

Seguridad en ASP.NET

Indice

13. Seguridad en ASP.NET.....	1
Determinar los requisitos de seguridad.....	1
1. Modelo de seguridad de ASP.NET.....	1
1.1 Autenticación y autorización.....	4
2. Autenticación con formularios.....	4
2.1 Configurar Web.config.....	6
2.2 Reglas de autorización.....	6
2.3 Configuración de acceso mediante WAT.....	9
2.4 La página de "login" o inicio de sesión.....	12
3. Autenticación Windows.....	16
3.1 Configurar el Web.....	19
4. Miembros.....	21
4.1 Almacenamiento de datos de los MemberShip.....	22
5. Controles de seguridad.....	35
5.1 Control de login.....	36
5.2 Permitir a los usuarios crear una cuenta.....	41
5.3 Control para recuperación de contraseña.....	45
5.4 Cambiar contraseña.....	49
6. Seguridad basada en roles.....	49
6.1 Restricciones de acceso basado en roles.....	52
6.2 Permisos en menús.....	55
7. Control LoginView.....	57
7.1 Información de roles.....	58
Ejercicios.....	61
Ejercicio 1.....	61
Ejercicio 2.....	62
Ejercicio 3.....	63
Ejercicio 4.....	66

13. Seguridad en ASP.NET

Habitualmente nuestra aplicación web estará accesible para todos los usuarios. Pero no siempre será así, por ejemplo en una aplicación web que sea una tienda o de comercio electrónico necesitaremos estudiar aplicar conceptos de seguridad. ASP.NET proporciona un amplio modelo de seguridad para hacer de una forma sencilla la labor de hacer seguro nuestro web. Disponemos de **varios niveles** de seguridad, en este tema veremos los mas utilizados que es la **autenticación con usuarios y la basada en Windows.**

El primer caso idóneo para aplicaciones Web en Internet y redes pequeñas (que no dispongan de un directorio activo) y el segundo caso perfecto para las Intranets de las empresas. El último sistema, y que no veremos en este curso sería la creación de web seguros utilizando encriptación de datos mediante SSL y que podéis ver en el curso de ASP Avanzado.

Determinar los requisitos de seguridad

El primer paso que tenemos que hacer es decidir el nivel de seguridad que queremos para nuestra web. Por ejemplo, podemos dejar una parte pública y otra privada que se identifique con una pantalla de inicio de sesión o podemos querer todo el web basado en autenticación de Windows. No es un tema complejo pero si debemos tener claro en el diseño de nuestra web cómo vamos a querer este tema.

Evidentemente el primer paso en nuestro web podría ser una pantalla de inicio de sesión para forzar a identificarse al usuario y es uno de los casos que haremos enseguida pero no siempre se basará en esto. Puede que necesitemos sistemas adicionales como encriptación de datos. Además la forma de movernos por la web es fundamental para preservar la seguridad.

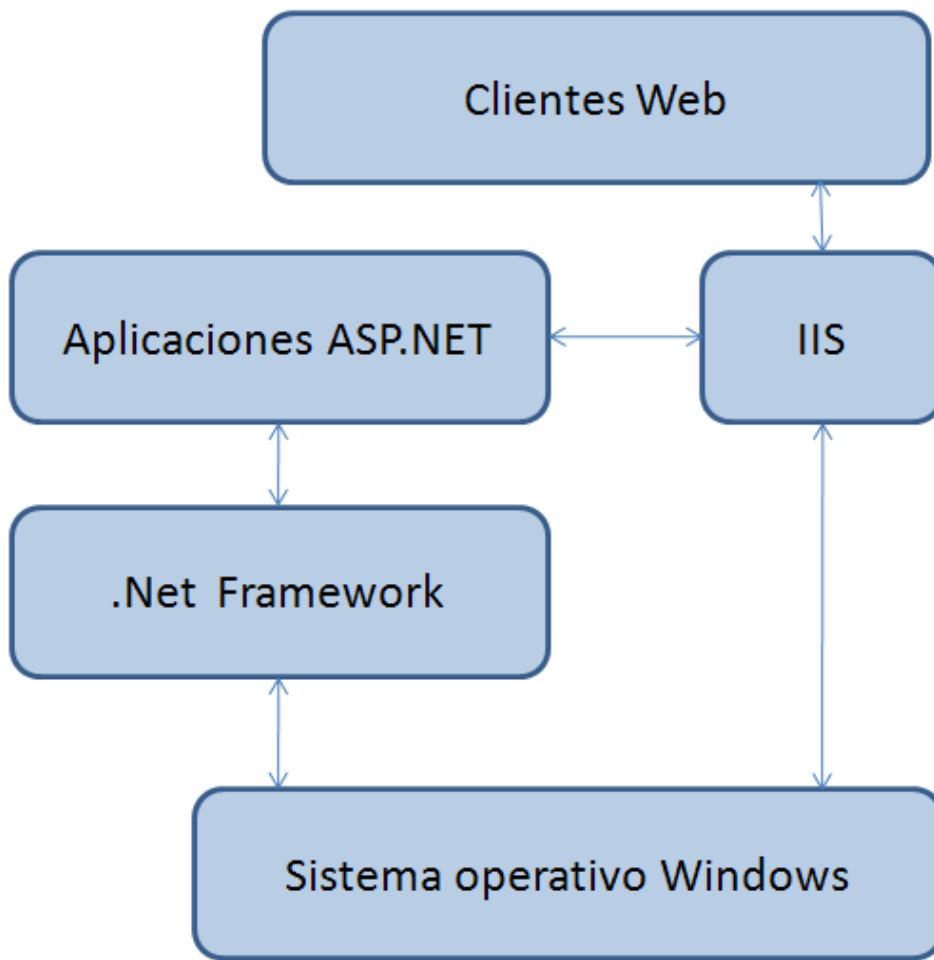
Imagina que hacemos una pantalla de inicio de sesión muy segura, el usuario se valida y va a su página personalizada de esta forma:

```
http://localhost/tienda/ver_pedidos.aspx?usuario=21234
```

Donde acabamos de descubrir la forma de entrar ya que inocentemente le hemos puesto el identificador en la dirección URL. Este mismo usuario podría intentar poner números aleatorios para ver si alguno es de otro cliente. Como ves es un tema que hay que tener en cuenta, aunque la noticia buena es que es un tema sencillo ya que ASP.NET tiene un modelo de seguridad muy completo e interesante.

1. Modelo de seguridad de ASP.NET

Según hemos aprendido en este curso, sabemos que las peticiones de páginas web las recoge el servidor de páginas WEB IIS. Éste examina la extensión del fichero y si es de ASP.NET le pasa la petición al motor de ASP.NET para que lo procese.

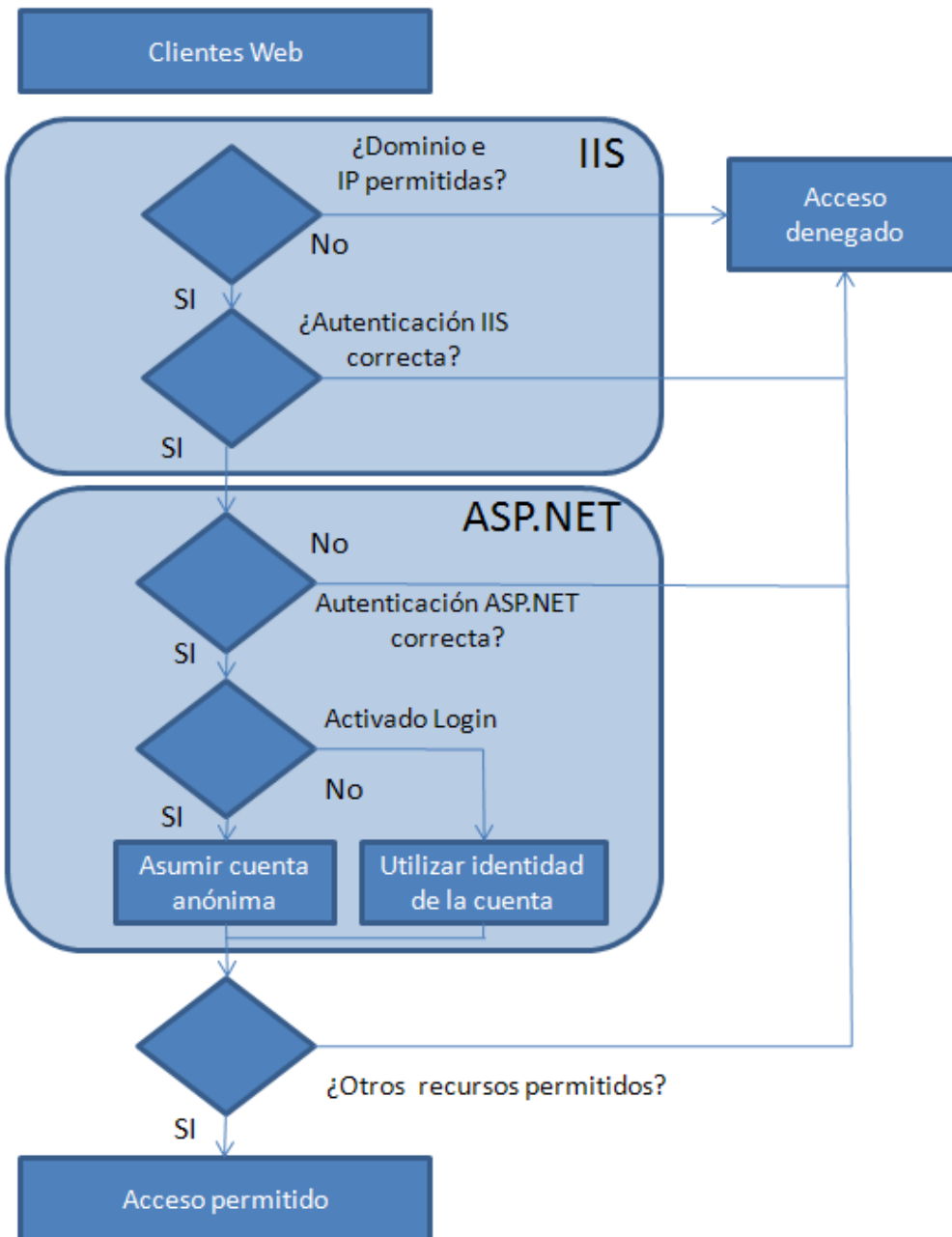


Podemos aplicar la seguridad en varios sitios de este esquema. Primero consideremos una petición de una página HTML normal:

1. IIS intenta autenticar (o autentificar) al usuario. Normalmente IIS permite solicitudes de todos los usuarios anónimos bajo una cuenta especial llamada "IUSR_ordenador" (revisa los capítulos 1 y 2). En Windows Vista esta cuenta se llama "IUSR"
2. SI IIS autentica correctamente al usuario, intentan mandar al usuario la página HTML. El sistema operativo realiza su propia comprobación de seguridad para verificar que el usuario tiene permisos para acceder a la carpeta y fichero

ASP.NET requiere varios pasos adicionales, fíjate en este esquema:

Seguridad en ASP.NET



1. El servidor de páginas Web IIS intenta autenticar al usuario. Normalmente IIS permite las peticiones de todos los usuarios anónimos (todos los de internet son anónimos en principio) y se identifican con la cuenta IUSR.
2. Si el usuario está autenticado correctamente pasa ahora a la autenticación de ASP.NET si es que tiene algún tipo de autenticación activada. ASP.NET puede utilizar sus propios servicios de seguridad, dependiendo del fichero "web.config" y de la página solicitada.
3. SI ASP.NET autentica al usuario permite las peticiones de páginas ASPX. El código puede realizar información adicional de seguridad como por ejemplo un segundo inicio de sesión para acceder a alguna página privada.
4. Cuando el código de ASP.NET solicite recursos, por ejemplo abrir un fichero de texto, el sistema operativo realiza su propio chequeo de seguridad. En un sitio Web ASP.NET se ejecuta bajo una cuenta de usuario del sistema operativo. Esta cuenta está definida en el fichero "machine.config" si

Seguridad en ASP.NET

estás ejecutando IIS5 (windows 2000) o en el administrador de IIS si estás ejecutando IIS6 ó IIS7 (Windows 2003, 2008, Vista).

La cuenta de usuario de IIS, la llamada IUSR que puedes ver en la administración de usuarios de tu equipo, no es la que utiliza ASP.NET para ejecutar las páginas. Esta cuenta IUSR no tiene los permisos suficientes para poder ejecutar el código ASP.NET, ya que éste necesita crear archivos temporales y otros procesos en la compilación de las aplicaciones web.

1.1 Autenticación y autorización

Estos dos conceptos son distintos y significan lo siguiente:

- Autenticación. Es el proceso de determinar la identidad del usuario y forzar al usuario a proporcionarla. Es lo que habitualmente realizaremos mediante un usuario y una contraseña.. Estos datos luego se contrastan contra una base de datos o un servidor de usuarios de Windows
- Autorización. Una vez que el usuario se ha autenticado la autorización es el proceso de proporcionarle permisos para realizar alguna acción. Por ejemplo tendrá autorización para consultar datos pero no para modificarlos.

Como ves son cosas distintas. Una cosa es darle permiso a alguien para entrar a un sitio y otra para ver que acciones puede hacer dentro de él. Lo mas parecido lo tenemos con nuestro usuario de red: primero nos identificamos en el ordenador para ver si podemos entrar. Y luego tendremos permisos para determinados recursos de red: unos con acceso, otros sin acceso, otros de lectura, ...

