*

Start Page

div

Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

Y en el evento Load haremos esta consulta mediante un Dataset que enlazamos a la cuadrícula:

Enlace a datos. Control de cuadrícula

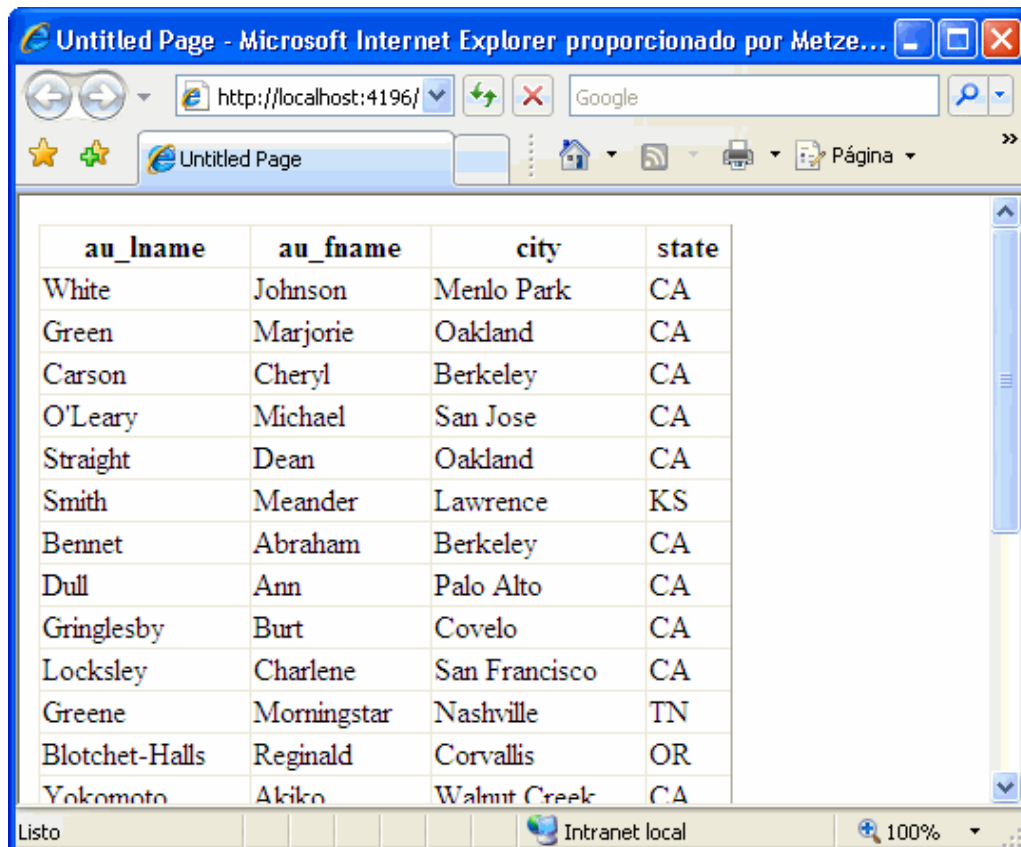
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs) Handles Me.Load
    ' Definimos los objetos ADO.NET...
    Dim cadena_conexion As String = WebConfigurationManager.ConnectionStrings("Conexion_Pubs").ConnectionString
    Dim sql As String = "SELECT au_lname, au_fname, city, state FROM authors"
    Dim conexion As New SqlConnection(cadena_conexion)
    Dim comando As New SqlCommand(sql, conexion)
    Dim adapter As New SqlDataAdapter(comando)

    ' Rellenamos el Dataset
    Dim ds As New DataSet()
    adapter.Fill(ds, "Productos")
    ' Enlazamos
    GridView1.DataSource = ds
    GridView1.DataBind()
End Sub
```

Recuerda los espacios de nombres que hemos añadido arriba para poder acceder a los objetos de "SQL..", del Dataset y de la fichero de configuración web.config:

```
Imports System.Web.Configuration
Imports System.Data.SqlClient
Imports System.Data
```

Ya ves que es realmente poco el código necesario, esto ya es cortar y pegar de los ejemplos anteriores, así que te debería ser ya muy familiar. El resultado será este:



au_lname	au_fname	city	state
White	Johnson	Menlo Park	CA
Green	Marjorie	Oakland	CA
Carson	Cheryl	Berkeley	CA
O'Leary	Michael	San Jose	CA
Straight	Dean	Oakland	CA
Smith	Meander	Lawrence	KS
Bennet	Abraham	Berkeley	CA
Dull	Ann	Palo Alto	CA
Gringlesby	Burt	Covelo	CA
Locksley	Charlene	San Francisco	CA
Greene	Morningstar	Nashville	TN
Blotchet-Halls	Reginald	Corvallis	OR
Yokamoto	Akiko	Walnut Creek	CA

Por supuesto no tendremos que escribir este código. Si recuerdas de lo aprendido en el capítulo anterior utilizaremos el control SqlDataSource para definir nuestra consulta para luego enlazar la consulta directamente con el control.

Enlace a datos. Control de cuadrícula

Aquí es como definiríamos un SqlDataSource realice la consulta:

```
<asp:SqlDataSource ID="sourceProducts" runat="server" ConnectionString="<%$ ConnectionStri  
SelectCommand="SELECT au_lname,au_fname,city,state FROM authors" />
```

Después establecemos la propiedad GridView.DataSourceID para enlazar el origen de datos con el control:

```
<asp:GridView ID="GridView1" runat="server" DataSourceID="sourceProducts" />
```

Estas dos últimas instrucciones son el doble que las vista al principio en el capítulo pero no tendremos que escribir ningún código para ejecutar la consulta y enlazar el Dataset. Utilizando SqlDataSource saldremos ganando ya que no tendremos que escribir nada sobre bases de datos en nuestra página de código ASP.NET. Veamos un ejemplo, crea una nueva página le pones un GridView y un objeto de tipo SQIDataSorce:

Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

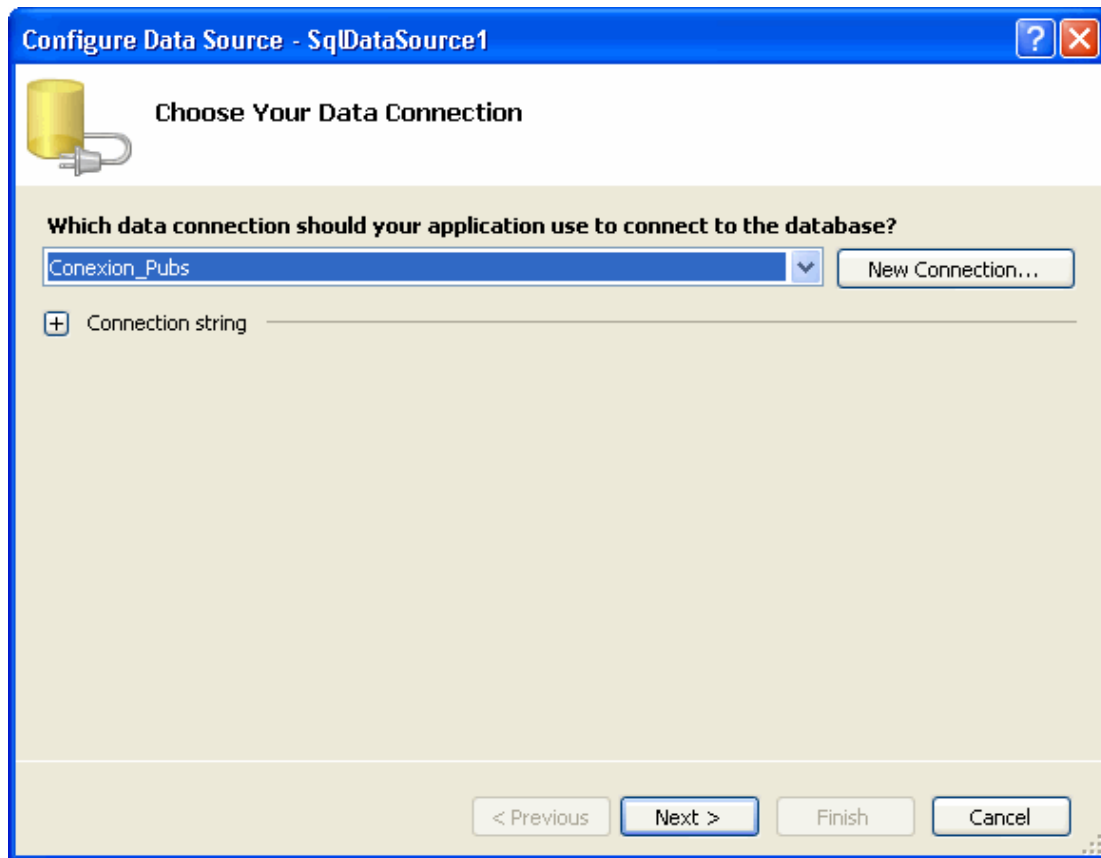
asp:sqldatasource#SqlDataSource1

SqlDataSource - SqlDataSource1

SqlDataSource Tasks


Configure Data Source...

Configuramos el origen de datos con la cadena de conexión que ya tenemos:



Pulsamos en siguiente:

Configure Data Source - SqlDataSource1

 **Configure the Select Statement**

How would you like to retrieve data from your database?

☐ Specify a custom SQL statement or stored procedure

☒ Specify columns from a table or view

Name:

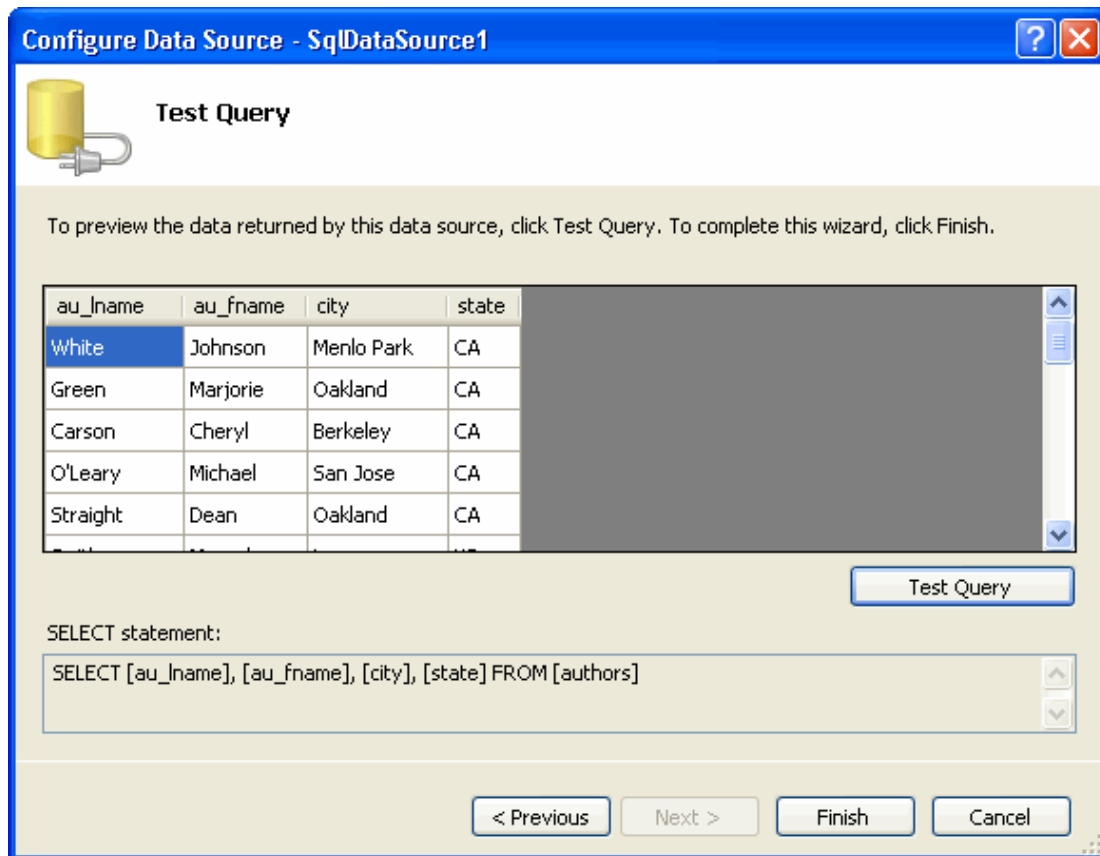
Columns:

<input type="checkbox"/> *	<input checked="" type="checkbox"/> city
<input type="checkbox"/> au_id	<input checked="" type="checkbox"/> state
<input checked="" type="checkbox"/> au_lname	<input type="checkbox"/> zip
<input checked="" type="checkbox"/> au_fname	<input type="checkbox"/> contract
<input type="checkbox"/> phone	
<input type="checkbox"/> address	

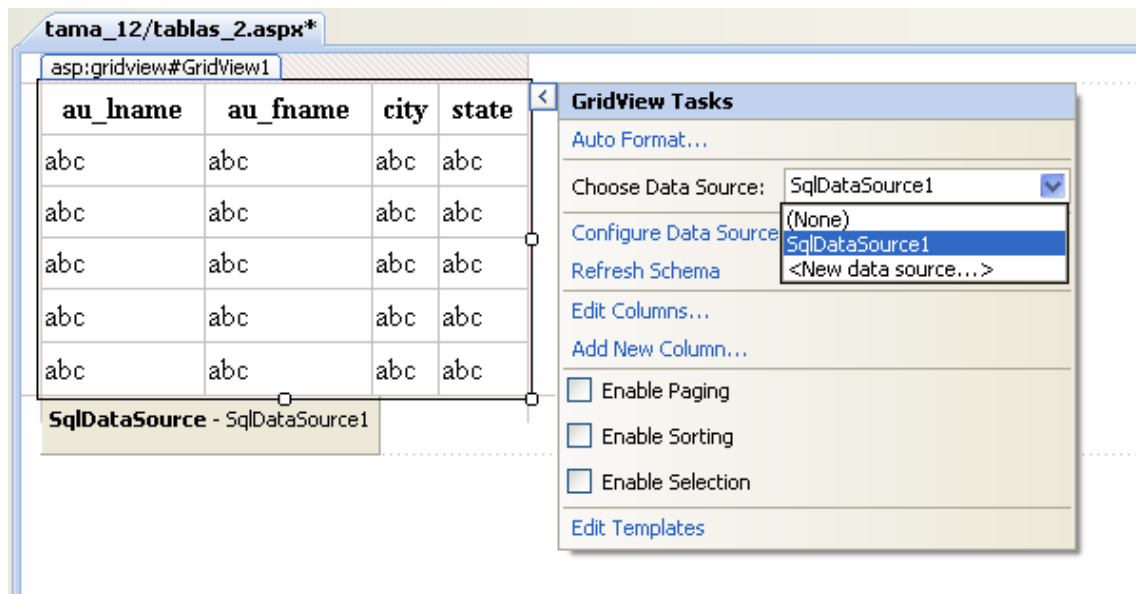
☐ Return only unique rows

SELECT statement:

Y seleccionamos los campos que queremos mostrar como columnas. Seguimos...



Para comprobar que todo funciona correctamente en la vista previa de la consulta. Por fin vamos a la cuadrícula y:



Le asignamos el origen de datos que acabamos de crear. Prueba la página y verás el mismo resultado que antes pero sin escribir ni una sola línea de código.

Nota Este es el momento donde culminan muchas cosas de las que hemos aprendido. Ya sabemos como funciona ASP.NET para realizar una consulta con todos sus pasos de conexión, consulta y enlace. Esto lo hemos vuelto a hacer con un control `SQLDataSource` que nos ha pedido prácticamente lo mismo: una cadena de conexión y luego que datos consultar. Finalmente hemos dicho a la cuadrícula que utilice ese origen de datos que es el enlace de los dos objetos.

6.2 Definir columnas

Por defecto la propiedad `AutoGenerateColumns` de la cuadrícula es `"true"`, creando una columna para cada campo de la `"datatable"` enlazada. Esto es bueno para cuando queremos crear páginas rápidamente, ya que con dos clics podemos mostrar los resultados al usuario pero no nos proporciona la flexibilidad que necesitamos, ya que en muchas ocasiones queremos seleccionar columnas o simplemente no mostrar las de los identificadores, que son valores sin información útil para los usuarios. O cambiar el orden o formato de algunas de ellas.

Nota. Es posible tener la propiedad `AutoGenerateColumns` a `"true"` y definir columnas en la sección `<Columns>`. En este caso las columnas que especifiquemos se añaden continuación de las autogeneradas. Pero para mas claridad seguramente será mas fácil especificar explícitamente las que queremos utilizar.

En el ejemplo anterior ya vimos como se enlazaron y generaron todas las columnas. En esta nueva situación cada columna puede ser cualquiera de los tipos descritos en la siguiente tabla, el orden de las etiquetas determina el orden de las columnas desde la izquierda hasta la derecha.

Clase	Descripción
<code>BoundField</code>	Muestra texto desde un campo del origen de datos
<code>ButtonField</code>	Muestra un botón en la columna
<code>CheckBoxField</code>	Muestra una casilla de verificación en la columna. Se utiliza para valores <code>"true/false"</code> o para el tipo de datos <code>"bit"</code> de SQL Server
<code>CommandField</code>	Proporciona botones para editar o seleccionar
<code>HyperLinkField</code>	Muestra el contenido como un hipervínculo (un texto fijo o un campo de la base de datos)
<code>ImageField</code>	Muestra una imagen de un campo binario
<code>TemplateField</code>	Permite especificar varios campos, controles personalizados y código HTML personalizado. Es el mas versátil pero necesita mas trabajo para adecuarlo a nuestro objetivo

El tipo mas básico es el `"BoundField"` que enlaza un campo con el objeto. Por ejemplo esta sería la definición para un enlace de un solo campo (a una columna) para mostrar el campo `"au_fname"`:

```
<asp:BoundField DataField="au_fname" HeaderText="Nombre" />
```

Esta etiqueta también muestra como podemos cambiar el texto del encabezamiento en el título, cambiando `"au_fname"` por `"Nombre"`. La cuadrícula especificando las columnas que queramos será:

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    DataSourceID="SqlDataSource1" Height="165px" Width="257px">
    <Columns>
        <asp:BoundField DataField="au_lname" HeaderText="Nombre" />
        <asp:BoundField DataField="au_fname" HeaderText="Apellidos" />
        <asp:BoundField DataField="city" HeaderText="Ciudad" />
        <asp:BoundField DataField="state" HeaderText="Estado" />
    </Columns>
</asp:GridView>
```

Enlace a datos. Control de cuadrícula

Esta declaración explícita de columnas tiene varias ventajas:

- Podemos ajustar el orden de las columnas, el encabezado y otros detalles individualmente
- Podemos ocultar columnas, obviamente no mostrándolas al no poder su etiqueta
- Podemos ver las columnas en modo de diseño, cuando lo hacemos en automático nos muestra un " gris genérico, ahora nos muestra las columnas seleccionadas
- Podemos añadir nuevas columnas mezclando varias de ellas, editándolas,

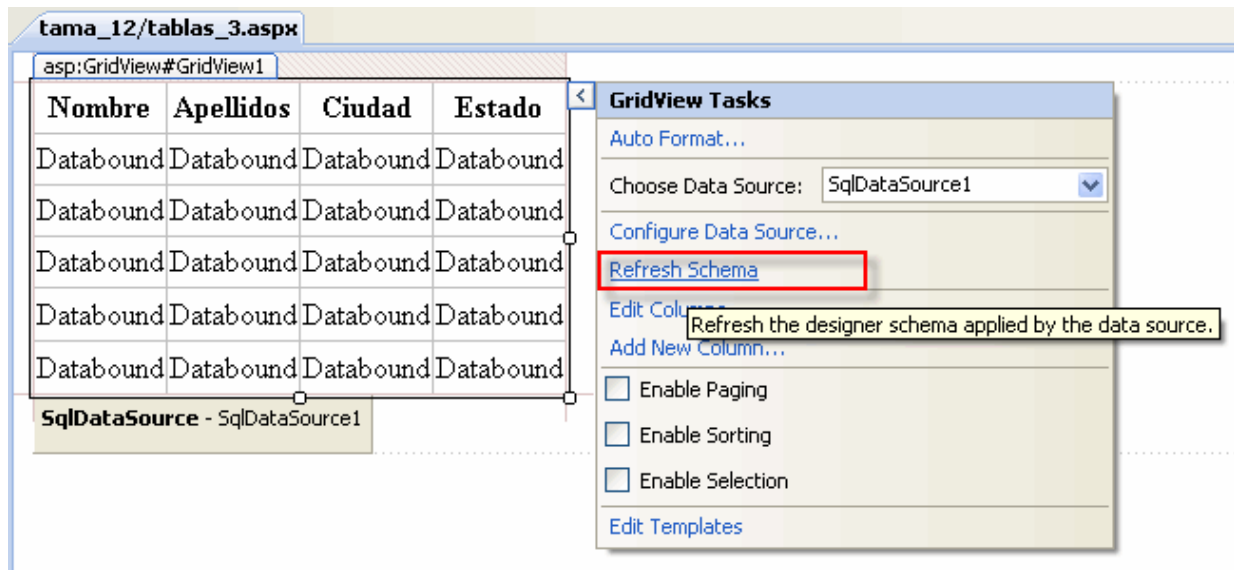
Hemos visto el cambio de la propiedad " HeaderText ", ésta es la única que podemos cambiar en ese momento. Veamos como podemos establecer otras propiedades en el momento del enlace:

Propiedad	Descripción
DataField	Identifica el nombre del campo que queremos mostrar en la columna
DataFormatString	Da formato al campo. Es muy útil a la hora de escribir números y fechas
ApplyFormatInEditMode	Si es " true " la cadena DataFormat se utiliza para dar formato al valor incluso cuando aparece dentro de un cuadro de texto para edición. Por defecto es " false " que significa que se utilizará el campo si n formato. Por ejemplo 1234,32 en lugar de 1.234,32 € si es de tipo moneda
FooterText, HeaderText y HeaderImageUrl	Establece el texto en el encabezado y pie de la cuadrícula si el encabezado y pie están configurados para mostrarse (GridView.ShowHeader a True y GridView.ShowFooter es True). Normalmente el encabezado contiene un texto descriptivo de la columna y el pie un valor dinámico calculado. Si queremos mostrar una imagen en el encabezado en lugar de un texto pondremos la propiedad HeaderImageUrl.
ReadOnly	Si es cierto, no se pueden editar las columnas. Muy útil para las columnas de las claves primarias de las tablas por ejemplo, ya que no pueden modificarse nunca
InsertVisible	Si es cierto no permitirá la edición. Lo utilizaremos en campos que sean calculados por código o con un valor predeterminado fijo.
Visible	Si es " false " la columna no se mostrará y no se enviará HTML para dibujarla. Proporciona una forma dinámica de mostrar u ocultar columnas según nuestras necesidades..
SortExpression	Ordena el resultado según una o varias columnas
HtmlEncode	Si es true se codifica en HTML el resultado para prevenir errores al utilizar caracteres problemáticos. Por ejemplo para escribir un texto que tenga los caracteres " < " y " > " ya que se podrían identificar como etiquetas.
NullDisplayText	Muestra el texto que se presenta en el caso de aparecer un valor nulo. Por defecto es una cadena vacía, pero podemos modificarlo para que sea por ejemplo " no aplicable
ConvertEmptyStringToNull	Si es cierto, convierte todas las cadenas vacías en valores nulos aplicando el valor del campo anterior
ControlStyle, HeaderStyle, FooterStyle y ItemStyle	Configura el aspecto para la columna sobrescribiendo los estilos de la fila

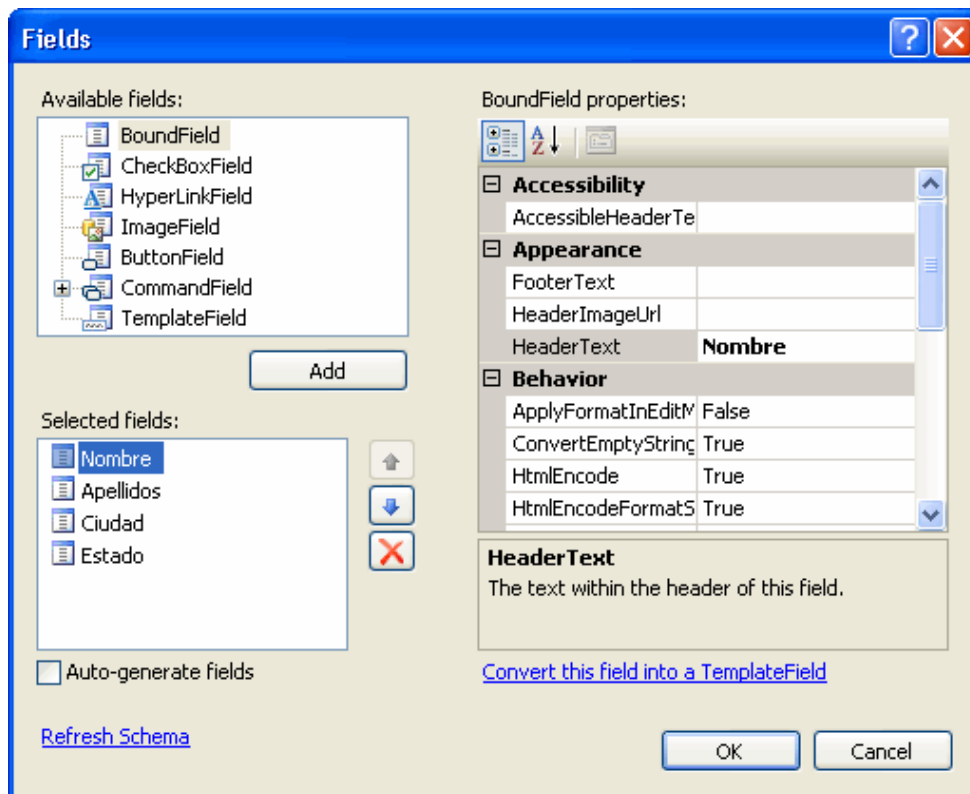
6.3 Generación de columnas

Hemos visto que podemos crear una cuadrícula estableciendo la propiedad AutoGenerateColumns a " true ". Pero veamos como perdíamos la posibilidad de detallar muchas opciones de las columnas, como el orden, formato, ordenación y otras. Para configurar estos detalles dejaremos AutoGenerateColumns a " False " y definiremos a mano nuestras columnas. Esto obviamente necesita de mas trabajo por nuestra parte pero la versatilidad será mucho mayor. Menos mal que nuestro editor nos va a ayudar con muchas de estas tareas para crear las columnas automáticamente. Para hacer esto seleccionaremos el control GridView y haremos clic en " Refresh Schema ". Aquí nuestro editor leerá el esquema o configuración de nuestro origen de datos: nombres y tipos de datos y nos permitirá enlazar los campos:

Enlace a datos. Control de cuadrícula



Una vez que hemos creado las columnas podemos utilizar como ves muchas propiedades para ajustar el aspecto y comportamiento de las columnas. Pulsamos en los puntos suspensivos () que hay junto a la propiedad "Columns" en la ventana de propiedades. Ahí se mostrará un cuadro de diálogo que nos permitirá añadir, eliminar y modificar nuestras columnas:

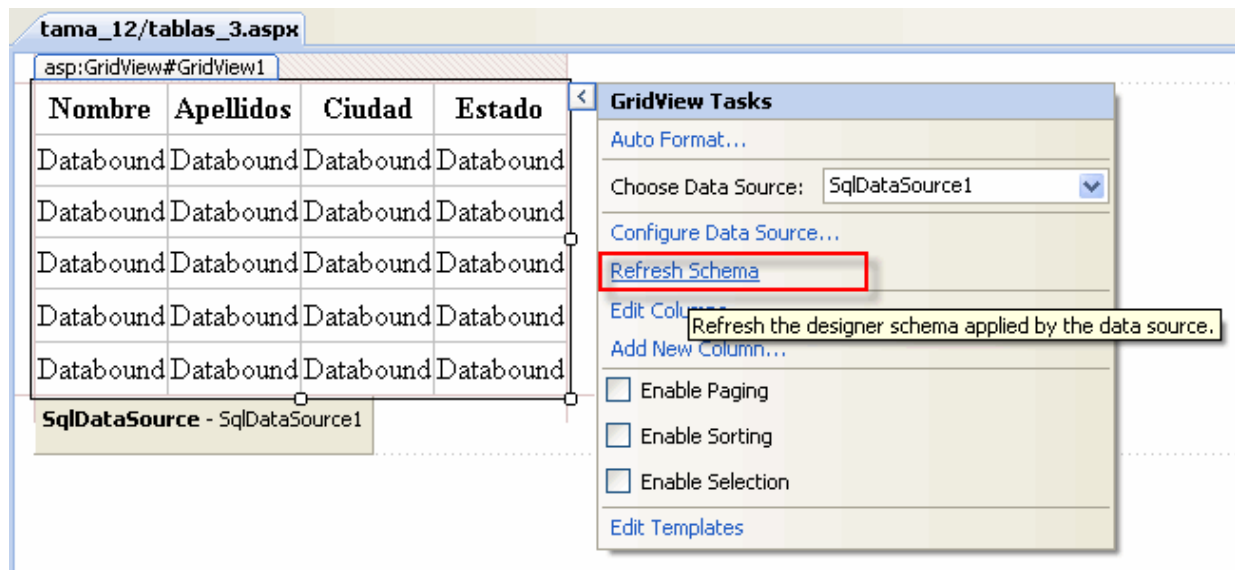


En esa pantalla he seleccionado el primer campo "Nombre" y puedes ver que es del tipo "BoundField", es decir, de momento un campo enlazado con "au_lname". Fíjate también en la cantidad de propiedades que tenemos para personalizar cada una de las columnas. De momento le hemos puesto un título "HeaderText"

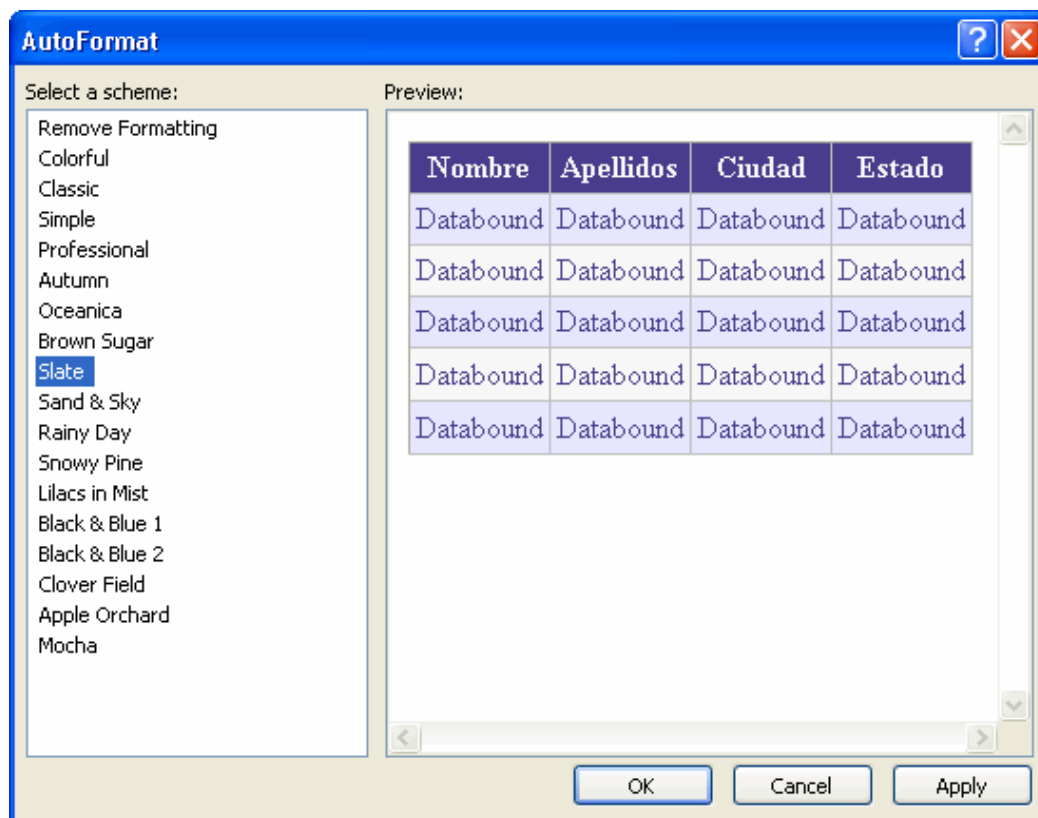
Enlace a datos. Control de cuadrícula

mas legible que el nombre del campo. A lo largo de esta sección iremos viendo todas o casi todas estas opciones.

Si te fijas en la pantalla anterior:



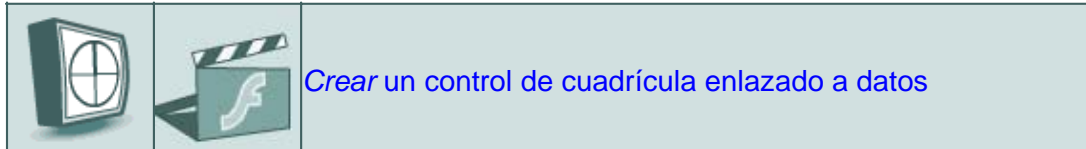
Tenemos en la primera opción el famoso "autoformato" que nos permite ponerle unos estilos completos a toda la tabla, por ejemplo:



Enlace a datos. Control de cuadrícula

Si le aplicas un estilo de estos fíjate en el HTML generado y verás los datos que ha cambiado respecto a los colores, espaciados, contornos... Ahora pasamos a detalles avanzados para hacer estas operaciones:

- Dar formato: Para dar formato a la columna y sus valores
- Seleccionar: Como seleccionará el usuario la fila
- Editar: Como modificará los datos el usuarios al insertar, eliminar o modificar datos
- Ordenar: Como ordenar dinámicamente según se seleccionen los encabezados de columna
- Pagar: Para dividir una resultado grande en varias páginas de datos
- Plantillas. Para tener un control completo del control diseñando, dando formato y editando las plantillas



6.4 Dar formato a la cuadrícula

La tarea de dar formato a nuestra cuadrícula consta de varias acciones. La primera es asegurarnos que los valores de fechas, monedas y otros valores se muestra de la forma adecuada. Esto lo realizaremos con la propiedad `DataFormatString`. El siguiente paso será aplicar un conjunto de colores adecuado, tipos de letra, contornos y la alineación de los datos. El control `GridView` nos permite poner todas estas características mediante los estilos. Después de todo esto podremos interceptar los eventos, examinar los datos de la fila y dar formato mediante programación. Además tendremos varias propiedades para el formato de las celdas como: `CellPadding`, `Gridlines`, `CellSpacing`, `Caption` y `CaptionAlign`.

Formato de los campos

Cada columna (boundfield) proporciona una propiedad “`DataFormatString`” que podemos utilizar para configurar el aspecto de valores numéricos y de fecha utilizando una “cadena de formato”.

Estas cadenas de formatos consisten en unas plantillas y un indicador de formato que se indica entre { y }. Por ejemplo:

{0:C}

El valor “0” representa el valor al que se le va a dar formato y la letra indica el estilo predeterminado. En ese caso “C” significa “currency” o de tipo moneda. Por lo que 3400,34 se convertirá a 3.400, . Aplicado a una columna será:

```
<asp:BoundField DataField="Price" HeaderText="Precio" DataFormatString="{0:C}" />
```

Veamos otros ejemplos:

Tipo	Cadena de formato	Ejemplo
Moneda	{0:C}	1.234,56
Científico (exponencial)	{0:E}	1.234.50E+004
Porcentaje	{0:P}	45,6%
Decimal fijo	{0:F?}	Según los decimales: {0:F3} dará 123,456. {0:F0} dará 123

Los valores de fecha y hora tienen también sus muchas posibilidades...

Enlace a datos. Control de cuadrícula

Tipo	Cadena de formato	Sintaxis	Ejemplo
Fecha corta	{0:d}	d/M/yyyy	30/10/2008
Fecha larga	{0:D}	dddd, dd MMMM, yyyy	Lunes, 10 Junio, 2008
Fecha larga y hora corta	{0:f}	dddd, dd MMMM, yyyy HH:mm:ss	Lunes, 10 Junio, 2008 10:00 AM
Fecha y horas largas	{0:F}	dddd, dd MMMM, yyyy HH:mm:ss aa	Lunes, 10 Junio, 2008 10:00:23 AM
Estandar ISO	{0:s}	yyyy-MM-ddTHH:mm:ss	2008-01-30T10:00:20
Día y mes	{0:M}	dd MMMM	30 Enero
General	{0:G}	d/M/yyyy HH:mm:ss aa	30/10/2008 10:00:23 AM

Por ejemplo pondríamos:

```
<asp:BoundField DataField="Date" HeaderText="Fecha" DataFormatString="{0:dd/MM/yy}" />
```

El formato para texto no está de momento contemplado, mas adelante podemos ver cómo realizarlo con otros controles o plantillas. Para los tipos de datos de Decimal y DateTime podremos utilizar además sus propios métodos como “ToString()” para dar formato.

Utilizar estilos

El control de cuadrícula tiene definidos muchos estilos, fíjate en esta tabla:

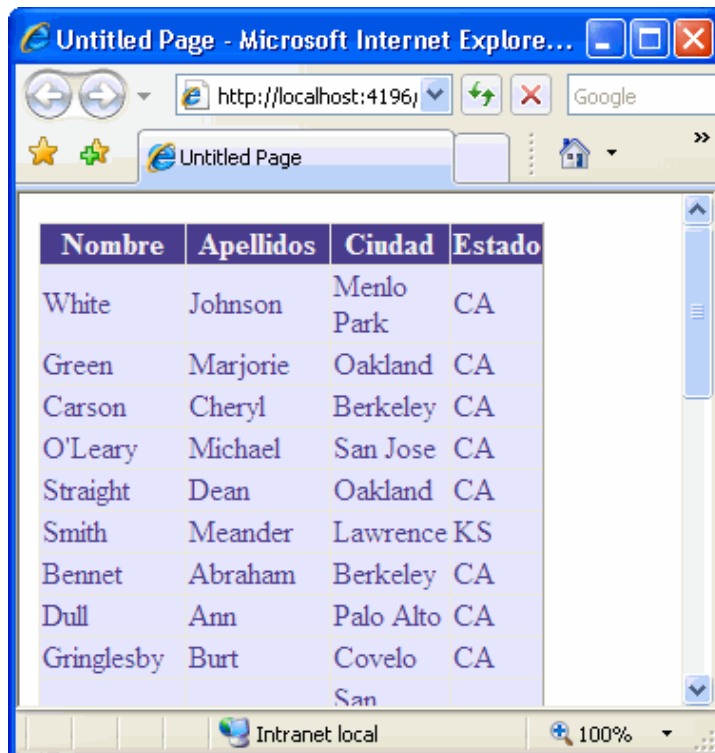
Estilo	Descripción
HeaderStyle	Configura el aspecto de la cabecera que contiene los títulos del encabezado de columna, si están visibles.
RowStyle	Configura el aspecto de las filas
AlternatingRowStyle	Si se establece, aplica formato adicional a cada fila de forma alternada. Este formato se aplica además del “RowStyle”
SelectedRowStyle	Configura el aspecto de la fila seleccionada. Este formato se suma al del “RowStyle”
EditRowStyle	Configura el aspecto de la fila que se está editando. Este formato se suma al del “RowStyle”
EmptyDataRowStyle	Configura el estilo para el caso de filas vacías
FooterStyle	Configura el aspecto para el pie de la cuadrícula, si se configura para que lo muestre
PagerStyle	Configura el aspecto de la fila con los enlaces de la paginación, si está activada

Cada estilo es un objeto “Style” con las propiedades del color, contornos, tamaños, alineación y tipos de letra: ForeColor, BackColor, BorderColor, BorderStyle, BorderWidth, Height, Width, HorizontalAlign, VerticalAlign, Font, ... Por ejemplo un estilo para los pies y encabezados:

Enlace a datos. Control de cuadrícula

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    DataSourceID="SqlDataSource1" Height="165px" Width="257px">
    <RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
    <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" />
    <Columns>
        <asp:BoundField DataField="au_lname" HeaderText="Nombre" />
        <asp:BoundField DataField="au_fname" HeaderText="Apellidos" />
        <asp:BoundField DataField="city" HeaderText="Ciudad" />
        <asp:BoundField DataField="state" HeaderText="Estado" />
    </Columns>
</asp:GridView>
```

Que nos da como resultado:



The screenshot shows a web browser window titled "Untitled Page - Microsoft Internet Explore...". The address bar shows "http://localhost:4196/". The page displays a table with four columns: "Nombre", "Apellidos", "Ciudad", and "Estado". The table contains data for various authors, including White, Green, Carson, O'Leary, Straight, Smith, Bennet, Dull, and Gringlesby. The table is styled with a light blue background and dark blue text.

Nombre	Apellidos	Ciudad	Estado
White	Johnson	Menlo Park	CA
Green	Marjorie	Oakland	CA
Carson	Cheryl	Berkeley	CA
O'Leary	Michael	San Jose	CA
Straight	Dean	Oakland	CA
Smith	Meander	Lawrence	KS
Bennet	Abraham	Berkeley	CA
Dull	Ann	Palo Alto	CA
Gringlesby	Burt	Covelo	CA
		San	

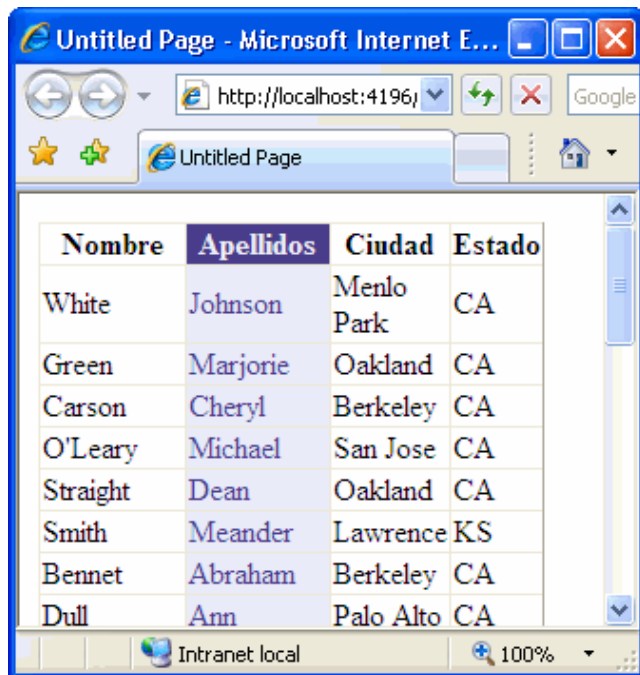
En este ejemplo, cada columna tiene el estilo aplicado, pero podemos definir estilos independientes además del general, simplemente poniendo los estilos dentro de la columna adecuada. Por ejemplo para dar formato a la columna del nombre del producto:

Enlace a datos. Control de cuadrícula

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    DataSourceID="SqlDataSource1" Height="165px" Width="257px">

    <Columns>
        <asp:BoundField DataField="au_lname" HeaderText="Nombre" />
        <asp:BoundField DataField="au_fname" HeaderText="Apellidos">
            <ItemStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
            <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" />
        </asp:BoundField >
        <asp:BoundField DataField="city" HeaderText="Ciudad" />
        <asp:BoundField DataField="state" HeaderText="Estado" />
    </Columns>
</asp:GridView>
```

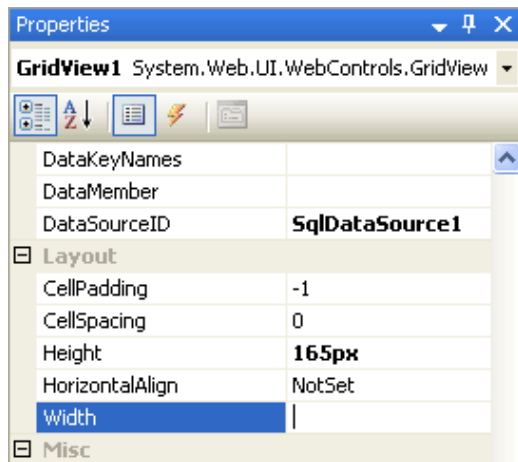
Que en este caso nos produce:



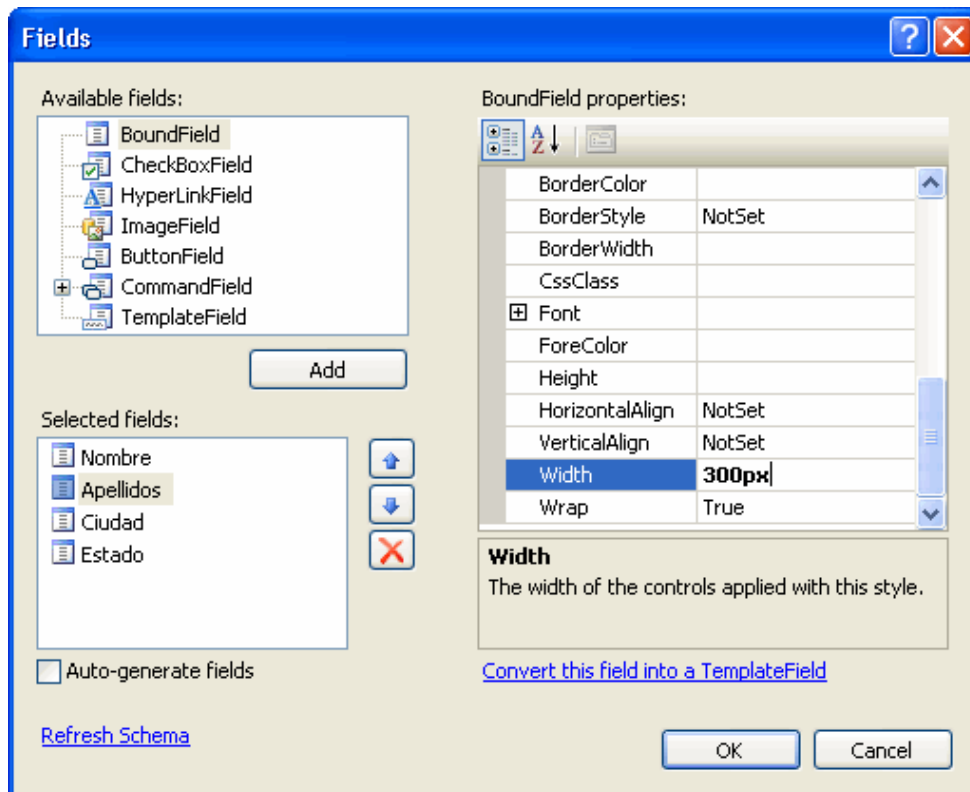
Nombre	Apellidos	Ciudad	Estado
White	Johnson	Menlo Park	CA
Green	Marjorie	Oakland	CA
Carson	Cheryl	Berkeley	CA
O'Leary	Michael	San Jose	CA
Straight	Dean	Oakland	CA
Smith	Meander	Lawrence	KS
Bennet	Abraham	Berkeley	CA
Dull	Ann	Palo Alto	CA

Otra de las razones para querer cambiar el aspecto de las columnas es por su anchura. ASP.NET ajusta esta anchura según los datos que contiene y en ocasiones seguramente queramos mas espacio que ese automático. Primero quitaremos la anchura que le pone de forma predeterminada al grid completo. Lo seleccionamos y en las propiedades a la derecha abajo borramos el tamaño prefijado para toda la tabla:

Enlace a datos. Control de cuadrícula

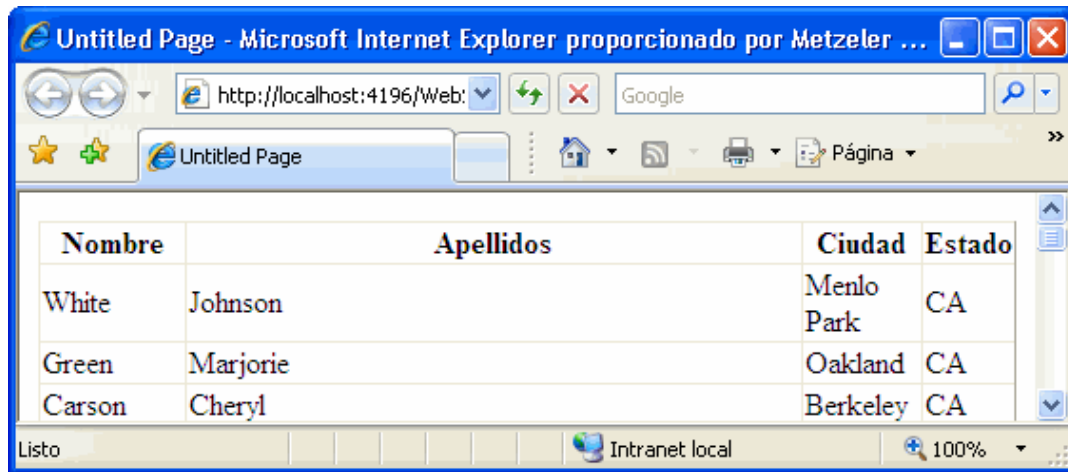


Luego en el campo que queremos modificaremos la anchura:

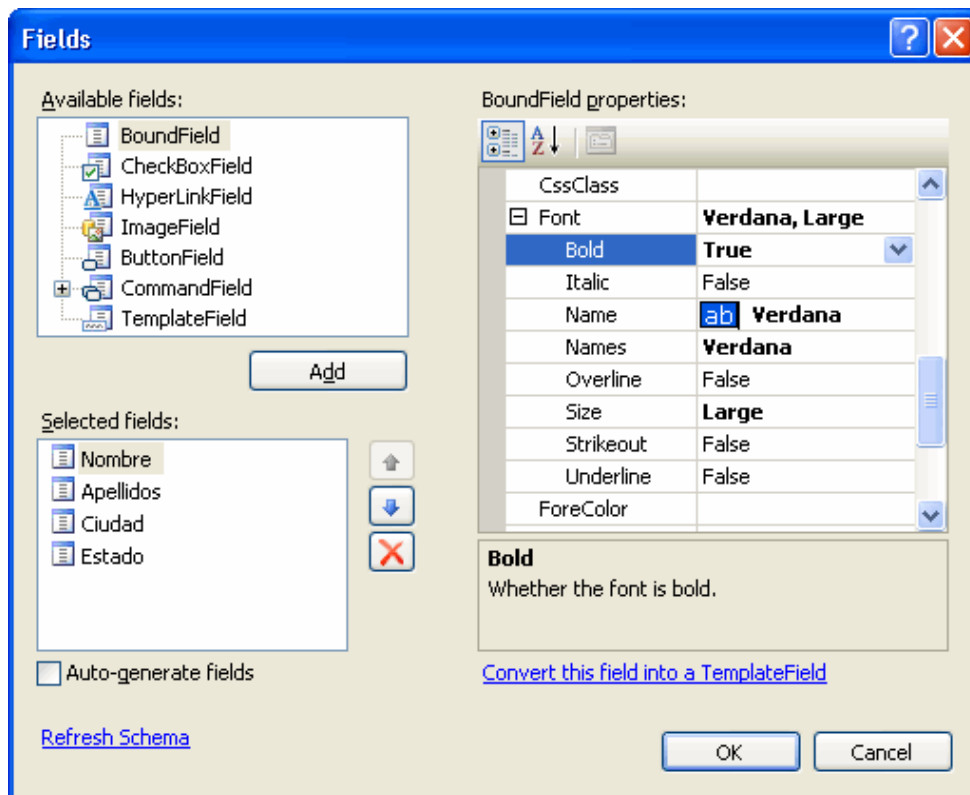


Y el resultado es un ancho personalizado:

Enlace a datos. Control de cuadrícula



También podemos ampliar la propiedad "HeaderStyle" para establecer el formato de la fuente:



Como ves las posibilidades son ilimitadas, así que a practicar un poco hasta que tengas un estilo uniforme en tu aplicación web.

Formato especial para datos

Ya sabemos como dar formato a toda la cuadrícula o a columnas de forma individual, pero en ocasiones necesitaremos personalizar el aspecto de las filas o incluso de una celda. Ya que quiero dar un formato especial a un campo de tipo moneda por ejemplo.

Enlace a datos. Control de cuadrícula

La solución para esto es hacerlo en “ejecución” que es cuando generamos la página donde pondremos el código adecuado en el evento “GridView.RowDataBound”, que es precisamente cuando se enlazan los datos. Este evento se dispara en cada fila, justo después de rellenarse con el dato. En ese momento accederemos al objeto adecuado de la colección de celdas “GridViewRow.Cells”. Podremos cambiar el color, el formato, la alineación... Vamos con un ejemplo, crea una nueva página, le pones un control de cuadrícula y un origen de datos y esta vez vas a enlazar la tabla de “titles” que son los libros disponibles con una columna que dice el precio. Las columnas que vamos a poner son:

Configure Data Source - SqlDataSource1

Configure the Select Statement

How would you like to retrieve data from your database?

☐ Specify a custom SQL statement or stored procedure

☒ Specify columns from a table or view

Name: titles

Columns:

<input type="checkbox"/> *	<input type="checkbox"/> advance
<input type="checkbox"/> title_id	<input type="checkbox"/> royalty
<input checked="" type="checkbox"/> title	<input type="checkbox"/> ytd_sales
<input checked="" type="checkbox"/> type	<input type="checkbox"/> notes
<input type="checkbox"/> pub_id	<input type="checkbox"/> pubdate
<input checked="" type="checkbox"/> price	

☐ Return only unique rows

WHERE...

ORDER BY...

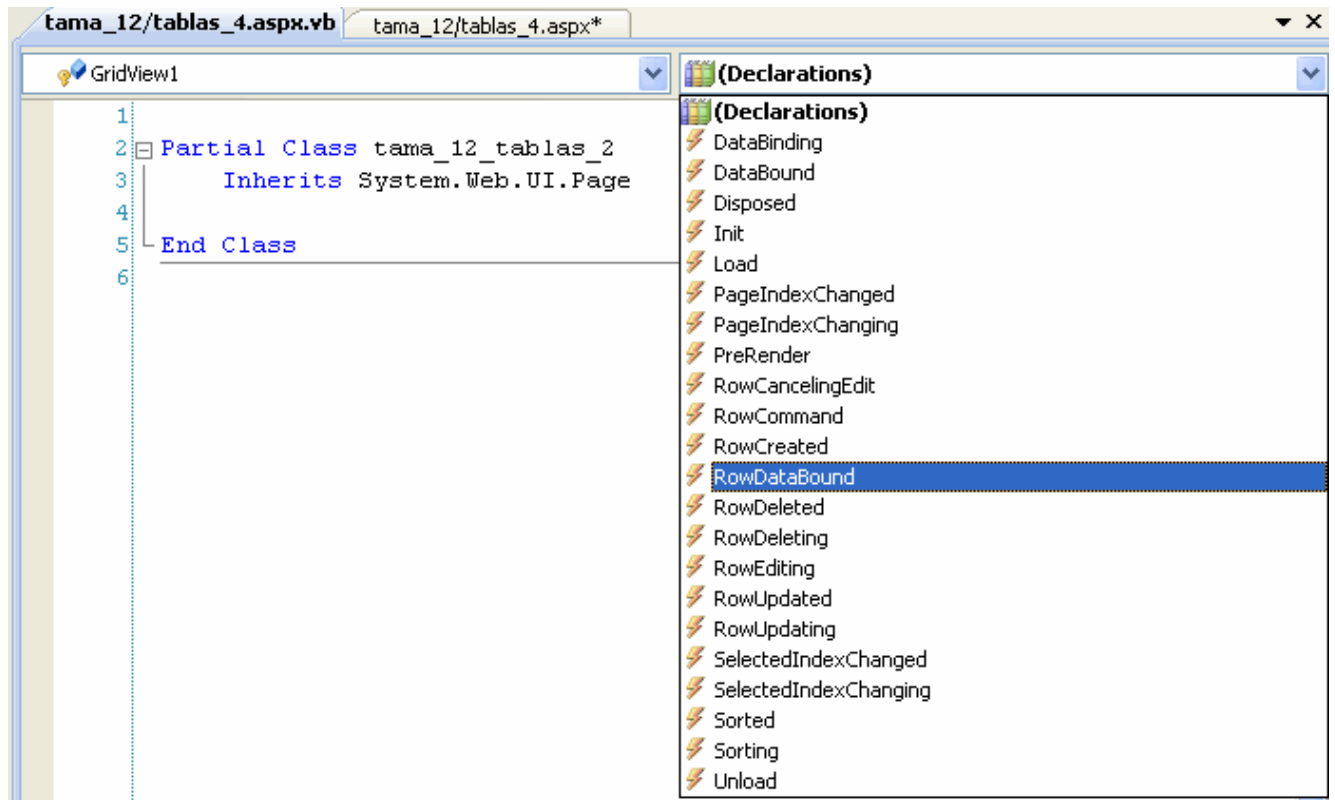
Advanced...

