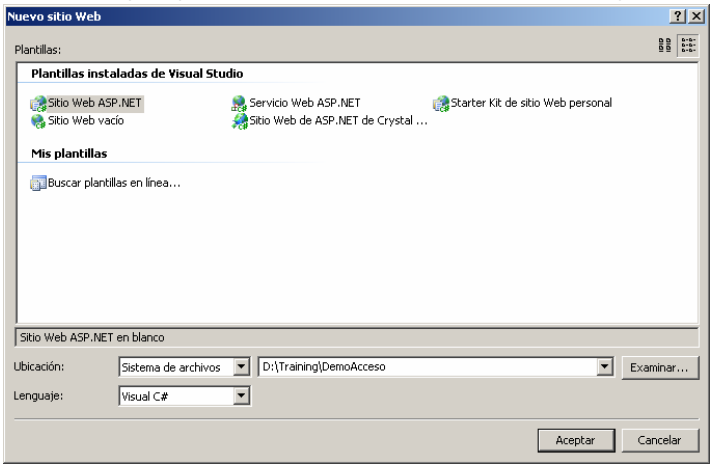
Este documento muestra la forma de acceder a una aplicación Web desarrollada por nosotros usando autenticación por Formularios, algún control de .NET y un poco de buenas prácticas en lenguaje C# y con una Base de Datos SQL Server 2005 tratando de ser lo más claro y explicativo posible.

¿Porque C# y no Visual Basic?… simple, Microsoft esta tratando de guiar las tendencias hacia este lenguaje, y lo esta consiguiendo, así, existe mucha documentación en C#, además, he comprobado que cuando programamos en VB .NET, normalmente utilizamos algunas instrucciones de VB 6, entonces, no aprovechamos al máximo el FrameWork de .NET. Para comprobar esto, bastaría con mencionar que cuando creamos un proyecto en VB .NET no nos molestamos en quitarle la Referencia de **Microsoft.VisualBasic**, que trae el soporte para compatibilidad con VB 6 y que nos permite ejecutar algunas instrucciones que son propias de Visual Basic 6. Por esta razón, conviene o quitarse definitivamente todas las costumbres de VB 6 y empezar a escribir bien VB .NET o nos sumergimos en C#, que nos quita todas esas costumbres y de paso nos damos cuenta que no es tan complicado como parece al principio.



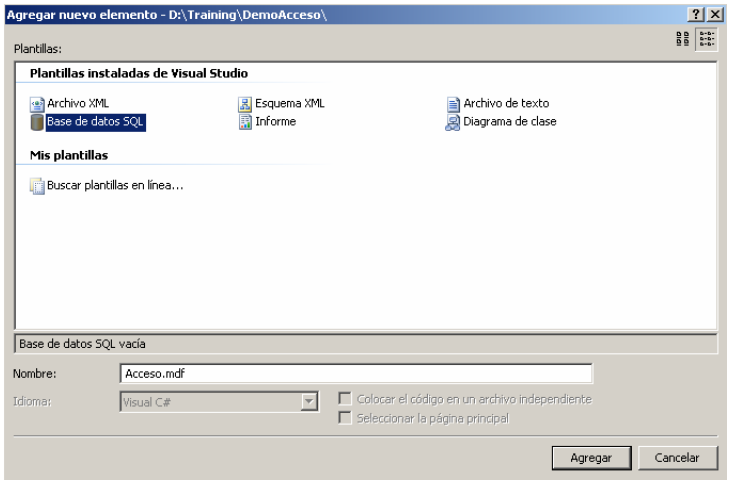
Para empezar, iniciaremos un Sitio Web y le llamaremos DemoAcceso tal y como se muestra en la siguiente imagen.

Siempre es recomendable utilizar otra unidad que no sea la unidad C para trabajar. Recordemos que al dejar la opción Ubicación en "Sistema de Archivos" no tenemos el problema de la versión anterior de .NET, porque la versión 2005 incluye un Servidor Web que podemos utilizar en nuestro ambiente de desarrollo sin necesidad de tener instalado el IIS, pero si será necesario el IIS cuando pasemos al ambiente de producción.