En nuestra aplicación Web vamos a utilizar dos tipos de autenticación:

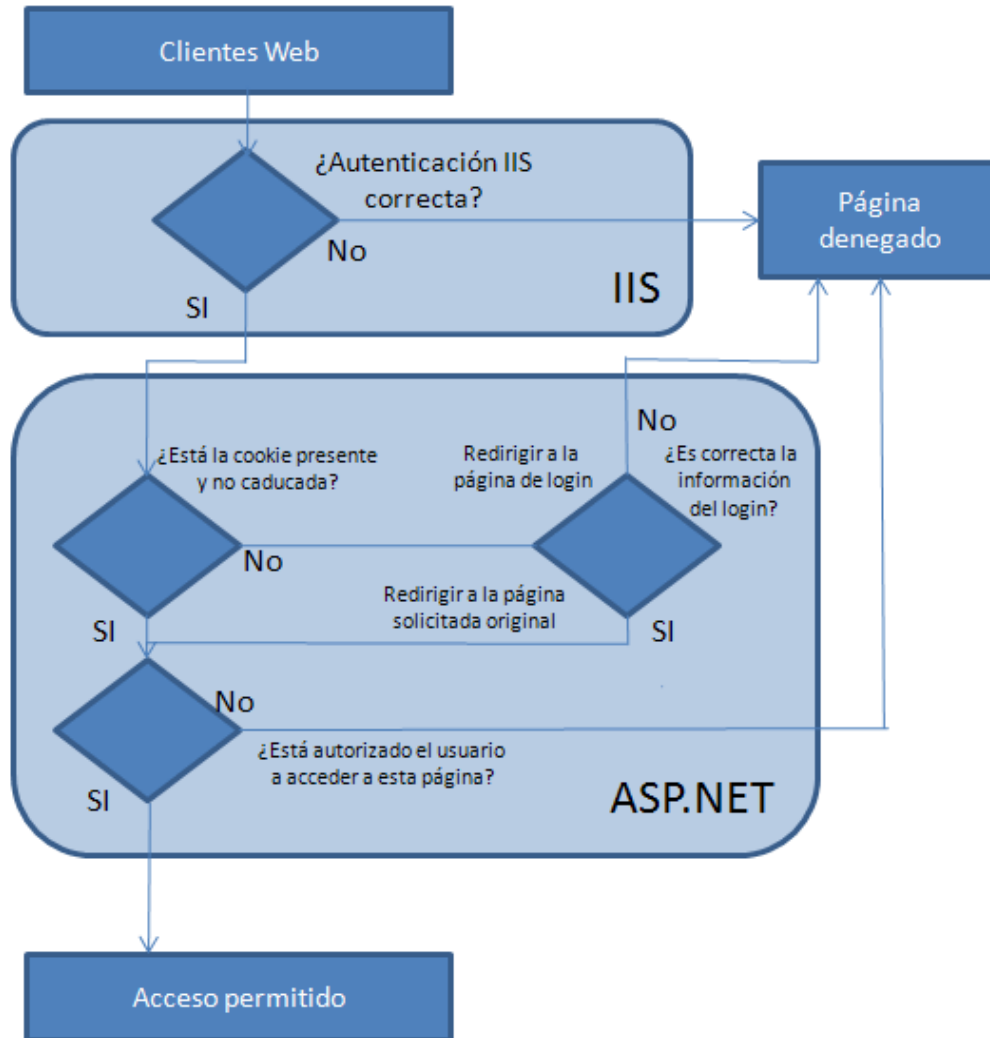
1. Autenticación mediante formularios. Por defecto IIS permite el acceso anónimo a las páginas, es decir todo el mundo puede ver todas las páginas. Con esta forma de autenticarnos lo que vamos a hacer es solicitar una autenticación para acceder a determinadas zonas que protegeremos.
2. Autenticación de Windows. Si en vuestra red tenéis un directorio activo para la administración de los recursos de la empresa entonces tenéis la posibilidad de utilizar la autenticación de Windows. La idea es que si ya tenemos una base de datos de usuarios, como es el directorio activo, sabemos los datos del usuario que ha hecho "login" en el ordenador y por tanto podemos evitar el proceso de pedirle usuario y contraseña porque utilizaremos el que puso al entrar al ordenador. Esta autenticación no vale para una aplicación web que esté publicada al exterior pero en cambio es perfecta para nuestra intranet.

2. Autenticación con formularios

Como hemos dicho, esta va a ser la forma de identificarnos si nuestra aplicación web va a estar situada en Internet o estamos en una red en la que no hay un directorio activo de Windows funcionando. Por tanto vamos a crear nosotros todo el sistema de seguridad. Veamos una idea de la autenticación... imagina que el usuario ha hecho login en una página, en ese momento guardamos una "cookie" con su sesión. Cada vez que se visite una página buscamos esa cookie, si no está quiere decir que alguien está accediendo directamente, sin haber pasado por la pantalla de inicio de sesión que es la que crea la "cookie" cuando todo ha ido correcto en la autenticación.

Seguridad en ASP.NET

Es una idea muy simple pero muy efectiva. ASP.NET nos va a ayudar de varias formas, para empezar los controles necesarios para hacer login nos lo va a poner automáticamente y además nos va a realizar un seguimiento de la autenticación del usuario, así que no debemos preocuparnos de crear, mantener y consultar nuestra cookie. Además ASP.NET nos proporciona algoritmos muy complejos para la validación de usuarios que hace imposible la intrusión. El esquema de la autenticación es el siguiente:



Para implementar nuestra seguridad a nivel de formularios necesitaremos realizar tres pasos:

1. Establecer el modo de autenticación en el fichero web.config o mediante la administración Web que vimos capítulos atrás.
2. Restringir a los usuarios anónimos el acceso a una página o carpeta
3. Crear la página de inicio de sesión o "login"

Vamos con los tres pasos...

2.1 Configurar Web.config

Definiremos el tipo de seguridad mediante la etiqueta <authentication>. Si abrimos el fichero ahora mismo tiene en la parte de abajo:

```
<authentication mode="Windows"/>
<!--
    The <customErrors> section enables configuration
    of what to do if/when an unhandled error occurs
    during the execution of a request. Specifically,
    it enables developers to configure html error pages
    to be displayed in place of a error stack trace.

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

-->
```

Es decir la de Windows activada. Esto no es lo que queremos para este ejemplo así que fijándonos en estos posibles atributos:

Atributo	Descripción
name	Nombre de la cookie que se va a utilizar para la autenticación.
LoginUrl	Página de inicio de sesión, será la que redirigirá ASP.NET cuando no encuentre la cookie. Por defecto es "login.aspx"
Protection	Tipo de encriptación y validación utilizada para asegurar la cookie: todas, ninguna, encryption y validation. La validación asegura que la cookie no se modifica en el tránsito y la encriptación codifica su contenido. El valor predeterminado es "All"
Timeout	Tiempo en minutos de expiración de la cookie. ASP.NET refresca esta cookie cada vez que haya una interacción del usuario, por lo tanto a los 30 minutos (valor predeterminado) de inactividad la sesión expira
Path	Ruta de las cookies de la aplicación. El valor por defecto recomendado es /

Pondremos estos valores:

```
<authentication mode="Forms">
    <forms name="MiGalleta"
        loginUrl="~/login.aspx"
        protection="All"
        timeout="20" path="/" />
</authentication>
```

2.2 Reglas de autorización

Si hemos tenido la curiosidad de ver que pasaba con estos cambios, habrás visto que no ha pasado nada. Esto es porque a pesar de haber habilitado la autenticación por formularios en nuestra aplicación todavía no hemos

Seguridad en ASP.NET

restringido a los usuarios anónimos, por tanto no se necesita ninguna validación todavía.

Para controlar los accesos a nuestra aplicación web necesitaremos añadir reglas de control de acceso a la sección `<authorization>` en nuestro fichero `web.config`. Por ejemplo:

```
<authentication mode="Forms">
  <forms name="MiGalleta"
    loginUrl="~/login.aspx"
    protection="All"
    timeout="20" path="/" />
</authentication>
<authorization>
  <allow users="*" />
</authorization>
```

El asterisco (*) es un carácter comodín que significa "todos", por tanto está permitido el acceso a todo el mundo: los que se hayan autenticado y los que no. Es el comportamiento por defecto, así que esa instrucción sobraría. un cambio significativo sería:

```
<authorization>
  <deny users="?" />
</authorization>
```

El carácter especial "?" significa "todos los usuarios anónimos". Por tanto esa instrucción deniega el acceso a todos los usuarios anónimos. Todos los usuarios deben autenticarse y tener su "cookie". Si intentamos ejecutar nuestra aplicación web al no encontrar la cookie nos redirigirá al formulario de autenticación "login.aspx". Que como no la hemos creado todavía nos dará un error.

Ahora fíjate si ponemos lo siguiente:

```
<authorization>
  <allow users="*" />
  <deny users="?" />
</authorization>
```

Comienza a analizar los permisos y como se encuentra que la primera le dice que deja permisos a todo el mundo, no evalúa la siguiente y deja el acceso completo a todos los usuarios. Si hiciéramos lo contrario:

```
<authorization>
  <deny users="?" />
  <allow users="*" />
</authorization>
```

Denegará el acceso a los anónimos y permitirá el acceso a los demás. Lógicamente en el primer caso se mandará a la página de login y si se autentica dejará de ser anónimo y por tanto podrá pasar a la página.

Control del acceso en directorios específicos

Lo normal en un sitio web es que varios directorios tengan permisos distintos. Con ASP.NET esto es muy fácil porque podemos poner otro fichero `web.config` con los permisos que queramos en las carpetas que queramos controlar. Por ejemplo denegar el acceso a los anónimos a la carpeta de "informes":

```
<authorization>
  <deny users="?" />
</authorization>
```

Control del acceso a ficheros específicos

Normalmente nos basta con organizar los permisos en carpetas, es mas claro y fácil de mantener, ya que por ejemplo podríamos crear una carpeta llamada "privada" y aplicarle los permisos necesarios. A pesar de esta recomendación podemos poner los permisos que queramos a ficheros individuales añadiendo una etiqueta "<location>" a nuestro fichero de configuración.

Esta etiqueta se coloca fuera de <system.web></system.web> y se anida directamente in la base <configuration>, fíjate en el ejemplo:

```
<configuration>
  <system.web>
    ...
    <authentication mode="Forms">
      <forms loginUrl="~/login.aspx" />
    </authentication>
    <authorization>
      <allow users="*" />
      <deny users="?" />
    </authorization>
  </system.web>
  ...

  <location path="informes.aspx">
    <system.web>
      <authorization>
        <deny users="?" />
      </authorization>
    </system.web>
  </location>
  <location path="privado.aspx">
    <system.web>
      <authorization>
        <deny users="?" />
      </authorization>
    </system.web>
  </location>
</configuration>
```

Como ves, están permitidas todas las páginas excepto las dos que hemos puesto en esas dos secciones.

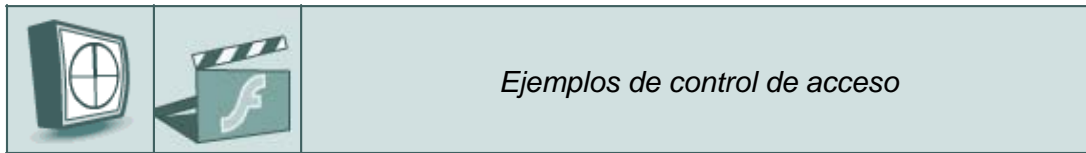
Controlar el acceso a usuarios específicos

Para otorgar o denegar permisos a algunas personas concretas podemos seguir ampliando la reglas de <allow> y <deny>. En este ejemplo no se permitirá el acceso a los no autenticados y a los usuarios: JoseM, AnaP y ManuelS, a pesar de que se hayan autenticado:

```
<authorization>
  <deny users="?"/>
  <deny users="JoseM, AnaP"/>
  <deny users="ManuelS"/>
  <allow users="*" />
</authorization>
```

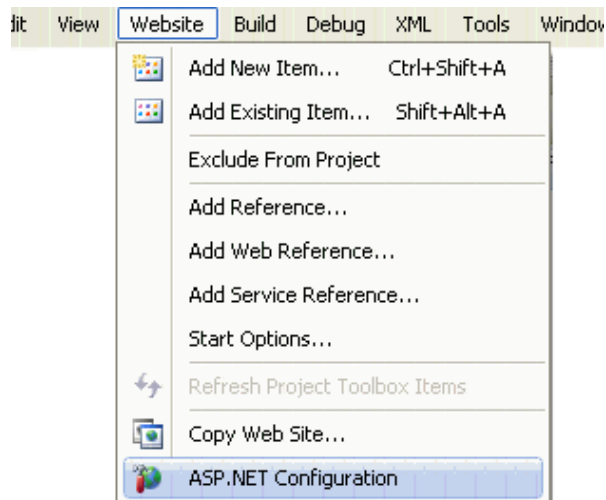
En este otro ejemplo sólo se permite el acceso a dos usuarios, todos los demás, autenticados o no tendrán permisos:

```
<authorization>
  <deny users="?"/>
  <allow users="JoseM, AnaP"/>
  <deny users="*" />
</authorization>
```



2.3 Configuración de acceso mediante WAT

WAT es el "website administration tool" que es la herramienta de administración del sitio web. Si recuerdas se accedía desde la opción de menú "Website" y luego la última opción de "ASP.NET Configuration".



