Navegación y ADO.NET

Indice

Nº 11 - Navegación y ADO.NET

1. Navegación por el sitio Web

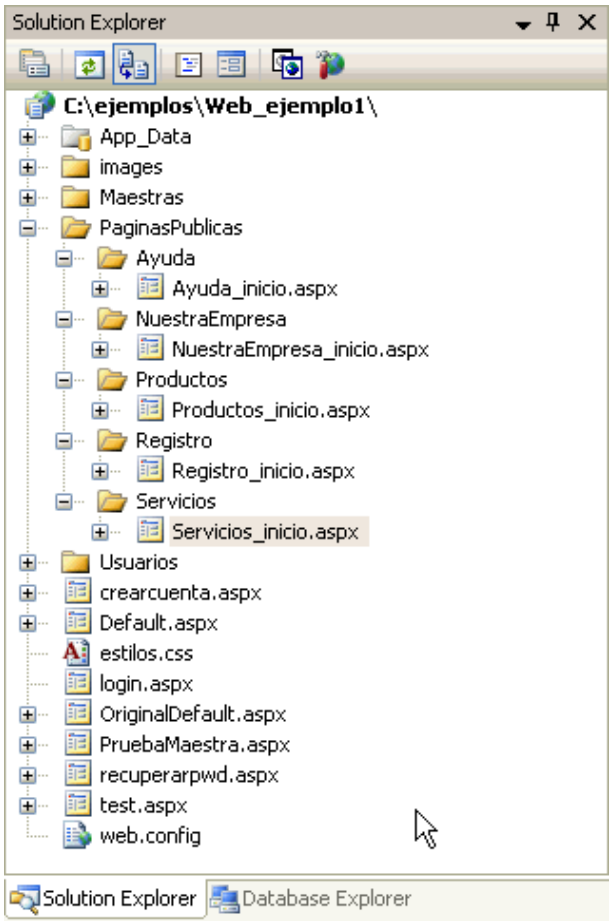
Sabemos como podemos movernos por las páginas web de nuestra aplicación mediante los enlaces o los métodos "Response.Redirect". En nuestro web debemos tener una organización en las páginas para permitir una buena accesibilidad a las páginas (término tan de moda) y que el acceso por las distintas secciones sea fluido y sobre todo intuitivo. Cuando nos encontramos una página compleja en la que no conseguimos encontrar ese driver que queremos descargarnos, la culpa no es nuestra torpeza sino que es del que ha hecho el sitio web por no hacer las cosas sencillas y fáciles. Muchas veces se confunden los términos de complejidad y potencia. Esto por ejemplo es muy de Linux: es potente porque es muy complejo. No es cierto! es complejo porque no está bien hecho. Lo realmente difícil en la programación y en general en el mundo de la informática es hacer que las tareas sean sencillas, por muy complejas que sean internamente.

Bueno, soltada esa retahíla la idea que os quiero transmitir es que tenemos que facilitar mucho la vida a los usuarios. Debemos aprender a utilizar estos sencillos controles para ayudarles en la navegación de nuestro sitio.

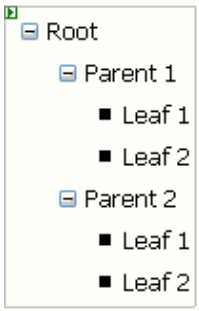
1.1 Organizar nuestras páginas

Debemos primero hacernos una composición de como queremos organizar nuestro sitio: secciones que va a tener, zonas privadas, ... no estaría mal crear unas páginas en blanco para las zonas principales para ir preparando esas páginas y así dejar los principales enlaces funcionando.

Por ejemplo, he creado una carpeta para las páginas de acceso público y en cada una de ellas una página en blanco:



Dentro de la carpeta de páginas públicas hemos creado varias secciones con una página en blanco de ejemplo en cada una. Los dos controles que tenemos especiales para facilitar la navegación son dos: "Menu" y "TreeView":



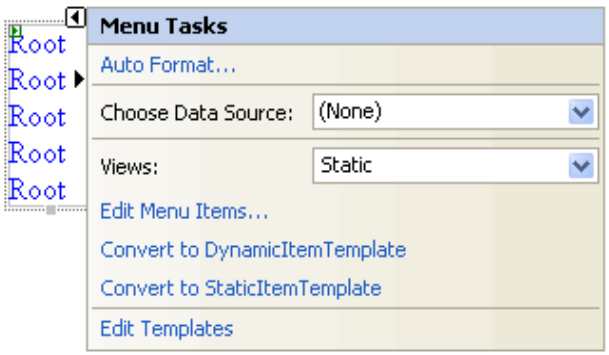
En el primer caso se muestra una lista de opciones que va mostrando otros paneles a la derecha a medida que

el usuario va explorando las opciones que tienen una flecha. En el otro caso es el típico árbol jerárquico muy práctico para navegar por sitios.

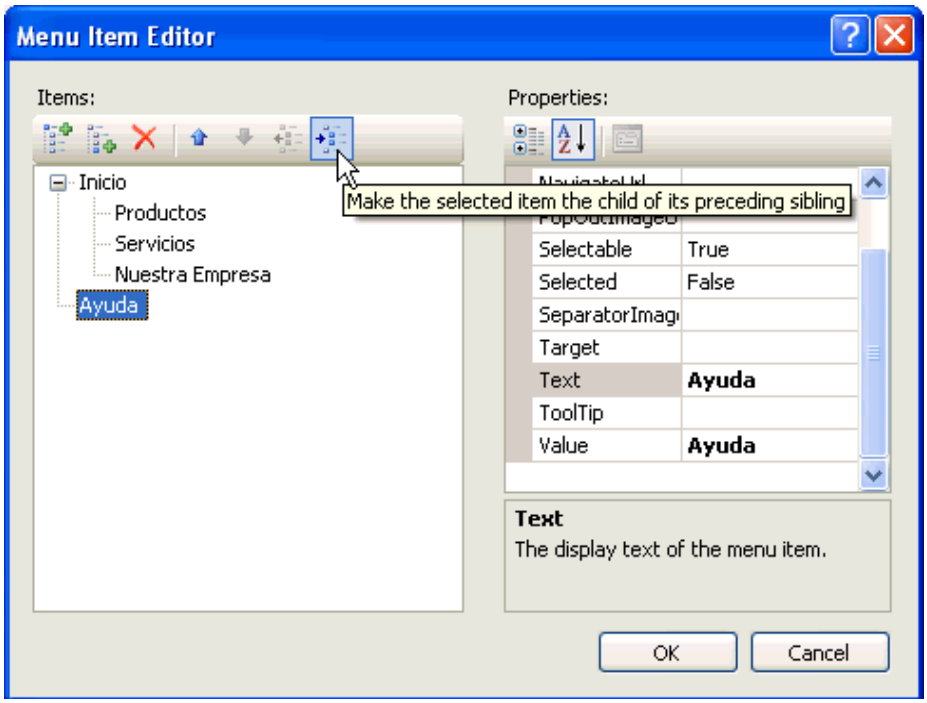
Los dos controles se pueden alimentar de datos estáticos o dinámicos. En éste último caso los datos se cargan desde un fichero externo llamado mapa del sitio "site map". Así que ya tenemos una gran ventaja, si este control está puesto en muchas páginas nos basta con modificar estos datos del fichero para que todas las páginas que tienen el control se actualicen con la nueva estructura.

Bueno empezamos, ahora resulta que claro debemos copiar y pegar este control en todas las páginas de nuestro sitio Web aunque claro por lo menos serán todos iguales ya que se van a cargar los datos desde un fichero externo. Aún así parece un poco tedioso así que mas adelante crearemos un "control de usuario" que nos va a facilitar esta labor. La otra posibilidad es poner estos controles en una página maestra pero no queremos hacer eso para no "condenar" esa página con esos controles que en varias partes del Web no hacen falta.

Puesto que los dos controles son muy similares en cuanto a su diseño y programación nos centraremos en uno de ellos: en el menú. Ahora vamos a crear una página Web que va a servir para navegar por el Web. Crea una página, ponle por ejemplo "menu.aspx" y le añades el control de menú que se encuentra en la barra de la izquierda en la sección "navigate":

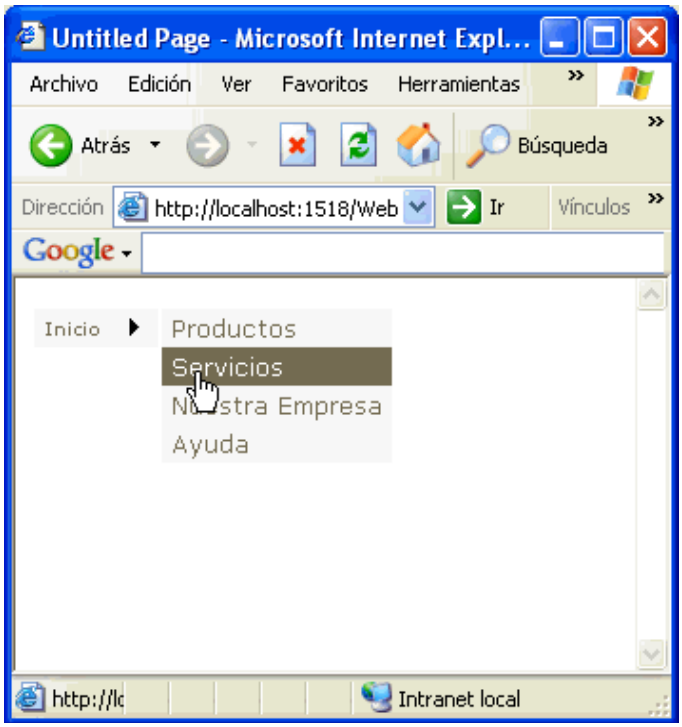


De las opciones que ves a la derecha selecciona la de "Edit Menu Items" para poder añadir los elementos de nuestro menú:



Vamos a poner una opción de "Inicio" y las otras las dejaremos una posición a la derecha que significa que dependen de la superior "Inicio". Para hacer esto vete añadiendo elementos con el primer botón y luego con el que tienes marcado en la imagen muévelos una posición a la derecha para que dependa de "Inicio". Además en cada elemento modifica la propiedad "text" para poner el valor correspondiente. Además debes poner las propiedades de "NavigateURL" que será el enlace de la opción del menú y la propiedad "Tooltip" para que le aparezca un texto de ayuda al usuario cuando ponga el ratón encima. Verás que para la selección de las páginas Web destino nos muestra las páginas Web que componen nuestro sitio para seleccionarla mas fácilmente.

Utiliza también la opción de "Autoformato" para mejorar el aspecto estándar. Ya ves que es muy sencillo de utilizar y el resultado es por supuesto el esperado:



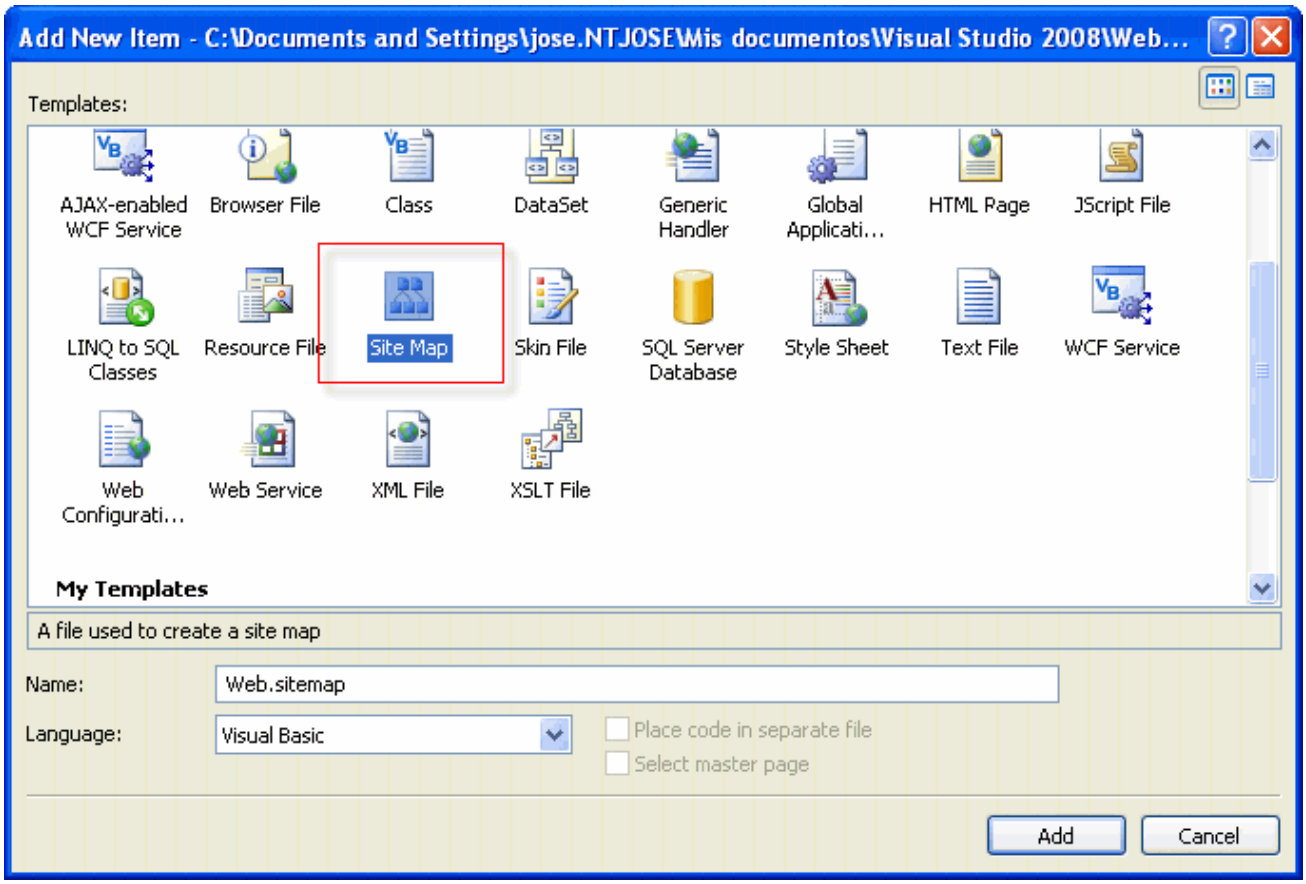
Algo que antes era bastante complejo ahora es muy sencillo de hacer... bueno lo mismo para el control jerárquico, prácticamente tiene las mismas opciones. Investiga tu en las propiedades y en los botones para mover los elementos ya metidos para que domines esta pantalla.


Los menús de navegación

1.2 Crear un mapa del sitio

Un mapa de nuestro sitio es una interesante forma de organizar nuestro web mediante un fichero externo. Es decir, si creamos nuestra estructura en un fichero externo y asociamos los controles de menú a este fichero los menús se harán dinámicos, pudiendo añadir y eliminar opciones de menú de forma automática. Otra vez nos encontramos con una gran facilidad para no tener que modificar a mano nuestras páginas de menús principales.

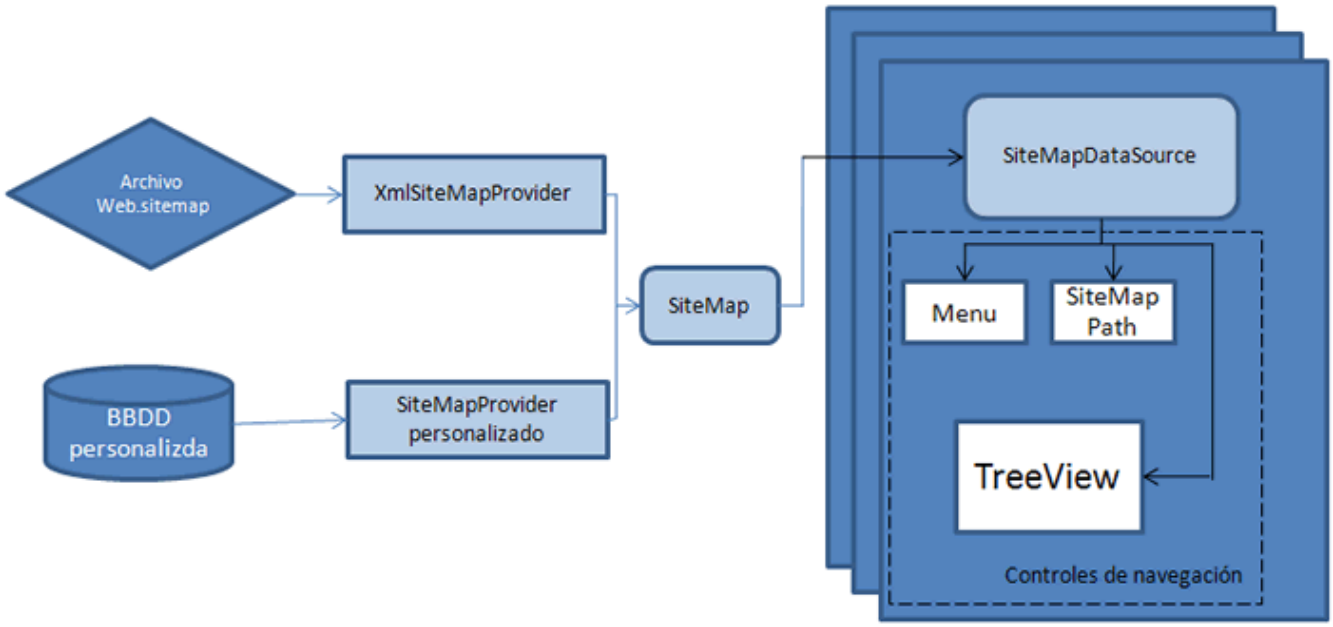
Hay varias formas de crear un mapa del sitio, la mas fácil de todas es crear un fichero llamado "Web.sitemap" en la raíz de la carpeta. Los pasos para esto sería estos, hacemos clic con el botón derecho en la raíz de nuestro sitio en el explorador de soluciones para añadir una nueva página:



Como ves, de tipo "Site Map" y el nombre predeterminado es el que he dicho antes... Nos mostrará una página en formato XML:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="" title=""  description="">
        <siteMapNode url="" title=""  description="" />
        <siteMapNode url="" title=""  description="" />
    </siteMapNode>
</siteMap>
```

Los controles "SiteMapDataSource" y "XmlSiteMapProvider" nos van a realizar estas operaciones de utilizar nuestro fichero del mapa del web. Fundamentalmente esta es la idea:



Podemos tener una base de datos personalizada para cargar dinámicamente las opciones en los menús. ¿ejemplo? Muy fácil, dependiendo del usuario que haya hecho login le cargaremos unas opciones u otras.

El punto de partida es la creación de un proveedor de mapas llamado "XmlSiteMapProvider" que será el encargado de recuperar la información necesaria del fichero XML. Este objeto simplemente buscará un fichero "web.sitemap" situado en la raíz de nuestro directorio virtual. Su tarea es extraer las opciones del mapa y crear un objeto de tipo "SiteMap". Por fin este SiteMap ya será un origen de datos (SiteMapDatasource) para nuestros controles que puedan alimentarse de esta información. Pondremos este control en todas las páginas que queramos poner el control de navegación.

Nota Si ya has descubierto las bondades de utilizar las páginas maestras (si no lo has hecho, lo harás pronto) habrás deducido que si ponemos este control en la página maestra estará disponible para todo el web que esté basado en esa página.

Vamos a entender ahora el formato de este fichero XML. Como ves en la pantalla del fichero XML, cada página se representa por un elemento de tipo "sitemapNode":

```
<siteMapNode url="~/default.aspx" title="Inicio"  description="Página de inicio" />
```

Las páginas comenzarán con el famoso símbolo "~/" que indican el inicio o raíz de nuestro web. En principio no es obligatoria esta sintaxis pero no cuesta nada ponerla así y nos evitará muchos problemas de que si hay que poner rutas relativas, absolutas, ... Está claro que si en otra carpeta de nuestro web hubiera otro fichero llamado así "default.aspx" cargaría ese en lugar del raíz, por eso es importante indicarle la ruta de esta forma, así evitamos muchas confusiones.

Como ves es una estructura jerárquica donde hay unos elementos de mas jerarquía que otros. El primer elemento "siteMapNode" es el raíz del sitio Web. Estrictamente debería ser así:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="~/Default.aspx" title="Inicio"  description="Pulsa aquí para...">
        <siteMapNode url="~/PaginasPublicas/Productos/Productos_inicio.aspx"
                     title="Productos"  description="Pulsa aquí para..." />
        <siteMapNode url="~/PaginasPublicas/Servicios/Servicios_inicio.aspx"
                     title="Servicios"  description="Pulsa aquí para..." />
        <siteMapNode url="~/PaginasPublicas/NuestraEmpresa/NuestraEmpresa_inicio.aspx"
                     title="Nuestra empresa"  description="Pulsa aquí para..." />
        <siteMapNode url="~/PaginasPublicas/Ayuda/Ayuda_inicio.aspx"
                     title="Ayuda"  description="Pulsa aquí para..." />
    </siteMapNode>
</siteMap>
```

En ese ejemplo que ves he creado una carpeta que se llama "PaginasPublicas" por que serán de libre acceso a todos los usuarios y dentro de ella otras carpetas con distintas secciones, así que he puesto finalmente la página destino de cada una de estas secciones. Al ser jerárquico estamos estableciendo dos niveles: uno para la opción del "inicio" y en otro nivel todas las opciones de navegación a las partes de mi web. Es decir, el resultado nos deberá dar este aspecto:

No tenemos límite de anidamientos, así que podemos poner los niveles que queramos siempre con la estructura jerárquica de abrir un "sitemapnode", poner los secundarios y cerrar el "/sitemapnode". La única limitación o, mejor dicho, restricción es la de que no podemos tener dos elementos que tengan la misma dirección "URL". Esto es porque el "SiteMapProvider" de ASP.NET crea una colección de elementos basados en la URL y por tanto, no puede tener dos duplicados.

Pero si tenemos una solución podríamos poner dos destinos de esta forma:

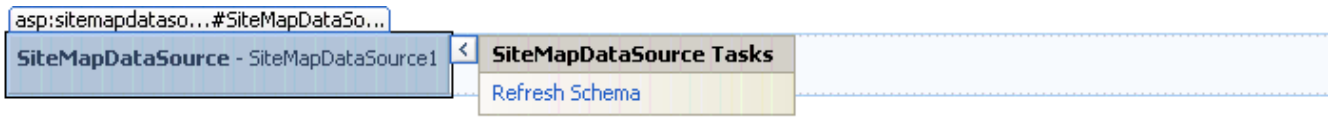
```
~/pagina_destino.aspx?numero=1

~/pagina_destino.aspx?numero=2
```

El nombre destino ya no es el mismo al tener un parámetro. Simplemente con no hacer nada con ese "querystring" en el destino ya está solucionado el poder poner dos páginas con la misma URL.

1.3 Ejemplo

Veamos como aplicamos todo esto a un ejemplo. Crea el fichero XML con el ejemplo de antes, este fichero nos servirá para mas operaciones así que si quieres puedes pulsar aquí para descargarlo.

Una vez definido vamos a ponerlo en marcha. Antes de ejecutarlo es muy buena idea crear las páginas destino en blanco, así nos aseguraremos de que no hay ningún enlace roto. Ahora debemos añadir un control del tipo "SitreMapDataSource" a nuestra página:

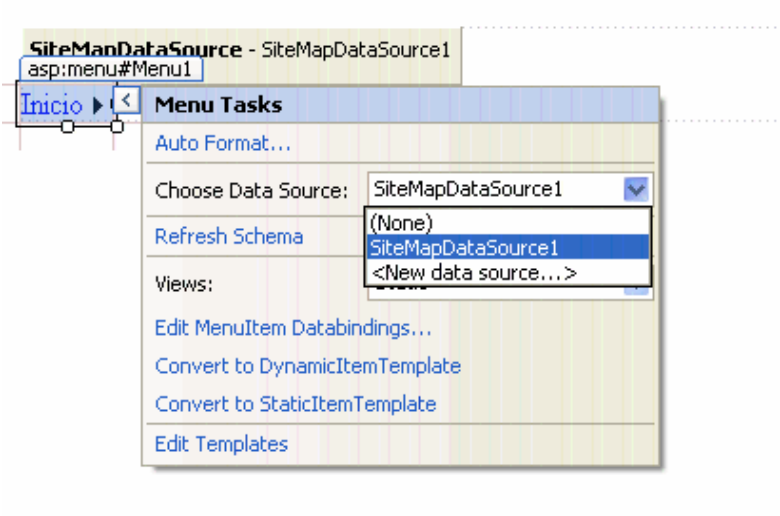


Este control lo tienes en el grupo "Data" de los controles de ASP.NET del cuadro de herramientas. Este control es visible en el diseño pero luego es invisible en la página destino. Lo que nos ha proporcionado es un origen de datos a la página, ahora debemos añadir cualquier control de navegación para obtener los resultados. Así que podemos utilizar cualquiera de estos controles de navegación:

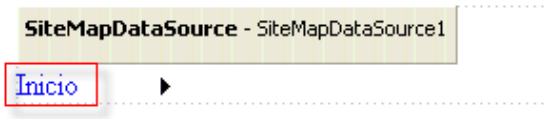
- TreeView. Muestra las opciones de forma jerárquica, como un árbol.
- Menu. Muestra un menú de varios niveles. Por defecto sólo nos muestra el primer nivel y al pasar el ratón por encima nos irá mostrando los menús interiores.
- SiteMapPath. Es un sencillo control de navegación, muestra en una línea los distintos niveles por los que está navegando. Es muy común en sitios grandes, por ejemplo en la página del el mundo se puede ver así:

Portada > Deportes > **Motor**

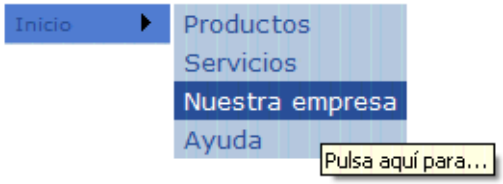
Ponemos ahora un control de los que hemos visto, por ejemplo un menú situado en el grupo "Navigation" de la barra de controles.



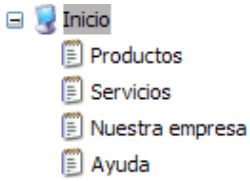
Pulsamos en la opción de "Choose Data Source" para seleccionar un tipo de datos. Veremos nuestro control de antes del tipo "SiteMapDataSource" y cuando lo seleccionamos automáticamente queda enlazado con el fichero XML de navegación:



Pulsa en la opción de "Auto Format" para poner un aspecto mas elegante y ejecutamos la página:



¡Y además funciona! Si seleccionas las opciones navegará a la página que hemos definido en nuestro fichero XML. Si utilizamos un control de tipo "TreeView" el resultado es:

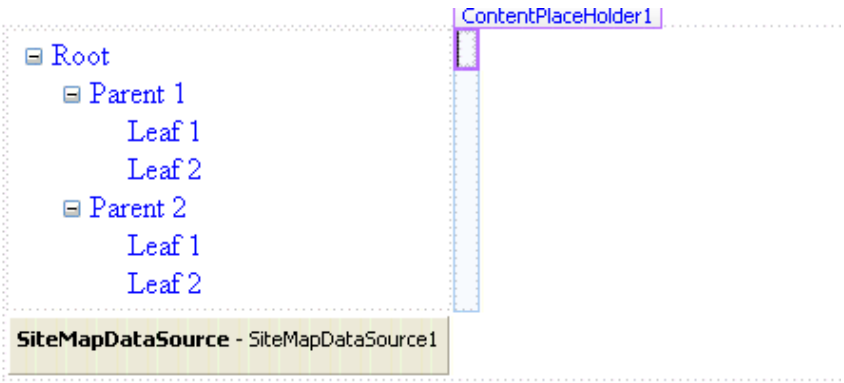


1.4 Enlazar páginas maestras con una mapa de sitio (Site Map)

La navegación web funciona mejor combinándola con la sencilla y práctica función de las páginas maestras. Es lógico ya que queremos mostrar los controles de navegación en todas las páginas y no sólo en la primera página. La solución mas sencilla es crear una página maestra con un control de tipo "SiteMapsDataSource". Por ejemplo con el control "Treeview":

```
<form id="form1" runat="server">
<table>
    <tr>
    <td style="width: 226px;vertical-align:top;">
        <asp:TreeView ID="menu" runat="server" DataSourceID="SiteMapDataSc
    </td>
    <td style="vertical-align: top;">
    <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
    </asp:ContentPlaceHolder>
    </td>
  </tr>
 <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
</table>
</form>
```

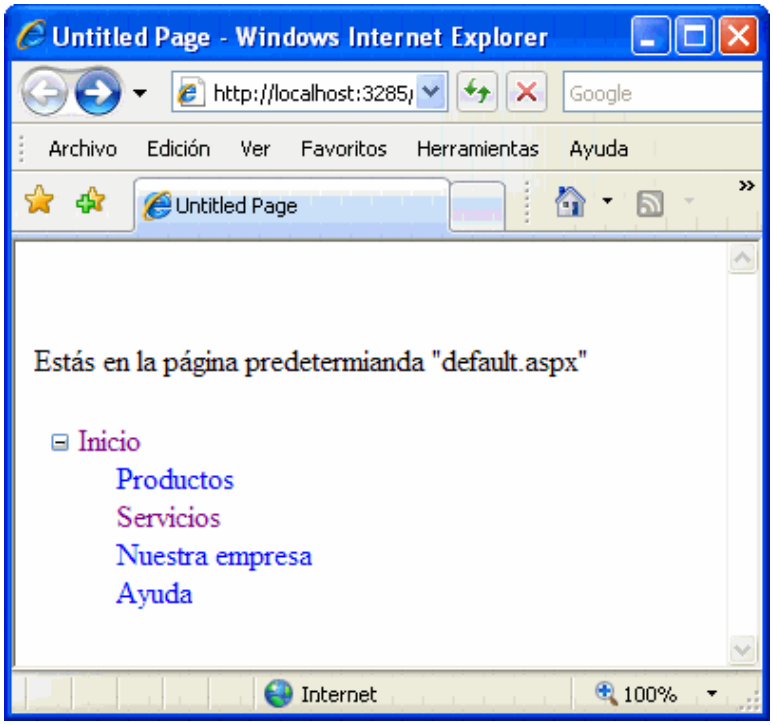
Que en diseño sería:



La página de contenido sería de esta forma:

```
<%@ Page Language="VB" MasterPageFile="~/Tema_11/master_sitemap.master" AutoEventWireup
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <br />
<br />
Estás en la página predetermianda "default.aspx"
</asp:Content>
```

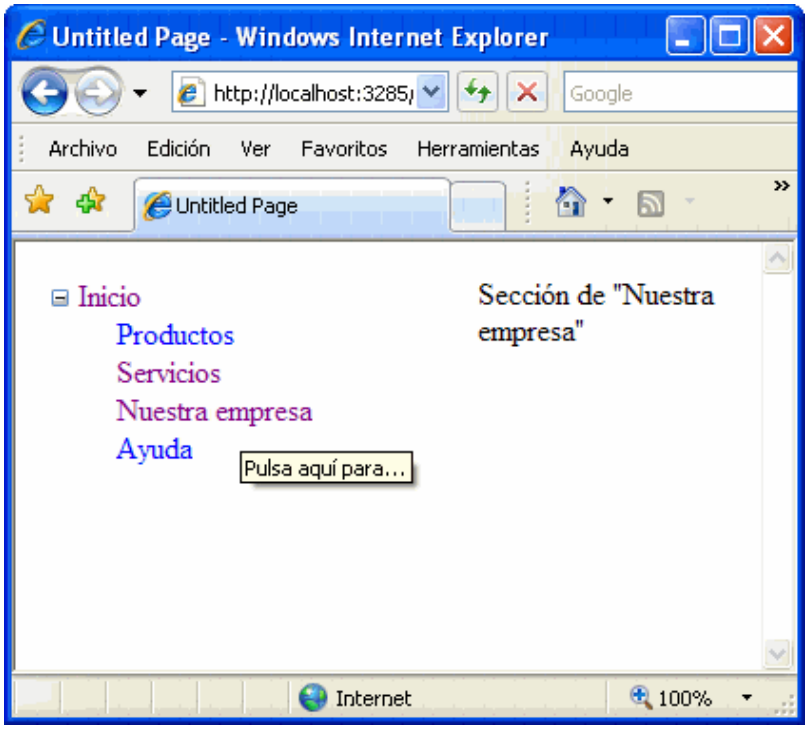
Que lógicamente nos incluirá el menú con los datos predeterminados que hemos puesto en la página del contenido anterior:



Ahora vamos a crear una segunda página de las que enlaza nuestro menú de ejemplo que esté asociada con nuestra página master con el menú de navegación, así podremos navegar perfectamente de un lado a otro.