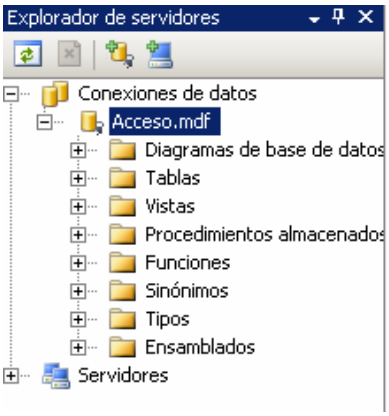
Al iniciar el proyecto tendremos la siguiente ventana, en la que se puede ver una Carpeta ASP .NET (se llama así porque es el framework quien la administrará y cada carpeta ASP .NET cumple un papel específico) de nombre App_Data, dentro de la cual se colocará nuestra Base de Datos. Además esta presente la página Default.aspx, que contiene la interfaz de Usuario, o sea la página web propiamente que los visitantes de nuestro Web verán. Hay que aclarar que por cada página que se cree se creará además por defecto un archivo con el mismo nombre y con extensión según el lenguaje (para el caso de C# será .cs, para Visual Basic será .vb, etc.) que contendrá el código que ejecutaremos por cada evento que se trabaje en la página.

Lo primero que haremos será crear nuestra Base de Datos y una tabla con el nombre **Usuarios**, que será la que contiene los datos de los usuarios con acceso a nuestra aplicación Web.

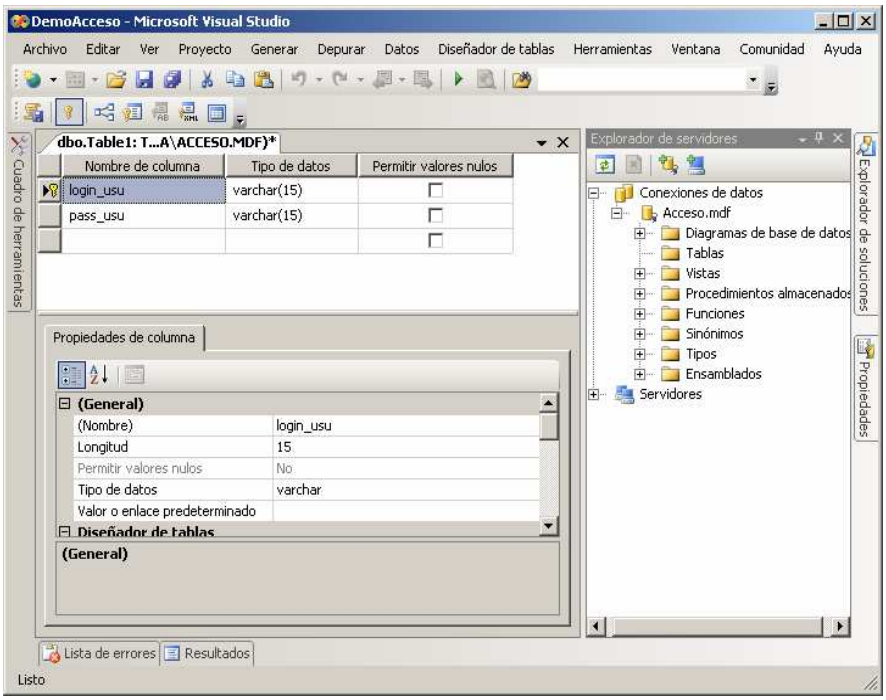
Haga clic sobre la carpeta ASP .NET llamada App_Data y ha continuación con el clic derecho del mouse sobre esta carpeta seleccione la opción **Agregar nuevo Elemento**, en la ventana que se mostrará a continuación seleccione el icono **Base de Datos SQL** y escriba el nombre **Acceso.mdf** tal como se muestra en la siguiente pantalla.