Con esta herramienta podemos modificar y mantener todo el tema de los permisos, es decir, en lugar de mantener a mano el fichero de configuración podemos utilizar esta herramienta. Ejecútala y luego vete a la opción de seguridad. Si todavía tenemos el tipo de autenticación a "Windows" nos lo pondrá en la parte inferior izquierda:

Users

The current authentication type is **Windows**. User management from within this tool is therefore disabled.

[Select authentication type](#)

Podemos cambiar la autenticación pulsando en ese enlace para seleccionar el tipo de autenticación:

Home

Security

Application

Provider

How will users access your site?

☒ **From the internet**

Select this option if users will access your web site from the public internet. Users will be required to log on using a web form. The site will use forms authentication to identify users according to user information that you store in a database.

☐ **From a local network**

Select this option if users will access your web site only from a private local network. The site will use built-in Microsoft Windows authentication to identify users. Users with a valid Windows user name and password will be able to access your site.

Seleccionamos "From the internet" que es con la que estamos trabajando todo el tiempo, donde se basa la seguridad en nuestro formulario de login "login.aspx" y aceptamos. De vuelta a la pantalla anterior... si te fijas, en la parte derecha tenemos :

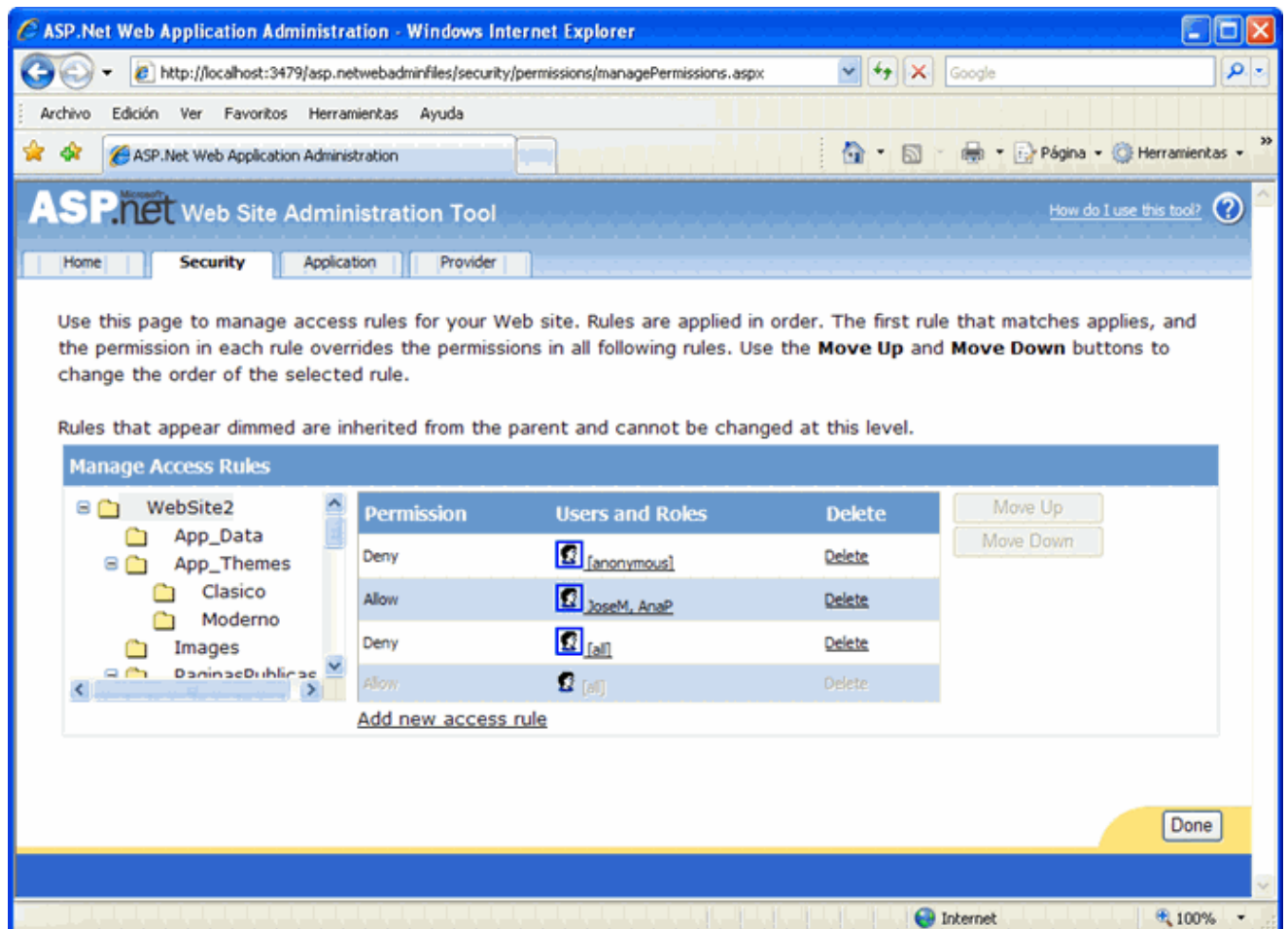
Access Rules

[Create access rules](#)

[Manage access rules](#)

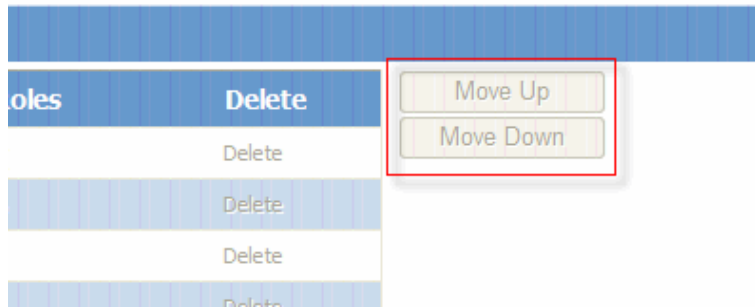
Donde podremos modificar las reglas de acceso que hemos estado poniendo antes. Por ejemplo:

Seguridad en ASP.NET

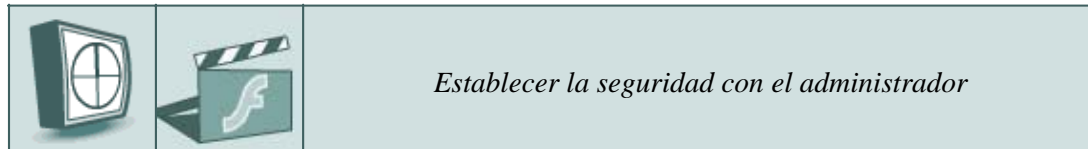


En la izquierda tenemos nuestras carpetas y a derecha los roles. Por ejemplo, selecciona una carpeta y dale a crear una regla de acceso:

Donde a la carpeta seleccionada le quitamos el acceso a "Antonio". Recuerda que los permisos se heredan, por tanto en una carpeta interior tendremos los del web general y luego los que le pongamos nosotros. Una vez puestas fíjate que tenemos unos botones para moverlas:



Y así establecer las propiedades que queramos.



2.4 La página de "login" o inicio de sesión

Una vez que hemos especificado el modo de autenticación y las reglas de acceso necesitaremos crear la página de acceso o de "login". que crearemos en una página .aspx estándar. ASP.NET proporciona una clase especial llamada **FormsAuthentication** en el espacio de nombres "System.Web.Security" que nos va a proporcionar todo lo necesario para estos procesos de login y autenticaciones.

Miembro	Descripción
FormsCookieName	Propiedad de solo lectura que proporciona el nombre de la cookie de autenticación del formulario
FormsCookiePath	Propiedad de solo lectura que proporciona la ruta de la cookie
Authenticate()	Comprueba si el usuario y contraseña existe en una lista de cuentas del archivo web.config
RedirectFromLoginPage()	Registra al usuario en la aplicación web creando la cookie, adjuntándola a la respuesta y redirigiendo al usuario a la página solicitada originalmente
SignOut()	Finaliza la sesión del usuario borrando la cookie
SetAuthCookie()	Registra al usuario en una aplicación ASP.NET adjuntando la cookie de formulario. Al contrario del método RedirectFromLoginPage() ese no devuelve al usuario la página de la solicitud
GetRedirectURL()	Proporciona la URL de la página solicitada originalmente. Se puede utilizar esto junto con SetAuthCookie() para registrar al usuario en la aplicación y tener la posibilidad en el código de redirigir a la página solicitada u otra.
GetAuthCookie()	Crea una cookie de autenticación pero no la adjunta a la actual respuesta. Se puede personalizar y añadirla posteriormente a la respuesta.

Seguridad en ASP.NET

HasPasswordForStoringInConfigFile()

Encripta una cadena utilizando los algoritmos (SHA1 o MD5). Este valor proporciona una forma segura de almacenar la contraseña en un fichero o base de datos.

Una página de inicio de sesión sencilla puede poner estos métodos para que funcione con poco código. Para intentarlo empezaremos habilitando la autenticación por formularios u denegando el acceso a los usuarios anónimos:

```
<authentication mode="Forms">
  <forms loginUrl="~/login.aspx" />
</authentication>
<authorization>
  <deny users="?" />
  <allow users="*" />
</authorization>
```

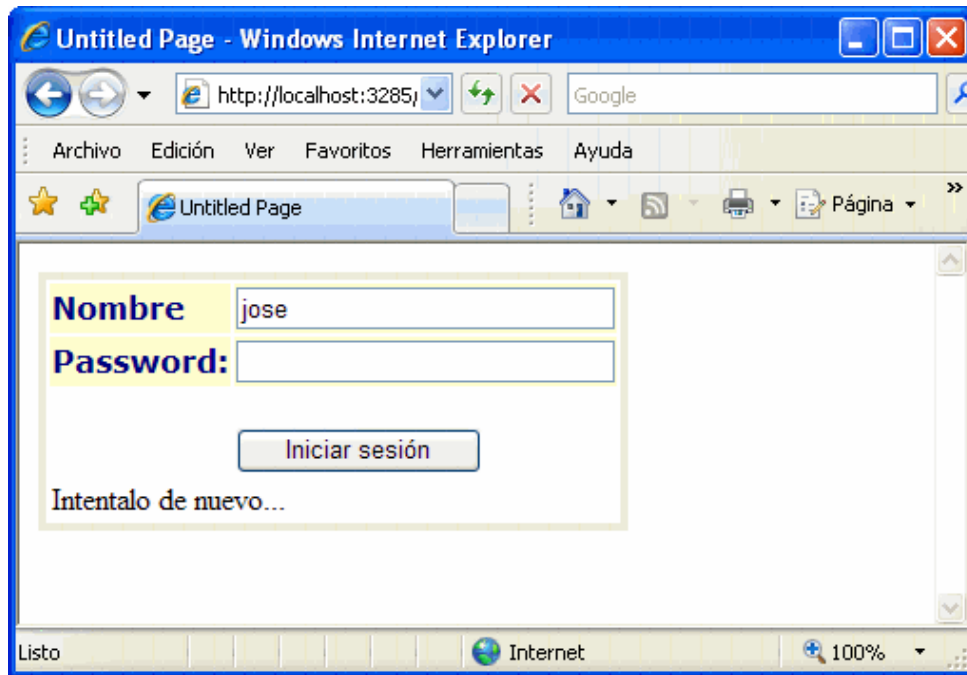
Ahora los usuarios serán redirigidos a una página sencilla de inicio de sesión. Por ejemplo:



Cuando el usuario haga clic en el botón de "Iniciar sesión" la página comprobará si el usuario ha tecleado la palabra "secreta" y utiliza el método RedirectFromLoginPage() para registrar al usuario. El código que estaría en el evento clic del botón sería:

```
Protected Sub btn_inicio_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    If txt_pass.Text.ToLower = "secreta" Then
        FormsAuthentication.RedirectFromLoginPage(txt_nombre.Text, False)
    Else
        lb_estado.text = "Intentalo de nuevo..."
    End If
End Sub
```

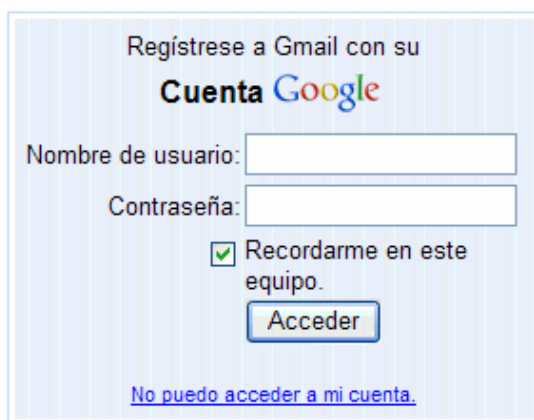
Si la validación va bien nos mandará a la página "default.aspx" que es la predeterminada de nuestro web. Si no es correcta la identificación volverá a pedirla:



Si no tienes una página llamada "default.aspx" crea una para que compruebes que si te identificas correctamente, poniendo cualquier nombre y de contraseña "secreta" podrás navegar a la página "default.aspx", de lo contrario volveremos al login.aspx

El método "RedirectFromLoginPage()" necesita dos parámetros, el primero es el nombre del usuario y el segundo una variable booleana para ver si se crea una cookie persistente o no. Una cookie persistente se crea en el equipo del usuario con una duración de 50 años, ¡mas que suficiente! Las nos persistentes acuérdate que tenían duración limitada. Puesto que es un inicio de sesión no queremos concederle el acceso para siempre por eso le hemos dicho que no sea persistente para que caduque y se vuelva a identificar cuando visite la página otra vez en una nueva sesión.