Así que la página destino "nuestraempresa_inicio.aspx" la crearemos a partir de la página maestra creada antes:

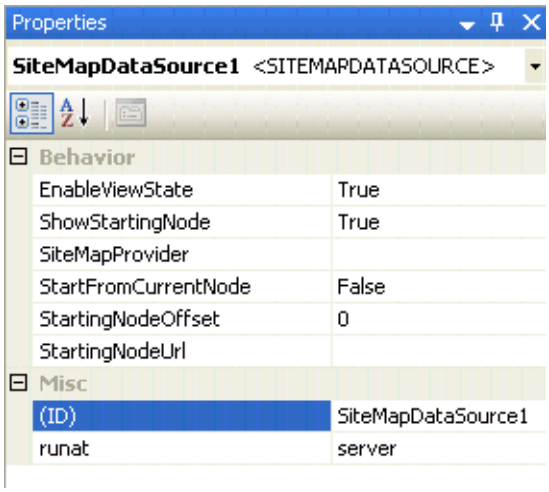
```
<%@ Page Language="VB" MasterPageFile="~/Tema_11/master_sitemap.master"
    AutoEventWireup="false" CodeFile="NuestraEmpresa_inicio.aspx.vb"
    Inherits="PaginasPublicas_NuestraEmpresa_NuestraEmpresa_inicio" title="Untitled Page" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <p>
        Sección de &quot;Nuestra empresa&quot;</p>
</asp:Content>
```



Y al ejecutar las dos páginas podremos navegar con estupendos resultados ya que el control permanecerá operativo para las páginas. Fíjate que de esta forma si añadimos una sección mas en nuestro menú además de tener que modificar solo el fichero XML, tendremos todos los menús de las páginas actualizados al trabajar con páginas master y tener un solo menú para todas.

Detalles del control SiteMapDataSource

Vamos a comentar algunos detalles del control "SiteMapDataSource" para poder tener mas opciones de visualización. Fíjate en sus propiedades:



Por ejemplo tenemos la posibilidad de poner la propiedad "ShowStartingNode" a "false", que provocará que no se muestre el primer nivel de opciones:

Productos
Servicios
Nuestra empresa
Ayuda

Con lo que ha desaparecido la opción de "Inicio". Otro tema interesante, por defecto sabemos que "SiteMapDataSource" muestra un árbol que comienza en el nodo raíz. Veamos las propiedades de antes a ver que nos pueden aportar:

Propiedad	Descripción
ShowStartingNode	Si está a "false" oculta el primer nivel, por defecto está activada.
StartingNodeUrl	Con esta propiedad cambiaremos el nodo de inicio. Estableciendo este valor a la URL del nodo que queremos asignar como primer nodo .
StartFromCurrentNode	Si está establecida a "true" asigna a la página actual como nodo de inicio. El árbol de navegación mostrará solo las páginas "hija" de este nodo.
StartingNodeOffset	Utilizaremos esta propiedad para desplazar el nodo de inicio hacia arriba o abajo en la jerarquía. Indicaremos unidades hacia arriba o abajo para desplazarse.

Practica un poco con estas opciones, los resultados son de verdad interesantes.

Utilizar distintos mapas en el mismo archivo

Imagina que tenemos dos tipos de perfiles en nuestro sitio web, uno para los clientes y otro para los empleados. Necesitaríamos dos grupos de opciones según que tipo de usuario esté visitando la página. Por ejemplo:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="~/Default.aspx" title="Inicio"  description="Inicio">
        <siteMapNode url="~/Default_clientes.aspx" title="Inicio clientes"  description="Inicio clientes" />

        <!--  lista de opciones para los clientes-->

        <siteMapNode url="~/Default_empleados.aspx" title="Inicio empleados"  description="Inicio empleados" />

        <!--  lista de opciones para los empleados-->

    </siteMapNode>
</siteMap>
```

Para enlazar con la parte de los clientes simplemente pondremos la propiedad "StaringNodeUrl" a "~/Default_clientes.aspx". Podemos hacer esto por programación: por ejemplo una pantalla de inicio de sesión y luego según el usuario que ha hecho login asignamos la propiedad en el Load de la página. O también lo podemos hacer creando dos páginas maestras una para cada grupo de usuarios. En la de empleados pondríamos la propiedad "StaringNodeUrl" a "~/Default_empleados.aspx". De esta forma solo mostraremos las páginas de los usuarios con su propio menú.

Todavía podemos hacerlo mas fácil indicando ficheros de mapas de sitios distintos para los dos menús, así los mantendría por separado:

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
3      <siteMapNode url="~/Default.aspx" title="Inicio"  description="Inicio">
4          <siteMapNode sitemapfile="clientes.sitemap" />
5          <siteMapNode sitemapfile="emeplados.sitemap" />
6      </siteMapNode>
7  </siteMap>
```

Aun con esta simplificación tenemos que tener siempre un único fichero en nuestra raíz llamado como sabemos "web.sitemap" pero al menos podemos tener ficheros independientes si tenemos que utilizarlo para distintas estructuras lógicas de nuestra aplicación web.


Crear menús con mapa del sitio

La clase SiteMap

Ahora veremos detalles interesantes para poder manejar mediante programación estos objetos de navegación. Por ejemplo para detectar y mostrar información del nodo que se ha seleccionado, para esto simplemente tendremos que interactuar en el siempre útil evento "Page_Load".

El punto de partida son las propiedades de navegación de la clase SiteMap, que proporciona información del nodo seleccionado "CurrentNode" y el nodo raíz "RootNode". Estas dos propiedades devuelven un objeto "SiteMapNode". Con este objeto ya podremos recuperar mucha información como el título, descripción y URL. Veamos ahora las propiedades de SiteMapNode:

Propiedad	Descripción
ParentNode	Devuelve el nodo un nivel superior en la jerarquía. Si estamos en la raíz devuelve "null"

ChildNodes	Proporciona una colección de todo los nodos "hijo". Se puede ir comprobando la propiedad "HasChildNodes" para comprobar si existen nodos
PreviousSibling	Devuelve el nodo anterior que está en el mismo nivel o "null" si no hay
NextSibling	Devuelve el siguiente nodo del mismo nivel o "null" si no hay

También podemos buscar nodos utilizando métodos del objeto "SiteMapProvider" que está disponible a través de la propiedad compartida (y por tanto accesible) "SiteMap.Provider". Por ejemplo el método "SiteMap.Provider.FindSiteMapNode()" permite buscar nodos por su URL.

Veamos un ejemplo, configuraremos dos etiquetas "label" en una página para mostrar la cabecera y descripción del nodo actual:

```
Sub Page_Load

label1.text=SiteMap.CurrentNode.Title

label2.text=SiteMap.CurrentNode.Description

End Sub
```

Si estamos utilizando páginas maestras, debemos poner este código en la página de código de la página maestra. Si quisiéramos navegar por los nodo con dos enlaces de "siguiente" y "anterior" podríamos hacerlo así:

```
Sub Page_Load

If SiteMap.CurrentNode.NextSibiling isnot Nothing then

label_navegacion.NavigateUrl=SiteMap.CurrentNode.NextSibiling.Url

label_enlace.Visible=True

else

     label_enlace.VIsible=False

end if

En Sub
```

Tranquilo, esto es para nota. El resultado nos mostrará una etiqueta label de hiperenlace solo si existen mas nodos para navegar. Y se le asocia un enlace que es el siguiente nodo.

Mapear direcciones URL

Hemos visto anteriormente el problema que podemos tener al querer enlazar dos sitios a la misma página web (URL). No podíamos hacerlo porque los elementos del mapa del sitio se trata como una colección de URL's, y por la definición de las colecciones no pueden haber dos elementos con el mismo nombre.

ASP.NET proporciona una forma de asignar nombres a una dirección URL en el fichero web.config en las secciones:

```
<configuration>
```

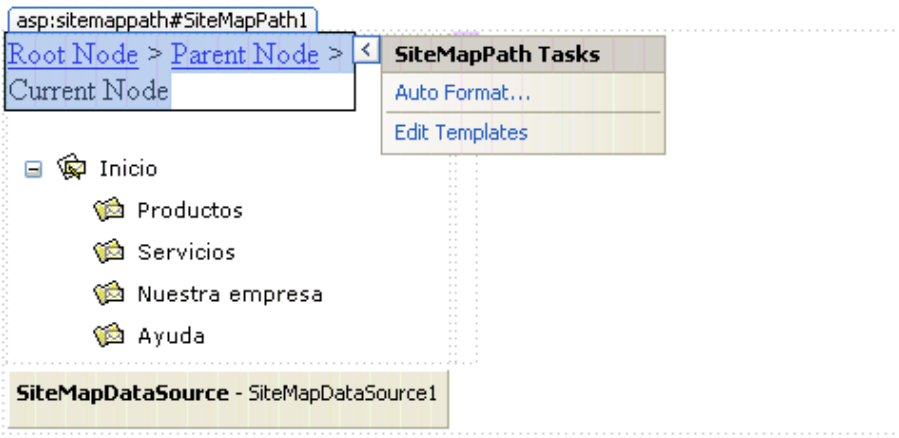
```
    <system.web>
        <urlMappings enabled="true">
            <add url="~/categorias.aspx" mappedUrl="~/default.aspx?opcion=categorias" />
            ...
         <urlMappings />
    </system.web>
</configuration>
```

Este tema también es un poco complejo y solo en situaciones un poco extrañas se nos podrá dar que tengamos que hacer uso de esto , pero ya ves, es fácil y rápido. Solo una cosa, distingue mayúsculas-minúsculas así que tendremos cuidado para que coincida exactamente.

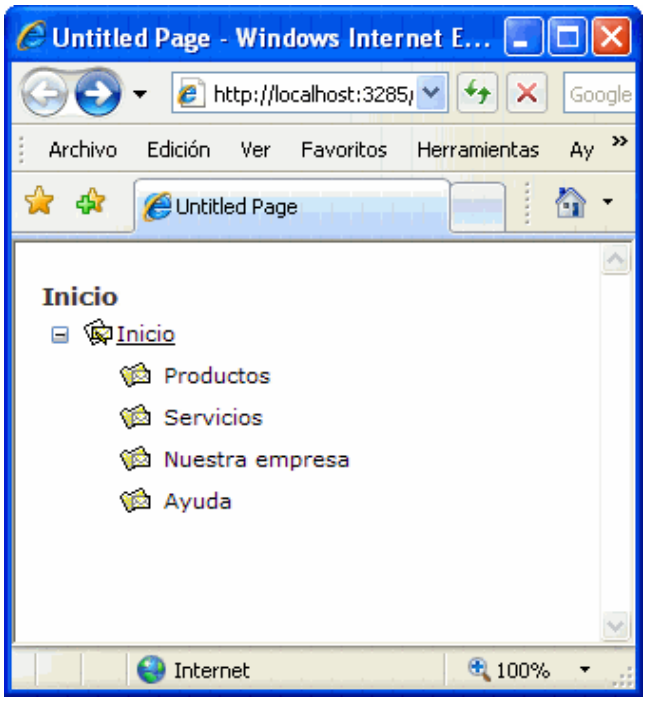
1.5 El control SiteMap

El control de árbol (TreeView) muestra las páginas disponibles pero no nos indica bien donde estamos situados dentro del sitio Web. Para poder tener esta situación vamos a complementar el control con otro muy sencillo llamado "SiteMapControl". Puesto que este control tiene ya información sobre la navegación porque se apoya en un origen de datos ya existente no tendremos que hacer prácticamente nada.

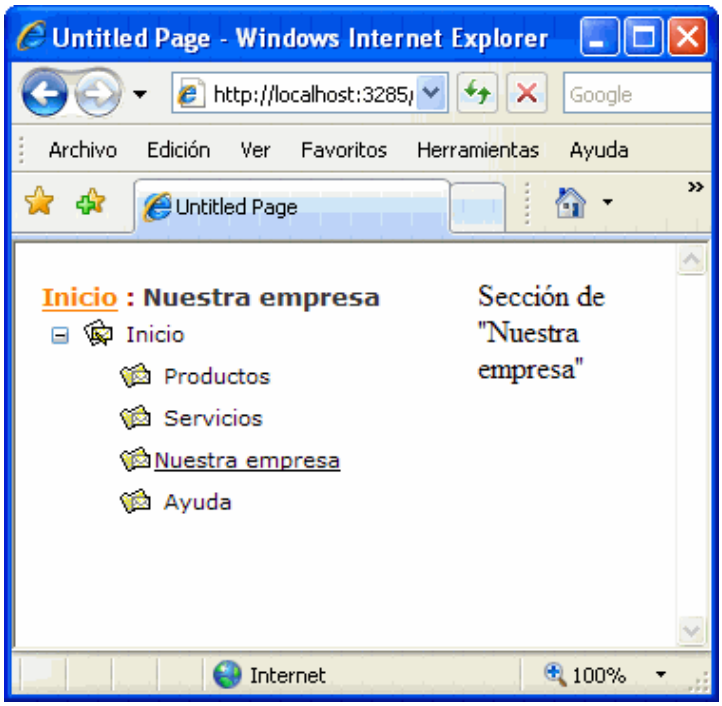
Vamos con nuestro ejemplo de páginas maestras con la navegación anterior, cambiando un poco el aspecto y poniendo este control, quedaría:



No tenemos que enlazar nada porque se enlaza automáticamente, así que si ejecutamos la página:



Vemos que aparece en la posición de arriba con el valor ý nivel de la opción de menú activa. Si navegamos a una página:



Como ves es una página de las que hemos puesto que se basara en la maestra, por eso aparecen los dos menús. Aunque aquí pierde sentido el menú del árbol porque el de arriba ya nos ofrece todo lo que queremos de una forma sencilla y rápida: nos identifica en que sección estamos y nos permite navegar a los niveles superiores. Por tanto para las páginas principales podemos poner menús de tipo árbol pero parece que para el resto de las página podíamos crear una página maestra con este control mas sencillo.

Las propiedades que ofrece este control son:

Propiedad	Descripción
ShowToolTips	Si está a "false" no muestra la ayuda la pasar el ratón por encima.
ParentLevelDisplay	Establece el máximo número de niveles que puede mostrar. Por defecto es -1, que significa que todos
RenderCurrentNodeAsLink	Si es cierto, el nodo del nivel actual se activa también como enlace.
PathDirection	El sentido de la jerarquía. RootToCurrent los muestra en orden. CurrentToRoot los muestra al revés, de donde está hasta la raíz.
PathSeparator	Indica el carácter que hará de separador entre los niveles.

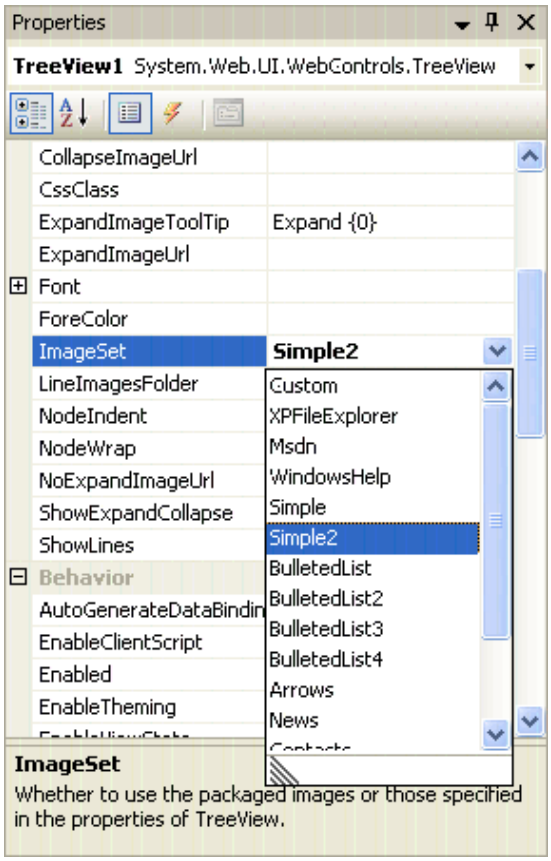
Plantillas y estilos

Si te has fijado, en este control hemos visto que tenemos también la opción de autoformato. Pero además de esto y puesto que estamos en el mundo de la versatilidad podemos poner los estilos y plantillas que queramos con estas propiedades:

Propiedad	Plantilla	Se aplica a
NodeStyle	NodeTemplate	Todas las partes excepto la raíz y el nodo actual
CurrentNodeStyle	CurrentNodeTemplate	El nodo representa la página actual
RootNodeStyle	RootNodeTemplate	El nodo representa la raíz. Si el nodo raíz es el mismo que el actual se utilizará el actual
PathSeparatorStyle	PathSeparatorTemplate	Separador en cada nodo

1.6 Control Treeview

Ya hemos visto que el control de árbol funciona mostrando sus nodos y una parte o toda de un sitio web. Si has curioseado con el autoformato habrás podido ver todos los formatos predefinidos que tenemos. No nos hacen falta mas la verdad, tenemos prácticamente todas las situaciones posibles, pero veamos que propiedades interesantes nos encontramos para que puedas curiosear con ellas. Por ejemplo las propiedades de "ImageSet" y la de "NodeIndent" producen unos cambios de aspecto radicales. Te recomiendo que las modifiques un poco para ver el grado de configuración tan amplio:



Ahora veamos que propiedades son las mas interesantes para nosotros...

Propiedad	Descripción
MaxDataBindDepth	Indica cuantos niveles se mostrarán. Por defecto es -1 que indica que todos los que se hayan especificado en el fichero XML Web.SiteMap. Si ponemos un 2, solo se mostrarán dos niveles.
ExpandDepth	Especifica cuantos niveles estarán expandidos al principio. Si es un 0 el árbol se mostrará completamente cerrado. Por defecto tiene -1 que significa que mostrará todos los niveles
PreviousSibling	Indica el número de píxeles entre los nodos de cada nivel.
ImageSet	Indica la imagen para mostrar. Podemos utilizar las predeterminadas o la indicada en CollapseImageUrl, ExpandImageUrl ó NoExpandImageURL. Para las imágenes del árbol según esté abierto o no
CollapseImageUrl, ExpandImageUrl ó NoExpandImageURL	Indican las imágenes para nodos cerrado, abiertos y para los que no tienen hijos.
NodeWrap	Permite poner el texto en varias líneas
ShowExpandCollapse	Muestra los cuadros de expandir-cerrar. No es recomendable porque al pulsar esta opción se fuerza un postback con la consiguiente recarga de la página
ShowLines	Añade líneas para conectar los nodos
ShowCheckBoxes	Muestra unas casillas de verificación. No es útil para árboles de navegación en el web pero si para otras ocasiones.

Por supuesto también podremos utilizar estas propiedades para los tamaños, distancias e imagen del nodo...

Propiedad	Descripción

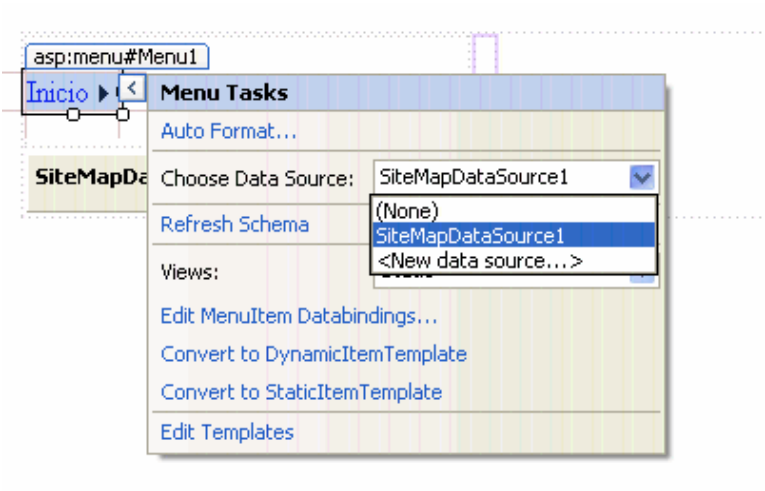
ImageUrl	La URL de la imagen para el nodo
NodeSpacing	Espacio entre píxeles entre los nodos
VerticalPadding	Espacio en píxeles entre la parte superior e inferior del texto de los nodos
HorizontalPadding	Espacio entre la parte izquierda y derecha del texto del nodo
ChildNodesPadding	Espacio entre el último nodo hijo expandido y el siguiente.

Juega un poco con todas esas propiedades y te darás cuenta que las opciones de personalización son tremendas. Y nos quedan los estilos que te comento también:

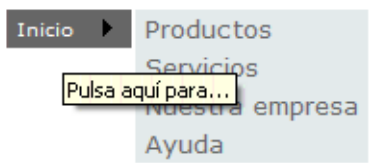
Propiedad	Descripción
NodeStyle	Se aplica a todos los nodos, los demás estilos se sobreescriben
RootNodeStyle	Se aplica solo al nodo del primer nivel
ParentNodeStyle	Se aplica a todos los nodos que contienen a otros, es decir los "padres" y no es el raíz
LeafNodeStye	Se aplica a los nodos que no tienen hijos y no es el raíz.
SelectedNodeStyle	Se aplica al nodo seleccionado.
HoverNodeStyle	Se aplica cuando el usuario pasa el ratón por encima.

1.7 El control Menu

Nos queda de ver un poco en detalle este último control de navegación. También soporta jerarquía y se enlaza de la misma forma que el anterior. Si ponemos uno en nuestra página y lo enlazamos con el "SiteMapDataSource" existente:



Y le aplicamos un autoformato elegante obtenemos inmediatamente un menú totalmente funcional:



No es muy diferente como puedes ver pero si que tiene algunas diferencias ya que el otro al basarse en nodos tiene un filosofía distinta, veamos las diferencias:

- El menú muestra un submenú sencillo. No como el árbol que puede expandir un número arbitrario de nodos a la vez
- El menú va destacando cada rama según se sitúa el ratón. El árbol muestra todo la rama expandida
- El menú soporta plantillas, los árboles no.
- Los árboles soportan casillas de verificación para cada nodo, el menú no.
- El menú se puede configurar de forma vertical y horizontal, el árbol solo se puede poner de forma vertical.

Estilos del menú

Los menús tienen sus propios estilos basado en los elementos de los menús con la clase "MenuItemStyle". Este estilo añade propiedades como "ItemSpacing", "HorizontalPadding" y "VerticalPadding" pero a diferencia del control anterior no dispone de imágenes.

La principal diferencia con el árbol es que los menús son dinámicos, al pasar el ratón por encima, se van mostrando los submenús. Tiene por tanto dos tipos de menú: estáticos como el de la raíz y dinámicos que son los mostrados de forma dinámica. Para proporcionar estas posibilidades la clase "Menu" define dos juegos paralelos de estilos unos para los estáticos y otros para los dinámicos:

Estilo estático	Estilo dinámico	Descripción
StaticMenuStyle	DynamicMenuStyle	Establece el aspecto del cuadro en el que está la opción del menú.
StaticMenuItemStyle	DynamicMenuItemStyle	Defina la apariencia de los elementos de menú
StaticSelectedStyle	DynamicSelectedStyle	Establece la apariencia del elemento seleccionado. Te en cuenta que el elemento seleccionado no es sobre el que se está pasando el ratón por encima, sino sobre el último que se hizo clic
StaticHoverStyle	DynamicHoverStyle	Aspecto cuando se pasa el ratón por encima.

Podemos también establecer estilos para los menús de primer nivel (LevelMenuItemStyles), segundo nivel (LevelSubMenuStyles) y para las opción seleccionada (LevelSelectedStyes). Igual que en el caso anterior es muy versátil y dejo a investigación describir todas las posibilidades visuales que ofrecen.

1.8 Crear controles de usuario

Este no es un tema exacto de navegación pero es suficientemente interesante como para que lo comentemos porque nos ayudará en el uso de los controles de navegación que hemos visto aquí.

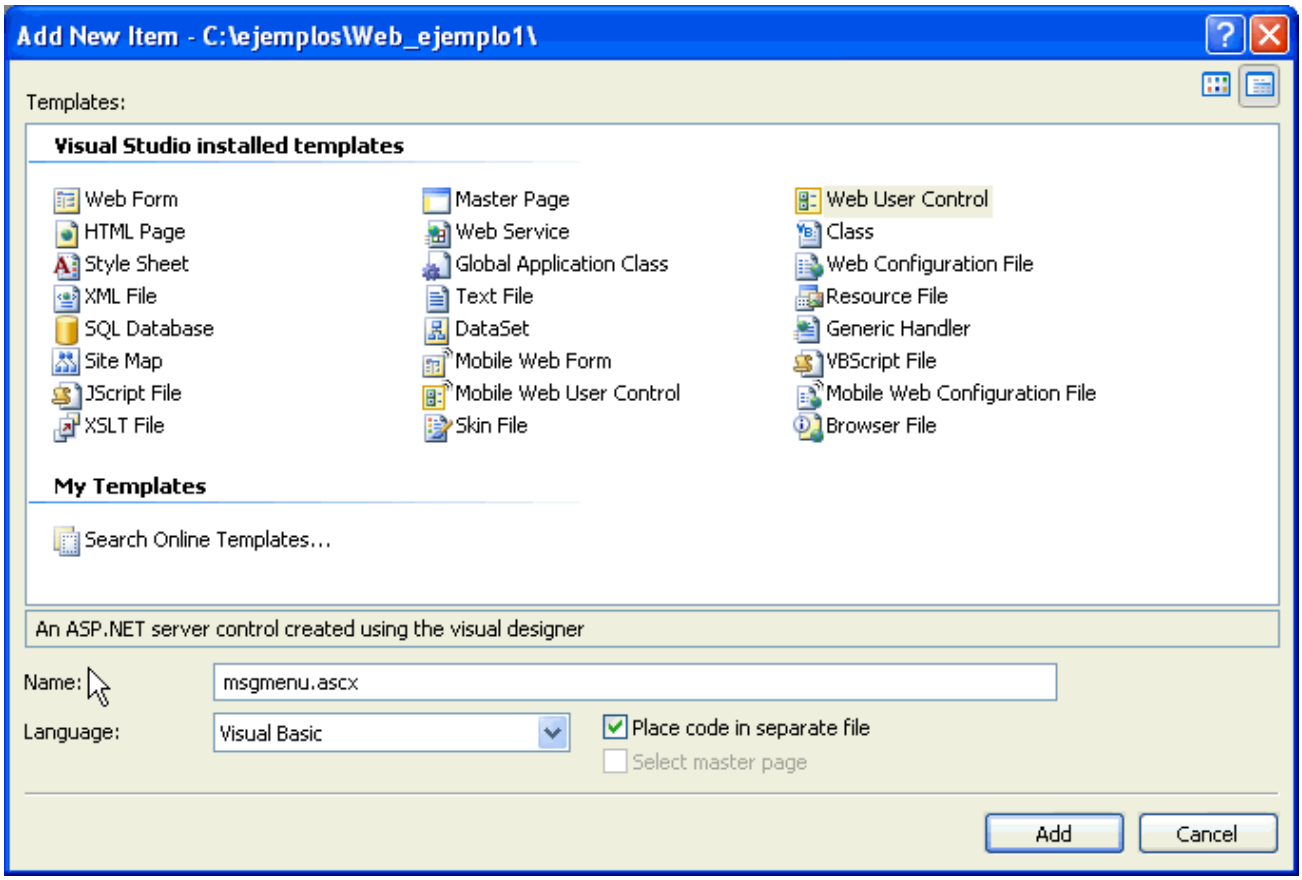
Ya comentamos antes que lo normal es no poner estas opciones de navegación en las páginas maestras sino que deben estar por ejemplo en las de primer nivel pero no en todas, es decir, es una decisión nuestra donde poner estos menús. Por ejemplo el último control que hemos visto no tiene sentido en una página de primer nivel porque no es necesario que lo indique.

Si colocamos un control de estos en una página debemos darle un formato y luego copiar ese control con exactamente el mismo estilo en las demás páginas. Si un día cambiamos de estilo debemos ir a todas esas páginas y cambiar el formato de esos controles. Para esto tenemos una sencilla solución: utilizar un control Web de usuario. Podemos poner el control anterior dentro de un control de este tipo. Así que cuando queramos poner ese control en las páginas no pondremos en realidad el de navegación sino este personalizado que nos mostrará el control. Es decir es este control de usuario es como un contenedor de otros controles, así

que lógicamente si lo ponemos en una página está poniendo también los que tiene definidos.

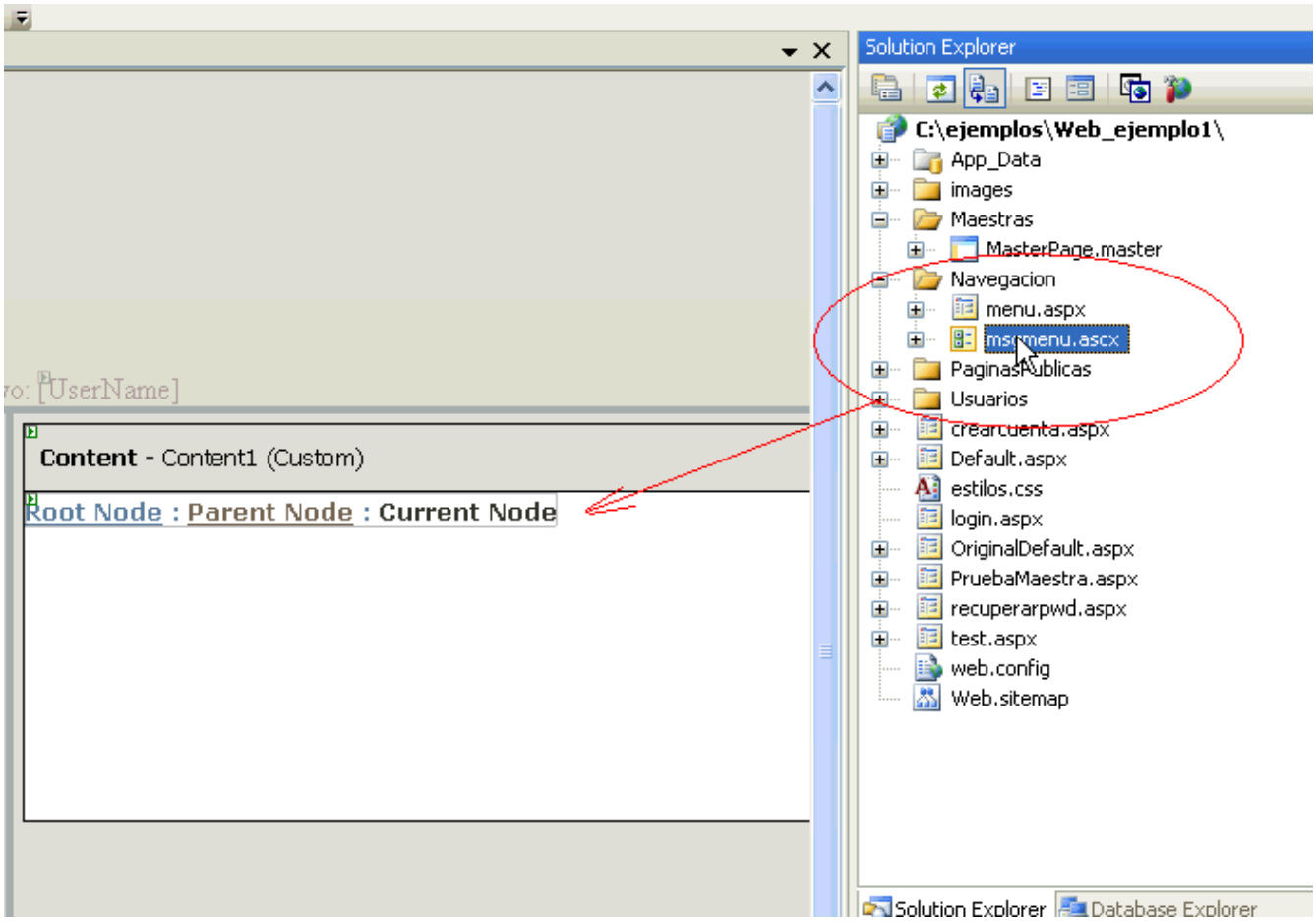
Con esto conseguimos que si un día cambiamos el formato de algo, solo debemos hacerlo en este control de usuario y como los demás están enlazados a este actualizaremos de una vez todas las páginas. Sencillo y fácil de implementar.