Luego de haber hecho esto se deberá cargar el **Explorador de Servidores** (si no se muestra, haga clic en el Menú Ver, opción Explorador de Servidores). Esta ventana muestra la Base de Datos que hemos creado, las carpetas corresponden a los diferentes Objetos de la Base de Datos que podemos crear. Es mas, ni siquiera necesito abrir el Servidor de SQL Server para probar mi aplicación como veremos en el presente manual; este archivo MDF que hemos creado también se podrá visualizar en el Explorador de Soluciones (no confundir Explorador de Soluciones, que muestra los archivos de nuestro proyecto, con el Explorador de Servidores, que muestra los diferentes Equipos de cómputo y de Datos que podemos utilizar).



Ahora, que ya tenemos nuestra Base de Datos, debemos crear nuestra tabla Usuarios. Sobre la carpeta Tablas del Explorador de Servidores haga clic con el botón derecho del mouse y seleccione la opción **Agregar Nueva Tabla**. Cree los campos *login_usu* y *pass_usu* para que quede como muestra la siguiente imagen. A decir verdad, esta pantalla es bastante intuitiva, de manera que no creo que tengan muchos problemas para crear la tabla. No se olviden de hacer clic en el botón **Establecer Clave Principal** en el campo login_usu, aunque Uds. Pueden considerar otros nombres de campos, esta es solo una recomendación.



Una vez hecho esto, haga clic en el botón Guardar (el clásico con forma de Disquete) y asigne a la tabla creada el nombre **Usuarios**.

Cierre la ventana de la tabla para a continuación crear un Procedimiento Almacenado que valide las entradas desde nuestro Sitio Web. Para hacerlo haga clic con el botón derecho del mouse sobre la carpeta **Procedimientos Almacenados**, y seleccione la opción *Agregar Nuevo Procedimiento Almacenado*. Se mostrará una ventana en la cual se puede escribir el SP (Stored Procedure). Una buena práctica consiste en no colocar el prefijo **sp** al Procedimiento almacenado, ya que estas siglas significan **System Procedure** y no Stored Procedure, como algunos pudieran pensar, de manera que le toma más tiempo al motor de base de datos ejecutarlo.

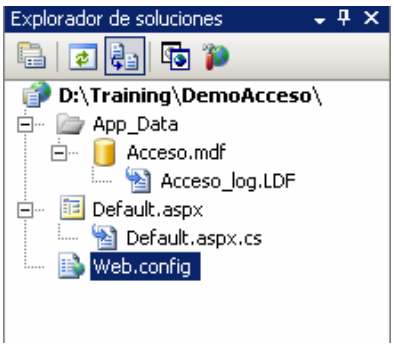
A continuación escriba el Siguiente código:

```
ALTER PROCEDURE dbo.usp_VerificarUsuario
(
	@user varchar(15),
	@pass varchar(15),
	@res bit OUTPUT
)
AS

	SET NOCOUNT ON

	if (select top 1 count(*) from usuarios
		where cast(login_usu as varbinary) = cast(@user as varbinary)
		and cast(pass_usu as varbinary) = cast(@pass as varbinary)) = 1
		set @res = 1
	else
		set @res = 0

RETURN 0
```

Hecho esto haga clic en el botón Guardar (si, ese de figura de Disquete). Como se darán cuenta, recibimos los parámetros @user y @pass, además tenemos un parámetro de salida de tipo bit que indicará si el usuario existe o no.

Una de las reglas de las buenas prácticas es que NUNCA se debe usar una instrucción SQL en la página Web, de manera que, cualquier instrucción SQL se debe hacer mediante Stored Procedures, de esa manera evitaremos ataques de SQL Injection en nuestras aplicaciones Web. Lo que ocurre con lo parámetros es que pasan a ser reconocidos por el SQL Server como elementos específicos y no se interpretarán como parte de la cadena de ejecución, en el caso anterior los parámetros @user y @pass además de esto se están convirtiendo al tipo de datos varBinary, de esa manera, el texto enviado no podrá afectar nuestra aplicación por un ataque de SQL Injection. Aunque esto parezca redundante, nunca esta de mas incrementar la seguridad de nuestras aplicaciones.