Cuando visitas sitios web donde te dice "Recordarme la próxima vez" en la identificación, por ejemplo en el Gmail:



O en Hotmail, lo que hace la página es que si seleccionamos la casilla de verificación lo que grabará será una "cookie persistente". Así sabe que ya estás identificado porque la cookie está creada y es válida y la próxima vez que entres al comprobarla no nos pedirá ya el inicio de sesión. Como ves estamos viendo el

Seguridad en ASP.NET

funcionamiento de todos los mecanismos de seguridad, y este en concreto, es muy curioso porque nos descubre lo "complicado" que es hacer que nos recuerde la próxima vez que entremos. Con esa casilla de verificación quedaría el segundo parámetro variable según haya chequeado o no:



Nombre

Password:

☐ Recordarme la próxima vez

Iniciar sesión

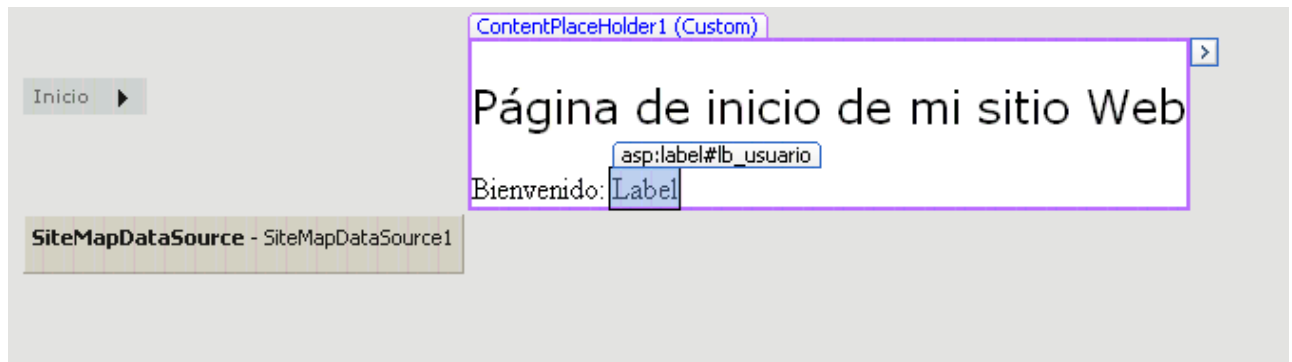
[lb_estado]

Y la línea de la validación:

```
FormsAuthentication.RedirectFromLoginPage(txt_nombre.Text, chk_persistente.Checked)
```

Recuperar la identidad del usuario.

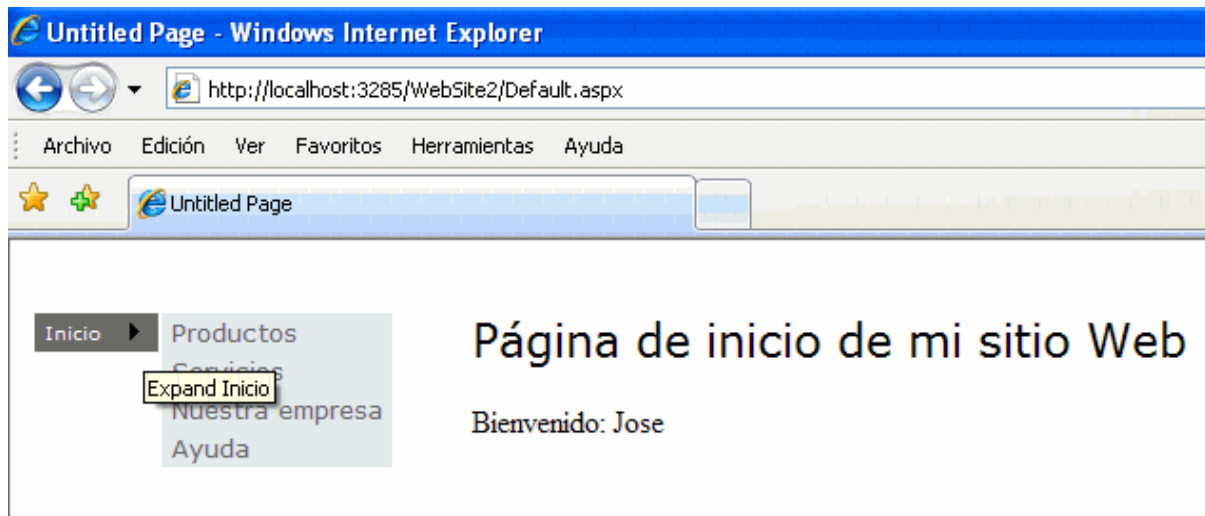
Una vez que se ha identificado correctamente tenemos "en memoria" el nombre del usuario que ha hecho login, vamos a ver como lo mostramos. Tenemos una página "default.aspx" que es la que se le muestra al usuario cuando ha hecho login, así que le pondremos en la parte superior el nombre de usuario, técnica habitual en las páginas con autenticación:



Es nuestra página default.aspx que está creada con páginas maestras. Le ponemos en el "placeholder" la bienvenida y el nombre del usuario que escribiremos en el Page_Load:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    lb_usuario.Text = User.Identity.Name
End Sub
```

Por tanto al validarnos si los credenciales son correcto iremos a la página default.aspx que nos escribirá el usuario:



Hemos utilizado el objeto "User" que contiene información sobre el usuario que ha hecho el inicio de sesión y entre sus propiedades tenemos la del nombre con "User.Identity.Name". No te preocupes, no estamos ante otro objeto "complejo" ya que solo tiene una propiedad y un método:

- La propiedad "Identity" nos permite obtener el nombre del usuario que ha hecho login y el tipo de autenticación que ha realizado.
- El método "IsInRole" nos proporciona información sobre los roles del usuario. Esto lo veremos un poco mas adelante.

Ya solo nos queda cerrar la sesión, es decir el proceso de "logout"

Cerrar la sesión

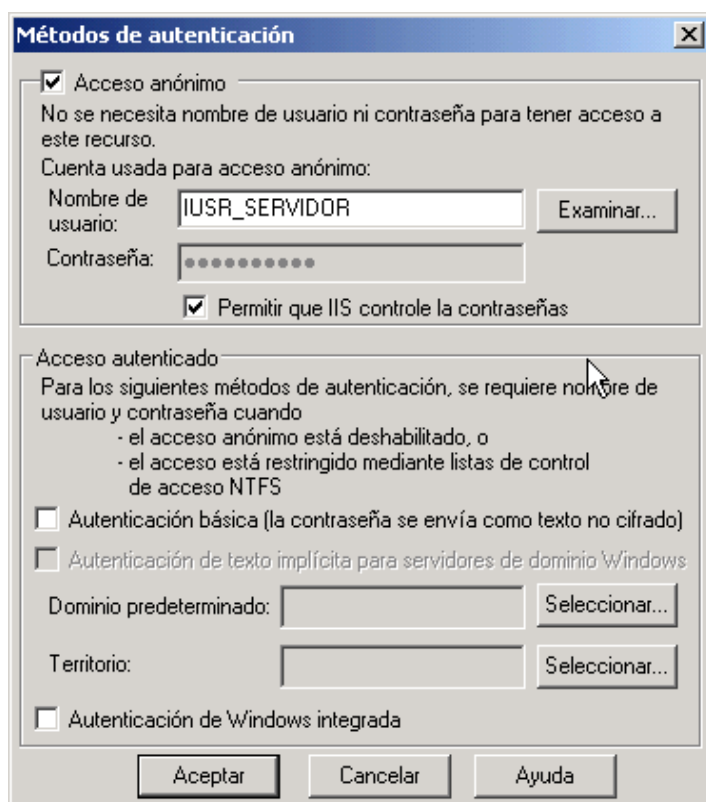
En todas las aplicaciones web en las que hay una validación de un usuario siempre hay un cierre de sesión. En este caso lo que hace ASP.NET es borrar la cookie, con lo que el usuario deja de estar presente y siempre que quiera obtener otra página pasará primero por la de login. Si ponemos un botón de cerrar sesión, el proceso será:

```
Protected Sub btn_cerrar_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    FormsAuthentication.SignOut()
    Response.Redirect("~/login.aspx")
End Sub
```

Está bastante claro ¿no? Destruimos su identificación con la cookie y le devolvemos a la página de inicio de sesión. Como siempre trato de hacer, primero aprendemos la teoría y luego lo aplicamos con controles que nos ayuden. En este caso tenemos unos controles que nos van a hacer de forma automática todo esto, pero los veremos un poco mas tarde. Ahora tenemos que tratar la otra forma de autenticación...

3. Autenticación Windows

Con este tipo de autenticación entramos en una configuración mas avanzada de nuestro servidor Web. Debes repasar los dos primeros capítulos donde estudiábamos a fondo nuestro servidor Web IIS. En una de las solapas teníamos la configuración del tipo de acceso:



Esta es la seguridad predeterminada, se permiten los acceso anónimos a nuestro web. En Internet todos los usuarios son anónimos así que es así como debe estar para utilizarse en Internet. Y si necesitamos seguridad crearemos entonces los formularios de login del capítulo anterior para proteger determinadas zonas. Como ves en la pantalla, cuando se utiliza el acceso anónimo IIS utiliza una cuenta genérica "IUSR_SERVIDOR" para acceder a los ficheros del disco duro, es decir a las páginas web, luego si miras a nivel del explorador de archivos los permisos de las carpetas donde están nuestras páginas verás que ese usuario tiene acceso a ellas.

Pero ahora estamos con otro tipo de autenticación, la de Windows. En este caso la identificación ya no es anónima, nadie que no se haya validado en un servidor Windows podrá visitar las páginas. Esto es lo habitual en las aplicaciones web de dentro de las empresas, es decir, en las Intranets.

Para activar este tipo de autenticación debemos hacer varias cosas:

1. Establecer este tipo de seguridad en el fichero web.config o mediante el administrador Web:

Seguridad en ASP.NET

The screenshot shows the 'Security' tab of the ASP.NET Security Configuration Wizard. The question 'How will users access your site?' is at the top. There are two radio button options: 'From the internet' (selected) and 'From a local network'. The 'From the internet' option is described as requiring users to log on using a web form with authentication against a database. The 'From a local network' option is described as using built-in Microsoft Windows authentication for users with valid Windows credentials.

2. Deshabilitar el acceso anónimo y poner solo en integrado de Windows:

The screenshot shows the 'Métodos de autenticación' (Authentication Methods) dialog box. The 'Acceso anónimo' (Anonymous) checkbox is unchecked. The 'Acceso autenticado' (Authenticated) section is expanded, showing that 'Autenticación de Windows integrada' (Integrated Windows Authentication) is checked and highlighted with a red rectangle. Other options like 'Autenticación básica' (Basic) and 'Autenticación de texto implícita' (Implicit) are unchecked. The dialog also includes fields for 'Nombre de usuario' and 'Contraseña' for anonymous access, and 'Dominio predeterminado' and 'Territorio' for authenticated access.

3. Configurar Windows creando las cuentas de usuario necesarias para que las utilicen los usuarios de nuestra red.

A partir de este momento sólo los usuarios de la red podrán acceder. Los usuarios se identificarán en el

ordenador desde la pantalla de inicio de sesión:



Luego, como el usuario ya se ha identificado, utilizaremos esas credenciales en nuestra página web. Nos evitamos el proceso de login porque ya sabemos que usuario ha sido el que se ha validado en ese ordenador. Por eso es tan útil en una Intranet. En éstas que son para las redes locales y de uso de la empresa, el usuario ya se ha identificado al entrar al ordenador por tanto ya sabemos quien es y no necesitamos volver a mostrarle una pantalla de login.

3.1 Configurar el Web

La configuración del web quedaría de esta forma:

```
<authentication mode="Windows">
</authentication>
<authorization>
  <deny users="?" />
</authorization>
```

Solo hay de momento una sola regla de autorización que es la de rechazar a todos los anónimos. Internet Explorer pasará los credenciales del usuario de Windows, por tanto para que funciones correctamente los clientes deben tener Microsoft Internet Explorer y deben estar registrados en un dominio de Windows.

También podemos añadir etiquetas como antes con "<allow>" y "<deny>" para restringir a usuarios a los directorios que queramos. Para indicar los usuarios debemos poner ahora el nombre completo de "dominio_windows\usuario":

```
<allow users="miempresa\Josem" />
```

Si es una cuenta local del servidor Web que está ejecutando IIS deberíamos poner ese servidor:

```
<allow users="ServidorWeb\JoseM_local" />
```

Seguridad en ASP.NET

o como atajo en este único caso de utilizar un usuario local del servidor IIS:

```
<allow users=".\\JoseM_local" />
```

Pero esto no se utiliza ya que usaremos siempre los usuarios de nuestro dominio y no los locales de los servidores. También podremos poner permisos a grupos de usuarios de Windows. Esto es especialmente útil. Imagina que en nuestra aplicación Web hay una carpeta llamada "Informes" a los que solo deben acceder los usuarios del departamento de administración que pertenecen al grupo de usuarios: "usuarios_Admon", el permiso sería:

```
<authorization>
```

```
<deny users="?" />
```

```
<allow roles="miempresa\\usuarios_admon, miempresa\\usuarios_informatica" />
```

```
<deny users="miempresa\\JesusP" />
```

```
</authorization>
```

En este interesante ejemplo estamos dando permisos a todos los usuarios del grupo del departamento de administración y del departamento de informática y denegando a "JesusP" siendo del departamento sea. Y además y por supuesto denegando a todos los anónimos.

Podemos comprobar si el usuario pertenece a un grupo de esta forma:

```
Private Sub Page_Load (...)  
  