Para crearlo nos iremos a la opción de añadir una página a nuestro Web de tipo:

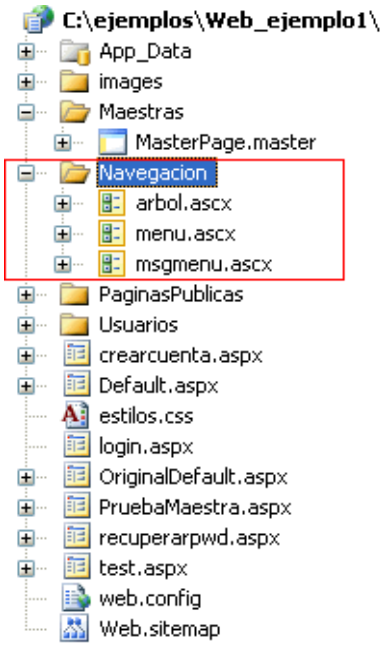


Es decir, de tipo "Web User Control" y le colocaremos el control del "SiteMapPath". El funcionamiento es igual que antes pero cuando pongamos un control en una página

Atención la diferencia ahora es esta: para ponerlo en las nuevas páginas en lugar de arrastrar un control de estos a la página lo que haremos es arrastrar la página nueva con el control de usuario que hemos creado, el resultado es el mismo pero el control está en esa página con todas las facilidades que da ahora para mantenerlo.



Como ves el icono es distinto y podías hacer lo mismo con las otras páginas de navegación para que te queden de esta forma:



Y así colocar los controles de usuario de navegación dentro de una carpeta, quedará mucho mas organizado y fácil de mantener. Recuerda que son estas páginas las que debes añadir y no los controles. Ya que esas páginas ya contienen los controles de navegación.

2. ADO.NET

Ahora vamos a realizar una introducción al componente de servidor mas importante incluido en nuestra instalación de ASP.NET: es el denominado componente de acceso a bases de datos.

El componente de acceso a bases de datos, es en realidad un conjunto de objetos que se agrupan dentro de la denominación ActiveX Data Objects (ADO.NET). Mediante este el conjunto vamos a tener acceso a bases de datos, es decir, nos va a permitir realizar conexiones a bases de datos, ejecutar sobre ellas sentencias SQL, procedimientos almacenados, obtener resultados, etc. Nos va a ofrecer desde nuestras páginas ASP.NET todas las operaciones posibles que se pueden realizar con una base de datos.

En esta primera parte del capítulo verás una buena cantidad de nombres que puede te desconcierten un poco, mas que nada por la cantidad de objetos. Una vez superado esto verás que siempre se utilizan las mismas instrucciones para acceder a las bases de datos y teniendo una buena colección de ejemplos, la manipulación de datos es muy sencilla.

Para los que vienen de ASP o de la anterior versión ADO os encontraréis con una buena cantidad de cosas que no existían antes y que son muy interesantes y en cuando a controles de servidor hay verdaderas maravillas para crear páginas de una forma rápida y con unos resultados verdaderamente buenos.

Para este curso de introducción no es estrictamente necesario ver tanto contenido de ADO.NET ya que el programa que veremos a partir del próximo capítulo nos va ayudar mucho con asistentes, pero nos quedaríamos sino sin conocer los detalles de su funcionamiento. Por otra parte cuando comiences con el curso avanzado te pedirá que tengas ciertos conocimientos de ADO.NET así que adelante y a leerlo con tranquilidad. Esta tecnología se aplica al mundo .NET así que si haces un programa de Visual Basic.NET todo esto lo tienes ya ganado...

Definición

ADO.NET es pues el nombre de un grupo de objetos proporcionados por .NET Framework para interactuar con datos almacenados. Con ADO.NET podemos trabajar con muchos formatos de datos, por ejemplo podremos conectarnos con estos orígenes de datos:

- Bases de datos empresariales como Oracle, Microsoft SQL Server, IBM DB2, ...
- Bases de datos personales como MS Access
- Ficheros y directorios como FAT32
- Ficheros de texto de longitud fija o delimitados por comas
- Hojas de tipo Excel
- Datos de MS Exchange
- Datos basados en XML

Lo mejor de todo es que esto no es fijo, la estructura de ADO.NET se basa en una serie de "proveedores" de información, controladores (drivers) y adaptadores que pueden crear y manipulas las bases de datos. Es decir, el formato final de los datos es indiferente a la hora de escribir el código ADO.NET. Primero estableceremos una conexión con el origen de datos indicando de que tipo es la base de datos y a partir de ahí los objetos y sintaxis son prácticamente idénticos en todos los casos. Esto hace que se muy sencillo por ejemplo pasar los

datos de MS Access a SQL Server, sólo haría falta cambiar la conexión y el código sería el mismo.

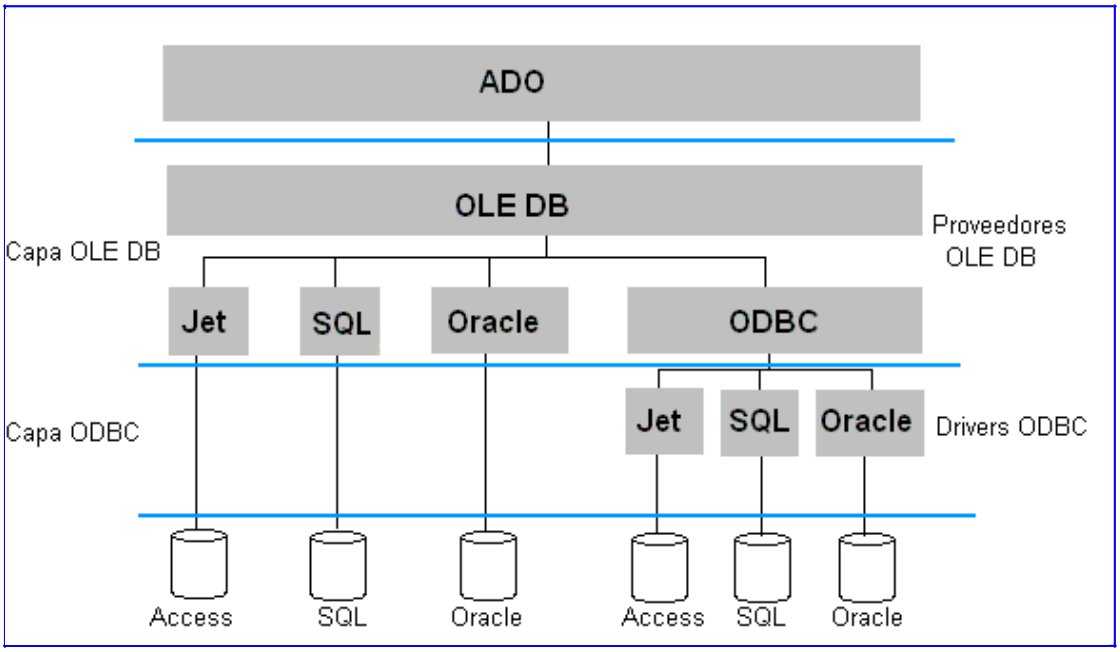
2.1 Administrar proveedores

Esta claro que para obtener datos de un origen debemos establecer una conexión. Antes en las conexiones con ADO se utilizaban los "Proveedores OLEDB" y los "Controladores ODBC" que eran los que establecían la conexión con el servidor de base de datos. En ADO.NET disponemos de un administrador de proveedores de datos para hacer esta operación.

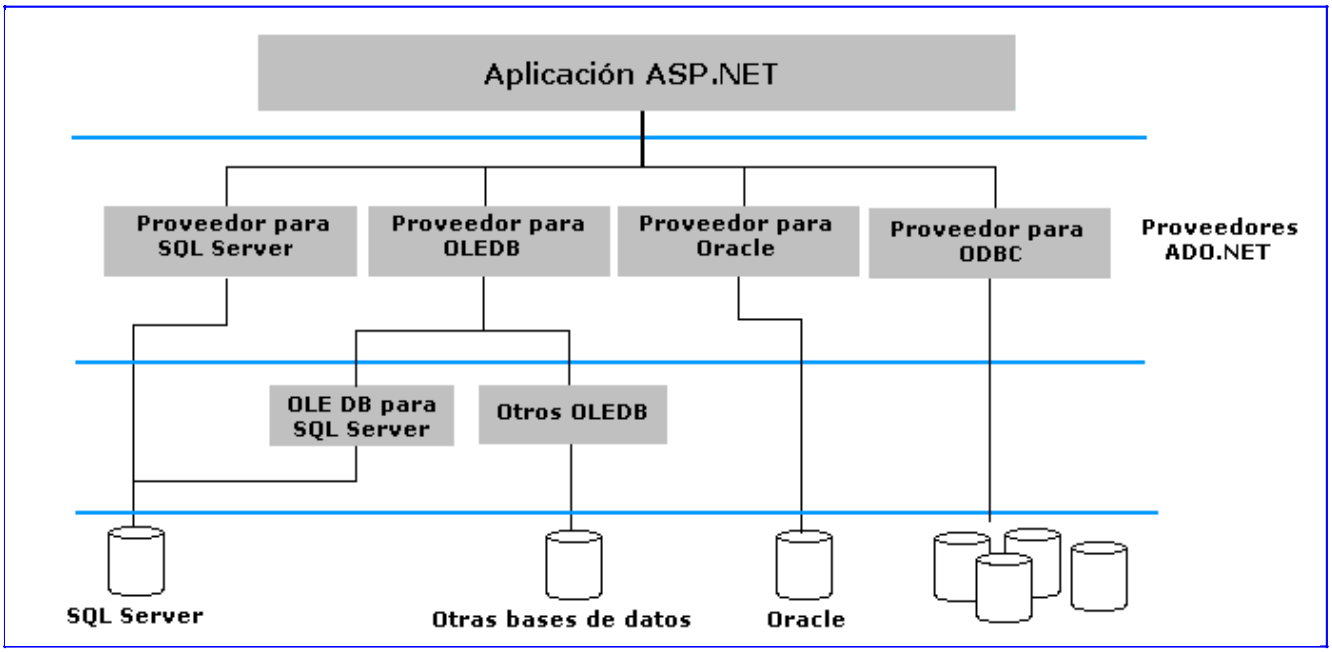
Con ADO.NET se proporcionan dos proveedores:

- Proveedor para SQL Server, sólo trabaja para la base de datos de Microsoft SQL Server.
- Proveedor para OLEDB, para los orígenes de datos OLEDB, que son el resto de las bases de datos.
- Proveedor para Oracle
- Proveedor para base de datos ODBC

La estructura de los controladores de la antigua tecnología ADO era:



Y en ADO.NET pasa a ser:



Por un lado vemos un controlador exclusivo para SQL Server y otro para Oracle, pero luego tenemos uno "Universal" que se conecta con cualquier base de datos que soporte OLEDB (prácticamente todas) y además uno para las antiguas bases de datos ODBC. Además hay una conexión entre OLEDB este último y SQL Server, es decir podríamos definir en la conexión que sea una OLEDB y el destino SQL Server, estaría bien y nos daría buen rendimiento pero Microsoft quiso dar un juego distinto para estas bases de datos de servidor y les ha proporcionado un proveedor exclusivo para ellos. El problema está en la sintaxis, pero verás que las diferencias son mínimas.

Nota La muy antigua tecnología ODBC para acceso a datos también tiene su sitio, hay un controlador para estas bases de datos y su utilización sería la misma, pero dada su antigüedad no la reflejamos en esta documentación. Si posees alguna base de datos con este método de conexión documéntate porque seguro que el fabricante proporciona el acceso mediante OLEDB. En este caso ya sería compatible con nuestro esquema.

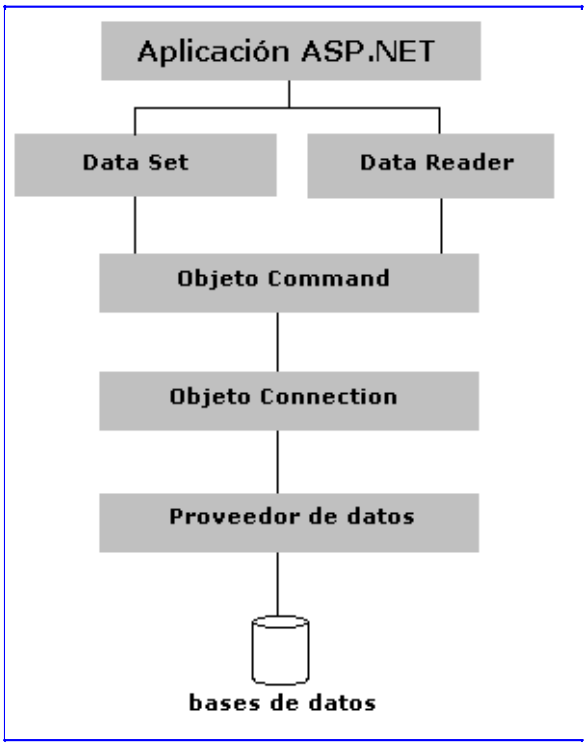
Microsoft tiene un programa que engloba todos los objetos de acceso a datos y controladores OLEDB, se llama Microsoft Data Access Components (MDAC) y está disponible para descarga gratuita desde Microsoft: http://msdn.microsoft.com/data/Default.aspx (observa que la extensión de la página es 'aspx')

La última versión disponible y recomendable es la 2.8

2.2 Objetos en ADO.NET

Llega el momento de ver los objetos que tenemos disponibles para trabajar con bases de datos. Sobre todo y como te he dicho antes, lee con calma y quédate con los conceptos generales, cuando hagamos ejemplo será todo mas sencillo.

El punto de partida del acceso a datos es la conexión con el proveedor de datos seleccionado, esta operación la hace el objeto Connection. El flujo de información podría verse de esta forma:



Visto de arriba hacia abajo, nuestro programa quiere hacer una consulta para rellenar una tabla de datos en pantalla ("dataset" ó "datareader") para esto utilizará un comando "command" que accederá a los datos mediante la conexión "connection". Ésta última ya sabe cómo y donde debe realizar la consulta.

Para poder trabajar con datos en nuestras páginas web necesitaremos 6 elementos, puede haber alguna variación en algún caso especial, pero hazte idea que estas son las seis partes que necesitaremos para acceder a datos:

- Origen de datos, es decir los datos en el formato admitido: SQL Server, Access, Oracle, ...
- Proveedor para acceder a esa base datos. Recuerda que ya tenemos para las bases de datos OLEDB y SQL Server. Las OLEDB son prácticamente todos las "pequeñas": Access, ...
- El objeto "Connection" de ADO.NET. La conexión de con los datos anteriores
- El objeto "Command", contiene instrucciones de cómo leer datos y otras operaciones
- Los objeto "DataReader" y "DataSet", son los datos que estamos leyendo o manipulando
- Los controles de servidor ASP.NET. Objetos que se ejecutan en el servidor (runat=server) y muestran los resultados. Por ejemplo, el mas utilizado: la cuadrícula <asp:DataGrid>

Como ves es bastante lógico: primero debemos saber dónde están los datos y tener el controlador para poder conectarnos a esos datos. Luego establecemos una conexión y ejecutamos una serie de comandos. Estos comandos almacenan los resultados en unos objetos que finalmente exponemos en una cuadrícula ASP. Lógico todo ¿no?

Como te digo, de momento tranquilo, estamos aprendiendo todos los conceptos de ADO.NET que para tu suerte al final serán muy pocos los necesarios, y sobre todo, con una buena colección de ejemplos tendremos todas las posibilidades que necesitemos y que nos servirán como punto de partida de nuevas aplicaciones.

2.3 Espacios de nombres

Los componentes de ADO.NET los tenemos en distintas bibliotecas de clases y todas juntas conforman ADO.NET. Fíjate en los espacios de nombres (Namespace) que tenemos disponibles en ADO.NET

Namespace	Función
System.Data	Contiene las clases fundamentales del núcleo de ADO.NET. Incluye los objetos "DataSet" y "DataRelation" para modificar la estructura relacional de los datos. Estas clases son independientes del tipo de controlador que se utilice.
System.Data.Common	No se utilizan en nuestro código. Las utilizan otras clases proveedoras de datos.
System.Data.Oledb	Contiene las clases necesarias para conectarnos a orígenes de datos OLE DB, y la ejecución de comandos: "OleDbConnection" y "OleDbCommand"
System.Data.SqlClient	Contiene las clases para conectarnos y ejecutar comandos en un servidor de base de datos SQL Server. Las propiedades y métodos de "SqlConnection" y "SqlCommand" son las mismas que cuando la bbdd es de tipo OLEDB con los objetos anteriores.
System.Data.SqlTypes	Contiene estructuras para tipos de datos específicos de SQL Server, como SqlMoney y SqlDateTime. De esta forma no hay que convertir tipos de datos.
System.Data.OracleClient	Contiene las clases para conectarnos y ejecutar comandos en un servidor de base de datos Oracle: "OracleConnection" y "OracleCommand
System.Data.ODBC	Contiene las clases para conectarnos y ejecutar comandos a través de un controlador ODBC: "OdbcConnection" y "OdbcCommand

2.4 Clases del proveedor de datos

Aunque tienen distintos espacios de nombres las propiedades y métodos son prácticamente iguales, tanto que incluso podríamos cambiar sólo los nombres de los objetos que vamos a ver ahora por su equivalente de SQL Server a Oracle y nuestra aplicación funcionará con normalidad. Veamos las clases de los proveedores de nombres:

Proveedores -->	SQL Server	**OleDB**	**Oracle**	**ODBC**
Connection	SqlConnection	OleDBConnection	OracleConnection	OdbcConnection
Command	SqlCommand	OleDBCommand	OracleCommand	OdbcCommand
DataReader	SqlDataReader	OleDBDataReader	OracleDataReader	OdbcDataReader
DataAdapter	SqlDataAdapter	OleDBDataAdapter	OracleDataAdapter	OdbcDataAdapter

La diferencia real entre ellas son los nombres, la cadena de conexión y algunas características avanzadas que pueden ofrecer algunos de ellos.

3. Administración de bases de datos

La manera lógica da manipular datos es obviamente una base de datos. Desde el almacenamiento de unos usuarios con sus contraseñas hasta la lista de artículos de una tienda. Esta información suele estar relacionada, es decir mantiene una coherencia de datos y comparten información: ventas a clientes, compras a proveedores... Las bases de datos tienen cada una su formato particular y su tecnología. En estos ejemplos veremos la utilización de MS Access por su facilidad y la utilización de SQL Server por su potencia. Ni que

decir tiene que SQL Server es un servidor de base de datos y por tanto muy potente. En cambio Access es una base de datos personal incluida con Office con muchas limitaciones. Si utilizamos esta última será para pequeñas operaciones en nuestro proceso de aprendizaje. Para cualquier aplicación web, por sencilla que sea, debemos utilizar SQL Server.

Aquí veremos la parte básica de SQL, recuerda que en Teleformación tienes un curso de SQL Server y un curso de ASP.NET avanzado donde se profundiza mas sobre estas cosas. Podríamos utilizar otras bases de datos teniendo el controlador adecuado: Oracle, MySQl, ... pero por coherencia y facilidad nos centraremos en SQL Server y algún sencillo ejemplo de Access.

SQL Server 2005 es un sistema de administración de bases de datos relacionales. Esta es una de las claves: las relaciones entre las tablas. Muy pocas veces nos encontraremos con bases de datos donde las tablas están aisladas y no tienen relaciones entre sí. Cualquier pequeño problema que te plantees con toda seguridad necesitará de varias tablas relacionadas para obtener una información coherente. Por ejemplo, imagina nuestra cuenta del banco, tiene una serie de datos personales: nombres, dirección, ... Habitualmente utilizamos la tarjeta para realizar toda clase de operaciones que obviamente se registrarán una base de datos. Estos movimientos o transacciones no almacena nuestros datos completos nombre, dirección, transacción realizada e importe. Sino que almacena sólo nuestro número de cliente que tenemos asignado en nuestra cuenta. Por tanto al sacar dinero solo almacenará: número de cliente, tipo de transacción, cantidad y fecha de la operación. Por tanto estamos relacionando por el número de cliente dos tablas: una para nuestros datos y otra para las operaciones.

Visto este ejemplo, vamos con otro mas práctico para nuestras pruebas. La forma mas natural de hacer las relaciones es de "uno a muchos":

- Un usuario puede comprar muchos elementos
- Un elemento es comprado por muchos usuarios

En el momento que tenemos dos relaciones de estas se convierte ya en tipo de "muchos a muchos": muchos usuarios compran muchos elementos.

Veamos de forma abreviada una tabla de usuarios y una de unos artículos que forman parte de un catálogo:

UserId	UserName
905682e6-f67b-...	antonio
a18fc8bf-ec1e-4...	jose2
1f0b3846-8e72-...	Josemari
1299e1eb-968e-...	antonio2
ff45a218-f080-4...	jose

ArticuloID	Nombre
1	Introducción a Visual Basic
2	Los For-Terrier
3	Fotografia
4	Fotografia II
5	ASP.NET

La tabla de la derecha muestra unos libros que forman parte de un catálogo de una tienda. En la de la izquierda la lista de los usuarios que son los posibles compradores. Ahora viene lo "complicado"... ¿cómo conectamos esas dos tablas? Quiero gestionar de alguna forma que esas personas que tengo en una tabla de usuarios puedan realizar compras de los libros de la tabla de la derecha.

La respuesta es "creando una tabla que almacene las compras realizadas". Cada vez que un usuario realice una compra debemos almacenar esta información por lo menos:

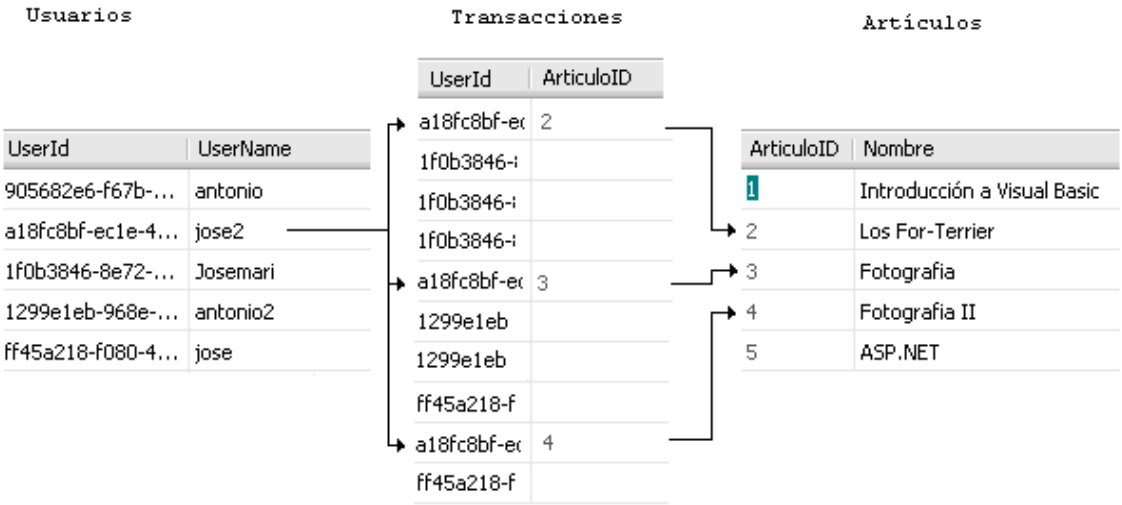
- Usuario
- Libro
- Fecha de compra

- ¿Está pagado?

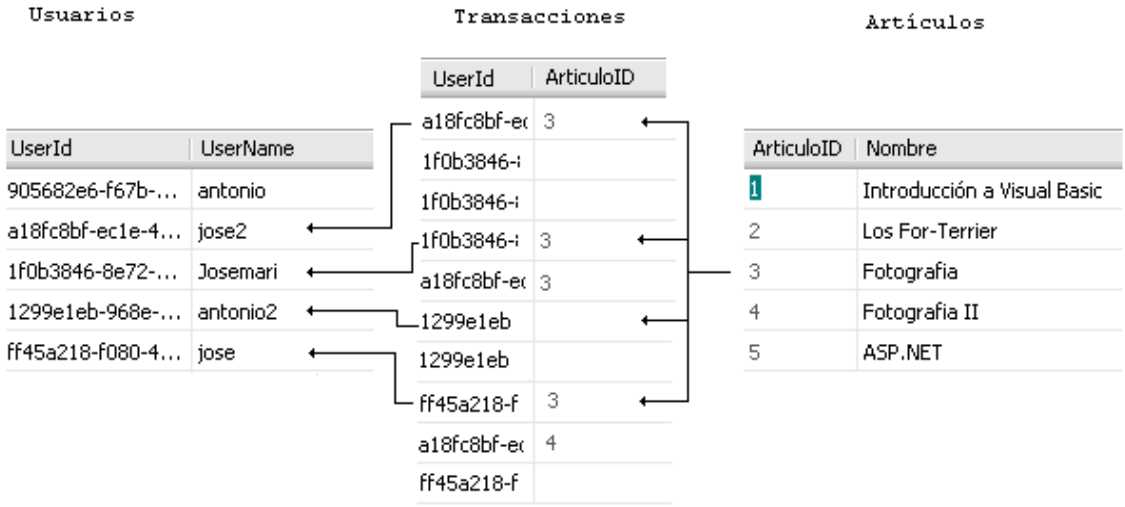
Así que debemos almacenar los identificadores de esas dos tablas. No es necesario que almacenemos el nombre y apellidos ni el título del libro no esos datos, simplemente almacenaremos el valor del identificador del usuario "Userid" y del libro "ArticulosId". Estos identificadores son únicos y apuntan a una sola fila de cada tabla, luego son perfectos para almacenar esta información. Nos ahorramos además toda esa información de usuarios y libros que no hacen falta para almacenar nuestra "transacción".

Luego, a la hora de hacer un informe que querer sacar la lista simplemente en lugar de poner esos indicadores los sustituiremos por una consulta en las tablas de los usuarios y artículos, de esto último se encarga el propio lenguaje de consulta. Como ves el contenido del identificador del usuario es un poco raro, no parece numérico, pero no te preocupes, es un tipo de datos especial que veremos mas adelante.

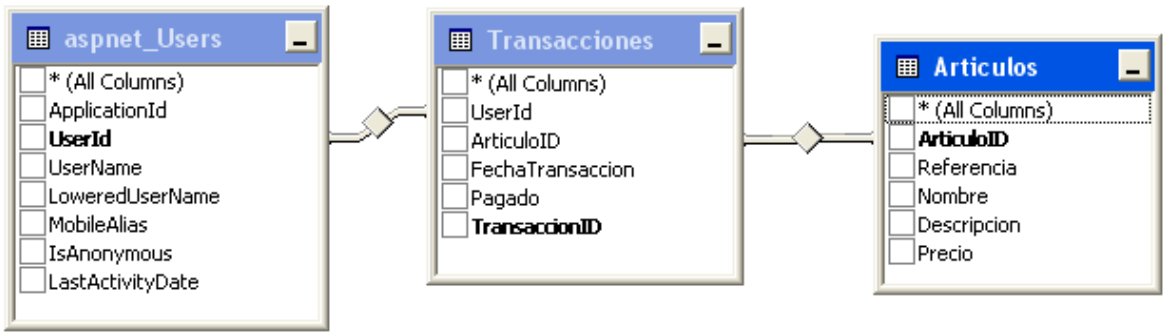
Así que en cada compra que hagamos meteremos una línea en la tabla de transacciones indicando quien y que ha comprado. Aquí la relación es "muchos a muchos" ya que muchos usuarios pueden comprar el mismo libro e igualmente muchos libros pueden ser comprados por un usuario.



Ahí vemos como el usuario "XXX" ha hecho 3 transacciones con 3 libros distintos. Y al revés:



A la hora de diseñar las tablas y estableces esas relaciones artículos-transacciones-usuarios utilizaremos las herramientas de diseño de bases de datos que nos muestran algo parecido a esto:



Para hacer relaciones las dos tablas deben tener un campo en común que tenga el mismo tipo de datos. Si nuestra tabla de transacciones tiene una lista de artículos, esos identificadores deben ser del mismo tipo de datos que los identificadores de la tabla de artículos. Al estar relacionadas se establece una interesante coherencia ya que si borramos un usuario borraremos todas las transacciones de ese "id" de la tabla correspondiente y ya no quedará información aislada.

El acceso a una base de datos desde una aplicación web es totalmente distinto que el realizado en una aplicación Windows. Esto ya lo podíamos imaginar, un usuario con su programa Windows realiza una consulta y la recibe en su formulario en una cuadrícula u otros controles. En nuestra aplicación web tendremos que construir la consulta en el servidor, que la ejecute y que nos devuelva la página correspondiente. Si esto lo multiplicamos por todos los usuarios simultáneos tendremos un uso masivo de estos accesos a datos en los que tendremos que cuidarnos en el diseño para que todo vaya a la velocidad que debe ir.

Imagina además la desconexión de un usuario, donde finaliza su famoso "state". Está claro que debemos finalizar las consultas que tuviera en proceso para no gastar memoria y tiempo de CPU del servidor.

3.1 Configurar la base de datos

Lo primero que debemos hacer antes de trabajar con datos es configurar la conexión a la base de datos y de paso realizar alguna conexión de prueba para ver si es correctamente accesible por nuestro servidor. En la

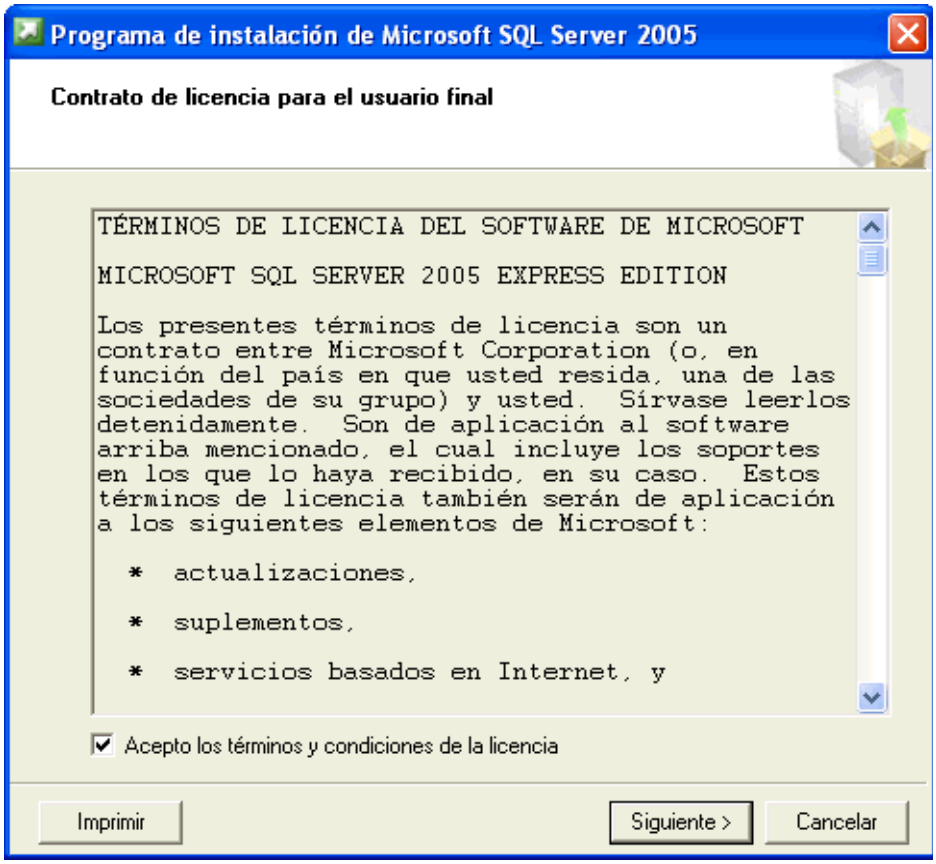
instalación de nuestro Visual Web Developer 2.008 le indicamos en el capítulo 1 que nos instalara la ayuda y el SQL Server 2.008 Express. Este último componente será nuestra base de datos. Nos puede vale en un servidor para una pequeña explotación y para pruebas pero para uso final debemos instalar una versión competa de SQL Server. Si no la instalaste en su momento o quieres instalarla en otro servidor, haremos estos pasos:

Como hemos dicho, dentro de las versiones que incorpora SQL Server hay una gratuita que se llama SQL Server Express Edition. Así que de momento descartamos Access como base de datos por dos cosas: la perfecta integración de esta versión de SQL Server con ASP.NET 3.5 y la segunda y mas importante, Access no es un servidor de base de datos, sólo es una base de datos personal de muy limitado rendimiento y no pensada como motor de servidor. Así que el primer paso es instalar este SQL Server en nuestro equipo...