SELECT statement:

```
SELECT [title], [type], [price] FROM [titles]
```

< Previous Next > Finish Cancel

Y ahora queremos destacar los libros que valgan mas de 20 . Esta operación la vamos a realizar justo en el momento en que se enlacen los datos a la cuadrícula ya que ahí es donde podremos leer los valores que se van a poner a las celdas y así aplicarle un formato ejecutar alguna acción. Elegimos pues el evento:



Donde pondremos...

```

If e.Row.RowType = DataControlRowType.DataRow Then
    ' Obtenemos el precio de la fila...
    Dim precio As Decimal

    'Comprobamos que no sea nulo, ya que sino nos daría error en la página
    If Not IsDBNull(DataBinder.Eval(e.Row.DataItem, "Price")) Then
        'Convertimos el tipo de datos adecuado
        precio = CType(DataBinder.Eval(e.Row.DataItem, "Price"), Decimal)
    Else
        precio = 0
    End If

    'Hacemos la comparación, si es >20 le ponemos propiedades de formato...
    If precio > 20 Then
        e.Row.BackColor = System.Drawing.Color.Maroon
        e.Row.ForeColor = System.Drawing.Color.White
        e.Row.Font.Bold = True
    End If
End If

```

En primer lugar comprobamos si lo que se está escribiendo es una fila u otro elemento de la tabla como el encabezado o pie, en ese caso no se hace nada. Si es una fila de datos se extrae el precio del elemento enlazado. Fijate en el código y recuerda que el parámetro de entrada en los controladores de eventos llamado “e” tiene el tipo de objeto adecuado al control que maneja. En este caso podemos ver los datos de las filas:

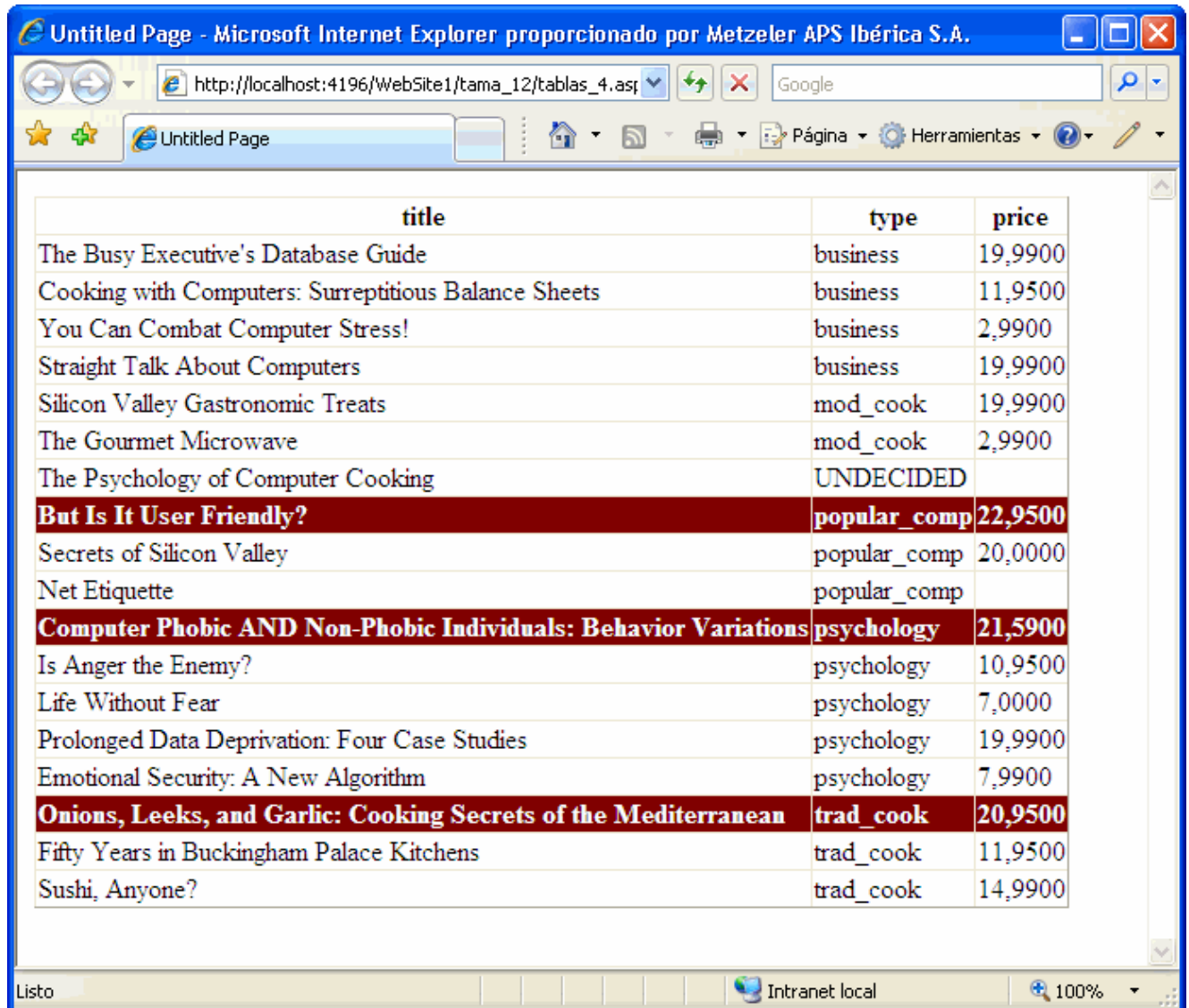
`e.Row.DataItem`

Enlace a datos. Control de cuadrícula

Ahora necesitaremos convertir al tipo de datos adecuado el dato que queremos tratar:

```
price = CType(DataBinder.Eval(e.Row.DataItem, "UnitPrice"), Decimal)
```

Así precio es ahora un valor numérico que podemos comparar y tratar posteriormente. “DataBinder.Eval” entiende cualquier tipo de datos así que nos lo extrae para luego poder hacer un “CType” y poder convertirlo a Decimal. El resultado final es:



title	type	price
The Busy Executive's Database Guide	business	19,9900
Cooking with Computers: Surreptitious Balance Sheets	business	11,9500
You Can Combat Computer Stress!	business	2,9900
Straight Talk About Computers	business	19,9900
Silicon Valley Gastronomic Treats	mod_cook	19,9900
The Gourmet Microwave	mod_cook	2,9900
The Psychology of Computer Cooking	UNDECIDED	
But Is It User Friendly?	popular_comp	22,9500
Secrets of Silicon Valley	popular_comp	20,0000
Net Etiquette	popular_comp	
Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	21,5900
Is Anger the Enemy?	psychology	10,9500
Life Without Fear	psychology	7,0000
Prolonged Data Deprivation: Four Case Studies	psychology	19,9900
Emotional Security: A New Algorithm	psychology	7,9900
Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	20,9500
Fifty Years in Buckingham Palace Kitchens	trad_cook	11,9500
Sushi, Anyone?	trad_cook	14,9900

6.5 Seleccionar una fila

El proceso de seleccionar una fila es obviamente cuando el usuario quiera seleccionar para, habitualmente, editarla o como selección de un proceso posterior. Sabemos que había un estilo especial para la selección de filas: `SelectedRowStyle` ya que además de destacarla vamos a querer escribir información adicional sobre esa fila, de ahí la importancia del proceso de selección.

Para encontrar el elemento que se ha seleccionado podemos utilizar la propiedad “`GridView.SelectedIndex`”

Enlace a datos. Control de cuadrícula

que será -1 si no hay ninguno seleccionado. También podremos reaccionar ante el evento "SelectedIndexChanged" para aportar información adicional.

Afladir un botón de selección.

Las cuadrículas incorporan un botón para que el usuario seleccione la fila. Para que nos aparezca debemos afladir una columna de tipo "Commandfield" con la propiedad "ShowSelectButton" a "true". ASP.NET nos creará el botón con tres formas distintas: botón, hipervínculo o imagen que seleccionaremos en la propiedad "ButtonType". Podremos especificar el texto en la propiedad "SelectText" o un enlace con la propiedad "SelectImageUrl". Por ejemplo:

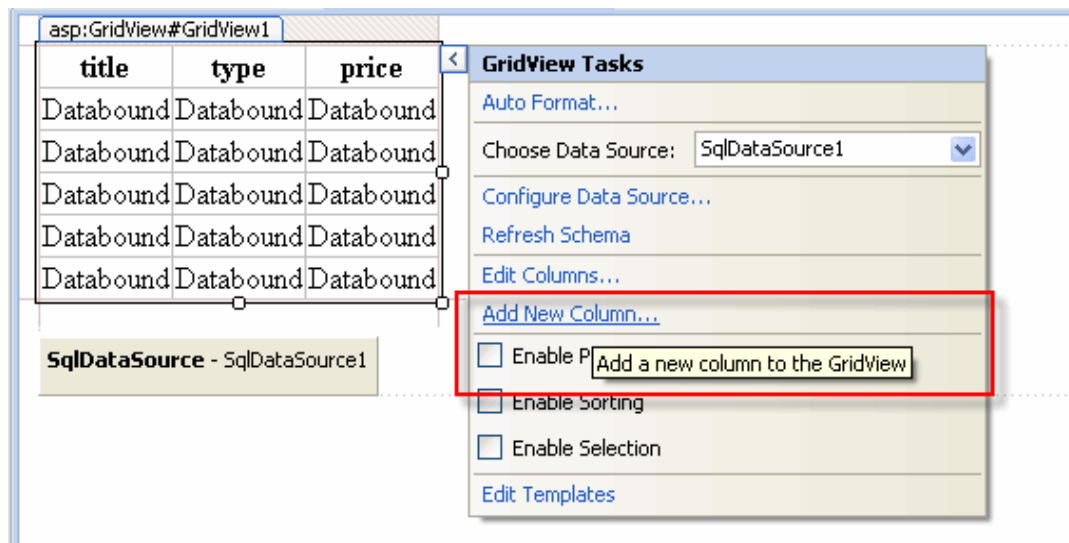
```
<asp:CommandField ShowSelectButton="True" ButtonType="Button" SelectText="Select" />
```

Y con un icono:

```
<asp:CommandField ShowSelectButton="True" ButtonType="Image" SelectImageUrl="select.gif" />
```

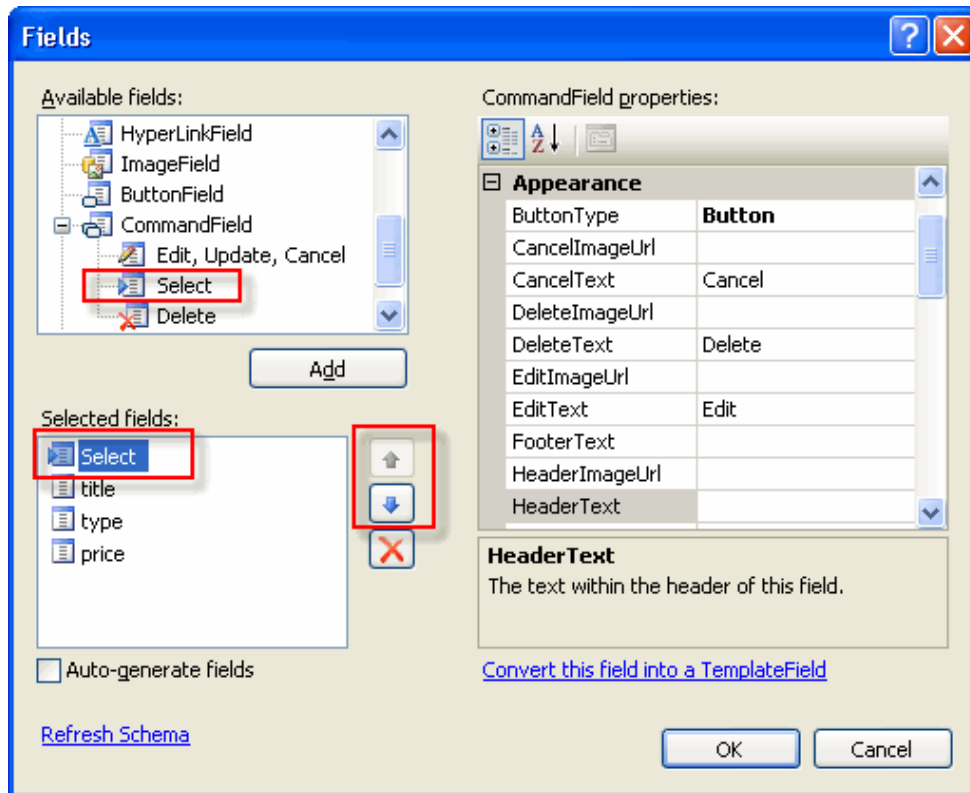
Veamos un ejemplo de cada:

Afladimos primero una columna, ésta la podemos afladir de dos formas, bien seleccionando:

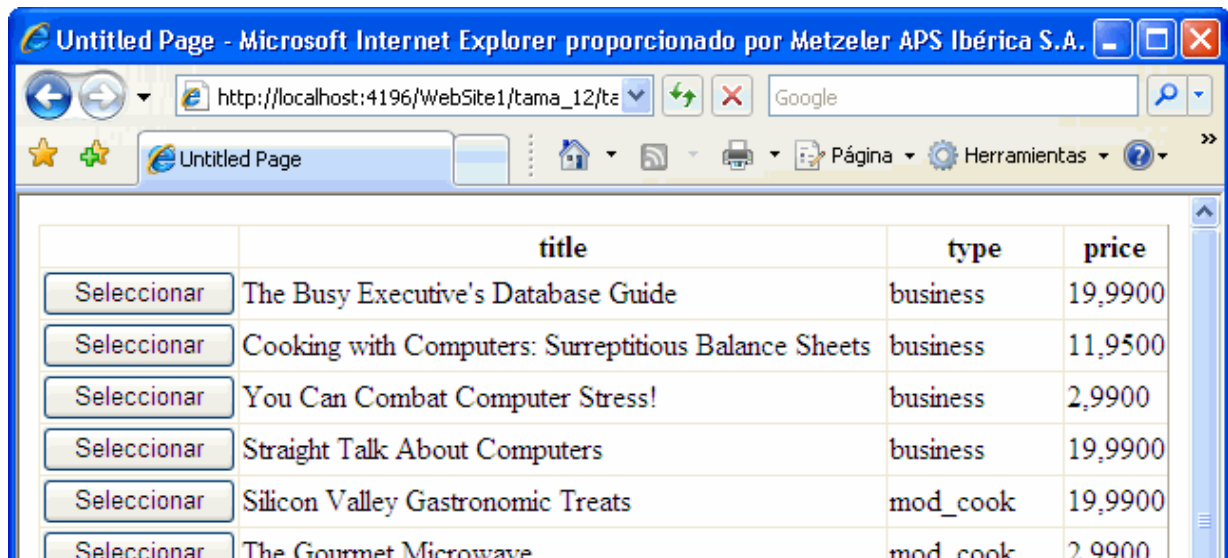


O desde el enlace de antes de "Edit Columns". Te recomiendo que utilices esta opción es mejor que la otra. Así que afladimos:

Enlace a datos. Control de cuadrícula

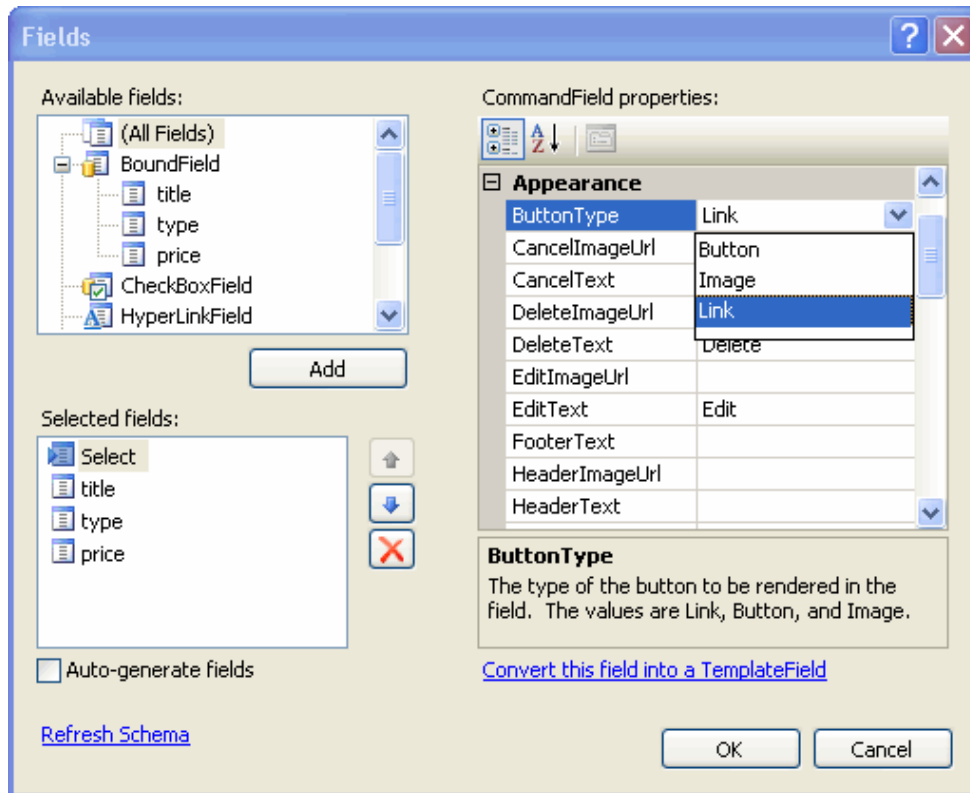


De los comandos elegimos "Select" y como aspecto le pondremos "ButtonType=Button". Puedes ordenar las columnas con las flechas que tienes a la derecha de los campos seleccionados. Esta selección nos dará como resultado:

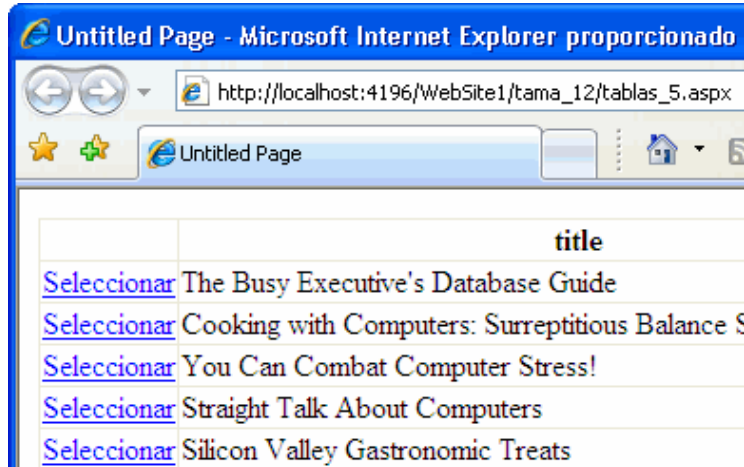


Si en el estilo le ponemos en vez de botón, enlace, quedará mas claro:

Enlace a datos. Control de cuadrícula



Con el resultado:

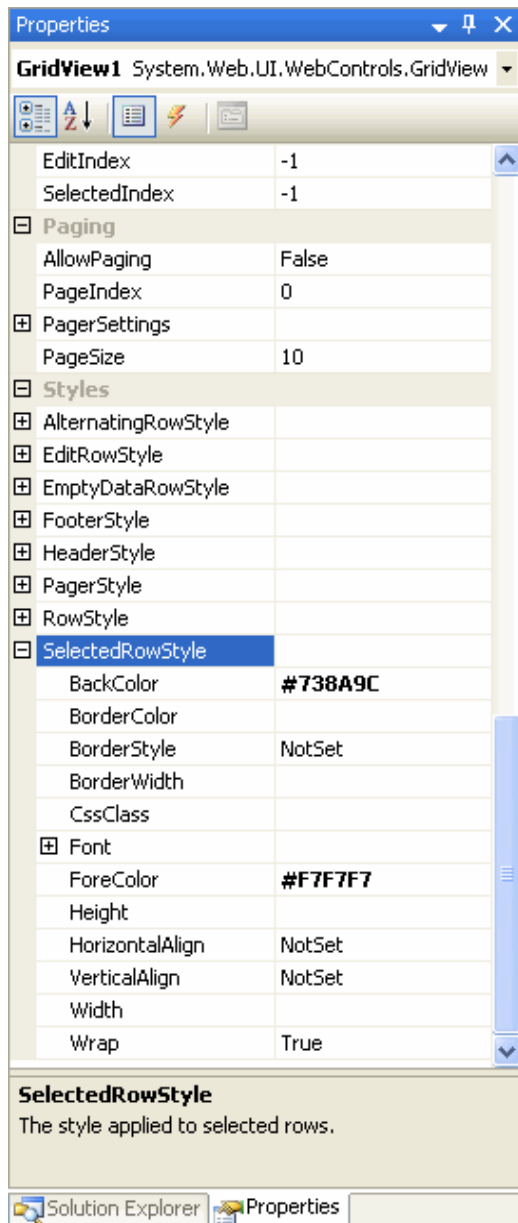


Y si queremos un dibujo como un lapicero por ejemplo, para que el usuario edite la linea, utilizariamos la otra opción:

```
<asp:CommandField ShowSelectButton="True" ButtonType="Image" SelectImageUrl="select.gif" />
```

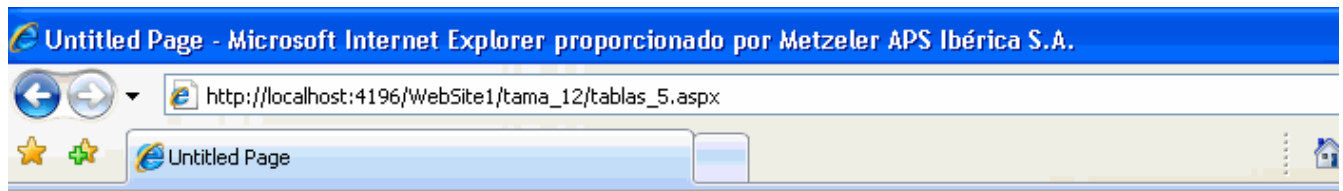
Si ejecutas la página verás que al pulsar no pasa nada!. No se seleccionada nada, esto es porque todavia no hemos puesto un estilo para la fila seleccionada, asi que tendríamos que ponerle un estilo en la ventana de propiedades de la cuadrícula. Ojo no confundas estas propiedades de la cuadrícula con las de la las columnas. Ahora lo que quiero es aplicar un estilo a la fila seleccionada, no tiene nada que ver con las definiciones de las columnas:

Enlace a datos. Control de cuadrícula



Así que le pondré los valores que queramos en el estilo "SelectedRowStyle". De todas formas y por simplificar, le voy a aplicar un autoformato a ver si funciona:

Enlace a datos. Control de cuadrícula



	title	type	price
Seleccionar	The Busy Executive's Database Guide	business	19,9900
Seleccionar	Cooking with Computers: Surreptitious Balance Sheets	business	11,9500
Seleccionar	You Can Combat Computer Stress!	business	2,9900
Seleccionar	Straight Talk About Computers	business	19,9900
Seleccionar	Silicon Valley Gastronomic Treats	mod_cook	19,9900
Seleccionar	The Gourmet Microwave	mod_cook	2,9900

Efectivamente, al pulsar en seleccionar tenemos nuestra fila seleccionada.

Sigamos, cuando pulsamos el botón la página realiza un “postback” y se producen los siguientes pasos... Primero, se dispara el evento “GridView.SelectedIndexChanging” que podremos interceptar para cancelar la operación, por ejemplo. Después la propiedad “GridView.SelectedIndex” se actualiza para apuntar a la fila seleccionada. Por último se dispara el evento “GridView.SelectedIndexChanged” que podremos editar para actualizar manualmente la selección. Por ejemplo:

Utilizar un campo de datos como botón de selección.

En ocasiones no vamos a querer añadir una columna más para una selección, sino que queremos convertir una columna en enlace para que luego podamos ir a una página de detalles por ejemplo. Así nos ahorramos ese espacio en la página web y además es la forma más utilizada. Para esto eliminamos la columna del botón que hemos añadido (CommandField) y añadimos una de tipo “ButtonField”. Finalmente establecemos la propiedad “DataTextField” al campo que queremos utilizar.

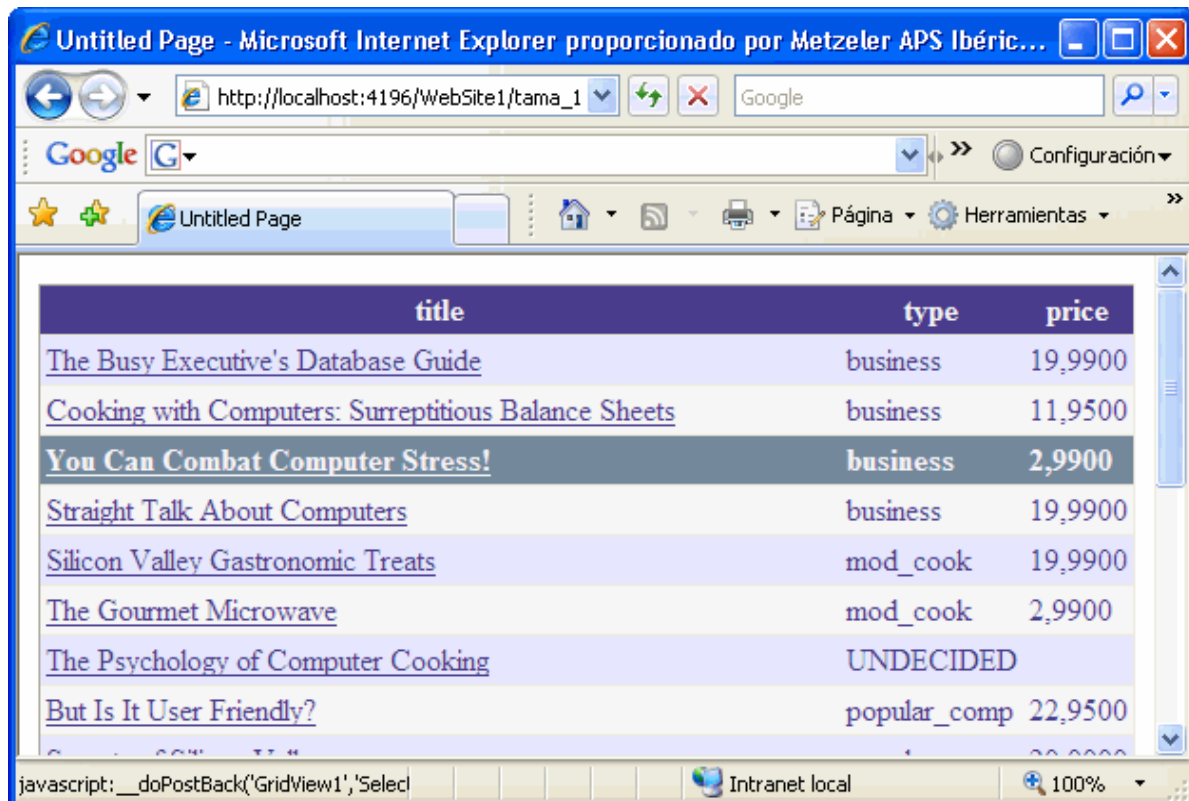
```
<asp:ButtonField ButtonType="Button" DataTextField="ProductID"/>
```

Este campo quedará subrayado y se comportará como un enlace que disparará el evento “GridView.RowCommand”. El resto de la programación ya la conoces para ver la fila que es. Aunque es sencillo podemos hacerlo más fácil todavía haciendo que el botón envíe además el índice del producto, así nos ahorramos el código en el evento anterior:

```
<asp:ButtonField CommandName="Select" ButtonType="Link" DataTextField="ProductID"/>
```

En nuestro ejemplo pondríamos:

Y el resultado es mejor que tener que poner una nueva columna para seleccionar:



Aunque esto depende de nuestra lógica de la página, porque a veces puede que sea simplemente un enlace a otra página donde lo pongamos detalles sobre ese libro.