If User.IsInRole ("miempresa\\usuarios_admon") then  
    'No hacemos nada ya que tiene permiso  
else  
    'No tiene acceso así que a la pantalla de login  
    Response.redirect ("~/login.aspx")  
end if  
  
End sub
```

En este punto entramos ya en la administración de Windows, donde encontraremos varios grupos de seguridad con los usuarios de nuestra red. Debes repasar esta parte de la administración de Windows 2003 Server para tener mas conceptos sobre las cuentas y grupos de usuarios predefinidos de Windows. En Teleformación tienes un curso extraordinario (curiosamente también realizado por mi) sobre Windows 2.003 Server que te enseñará todo lo necesario para montar y administrar una red basada en dominios.

Y para terminar esta sección veamos una página donde nos podrá toda la información del usuario que ha hecho login:

Seguridad en ASP.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim texto As New StringBuilder()
    texto.Append("Estás visitando una página segura, ")
    texto.Append(User.Identity.Name)

    Dim Identidad_Windows As WindowsIdentity
    Identidad_Windows = CType(User.Identity, WindowsIdentity)
    texto.Append("<br /><br />Tipo de autenticación: ")
    texto.Append(Identidad_Windows.AuthenticationType)
    texto.Append("<br />Anónimo: ")
    texto.Append(Identidad_Windows.IsAnonymous)
    texto.Append("<br />Autenticado: ")
    texto.Append(Identidad_Windows.IsAuthenticated)
    texto.Append("<br />Invitado: ")
    texto.Append(Identidad_Windows.IsGuest)
    texto.Append("<br />Sistema: ")
    texto.Append(Identidad_Windows.IsSystem)
    texto.Append("<br />Administrador: ")
    texto.Append(User.IsInRole("BUILTIN\Administrators"))
    lb_info.Text = texto.ToString()
End Sub
```

Sabiendo el usuario que ha hecho login y según los "allow" y "deny" podemos dar permisos a los usuarios de nuestra Intranet.

4. Miembros

Hemos aprendido a asegurar nuestro sitio web mediante las configuraciones del control de acceso bien configurado por formularios de autenticación o por los credenciales de un usuarios de Windows. Ahora debemos diseñar la lógica de uso de nuestro web ya que hay partes de acceso restringido y otras partes con acceso anónimo. Para organizar todo esto ASP.NET implementa otra serie de elementos bajo el nombre de "Membership", que como no se traduce pues lo seguiremos llamando así. Pero hace referencia a usuarios o "miembros"

La clase **Membership** proporciona los medios para:

- **Crear nuevos usuarios.**
- **Almacenar la información de suscripción** (nombres de usuario, contraseñas, direcciones de correo electrónico y datos compatibles) en Microsoft SQL Server o en un almacén de datos alternativo.
- **Autenticar a los usuarios que visitan su sitio.** Mediante programación podemos autenticar a los usuarios o podemos utilizar el control Login para crear un sistema de autenticación completo que requiere poco o ningún código.
- **Administrar contraseñas** que incluyen su creación, cambio, recuperación y restablecimiento, etc. Opcionalmente puede configurar la suscripción a ASP.NET para que requiera una pregunta y una respuesta de contraseña para autenticar las peticiones de restablecimiento o recuperación de la contraseña para aquellos usuarios que la hayan olvidado

Es decir es igual que cuando estamos visitando un sitio web profesional donde nos podemos dar de alta, mantener nuestras contraseñas e incluso decirle que nos la recuerde si la hemos olvidado. Es el complemento perfecto al tema de los usuarios anterior y funcionarán de forma conjunta. Vemos mas detalles de estos "membership":

Seguridad en ASP.NET

- Tiene su propia administración de los registros. Podemos crear los usuarios y mantenerlo en una base de datos, normalmente SQL Server.
- Proporciona controles de seguridad. Podemos poner los controles de login avanzados que incorpora ASP.NET para todas las operaciones de los usuarios
- Seguridad basada en "roles" (mantenemos también la palabra sin traducción). Ya se nos ha presentado la necesidad de dar permisos a unas personas a una carpeta de nuestro web y otros permisos a otros, ahora lo manejaremos todo con los "roles" que nos simplificarán mucho esta labor agrupando los usuarios.

4.1 Almacenamiento de datos de los MemberShip

La clave de todo este pequeño grupo de herramientas está en la capacidad de ASP.NET para almacenar los credenciales de los usuarios en una base de datos. Definiremos una serie de valores que queremos almacenar sobre los usuarios y ASP.NET se encargará de administrar y controlar esa base de datos. Podremos reducir mucho el código de nuestra aplicación Web porque ASP.NET se va a encargar absolutamente de todo referente a los usuarios, evitando además errores en estas páginas tan importantes como es la gestión de usuarios y permisos.

El primer paso va a ser el de crear la base de datos en nuestro SQL Server. Si estamos utilizando la versión Express lo tenemos muy fácil porque ya está incorporado todo: por defecto "membership" está habilitado en cada nuevo sitios web que creemos. Y por defecto asume que:

- Queremos almacenar la base de datos de Membership en un SQL Server 2005 Express
- Tenemos instalado y funcionando un SQL Server 2005 Express, con el nombre de instancia "SQLEXPRESS". Recuerda cuando creábamos la conexión a la base de datos.
- Los datos de Membership se almacenan en un fichero "aspnetdb.mdf" que se guarda en la carpeta "App_Data" de nuestra aplicación web. Por eso hay uno distinto para cada web.

Las bases de datos pues están "adosadas" a cada web y no incluidas dentro de SQL Server, de esta forma es mucho mas fácil desplegar nuestra aplicación Web. Sobre todo si en el servidor final estamos utilizando SQL Server Express ya que no necesitaremos ningún cambio. No tendremos que instalar la base de datos, basta con tenerla en esa carpeta especial "App_Data".

Vamos a probar todo esto con un ejemplo muy sencillo. Crea una página web y le añades un control de tipo "CreateUserWizard" y ejecuta sin añadir nada mas:

Seguridad en ASP.NET

The screenshot shows the 'asp:createuserwizard#CreateUserWiz...' page. The main form is titled 'Sign Up for Your New Account' and contains the following fields, each with a red asterisk indicating a required field:

- User Name:
- Password:
- Confirm Password:
- E-mail:
- Security Question:
- Security Answer:

Below the fields, a red error message states: 'The Password and Confirmation Password must match.' At the bottom of the form is a 'Create User' button.

On the right side, there is a 'CreateUserWizard Tasks' panel with the following options:

- Auto Format...
- Step: Sign Up for Your New (dropdown)
- Add/Remove WizardSteps...
- Convert to StartNavigationTemplate
- Convert to StepNavigationTemplate
- Convert to FinishNavigationTemplate
- Convert to CustomNavigationTemplate
- Customize Create User Step
- Customize Complete Step
- Administer Website
- Edit Templates

Recuerda dejar acceso a todo el mundo anónimo para seguir con las pruebas. Vamos a la página:

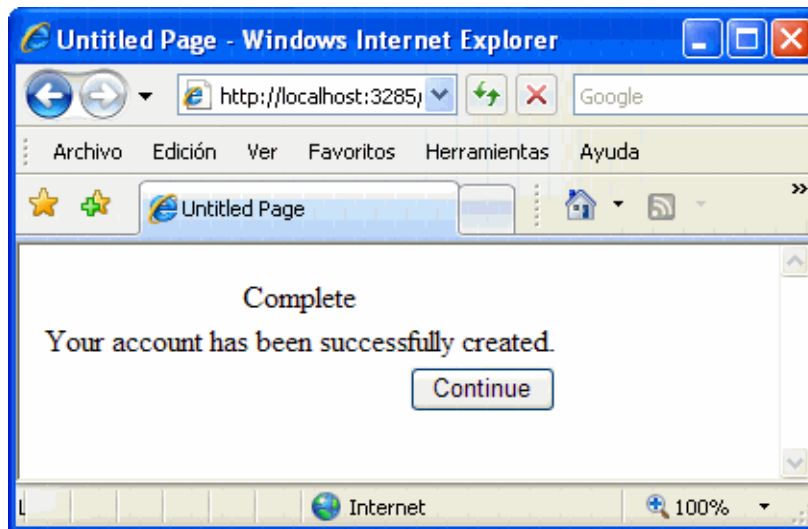
The screenshot shows the 'Sign Up for Your New Account' form displayed in a Windows Internet Explorer browser window. The form is titled 'Sign Up for Your New Account' and contains the following fields:

- User Name:
- Password:
- Confirm Password:
- E-mail:
- Security Question:
- Security Answer:

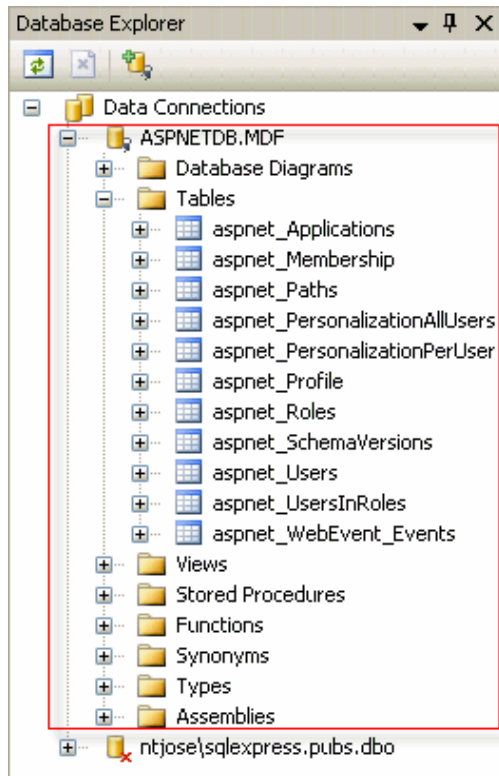
At the bottom of the form is a 'Create User' button.

E introduce los datos para crear un usuario, la contraseña debe ser compleja así que pon por ejemplo letras mayúsculas y minúsculas, de longitud mínima de 7 caracteres y al menos con un carácter que no sea ni letras ni números, por ejemplo un signo de admiración: "password!"

Seguridad en ASP.NET



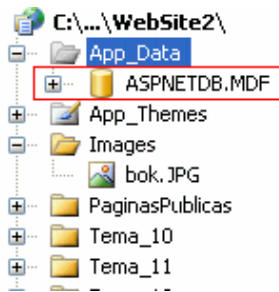
Si todo es correcto nos habrá creado el usuario, pero lo mas curioso del todo lo tienes en el IDE, vete ahora a la parte de las bases de datos, el llamado explorador de base de datos que ya conoces:



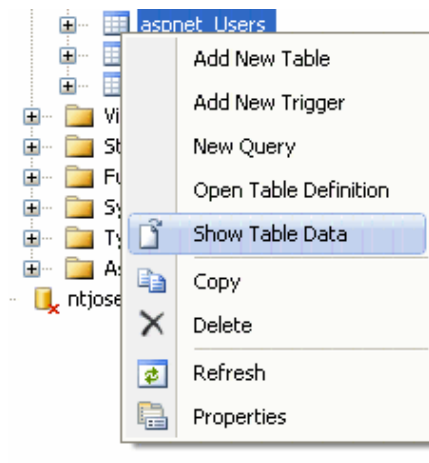
Ha aparecido una nueva base de datos en nuestro proyecto, sin hacer nada y si exploras las tablas tiene una de usuarios "aspnet_users" entre otras. O sea, que por el hecho de poner un control para crear usuarios ASP.NET me ha enlazado mi aplicación web con una base de datos nueva que ha creado para almacenar todo lo relacionado con los usuarios, el llamado "membership".

Además si exploras los ficheros, efectivamente dentro de App_Data ha metido este fichero de base de datos de SQL Server 2005 Express:

Seguridad en ASP.NET



Ahora en la pantalla del explorador de base de datos, selecciona la tabla de "aspnet_users" para con el botón derecho explorar su contenido:



Donde debería estar nuestro usuario:

	ApplicationId	UserId	UserName	LoweredUserN...	MobileAlias	IsAnonymous	LastActivity
▶	11c-3d3d1b207684	d32b50f2-e49e-...	Jose	jose	NULL	False	14/02/2008
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Los campos de "ApplicationId" y "UserId" son de un tipo de datos especial que se genera de forma aleatoria y es único. Si con el botón derecho en la tabla seleccionamos "Open Table Definition" podemos ver la definición de los campos de la tabla:

	Column Name	Data Type	Allow Nulls
▶	ApplicationId	uniqueidentifier	<input type="checkbox"/>
🔑	UserId	uniqueidentifier	<input type="checkbox"/>
	UserName	nvarchar(256)	<input type="checkbox"/>
	LoweredUserName	nvarchar(256)	<input type="checkbox"/>
	MobileAlias	nvarchar(16)	<input checked="" type="checkbox"/>
	IsAnonymous	bit	<input type="checkbox"/>
	LastActivityDate	datetime	<input type="checkbox"/>
			<input type="checkbox"/>

Ahí puedes ver tipo de datos de estos dos campos, la idea es meter un valor único para cada usuario que haga de clave, para eso tiene este tipo de datos SQL Server.

Configurar el proveedor "membership"

Por defecto la gestión de los usuarios del "membership" puede ser suficiente pero es posible que queramos modificar algún parámetro de su funcionamiento, por ejemplo la validación de las contraseñas.