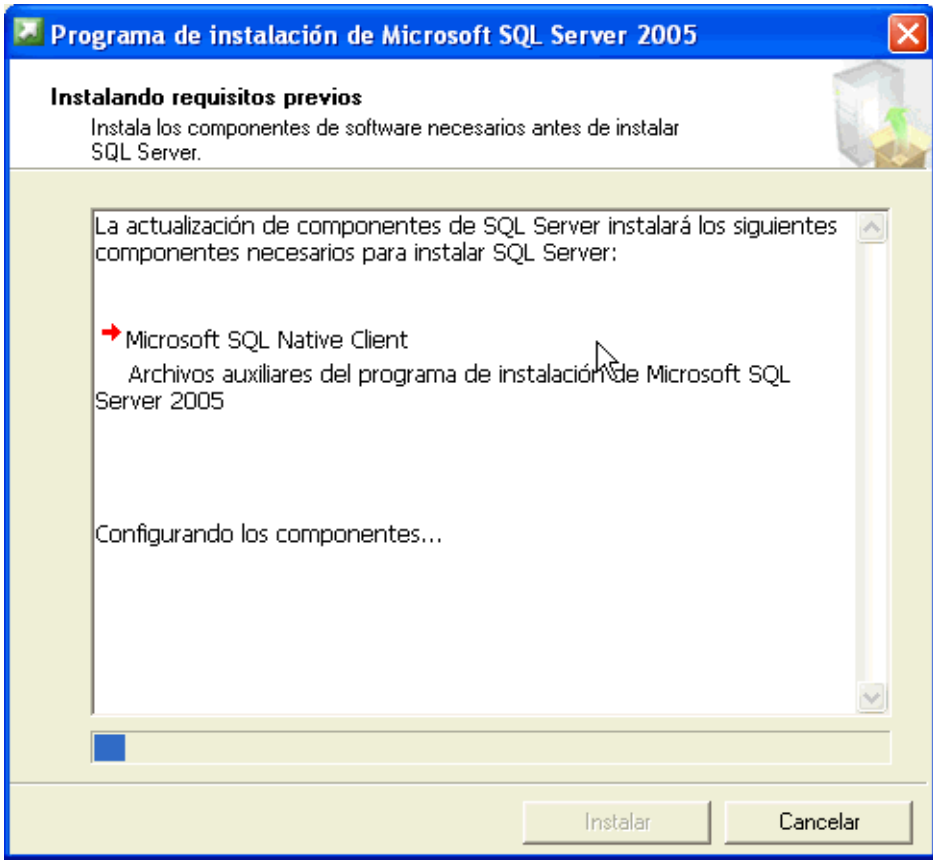
Instalar SQL Server 2005 Express Edition

Para instalarlo nos iremos a la Web de Microsoft en este enlace: http://msdn.microsoft.com/vstudio/express/sql/download/ Y nos vamos al enlace inferior donde pone "Download", luego en la siguiente pantalla:

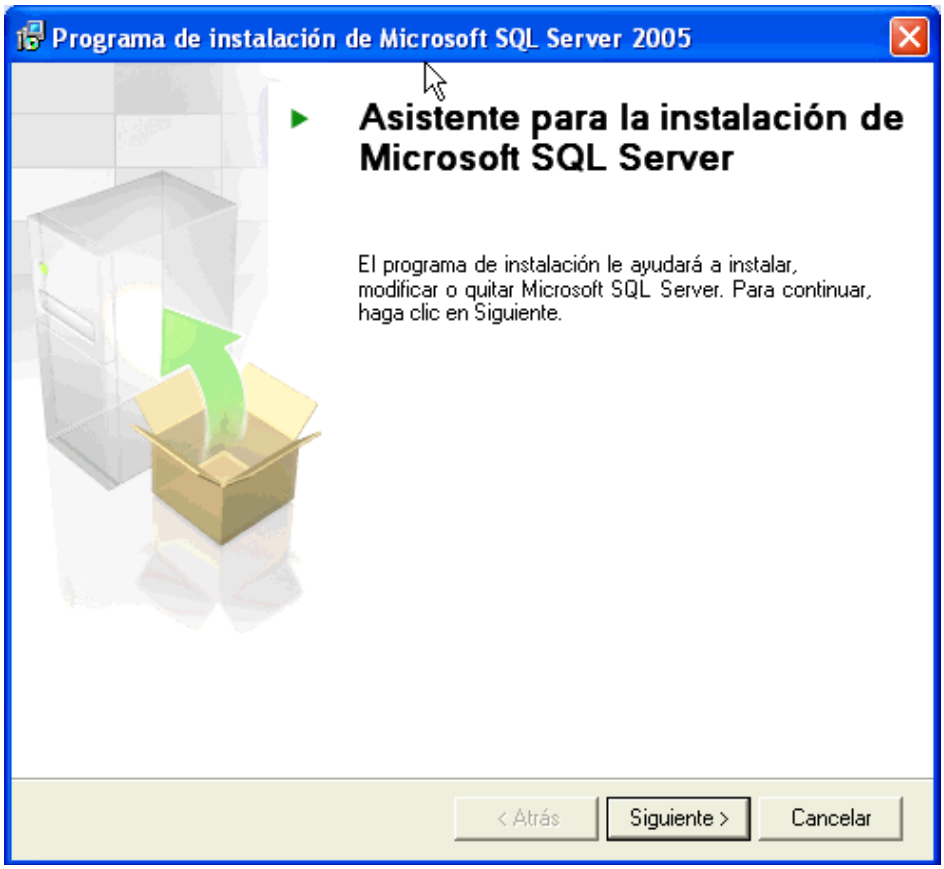
Cambiamos el idioma y descargamos el programa para luego darle a instalar:



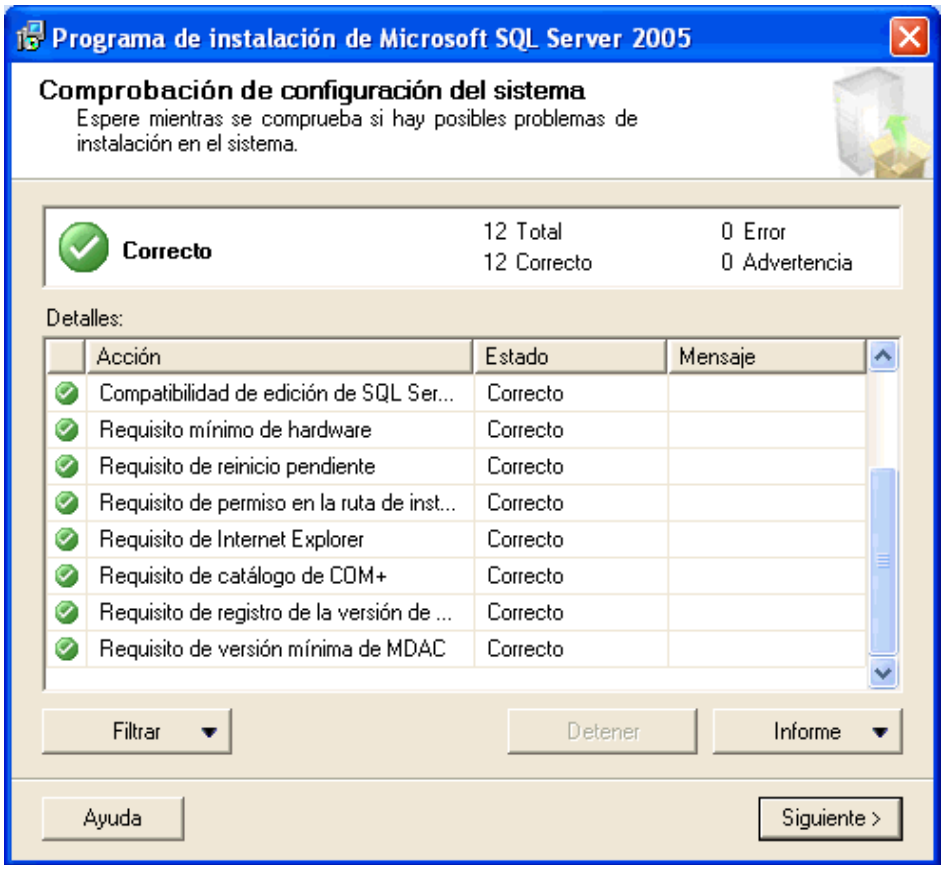
Y comenzará la preinstalación.



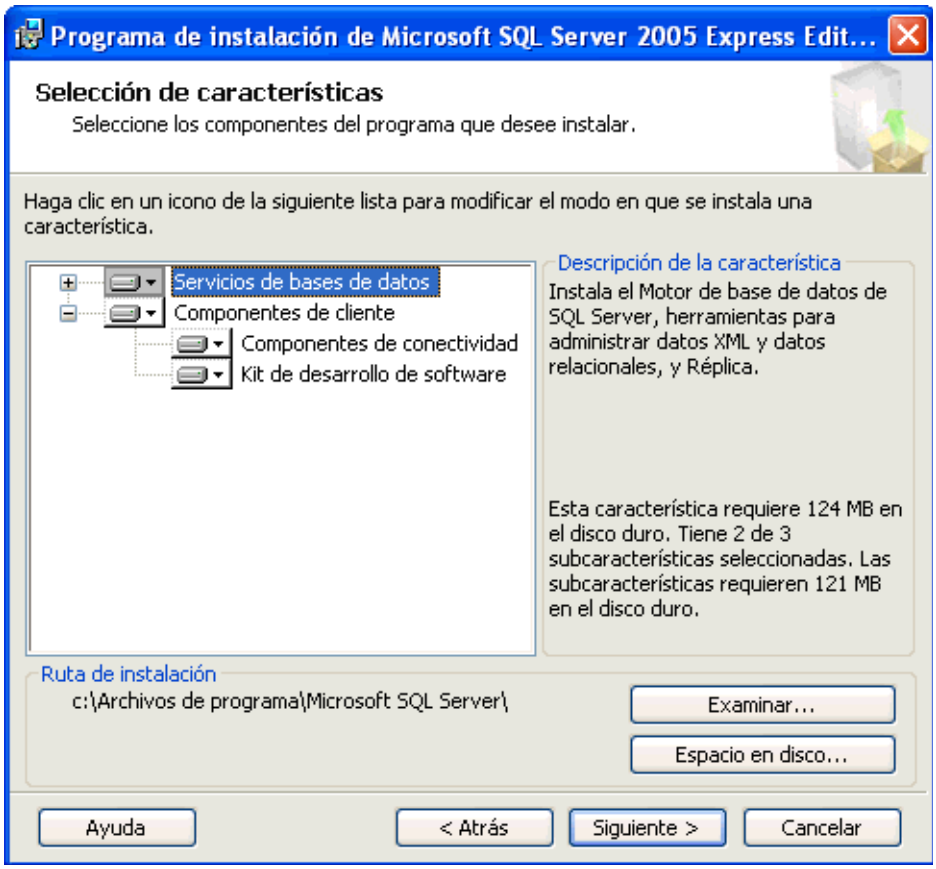
Una vez terminada esta preparación el programa continuará con la instalación "de verdad":



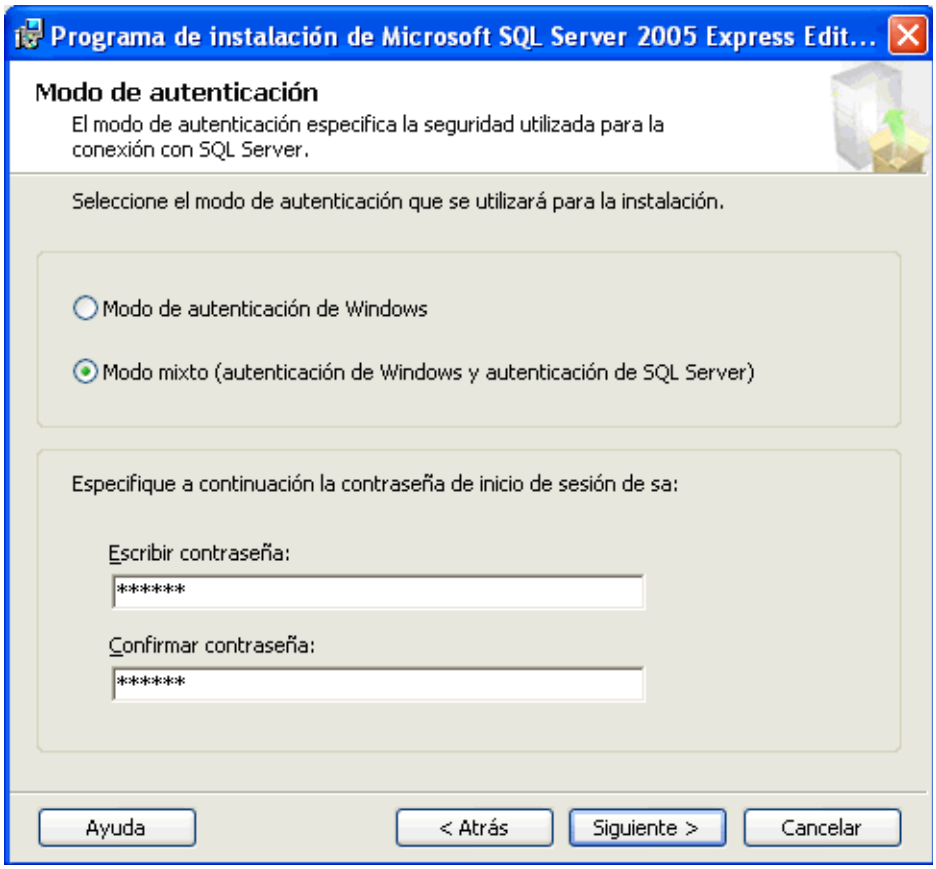
Omito las pantallas menos importantes ya que mostrará alguna mas de información sin importancia. Comprobará primero el sistema para ver si cumple con los prerequisitos:



Y después ya nos pide que componentes queremos instalar...



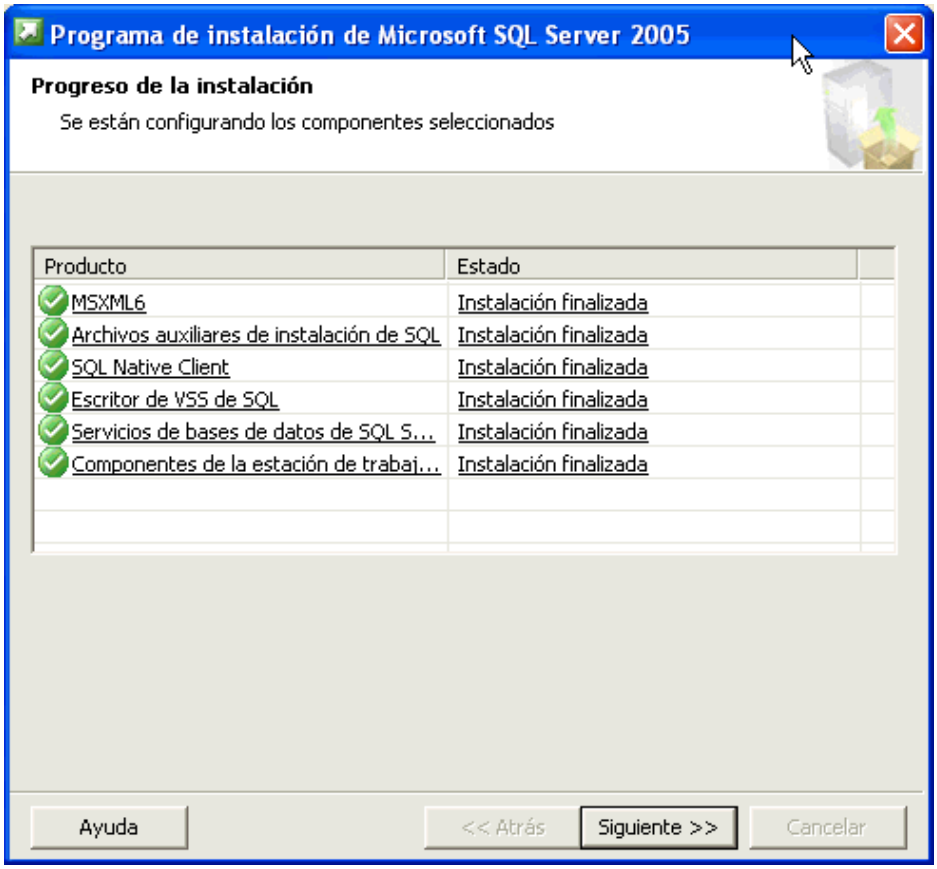
Marcaremos todas las opciones para que así nos instale las base de datos de pruebas llamada "pubs". Y ahora el tipo de autenticación.



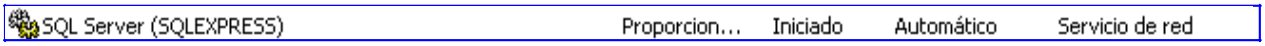
Te explico esto un poco mas... Hay dos formas de autentificarse o autenticarse, una mediante lo usuarios de un directorio activo de nuestra red o mediante una base de datos de usuarios del propio SQL Server. Lo habitual es utilizar los dos tipos de autenticación ya que puede que creemos usuarios que puedan acceder a esta base de datos y que nos son usuarios de nuestro dominio. Así que seleccionaremos la "mixta".

¿Por qué no hace falta contraseña en el primer modo? Muy sencillo, si los usuarios de SQL Server van a ser los mismos que el dominio el grupo de dominio de "Administradores" tiene automáticamente los derechos de administradores sobre el SQL Server. En el caso de la mixta el servidor crea una cuenta llamada "sa" (superadministrador) y lo que nos está pidiendo es esos cuadros es la contraseña de inicio de sesión de "sa". Podremos una y guárdala bien!

Pulsamos en "siguiente" para finalizar por fin la instalación, y después de un rato...

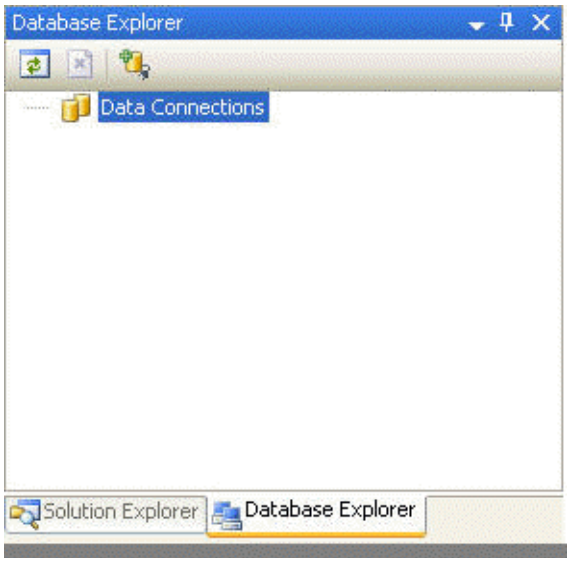


Que nos avisa de que está todo terminado... ahora vete al panel de control y en herramientas administrativas vete a "Servicios" para que veas lo que nos ha instalado:

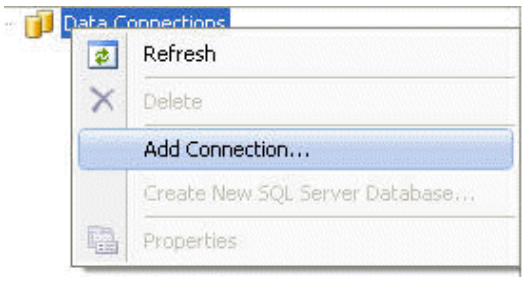


Bien, el primer paso terminado. Ahora vamos a hacer "browser" o visualizar las bases de datos que tenemos en este SQL Server. Crea una página en blanco y atento a los pasos, es sencillo pero debes seguirlos ordenadamente.

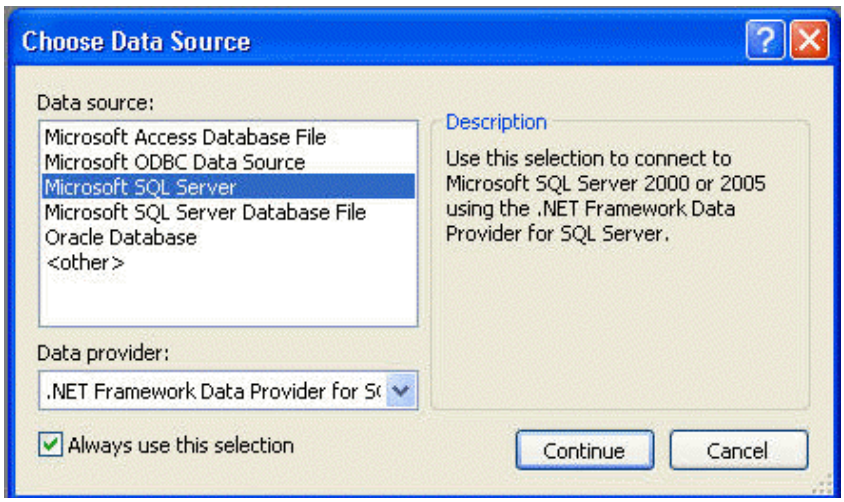
Hacemos clic en el "Database Explorer", que lo tienes como una solapa adicional en el explorador de soluciones o de proyectos. Si no te aparece la activas en la vista "view":



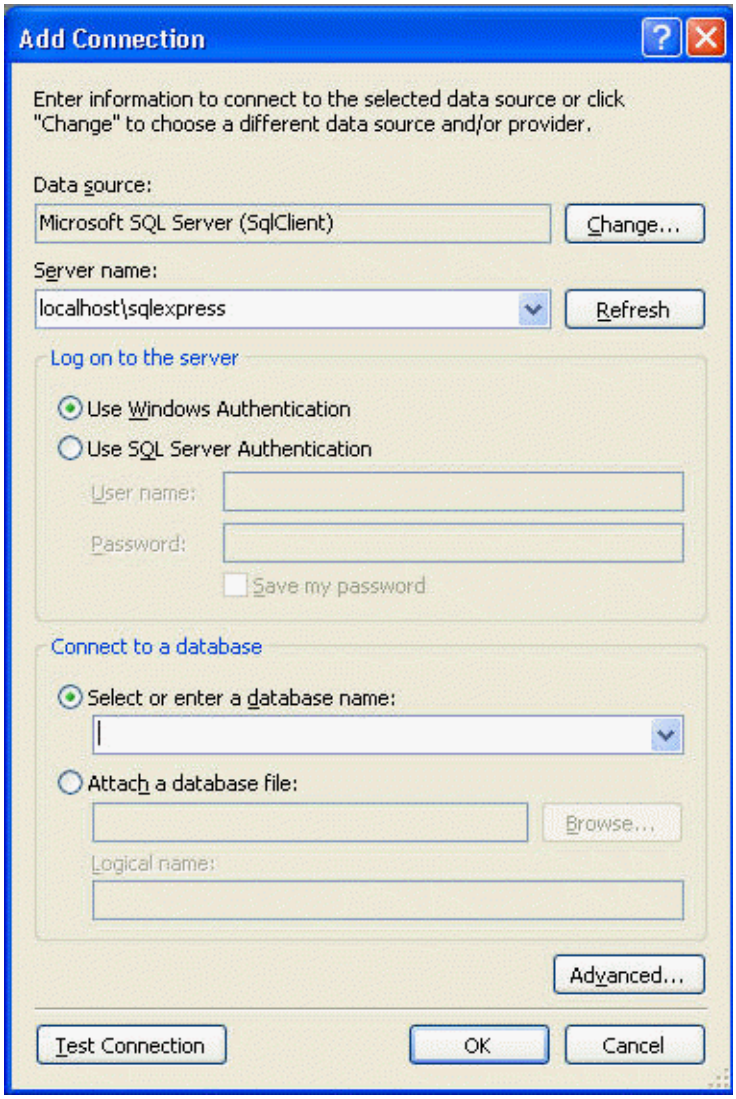
Le pulsamos con el botón derecho para añadir una conexión nueva



Le decimos que sea de SQL Server:



Y pulsamos en continuar:

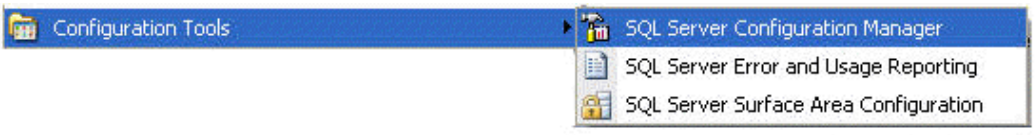


En el nombre del servidor debes poner el que tenga el Sql Server ejecutándose. Si es un servidor remoto pondríamos su nombre, si es un SQL Server Express de pruebas en nuestro equipo le pondremos este nombre especial:

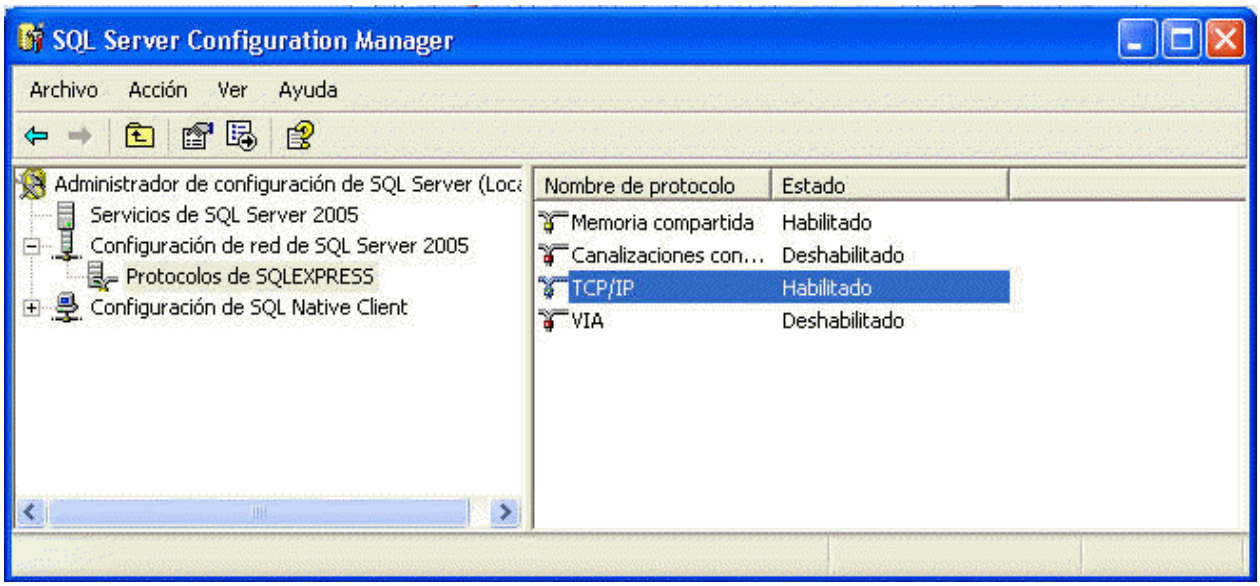
localhost\sqlexpress

Que significa "localhost" es el propio equipo, es decir apunta a si mismo y el nombre de la instancia de SQL Server que se crea en la instalación. Luego debemos pulsar en "Test Connection" para ver si todo es correcto.

En ocasiones puede que te de errores porque por defecto deja deshabilitada la comunicación de SQL Server por TCP/IP que es el protocolo de red, así que debemos irnos al menú de Windows: Inicio, Programas y luego en SQL Server 2005:

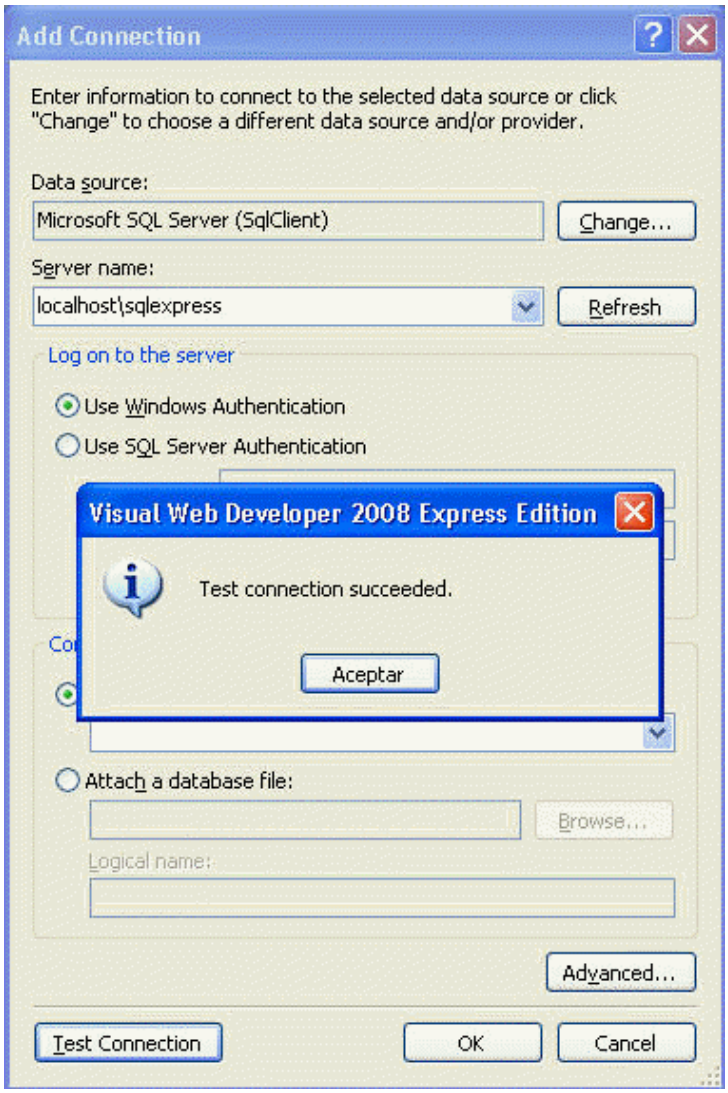


Para llegar a este programa:

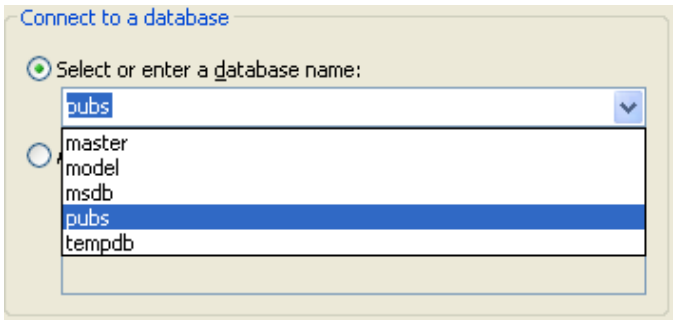


Como ves en la pantalla selecciona la parte de la izquierda de la configuración y "Protocolos de SQLExpress". A la derecha busca "TCP/IP", le das doble clic y lo habilitas. Después de esto tendrás que irte al panel de control de Windows y en las herramientas administrativas abrir los servicios para reiniciar el de SQL Server.

Con este cambio haremos la prueba de conexión otra vez y:



Una vez que estamos conectados con el servidor podremos ahora conectarnos a la base de datos con la que queremos trabajar. Pulsa en el desplegable:



Y seleccionas la base de datos "pubs". Si no te aparece es porque no seleccionaste todas las opciones al instalar SQL Server ya que una de las opciones es instalar las bases de datos de prueba.

Nota Si aun así no te aparece vamos a instalarla "a mano". Descargarte el fichero que tienes en esta enlace en tu disco duro: fichero_de_pruebas. Y ahora vamos a ejecutar el "script" o secuencia de comandos que tiene para instalar los datos de prueba. Abrimos una sesión de comandos en nuestro Windows, nos vamos donde hemos almacenado el fichero y ejecutamos:

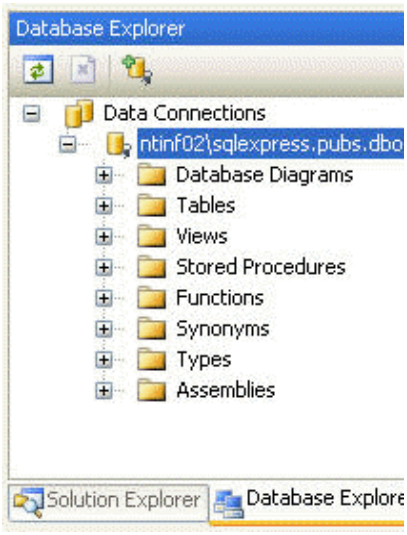
```
sqlcmd -S localhost\SQLEXPRESS -i InstPubs.sql
```

Que nos mostrará unos textos con la instalación de la base de datos de ejemplo.