Utilizar la selección en páginas maestro-detalle

Ya hemos visto hace unos cuantos ejemplos como seleccionábamos un elemento y luego le mostrábamos los detalles al usuario. Esta forma de mostrar los datos: primero una lista principal y luego los detalles del elemento seleccionado es una de las mas habituales en las explotaciones de las bases de datos y se llama “maestro-detalle“. Una página de este tipo suele tener dos cuadrículas, una con la lista de los elementos y la de debajo con todos los detalles del seleccionado: una lista de clientes arriba y la lista de pedidos por cliente seleccionado debajo.

Para crear una página de este tipo necesitaremos extraer la propiedad “selectedIndex“ del primer datagrid que es el que ha seleccionado el usuario y utilizarlo como índice de consulta para el segundo. Pero esto todavía no es demasiado útil porque nos devuelve el número de fila seleccionada que no es ningún valor de la tabla. Debería devolvernos el índice de esa fila, que sería la referencia del cliente, por ejemplo. Ya que por su posición en la tabla no podemos realizar ninguna consulta a la base de datos. Por tanto si es una lista de productos, tenemos que obtener el identificador del libro seleccionado.

La forma de realizar esto es estableciendo la propiedad “DataKeyNames“ de la cuadrícula. Esta propiedad necesita una lista de campos del origen de datos separados por comas. Lo normal es que utilizemos solo uno, veamos un ejemplo. Vamos a utilizar las tablas de los libros “titles“ y luego un detalle de las ventas de ellos (sales) en otra cuadrícula debajo. La relación está en el “title_id“ que está presente lógicamente en las dos tablas, en una identificando cada libro y en la otra indicando las ventas de cada “id“ de libro.

Enlace a datos. Control de cuadrícula

Gráficamente crea un enlace de datos a la tabla de libros (“titles”) como en otras ocasiones y ahora pon otra cuadrícula con otro origen de datos que vamos a configurar ahora. Quedaria así con autoformatos:

Libros:

	title_id	title	type	price	pubdate
Selecc.	abc	abc	abc	0	14/02/2008 0:00:00
Selecc.	abc	abc	abc	0,1	14/02/2008 0:00:00
Selecc.	abc	abc	abc	0,2	14/02/2008 0:00:00
Selecc.	abc	abc	abc	0,3	14/02/2008 0:00:00
Selecc.	abc	abc	abc	0,4	14/02/2008 0:00:00

SqlDataSource - SqlDataSource1

asp:gridview#GridView2

Pedido	Fecha	Cantidad	Pago
abc	14/02/2008 0:00:00	0	abc
abc	14/02/2008 0:00:00	1	abc
abc	14/02/2008 0:00:00	2	abc
abc	14/02/2008 0:00:00	3	abc
abc	14/02/2008 0:00:00	4	abc

SqlDataSource - SqlDataSource2

GridView Tasks

Auto Format...

Choose Data Source: SqlDataSource2

Configure Data Source...

Refresh Schema

Edit Columns...

Add New Column...

☐ Enable Paging

☐ Enable Sorting

☐ Enable Selection

Edit Templates

Vamos a configurar el segundo origen de datos, le decimos es la tabla “sales”:

Configure Data Source - SqlDataSource2

Configure the Select Statement

How would you like to retrieve data from your database?

☐ Specify a custom SQL statement or stored procedure

☒ Specify columns from a table or view

Name: sales

Columns:

<input type="checkbox"/>	*
<input type="checkbox"/>	stor_id
<input checked="" type="checkbox"/>	ord_num
<input checked="" type="checkbox"/>	ord_date
<input checked="" type="checkbox"/>	qty
<input checked="" type="checkbox"/>	payterms
<input checked="" type="checkbox"/>	title_id

☐ Return only unique rows

WHERE...

ORDER BY...

Advanced...

SELECT statement:

```
SELECT [ord_num], [ord_date], [qty], [payterms], [title_id] FROM [sales]
```

< Previous Next > Finish Cancel

Y ahora le tendremos que indicar que filtre para que nos muestre solo los del indice seleccionado en el grid de arriba, asi que pulsamos en “where” para añadirle una restricción:

Add WHERE Clause

Add one or more conditions to the WHERE clause for the statement. For each condition you can specify either a literal value or a parameterized value. Parameterized values get their values at runtime based on their properties.

Column: title_id

Operator: =

Source: Control

Parameter properties

Control ID: GridView1

Default value:

SQL Expression: [title_id] = @title_id2

Value: GridView1.SelectedValue

Add

WHERE clause:

SQL Expression	Value
[title_id] = @title_id	GridView1.SelectedValue

Remove

OK Cancel

Enlace a datos. Control de cuadrícula

Así que le estamos diciendo que consulta todas la ventas cuyo índice de libro sea el índice seleccionado en el grid1. Por eso lo hemos puesto que ese campo es de “Source=Control” y a la derecha le hemos dicho que el control es la primera cuadrícula “Gridview1”. Al final pulsamos en “add” y queda añadida la restricción, que si ves debajo pone:

```
[title_id]=@title_id
```

Que es justo un parámetro de SQL que ya conocemos. Pulsamos OK y terminamos para ejecutar la página:



	title_id	title	type	price	pubdate
Selecc.	BU1032	The Busy Executive's Database Guide	business	19,9900	12/06/1991 0:00:00
Selecc.	BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	11,9500	09/06/1991 0:00:00
Selecc.	BU2075	You Can Combat Computer Stress!	business	2,9900	30/06/1991 0:00:00
Selecc.	BU7832	Straight Talk About Computers	business	19,9900	22/06/1991 0:00:00
Selecc.	MC2222	Silicon Valley Gastronomic Treats	mod_cook	19,9900	09/06/1991 0:00:00
Selecc.	MC3021	The Gourmet Microwave	mod_cook	2,9900	18/06/1991 0:00:00
Selecc.	MC3026	The Psychology of Computer Cooking	UNDECIDED		06/08/2000 1:33:54
Selecc.	PC1035	But Is It User Friendly?	popular_comp	22,9500	30/06/1991 0:00:00
Selecc.	PC8888	Secrets of Silicon Valley	popular_comp	20,0000	12/06/1994 0:00:00
Selecc.	PC9999	Net Etiquette	popular_comp		06/08/2000 1:33:54
Selecc.	PS1372	Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	21,5900	21/10/1991 0:00:00
Selecc.	PS2091	Is Anger the Enemy?	psychology	10,9500	15/06/1991 0:00:00
Selecc.	PS2106	Life Without Fear	psychology	7,0000	05/10/1991 0:00:00
Selecc.	PS3333	Prolonged Data Deprivation: Four Case Studies	psychology	19,9900	12/06/1991 0:00:00
Selecc.	PS7777	Emotional Security: A New Algorithm	psychology	7,9900	12/06/1991 0:00:00
Selecc.	TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	20,9500	21/10/1991 0:00:00
Selecc.	TC4202	Fifty Veggies in Sixty Minutes: Deluxe Kitchen	trad_cook	11,9500	12/06/1991 0:00:00

Nos muestra una cuadrícula, pero cuando seleccionemos una fila:

Enlace a datos. Control de cuadrícula

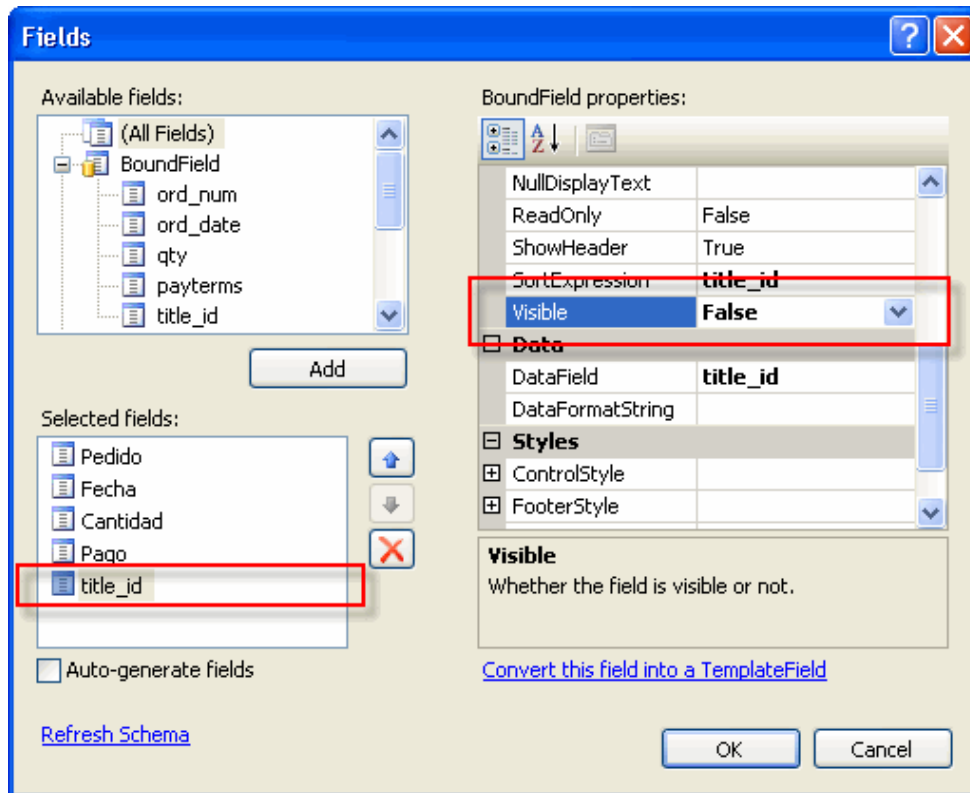
Libros:

	title_id	title	type	price	pubdate
Selecc.	BU1032	The Busy Executive's Database Guide	business	19,9900	12/06/1991 0:00:00
Selecc.	BU1111	Cooking with Computers: Surreptitious Balance Sheets	business	11,9500	09/06/1991 0:00:00
Selecc.	BU2075	You Can Combat Computer Stress!	business	2,9900	30/06/1991 0:00:00
Selecc.	BU7832	Straight Talk About Computers	business	19,9900	22/06/1991 0:00:00
Selecc.	MC2222	Silicon Valley Gastronomic Treats	mod_cook	19,9900	09/06/1991 0:00:00
Selecc.	MC3021	The Gourmet Microwave	mod_cook	2,9900	18/06/1991 0:00:00
Selecc.	MC3026	The Psychology of Computer Cooking	UNDECIDED		06/08/2000 1:33:54
Selecc.	PC1035	But Is It User Friendly?	popular_comp	22,9500	30/06/1991 0:00:00
Selecc.	PC8888	Secrets of Silicon Valley	popular_comp	20,0000	12/06/1994 0:00:00
Selecc.	PC9999	Net Etiquette	popular_comp		06/08/2000 1:33:54
Selecc.	PS1372	Computer Phobic AND Non-Phobic Individuals: Behavior Variations	psychology	21,5900	21/10/1991 0:00:00
Selecc.	PS2091	Is Anger the Enemy?	psychology	10,9500	15/06/1991 0:00:00
Selecc.	PS2106	Life Without Fear	psychology	7,0000	05/10/1991 0:00:00
Selecc.	PS3333	Prolonged Data Deprivation: Four Case Studies	psychology	19,9900	12/06/1991 0:00:00
Selecc.	PS7777	Emotional Security: A New Algorithm	psychology	7,9900	12/06/1991 0:00:00
Selecc.	TC3218	Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean	trad_cook	20,9500	21/10/1991 0:00:00
Selecc.	TC4203	Fifty Years in Buckingham Palace Kitchens	trad_cook	11,9500	12/06/1991 0:00:00
Selecc.	TC7777	Sushi, Anyone?	trad_cook	14,9900	12/06/1991 0:00:00

Pedido	Fecha	Cantidad	Pago
N914014	14/09/1994 0:00:00	25	Net 30
423LL922	14/09/1994 0:00:00	15	ON invoice

Nos muestra debajo la otra consulta, con las ventas de ese libro. Para no mostrar el identificador del libro en esta segunda tabla, le he puesto a la columna del "title_id" la propiedad visible=false:

Enlace a datos. Control de cuadrícula



Como ves es una muy potente combinación que muestra la típica consulta en forma de maestro-detalle. El quiz de la cuestión está en la parametrización del segundo datasource:

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%$ ConnectionStrings:Conexion_Pubs %>"
    SelectCommand="SELECT [ord num], [ord date], [qty], [payterms], [title_id]
        FROM [sales] WHERE ([title id] = @title id)">
    <SelectParameters>
        <asp:ControlParameter ControlID="GridView1" Name="title_id"
            PropertyName="SelectedValue" Type="String" />
    </SelectParameters>
</asp:SqlDataSource>
```

Si te fijas en la sentencia SQL la comparación del Where es con el “placeholder @title_id”. En el primer Datagrid también tenemos un detalle importante porque de alguna forma tiene que “exportar” el índice.

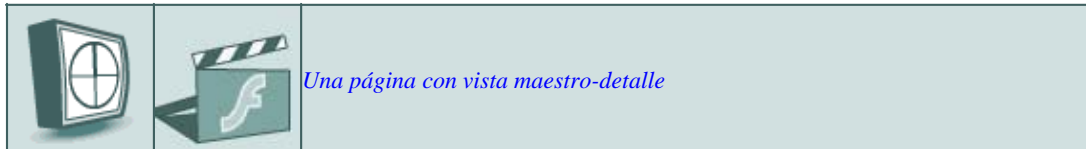
Una vez que hemos establecido el enlace, nuestro Datagrid almacenará ese valor que podremos recuperar en el evento que queramos tratar mediante la propiedad “SelectedDataKey”.

Enlace a datos. Control de cuadrícula

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
  DataKeyNames="title_id" DataSourceID="SqlDataSource1">
  <Columns>
    <asp:CommandField SelectText="Selecc." ShowSelectButton="True" />
    <asp:BoundField DataField="title_id" HeaderText="title_id" ReadOnly="True"
      SortExpression="title_id" />
    <asp:BoundField DataField="title" HeaderText="title" SortExpression="title" />
    <asp:BoundField DataField="type" HeaderText="type" SortExpression="type" />
    <asp:BoundField DataField="price" HeaderText="price" SortExpression="price" />
    <asp:BoundField DataField="pubdate" HeaderText="pubdate"
      SortExpression="pubdate" />
  </Columns>
</asp:GridView>
```

Como puedes ver, hemos utilizado los dos Datagrid que hemos comentado, el segundo origen de datos utiliza un “ControlParameter” que lo enlaza con la propiedad “SelectedDataKey” del primer Datagrid. Y lo mejor de todo: ¡no hemos tenido que escribir ni una sola línea de código, ni siquiera un controlador de eventos!

El ejemplo completo lo puedes descargar como siempre al final de la página.



6.6 Editar datos en la cuadrícula.

Una vez que sabemos como seleccionar los datos, la cuadrícula nos va a proporcionar ahora la posibilidad de editarlos. Para cambiar una fila a modo de selección simplemente cambiábamos la propiedad “selectedIndex” al número de fila. Para activar ahora la edición haremos lo mismo con la propiedad “EditIndex”. Estas dos acciones las realizaremos a la vez utilizando botones especiales. Para la selección utilizamos la comuna CommandField con la propiedad ShowSelectButton establecida a True. Para añadir controles de edición haremos el mismo paso con la columna “CommandFiled” pero esta vez con “Show EditButton” a “true”. Fijate en este ejemplo:

Enlace a datos. Control de cuadrícula

```
<form id="form1" runat="server">
<div>

    <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
        DataSourceID="SqlDataSource1">
        <Columns>
            <asp:BoundField DataField="au_id" HeaderText="ID" ReadOnly="True" />
            <asp:BoundField DataField="au_lname" HeaderText="Nombre" />
            <asp:BoundField DataField="au_fname" HeaderText="Apellidos" />
            <asp:BoundField DataField="address" HeaderText="Dirección" />
            <asp:BoundField DataField="city" HeaderText="Ciudad" />
            <asp:CommandField ShowEditButton="True" />
        </Columns>
    </asp:GridView>
    <br />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%= $ConnectionStrings:Conexion_Pubs %>"
        SelectCommand="SELECT [au_lname], [au_fname], [address], [city], [au_id] FROM [a
    </asp:SqlDataSource>

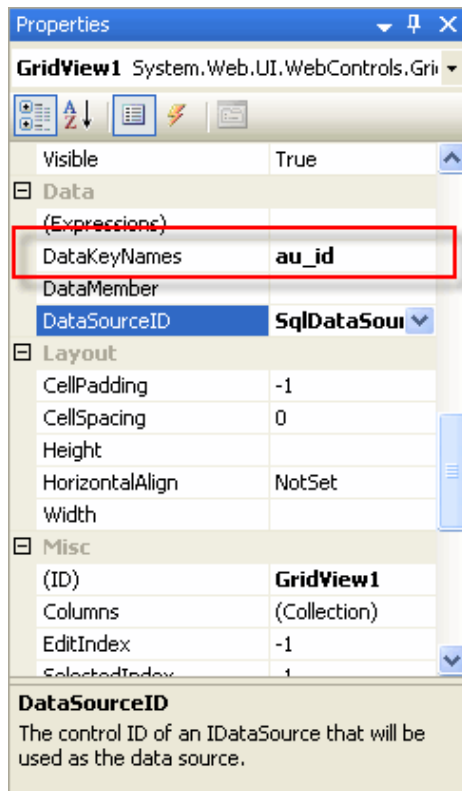
</div>
</form>
```

fijate que hemos puesto el parámetro “DataKeys” para que podamos disponer de esa variable en la ejecución de los comandos en el datasource. En este caso el índice de la fila lo necesitamos obligatoriamente para poder hacer la actualización:

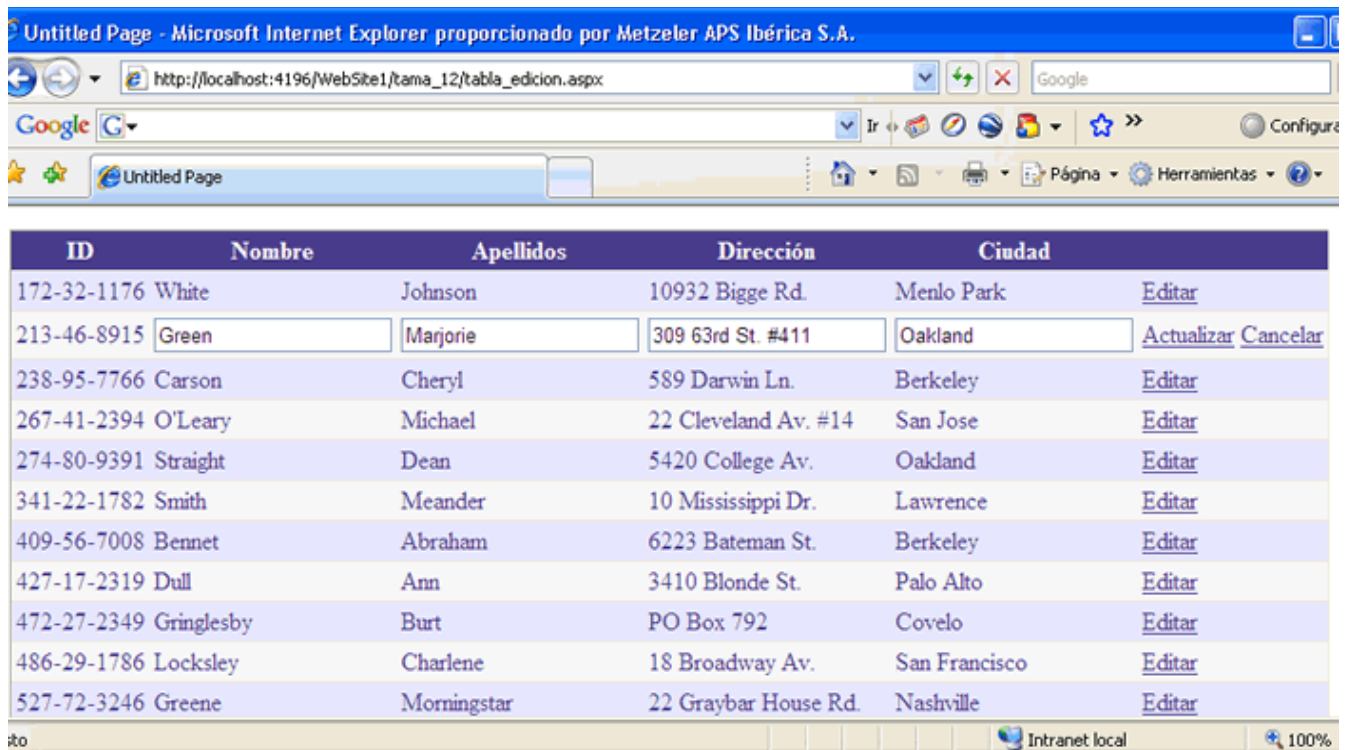
```
"Update authors Set au_lname=@au_lname,au_fname=@au_fname,
    address=@address,city=@city where au_id=@au_id"
```

Los DataKeys son imprescindibles, y son entonces las claves primarias de las tablas para así poder hacer las operaciones necesarias. Como son colecciones de la cuadrícula y no de las columnas, por eso aparecen en la parte superior, las tendremos que definir en la ventana de propiedades de la cuadrícula abajo a la derecha:

Enlace a datos. Control de cuadrícula



Sigamos, hemos puesto un botón de selección para que nos permita editar la fila. En el datasource tenemos el comando para “SelectCommand” que va a ser nuestra consulta. Ahora ejecutaremos la página:



Le he puesto un autoformato para que se vea mejor. Al pulsar en editar cambia la fila para rellenarse con cuadros de texto y así que el usuario pueda editar la fila para grabar. Fíjate a la derecha que donde había un

Enlace a datos. Control de cuadrícula

botón de “Editar” ahora es para “Actualizar” o “Cancelar”, y además, como la primera columna es el índice y no debe modificarse y le pusimos de solo lectura, no permite editarse.

Pero le falta algo... si te fijas en el origen de dato que le hemos puesto:

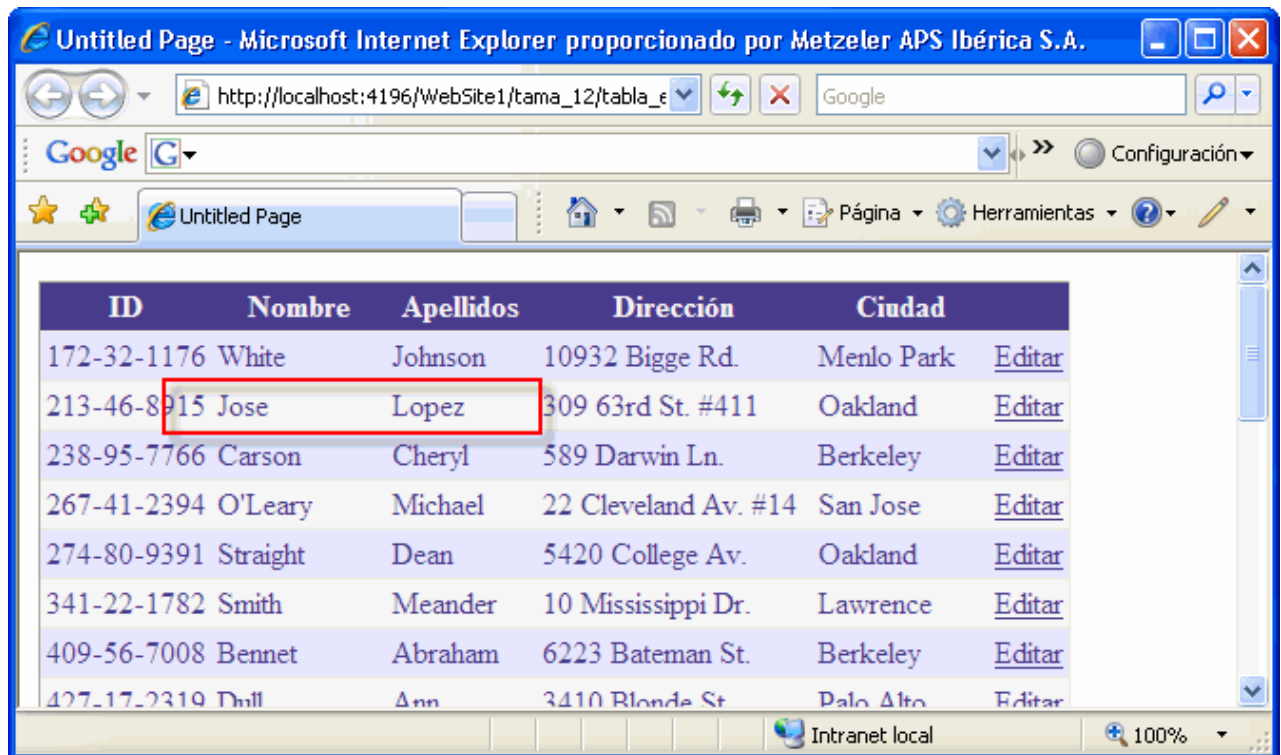
```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= ConnectionStrings:Conexion_Pubs %>"
    SelectCommand="SELECT [au_lname], [au_fname], [address], [city],
</asp:SqlDataSource>
```