Hemos visto a la hora de poner contraseñas que eran un poco engorrosas al tener que escribir un carácter no alfanumérico como era un signo de puntuación, si no lo explicamos detalladamente los usuarios con menos práctica seguramente tengan problemas para averiguar que es un "carácter no alfanumérico", así que vamos a disminuir el nivel de complejidad de las contraseñas.

Este cambio lo realizaremos en la configuración del Web así que abre la página ya conocida de "web.config" y añade estas líneas justo antes del final de la sección </system.web>

```
<membership>
  <providers>
    <remove name="AspNetSqlMembershipProvider" />
    <add name="AspNetSqlMembershipProvider"
        type="System.Web.Security.SqlMembershipProvider"
        connectionString="LocalSqlServer"
        minRequiredPasswordLength="5"
        minRequiredNonalphanumericCharacters="0"
        passwordStrengthRegularExpression=""
    />
  </providers>
</membership>
</system.web>
```

Como ves son simples propiedades que deben estar en nuestra ayuda, así que se terminó aquello de saberse los objetos y variables de memoria, hay que tener un buen guión y luego la ayuda a mano. Fíjate en las propiedades, le hemos dicho que ahora la contraseña debe ser al menos de 5 caracteres y que el número de caracteres no alfanuméricos puede ser 0.

Ojo con la sintaxis, cualquier falta de ortografía nos dará un error en tiempo de ejecución...

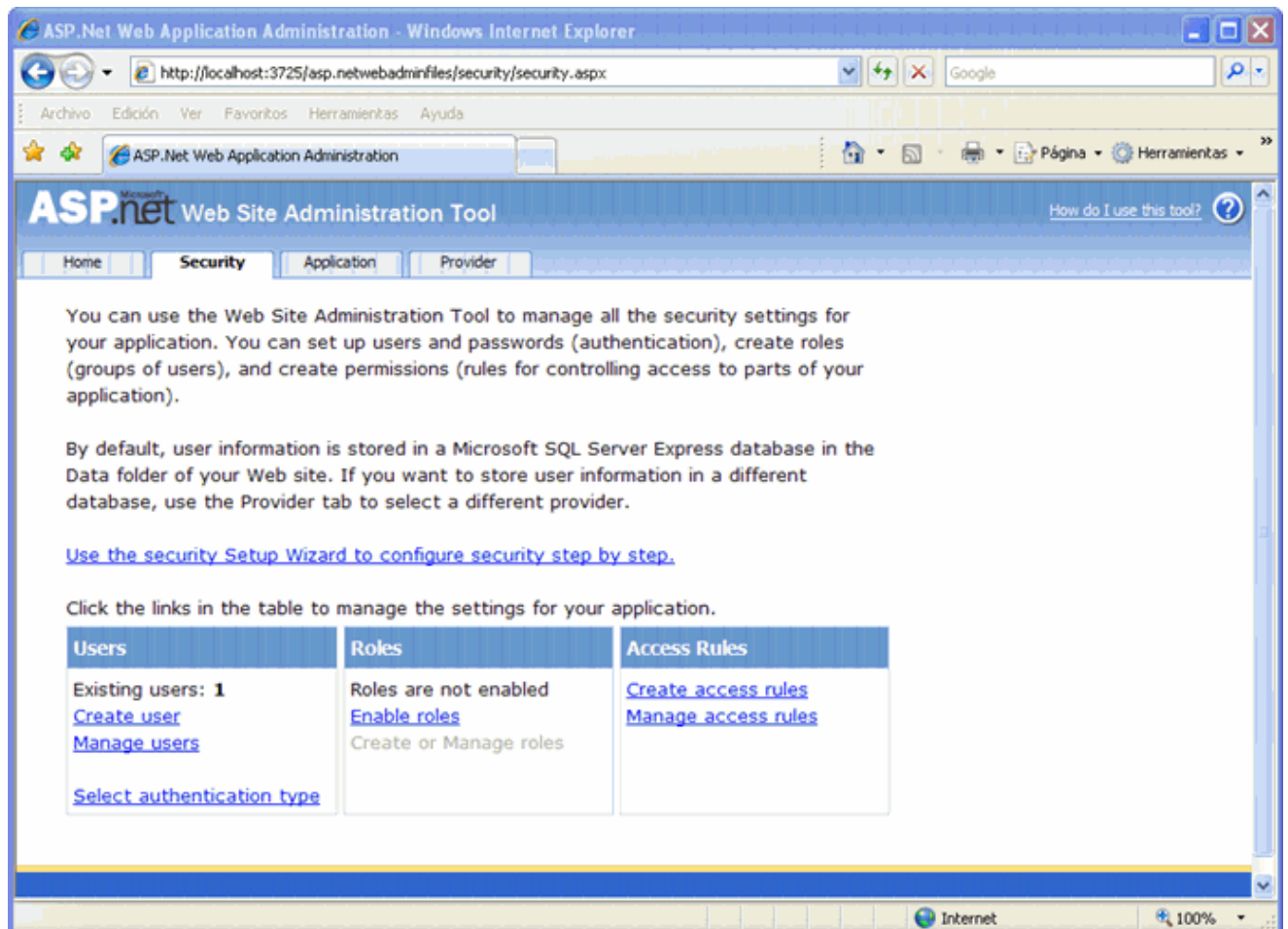
Una cosa, al principio ves que se está "remove" eliminando la gestión de usuarios, esto hace que se borren los usuarios que hemos creado, así que debes crearlos otra vez, como estamos en entornos de pruebas no pasa nada...

No vamos a detenernos mucho en esta configuración porque es de nivel ya a avanzado y apenas se toca. Sólo he querido que vieseis las tres últimas propiedades para dejar que sea de 5 caracteres como mínimo y que no hace falta que incluya un carácter no alfanumérico. Hay información adicional en Internet para poner muchas posibilidades a las contraseñas, por ejemplo limitar el número de intentos, ... pero con lo que hemos visto nos vale para nuestro curso.

Crear usuario con la administración Web

Si necesitamos crear unos usuarios podemos utilizar el administrador web desde la opción de "Seguridad":

Seguridad en ASP.NET



Abajo a la izquierda podemos "crear usuarios". Nos pone un "1" porque hemos creado antes nuestro usuario de ejemplo. Pulsamos para crear:

Create User

Sign Up for Your New Account

User Name:

Password:

Confirm Password:

E-mail:

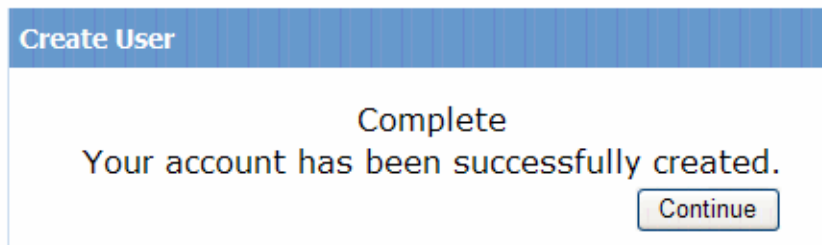
Security Question:

Security Answer:

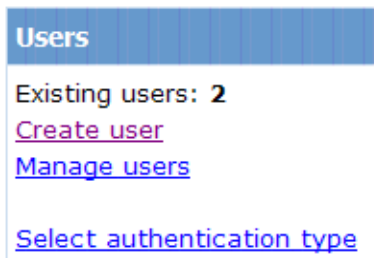
☒ Active User

Seguridad en ASP.NET

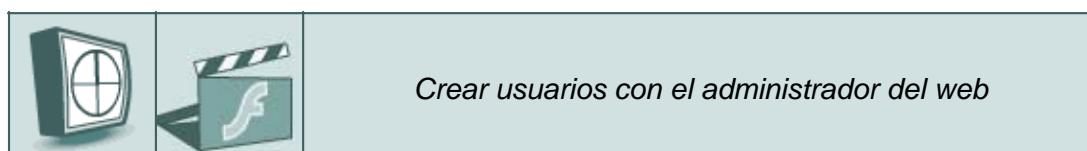
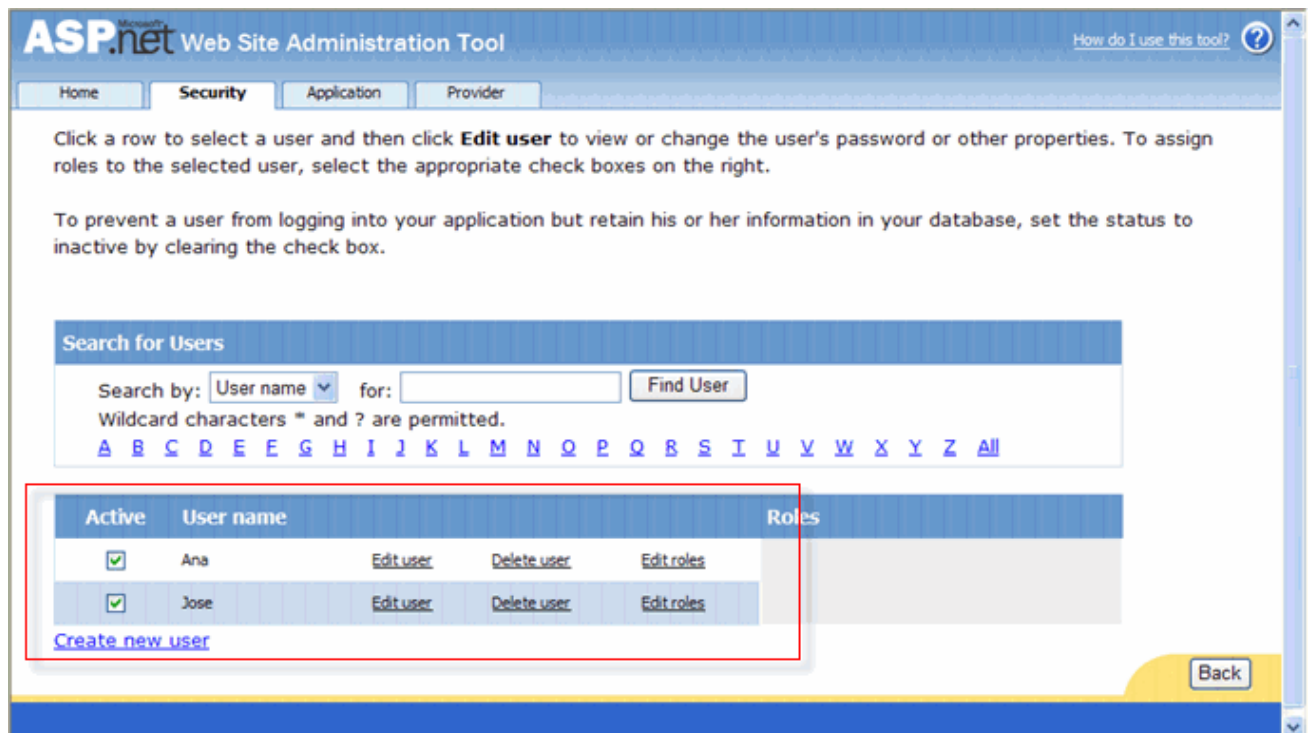
Y si ha ido todo correcto:



Si pulsamos ahora en administrar usuarios:



Podemos verlos para modificar sus datos:



Seguridad en ASP.NET

Donde podemos navegar por todo los usuarios para modificar sus datos. Si necesitamos crear los usuarios mediante código, utilizaremos los objetos "membership" que tienen unos métodos muy sencillos para realizar esto, por ejemplo:

```
'Crear un usuario con su nombre, contraseña y email:
```

```
Membership.CreateUser (usuario, contraseña, mail)
```

```
'Por ejemplo:
```

```
'Membership.CreateUser ("joserma", "password!", "mail@mimail.com")
```

Este código crea un usuario con una sola línea de código. Este método está sobrecargado para permitir poner si queremos todos los valores del usuario:

```
'Por ejemplo:
```

```
'Membership.CreateUser ("joserma", "password!", "mail@mimail.com",
```

```
'      "¿Como se llama mi mascota?", "Manolo", True, createStatus)
```

Los tres primeros parámetros ya los conoces, los dos siguientes son la pregunta y respuesta para validar al usuario en caso de que quiera que se le mande la contraseña por olvido. El siguiente valor, booleano, indica si está activa o no. Si es "false" se crea pero inactiva y a "true" se crea operativa. Para reactivarla habría que utilizar el método "Membership.UpdateUser()". El último parámetro devuelve un valor de la enumeración "MembershipCreateStatus". Si es "success" quiere decir que se ha creado correctamente sino, un mensaje de error con la descripción exacta de lo que ha sucedido.

Las clases Membership y MembershipUsers

Membership, contenida en el espacio de nombres "System.Web.Security" es una útil clase que tiene una serie de métodos compartidos muy prácticos como "CreateUser()". Veamos una lista de los métodos más útiles:

Método	Descripción
CreateUser()	Añade un usuario nuevo a la base de datos
DeleteUser()	Borrar un usuario existente de la base de datos. Se especifica por el nombre de usuario, que es siempre único. Se puede indicar también que se borren todos los datos de las tablas relacionadas
GetUser()	Obtiene un usuario de la base de datos por su nombre
GetUserNameByEmail()	Recupera un nombre de usuario por su correo electrónico. Si existen varios usuarios con el mismo mail se recupera el primero de ellos.
FindUserByName()	Obtiene los usuarios que coincidan con ese nombre, permite nombres parciales para extraer la colección de coincidentes.
FindUserByEmail()	Recupera una lista de los usuarios que contengan ese mail, se puede indicar un dominio para obtener todos los usuarios de ese dominio
GetAllUsers()	Recupera una colección con todos los usuarios existentes.