Ahora vamos a configurar nuestra aplicación Web para que le podamos dar una Autenticación por Formularios. Abra el Explorador de Soluciones y haga clic con el botón derecho del mouse sobre el primer elemento que se ve (Siempre se muestra la ruta del Sitio Web, en mi caso *D:\Training\DemoAcceso*) y haga clic en la opción **Agregar Nuevo Elemento**, en la ventana que se mostrará seleccione el icono **Archivo de Configuración Web**, por defecto el archivo tiene el nombre Web.Config, no cambie el nombre del archivo y haga clic en el botón **Agregar**. El Explorador de Soluciones se debe ver como se muestra en la imagen de la derecha.



El archivo Web.Config, es un archivo de Configuración para la aplicación que estamos creando y para cada carpeta. Pero debemos hacerle algunos cambios. Para empezar notará que existe una etiqueta igual a esta: `<connectionStrings/>`, esta etiqueta debe ser reemplazada por la siguiente instrucción:

```
<connectionStrings>
  <add name="DemoAcceso"
   providerName="System.Data.SqlClient"
   connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\Training\DemoAcceso\App_Data\Acceso.mdf;Integrated
Security=True;User Instance=True"/>
</connectionStrings>
```

Explicando: Dentro de las etiquetas **connectionStrings** se pueden configurar las cadenas de conexión que se usarán en la aplicación. Cada cadena se agregará con la etiqueta **add** que tiene los atributos **name** (que identificará a la cadena de conexión), **providerName** (que indicará el tipo de proveedor para esa cadena, por ej. Para sql, para oracle, etc.) y el atributo **connectionString** (que se utiliza para indicar la cadena de conexión).

Además de este cambio, debemos hacer lo siguiente, dentro del mismo archivo Web.Config se mostrará la siguiente etiqueta: `<authentication mode="Windows" />`, esta debe ser reemplazada por las siguientes líneas:

```
<authentication mode="Forms">
     <forms loginUrl="Login.aspx" defaultUrl="Sistema/Inicio.aspx" />
</authentication>
```

El elemento **authentication** permite indicar el tipo de autenticación para nuestra aplicación, en nuestro caso será la Autenticación por Formularios, aunque existen otras formas, pero no son motivo de este artículo.

Autenticación significa que debemos identificar, quién es el visitante. No debemos confundir con Autorización que se refiere a verificar cuales son los recursos que puede usar un visitante que ya ha sido autenticado.

También encontraremos lo siguiente:

```
<customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
       <error statusCode="403" redirect="NoAccess.htm" />
       <error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>
```

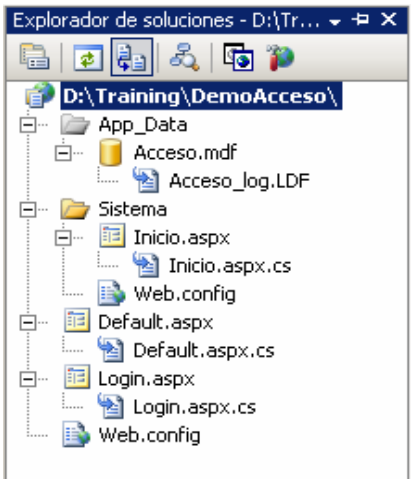
Esta en color verde, porque esta como comentario, lo mejor será quitarlo de allí y hacer que quede de la siguiente manera:

```
<customErrors mode="RemoteOnly" defaultRedirect="Error.htm">
<!--
       <error statusCode="403" redirect="NoAccess.htm" />
       <error statusCode="404" redirect="FileNotFound.htm" />
-->
</customErrors>
```

Con esto nos aseguramos que al producirse un error nos muestre automáticamente una página hecha por nosotros, la cual no debe contener muchos detalles sobre el error (de preferencia, ninguno). El atributo mode tiene

el valor RemoteOnly, lo que indica que solo se redireccionará a la página indicada si se trata de acceder a dicha página en forma remota, mientras que, de forma local, se mostrará el seguimiento completo del error, que puede servir, en la etapa de desarrollo.

Una característica que tiene ASP .NET es que cuando se produce un error no controlado, la página Web muestra el Seguimiento completo del error, indicando al visitante parte del código que generó el error y todo lo que hizo el .Net framework antes de que ocurra dicho error.