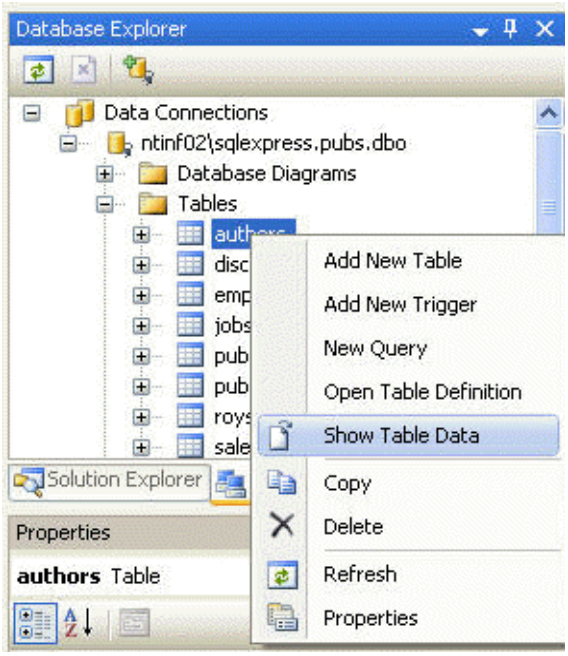
```
D:\Code\Beginning ASP.NET 2.0>sqlcmd -i InstPubs.sql
Changed database context to 'master'.
Beginning InstPubs.SQL at 10 Jul 2007 20:55:44:637 ....
Creating pubs database....
Changed database context to 'pubs'.
Now at the create table section ....
Now at the create trigger section ...
Now at the inserts to authors ....
Now at the inserts to publishers ....
Now at the inserts to pub_info ....
Now at the inserts to titles ....
Now at the inserts to titleauthor ....
Now at the inserts to stores ....
Now at the inserts to sales ....
Now at the inserts to roysched ....
Now at the inserts to discounts ....
Now at the inserts to jobs ....
Now at the inserts to employee ....
Now at the create index section ....
Now at the create view section ....
Now at the create procedure section ....
Changed database context to 'master'.
Ending InstPubs.SQL at 10 Jul 2007 20:55:49:463 ....

D:\Code\Beginning ASP.NET 2.0>
```

Y ya tendremos lista nuestra base de datos de pruebas en nuestro servidor SQL Server. Cancela ahora todos los pasos para hacer este enlace y repite todos los pasos, verás como ahora si te aparece la base de datos "pubs"

Así que ya tenemos nuestra conexión lista con ASP.NET:



Estos son los objetos que nos proporciona SQL Server. Dentro de ellos tenemos las tablas disponibles, así le damos para ver cuales tiene, seleccionamos la de "authors" y elegimos la opción de mostrar los datos de la tabla:



Que efectivamente nos mostrará los datos contenidos en esa tabla:

authors: Query...qlexpress.pubs)

au_id	au_lname	au_fname	phone	address	city
172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411	Oakland
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley
267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland Av...	San Jose
274-80-9391	Straight	Dean	415 834-2919	5420 College Av.	Oakland
341-22-1782	Smith	Meander	913 843-0462	10 Mississippi Dr.	Lawrence
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bateman St.	Berkeley
427-17-2319	Dull	Ann	415 836-7128	3410 Blonde St.	Palo Alto
472-27-2349	Gringlesby	Burt	707 938-6445	PO Box 792	Covelo

Vamos bien, hemos conectado a través de la red con un servidor de base de datos (aunque sea mi propio equipo, el procedimiento es igual para otro servidor), hemos seleccionado una tabla de una base de datos que le hemos indicado en la conexión y hemos abierto su contenido para explorarlo.

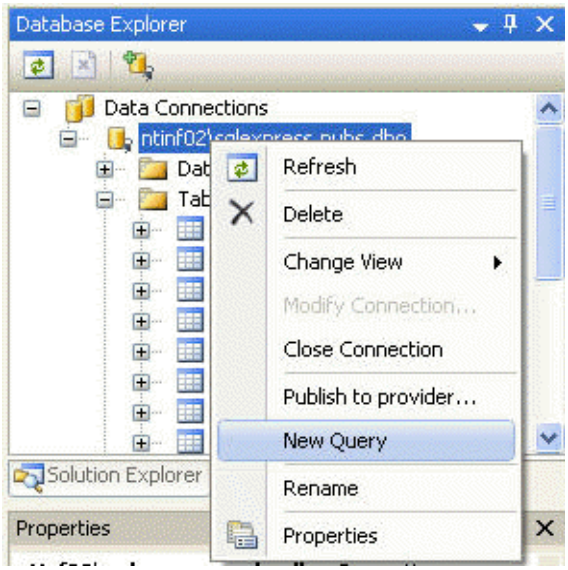
Un paseo por el explorador de bases de datos

4. SQL y las tablas en las bases de datos

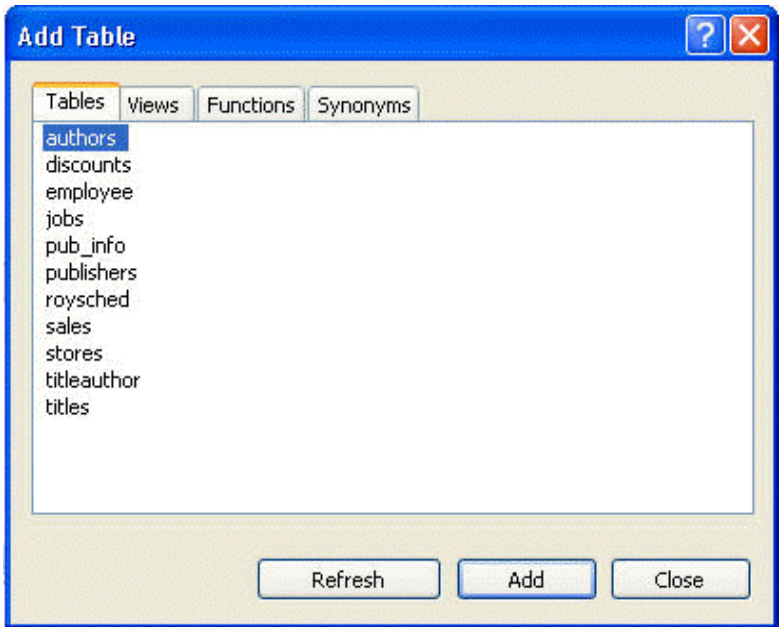
El lenguaje SQL es el lenguaje para realizar el mantenimiento de la base de datos. Este mantenimiento consta de las operaciones ya conocidas de: altas, bajas, consultas y modificaciones.

Nota En el anterior curso de ASP dedicábamos un tema entero al lenguaje SQL. Este lenguaje es el utilizado universalmente para acceder a las bases de datos: consultar, modificar, borrar datos... Puesto que el sería una pena perder un capítulo entero con este lenguaje dada la cantidad de cosas que hay que ver he optado por ponerte para descargar este tutorial de SQL. Pulsa aquí para descárgatelo y tenlo cerca cuando operemos con bases de datos...

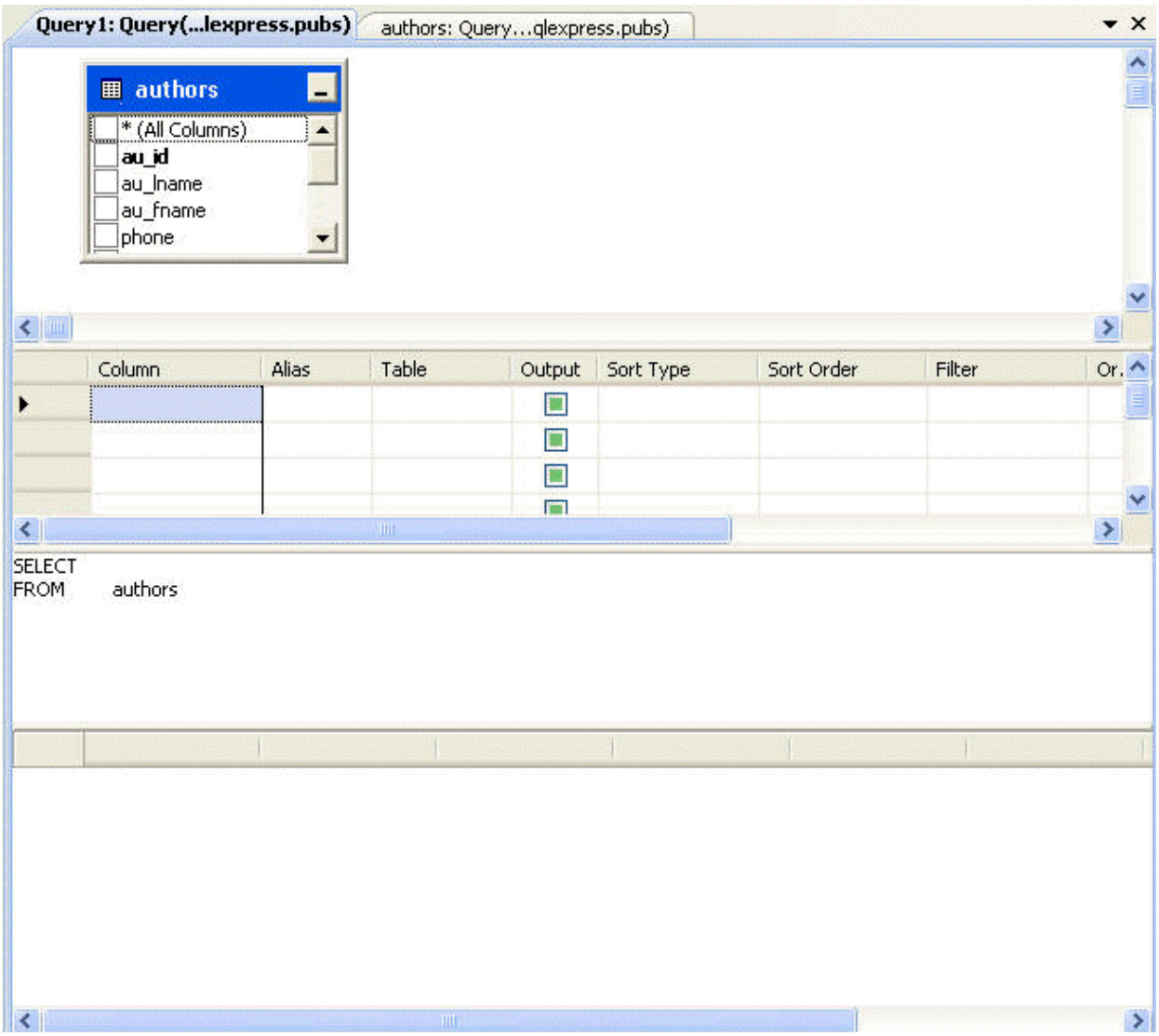
Veamos ahora algún ejemplo utilizando nuestro entorno del Visual Web Developer 2.008. Haz clic con el botón derecho en la conexión para seleccionar la opción de realizar una nueva consulta:



Que nos mostrará una pantalla nueva con varias opciones detrás y en primer plano:

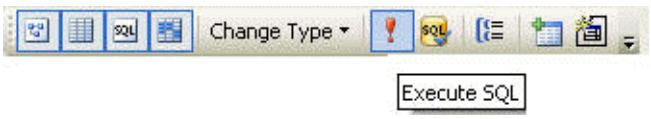


La lista de tablas disponibles en la base de datos. Seleccionamos la primera "authors" y le damos a añadir "add" y luego cerramos esa ventana "Close". Tendremos entonces este aspecto:

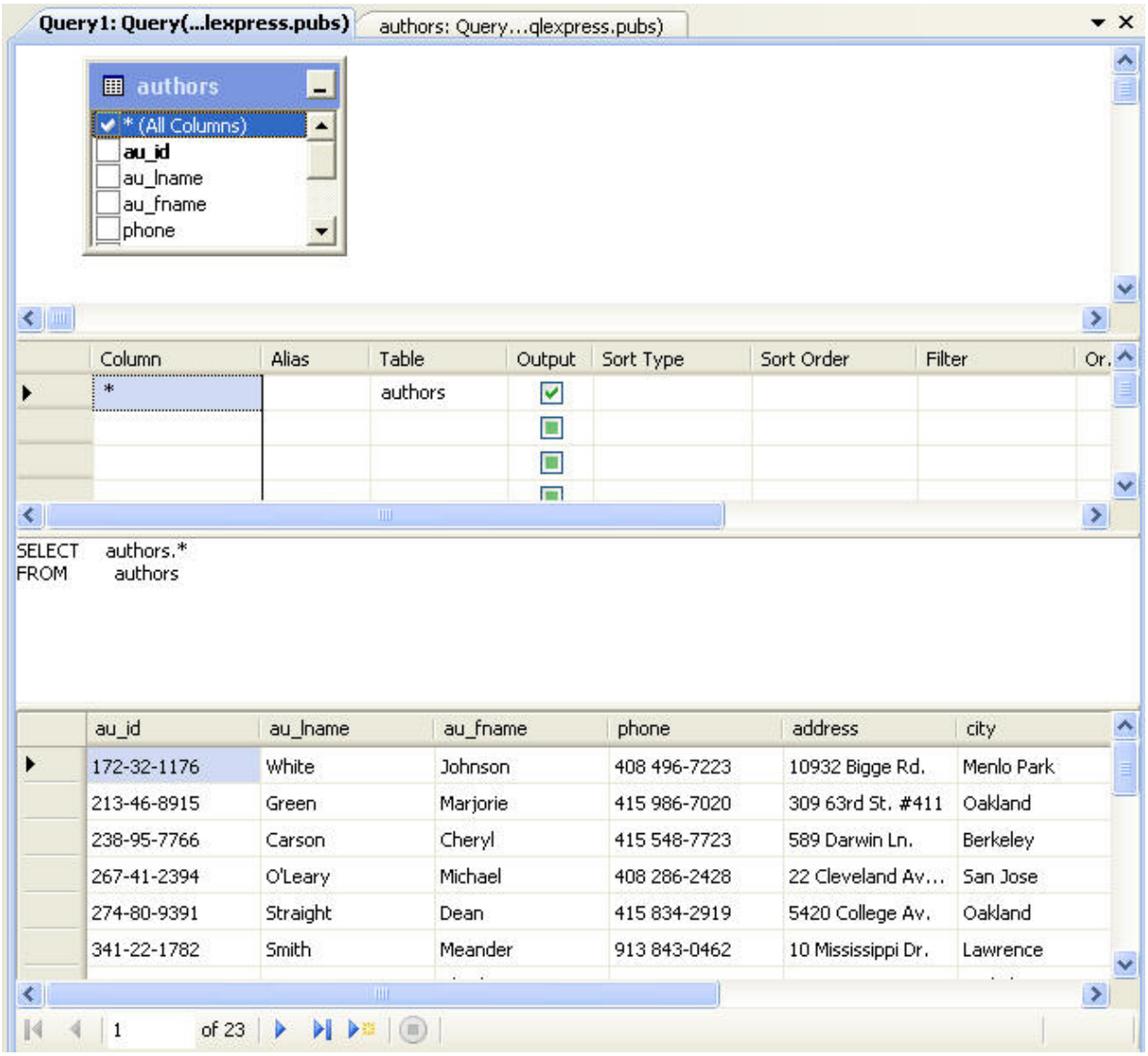


Selecciona todas las columnas del cuadro que ha puesto representando a la tabla "authors":

Ahora pulsa en la admiración de la barra del diseñador de consultas "query designer", si no la tienes visible, selecciónala con el botón derecho del ratón encima de las barras de botones:



Al ejecutarla la pantalla anterior cambiará a:



En la primera sección tenemos las tablas que van a componer la consulta en este generador. El siguiente cuadro es muy versátil y nos permitirá seleccionar columnas y añadir criterios. El tercer panel nos mostrará la sentencia SQL que va a ejecutar y por fin en el panel de abajo tenemos el resultado de ejecutar esa sentencia SQL.

Como te he comentado antes no podemos ponernos como objetivo aprender lenguaje SQL así que puedes repasar el tutorial que os he escrito con un resumen de lo mas importante sobre SQL. Es imprescindible para continuar con estos temas ya que si SQL es lenguaje utilizado para las bases de datos y estamos trabajando con bases de datos... pues como que sobran las palabras!.

Tenemos cuatro operaciones básicas en las bases de datos, que te recuerdo ahora con su sentencia equivalente debajo:

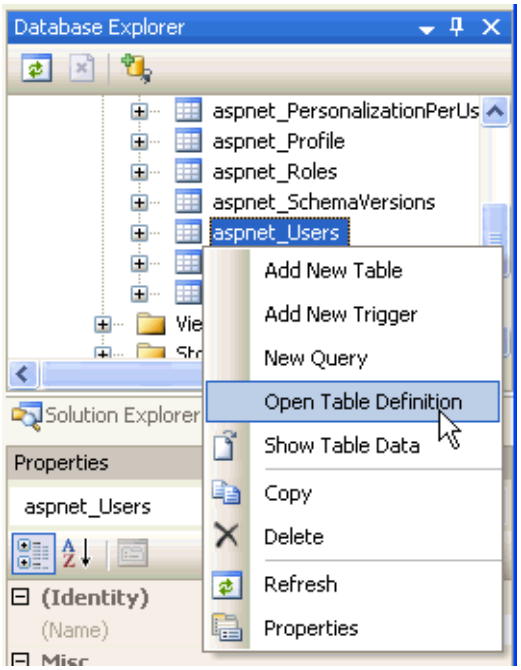
- Altas. "insert into clientes (nombre,dni) values ('JoseRodriguez',123456789)
- Bajas. "delete from clientes where dni=123456789"
- Consultas. "select * from clientes"
- Modificaciones. "update clientes set poblacion='Logroño' where dni=1234567890"

Podemos utilizar cualquiera de estas instrucciones en nuestro generador de consultas escribiéndolas en el panel y luego ejecutándola desde el botón que utilizamos antes.

Realizar consultas en el IDE

4.1 Tablas en SQL Server

Cada columna en cada tabla tiene un tipo de datos, para ver estos tipos abriremos la definición de la tabla así:



Esta tabla no la tienes en tu ejemplo que hemos cargado antes, pero tiene alguna cosa interesante que vamos a ver. Puedes editar una de las que hemos cargado para comparar y comprobar las pantallas de esta edición de tablas. Al abrirla nos mostrará las columnas y con sus tipos de datos:

Column Name	Data Type	Allow Nulls
ApplicationId	uniqueidentifier	☐
UserId	uniqueidentifier	☐
UserName	nvarchar(256)	☐
LoweredUserName	nvarchar(256)	☐
MobileAlias	nvarchar(16)	☑
IsAnonymous	bit	☐
LastActivityDate	datetime	☐
		☐

Los dos primeros campos tienen un tipo especial de datos que es un identificador único, es como si internamente le asignara un valor único para esos campos. Así debe ser ya que fíjate que uno es el identificador de usuario: "UserId" y obviamente no puede haber dos usuarios con el mismo identificador.

También vemos que la clave única para esta tabla es el campo "UserId" que nos lo marca con esa llave a la izquierda. Cuando nosotros creemos tablas obviamente debemos pensar como cuando vimos los capítulos de VB.NET en el tipo de datos que debemos utilizar, estos son los mas habituales:

- Text
- Number
- Date/Time
- Boolean
- Binary
- Otros como es identificador único: uniqueidentifier, xml, timestamp, sql_variant.

Dentro de los texto nos encontramos con varios tipos que tienen sus peculiaridades:

- char (n). Texto de longitud "n" fija
- nchar(n). Texto de longitud "n" fija Unicode
- varchar(n). Texto de longitud variable con un valor máximo de "n" no unicode y de hasta 8.000 caracteres
- nvarchar(n). Texto de longitud variable con un valor máximo de "n" unicode y de hasta 4.000 caracteres

Unicode es el tipo de codificación habitual y es la que debemos utilizar es un formato mucho mas amplio que el antiguo código ASCII.

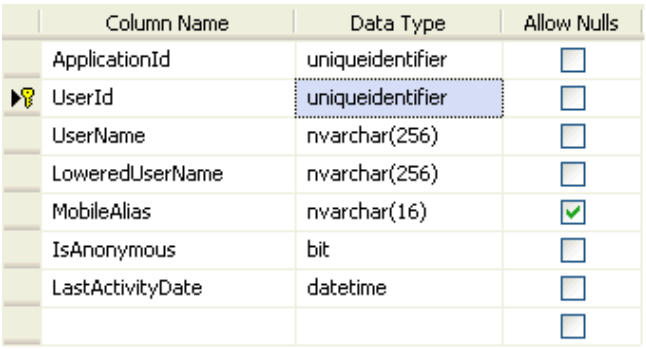
En los numéricos tenemos dos tipos:

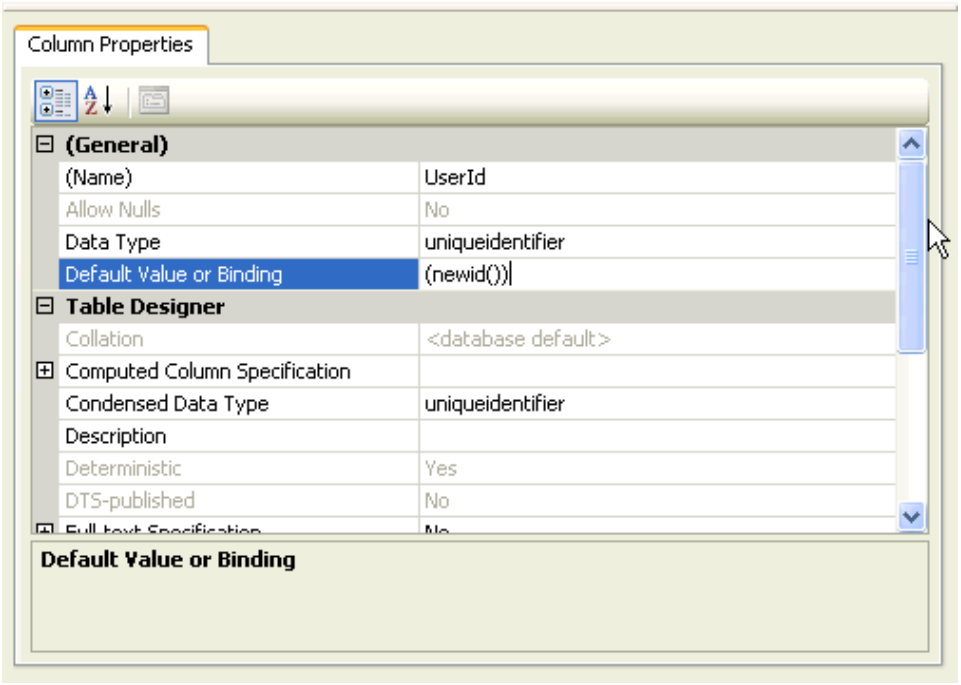
- Números enteros:
  - tiniyint: admite valores de 0 a 255
  - smallint: desde -32.768 hasta 32.767
  - int: desde -2.147.483.648 hasta 2.147.483.647
  - bigint, para valores mucho mas grandes.
- Números en coma flotante, es decir, con decimales los vemos en la siguiente tabla

Otros tipos de datos son:

- Boolean (bit) admite un valor de 0 ó de 1
- smalldatetime: para las fechas/horas
- uniqueidentifier: define un campo que almacena un "Globally Unique Identifier", es una cadena de caracteres que hace de identificador único. Es un valor que no tiene significado pero que es único así que se puede utilizar para relacionar las tablas o como índice de la tabla.

Un comentario para estos valores únicos, es muy importante decirle en este campo que:





Como ves marcado, el valor predeterminado es una función especial que genera este valor aleatorio, si no ponemos esto nos dejará siempre el mismo valor, así que ojo. Veamos ahora un resumen de los tipos de datos en SQL Server:

Tipo de datos	Detalles	Espacio
bigint	enteros largos -2^63 hasta 2^63	8 bytes
binary	binario fijo hasta 8000	n bytes
bit	0, 1 ó null	1 byte para 8 campos de bit
char (n)	texto fijo hasta 8.000	n bytes
datetime	desde 1/1/1753 hasta 31/12/9999	8 bytes
decimal (p,s)	números -10^38 +1 hasta 10^38 -1	según p
float (n)	números desde -1,79E+38 hasta -1,79E-38	según n
image	variable hasta 2^31	longitud + 2 bytes
int	enteros desde -2.147.483.648 hasta 2.147.483.648	4 bytes
money	cantidad para moneda	8 bytes

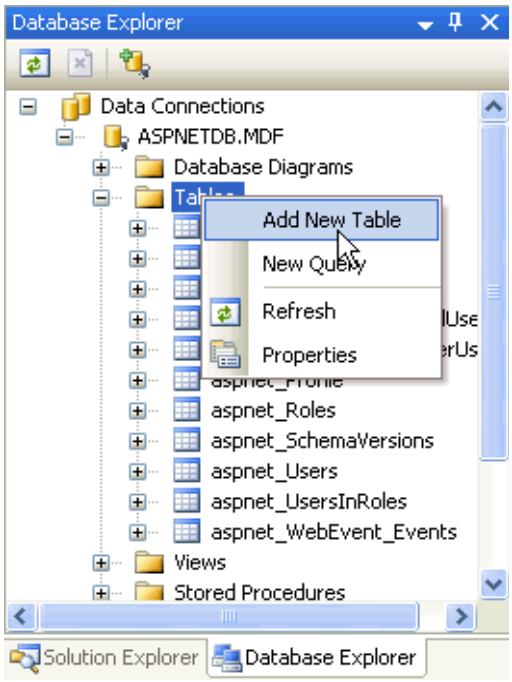
nchar(n)	cadena de longitud fija unicode de hasta 4.000 caracteres	2 * longitud
ntext	cadena de longitud variable de hasta 2^30	2 * longitud bytes
numeric (p,2)	números desde -10^38 hasta 10^38	Depende de p
nvarchar (n)	Cadena de longitud variable unicode de hasta 4.000 caracteres	2 * longitud + 2 bytes
nvarchar (MAX)	Cadena de longitud variable unicode de hasta 2^31 bytes	2 * longitud + 2 bytes
real	-1.18E-38 and 1,18E-38 hasta 3,40E+38	4 bytes
smalldatetime	fechas desde 1/1/1900 hasta 6/16/2079	4 bytes
smallint	enteros desde -32768 hasta 32767	2 bytes
smallmoney	-214748,3648 hasta 214748,3648	4 bytes
sql_variant	hasta 8016 bytes	depende del almacenamiento de datos
text	texto de longitud variable de hasta 2^31 caracteres	depende de la página de códigos del servidor
timestamp	hora del último cambio de la fila	8 bytes
tinyint	enteros de 0 a 255	1 byte
uniqueidentifier	Global Unique Indentifiers	16 bytes
varbinary(n)	datos binarios de longitud variable hasta 8000	longitud + 2 bytes
varbinary(MAX)	datos binarios de longitud variable hasta 2^31	longitud + 2 bytes
varchar(n)	texto de longitud variable hasta 8000 caracteres	n+2 bytes
varchar(MAX)	texto de longitud variable hasta 2^31 caracteres	longitud + 2 bytes
xml	tipo de datos XML	depende de los datos almacenados

4.2 Administrar tablas

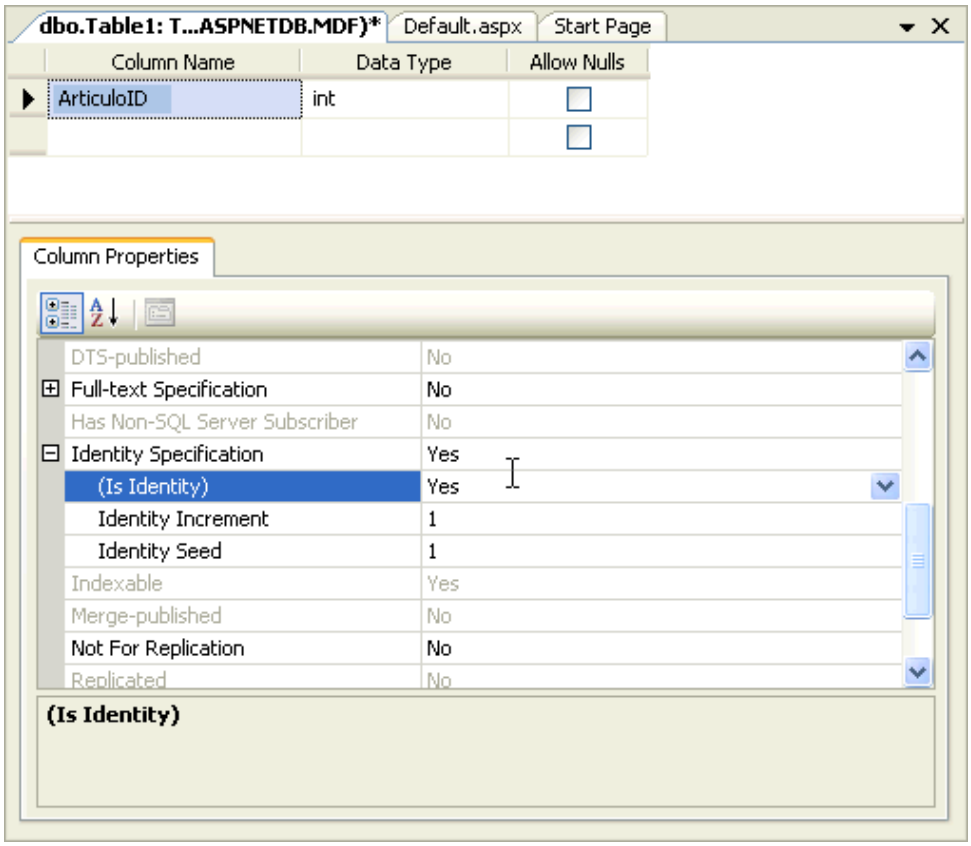
Bueno, es interesante siempre el acceso al servidor de base de datos desde nuestro entorno, ya hemos visto antes que podíamos ver entre otras cosas las tablas de nuestra base de datos de SQL Server. Ahora vamos a ver como crear tablas desde este entorno...

Crear tablas

Para crear tablas simplemente iremos a la vista de tablas y con el botón derecho indicaremos:



Luego definiremos los campos que queramos:

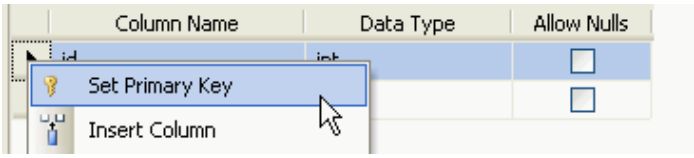


He creado un campo llamado "Articuloid" que será un identificador único para nuestra fila. ¿es obligatorio? Hombre pues si, todas las tablas deben tener una clave única. Imagina por ejemplo que tenemos dos "fernandez lopez" en nuestra tabla de empleados y ejecutamos una instrucción para cambiar un dato en un "fernandez lopez" que tengo en pantalla, la base de datos no sabrá distinguir entre los dos si no existe un

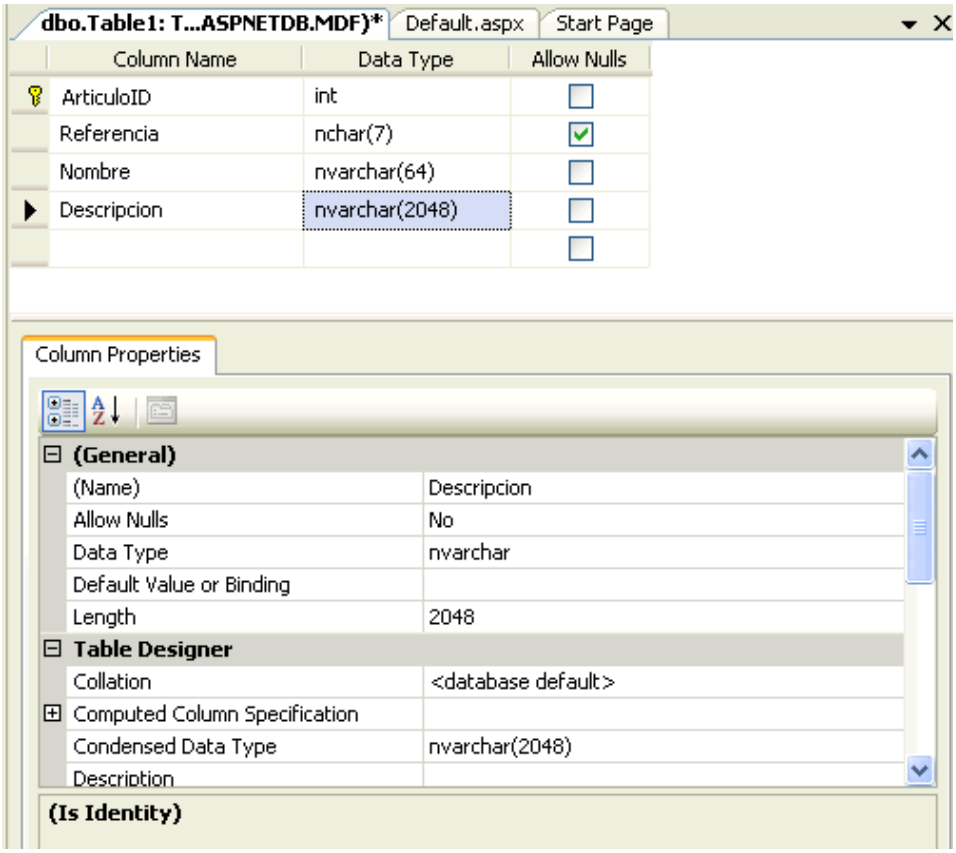
identificador único que los diferencie, como por ejemplo el DNI: actualiza los datos del empleado con DNI número XXX.