Seguridad en ASP.NET

GetNumberOfUsersOnline()	Devuelve el número de usuarios que actualmente están accediendo a la aplicación web. Este cálculo asume que está activo cuando su última actividad no ha sido superior a 20 minutos.
GeneratePassword()	Genera una contraseña aleatoria con la longitud especificada. Esto es interesante cuando se crean usuarios mediante programación
UpdateUser()	Actualiza los datos del usuario en la base de datos
ValidateUser()	Comprueba si el nombre usuario y contraseña son válidos

La clase Membership también proporciona unas propiedades compartidas de solo lectura que permite obtener información adicional. Por ejemplo para saber el número máximo de intentos realizados o la longitud de la contraseña.

Muchos de estos métodos utilizan la clase "MembershipUser" que representa un registro. Por ejemplo, GetUser() devuelve la información de un objeto "MembershipUser". Esta clase tienen también algunos métodos interesantes. Date cuenta que antes era para la colección global de usuarios, ahora estos métodos son para cada usuario (MembershipUser).

Método	Descripción
UnLockUser()	Reactiva a un usuario que se le ha bloqueado la contraseña por exceso de intentos erróneos
GetPassword ()	Obtiene la contraseña de un usuario. Si "requiresQuestionAndAnswer" es "true" se debe indicar la respuesta a la pregunta de comprobación. Este método no funciona con la opción por defecto del formato de password, es decir si "FormatPassword=Hasded"
ResetPassword ()	Restaura la contraseña de un usuario generando una nueva aleatoria. Si "requiresQuestionAndAnswer" es "true" se debe indicar la respuesta a la pregunta de comprobación. Se puede mostrar la contraseña o enviarla por mail.
ChangePassword ()	Cambia la contraseña de un usuario. Se debe proporcionar la contraseña actual para poder realizar esta operación.
ChangePasswordQuestionAndAnswer	Cambia la pregunta y respuesta de la contraseña de un usuario. Se debe proporcionar la contraseña actual para poder realizar esta operación.

Vamos con algún ejemplo. Vamos a crear una página para mostrar todos los usuarios. Para esto crea una página en blanco y pon de momento un DataGrid:

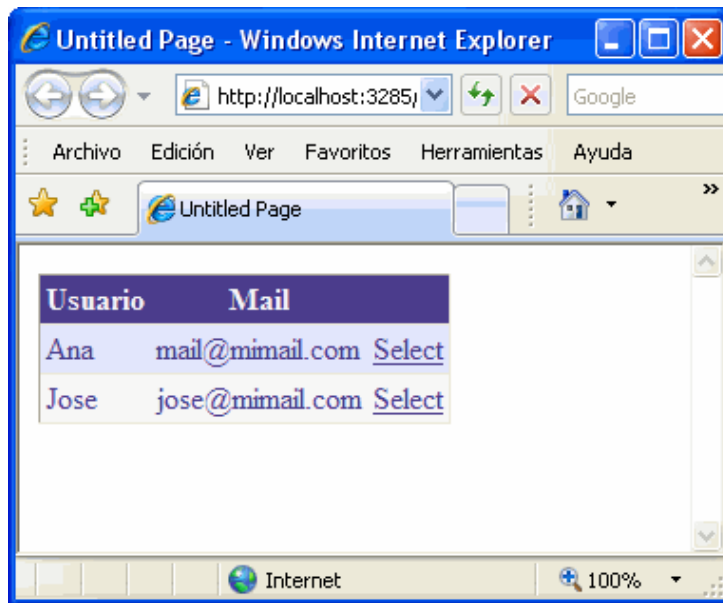
```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False">
  <Columns>
    <asp:BoundField DataField="UserName" HeaderText="Usuario" />
    <asp:BoundField DataField="email" HeaderText="Mail" />
    <asp:CommandField ShowSelectButton="True" />
  </Columns>
</asp:GridView>
```

Aplícale un autoformato un poco elegante y vamos a rellenarlo. Para esto nos vamos al "Page_load" para ponerle a la cuadrícula que el origen de datos es la colección de usuarios:

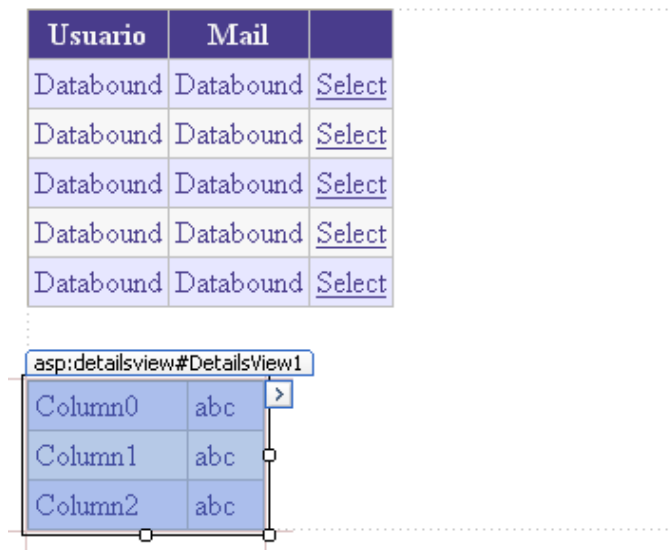
Seguridad en ASP.NET

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
    GridView1.DataSource = Membership.GetAllUsers()
    GridView1.DataBind()
End Sub
```

Que nos dará como resultado en mi caso:



Vamos a ampliar un poco este ejemplo. Colocaremos debajo de esta cuadrícula un control del tipo "detailsview" para que nos muestra todos los datos del usuario:



Vamos a dejar la propiedad "AutoGenerateRows" del "detailsview" a "true" que es su valor predeterminado y lo enlazamos cuando se seleccione un usuario, es decir, cuando se active el evento "SelectedIndexChanged" de la cuadrícula:

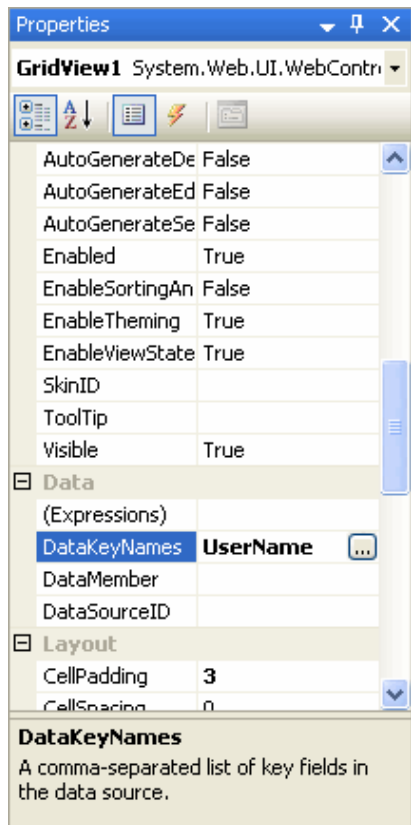
Seguridad en ASP.NET

```
Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs)
    'Declaramos una lista de tipo MembershipUser
    Dim lista As New List(Of MembershipUser) ()

    'Le asignamos el getuser ("usuario")
    lista.Add(Membership.GetUser(GridView1.SelectedValue.ToString()))

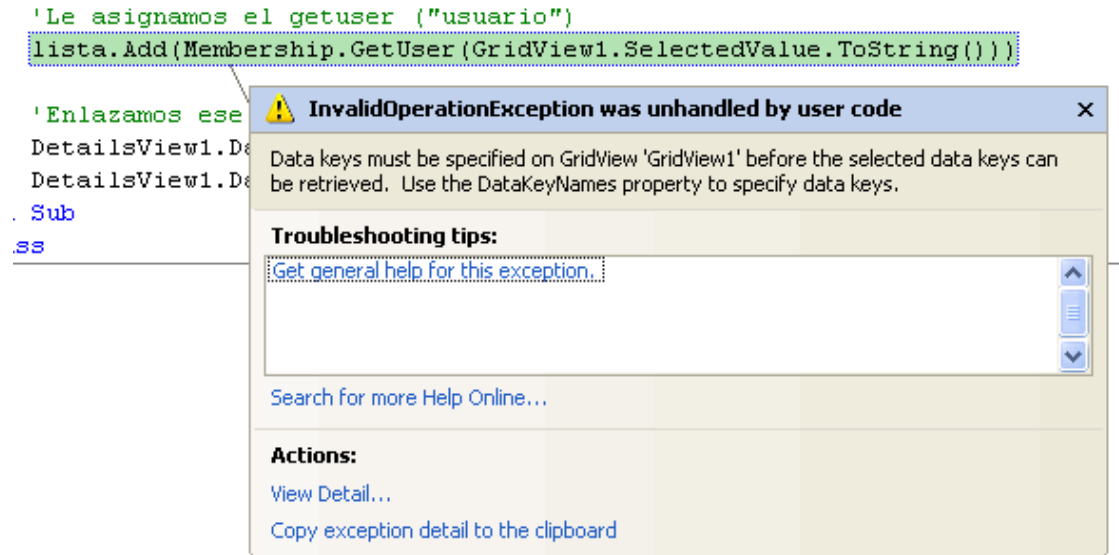
    'Enlazamos ese registro
    DetailsView1.DataSource = lista
    DetailsView1.DataBind()
End Sub
```

Recuerda añadir en la cuadrícula la propiedad;

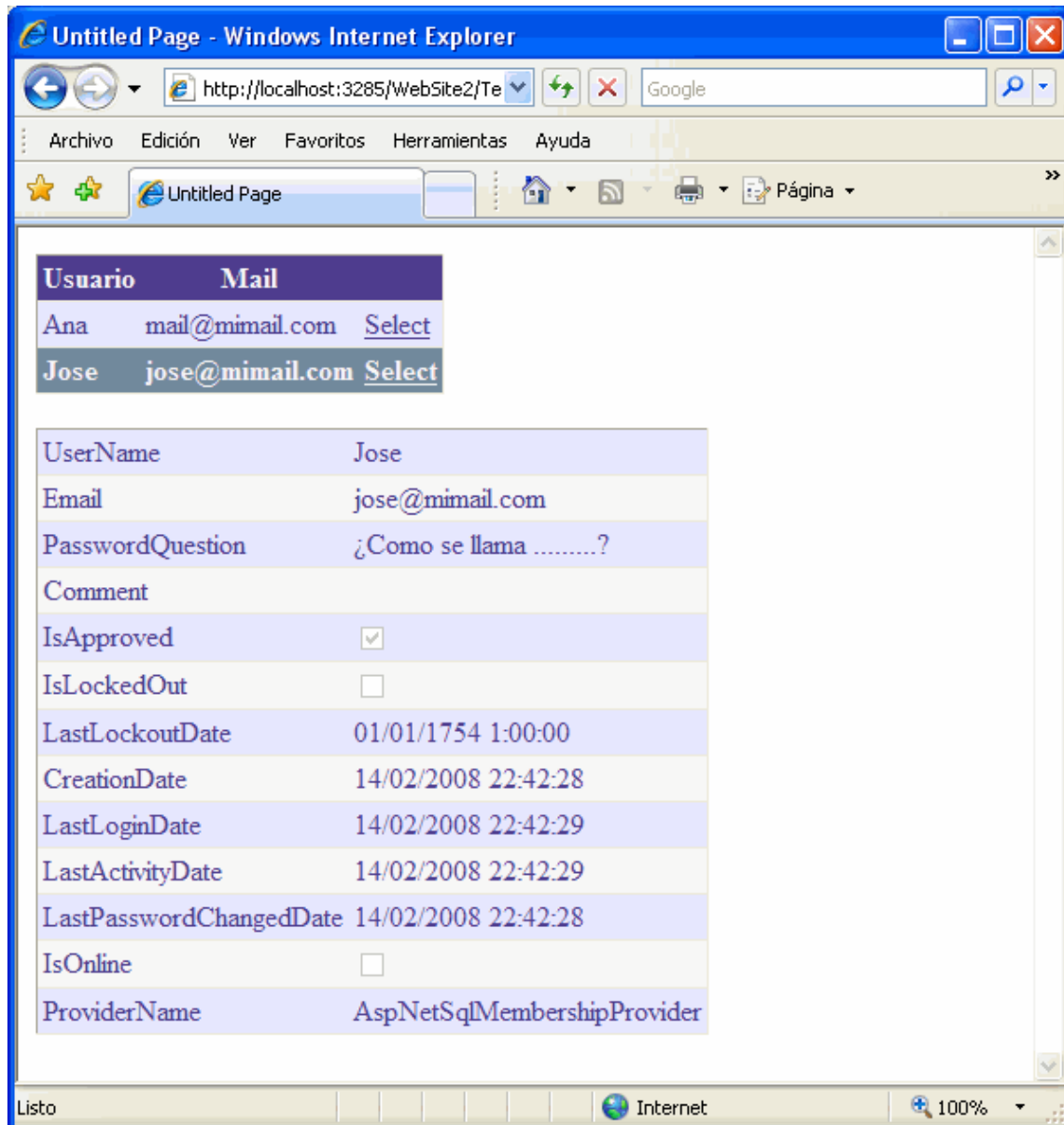


DataKeyNames con el valor "UserName". Recuerda que los campos que pongamos en esta propiedad son los que nos devolverá la cuadrícula cuando seleccionemos un elemento. Si nos dejamos esto obtendremos un mensaje de:

Seguridad en ASP.NET



Es decir, debemos especificar la propiedad "DataKeys" para que puedan recuperarse los datos de la cuadrícula. Una vez puesta la propiedad, ejecutamos y seleccionamos un usuario:



Tendremos todos los datos del usuario, fíjate que las propiedades del usuario ya te empiezan a ser familiares...

Autenticación con "MemberShip"

Una vez que hemos aceptado esta forma de autenticación, en lugar de la de los formularios debemos hacer un pequeño cambio en nuestra página de login para que compare los credenciales introducidos con la base de datos de miembros. Recuerda que antes simplemente comprobábamos si había puesto una contraseña, ya que no disponíamos todavía de una base de datos de usuarios. Antes era así:

Seguridad en ASP.NET