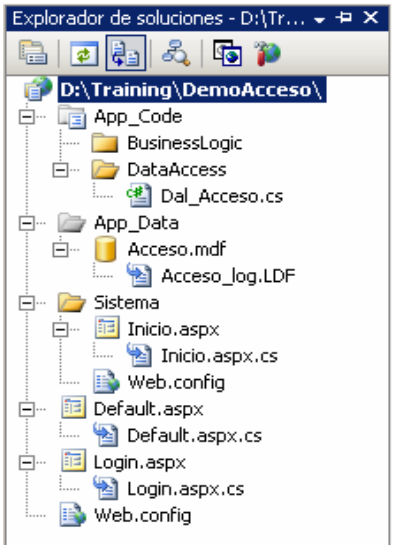
Esto, aunque parezca un detalle sin mucha importancia, es el motivo por el cual, muchas páginas Web de Empresas importantes han caído, porque se producían estos mensajes en la página Web y usuarios malintencionados con ciertos conocimientos técnicos usaban esta información para acceder a los servidores.

Ahora debemos crear una página Web llamada **Login.aspx** en la raiz de la aplicación; luego crearemos una nueva carpeta (no carpeta de ASP NET, sino una común y silvestre) llamada Sistema y agregaremos una página llamada **Inicio.aspx** y un Archivo de Configuración Web con el nombre por defecto Web.Config. El explorador de Soluciones debe quedar como se muestra en el gráfico de la derecha.

En el archivo Web.Config de la carpeta Sistema debemos escribir las siguientes instrucciones:

```
<configuration>
    <appSettings/>
    <connectionStrings/>
    <system.web>
      <authorization>
        <deny users="?"/>
      </authorization>
    </system.web>
</configuration>
```

Este archivo no es muy diferente del archivo principal, pero sirve, a diferencia del anterior para configurar la carpeta Sistema indicándole que todos los usuarios no autenticados no podrán ingresar a ver las páginas de esa carpeta.



Ahora, hemos llegado a la etapa de la creación de las Capas de Programación, para esto agregue la Carpeta de ASP NET llamada **App_Code**, esta carpeta se usa para escribir aquí las clases que conformarán lo lógica del sistema.

Dentro de esta carpeta crearemos las carpetas DataAccess (para las clases de acceso a Datos) y BusinessLogic (para las clases que contendrán la lógica de Negocios). Sobre la Carpeta **DataAccess** haga clic con el botón derecho del mouse y seleccione la opción **Agregar Nuevo Elemento** y seleccione el icono **clase**, a continuación escriba el nombre **Dal_Acceso.cs** (el prefijo Dal se refiere a Data Access Layer, o sea Capa de Acceso a Datos) para la nueva clase. Aunque en este proyectito no se utilizarán Clases que controlen la lógica del negocio, es bueno tenerla presente, ya que las validaciones y operaciones que no tienen que ver con la BD se deben hacer en clases que deben estar en esta carpeta.

El Explorador de Soluciones, esta vez, debe quedar como se muestra en la figura de la izquierda.

Aunque toda esta serie de pasos resultan, aparentemente muy engorrosos, resulta necesario pues forma parte de las **Buenas Prácticas** que hay que tener en cuenta en programación, no solo con .NET, sino también con cualquier otra herramienta o lenguaje Orientado a Objetos.

Ahora, en la clase Dal_Acceso.cs se debe escribir el siguiente código:

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
```

```
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

/// <summary>
/// Dal_Acceso: Se usa para verificar si un usuario existe o no en la Base de Datos
/// </summary>

public class Dal_Acceso
{
    private SqlConnection cnn;
    private Boolean autorizado = false;

    public Dal_Acceso()
    {
        cnn = new
SqlConnection(ConfigurationManager.ConnectionStrings["DemoAcceso"].ConnectionString);
    }

    public Boolean Autorizado
    {
        get
        {
            //throw new System.NotImplementedException();
            return autorizado;
        }
    }