Por un lado está la cadena de conexión y por otro lo que tendrá ejecutar en el caso de una selección: “SelectCommand”. Así que tendremos que añadirle código para que nos ponga lo que queremos realizar en un comando de actualizar: “UpdateCommand”, así que si lo ponemos a mano será:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= ConnectionStrings:Conexion_Pubs %>"
    SelectCommand="SELECT [au_lname], [au_fname], [address], [city], [au_id] FROM [authors]"
    UpdateCommand="Update authors Set au_lname=@au_lname, au_fname=@au_fname,
        address=@address, city=@city where au_id=@au_id ">
</asp:SqlDataSource>
```

Que es una instrucción SQL estándar para que haga la actualización. Estas instrucciones son el precio que tenemos que pagar porque nos ofrezca todas las funciones sin tener que escribir luego ninguna línea de código.

Si ejecutas la página ahora te funcionará correctamente y te permitirá actualizar las filas:



ID	Nombre	Apellidos	Dirección	Ciudad	
172-32-1176	White	Johnson	10932 Bigge Rd.	Menlo Park	Editar
213-46-8915	Jose	Lopez	309 63rd St. #411	Oakland	Editar
238-95-7766	Carson	Cheryl	589 Darwin Ln.	Berkeley	Editar
267-41-2394	O'Leary	Michael	22 Cleveland Av. #14	San Jose	Editar
274-80-9391	Straight	Dean	5420 College Av.	Oakland	Editar
341-22-1782	Smith	Meander	10 Mississippi Dr.	Lawrence	Editar
409-56-7008	Bennet	Abraham	6223 Bateman St.	Berkeley	Editar
427-17-2319	Thill	Ann	3410 Blonde St.	Palo Alto	Editar

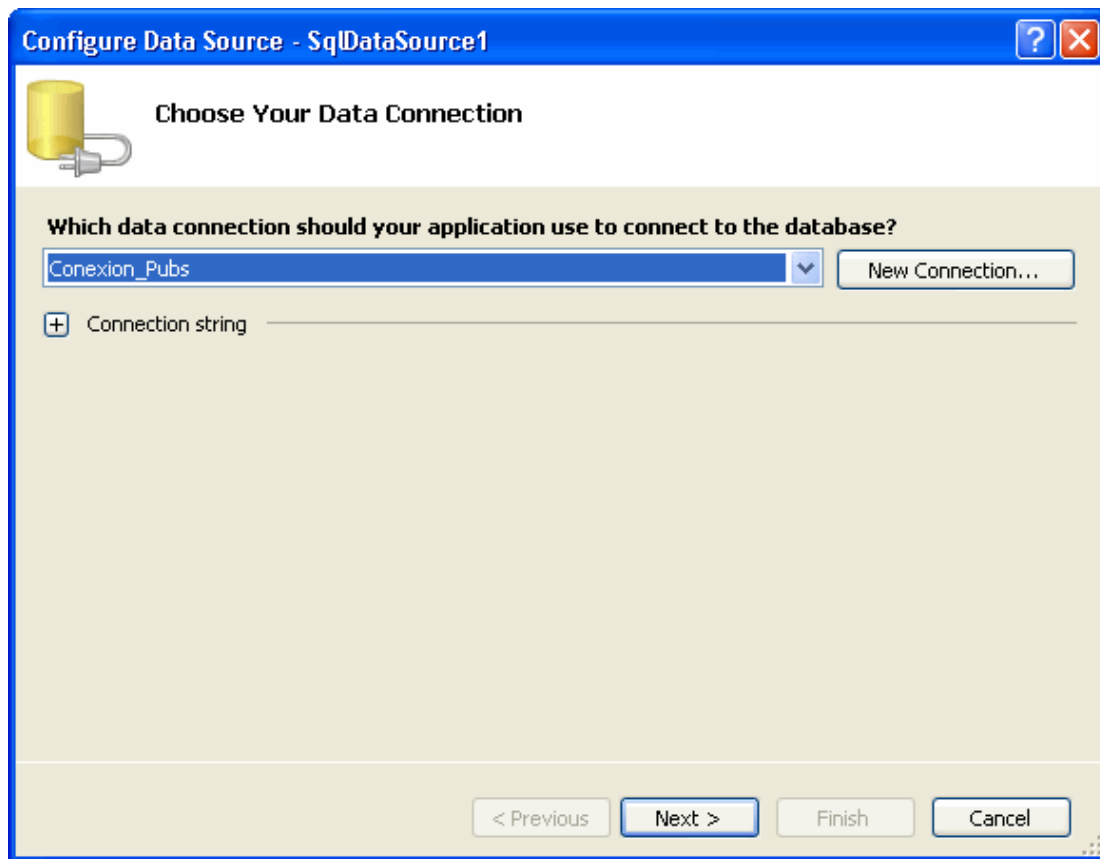
Nota. Nota, si recibes un error que dice “Must declare the scalar variable @productid” lo mas seguro es que no has establecido la propiedad GridView.DataKeyName.

Enlace a datos. Control de cuadrícula

Recuerda que no necesitamos definir los parámetros de actualización habiendo tenido el cuidado de haber puesto los mismos nombres que los campos “@campo”.


Cuando hacemos clic cambia a modo de edición y todos los campos se cambian a cuadros de texto con excepción de los campos de solo lectura que no son editables y los de tipo true/false que se muestran como casillas de verificación. Y además nos muestra dos botones adicionales para “Actualizar” o “Cancelar”. Si pulsamos éste último la fila vuelve a su estado inicial. Si pulsamos en actualizar se pasan los valores a la colección “SqlDataSource.UpdateParameters” como los nombres de los campos y activa el método “SqlDataSource.Update()” para aplicar los cambios a la base de datos. Otra vez hemos conseguido realizar una operación sin escribir código.

Si queremos poner estas instrucciones de consulta y actualización con el editor, sería de esta forma. Pulsamos en configurar el datasource:



Indicamos que queremos ejecutar comandos. En las consultas indicábamos la tabla pero ahora trabajamos con comandos:

Configure Data Source - SqlDataSource1

 **Configure the Select Statement**

How would you like to retrieve data from your database?

☒ Specify a custom SQL statement or stored procedure

☐ Specify columns from a table or view

Name:
authors

Columns:

<input type="checkbox"/> *	<input type="checkbox"/> city
<input type="checkbox"/> au_id	<input type="checkbox"/> state
<input type="checkbox"/> au_lname	<input type="checkbox"/> zip
<input type="checkbox"/> au_fname	<input type="checkbox"/> contract
<input type="checkbox"/> phone	
<input type="checkbox"/> address	

☐ Return only unique rows

WHERE...

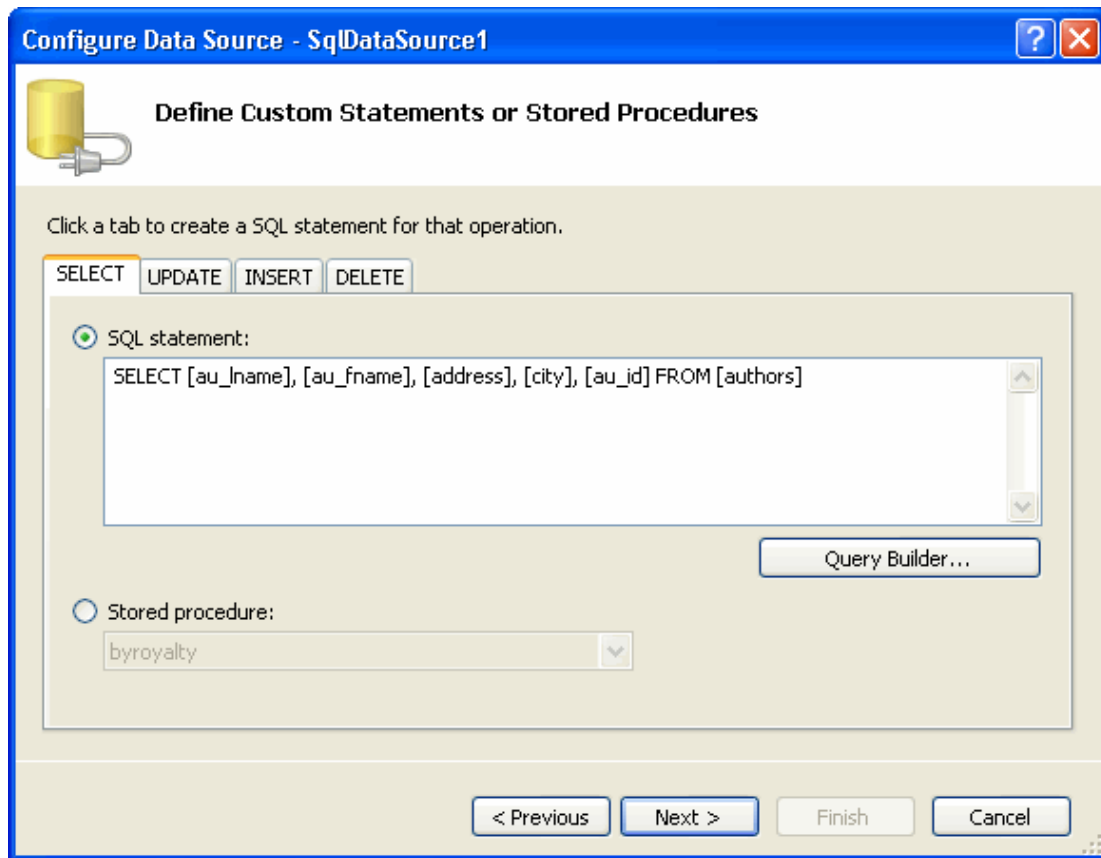
ORDER BY...

Advanced...

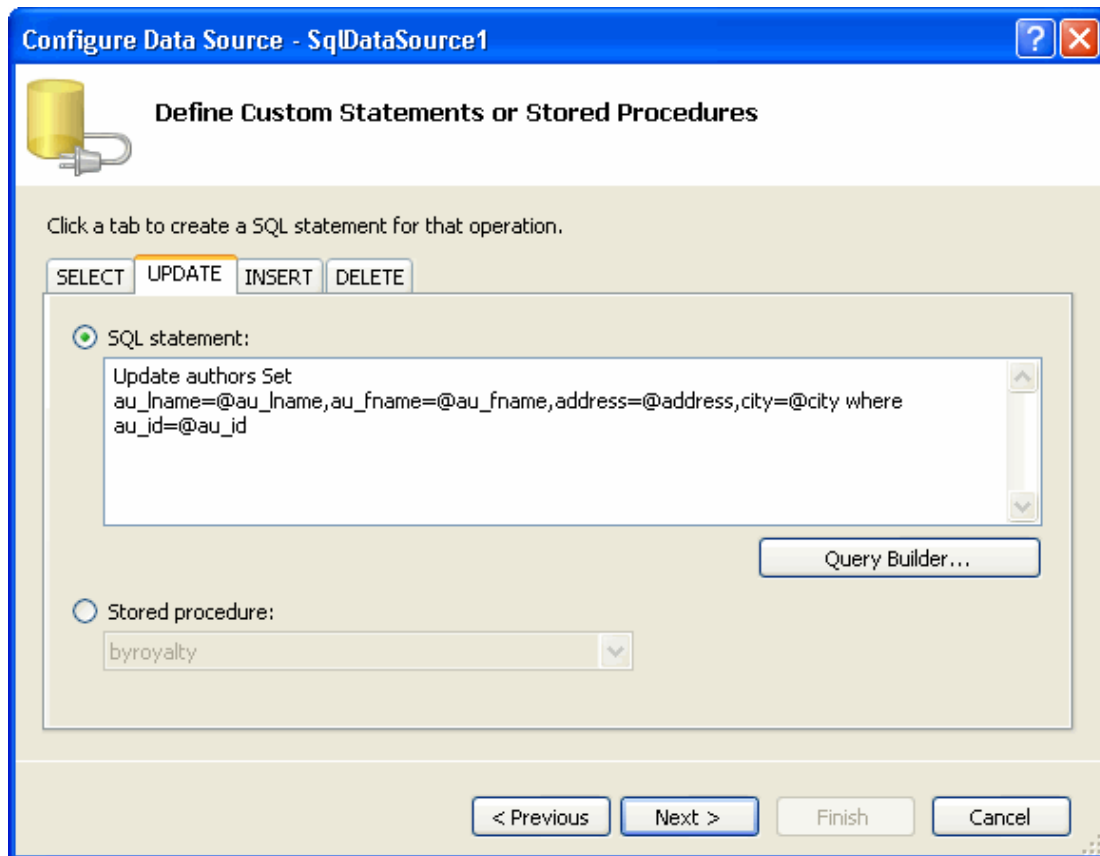
SELECT statement:

< Previous Next > Finish Cancel

Vemos la sentencia para el Select, es decir, para las consultas:



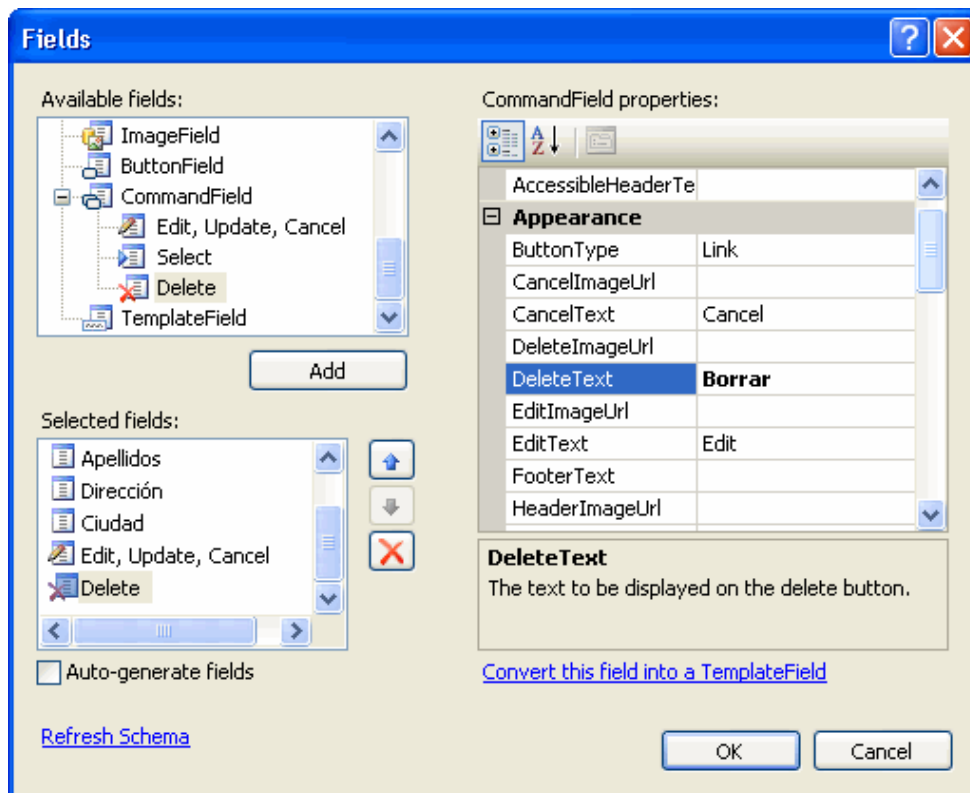
Y la que hemos escrito para la actualización:



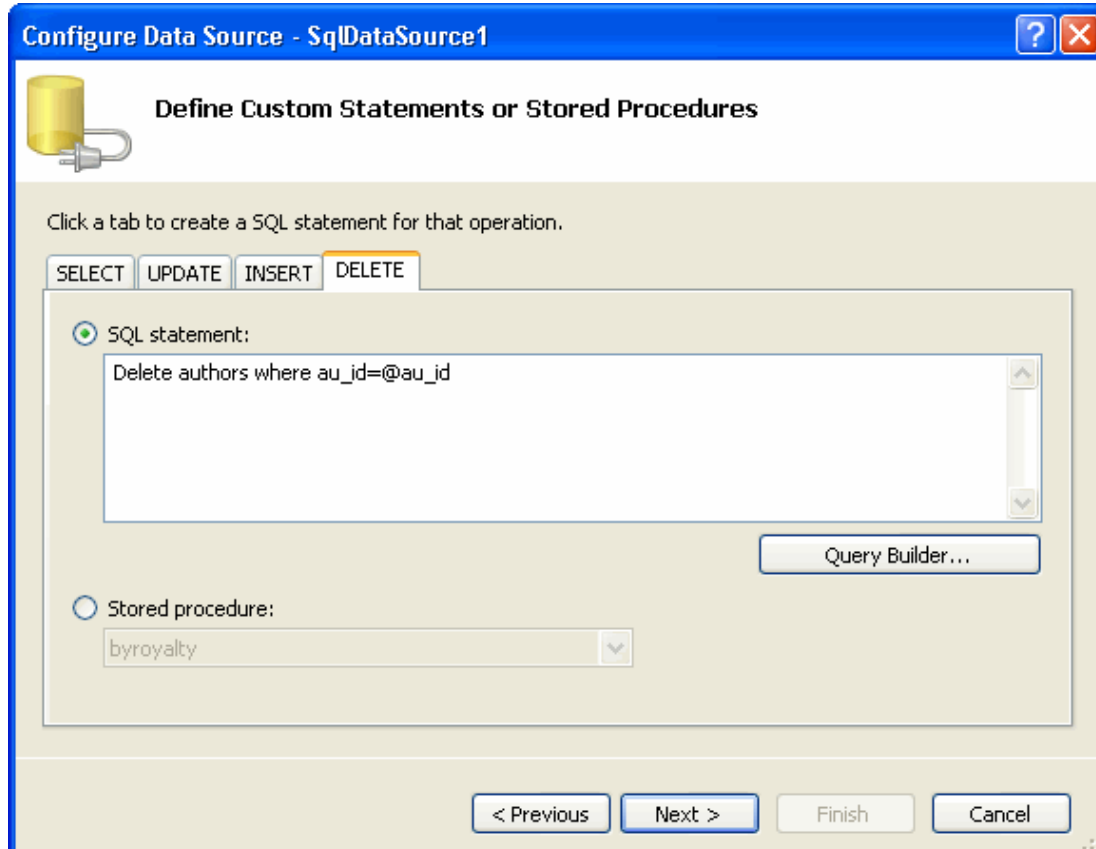
Y con esto bastaría.

Para el borrado de registros haremos parecido. Afladiremos una columna nueva de tipo Delete en el editor o si la escribimos a mano con la propiedad "ShowDeleteButton" a "true".

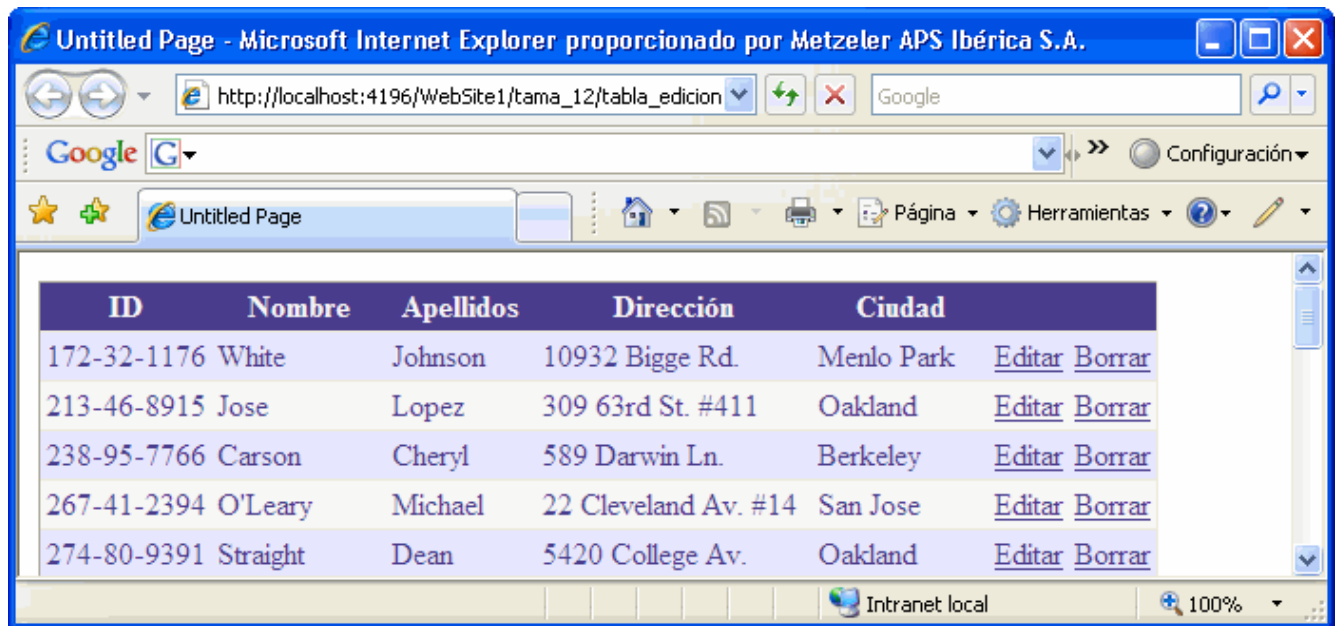
Enlace a datos. Control de cuadrícula



Y ponemos el comando en la solapa “Delete”:



La ejecutamos:



Y tendremos un nuevo botón que nos permitirá borrar la fila. Ojo, puede que te de un error por temas de diseño de esta tabla de ejemplo ya que tiene internamente definidas una serie de relaciones entre tablas.

Por último comentar que en las cuadrículas no tenemos la opción de insertar. Esto lo tendremos que hacer mostrando un registro completo con los cuadros de texto necesarios como veremos un poco mas adelante.

7. Ordenar y paginar la cuadrícula

Las cuadrículas son una forma perfecta de mostrar datos pero claro, tiene algunos pequeños inconvenientes. El mas cercano que imaginarás es cuando la página de resultados es muy larga. Pero, como no podía ser menos, tendremos la posibilidad de realizar esto de una forma rápida y sencilla que si la combinamos con la ordenación nos encontramos, una vez mas, con una potente forma de presentar los datos con muchas opciones para los usuarios.

Las dos operaciones se realizarán, por supuesto, en el servidor se base de datos que ejecutará la clausula de ordenación "Order by" adecuada.

7.1 Ordenación

La forma mas sencilla de permitir al usuario ordenar los resultados es haciendo clic en el encabezado de la columna. Para habilitar la ordenación estableceremos la propiedad "GridView.AllowSorting" a "true", después definiremos una "SortExpression" para cada columna que pueda ordenarse, que debería ser para construir correctamente la sentencia SQL. Veamos ya un ejemplo:

```
<asp:BoundField DataField="au_lname" HeaderText="Nombre" SortExpression="au_lname" />
```

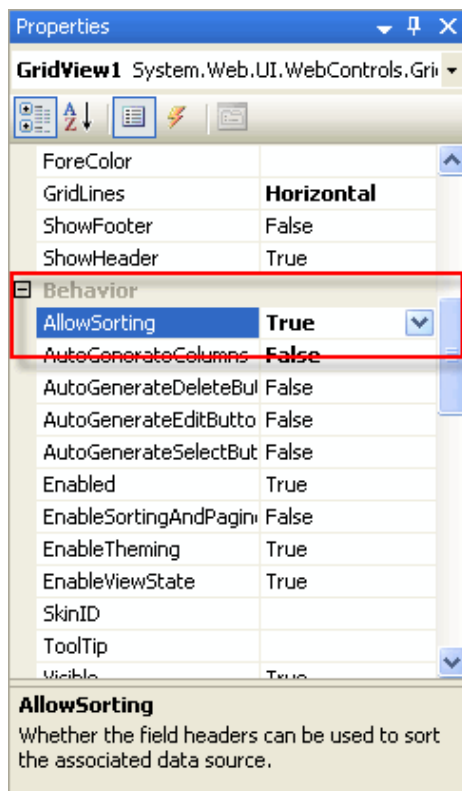
Enlace a datos. Control de cuadrícula

Como ves, hemos tenido la precaución de ponerle en la propiedad de la expresión de ordenación el nombre del campo del origen de datos.

Una vez que hemos asociado una expresión de ordenación con la columna y establecida la propiedad “AllowSorting” a true, la cuadrícula generará los encabezados como hipervínculos para que podamos hacer clic sobre ellos y así poder ordenarlos.

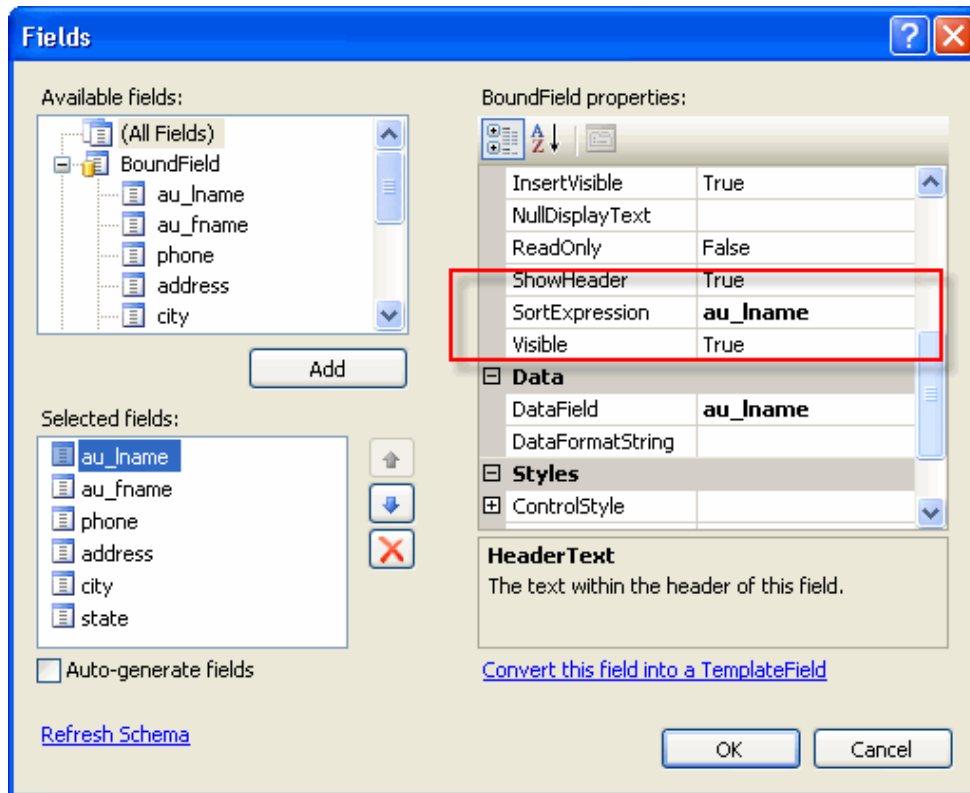
Hay que tener en cuenta que no todos los orígenes de datos permiten la ordenación., recuerda que dijimos que por defecto SqlDataSource tiene su propiedad DataSourceMode establecida a “DataSet” que sí permite este tipo de operaciones. Los DataReaders, mas sencillos y rápidos no permiten ni paginación ni ordenación.

Así que por un lado activamos en las propiedades de la cuadrícula que se permite la ordenación. Crea una nueva página con una cuadrícula enlazada con el origen de datos que quieras y en la propiedades de la cuadrícula:



Y ahora en las columnas que queremos ordenación le pondremos exactamente el nombre del campo de la base de datos:

Enlace a datos. Control de cuadrícula



Ejecutamos la página y:

Untitled Page - Microsoft Internet Explorer proporcionado por Metzeler APS Ibéri...

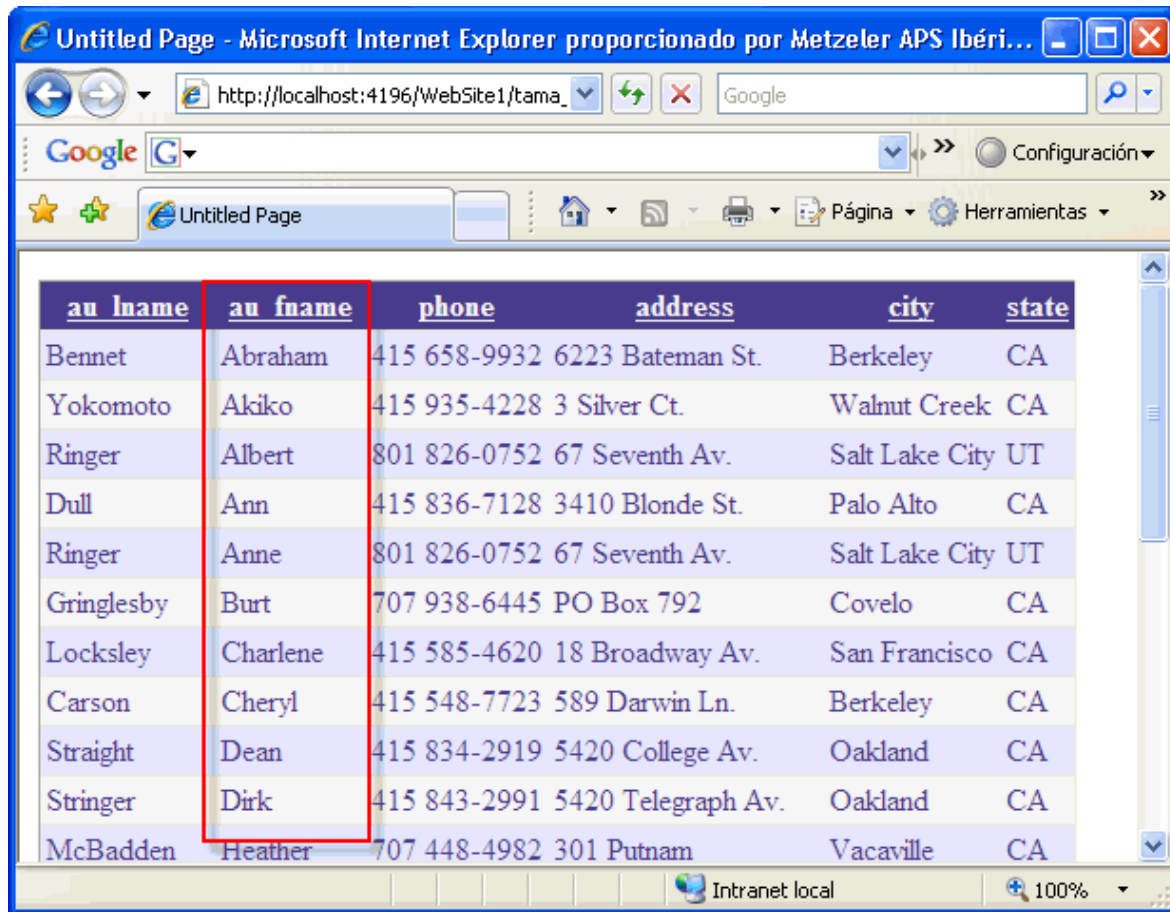
http://localhost:4196/WebSite1/tama_ Google

Google Configuración

Untitled Page Página Herramientas

<u>au lname</u>	<u>au fname</u>	<u>phone</u>	<u>address</u>	<u>city</u>	<u>state</u>
Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA
Yokomoto	Akiko	415 935-4228	3 Silver Ct.	Walnut Creek	CA
Ringer	Albert	801 826-0752	67 Seventh Av.	Salt Lake City	UT
Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA
Ringer	Anne	801 826-0752	67 Seventh Av.	Salt Lake City	UT
Gringlesby	Burt	707 938-6445	PO Box 792	Covelo	CA
Locksley	Charlene	415 585-4620	18 Broadway Av.	San Francisco	CA
Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA
Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA
Stringer	Dirk	415 843-2991	5420 Telegraph Av.	Oakland	CA
McBadden	Heather	707 448-4982	301 Putnam	Vacaville	CA

Intranet local 100%



<u>au lname</u>	<u>au fname</u>	<u>phone</u>	<u>address</u>	<u>city</u>	<u>state</u>
Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA
Yokomoto	Akiko	415 935-4228	3 Silver Ct.	Walnut Creek	CA
Ringer	Albert	801 826-0752	67 Seventh Av.	Salt Lake City	UT
Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA
Ringer	Anne	801 826-0752	67 Seventh Av.	Salt Lake City	UT
Gringlesby	Burt	707 938-6445	PO Box 792	Covelo	CA
Locksley	Charlene	415 585-4620	18 Broadway Av.	San Francisco	CA
Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA
Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA
Stringer	Dirk	415 843-2991	5420 Telegraph Av.	Oakland	CA
McBadden	Heather	707 448-4982	301 Putnam	Vacaville	CA

7.2 Ordenar y seleccionar.

Si queremos utilizar estas dos operaciones a la vez debemos tener en cuenta un detalle. Vamos a ver primero que problema nos puede generar. Seleccionamos una fila y luego le damos a ordenar por una columna. Activa la selección de filas en la cuadrícula, selecciona una:

Enlace a datos. Control de cuadrícula

Untitled Page - Microsoft Internet Explorer proporcionado por Metzeler APS Ibérica S.A.

http://localhost:4196/WebSite1/tama_12/tabla_ordenaci

Google

Google

Untitled Page

<u>au lname</u>	<u>au fname</u>	<u>phone</u>	<u>address</u>	<u>city</u>	<u>state</u>	
White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	Seleccionar
Jose	Lopez	415 986-7020	309 63rd St. #411	Oakland	CA	Seleccionar
Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	Seleccionar
O'Leary	Michael	408 286-2428	22 Cleveland Av. #14	San Jose	CA	Seleccionar
Straight	Dean	415 834-2919	5420 College Av.	Oakland	CA	Seleccionar
Smith	Meander	913 843-0462	10 Mississippi Dr.	Lawrence	KS	Seleccionar
Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA	Seleccionar
Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA	Seleccionar

Intranet local 100%

Y ahora pulsa en ordenar por una columna:

Untitled Page - Microsoft Internet Explorer proporcionado por Metzeler APS Ibérica S.A.

http://localhost:4196/WebSite1/tama_12/tabla_ordenaci

Google

Google

Untitled Page

<u>au lname</u>	<u>au fname</u>	<u>phone</u>	<u>address</u>	<u>city</u>	<u>state</u>	
Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley	CA	Seleccionar
Yokomoto	Akiko	415 935-4228	3 Silver Ct.	Walnut Creek	CA	Seleccionar
Ringer	Albert	801 826-0752	67 Seventh Av.	Salt Lake City	UT	Seleccionar
Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto	CA	Seleccionar
Ringer	Anne	801 826-0752	67 Seventh Av.	Salt Lake City	UT	Seleccionar
Gringlesby	Burt	707 938-6445	PO Box 792	Covelo	CA	Seleccionar
Locksley	Charlene	415 585-4620	18 Broadway Av.	San Francisco	CA	Seleccionar
Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	Seleccionar

javascript:__doPostBack('GridView1','Sort\$au_fname')

Intranet local 100%

Como ves sigue seleccionada la tercera fila aunque le hayamos pulsado para ordenar. Está mal hecho ya que

Enlace a datos. Control de cuadrícula

me ha seleccionado una nueva fila que tiene el mismo índice (posición en pantallas) que la anterior. Es decir, si seleccionamos la segunda fila y ordenamos, seguiremos con la segunda fila seleccionada pero de la nueva lista de elementos ordenadas. Esto solo lo podemos resolver mediante programación cambiando la selección cada vez que se haga clic en la columna. Tranquilo, sabemos hacer todos los pasos. Primero quitaremos la selección realizada por el usuario cuando se detecte el evento de clic en la columna para ordenación “gridview1_Sorted”:

```
Protected Sub GridView1_Sorted(ByVal sender As Object, ByVal e As System.EventArgs) H
    GridView1.SelectedIndex = -1
End Sub
```

Con esto conseguimos quitar la selección del usuario y por lo menos queda coherente. Pero en algunas ocasiones seguramente queramos asegurarnos de que queda seleccionada la fila cuando cambia la ordenación. La idea es almacenar el valor del campo clave en el “view state” cada vez que cambie ese índice, es decir:

```
Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.I
    If GridView1.SelectedIndex <> -1 Then
        ViewState("au_id") = GridView1.SelectedValue.ToString()
    End If
End Sub
```

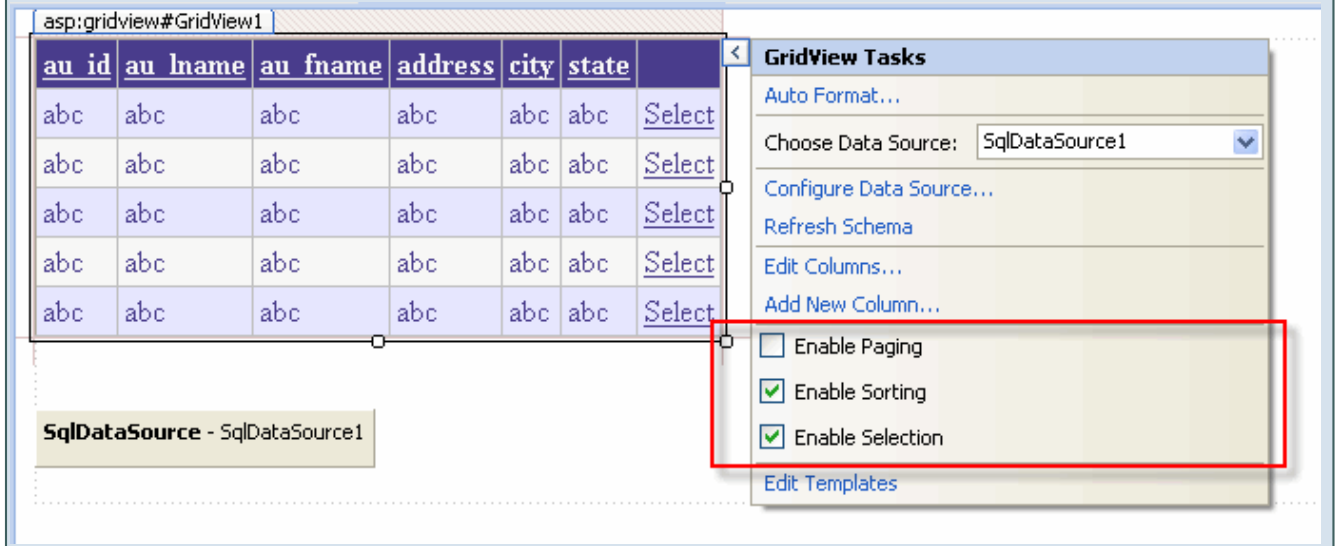
Es decir, cuando se haya seleccionado algún elemento almacenamos en una variable de estado el valor del índice. Esta columna no la habia seleccionado antes y la necesitamos para este truco así que reconstruye la página para que la incorpore. Lo mejor es quitarlo todo y volver a ponerlo, aunque si lo haces a mano, recuerda exportar la clave “au_id” con la propiedad --> DataKeyNames=“au_id”

Ahora, cuando se enlaza la cuadrícula con el origen de datos (evento DataBound), por ejemplo por una ordenación, podemos volver a aplicar la selección al elemento almacenado anteriormente:

```
Protected Sub GridView1_DataBound(ByVal sender As Object, ByVal e As System.EventArgs)
    'Comprobamos que existe el valor exportado por el Datagrid:
    If ViewState("au_id") IsNot Nothing Then
        'Creamos una variable y le asignamos el valor del índice
        Dim valor_seleccionado As String = CType(ViewState("au_id"), String)
        ' Exploramos la líneas hasta que coincida
        For Each fila As GridViewRow In GridView1.Rows
            Dim valor_clave As String
            valor_clave = GridView1.DataKeys(fila.RowIndex).Value.ToString()
            'La fila que coincida la seleccionamos:
            If valor_clave = valor_seleccionado Then
                GridView1.SelectedIndex = fila.RowIndex
                Return
            End If
        Next
    End If
End Sub
```

Ten en cuenta que ahora vamos a ver la paginación y esto nos puede complicar un poco la cosa. Date cuenta que la ordenación nos va a mover las filas de sitio, seguramente a otra página. Es decir debería estar visible la selección pero en otra página, pero tranquilo, algo podremos hacer...

Nota. Una forma rápida de habilitar la selección, ordenación y paginación es desde aquí:



7.3 Paginación

Esta es una de las opciones mas útiles y potentes de este control ya que es normal que nos encontremos con grandes conjuntos de resultados que no nos van a caber en una página de un tamaño normal. Siempre tendremos que cuidar este detalle, el de no permitir que el usuario pueda realizar una consulta de algo muy grandes, con el consiguiente problema para el servidor, por tener que construir una página con centenares de filas y el cliente porque obtendrá una página Web lenta e inmanejable.

El control de cuadrícula nos va a realizar esta operación de forma totalmente automática. Realizará la consulta en un Dataset obligatoriamente y colocará grupos de filas, por ejemplo 20 en pantalla con la posibilidad de navegar entre las páginas totales del resultado. La consulta se hace en el servidor y es el Datagrid el que extraerá sólo la página necesaria. Veamos que propiedades tenemos con esta opción de paginación.

Propiedad	Descripción
AllowPaging	Habilita o deshabilita la paginación, por defecto es "false"
Pagesize	Obtiene o establece el número de filas que se muestran
PageIndex	Obtiene o establece el índice la página que se muestra actualmente
PagerSettings	Proporciona formato al objeto "pagersettings". Define como se muestran los controles de paginación: textos, gráficos, personalizado...
PageStyle	Proporciona un objeto de estilo para configurar los tipos de letra, colores y alineación de texto para los controles de paginación
Eventos PageIndexChanging y PageIndexChanged	Se producen cuando seleccionamos las paginaciones, antes y después de cambiar PageIndex

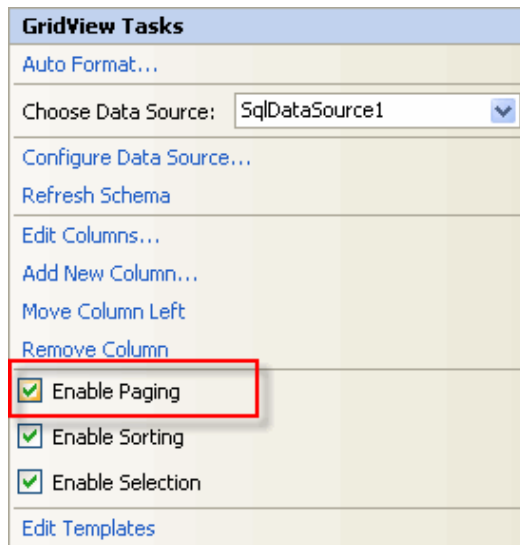
Para activar esta imprescindible opción de la paginación sólo tendremos que poner la propiedad "AllowPaging" a "true" e indicar en "PageSize" las filas que queremos mostrar en cada página. Por ejemplo:

```
<asp:GridView ID="GridView1" runat="server" DataSourceID="sourceProducts"
    PageSize="10" AllowPaging="True" ...>
...
</asp:GridView>
```

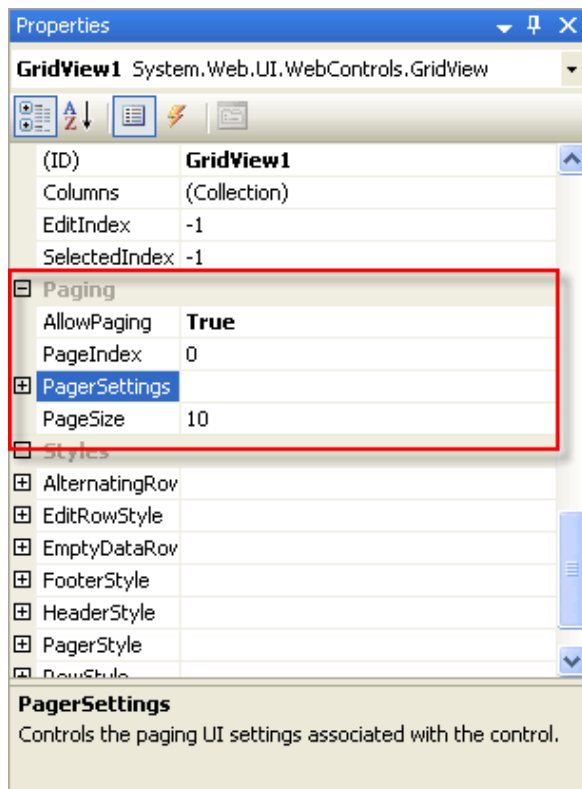
Enlace a datos. Control de cuadrícula

```
</asp:GridView>
```

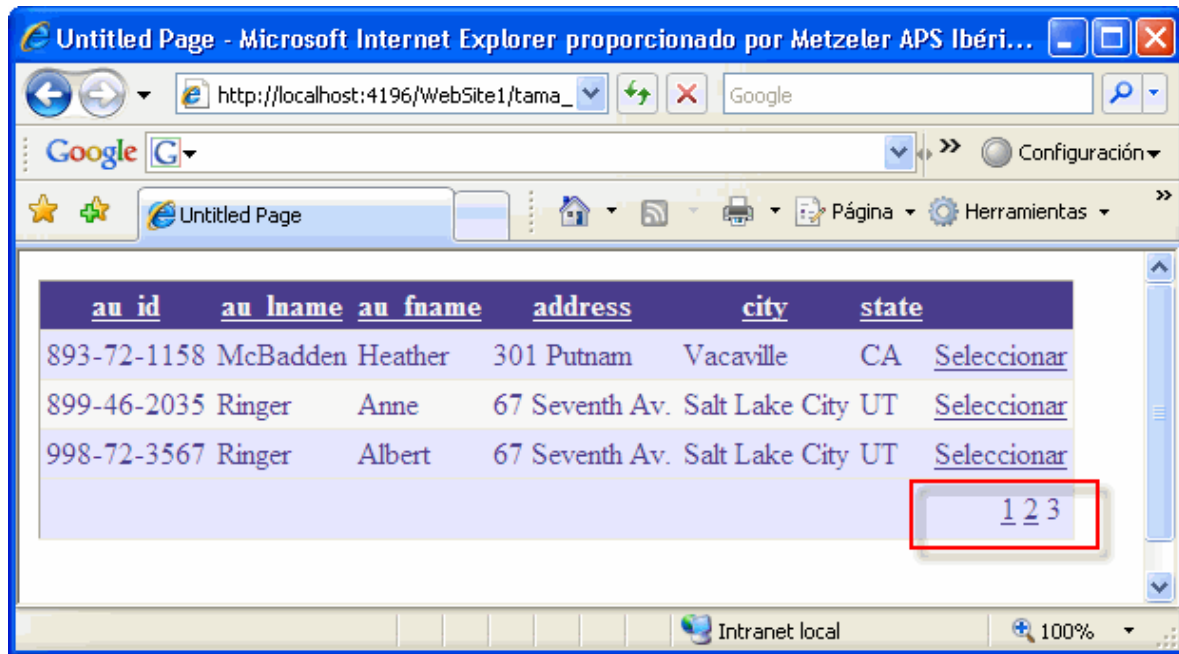
Veamos un ejemplo. Copia el ejemplo anterior en otra página y activa las propiedades para la paginación desde el IDE:



Pero nos faltan detalles así que miramos las propiedades del Datagrid referentes a la paginación:



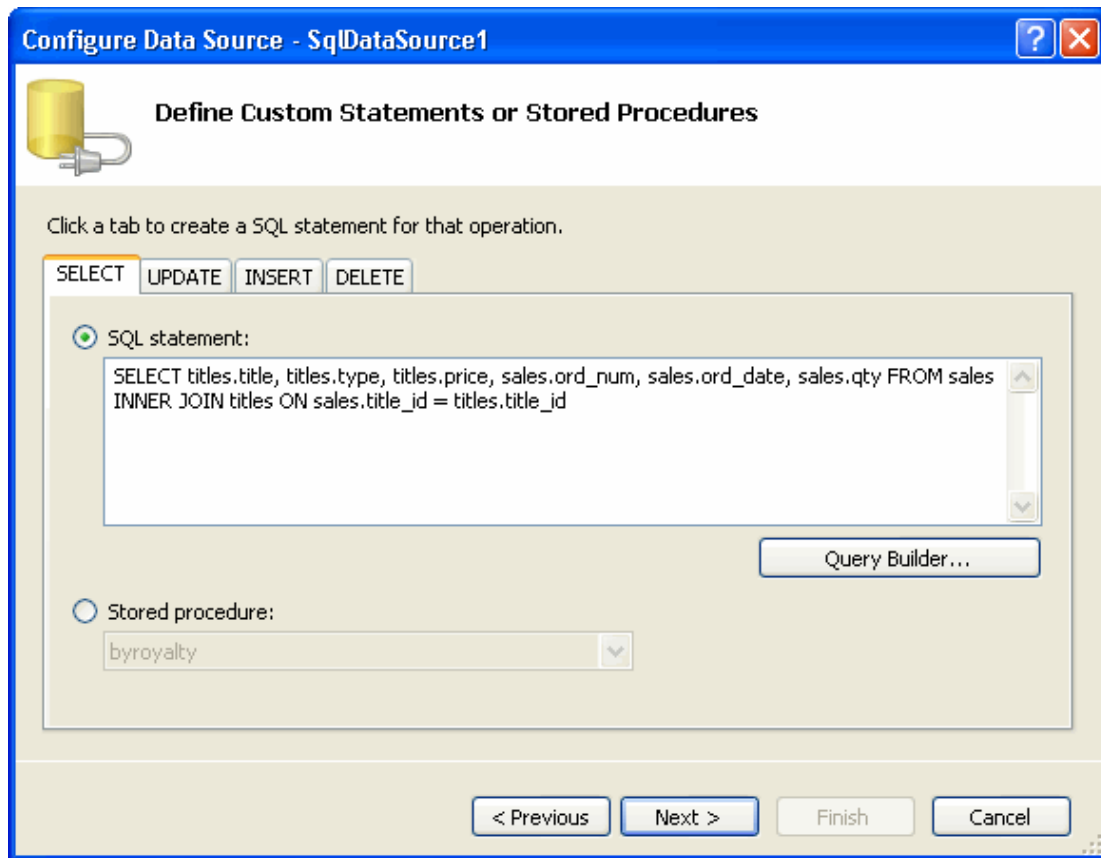
Nos producirá una página con 10 registros por página y nos indica además cuantas páginas de datos tenemos:



En cuanto al rendimiento normalmente es bueno pero en cada petición el servidor realiza la consulta la base de datos y luego se extrae la página adecuada. Hay una técnica avanzada llamada “Caching” que permite que esa consulta se quede en el servidor almacenada temporalmente así la paginación iría muy rápida porque no hay que volver a consultar la base de datos entera en cada paginación. Esta técnica la puedes ver en el curso de ASP.NET avanzado.

8. Utilizar plantillas

Hasta hemos utilizado formatos básicos para presentar datos en nuestras tablas. Si queremos hacer cosas avanzadas como poner varios valores en la celda o personalizar su contenido tendremos que utilizar las plantillas: “TemplateField”. Nos van a permitir definir una plantilla para cada columna en la que pondremos las etiquetas y elementos HTML a nuestro gusto. Aquí tendremos libertad absoluta...Por ejemplo imagina que queremos crear una columna que sea la combinación de dos de ellas, construiremos una plantilla “ItemTemplate”. Crea una página nueva con esta consulta:



Hemos cruzado la tabla de libros con la de pedidos, puedes utilizar el “Query Builder” para realizarlo:

Enlace a datos. Control de cuadrícula

Query Builder

sales

- * (All Columns)
- ☐ stor_id
- ☒ ord_num
- ☒ ord_date
- ☒ qty

titles

- * (All Columns)
- ☐ title_id
- ☒ title
- ☒ type
- ☐ pub_id

Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or...
title		titles	<input checked="" type="checkbox"/>				
type		titles	<input checked="" type="checkbox"/>				
price		titles	<input checked="" type="checkbox"/>				
ord_num		sales	<input checked="" type="checkbox"/>				
ord_date		sales	<input checked="" type="checkbox"/>				
qty		sales	<input checked="" type="checkbox"/>				
			<input type="checkbox"/>				
			<input type="checkbox"/>				

```

SELECT titles.title, titles.type, titles.price, sales.ord_num, sales.ord_date, sales.qty
FROM sales INNER JOIN
      titles ON sales.title_id = titles.title_id
  
```

title	type	price	ord_num	ord_date	qty
The Busy Execut...	business	19,9900	6871	14/09/1994 0:0...	5
Is Anger the En...	psychology	10,9500	722a	13/09/1994 0:0...	3
Secrets of Silico...	popular_comp	20,0000	A2976	24/05/1993 0:0...	50
Is Anger the En...	psychology	10,9500	QA7442.3	13/09/1994 0:0...	75
Is Anger the En...	psychology	10,9500	D4482	14/09/1994 0:0...	10

1 of 21 | Cell is Read Only.

Execute Query OK Cancel

Queremos crear una plantilla para que nos ponga los datos del número del pedido, fecha y cantidad en la misma celda y lo pondremos en las definiciones de las columnas;

Enlace a datos. Control de cuadrícula

```
<Columns>
  <asp:BoundField DataField="title" HeaderText="title"
  <asp:BoundField DataField="type" HeaderText="type"
  <asp:BoundField DataField="price" HeaderText="price"
  <asp:TemplateField HeaderText="Pedido">
    <ItemTemplate>
      <b>Pedido:</b><br />
      Número: <%# Eval("ord_num") %><br />
      Fecha: <%# Eval("ord_date") %><br />
      Cantidad: <%# Eval("qty") %>
    </ItemTemplate>
  </asp:TemplateField>
</Columns>
```

Ya en la edición nos muestra su futuro aspecto:

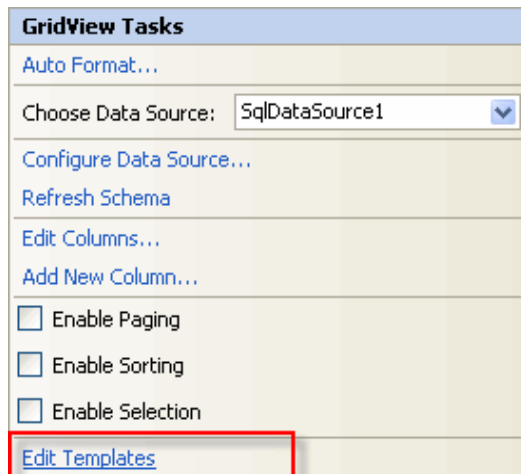
asp:GridView#GridView1			
title	type	price	Pedido
abc	abc	0	Pedido: Número: abc Fecha: 15/02/2008 0:00:00 Cantidad: 0

Con este resultado.

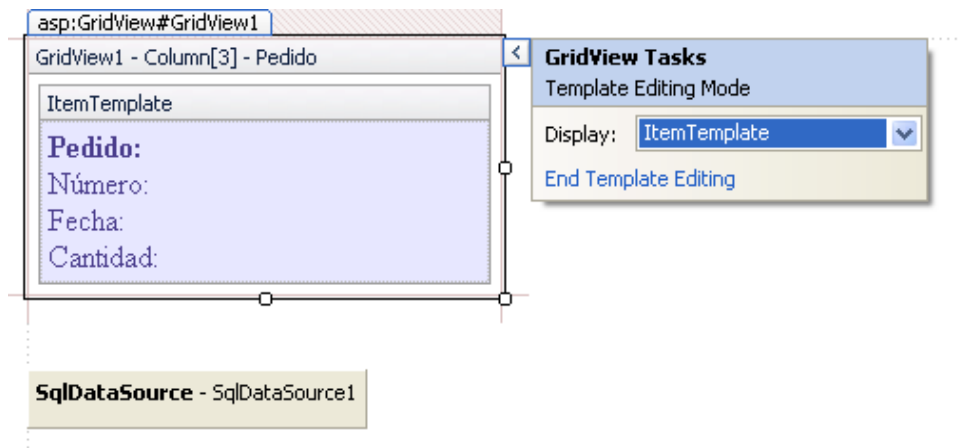
title	type	price	Pedido
Executive's Database Guide	business	19,990	Pedido: Número: 6871 Fecha: 14/09/1994 0:00:00 Cantidad: 5
Enemy?	psychology	10,950	Pedido: Número: 722a Fecha: 13/09/1994 0:00:00 Cantidad: 3

Si queremos hacerlo con el editor, seleccionaremos:

Enlace a datos. Control de cuadrícula



Y nos muestra un cuadro para poner todos los controles que queramos, en nuestro caso solo unos textos:



Pero las variables las tendremos que poner a mano en la vista HTML de la página ya que se trata de código ASP.NET incluido dentro de la página y es para extraer el valor de las filas.

Obviamente podremos trabajar con los campos que tenemos enlazados en la consulta. Si no han intervenido determinados campos en la sentencia "Select" no podremos acceder a ellos, así que habría que ampliar la consulta.

Para crear el enlace de datos la plantilla utilizamos el método "Eval()" que es un método compartido de la clase "System.Web.UI.DataBinder". Es una llamada imprescindible y nos devolverá automáticamente el dato del elemento de la fila actual.

Nota: El método Eval() es muy útil porque nos permite dar formato al texto en el momento. Para esto simplemente le haremos una llamada pero con un parámetro (está sobrecargado) con el formato adecuado:

```
<%# Eval("Fecha_nacimiento", "{0:dd/MM/yy}") %>
```

Te habrás dado cuenta que en el ejemplo hemos utilizado tres enlaces para obtener la información de la fila. Y el resto del código ha sido texto estático, etiquetas y controles. Necesitaremos asegurarnos de que el origen de datos nos proporciona estos datos, de lo contrario nos daría un error en tiempo de ejecución. Si recuperamos esa información antes evitaremos problemas aunque con un poco de cuidado no los deberíamos tener.

Enlace a datos. Control de cuadrícula

Cuando enlazamos la cuadrícula ésta revisa el origen de datos para poner las columnas que recuperará. Procesa el “ItemTemplate” para cada elemento, evalúa la expresión y finalmente genera el HTML de esa celda.

Utilizar varias plantillas.

Hemos utilizado antes una sola plantilla para configurar el aspecto de los datos pero “ItemTemplate” es sólo una de las posibilidades que nos proporciona “TemplateField”. Fijate en la cantidad de plantillas que tenemos:

Propiedad	Descripción
HeaderTemplate	Define el aspecto y contenido de la celda cabecera
FooterTemplate	Define el aspecto y contenido de la celda que hace de pie de tabla, si se muestra
ItemTemplate	Define el aspecto y contenido de las celdas
AlternatingItemTemplate	Determina el aspecto de las filas impares, por ejemplo para ponerlas en gris y las pares en blanco
EditItemTemplate	Define el aspecto cuando los controles están en modo de edición
InsertItemTemplate	Define el aspecto y los controles utilizados en el modo de edición

De estos estilos el mas utilizado es el de ItemTemplate porque podemos controlar la escritura del campo. Si no utilizáramos plantillas tendríamos todas las celdas como cuadros de texto sin ningún tipo de validación. Tenemos un último control “EmptyDataTemplate” que utilizaremos cuando queramos personalizar el aspecto de datos que puedan estar en blanco.

Controlar eventos en una plantilla

Puede que necesitemos controlar algún evento de los que tenemos configurados con estas plantillas. Por ejemplo, queremos poner una imagen en la que se pueda hacer clic, es decir, queremos controlar el evento clic de una imagen que hemos puesto en una plantilla de una columna.