Así que sabemos que hay que crear un valor que haga de clave primaria. Hemos visto antes que el "uniqueidentifier" nos crea, añadiendo una función, un valor único de tipo un poco engorroso de utilizar, así que utilizaremos otro mas facilito que es el "identity". Este valor es un numérico que se autoincrementa cada vez que insertamos una fila, con lo que nunca tendrá valores repetidos y será, por tanto, único. Mira en la pantalla anterior, he creado un campo numérico de tipo "int" y debajo he marcado que es de tipo "identity" que va a empezar en el valor 1 y que se incremento será de 1 en 1.

Además con el botón derecho le indicamos que oficialmente sea la clave primaria de esta tabla:



Ahora añadiremos campos de texto:



El primero es una referencia interna de un producto que siempre va a tener 7 caracteres, de ahí que sea "nchar", además puede contener valores nulos, así que se lo indico. Luego dos campos mas con el nombre y la descripción del producto.

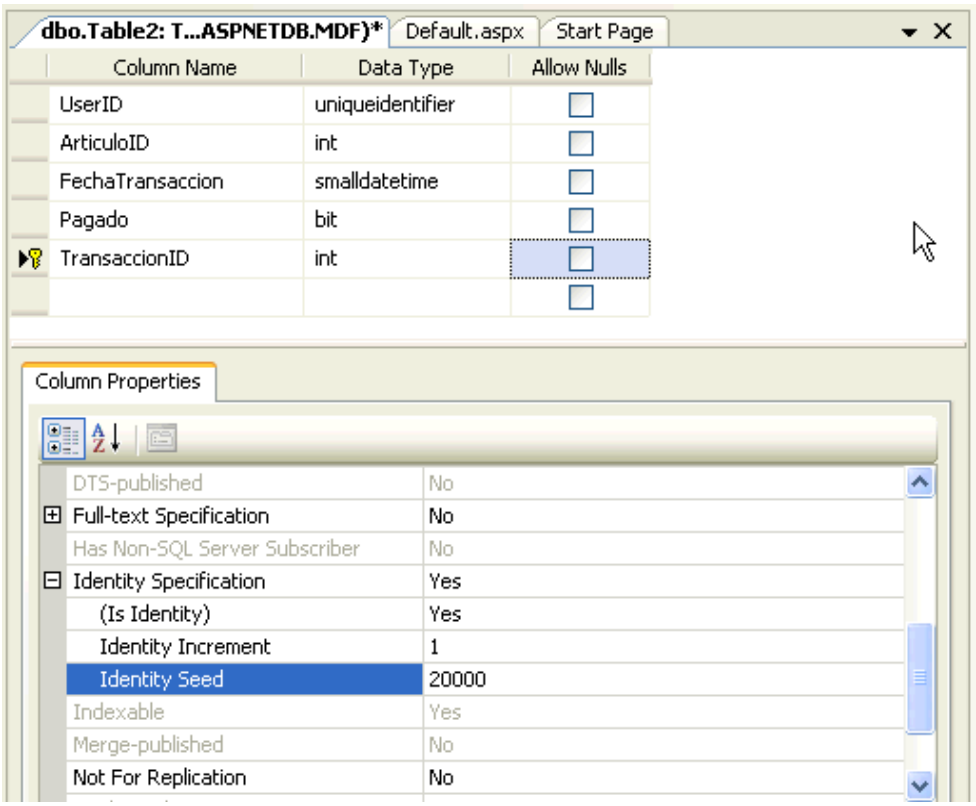
Ahora vamos a añadir un campo de tipo moneda, no es muy habitual pero bueno, como estamos repasando todo lo ponemos:

Column Name	Data Type	Allow Nulls
ArticuloID	int	☐
Referencia	nchar(7)	☑
Nombre	nvarchar(64)	☐
Descripcion	nvarchar(2048)	☐
Precio	smallmoney	☐

Ahora guarda la tabla, por ejemplo cerrando con la "X" la ventana para que nos pregunte si queremos guardarla o pulsando en el habitual icono del disco.

Crear una tabla de transacciones

Una tabla de transacciones es una tabla donde iremos apuntando las operaciones que van haciendo los usuarios, cuando la tengamos en marcha verás que es realmente necesario, ahora simplemente la crearemos de una forma prácticamente igual que antes. Aquí apuntaremos las operaciones de los usuarios que tienen cada uno su identificador "userid" así que ese es el primer campo, que en lugar de ser el nombre y apellidos es simplemente el identificador único de ese usuario, si cruzamos luego estas dos tablas podremos acceder entonces a sus datos personales, de momento crea estos campos...



Fíjate en el segundo campo, es el identificador del artículo. Es decir ponemos el número único del artículo generado de forma automática en la otra tabla con el campo "identity". Bien, tenemos un identificador que nos

apunta al registro de la otra tabla. Ahora debemos poner el identificador del usuario que está haciendo la transacción. En este caso no podemos utilizar el tipo de datos entero porque en la tabla de usuarios nuestro entorno al crear la tabla utilizó ese otro tipo de datos utilizado como identificador único: "uniqueidentifiar". Recuerda que es otro tipo de identificador, en este caso es un valor alfanumérico aleatorio mas complicado de manejar pero igual de efectivo.

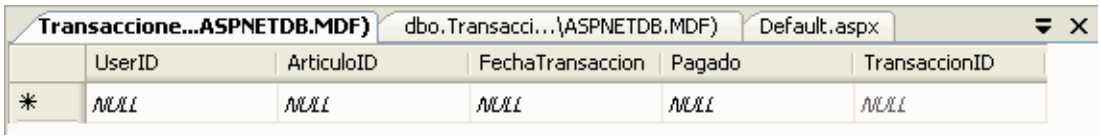
Bueno, como en esta tabla debemos poner la referencia a este usuario debemos poner entonces una clave con el mismo tipo de datos de ahí que haya puesto el primer campo "UserID" de tipo de datos "uniqueidentifiar".

Luego le ponemos una fecha de transacción para saber cuando se ha realizado un valor de tipo "boleano" (cierto/falso) indicando si se ha pagado ya la operación y como siempre en nuestras tablas un campo que nos identifique a esta transacción así que por último añadimos ese "TransaccionID" de tipo "Identity" y como novedad cambiaremos el valor inicial ya que quiero identificar las transacciones con números a partir del 20000. Y ponle como en el caso anterior que sea una clave única.

Apenas hay cosas nuevas, ya conoces muchas de estas cosas de la programación de Visual Basic.net

Rellenar datos

Ahora debemos completar ya unos datos para seguir con nuestras pruebas. Cuando hayas creado la tabla haz clic con el botón derecho para mostrar la tabla, vacía por supuesto:



Podemos añadir datos en la celdas que todavía no tienen que están con "null" pero sería una tarea imposible, debemos ir a la tabla de artículos para ver identificador y ponerlo, luego a la tabla de usuarios y ponerlo... muy complicado. Para empezar vamos a poner datos en la tabla de artículos que creamos antes y que ahora está vacía, por ejemplo:

ArticuloID	Referencia	Nombre	Descripcion	Precio
2	ARC-001	Introducción a V...	Libro que descri...	10,0000
NULL	ARC-002	Los Fox-Terrier	Descripción de la	34
NULL	NULL	NULL	NULL	NULL

Como la primera columna es autonumérica no debemos meter valores ahí, así que completa los demás datos. Verás que aparecen signos de admiración hasta que esté completa la fila.

ArticuloID	Referencia	Nombre	Descripcion	Precio
2	ARC-001	Introducción a V...	Libro que descri...	10,0000
3	ARC-002	Los Fox-Terrier	Descripción de la...	34,0000
4	IMG-001	Fotografia	Manual de fotog...	120,0000
6	IMG-002	Fotografia II	Mas manual	130,0000
7	LAB-001	ASP.NET	Manual de ASP....	100,0000
NULL	NULL	NULL	NULL	NULL

Luego lo haremos bastante mejor, de momento todo un poco manual pero así practicas... Bueno, ahora vamos a simular unas transacciones que no son mas que compras de libros por parte de nuestros usuarios.

Para identificar a los usuarios abriremos la tabla de usuarios y cogeremos sus identificadores así podemos completar ya unas transacciones:

	UserId	ArticuloID	FechaTransaccion	Pagado	TransaccionID
	ff45a218-f080-4e08-a3b0-5802a60df513	1	01/04/2006 0:0...	True	20000
	ff45a218-f080-4e08-a3b0-5802a60df513	2	01/01/2006 0:0...	True	20003
	ff45a218-f080-4e08-a3b0-5802a60df513	3	01/01/2006 0:0...	True	20006
	57e2cb02-0832-46ff-8bd2-899827148f4d	3	01/01/2006 0:0...	True	20007
	57e2cb02-0832-46ff-8bd2-899827148f4d	4	01/01/2006 0:0...	True	20008
▶*	NULL	NULL	NULL	NULL	NULL

Si lo lees un poco lo que he hecho ha sido ponerle a mi usuario Jose que ha comprado los libros de referencia 1, 2 y 3. Y a otro usuario (que he copiado y pegado su identificador único) le he asignado los libros 3, 4 y 5. Como ves el autonumérico de tipo "identity" ha comenzado a numerar por el valor 20000.

Enlazar tablas

Ya tenemos tres tablas ahora hay que enlazarlas pero esto ya te imaginas que es una tarea sencilla ya que la tabla de transacciones enlaza perfectamente las otra dos: por el índice del artículo y por el índice del usuario.

En este punto debemos saber ya un poco del lenguaje SQL, recuerda que ya te puse un pequeño manual de emergencia de SQL y que debes repasar sino en este curso no terminaríamos nunca, así que como es un lenguaje sencillo te dejo que lo repases tu mismo. Te lo pongo aquí otra vez para que lo tengas a mano, haz clic aquí para descargarlo.

Para consultar toda la tabla de artículos la consulta será:

```
select * from articulos
```

Que nos recupera todas las columnas o campos de la tabla. Si quiero solo alguna columna lo indicaré en el lugar del asterisco:

```
select UsuarioID, UserName from aspnet_Users
```

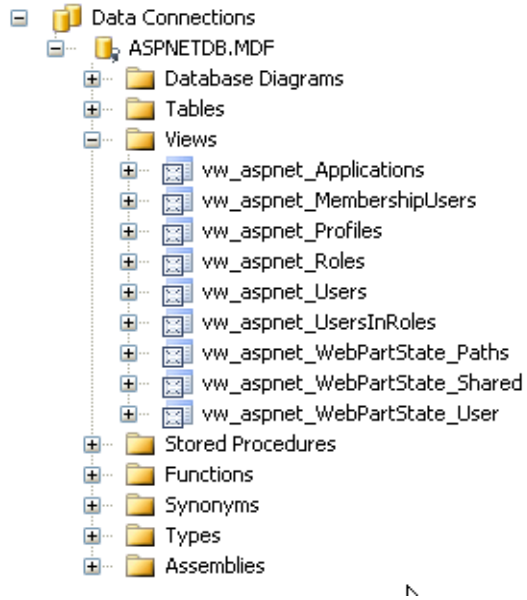
Que me devolverá solo las columnas UsuarioID y UserName. Pero devuelve todas las filas así que tenemos que ver como ponerle alguna restricción con "where", así que:

```
select * from articulos where articuloid=2
```

Nos devuelve todas las filas cuyo identificador sea el valor "2". Como es un identificador único obviamente solo nos devolverá una fila. Bueno y así sucesivamente con este leguaje, así que pégale un repasito al manual que te he adjuntado antes que te vendrá muy bien y se vas a trabajar con bases de datos debes manejarlo. Aunque nosotros ahora trabajaremos con un generador de SQL donde gráficamente crearemos esas consultas para ver luego su equivalente en SQL.

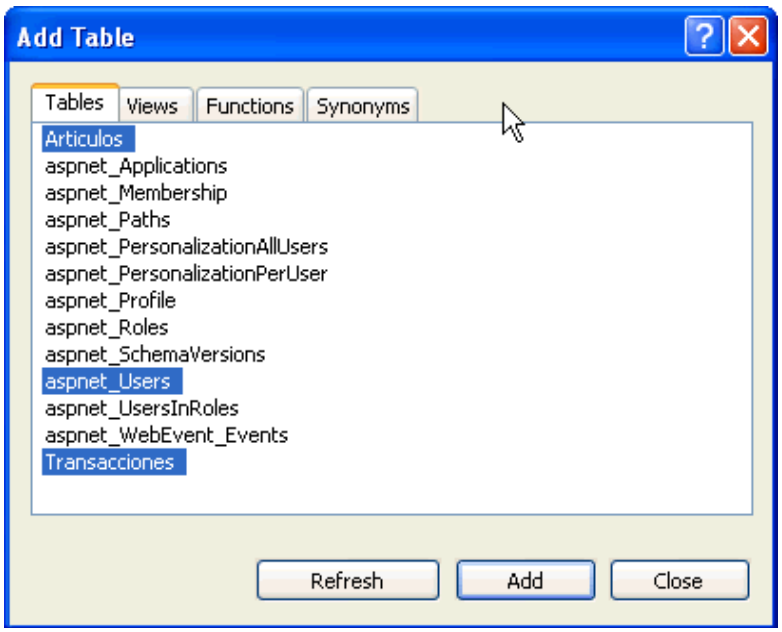
Crear vistas

Una vista es una consulta almacenada. Es decir, antes vimos como hacer una consulta con el lenguaje SQL, pues bien, el resultado de esa consulta es una vista. Fíjate en el explorador de soluciones, en la parte de acceso a datos:

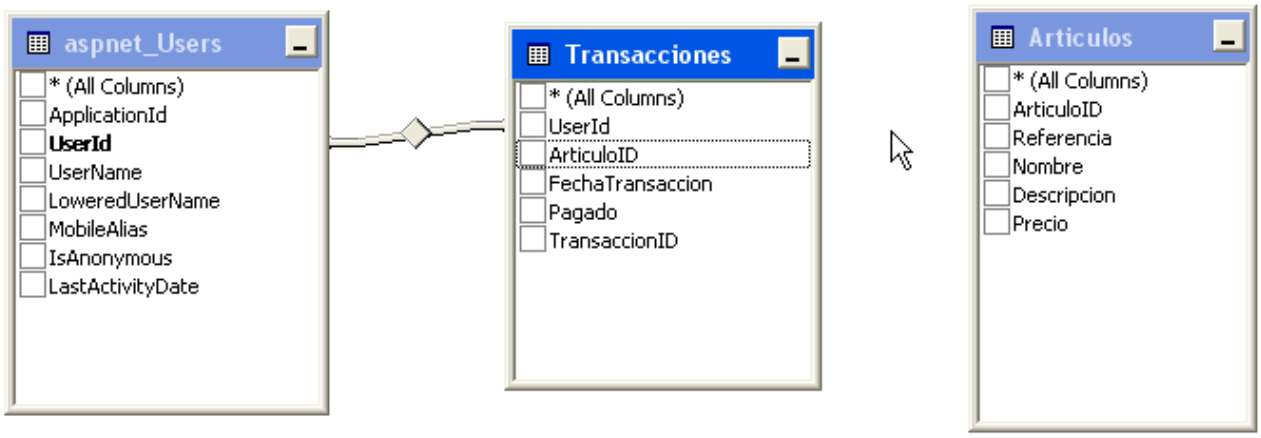


Estas vistas nos las ha creado nuestro sistema así que ten cuidado en no borrarlas pero si puedes abrirlas para ver los resultados que serán similares a los de abrir las tablas ya que en muchos casos las han creado con los mismos campos que tienen las tablas. Bien, ahora utilizaremos nuestro generador de consultas para enlazar las tres tablas: usuarios, artículos y transacciones.

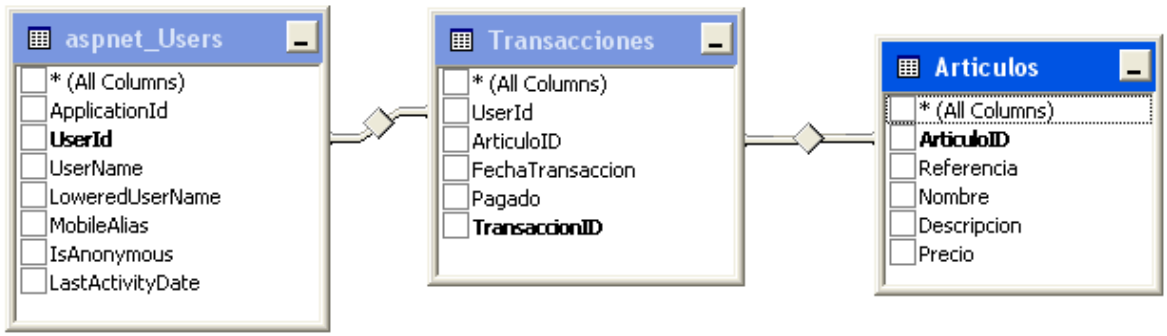
Así que dale con el botón derecho en las vistas para añadir una nueva:



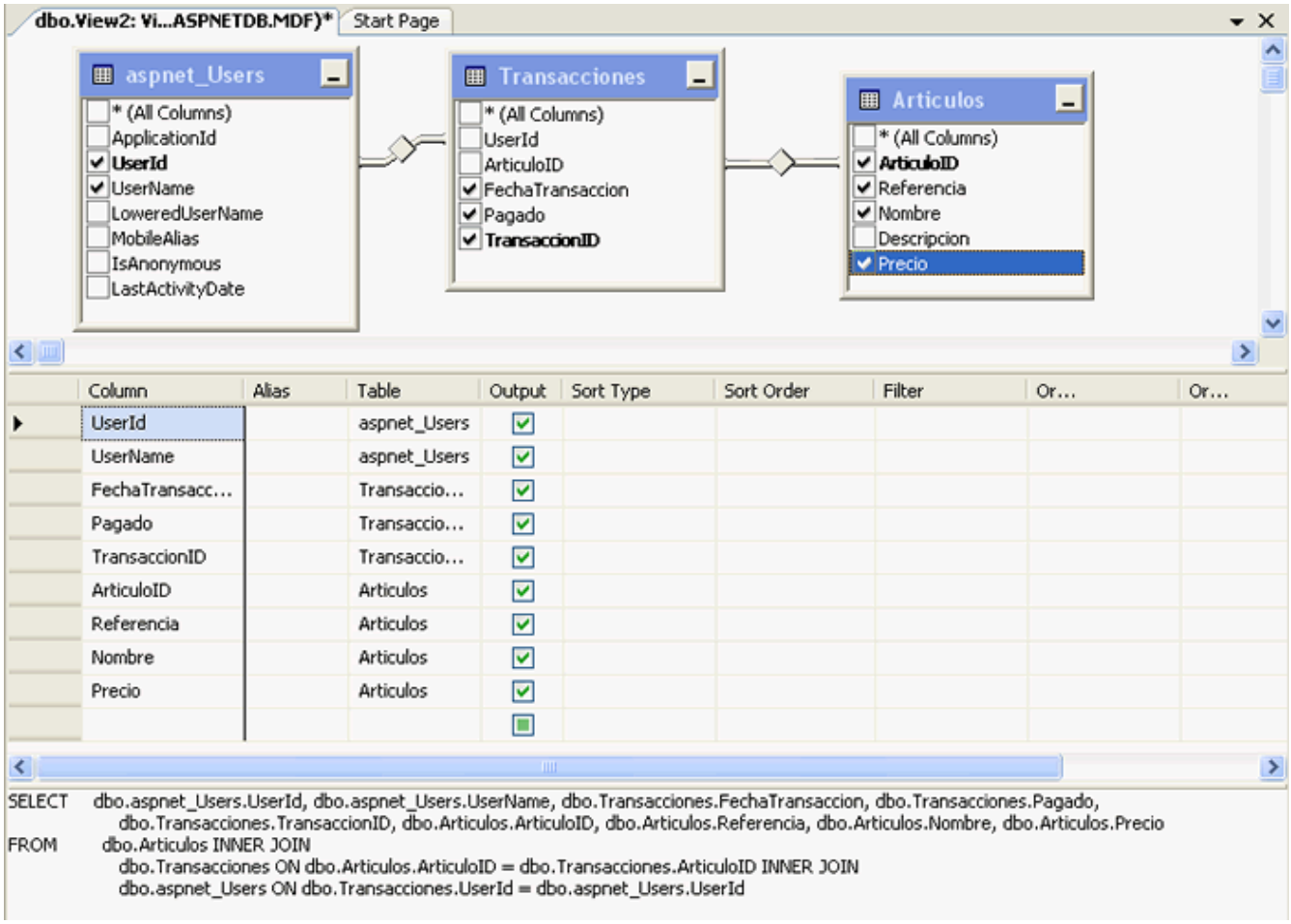
El primer paso es indicar que tablas van a componer esta vista así que marcamos las tres implicadas y le decimos "add":



En mi caso me ha añadido la relación existente entre las tablas "Users" y "Transacciones" mediante el campo "UserID" pero falta enlazar la tabla "Transacciones" con "Articulos" así que vamos a hacer este enlace. Simplemente vamos a arrastrar el campo "ArticuloID" de la tabla "Transacciones" y lo vamos a soltar encima del campo "ArticuloID" de la tabla artículos que debe producir esto:

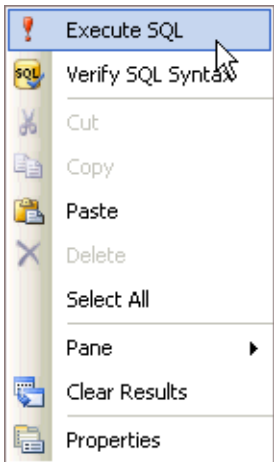


Ya sabe el generador de consultas que el "userid" debe extraerlo de la tabla de usuarios y lo mismo con la de artículos para el campo "articuloid". Ahora vamos a marcar los campos que queremos que nos muestre en la consulta:



Fíjate que en el panel central nos va mostrando los campos y en la parte inferior la consulta SQL resultante. Tranquilo no es tan complicada pero es que estos generadores al poner los nombres completos suelen hacer que sean muy largas pero son fáciles de leer, lo importante está al hacer la combinación de las tablas con "Inner Join" que es donde establece el vínculo con los índices en las dos tablas.

PAra ejecutar esta consulta haz clic por ejemplo con el botón derecho encima del panel donde está la sentencia SQL:



Y al cabo de unos segundos en la parte de abajo podrás ver el resultado:

	UserId	UserName	FechaTransaccion	Pagado	TransaccionID	ArticuloID	Referencia	Nombre	Precio
▶	3b0-5802a60df513	jose	01/04/2006 0:0...	True	20000	1	ARC-001	Introducción a Vi...	10,0000
	ff45a218-f080-4...	jose	01/01/2006 0:0...	True	20003	2	ARC-002	Los For-Terrier	34,0000
	ff45a218-f080-4...	jose	01/01/2006 0:0...	True	20006	3	IMG-001	Fotografia	120,0000
	1299e1eb-968e-...	antonio	01/01/2006 0:0...	True	20007	3	IMG-001	Fotografia	120,0000
	1299e1eb-968e-...	antonio	01/01/2006 0:0...	True	20008	4	IMG-002	Fotografia II	140,0000

Cada fila o registro representa una transacción, ya ves que tenemos varias ventajas ya que al combinarlas hemos extraído los campos asociados a los índices con los ID del artículo y del usuario, es mas, estos dos valores los podríamos omitir porque no le interesan al usuario. Las vistas proporcionan una forma rápida y sencilla de mostrar datos en pantalla... de solo lectura. Eso es, los datos que mostraremos en las páginas Web a los usuarios serán de solo lectura. Vale, guardamos esta vista y seguimos avanzado con una vista mas detallada.

Vista detallada

Bueno, antes ampliamos los datos de los usuarios como su mail pero aquí no se reflejan así que debemos ampliar esta vista porque queremos mostrar mas detalles de estos usuarios. Los datos adicionales se almacenaban en la tabla "Membership" que tiene este aspecto:

	ApplicationId	UserId	Password	PasswordFormat	PasswordSalt	MobilePIN	Email	Lo
▶	bd2-899827148f4d	a18fc8bf-ec1e-4...	10UFdc0UKsFFV...	1	vAb1yR2kDKs1o...	NULL	jose@jose	jos
	57e2cb02-0832-...	1f0b3846-8e72-...	dsIFvlgOFnMLoF...	1	nXB4u4TYLWzp2...	NULL	Josemari@josem...	jos
	9ee085fd-a850-...	1299e1eb-968e-...	rpF7IHffjYp3ri6z...	1	xLU1lb5XMlu/ux...	NULL	antonio@miempr...	an
	9ee085fd-a850-...	ff45a218-f080-4...	P0lGs94zwxAu9...	1	WtPy7QNeZvfp...	NULL	jose@miempres...	jos
✳	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NU

5. Acceso directo a datos

Y vistos unos ejemplos vamos a trabajar ya sobre nuestra base de datos "pubs" con páginas en ASP.NET. La forma mas sencilla para interactuar con los datos es realizando un acceso directo a ellos. Ahora veremos como realizar todo esto paso a paso conociendo todos los objetos. Es la parte importante ya que aprenderemos a hacer todo a mano. Luego ya vendrán los controles ASP.NET con toda su potencia.

Cuando utilizamos acceso directo haremos uso de los comandos SQL que hemos visto anteriormente. Cuando hagamos una consulta directa no mantendremos nada en memoria, la idea es abrir una conexión, realizar la consulta u operación y luego cerrar la conexión. Los pasos para realizar todo el proceso van a ser:

1. Creamos los objetos de conexión, ejecución de comandos y recuperación de datos: Connection, Command y DataReader
2. Utilizaremos DataReader para recuperar información de la base de datos y la mostraremos en un formulario web
3. Cerramos la conexión
4. Enviamos la página al usuario. En este punto se han destruido ya todos los objetos de ADO.NET

Para actualizar o modificar información haremos estos pasos:

1. Crear una nueva conexión y crear los comandos "command"
2. Ejecutar los comandos con las instrucciones SQL apropiadas.

5.1 Crear una conexión

El objeto Connection se utiliza para conectarse con un origen de datos. Para abrir una conexión el objeto Connection disponemos de un método llamado "Open ()" que abre una conexión especificada por la **cadena de conexión**. Por lo tanto esta cadena de conexión es la que contiene el tipo de conexión que se realiza y con quien y se compone de tres partes:

- Tipo de proveedor o controlador utilizado
- La base de datos a utilizar
- "Credenciales" de identificación, es decir el nombre de usuario y contraseña utilizados para la conexión.

Las cadenas mas utilizadas son la Access y la de SQL Server así que nos centraremos en ellas, en Access sólo decir que el motor de base de datos que utiliza se llama Jet, de ahí que nos refiramos al "Jet" al configurar una conexión a Access:

Origen de datos	Cadena de conexión OLE DB
Estándar OLE DB	Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\datos\mibasededatos.mdb
OLE DB para SQL Server	Provider=SQLOLEDB.1;server=miservidor;database:mibasededatos;uid=usuario;pwd=contraseña
SQL Server	server=miservidor;database:mibasededatos;uid=usuario;pwd=contraseña

Podemos acceder a un servidor de datos SQL Server utilizando una conexión OLEDB, esa es la segunda fila de arriba. Pero es mejor utilizar los objetos específicos para SQL Server que no los genéricos OLEDB para obtener mejores resultados. En la tercera fila como se va a utilizar un objeto específico de SQL Server no es necesario decirle la primera parte de "Provider=SQLOLDB.1".

La sintaxis para esta cadena es muy importante que esté correcta, hay que tener cuidado de:

- Separar los argumentos con ";"
- Algunos argumentos tienen espacios entre ellos ("Initial Catalog") y debe ser así con el espacio.
- No se utilizan las comillas " ya que toda la cadena se introduce como un string normal entre ellas: "...."

Veamos algunos ejemplos de cadena de conexión. Utilizando el controlador de OleDB para SQL Server:

```
Dim conexion as New OleDBConnection()

conexion.ConnectionString="Provider=SQLOLEDB.1;Data Source=localhost;

        Initial Catalog=Pubs;Integrated Security=SSPI"
```

Utilizando los objetos de SQL Server:

```
Dim conexion as New SqlConnection()

conexion.ConnectionString="Data Source=localhost;Initial Catalog=Pubs

        ;Integrated Security=SSPI"
```

Si estamos utilizando SQL Server Express, es decir, en nuestro curso:

```
Dim conexion as New SqlConnection()

conexion.ConnectionString="Data Source=localhost\SQLExpress;Initial Catalog=Pubs;Integrate
```

Esta cadena de conexión se compone de varias partes separadas por ";" y que son:

- Data Source. Indica el nombre del servidor. Si está en el mismo servidor que ejecuta ASP.NET podemos poner "localhost". Teniendo como excepción si utilizamos Sql Server Express que habría que poner "localhost\SqlExpress"
- Initial Catalog. La base de datos de conexión inicial. Luego se puede modificar para conectarse a otra.
- Integrated Security. Indica si nos conectamos con un usuario de Windows o de SQL Server. En este segundo caso le pondríamos el nombre de usuario y contraseña en la conexión.
- Connection Timeout. Opcional e indica el tiempo de espera hasta devolver un error al establecer la conexión. Si ponemos 0, esperará indefinidamente, pero no es buena idea así que por ejemplo 15 segundos es un buen valor.

Solo por curiosidad, una cadena de conexión para MS Access sería:

```
cadenaconexion="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\datos\neptuno.mdb"
```

Autenticación de Windows

Sabemos, porque lo comentamos en la instalación, que SQL Server permite autenticarse de dos formas: contra un directorio activo de Windows y mixta, es decir teniendo SQL Server sus propios usuarios. Habitualmente esta la mejor forma, ya que no siempre vamos a tener en nuestra red un directorio activo que, si recuerdas, es la estructura de Microsoft para crear y administrar las cuentas de equipos y usuarios (y mas cosas, pero básicamente eso).

En los ejemplos anterior le dijimos de utilizar la autenticación integrada de Windows: Integrated Security=SSPI. En este caso no hay que proporcionar usuario y contraseña por se comprueba la del propio servicio. Veamos las diferencias:

- Con la autenticación de SQL Sever. La base de datos mantiene sus propias cuentas de usuarios con los tipos de acceso que pueden tener a las bases de datos.
- Con la autenticación integrada de Windows. SQL Server utiliza la información de las cuentas de Windows y proporciona los tipos de acceso necesarios.