    public void VerificarAcceso(String usuario, String password)
    {
        try
        {
            SqlCommand cmd = new SqlCommand("usp_VerificarUsuario", cnn);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add("@user", SqlDbType.NVarChar, 15);
            cmd.Parameters["@user"].Value = usuario;

            cmd.Parameters.Add("@pass", SqlDbType.NVarChar, 15);
            cmd.Parameters["@pass"].Value = password;

            cmd.Parameters.Add("@res", SqlDbType.Bit);
            cmd.Parameters["@res"].Direction = ParameterDirection.Output;

            cnn.Open();
            cmd.ExecuteNonQuery();
            autorizado = Convert.ToBoolean(cmd.Parameters["@res"].Value);
            cmd.Dispose();
            cnn.Close();
        }
        catch (Exception)
        {
            throw;
        }
    }
}
```

Nótese que le hemos agregado la referencia a System.Data.SqlClient que es donde se encuentran todos los objetos necesarios para acceder a las Bases de Datos de SQL Server.

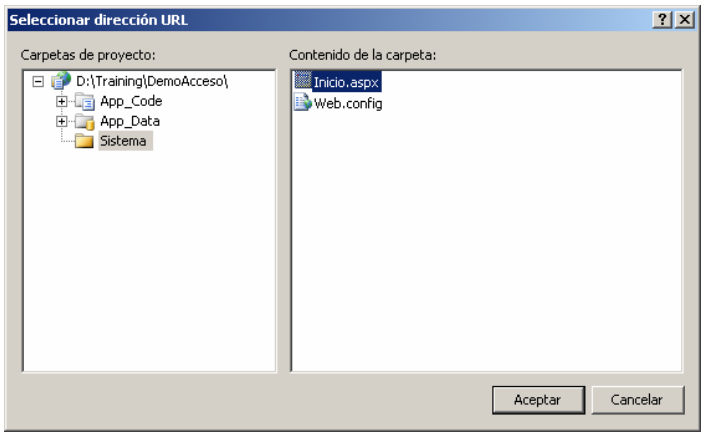
Como se puede observar tenemos un Constructor que instancia un objeto de tipo conexión. El **ConfigurationManager** administra las cadenas de conexión escritas en el Archivo de Configuración Web, de manera que solo le indicamos el nombre tal como lo escribimos en el Web.Config.

Esta clase tiene también la propiedad de solo lectura Autorizado de tipo lógico (verdadero o falso), que se usará para determinar si el usuario puede o no ingresar a la Aplicación.

Ahora, nótese que hemos utilizado la instrucción **try** para control de errores, en este caso lo hemos dejado en blanco, es decir, sin controlar el error como es debido ya que me di cuenta de este detalle al terminar este artículo y la verdad ya me dio pereza hacer algo al respecto, así que hablare sobre eso en otra oportunidad… je je je, ay! h@nz, que flojo!

El método VerficarAcceso se usará para recibir el nombre de usuario y contraseña que el usuario ha escrito en la página principal y hacer la comparación con la Base de Datos, llamando al Procedimiento Almacenado que ya hemos creado, mientras que la propiedad Autorizado retornará un valor lógico indicando si el usuario es o no válido. Ahora trabajaremos con la parte del cliente, que por cierto, después de todo esto, será mucho más fácil.