```
<asp:TemplateField HeaderText="Status">

    <ItemTemplate>

        <asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="img_estado.gif" />

    </ItemTemplate>

</asp:TemplateField>
```

El problema está en que si añadimos un control a la plantilla, el GridView crea varias copias del control, una por cada línea. Así que si hacemos clic en una imagen debemos saber que imagen de todas las filas hemos hecho clic. Es decir, en un formulario podemos capturar el clic de la imagen muy fácil: su evento “click”. Pero aquí va a ser una tabla con 50 filas por ejemplo, cada una con su imagen y queremos tratar el clic sobre esa imagen. Está claro que necesitaremos saber sobre que imagen se ha hecho clic. Vamos primero con la página, crea una nueva donde uno de los campos sea una imagen:

Enlace a datos. Control de cuadrícula



```

lums>
<asp:BoundField DataField="au_lname" HeaderText="Nombre" SortExpression="au_lname" />
<asp:BoundField DataField="au_fname" HeaderText="Apellidos" SortExpression="au_fname" />
<asp:BoundField DataField="city" HeaderText="Ciudad" SortExpression="city" />
<asp:TemplateField HeaderText="Mas información">
    <ItemTemplate>
        <asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="zoom_in.png"
            CommandName="estado_clic" CommandArgument='<%# Eval("au_id") %>' />
    </ItemTemplate>
</asp:TemplateField>
columns>

```

Que quedaria:

div

Nombre	Apellidos	Ciudad	Mas información
Databound	Databound	Databound	
Databound	Databound	Databound	

La forma de resolverlo es utilizando el evento adecuado de la cuadrícula, no de la imagen. El evento `GridView.RowCommand` nos vendrá perfecto para esto ya que se activa cuando se pulsa un botón en las plantillas. Por supuesto, necesitaremos todavía pasar la información al evento “RowCommand” para averiguar la fila que ha provocado el evento. El truco lo tenemos en dos propiedades que tienen todos los botones: “CommandName” y “CommandArgument”. El primero establece un nombre descriptivo para diferenciar los clic realizados en nuestro “ImageButton” de los clic en otros controles de tipo botón de la cuadrícula. El segundo nos proporciona un dato específico de la fila que podremos utilizar para identificar la fila que se ha seleccionado. Podemos proporcionar esta información utilizando una instrucción de enlace de datos, veamos como quedaria:

```

<ItemTemplate>
    <asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="zoom_in.png"
        CommandName="estado_clic" CommandArgument='<%# Eval("au_id") %>' />
</ItemTemplate>

```

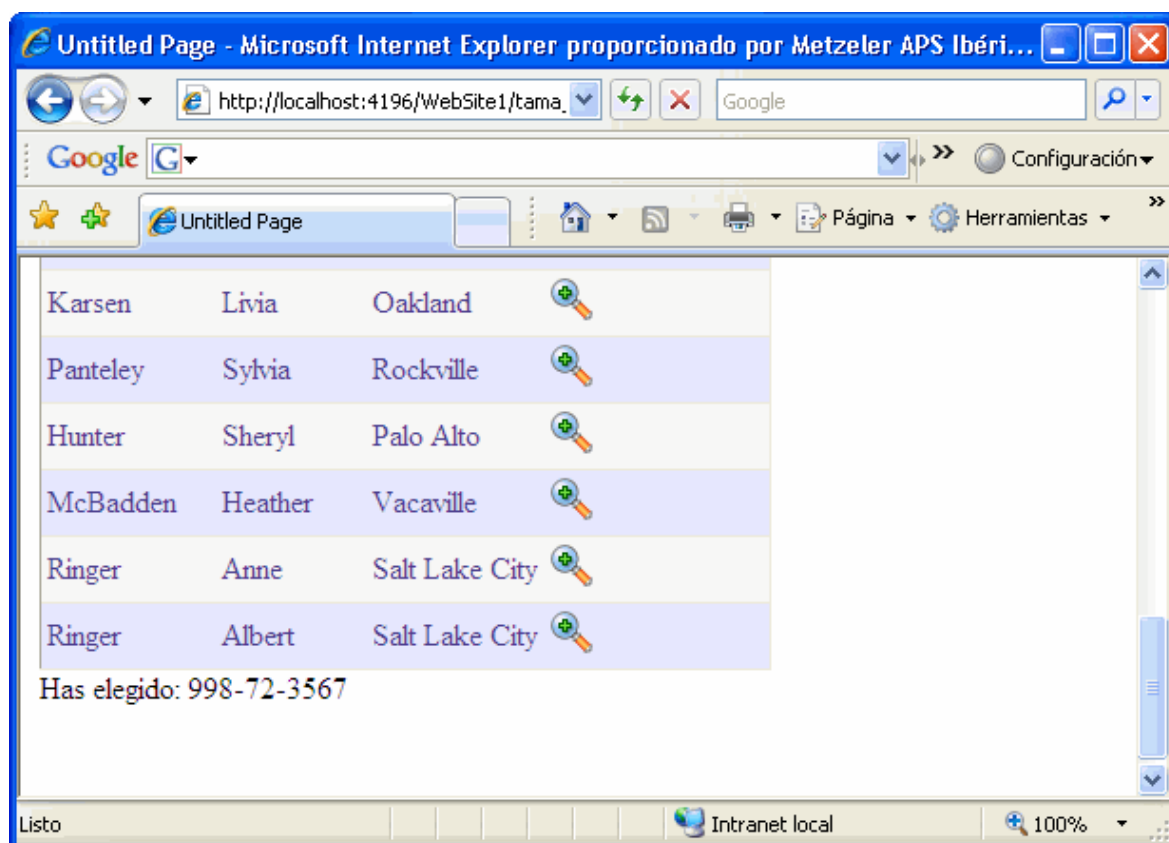
Y este seria el código necesario para responder al clic de una imagen:

```

Protected Sub GridView1_RowCommand(ByVal sender As Object, ByVal e As System.Web.UI.Web
    If e.CommandName = "estado_clic" Then
        lb_info.Text = "Has elegido: " & e.CommandArgument.ToString()
    End If
End Sub

```

La idea es que podemos poner en las plantillas lo que queramos, incluidos comandos que le pondremos el nombre que queramos con valores que luego recogemos en los eventos adecuados. Y de resultado los iconos con el valor del indice como dato asociado:



8.1 Edición con una plantilla.

Una de las razones por las que necesitaríamos utilizar plantillas es por la posibilidad de poder cambiar los controles de edición para el usuario, ya que por defecto sabes que son unos simples cuadros de texto. Este formato no va a ser siempre el mas adecuado, por ejemplo si es una ciudad podriamos mostrar un desplegable con una lista de ciudades, asi no tiene que escribir nada y sabemos que el dato es coherente. Además no disponemos de validación de los datos: muy mal para los valores numéricos.

Vamos a ver pues como podemos poner los controles que queramos utilizando “EditItemTemplate“. Puede que sea un poco laborioso pero merecerá la pena. Vamos con un ejemplo sencillo, crea una tabla con estos campos en el datasource:

Configure Data Source - SqlDataSource1

Configure the Select Statement

How would you like to retrieve data from your database?

☐ Specify a custom SQL statement or stored procedure

☒ Specify columns from a table or view

Name: titles

Columns:

<input type="checkbox"/> *	<input type="checkbox"/> advance
<input checked="" type="checkbox"/> title_id	<input checked="" type="checkbox"/> royalty
<input checked="" type="checkbox"/> title	<input type="checkbox"/> ytd_sales
<input checked="" type="checkbox"/> type	<input checked="" type="checkbox"/> notes
<input type="checkbox"/> pub_id	<input checked="" type="checkbox"/> pubdate
<input checked="" type="checkbox"/> price	

☐ Return only unique rows

WHERE...

ORDER BY...

Advanced...

SELECT statement:

SELECT [title_id], [title], [type], [price], [royalty], [notes], [pubdate] FROM [titles]

< Previous Next > Finish Cancel

Ahora vamos con una plantilla personalizada con:

```
<Columns>
  <asp:BoundField DataField="title_id" HeaderText="title_id" ReadOnly="True"
    SortExpression="title_id" />
  <asp:BoundField DataField="title" HeaderText="title" SortExpression="title" />
  <asp:BoundField DataField="type" HeaderText="type" SortExpression="type" />
  <asp:BoundField DataField="pubdate" HeaderText="pubdate" SortExpression="pubdate" />
  <asp:templatefield HeaderText="Precio">
    <itemtemplate>
      Precio: <#Eval("price") %><br />
      Royalty: <#Eval("royalty") %>
    </itemtemplate>
    <EditItemTemplate>
      <itemtemplate>
        Precio: <#Eval("price") %><br />
        Royalty: <asp:TextBox id="txt_royalty" Text='<#Bind("royalty") %>' runat="server" />
      </itemtemplate>
    </EditItemTemplate>
    <ControlStyle Width="250px" />
  </asp:templatefield>

  <asp:CommandField ShowEditButton="True" />
</Columns>
```

Nos da este resultado:

Enlace a datos. Control de cuadrícula

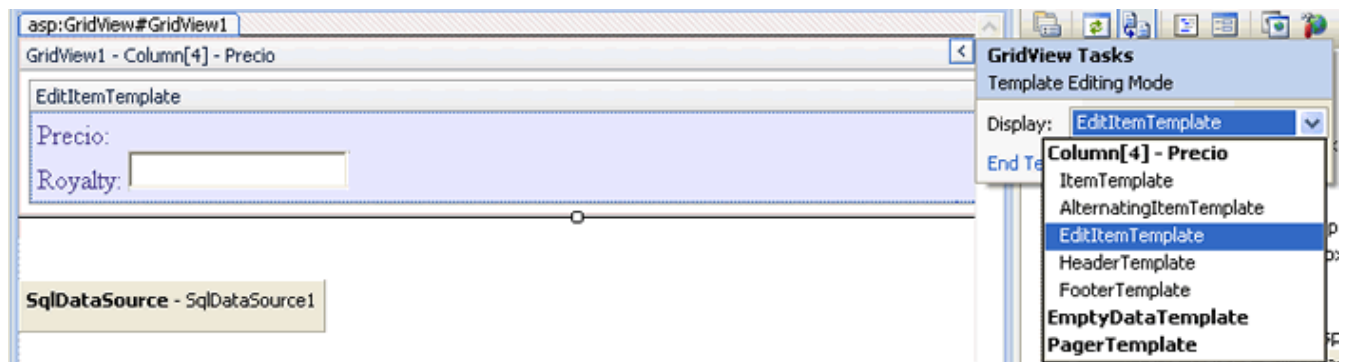
title_id	title	type	pubdate	Precio
BU1032	The Busy Executive's Dat	business	12/06/1991 0:00:00	Precio: 19,9900 Royalty: 10
	Cooking with Computers:			Precio: 11.9500

fijate en la novedad y es que hemos utilizado otra plantilla para cuando estamos en edición. En este caso además en la columna del precio solo hemos dejado que el usuario edite una de las dos variables. Cuando enlazamos un control editable debemos utilizar el método Bind() en nuestro código de enlace en lugar del habitual “Eval()”. Solo este método, el Bind() permite crear un enlace en los dos sentidos, asegurando que los valores modificados se actualicen en el servidor.

Un detalle interesante es que en el caso anterior solo se actualizarán los datos de la cuarta columna, ya que solo se podrán enviar los que tengan el método anterior. Esto es importante porque debe estar reflejado en la sentencia de Update insertada en nuestro SqlDataSource:

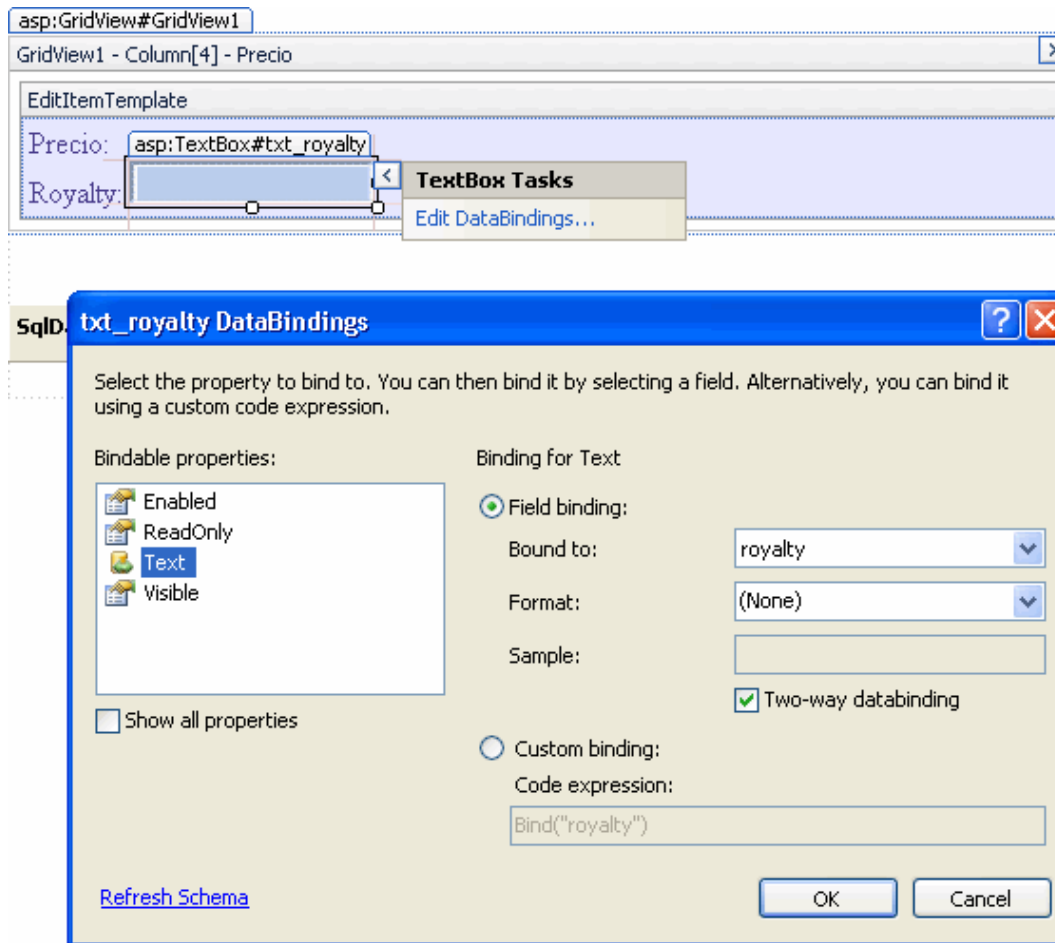
```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= ConnectionStrings:Conexion_Pubs %>"
    SelectCommand="SELECT [title_id], [title], [type], [price], [royalty], [notes], [pubdate] FROM [titles]"
    updatecommand="Update titles set title=@title,type=@type,pubdate=@pubdate
        ,royalty=@royalty FROM [titles] where title_id=@title_id" >
</asp:SqlDataSource>
```

Para editar esta plantilla, en el modo de edición debemos seleccionar que es la “template” de edición:



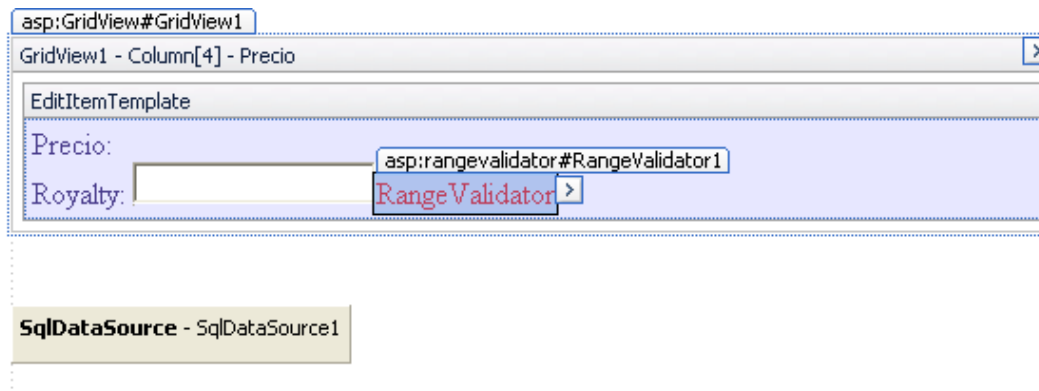
Y si te fijas, al editar el cuadro de texto, nos permite editar su enlace:

Enlace a datos. Control de cuadrícula



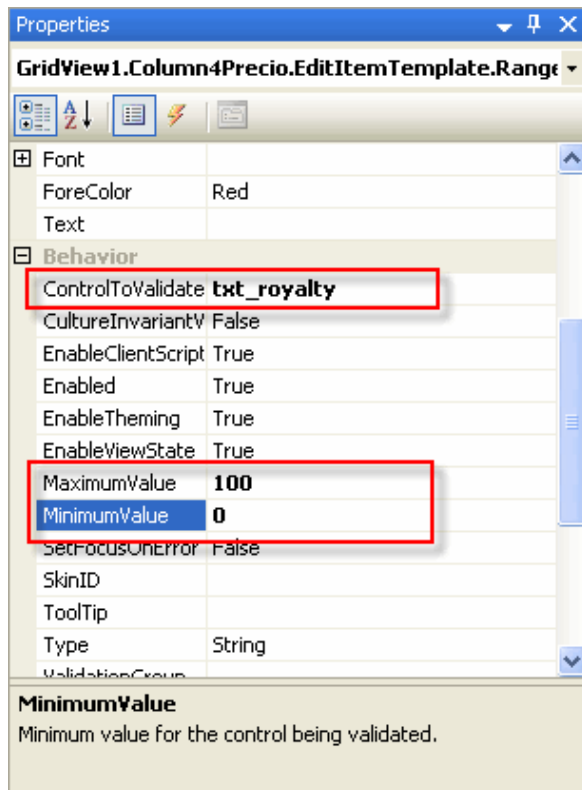
Editar con validación.

Ahora que tenemos nuestra plantilla preparada vamos a ponerle mas complementos, como los útiles validadores para asegurarnos de que el usuario introduce los valores correctos. Por ejemplo para que el usuario introduzca un valor entre 0 y 100. En modo de edición le añadimos un validador:



Enlace a datos. Control de cuadrícula

Y en sus propiedades lo enlazamos con el control y le ponemos el intervalo aceptado:



En el código quedará:

```
<EditItemTemplate>
    <itemtemplate>
        Precio: <%#Eval("price")%><br />
        Royalty: <asp:TextBox id="txt_royalty" Text='<%#Bind("royalty")%>' runat="server" />
    </itemtemplate>
    <asp:RangeValidator ID="RangeValidator1" runat="server"
        ControlToValidate="txt_royalty" ErrorMessage="RangeValidator"
        MaximumValue="100" MinimumValue="0"></asp:RangeValidator>
</EditItemTemplate>
```

Si no se valida el dato no se podrá realizar el postback para actualizar los datos.

Editar sin un columna "comand"

Hasta ahora hemos visto como utilizar un “CommandField” para que nos generase los controles de edición. Sin embargo podremos querer en ocasiones modificar este funcionamiento para afilar nuestros propios controles. Para esto solo necesitamos afilar un control de botón a la plantilla del elemento y establecer la propiedad “CommandName” a “Edit”. Esto activa automáticamente la edición de la fila, que activa los eventos para poner en edición la fila:

Enlace a datos. Control de cuadrícula

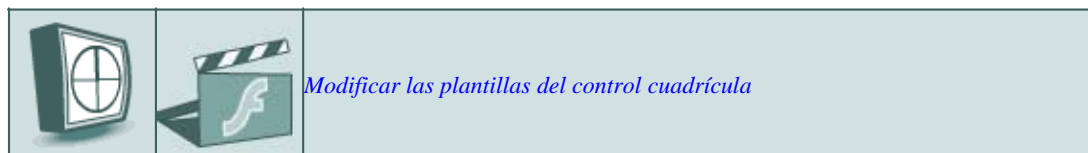
```
<itemtemplate>
    Precio: <%#Eval("price")%><br />
    Royalty: <%#Eval("royalty")%>
    <asp:LinkButton runat="server" Text="Editar" CommandName="Editar" ID="Linkbutton1" />
</itemtemplate>
```

En la plantilla de edición necesitaremos dos botones con CommandName y los valores de “Actualizar” y “Cancelar”

Update and Cancel:

```
<EditItemTemplate>
<itemtemplate>
    Precio: <%#Eval("price")%><br />
    Royalty: <asp:TextBox id="txt_royalty" Text='<%#Bind("royalty")%>' runat="server" />
    <asp:LinkButton runat="server" Text="Actualizar" CommandName="Actualizar" ID="Linkbutton1" />
    <asp:LinkButton runat="server" Text="Cancelar" CommandName="Cancelar"
        ID="Linkbutton2" CausesValidation="False" />
</itemtemplate>
```

Observa que el botón Cancelar debe tener la propiedad “CausesValidation” establecida a False para que no haga la validación. De esta forma podemos cancelar la edición incluso si el dato actual no es válido. Los eventos harán que la cuadrícula reaccione igual que si se hubieran utilizado los controles.



9. Controles "DetailView" y "FromView".

Las cuadrícula nos muestra una densa tabla con muchas filas resultantes de la consulta a la base de datos. Sin embargo en ocasiones vamos a necesitar mostrar una vista detallada del registro actual. Esto es muy normal, ya que en la tabla le mostramos unos datos genéricos y luego podemos mostrar los datos completos por separado. ASP.NET nos proporciona dos controles perfectos para estos casos: DetailView y FormView. Los dos nos muestran un sólo registro a la vez y pueden incluir botones para poder movernos por un conjunto de resultados. Los dos tienen una forma sencilla de introducir datos (por fin vamos a poder crear una fila nueva que no permitía la cuadrícula. Los dos permiten plantillas pero en el caso del FromView la requiere obligatoriamente.

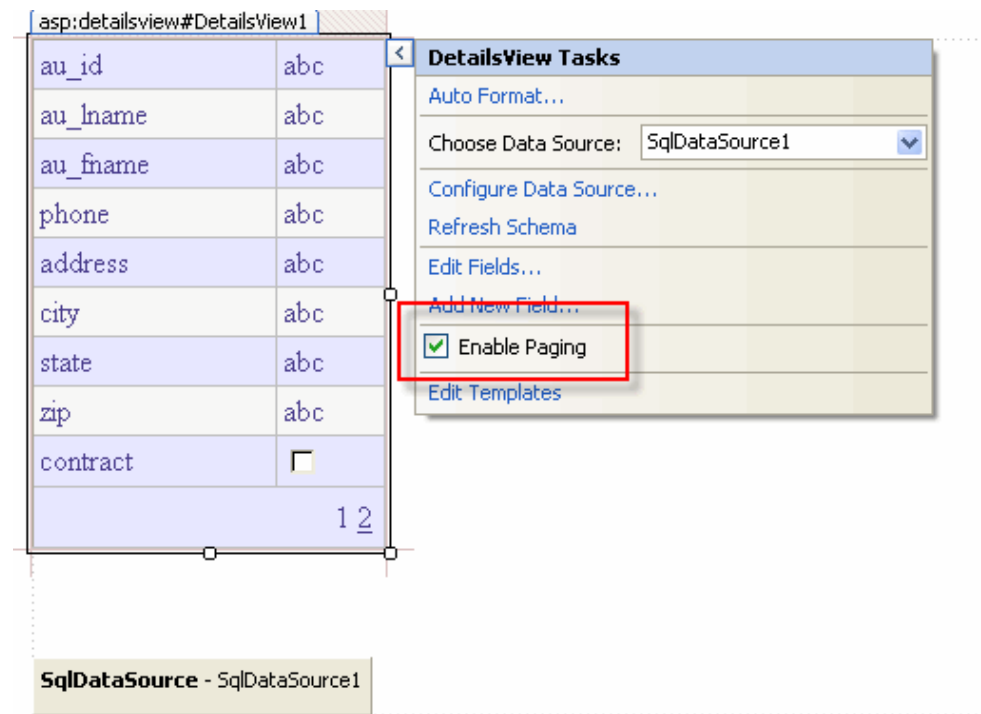
Otra diferencia es que el DetailView construye una tabla para poner los elementos y el FormView no, por lo tanto es mas adecuado para cuando queramos una libertad total de colocación de los elementos. Por contra ya te he comentado antes que trabaja siempre con plantillas que pueden ser muy flexibles pero siempre un poco mas complejas. Vamos a trabajar con los dos.

9.1 DetailView.

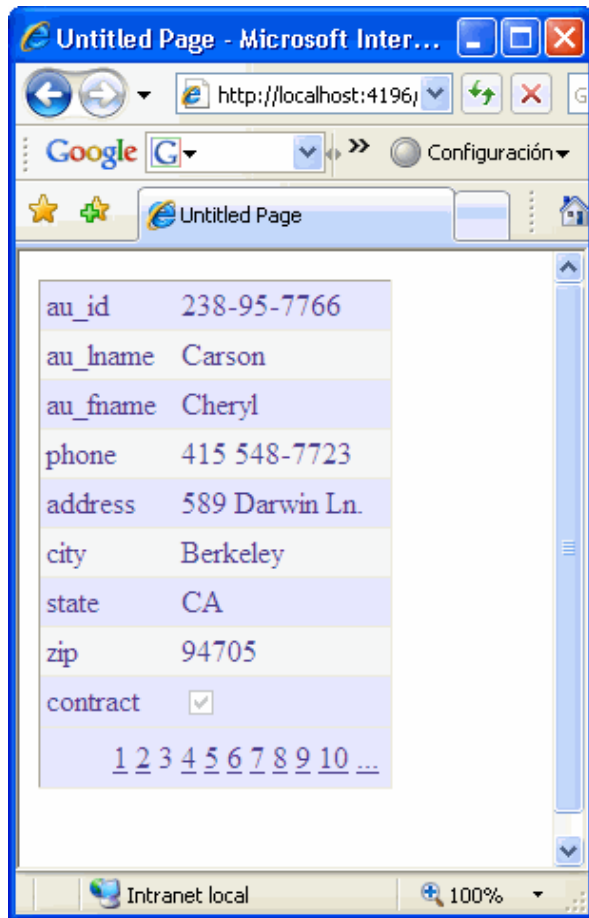
Esta vista de detalles muestra un solo registro a la vez mostrando una tabla donde cada fila es un campo. Podemos utilizar esta vista para mostrar el registro que está seleccionado actualmente. Además podremos movernos por los registros porque incorpora controles para paginación si establecemos la propiedad

Enlace a datos. Control de cuadrícula

“AllowPaging” a “true”. Podemos configurar la paginación de forma similar a la cuadrícula: con las propiedades `PagerSettings` y `PagerStyle`. Por ejemplo, crea una página y enlaza una tabla con este control:



Ejecuta la página:



Parece que simulamos una navegación por registros pero es un poco lento porque provocamos un “postback” cada vez que queremos movernos a uno, por eso el GridView es mas adecuado para ver de un vistazo un grupo de registros. Pero lo peor de todo es que ese postback produce un refresco de toda la consulta, por tanto es mucho trabajo para el servidor, ya que no va directamente al registro sino que vuelve a hacer la consulta y a situarse en el registro que le toque mostrar. La solución para mejorar esto es el famoso “caching” que ya te he contado que puedes ver en el curso ASP.NET Avanzado.

Definir los campos

La definición es muy sencilla porque el control consulta el origen de datos y nos muestra una fila para cada campo. Podemos deshabilitar esta generación automática estableciendo “AutoGenerateRows” a “false”.

Internamente no es muy diferente a los Datagrid. Los elementos están representados por etiquetas de tipo “BoundField”, los botones pueden crearse con “ButtonField” y así sucesivamente. En este ejemplo definimos un “DetailView” que muestra información de un producto:

```
<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False"
    DataSourceID="sourceProducts">
    <Fields>
        <asp:BoundField DataField="ProductID" HeaderText="ProductID"
```

Enlace a datos. Control de cuadrícula

```
ReadOnly="True" />

<asp:BoundField DataField="ProductName" HeaderText="ProductName" />
<asp:BoundField DataField="SupplierID" HeaderText="SupplierID" />
<asp:BoundField DataField="CategoryID" HeaderText="CategoryID" />
<asp:BoundField DataField="QuantityPerUnit" HeaderText="QuantityPerUnit" />
<asp:BoundField DataField="UnitPrice" HeaderText="UnitPrice" />
<asp:BoundField DataField="UnitsInStock" HeaderText="UnitsInStock" />
<asp:BoundField DataField="UnitsOnOrder" HeaderText="UnitsOnOrder" />
<asp:BoundField DataField="ReorderLevel" HeaderText="ReorderLevel" />
<asp:CheckBoxField DataField="Discontinued" HeaderText="Discontinued" />

</Fields>

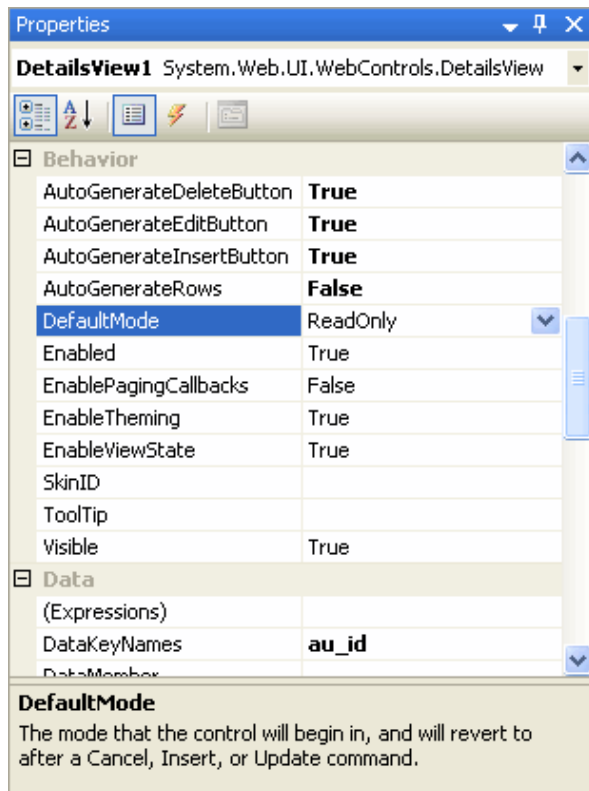
...