Recuerda la instalación de SQL Server de este mismo tema y verás cómo en una de las pantallas se indicó que la forma de autenticación sería la mixta. Si utilizamos la seguridad integrada debemos conceder acceso a SQL Server al usuario del servidor que crea automáticamente. Atento a los usuarios que se crean en los servidores dependiendo de la versión del IIS:

- Si utilizamos el IIS 5.1 que es el que viene con Windows XP el usuario que crea se llama "ASPNET"
- Con IIS6 (Windows 2003 Server), daremos acceso a las bbdd al grupo "IIS_WPG"
- Con IIS7 (Windows Vista y Server 2.008), daremos acceso al grupo "IIS_USERS"

La cadena de conexión que utilizaremos con Visual Web Developer 2.008 ejecutándose en nuestro ordenador será:

conexion.ConnectionString="Data Source=localhost\SQLExpress;Initial Catalog=Pubs;Integrated Security=SSPI"

Que ya está bien explicada: la base de datos "pubs" de mi equipo "localhost" con Sql Server Express y seguridad integrada.

Almacenar la cadena de conexión

Una vez que tenemos nuestra cadena de conexión creada, debemos almacenarla en algún sitio y que además esté accesible por todo nuestro web. Date cuenta que crearemos muchas páginas web y todas utilizarán esta cadena de conexión. Si la ponemos en todas funcionará bien pero no es buena idea porque si un día

cambiamos la base de datos para utilizar otro servidor debemos cambiar esta cadena de conexión en todas las páginas!. Para evitar esto debemos buscar un sitio común, que puede estar accesible por todas las páginas... un fichero de configuración de nuestra aplicación web... Efectivamente, su sitio adecuado será nuestro famoso "web.config".

Edita el fichero que tenemos en la raíz de nuestro web y pon lo siguiente antes de la etiqueta de "<system.web>":

```
<connectionStrings>
    <add name="Pubs" connectionString=
        "Data Source=localhost\SQLExpress;Initial Catalog=Pubs;Integrated Security=SSPI" />
</connectionStrings>
<system.web>
```

Como ves en la estructura le hemos dado un nombre a la conexión "Pubs" y así haremos referencia a ella en nuestro código. Además podemos crear mas cadenas para realizar varias conexiones distintas a otras bases de datos, aunque de momento nos vale con una.

En la declaración de nuestro código simplemente pondríamos:

```
Dim cadena_conexion as string= _
        WebConfigurationManager.ConnectionStrings ("Pubs").ConnectionString
```

Que utiliza la clase especial que lee valores del fichero de configuración, en este caso una conexión llamada "pubs". Vamos a crear pues una página en blanco que tenga estos controles:

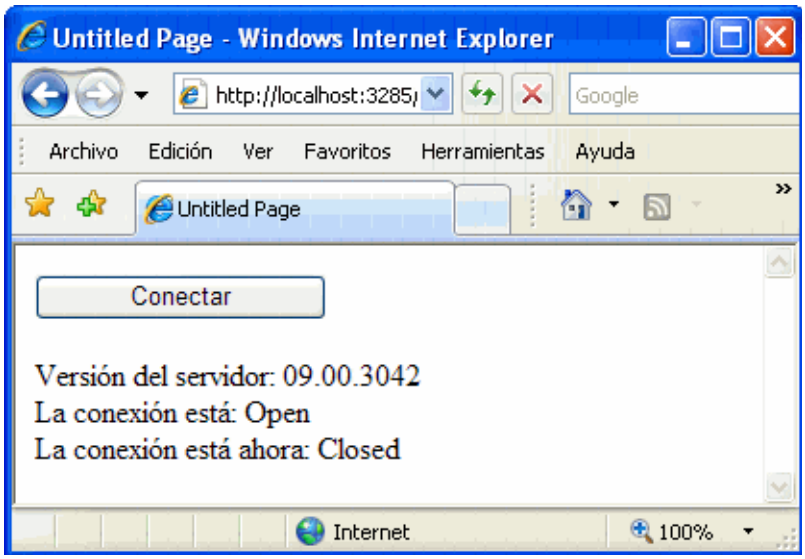
Y en el evento clic pondremos:

```
Protected Sub btn_conectar_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim cadena_conexion As String = _
            WebConfigurationManager.ConnectionStrings("Pubs").ConnectionString
    Dim conexion As New SqlConnection(cadena_conexion)
    Try
        'Intentamos abrir la conexion
        conexion.Open()
        lb_estado.Text = "Versión del servidor: " & conexion.ServerVersion & "<br />"
        lb_estado.Text &= "La conexión está: " & conexion.State.ToString() & "<br />"
    Catch err As Exception
        'Vemos el error producido:
        lb_estado.Text = "Error en la conexión de la base de datos" & "<br />"
        lb_estado.Text &= err.Message
    Finally
        'Ha ido todo bien cerramos la conexión
        conexion.Close()
        lb_estado.Text &= "La conexión está ahora: " & conexion.State.ToString()
    End Try

End Sub
```

Tendremos que poner arriba del todo de la página la importación de los espacios de nombres para nuestro SQL Server y para acceder al web.config:

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

No hay ninguna novedad, cargamos la cadena de conexión de nuestro fichero global "web.config" e intentamos abrir la conexión. Si no hay error mostramos un mensaje del estado de la conexión y en caso contrario mostramos el error. Veamos el resultado:



Hemos abierto la conexión, luego hemos escrito unos datos de la versión del servidor de base de datos y su estado y al finalizar hemos cerrado la conexión.

Es un momento interesante para aprender una nueva instrucción: "Using". Que declara que vamos a utilizar un objeto durante un corto espacio de tiempo. Cuando terminamos de utilizar el objeto inmediatamente se liberan

los recursos. El código sería:

```
Using objeto
...
end Using
```

Al finalizar el uso del objeto fuerza a un "dispose()" del objeto que es lo mismo que destruirlo y así liberamos los recursos. En nuestro objeto de conexión equivaldría a un "close()". El código quedaría:

```
Dim conexion as new sqlconnection(cadena_conexion)
try
        Using conexion
                'Intentamos abrir la conexion
                conexion.open()
                lb_estado.text="Versión del servidor: " & conexion.serverversion & " <br /
                lb_estado.text &="La conexión es:" & conexion.state.tostring() & " <br />
        end Using
Cath err as exception
        'Vemos el error producido:
        lb_estado.text="Error en la conexión de la base de datos..." & "<br />
        lb_estado.text &= err.message
end Try
'Ha ido todo bien cerramos la conexión
conexion.close()
lb_estado.text="La conexión está ahora:" & "<br />" & conexion.state.tostring()
```

Es decir, cuando hemos dejado de utilizar el objeto, después del using() se destruye, por lo tanto no hace falta cerrarlo.

5.2 . Los objetos Command y Data Reader

Una vez conectados (que no es poco) ya podemos empezar a "explotar" los datos. Para hacer esta explotación utilizaremos comandos que en ADO.NET se representa por el objeto "Command" que contiene las instrucciones que podemos ejecutar con la base de datos. Uno de estos comandos es "ExecuteReader" que crear un "Datareader" que muestra el resultado de una consulta de una tabla.

Para realizar todas las operaciones con las bases de datos ya sabes que utilizaremos el lenguaje estándar de bases de datos llamado SQL. Antes ya hablamos de él y te proporcioné un buen tutorial para aprender cómo hacer cosas con las bases de datos.

Consultas SQL

Los comandos o instrucciones SQL se envían a un servidor de base de datos o a un origen de datos, éste realiza la consulta y nos devuelve un conjunto de registros. Unas consultas son sencillas, es decir, nos devuelven todos los datos, y otras son complejas realizando filtros para obtener por ejemplo los datos bancarios de la persona con dni 165432101. Otras instrucciones no devuelven datos, simplemente ejecutan una acción como es la inserción de datos, la modificación o el borrado. Estas son las instrucciones SQL para realizar estas acciones:

- SELECT. Consulta, devuelve un conjunto de resultados
- DELETE. Borrado, elimina las filas especificadas
- INSERT. Inserción, inserta la fila con los datos proporcionados
- UPDATE. Actualización, actualiza la ficha indicada con los datos proporcionados

Una consulta de una fila en una tabla sería:

"Select nombre,apellidos from Usuarios where numero=12"

Esta es la instrucción que le vamos a proporcionar precisamente al comando para que nos devuelva un conjunto de registros. Veamos un ejemplo.

En la variable "cadenasql" hemos introducido una sencilla instrucción sql que cuando se ejecute recuperará todas las filas de la tabla empleados. Sigamos ahora necesitaremos tres objetos ADO.NET:

- Objeto Conexión, para establecer la conexión con la base de datos
- Objeto Command, para ejecutar la instrucción SQL
- Objeto DataReader, para almacenar los datos devueltos por la ejecución de la instrucción SQL:

Teniendo ya la conexión abierta podemos ejecutar entonces nuestro comando:

```
Dim comando as new sqlcommand()

comando.connection=conexion

comando.commandtext="select * from authors"
```

O podemos utilizar el constructor del objeto del comando para simplificar:

```
Dim comando as new sqlcommand("select * from authors",conexion)
```

Como ves el objeto "objcomando" necesita dos argumentos, uno es la consulta y otro una conexión. Ya tenemos la conexión, el comando de consulta y nos falta solo la recuperación de los datos mediante un "DataReader".

El objeto Datareader es el objeto mas sencillo para la recuperación de datos. Cuando queremos una consulta rápida utilizaremos siempre este método. En nuestro ejemplo nos faltaría nada mas que esto:

```
Dim resultado as SqlDataReader

resultado=comando.ExecuteReader()
```

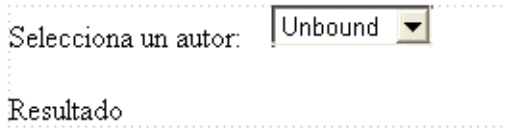
Sólo tenemos que definir la variable y ejecutar el método del comando que declaramos antes para que le asigne el resultado al datareader. Ahora ya podríamos leer todos los datos de nuestra consulta. Por ejemplo añadimos a una lista los elementos:

```
lista.Item.Add (resultado("nombre") & ", " & resultado ("apellidos"))
```

Lo combinaremos con un bucle y listos. Veamos el ejemplo completo de esta consulta.

Ejemplo 1

Creamos una página nueva "consulta.aspx". Ponemos dos "label", uno para que el usuario seleccione un autor, otro para escribir el resultado y un cuadro de lista de esta forma:



Declaramos la conexión y ponemos antes del Load para que esté a nivel global en la página y en el evento Load llamamos a un método o procedimiento para cargar los datos:

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

```
Partial Class Tema_11_base_datos_1
    Inherits System.Web.UI.Page

    Dim cadena_conexion As String = _
        WebConfigurationManager.ConnectionStrings("Pubs").ConnectionString
```

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handle
        If Not Page.IsPostBack Then
            'LLamamos al método para rellenar los datos de la lista
            rellena_autores()
        End If
    End Sub
```

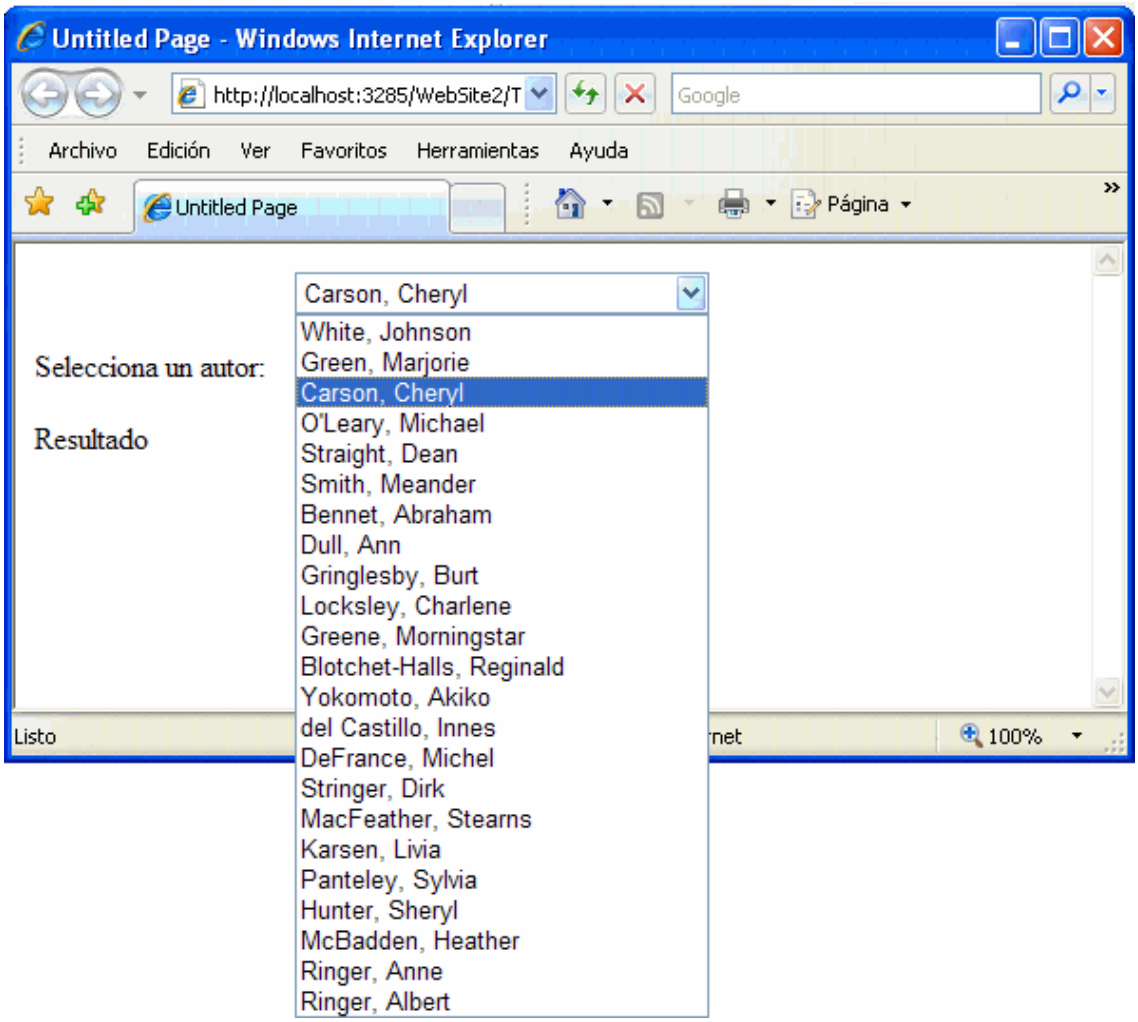
Esta es una forma muy habitual de realizar estas operaciones, en el Load llamamos a un método para preparar los datos de la página.

```
Protected Sub rellena_autores()
    'Limpiamos el cuadro de lista:
    lista_autores.Items.Clear()
    'Definimos la SQL:
    Dim sql As String = "select * 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:
    Try
        conexion.open()
        resultado = comando.executereader()
        'Bucle para la lectura:
        Do While resultado.read
            Dim dato As New ListItem()
            dato.Text = resultado("au_lname") & ", " & resultado("au_fname")
            dato.value = resultado("au_id").Tostring()
            lista_autores.Items.Add(dato)
        Loop
        resultado.close()
    Catch err As Exception
        lb_resultado.Text = "Error leyendo la base de datos. "
        lb_resultado.Text &= err.Message
    Finally
        conexion.Close()
    End Try
End Sub
```

Y el resultado es que en la carga de la página nos rellena la lista de autores, y además cuando los seleccionemos nos devolverá el identificador del autor, ya que hemos puesto:

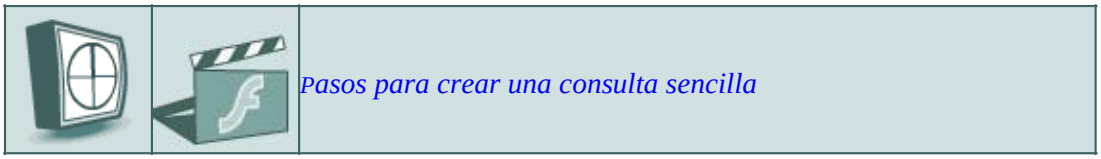
```
dato.value = resultado("au_id").Tostring()
```

y "au_id" es el índice de la tabla de autores. Veamos el resultado:



El esperado con todos los pasos que hemos dicho:

1. Crear una conexión
2. Abrirla
3. Crear un comando
4. Ejecutarlo



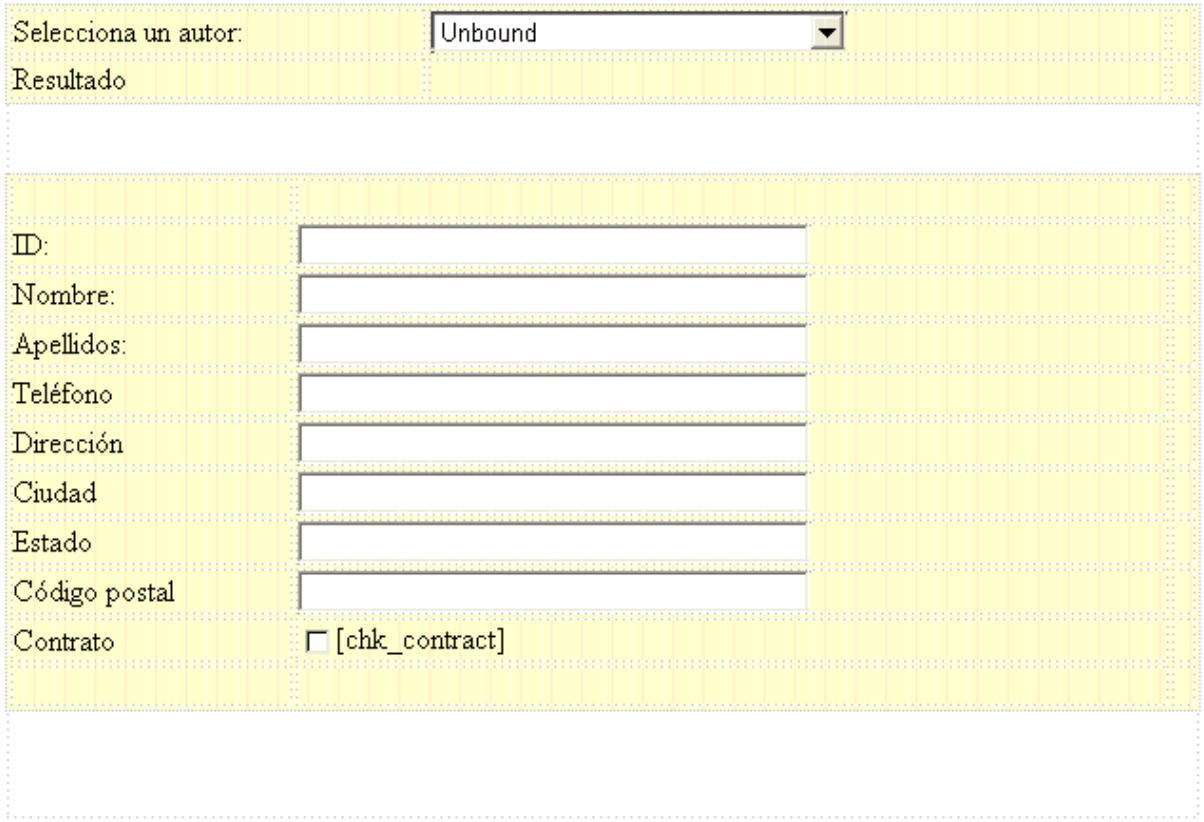
Pasos para crear una consulta sencilla

La buena noticia es que **siempre será igual** así que este es ya un buen ejemplo de cómo realizar todos los pasos de consulta a una base de datos. Ahora vamos a hacer una pequeña gran mejora. Fíjate lo que te he dicho antes de que en el cuadro de lista le hemos puesto un valor al ir añadiendo los elementos:

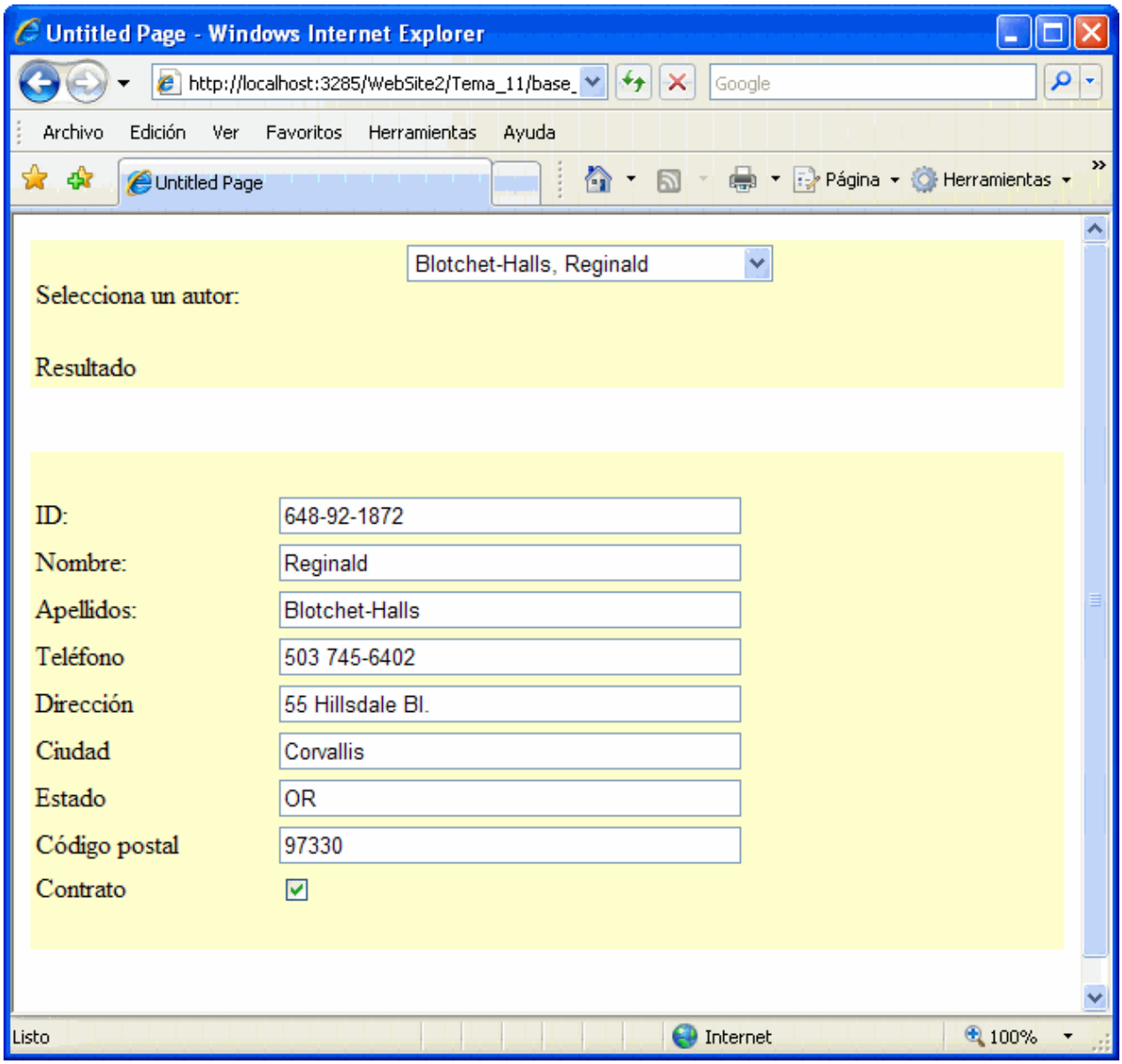
```
dato.Text=resultado ("au_lname") & ", " & resultado ("au_fname")

dato.value=resultado ("au_id").Tostring()
```

Sabemos que en los cuadros de lista podemos mostrar un texto y asignarle un valor, así cuando se selecciona podemos recuperar ese valor. En este caso le hemos añadido el índice de ese registro: resultado ("au_id")

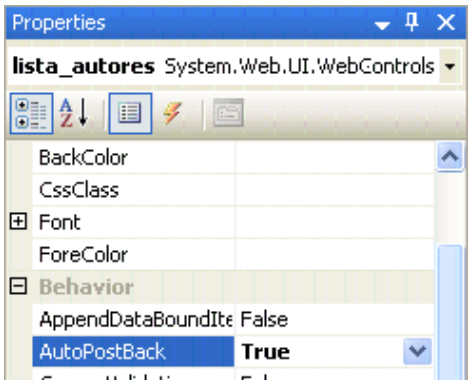
Luego podemos hacer lo siguiente. Podemos poner una serie de labels y cuadros de texto en nuestra pantalla de esta forma:



Y cuando se dispare el evento "SelectedIndexChanged" del cuadro de lista haremos una consulta de todos los registros que tienen ese índice, y que lógicamente, será sólo uno. Así leeremos todos los detalles y los mostramos en pantalla de esta forma:



Una cosa tenemos que recordar, si seleccionamos un elemento del cuadro de lista no funcionará porque por defecto no hay postback. Si recuerdas debemos activar la propiedad "autopostback" en este cuadro de lista:



El código no será muy complicado porque es un simple consulta como la anterior, pero claro tenemos que hacer todos los pasos, aunque ya sólo es un copiar-pegar de lo de antes.

```
Protected Sub lista_autores_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs)
    'Definimos la SQL:
    Dim sql As String
    sql = "select * from authors where au_id='" & lista_autores.SelectedItem.Value & "'"
    '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:
    Try
        conexion.Open()
        resultado = comando.ExecuteReader()
        'Sin bucle porque no hay mas que un elemento con ese ID
        resultado.Read()

        txt_id.Text = resultado("au_id").ToString
        txt_nombre.Text = resultado("au_fname").ToString
        txt_apellidos.Text = resultado("au_lname").ToString
        txt_telefono.Text = resultado("phone").ToString
        txt_direccion.Text = resultado("address").ToString
        txt_ciudad.Text = resultado("city").ToString
        txt_estado.Text = resultado("state").ToString
        txt_codigopostal.Text = resultado("zip").ToString
        chk_contract.Checked = CType(resultado("contract"), Boolean)
    Catch err As Exception
        lb_resultado.Text = "Error leyendo la base de datos. "
        lb_resultado.Text &= err.Message
    Finally
        conexion.Close()
    End Try

End Sub
```

En este caso como la consunta es sencilla ya que estamos haciendo la consulta con un índice único, por tanto no hay que hacer un bucle. El quiz de la cuestión está en la consulta sql, donde vamos directamente al índice que devuelve el cuadro de lista:

```
sql = "select * from authors where au_id='" & lista_autores.SelectedItem.Value & "'"
```

Sigamos con un detalle, para escribir los datos en lugar de todos esos "label" podíamos haber creado una cadena de caracteres grande para ir poniendo todos los datos. Por ejemplo:

```
dim cadena as New StringBuilder()

cadena.append ("<b>")

cadena.append (reader ("au_lname"))

cadena.append (", ")

cadena.append (reader ("au_fname"))

cadena.append ("</b><BR />")

cadena.append ("Teléfono: ")

cadena.append (reader ("phone"))
```

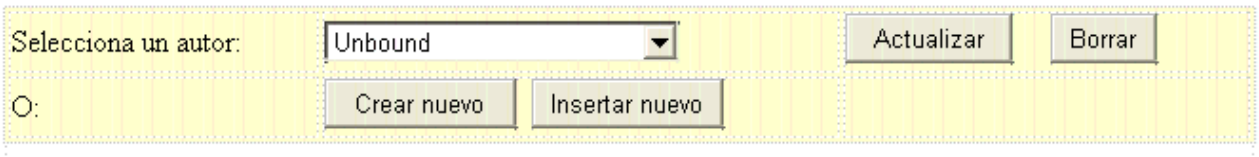
```
cadena.append ("<BR />")

...

lb_resultado.Text=cadena.Tostring()
```

Toma nota de este sistema para generar strings largos, en ocasiones viene muy bien para enlazar un resultado que luego asignaremos a un control label. Aunque es mas habitual escribirlo de forma ordenada en cuadros de texto como hemos hecho en el ejemplo anterior.

5.3 Actualizar datos

Ya sabemos recuperar datos, es ya muy sencillo y además será igual siempre. La actualización de datos implica la ejecución de un comando "Update" por ejemplo en lugar del anterior "Select". La gran diferencia que vamos a encontrar es que como aquí no se devuelven datos no necesitaremos ningún objeto "DataReader", así que la tarea en principio parece mas sencilla.

Vamos a mejorar el aspecto de nuestra consulta anterior para poder hacer todas las operaciones poco a poco. Ponemos dos áreas una para las operaciones y otra para mostrar los datos, así que nuestra parte superior quedará así:



El código es el mismo que antes, todavía no hemos hecho nada nuevo. Vamos a ver como actualizaríamos un registro. Esto lo haremos en el evento clic del botón "nuevo registro" . Lo primero que haremos será limpiar los cuadros de texto que mostramos a los usuarios y le mostraremos un texto para que meta los datos:

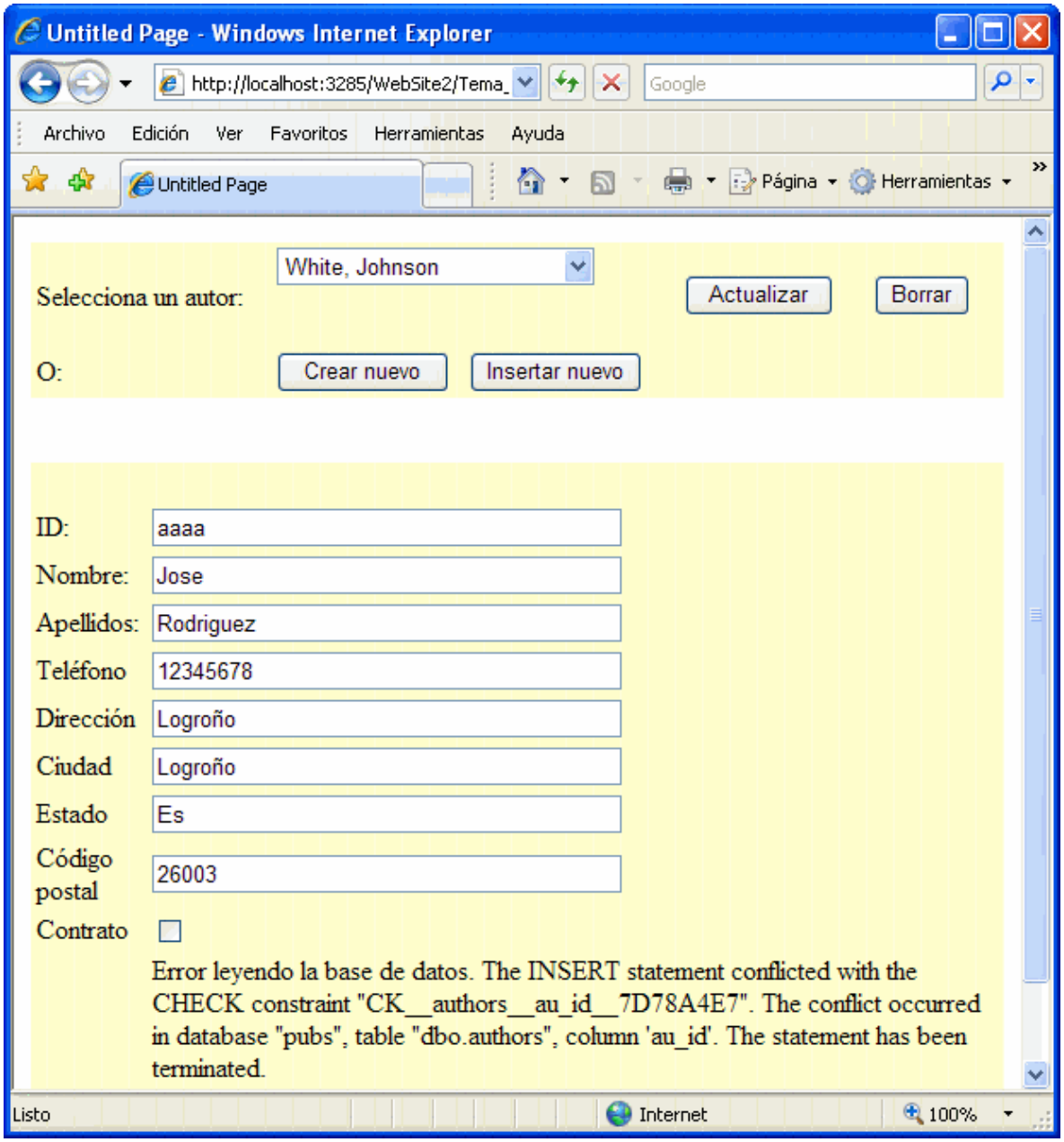
```
Protected Sub btn_crear_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handl
    txt_id.Text = ""
    txt_nombre.Text = ""
    txt_apellidos.Text = ""
    txt_telefono.Text = ""
    txt_direccion.Text = ""
    txt_ciudad.Text = ""
    txt_estado.Text = ""
    txt_codigopostal.Text = ""
    chk_contract.Checked = False
    lb_resultado.Text = "Haga clic en Crear nuevo para inserta un nuevo registro..."
End Sub
```

Ahora meteremos en el evento clic del botón de "Insertar" el código necesario que será parecido hasta ahora excepto en la creación de un datareader que ahora no tiene sentido:

```
Protected Sub btn_insertar_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
    'Podemos hacer aqui la comprobación o poner controles de validadción
    If txt_id.Text = "" Or txt_nombre.Text = "" Or txt_apellidos.Text = "" Then
        lb_resultado.Text = "No has introducido el identificador, nombre o apellido"
        Return
    End If

    'Definimos los objetos
    Dim sql As String
    sql = "Insert into authors (au_id,au_fname,au_lname,phone,address,city,state,zip,contract) "
    sql &= " values ('"
    sql &= txt_id.Text & " ','"
    sql &= txt_nombre.Text & " ','"
    sql &= txt_apellidos.Text & " ','"
    sql &= txt_telefono.Text & " ','"
    sql &= txt_direccion.Text & " ','"
    sql &= txt_ciudad.Text & " ','"
    sql &= txt_estado.Text & " ','"
    sql &= txt_codigopostal.Text & " ','"
    sql &= Val(chk_contract.Checked) & "')"

    'Definimos los objeto ADO.NET:
    Dim conexion As New SqlConnection(cadena_conexion)
    Dim comando As New SqlCommand(sql, conexion)
    Dim operacion As Integer = 0
    'Abrimos la base de datos y leemos:
    Try
        conexion.Open()
        operacion = comando.ExecuteNonQuery
        lb_resultado.Text = "Insertados " & operacion & " registros."
    Catch err As Exception
        lb_resultado.Text = "Error leyendo la base de datos. "
        lb_resultado.Text &= err.Message
    Finally
        conexion.Close()
    End Try

    'Para terminar si hemos añadido alguno volvemos a cargar la lista:
    If operacion > 0 Then
        rellena_autores()
    End If
End Sub
```

Las novedades son pocas pero interesantes. Por un lado nos ha sido laboriosa la labor de crear la sentencia SQL, son muchos campos y hasta concatenar la instrucción completa hemos tenido que teclear un poco.

Luego hemos ejecutado el método operación=comando.executeNonQuery que se ejecuta cuando no se devuelve un conjunto de resultados. En este caso devuelve un valor entero indicando las filas que se han visto afectadas. Si ejecutamos la página, nos devuelve un extraño error:



Que veremos mas adelante porque tiene que ver precisamente por las relaciones que existen en las tablas. De momento no es importante, quédate con lo aprendido. Pero la sintaxis de la escritura de SQL ha sido muy laboriosa, esto hay que mejorarlo...

Crear comandos mas robustos

Hemos visto lo laborioso de la escritura de la sentencia SQL y que además es tremendamente frágil porque se puede dar que:

- Haya caracteres erróneos. Por ejemplo un apellido como "O'brian" daría un problema, porque el carácter ' ya has visto que es el separador de los literales. En este caso el usuario recibiría un error de ejecución
- Los usuarios pueden realizar una "inyección de SQL" que es uno de los modos de ataques mas comunes para intentar acceder a sitios web. Esta técnica es muy utilizada y es la de meter código SQL dentro de un campo de datos. No es de este curso pero te pongo dos enlaces de información sobre esta técnica: Wikipedia Microsoft

Vemos entonces la necesidad de mejorar esa retahíla que hemos escrito antes para construir la sentencia y prevenir los errores por introducción de caracteres erróneos. Una forma sencilla aunque poco elegante sería:

```
Dim id_autor as string = txt_id.text.Replace ("'","''")
```

Es decir creamos otra variable donde hemos sustituido ese carácter ' por una comilla doble ", entonces si nos funcionará. Pero claro deberíamos crear tantas variables como campos. La mejor solución va a ser crear un comando parametrizado que simplemente nos va a sustituir unos valores por otros según le decimos la posición. Tranquilo, es fácil, mira la sentencia:

```
Select * from Clientes where cliente_id= 'ABCD'
```

que sustituiremos por:

```
Select * from Clientes where cliente_id= @cliente_id
```

En esa variable @cliente_id meteremos nuestro contenido del cuadro de texto de antes pero ahora la construcción de la SQL será siempre correcta, evitando tanto los caracteres erróneos como el fallo de seguridad de la "inyección de SQL". En inglés le llama "placeholders" que digamos vienen a ser unos "huecos" donde luego pondremos los valores definitivos. Veamos como quedaría y verás que es sencillo. En la parte de la SQL es solo una cadena "string", toda seguida con esos indicadores:

```
Dim sql As String
sql = "Insert into authors (au_id,au_fname,au_lname,phone,address,city,state,zip,contract)
sql &= " values ("
sql &= " @au_id,@au_fname,@au_lname,@phone,"
sql &= " @address,@city,@state,@zip,@contract)"
```

Y luego le asignamos los valores reales (omito parte del código pare solo lo interesante):

```
'Definimos los objeto ADO.NET:
Dim conexion As New SqlConnection(cadena_conexion)
Dim comando As New SqlCommand(sql, conexion)

comando.Parameters.AddWithValue("@au_id", txt_id.Text)
comando.Parameters.AddWithValue("@au_fname", txt_nombre.Text)
comando.Parameters.AddWithValue("@au_lname", txt_apellidos.Text)
comando.Parameters.AddWithValue("@phone", txt_telefono.Text)
comando.Parameters.AddWithValue("@address", txt_direccion.Text)
comando.Parameters.AddWithValue("@city", txt_ciudad.Text)
comando.Parameters.AddWithValue("@state", txt_estado.Text)
comando.Parameters.AddWithValue("@zip", txt_codigopostal.Text)
comando.Parameters.AddWithValue("@contract", Val(chk_contract.Checked))

Dim operacion As Integer = 0
'Abrimos la base de datos y leemos:
```

Las diferencias son muy pocas, simplemente creamos una sentencia SQL mucho mas sencilla al no tener que concatenar los valores de los cuadros de texto y luego hacemos la asignación. Es imprescindible que trabajes así. Con esto evitarás muchos problemas en la creación de la SQL, sobre todo cuando son muy extensas.

Nota para el proveedor de datos de OLEDB este indicador de campos se debe poner con un "?"

Actualizar registros

Ya hemos hecho dos procesos, el de consultar y el de insertar, vamos ahora a actualizar un registro. Como valor absolutamente imprescindible tenemos el índice único del registro. Es decir el "au_id" del registro que queremos actualizar. Esto es tan importante que si no existiera un índice único sería imposible actualizar o borrar una fila. Aunque siempre habrá un valor: código de cliente, DNI del usuario, número de empleado, ... y si no lo hubiera crearíamos un valor autonumérico, es decir un entero que se incrementan automáticamente y que definíamos en SQl Server como "identity" que nos va a garantizar que siempre vamos a tener un valor único en la fila y así lo utilizaremos como índice.

Sigamos, sabemos que la sintaxis SQL de la actualización es "Update...." así que tendremos que realizar las mismas operaciones que antes pero en el evento clic del botón de actualizar y construyendo la sentencia SQL adecuada, todo lo demás es igual:

```
'Definimos los objetos
Dim sql As String
sql = "update authors set"
sql &= " au_fname=@au_fname, au_lname =@au_lname,"
sql &= " phone=@phone, address=@address,"
sql &= " city=@city,state=@state,"
sql &= " zip=@zip,contract=@contract"
sql &= " where au_id=@au_id_original"

'Definimos los objeto ADO.NET:
Dim conexion As New SqlConnection(cadena_conexion)
Dim comando As New SqlCommand(sql, conexion)

comando.Parameters.AddWithValue("@au_fname", txt_nombre.Text)
comando.Parameters.AddWithValue("@au_lname", txt_apellidos.Text)
comando.Parameters.AddWithValue("@phone", txt_telefono.Text)
comando.Parameters.AddWithValue("@address", txt_direccion.Text)
comando.Parameters.AddWithValue("@city", txt_ciudad.Text)
comando.Parameters.AddWithValue("@state", txt_estado.Text)
comando.Parameters.AddWithValue("@zip", txt_codigopostal.Text)
comando.Parameters.AddWithValue("@contract", Val(chk_contract.Checked))
comando.Parameters.AddWithValue("@au_id_original", lista_autores.SelectedItem.Value)
```

Como ves es muy parecido a lo visto antes pero tenemos una diferencia muy grande e importante, fíjate al final de la sentencia SQL:

```
sql &=" where au_id=@au_id_original"
```

Significa que actualizaremos todos los registros que coincidan con ese identificador de registro. Como es clave única será sólo uno. ¿y cual debo actualizar? pues precisamente el del valor de la fila seleccionado en el cuadro de lista. Además tenemos estos detalles:

- No necesitamos un Datareader por que no se devuelven datos
- Se utiliza un comando dinámico "Update" mediante un objeto "Command". Este comando busca los campos del registro y los sustituye por los valores proporcionados por los cuadros de texto
- No actualizamos el identificado único "ID" porque es el índice del elemento que debemos actualizar
- El método "ExecuteNonQuery" devuelve el número de filas afectadas, que solo será una al decirle la clave única del registro que debe modificar. Una vez realizada llamada llamamos a rellenar el cuadro de lista otra vez.

Borrar un registro

Cuando el usuario pulse el botón debemos borrar el elemento de la lista con el índice adecuado que será el mismo que antes. Este caso es el mas sencillo de todos porque la instrucción SQL es mas sencilla:

```
'Definimos los objetos
Dim sql As String
sql = "Delete from authors "
sql &= " where au_id=@au_id_original"

'Definimos los objeto ADO.NET:
Dim conexion As New SqlConnection(cadena_conexion)
Dim comando As New SqlCommand(sql, conexion)

comando.Parameters.AddWithValue("@au_id_original", lista_autores.SelectedItem.Value)

Dim operacion As Integer = 0
```

Nos funciona pero nos da un problema ya que como este autor tiene libros suyos en la tabla de los libros, nos avisa que se rompe la relación. Es normal, imagina el caso del banco con las operaciones del cajero de un usuario. Si el banco da de baja al usuario de la tabla de clientes deberá borrar también los movimientos de la tabla de movimientos. Sino, puesto que la tabla de movimientos tenía el como índice el número de cliente no se podrán acceder a sus datos ya que ese número de cliente desaparece al borrar su ficha. Luego por coherencia de datos al borrar un "dato maestro" habría que borrar todos los datos asociados de las tablas.

Instrucciones SQL con parámetros

5.4 Acceso a datos desconectado

Cuando utilizamos datos desconectados lo que hacemos es mantener en memoria una copia de los datos utilizando un "DataSet". Cuando haya cambios nos volveremos a conectar y actualizaremos los datos de este DataSet en la base de datos.

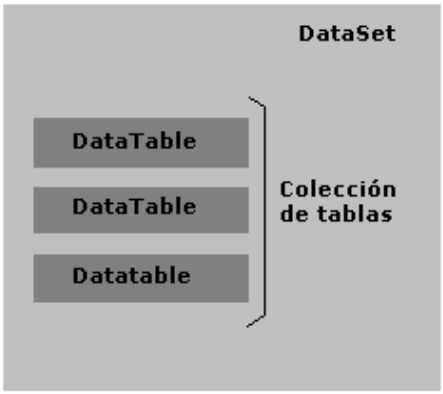
Antes simplemente hacíamos una lectura de un DataReader que no almacena nada en memoria, según se lee se vuelca en la página web y no volvemos a acceder a él (aunque mantiene viva una conexión). Con los DataSet mantendremos en memoria un conjunto de resultados que se desconecta completamente de la bbdd, que podremos modificar y luego volcarlos otra vez al servidor de base de datos. Tenemos varias razones por las que hay que utilizar esta otra forma de trabajar con los datos en memoria:

- Necesitamos utilizar mucho tiempo la base de datos. Con un DataSet conseguimos volcar desde la base de datos a nuestra aplicación y viceversa de una forma rápida y así reducimos el tiempo de uso.
- Necesitamos rellenar un control de datos, por ejemplo, de cuadrícula de ASP.NET. El Datareader es muy limitado y no nos permitirá muchas de las funciones que proporcionan los controles de cuadrícula. Por ejemplo, la paginación de resultados, con el Datareader no se puede y con los DataSet si
- Navegación por los resultados en los dos sentidos. Con los DataReader solo podemos hacer un bucle desde el primer registro al último. Con lo Dataset podemos movernos libremente por los registros en los dos sentidos.
- Podemos movernos de una tabla a otra. En un DataSet podemos almacenar varias tablas con datos e incluso definir relaciones entre ellas para mejorar la navegación.
- Con los DataSet podemos almacenar los datos para su uso posterior. Por ejemplo, exportar una consulta a XML.
- Proporciona una forma de empaquetar datos que podemos enviar a otros objetos.

- Podemos almacenar datos que pueden ser utilizados mas adelante.

Los DataSet almacenan los cambios que se realizan en él (modificar un nombre de un empleado en la tabla de empleados cargada en el DataSet). Cuando hemos hecho los cambios provocaremos un "Update" que actualizará los cambios en la base de datos. Por eso se llama acceso desconectado, porque se carga un DataSet, se cierra la conexión de la bbdd, se realizan los cambios necesarios y se vuelcan a la bbdd.

De todas formas en la práctica se utilizan mucho los Dataset pero a la hora de modificar datos se ejecuta la SQL necesaria. En una aplicación Windows, el Dataset permanece en memoria mucho tiempo y la sincronización es sencilla. En Web se carga el DataSet y se desconecta hasta que el usuario hace alguna solicitud en la página. Esto hace que el Dataset es distinto en los momentos en los que se sirve al usuario y cuando vuelve a solicitarlos para una operación. En el programa Windows está permanentemente en memoria pero aquí no por lo tanto puede haber estas diferencia. Por eso te comento lo de que se utiliza mucho para enlazarlo con controles complejos porque proporcionan mucha potencia y a la hora de modificar datos se utiliza SQL. Recuerda que en un Dataset podemos almacenar varias tablas:



Esto nos será muy útil cuando queramos tener varias tablas dentro..

Seleccionar datos desconectados

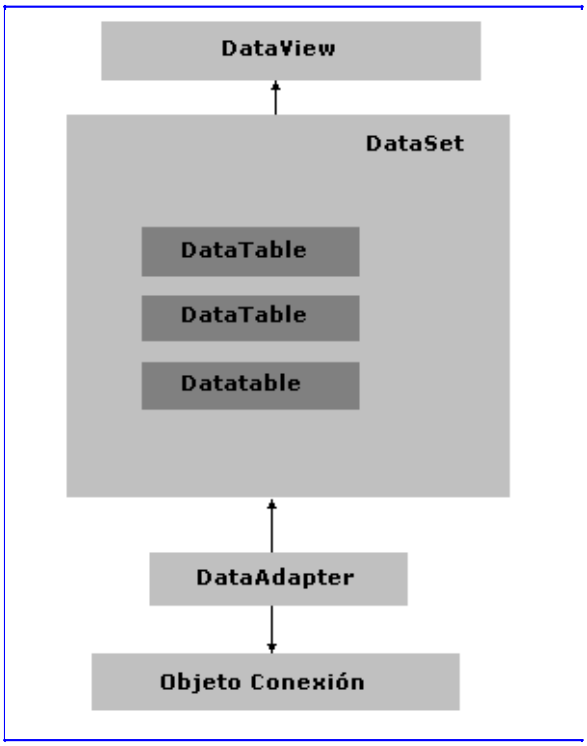
Con el acceso a datos desconectados se mantiene una copia de los datos en memoria mientras el código se está ejecutando. Para utilizar el objeto dataset debemos importar el espacio de nombres " " a nuestra página:

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

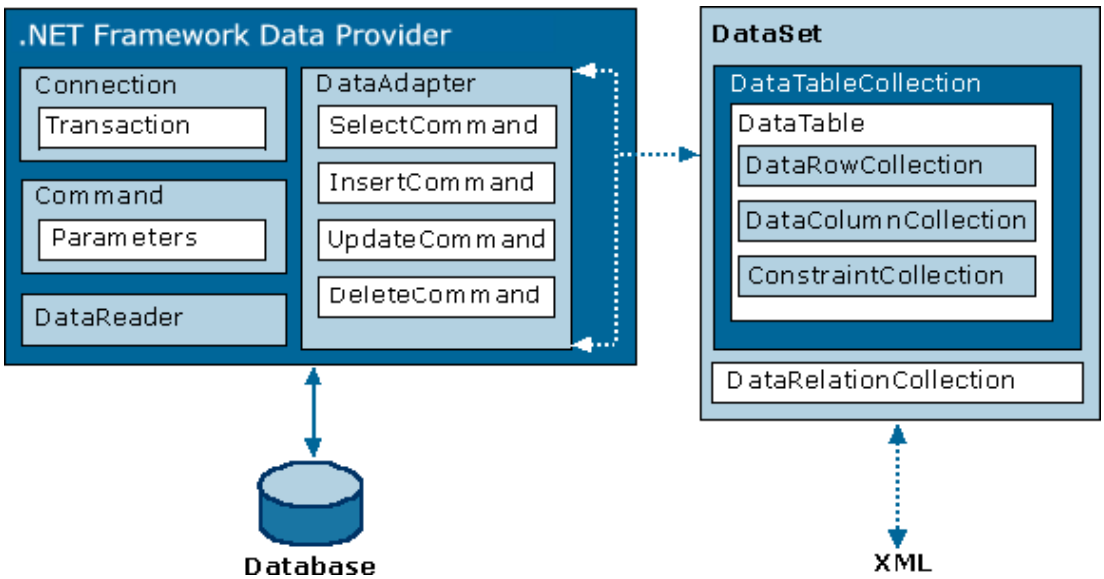
Rellenaremos el DataSet de la misma forma que el DataReader, pero mientras el DataReader mantiene viva una conexión, el DataSet siempre está desconectado. Veamos un ejemplo para realizar una consulta y almacenarla en un DataSet:

```
Protected Sub rellena_autores()
    'Limpiamos el cuadro de lista:
    lista_autores.Items.Clear()
    'Definimos los objetos ADO
    Dim sql As String
    sql = "Select au_lname,au_fname,au_id from authors"
    Dim conexion As New SqlConnection(cadena_conexion)
    Dim comando As New SqlCommand(sql, conexion)
    Dim Adapter As New SqlDataAdapter(comando)
    Dim datos As New DataSet()
    'Abrimos la base de datos y leemos:
    Try
        conexion.open()
        'El comando va a crear un DataTable llamada "autores" dentro del Dataset:
        Adapter.Fill(datos, "Autores")

    Catch err As Exception
        lb_resultado.Text = "Error leyendo la base de datos. "
        lb_resultado.Text &= err.Message
    End Try
    'Bucle para recorrer el DataSet:
    For Each fila As DataRow In datos.tables("autores").Rows
        Dim elemento As New ListItem()
        elemento.Text = fila("au_lname") & ", " & fila("au_fname")
        elemento.Value = fila("au_id").ToString()
        lista_autores.Items.Add(elemento)
    Next

End Sub
```

Veamos ahora la explicación. En este caso también nos va a pasar como en los otros casos, aprenderemos algo nuevo con esto de los DataSet pero serán siempre iguales, así que esta colección de ejemplos será tu punto de partida para cuando quieras hacer estas operaciones. Siempre que queramos extraer los datos a un Dataset necesitaremos crear un DataAdapter, fíjate en el esquema:



Cada objeto DataAdapter puede tener cuatro comandos que te imaginarás serán para las cuatro operaciones (altas, bajas, consultas y modificaciones): SelectCommand, InsertCommand, UpdateCommnad y DeleteCommand:

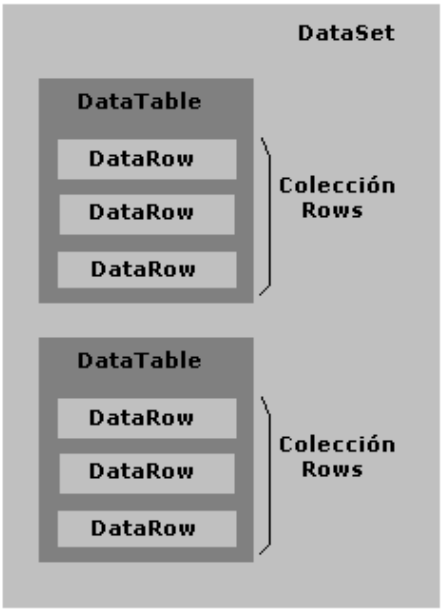


Esto nos permite utilizar un DataAdapter para realizar múltiples tareas. El objeto Command proporcionado por el constructor se asigna automáticamente a la propiedad "DataAdapter.SelectCommand", ya que es una operación de consulta la que estamos realizando.

El método DataAdapter.Fill por fin nos rellena una tabla de información en el DataSet. En nuestro caso la tabla que le hemos puesto de nombre "autores", como ves es arbitrario, le podemos poner cualquier sin tener

porqué coincidir con las tablas de la base de datos accedidas. Luego utilizaremos este nombre para acceder a sus datos.

Para acceder a las filas individuales podemos hacer un bucle por la colección de filas "rows" de la tabla que indiquemos. Fíjate en el esquema de cómo es un DataSet



Por tanto puede almacenar muchas DataTables y cada una con su colección de filas (rows) y además podemos acceder individualmente a sus campos row ("au_lname")

Seleccionar varias tablas

Como hemos comentado podemos almacenar varias tablas en nuestro DataSet, según vayamos necesitando. Y además podremos establecer una relación entre ellas de forma similar a la que tienen en el origen. SQL Server o Access pueden definir una serie de relaciones en las tablas en el diseño de la base de datos. Aquí podemos crear esa relación entre las tablas pero no podemos cargar las tablas y relaciones que están definidas en la base de datos SQL Server.

Me explico, el diseñador de la base de datos creó las tablas para su aplicación en SQL Server y luego utilizó una herramienta para, dentro de SQL Server, definir las relaciones entre las tablas de clientes y pedidos por ejemplo. Dentro de un Dataset podemos cargar estas tablas pero no nos carga la relación ya que es un elemento que se puso en SQL Server. en cualquier caso en los DataSet podemos crear estas relaciones para poder trabajar con ellas de forma similar. La idea es que aunque hayamos definido relaciones en el origen, éstas no se cargarán y tendremos que definirlas en nuestro DataSet.

En nuestra base de datos de ejemplo los autores están relacionados con los libros utilizando tres tablas:

Esta relación de tipo "muchos" a "muchos" permite a varios autores relacionarse con un título y un título con varios autores. Como vimos al aprender algo de las relaciones muy pocas veces accederemos a las tablas individualmente. Lo normal es que accedamos a varias a la vez, por ejemplo la lista de autores con sus. Si sacásemos la lista de libros de un autor tendríamos como nombre su identificador, por tanto tendremos que

cruzar ese identificador de autor con la tabla de autores para poder extraer los datos del nombre y apellidos de los autores.

En este ejemplo crearemos una página con la lista de autores y libros escritos con el enlace de las tablas como novedad. Utilizaremos el esquema visto antes de las tres tablas. Para empezar crearemos los objetos estándar de ADO.NET, incluido el DataSet, esto ya es universal. Los cambios vienen ahora cuando metamos datos en él. Estos pasos los haremos en el método "Crear_lista" que se llama desde el "Page_Load":

```
Protected Sub rellena_autores()
    'Limpiamos el cuadro de lista:
    lista_autores.Items.Clear()
    'Definimos los objetos ADO
    'Definimos los objetos
    Dim sql As String = "select au_lname,au_fname,au_id from authors"
    Dim conexion As New SqlConnection(cadena_conexion)
    Dim comando As New SqlCommand(sql, conexion)
    Dim adapter As New SqlDataAdapter(comando)
    Dim datos As New DataSet
    'Abrimos la base de datos y leemos:
    Try
        conexion.Open()
        adapter.Fill(datos, "Autores")
        comando.CommandText = "Select au_id,title_id from Titleauthor"
        adapter.Fill(datos, "libros_autores")
        comando.CommandText = "Select title_id,title from Titles"
        adapter.Fill(datos, "libros")
    Catch err As Exception
        lb_resultado.Text = "Error leyendo la base de datos. "
        lb_resultado.Text &= err.Message
    Finally
        conexion.Close()
    End Try


End Sub
```

Como ves primero hemos añadido en un Dataset la tabla de "Autores", luego la de "libros_autores" y finalmente la de "libros". La primera tiene la lista de autores de nuestra tienda, la tercera la lista de libros que tenemos para vender y la segunda tabla relaciona que libros son de que autores. Ahora es cuando veremos como hacer internamente esta relación...

Definir relaciones

Ahora que tenemos las tablas en nuestro Dataset tenemos que definir dos objetos para definir las relaciones (DataRelation) para poder realizar la navegación de una forma mas sencilla. En este caso los objetos DataRelation coinciden con las claves externa definidas en la base de datos.

Para crear estas relaciones necesitamos especificar los campos que se van a enlazar en las dos tablas y por fin crear la DataRelation con un nombre. El orden de los enlaces es importante: el primero es el padre y el segundo el hijo, ya que la idea es que un padre tenga muchos hijos pero los hijos un solo padre (así suele ser en la realidad :-)). O lo que es lo mismo, una relación de uno a muchos de esta forma:

```
Dim libros_librosautor as new datarelation ("libros_librosautor", datos.tables ("titles").columns

 ("title_id"), datos.tables ("titulosautor").columns ("title_id"))

Dim autores_librosautor as new datarelation ("autores_librosautor",_

datos.tables ("authors").columns("au_id"), datos.tables ("titulosautor").columns ("au_id"))
```

Una vez creadas las relaciones, las añadimos al DataSet:

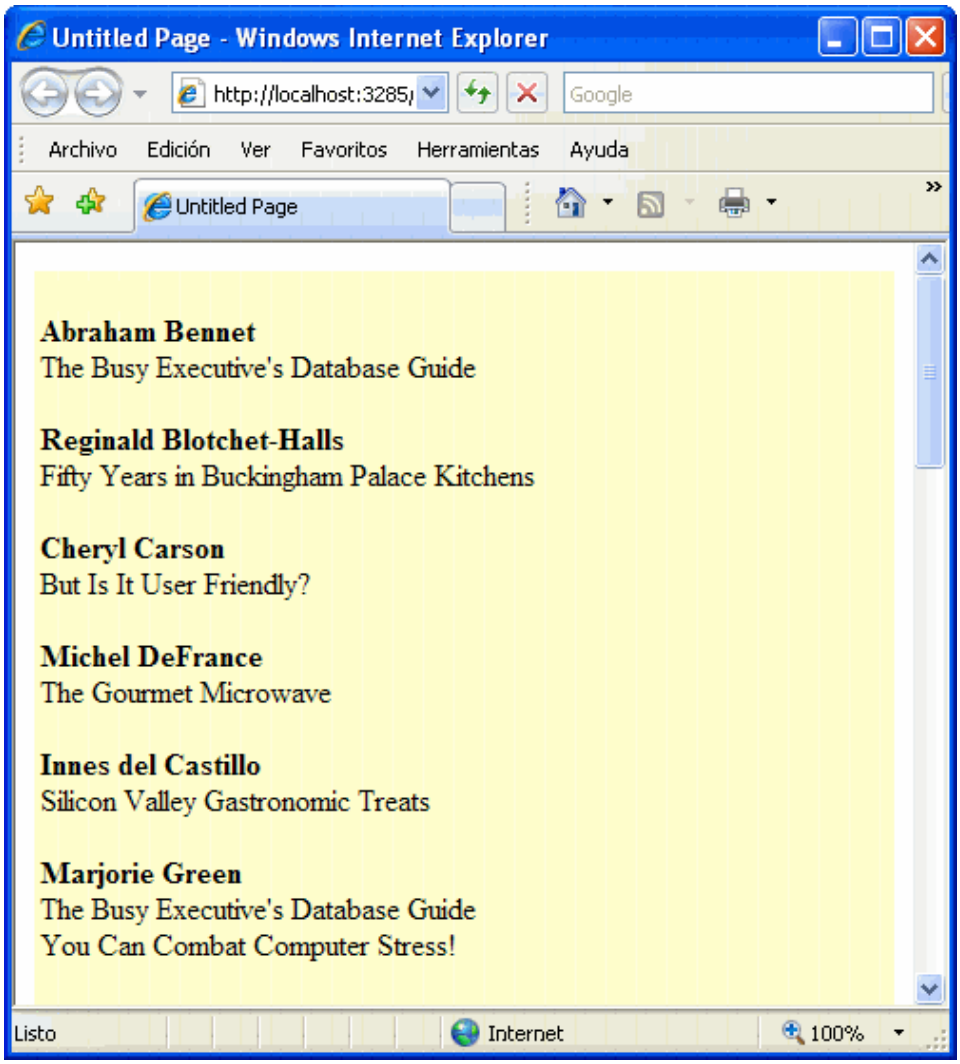
```
datos.Relations.Add (libros_librosautor)

datos.Relations.Add (autores_librosautor)
```

El siguiente código ya es recorrer el Dataset:

1. Seleccionamos el primer registro de la tabla de autores
2. Con la relación "autores_librosautor" buscamos el primer hijo que corresponda a ese autor. Este paso utiliza el método "GetChildRows" de "DataRow"
3. Para cada registro encontrado en "Titleauthor" buscamos el registro del libro para obtener el título completo. Utilizaremos el método GetParentRows de "Datarow"
4. Nos movemos al siguiente autor para repetir el proceso.

```
For Each fila_autor As DataRow In datos.Tables("Autores").Rows
    lb_lista.Text &= "<br /><b>" & fila_autor("au_fname")
    lb_lista.Text &= " " & fila_autor("au_lname") & "</b><br />"
    For Each fila_tituloautor As DataRow In fila_autor.GetChildRows(autores_librosautor)
        For Each fila_libro As DataRow In fila_tituloautor.GetParentRows(libros_librosautor)
            lb_lista.Text &= "  "
            lb_lista.Text &= " " & fila_libro("title") & "<br />"
        Next
    Next
Next
```

El resultado es pues:



Si los autores y los libros tienen una relación de uno a muchos podríamos simplificar el algoritmo anterior:

```
For each fila_autor as Datarow in datos.tables ("Autores").Rows

    For each fila_libro as Datarow in fila_tituloautor.GetParentRows (libros_librosautor)

       next

       next
```

Esto es lo mas complejo que nos podremos encontrar, pero tranquilo, estamos en la fase de aprendizaje. Verás en el capítulo siguiente como se hace todo mas fácil con el capítulo de los controles enlazados a datos.

Además utilizaremos formas mas sencillas porque en la tabla de los Dataset meteremos consultas que ya hacen el enlace entre las tablas, o lo que es lo mismo en SQL, las "join"

Pulsa aquí para descargar los ejemplos de este tema

Ejercicios

Ejercicio 1

Crea un mapa del sitio "web.sitemap" para poner un menú del ejercicio anterior. Crea dos menús, uno de árbol y otro desplegable para los dos menús de la página maestra:

Debajo del título en la página maestra pon un menú de tipo:

Ejercicio 2:

Comprueba que tienes correctamente instalado y funcionando el servidor de base de datos SQL Server. Crea dos tablas:

- "Usuarios" con los campos:

- id: que será un identificado único de tipo identity
- nombre: de tipo string (50)
- apellidos: de tipo string (50)
- dirección: de tipo string (50)
- fecha de nacimiento de tipo Fecha
- localidad de tipo string (50)
- provincia: será un entero que luego apuntará a la tabla de provincias
- sexo: de tipo bit. True será hombre y false mujer

Nombre de columna	Tipo de datos	Permitir v...
id	int	☐
nombre	nvarchar(50)	☑
apellidos	nvarchar(50)	☑
direccion	nvarchar(50)	☑
fecha	smalldatetime	☑
localidad	nvarchar(50)	☑
provincia	int	☑
sexo	bit	☑
		☐

- "provincias"

  - id: identificado único para la provincia de tipo identity
  - provincia de tipo string (50)

Nombre de columna	Tipo de datos	Permitir v...
id	int	☐
provincia	nvarchar(50)	☑
		☐

Pon unos datos de prueba de esta forma:

id	provincia
1	Barcelona
2	Vizcaya
3	La Rioja
4	Madrid
5	Valencia
6	Guipuzcoa
7	Cantabria
8	Sevilla
9	Navarra
NULL	NULL

y:

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