En la página **Default.aspx** agregaremos un control **HyperLink** y escribiremos en su propiedad Text el valor: *Visitar la página principal del Sistema*. Luego haga clic en el botón punteado que aparece en la propiedad NavigateURL para mostrar la ventana que se ve en el lado derecho.



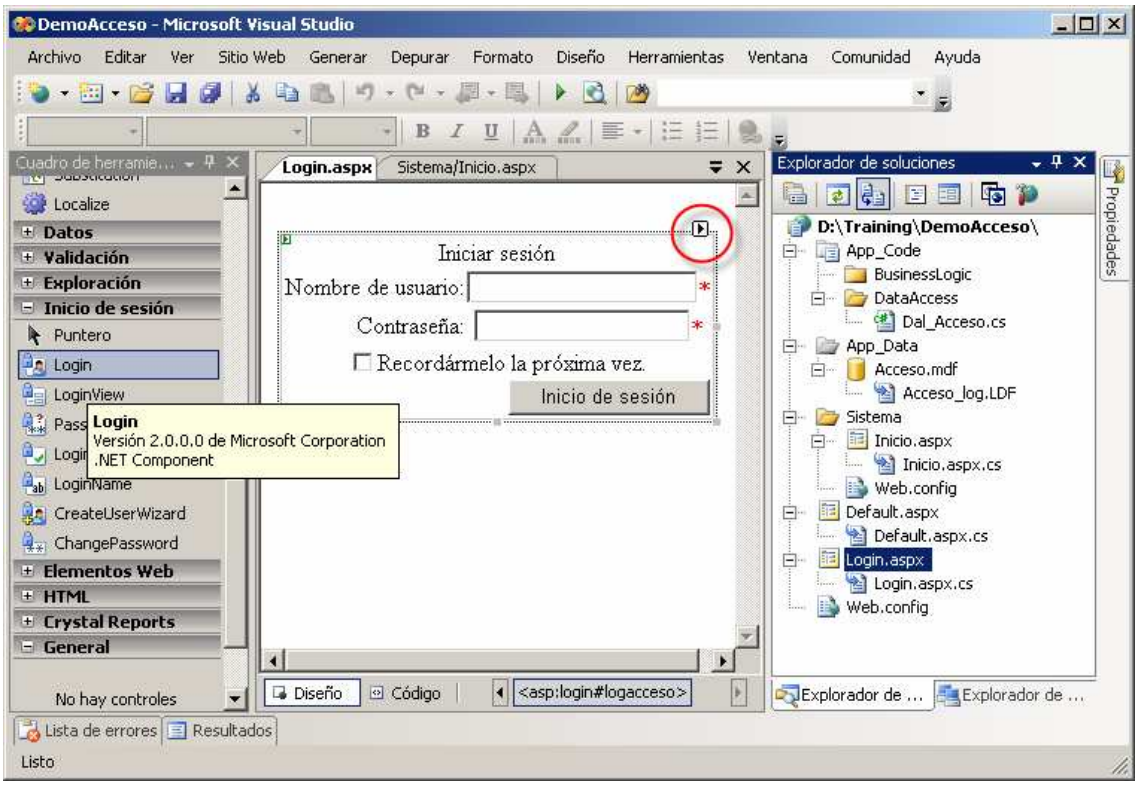
A continuación seleccione la carpeta Sistema en la sección izquierda y luego la página Inicio.aspx en la sección derecha de la ventana.

Una vez hecho esto se hará clic en el botón aceptar.

Con esto, ha quedado definido un enlace a través de este control HyperLink entre la página principal y la página de Inicio de nuestro sistema. Ahora haga lo mismo en la página Inicio.aspx, es decir, agregue un objeto HyperLink en la página Inicio.aspx que dirija a la página Default.aspx.

Hasta aquí todo quedaría muy bien, pero recordemos que la carpeta Sistema tiene un archivo Web.config en el cual se indica que solo podrán acceder a esta carpeta para visualizar las diferentes páginas aquellos usuarios que estén autenticados. De manera que si tratamos de ejecutar nuestra aplicación, al hacer clic en el vínculo para ir a la página Sistema/Inicio.aspx nos redireccionará a la página de Login.

Ahora solo necesitamos escribir algo más en la página de Login y habremos terminado.



En este gráfico se puede ver la página Login.aspx, en la cual se ha dibujado un control Login, este control esta disponible dentro de las categorías **Inicio de sesión** del Cuadro de herramientas. Si desea darle un formato predefinido puede hacer clic en el icono que se encuentra en la parte superior derecha del control, (en el gráfico se ha encerrado con un círculo rojo) y seleccionando la opción **Formato Automático.**

Cámbiele el nombre al control (con la propiedad ID de la ventana de propiedades) y asígnele el nombre **LogAcceso**. Hecho esto, haga doble clic sobre el control para escribir el poquísimo código que nos hace falta, bueno, es un decir.

En la ventana de código escriba lo siguiente:

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class Login : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (HttpContext.Current.User.Identity.IsAuthenticated)
        {
            Response.Redirect("Sistema/Inicio.aspx");
        }
    }

    protected void LogAcceso_Authenticate(object sender, AuthenticateEventArgs e)
    {
        Dal_Acceso x = new Dal_Acceso();
        x.VerificarAcceso(Server.HtmlEncode(LogAcceso.UserName),
Server.HtmlEncode(LogAcceso.Password));
        e.Authenticated = x.Autorizado;
    }

}
```