</asp:DetailsView>
```

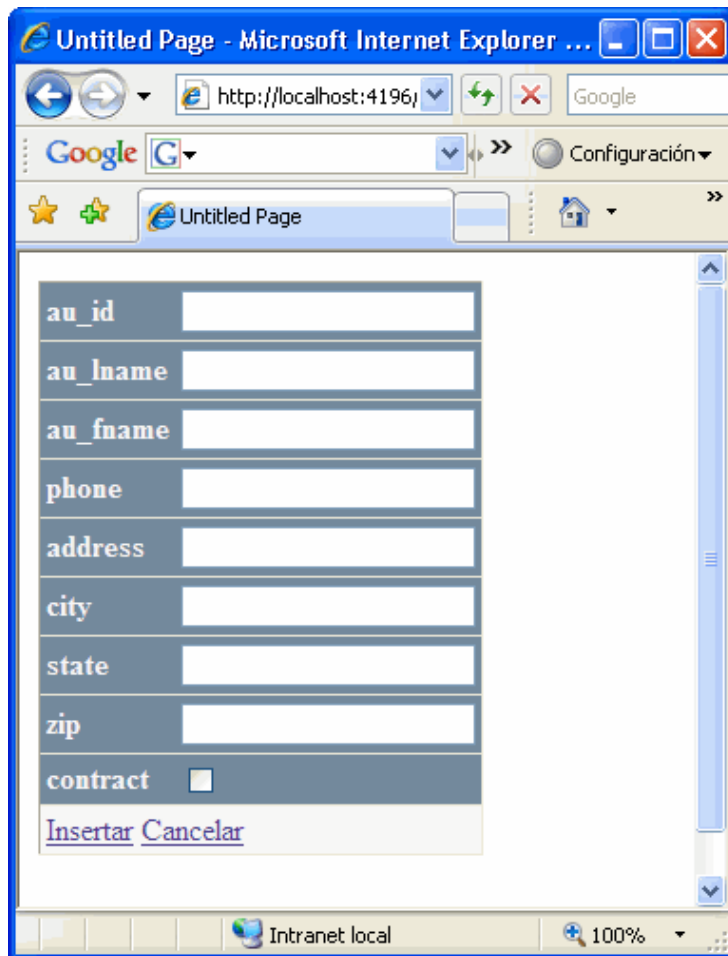
Podemos utilizar “BoundFiled” para establecer las propiedades como el texto del encabezado, formato del texto, comportamiento de la edición... Además podemos utilizar “ShowHeader” que si está a “false” el texto no se muestra en la fila y el campo ocupa las dos celdas.

El funcionamiento general en cuanto a estilos, eventos y edición es igual que en el DataGrid. La única diferencia es que en lugar de crear una columna para los controles de edición ahora simplemente tendremos que establecer las propiedades booleanas “AutoGenerateDeleteButton”, “AutoGenerateEditButton” y “AutoGenerateInsertButton”:

Enlace a datos. Control de cuadrícula



Estos enlaces se colocan en la parte inferior del control. Cuando de afluaden o editan registros siempre se utilizan controles de texto estándar, igual que con el Datagrid. Si ejecutas el ejemplo con todos los botones podremos por fin insertar datos. Así que si le das al botón de Insertar tendrás:



The screenshot shows a web browser window titled 'Untitled Page - Microsoft Internet Explorer ...'. The address bar shows 'http://localhost:4196/'. The page content is a form with the following fields: 'au_id', 'au_lname', 'au_fname', 'phone', 'address', 'city', 'state', 'zip', and 'contract' (a checkbox). Below the fields are two buttons: 'Insertar' and 'Cancelar'. The browser's status bar at the bottom shows 'Intranet local' and '100%' zoom.

La edición de los campos para poder insertar los datos. Si queremos mas flexibilidad tendremos que ver el último control por ahora que es el control FormView

10. FormView

Si necesitamos la flexibilidad de las plantillas este control nos ofrece mucha mas posibilidades que el anterior. La buena noticia es que la plantilla “FormView” coincide prácticamente con la “TemplateFiled” del “GridView”. Esto significa que podremos trabajar con las plantillas:

- ItemTemplate
- EditItemTemplate
- InsertItemTemplate
- FooterTemplate
- HeaderTemplate
- EmptyDataTemplate
- PagerTemplate

Para que veas que es igual fijate como seria una plantilla básica de tipo “itemtemplate” donde queremos poner tres valores

```
<asp:FormView ID="FormView1" runat="server" DataSourceID="sourceProducts">  
    <ItemTemplate>
```


Enlace a datos. Control de cuadrícula

```
<b>En sock:</b>

<%# Eval("Unidades_Stock") %>

<br />

<b>Pedidos:</b>

<%# Eval("Unidades_Pedidas") %>

<br />

<b>Reclamadas:</b>

<%# Eval("Unidades_reclamadas") %>

<br />

</ItemTemplate>

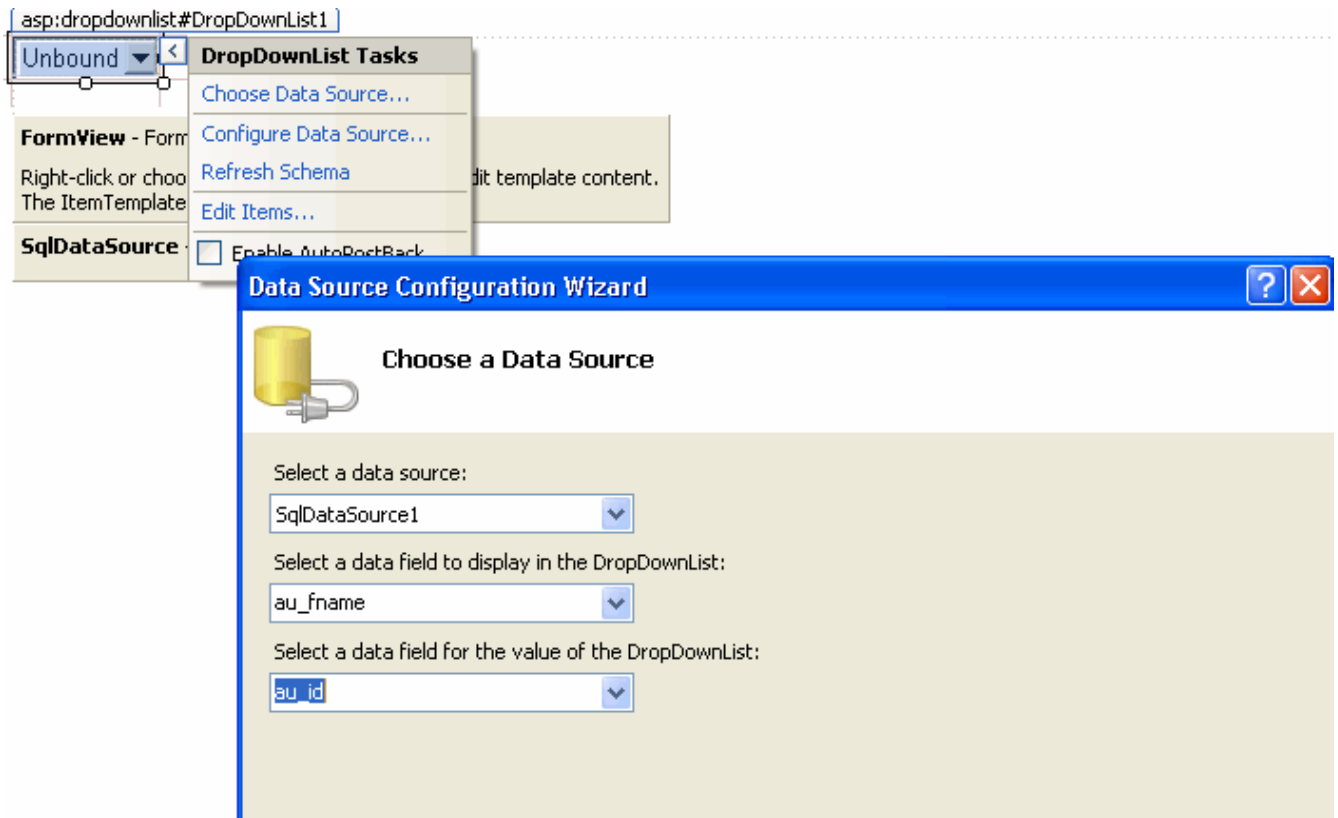
</asp:FormView>
```

Al igual que en el otro control, en el FormView mostraremos un solo registro a la vez. Podremos movernos de igual forma paginando los registros. Otra opción es enlazar el dato para que devuelva un registro. Por ejemplo seleccionando uno de un cuadro desplegable:

The screenshot shows a web form with a dropdown menu labeled 'Databound'. Below the dropdown, there are two data source controls. The first control, 'SqlDataSource - SqlDataSource2', displays a list of fields: au_id: abc, au_lname: abc, au_fname: abc, phone: abc, address: abc, city: abc, state: abc, zip: abc, and contract: ☐. The second control, 'SqlDataSource - SqlDataSource1', is partially visible below the first one.

Necesitaremos dos orígenes de datos, como ves en la pantalla. En primero de ellos, par el cuadro desplegable será:

Enlace a datos. Control de cuadrícula



Que lo estoy enlazando con el mismo origen de datos, le digo que muestre el campo “au_fname” y cuando se seleccione que devuelva el índice “au_id”. Esto en el código queda:

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
    DataSourceID="SqlDataSource2" DataTextField="au_fname" DataValueField="au_id">
</asp:DropDownList>
```

Recuerda que el “autopostback” debe estar actualizado para que haga un “postback” al seleccionar un elemento de la lista. Para el “formview” haremos lo mismo pero la consulta estará parametrizada para que cargue el registro del índice del elemento seleccionado. Así que seleccionamos la misma tabla:

Configure Data Source - SqlDataSource1

Configure the Select Statement

How would you like to retrieve data from your database?

☐ Specify a custom SQL statement or stored procedure

☒ Specify columns from a table or view

Name:

Columns:

<input checked="" type="checkbox"/> *	<input type="checkbox"/> city
<input type="checkbox"/> au_id	<input type="checkbox"/> state
<input type="checkbox"/> au_lname	<input type="checkbox"/> zip
<input type="checkbox"/> au_fname	<input type="checkbox"/> contract
<input type="checkbox"/> phone	
<input type="checkbox"/> address	

☐ Return only unique rows

SELECT statement:

Y en la restricción le decimos que es el campo del otro control:

Add WHERE Clause

Add one or more conditions to the WHERE clause for the statement. For each condition you can specify either a literal value or a parameterized value. Parameterized values get their values at runtime based on their properties.

Column:

Operator:

Source:

Parameter properties

Control ID:

Default value:

SQL Expression:

Value:

WHERE clause:

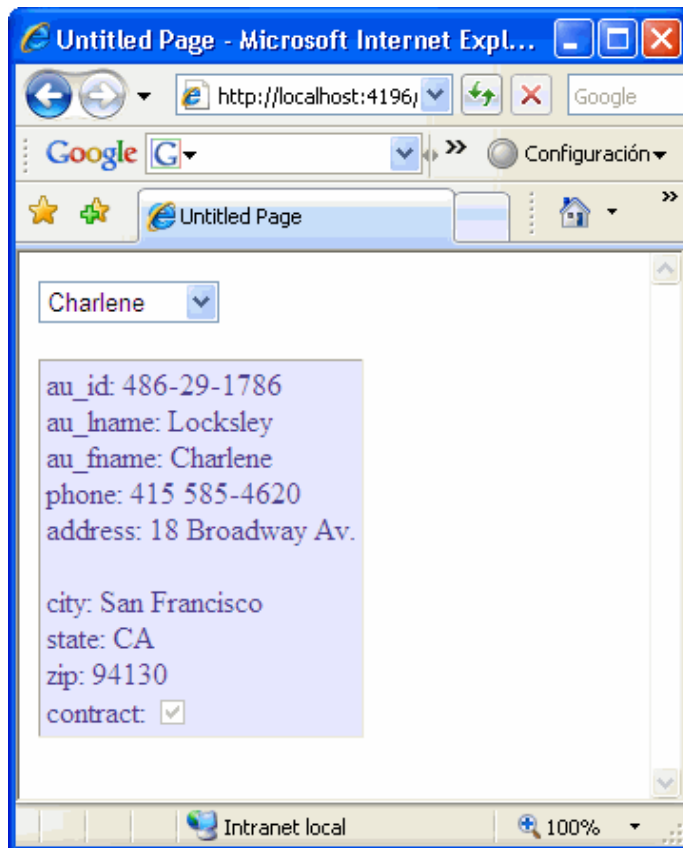
SQL Expression	Value
[au_id] = @au_id	DropDownList1.SelectedValue

Enlace a datos. Control de cuadrícula

Que en el código sería:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:Conexion_Pubs %>"
    SelectCommand="SELECT * FROM [authors] WHERE ([au_id] = @au_id)">
    <SelectParameters>
        <asp:ControlParameter ControlID="DropDownList1" Name="au_id"
            PropertyName="SelectedValue" Type="String" />
    </SelectParameters>
</asp:SqlDataSource>
```

Y el resultado es que al seleccionar el elemento en la lista nos lo actualiza debajo:



Revisa el código para que veas las plantillas que nos ha puesto y que puedes modificar. Además puedes ponerle los botones para poder modificar los datos y todo funcionará, añadiendo, ya sabes las sentencias SQL de los comandos que quieras implementar.

[Pulsa aquí para descargar los ejemplos de este tema](#)

Ejercicios

Ejercicio 1

Crea una página con todas las operaciones necesarias para mantener la tabla de las provincias creada anteriormente. Utiliza el control que te sea mas cómodo.

<u>provincia</u>	
Editar Eliminar	Barcelona
Editar Eliminar	Vizcaya
Editar Eliminar	La Rioja
Actualizar Cancelar	<input type="text" value="Madrid"/>
Editar Eliminar	Valencia
Editar Eliminar	Guipuzcoa
Editar Eliminar	Cantabria
Editar Eliminar	Sevilla
Editar Eliminar	Navarra

Recuerda que tablas deben tener claves únicas, si no la pusiste deberás hacerlo para que te funcione bien.

Ejercicio 2

Crea una página para mostrar la tabla de datos, cruzando las tablas de usuarios y provincias para el campo “provincia“. Admite paginación, ordenación y selección.

Enlace a datos. Control de cuadrícula

Generador de consultas

usuarios

- ☒ id
- ☒ nombre
- ☒ apellidos
- ☒ direccion
- ☒ fecha
- ☒ localidad
- ☐ provincia
- ☒ sexo

provincias

- ☐ *(Todas las columnas)
- ☐ id
- ☒ provincia

Columna	Alias	Tabla	Resu...	Tipo de orden	Criterio de ord...	Filtro	O...
direccion		usuarios	<input checked="" type="checkbox"/>				
fecha		usuarios	<input checked="" type="checkbox"/>				
localidad		usuarios	<input checked="" type="checkbox"/>				
sexo		usuarios	<input checked="" type="checkbox"/>				
provincia	Provincia	provincias	<input checked="" type="checkbox"/>				

```

SELECT usuarios.id, usuarios.nombre, usuarios.apellidos, usuarios.direccion, usuarios.fecha, usuarios.localidad, usuarios.sexo,
       provincias.provincia AS Provincia
FROM usuarios INNER JOIN
       provincias ON usuarios.provincia = provincias.id
    
```

id	nombre	apellidos	direccion	fecha	localidad	sexo
1	Jose	Rodriguez	Jorge Vigón	19/03/1965 0:0...	Logroño	True

1 de 6 | Celda de sólo lectura. | Consulta cambiada

Ejecutar consulta | Aceptar | Cancelar

Construye la consulta con el generador de consultas:

Página sin título - Microsoft Internet Explorer proporcionado por Metzeler APS Ibérica S.A.

http://localhost:1353/Enunciados/tema_12/tabla_usuarios.aspx

Google

Google

Ir >> Configuración

Página sin título

Página Herramientas

nombre	apellidos	direccion	fecha	localidad	Provincia	sexo	
Jose	Rodriguez	Jorge Vigón	19/03/1965 0:00:00	Logroño	La Rioja	<input checked="" type="checkbox"/>	Seleccionar
aaa	apellidosaaa	calle 1	01/01/1970 0:00:00	Badalona	Barcelona	<input checked="" type="checkbox"/>	Seleccionar
bbb	apellidosbbb	calle 2	10/04/1972 0:00:00	Santander	Cantabria	<input type="checkbox"/>	Seleccionar
ccc	apellidosccc	calld 3	28/12/1973 0:00:00	Madrid	Madrid	<input type="checkbox"/>	Seleccionar
ddd	apellidosddd	calle 4	07/07/1980 0:00:00	Pamplona	Navarra	<input type="checkbox"/>	Seleccionar
eee	apellidoseee	calle 5	02/04/1990 0:00:00	Logroño	La Rioja	<input checked="" type="checkbox"/>	Seleccionar

Intranet local 100%

Ejercicio3

Página sin título - Microsoft Internet Explorer proporcionado por Metzeler APS Ibérica S.A.

http://localhost:1353/Enunciados/tema_12/edicion_usuarios.aspx

Google

Página sin título

	id	nombre	apellidos	direccion	fecha	localidad	provincia	sexo
Seleccionar	1	Jose	Rodriguez	Jorge Vigón	19/03/1965 0:00:00	Logroño	3	<input checked="" type="checkbox"/>
Seleccionar	2	aaa	apellidosaaa	calle 1	01/01/1970 0:00:00	Badalona	1	<input checked="" type="checkbox"/>
Seleccionar	3	bbb	apellidosbbb	calle 2	10/04/1972 0:00:00	Santander	7	<input type="checkbox"/>
Seleccionar	4	ccc	apellidosccc	calle 3	28/12/1973 0:00:00	Madrid	4	<input type="checkbox"/>
Seleccionar	5	ddd	apellidosddd	calle 4	07/07/1980 0:00:00	Pamplona	9	<input type="checkbox"/>
Seleccionar	6	eee	apellidoseee	calle 5	02/04/1990 0:00:00	Logroño	3	<input checked="" type="checkbox"/>

id	4
nombre	ccc
apellidos	apellidosccc
direccion	calle 3
fecha	28/12/1973 0:00:00
localidad	Madrid
provincia	4
sexo	<input type="checkbox"/>
Editar Eliminar Nuevo	

Completa el caso del ejemplo anterior. Cuando se seleccione una fila debemos poner los campos debajo para poderse editar:

Ejercicio 4

Modifica el caso anterior para que incluya:

- Un control calendario para editar la fecha de nacimiento
- Un cuadro de lista para seleccionar la provincia

El resultado será:

Enlace a datos. Control de cuadrícula

	<u>id</u>	<u>nombre</u>	<u>apellidos</u>	<u>direccion</u>	<u>fecha</u>	<u>localidad</u>	<u>provincia</u>	<u>sexo</u>
Seleccionar	1	Jose	Rodriguez	Jorge Vigón	19/03/1965 0:00:00	Logroño	3	<input checked="" type="checkbox"/>
Seleccionar	2	aaa	apellidosaaa	calle 1	02/01/1970 0:00:00	Badalona	1	<input checked="" type="checkbox"/>
Seleccionar	3	bbb	apellidosbbb	calle 2	10/04/1972 0:00:00	Santander	2	<input type="checkbox"/>
Seleccionar	4	ccc	apellidosccc	calle 3	12/02/2008 0:00:00	Madrid	4	<input checked="" type="checkbox"/>
Seleccionar	5	ddd	apellidosddd	calle 4	09/07/2008 0:00:00	Pamplona	9	<input type="checkbox"/>
Seleccionar	6	eee	apellidoseee	calle 5	02/04/1990 0:00:00	Logroño	3	<input checked="" type="checkbox"/>

id: 3
 nombre:
 apellidos:
 direccion:
 fecha:

≤ abril de 1972 ≥						
lu	ma	mi	ju	vi	sá	do
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

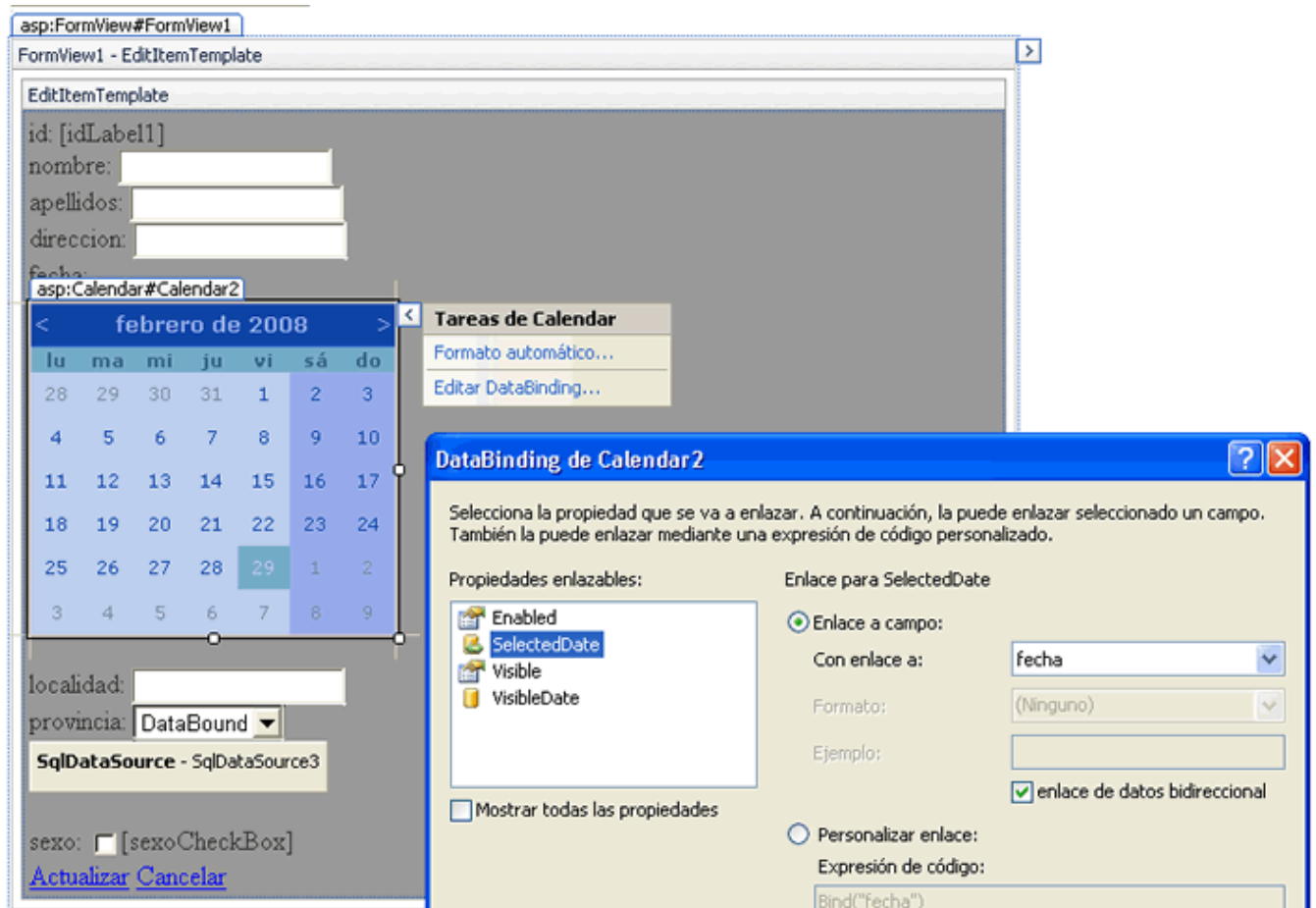
 localidad:
 provincia:
 sexo: ☐
[Actualizar](#) [Cancelar](#)

Donde podemos poner la fecha con el control calendario y la provincia seleccionarla de la lista desplegable.

Ayuda:

- Utilizar un control formview y editas las plantillas de "EditItem"
- Coloca los controles y haz este enlace para el calendario:

Enlace a datos. Control de cuadrícula



Y pones también la propiedad de “VisibleDate“, sino no actualiza el control.

- Para el cuadro desplegable haremos este enlace:

Enlace a datos. Control de cuadrícula

The screenshot shows the 'DataBinding de DropDownList1' dialog box in Visual Studio. The dialog is titled 'DataBinding de DropDownList1' and contains the following information:

- Propiedades enlazables:** A list of properties including DataSource, Enabled, SelectedIndex, SelectedValue, and Visible. 'SelectedValue' is selected.
- Enlace para SelectedValue:** The 'Enlace a campo:' radio button is selected. The 'Con enlace a:' field contains 'provincia'. The 'Formato:' field contains '(Ninguno)'. The 'Ejemplo:' field is empty.
- Personalizar enlace:** The 'Personalizar enlace:' radio button is unselected. The 'Expresión de código:' field contains 'Bind("provincia")'.
- Mostrar todas las propiedades:** An unchecked checkbox.
- Actualizar esquema:** A link at the bottom left.
- Buttons:** 'Aceptar' and 'Cancelar' buttons at the bottom right.

The background shows a web form titled 'FormView1 - EditItemTemplate'. It contains fields for 'id', 'nombre', 'apellidos', 'direccion', and 'fecha'. There is also a calendar for February 2008 and a dropdown menu for 'provincia'.

Y le pondrás un origen de datos para que se rellene con las provincias:

The screenshot shows the 'Asistente para la configuración de orígenes de datos' (Data Source Configuration Wizard) dialog box. The wizard is titled 'Asistente para la configuración de orígenes de datos' and has a yellow plug icon. It is at the 'Elegir un origen de datos' (Choose a data source) step.

The wizard contains the following information:

- Seleccionar un origen de datos:** A dropdown menu showing 'SqlDataSource3'.
- Seleccionar un campo de datos para mostrar en DropDownList:** A dropdown menu showing 'provincia'.
- Seleccionar un campo de datos para el valor de DropDownList:** A dropdown menu showing 'id'.
- Actualizar esquema:** A link at the bottom left.
- Buttons:** 'Aceptar' and 'Cancelar' buttons at the bottom right.

Enlace a datos. Control de cuadrícula

Por último para que se actualice la cuadrícula cuando se modifiquen los datos le pondremos un “databind” cuando se dispare el evento apropiado:

```
Protected Sub FormView1_ItemUpdated(ByVal sender As Object)
    GridView1.DataBind()
End Sub
```