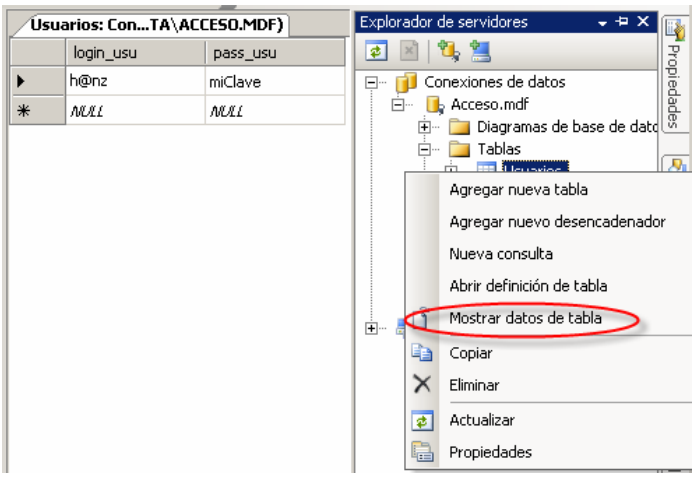
Ahora expliquemos esta última parte. *`HttpContext.Current.User.Identity.IsAuthenticated`* es una propiedad de solo lectura que devuelve un valor *true* si el usuario actual esta autenticado y devuelve *false* si el usuario no esta autenticado. De manera que si al cargar la página, el usuario esta autenticado, se redireccionará a la página de Inicio del sistema.

El Evento `LogAcceso_Authenticate` corresponde al evento de autenticación del objeto LogAcceso que habíamos creado. En este evento, se instancia un objeto de la clase Dal_Acceso (que ya hemos creado y que contiene la funcionalidad para la verificación con la base de datos) con la siguiente línea: `Dal_Acceso x = new Dal_Acceso();`

La siguiente línea invoca a nuestro método VerificarAcceso que se ha escrito en la clase de Acceso a Datos. Ahora lo importante aquí es lo siguiente: `Server.HtmlEncode(LogAcceso.UserName)`

`LogAcceso.UserName` contiene el nombre de usuario que el visitante ha escrito en el control Login; pero ¿porque agregar `Server.HtmlEncode`? La respuesta es simple, esta instrucción permite codificar algunos caracteres HTML potencialmente peligrosos que hubiera en la cadena que el visitante ingresa, para darle un poco más de seguridad a nuestra aplicación entonces, agregamos esta línea y evitamos un posible ataque de Cross Site Scripting. Ejemplo: Si un visitante ingresa el signo **<**, con esta instrucción se enviará **<** en su lugar, de esa manera se evitan posibles ataques.

Lo único que nos falta para que nuestra aplicación funcione es ingresar datos de usuario a la tabla, para agregarlos, en el Explorador de Servidores haga clic con el botón derecho del mouse sobre la tabla que tenemos creada y seleccione la opción **Mostrar Datos de Tabla**. A continuación ingrese al menos un usuario. Ahora si, ya esta lista la aplicación para ser probada.



Espero que este artículo les ayude en algo a entender la programación en .Net y que se entienda el principio de buenas prácticas con la programación en Capas y algo de seguridad para defendernos de SQL Injection y Cross Site Scripting.

Saludos…

PD: *Quería hacer algo pequeño y me salieron 7 páginas… plop!*
Otro PD: *El código fuente de este artículo también se adjunta, asi que pueden descargarlo para que se ahorren el trabajo de escribir todo el código, aunque recomiendo que lo escriban todo, así se entenderá mejor.*