```
Protected Sub btn_inicio_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles bt
    If txt_pass.Text.ToLower = "secreta" Then
        FormsAuthentication.RedirectFromLoginPage(txt_nombre.Text, chk_persistente.Checked)
    Else
        lb_estado.text = "Intentalo de nuevo..."
    End If
End Sub
```

Ahora ya tenemos una base de datos usuarios en las que nos buscará automáticamente si el usuario y contraseña que han metido en la página de login es correcta. Luego en el clic del botón pondremos ahora:

```
Protected Sub btn_inicio_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    If Membership.ValidateUser(txt_nombre.Text, txt_pass.Text) Then
        FormsAuthentication.RedirectFromLoginPage(txt_nombre.Text, False)
    Else
        lb_estado.Text = "Nombre de usuario o contraseña no válidos"
    End If
End Sub
```

Es un cambio fundamental porque antes no hacíamos mas que comparar con una cadena "secreta". Ahora esa instrucción que es uno de los métodos que vimos antes nos busca en la base de datos si existe el usuario.

Deshabilitar cuentas

Una cuenta de usuario puede deshabilitarse por dos razones:

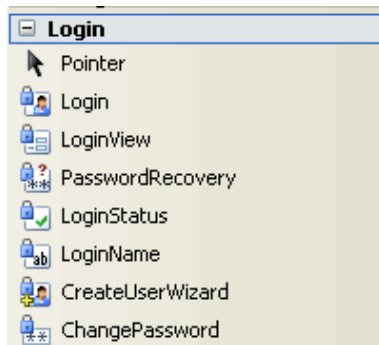
- La hemos creado por código y nos ha devuelto un "false" en el parámetro "isApproved" por la razón que sea. En ese caso tendremos que obtener el objeto "MembershipUser" de ese usuario, establecer "MembershipUser.IsApproved" a "True" y después llamar a "Membership.UpdateUser()" para actualizar los datos.
- La cuenta está bloqueada. Esto se puede dar porque el usuario ha excedido los intentos permitidos de inicio de sesión y ASP.NET bloquea la cuenta. En ese caso tendremos que obtener el objeto "MembershipUser" de ese usuario, llamar al método "MembershipUser.UnlockUser()". También podemos reiniciarle la contraseña con "Membership.ResetPassword()".

Para ayudarnos en esta labor podemos crear una sencilla página como la que vimos antes pero solo con los usuarios bloqueados o deshabilitados. Así podemos comprobar cada cierto tiempo las cuentas que están esos estados. También podemos deshabilitar la cuenta de un usuario cuando queramos simplemente poniendo "MembershipUser.IsApproved" a "False"

Vamos a ver ahora los controles sobre todo este tema de la seguridad que nos proporciona ASP.NET

5. Controles de seguridad

Todo lo que hemos visto hasta ahora de las bases de datos de usuarios y de las clases de "Membership" son muy interesante y realmente se utilizan mucho pero ahora vamos a utilizar los controles de ASP.NET para que nos haga todo el trabajo de, incluso, mostrarnos la pantalla de login. Vamos a ver ahora los controles que tenemos para mantener todo esto:



Que hacen:

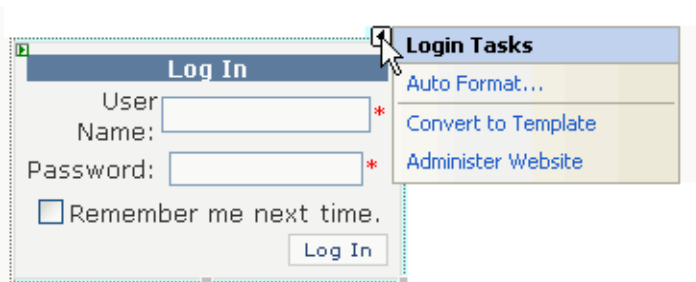
Control	Descripción
Login	Muestra unos cuadros de texto para el inicio de sesión
LoginStatus	Muestra un botón de login si el usuario todavía no se ha registrado y que le manda a la página de login. Si ya está registrado le muestra la opción contraria para cerrar la sesión
LoginName	Muestra el usuario que ha hecho login
LoginView	Muestra distinto contenido dependiendo de si el usuario ha hecho login
PasswordRecovery	Permite al usuario solicitar la contraseña por mail o reiniciarla, habitualmente se le solicita la respuesta a la pregunta de seguridad.
ChangePassword	Permite al usuario cambiar la contraseña
CreateUserWizard	Permite crear un nuevo usuario solicitando al usuario todos los valores de la cuenta.

5.1 Control de login

El control del inicio de sesión o login contiene los cuadros de texto para el usuario y contraseña pero además tiene:

- Un validador para prevenir que no se envía hasta que el usuario ha escrito los dos valores.
- Redirige automáticamente cuando la identificación ha sido correcta
- Proporciona una casilla de verificación para "recordarme la próxima vez" creando una "cookie" persistente.

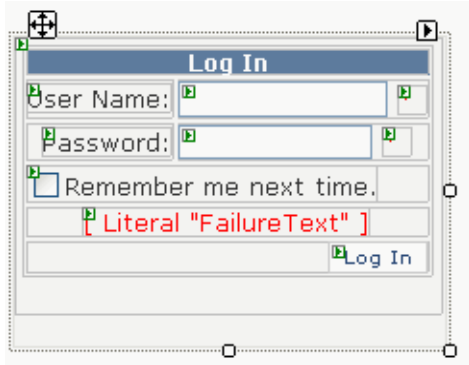
Crea una página nueva "login.aspx" y añade un control de este tipo



Seguridad en ASP.NET

Es para comentar que cuando utilizemos estos controles de servidor podremos editar algunos datos desde estas opciones:

- Formato automático. El autoformato como ya hemos visto nos muestra una útil lista de formatos predefinidos, obviamente internamente no es mas que una lista de estilos para la fuente, cuadro, colores... pero que nos ayudan a dar un rápido y vistoso formato a los controles de servidor.
- Convertir un control en plantilla.



Como hemos dicho antes el autoformato consiste en aplicarle una serie de estilos a todos y cada uno de los elementos que forman la tabla o en este caso, el control de "login". Todo esto se agrupa en una plantilla de tipo "login" que podemos modificar aquí.

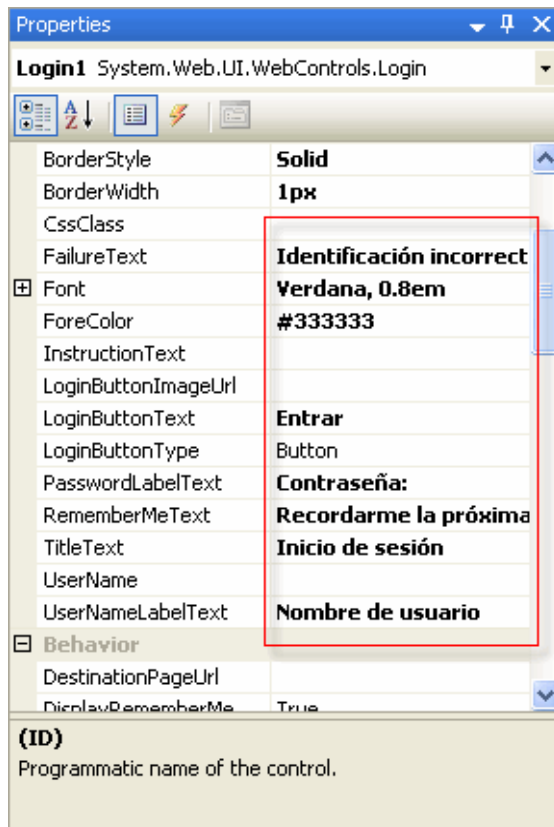
Cada vez que seleccionas uno de estos elementos puedes editar sus propiedades en el panel de propiedades de la derecha.

- Administrar el sitio Web. Abre el sitio Web de administración, este Web ya lo vimos en el capítulo anterior y con él creamos si recuerdas los usuarios y las reglas de acceso.
- Editar plantillas. Esta opción no aparece en todos los elementos, como has visto en este caso del Login. Pero si ponemos un control de tipo tabla de datos:

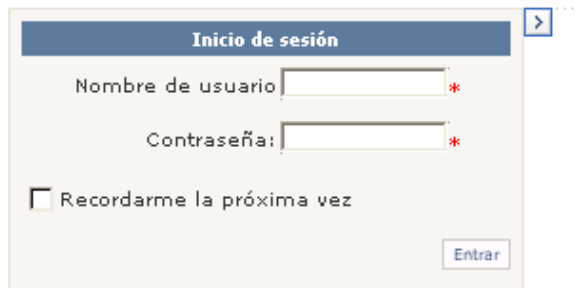
Podremos editar las plantillas como si fueran pequeñas páginas Web en el sentido de que podemos escribir texto o añadir controles. Bien, una vez vistas las opciones mas o menos comunes a los controles vamos a empezar a trabajar con ellos.

Edita las propiedades de la derecha para poner todos los textos en español:

Seguridad en ASP.NET



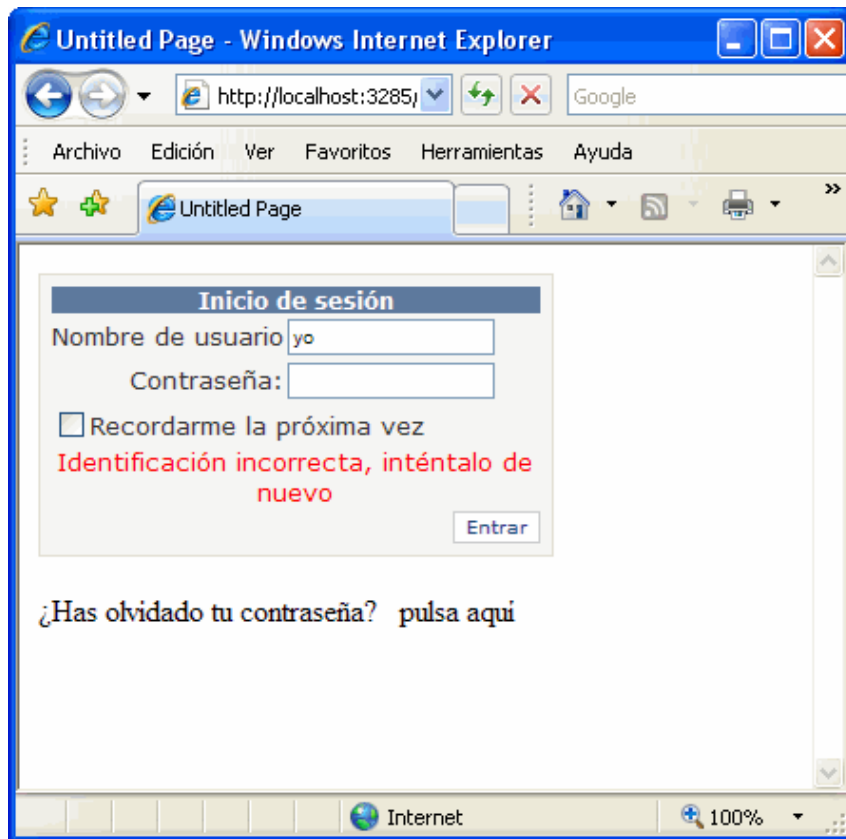
Y así obtener una pantalla como esta:



Los eventos que podemos manejar son:

Evento	Descripción
LogginIn	Se activa cuando se ha validado al usuario
LoggedIn	Se activa después de haberse autenticado
LoginError	Cuando introduce algún dato erróneo y no puede validarse
Authenticate	Se activa cuando se ha validado el usuario, ahora veremos un ejemplo

Los eventos son interesantes para mostrar información adicional según lo que haya sucedido. Por ejemplo, si no se ha validado la primera vez correctamente podemos ponerle un enlace para que recupere su contraseña:



El código estaría en el evento de error en el login:

```
Protected Sub Login1_LoginError(ByVal sender As Object, ByVal e As System
    lb_estado.Text = "¿Has olvidado tu contraseña?"
    HyperLink1.Visible = True
End Sub
```

El texto de "pulsa aquí" es un control de tipo hipervínculo que llevaría a la página con los controles para recordar la contraseña y que veremos mas adelante. Otro de los eventos mas importantes es el de "Authenticate" que es el que ya conocemos para comprobar si es un usuario correcto o no. Esto lo poníamos antes en el evento clic del botón:

```
Protected Sub Login1_Authenticate(ByVal sender As Object, ByVal e As
    If Membership.ValidateUser(Login1.UserName, Login1.Password) Then
        e.Authenticated = True
    Else
        e.Authenticated = False
    End If
End Sub
```

Aquí es donde decidimos si se autentica o no, así que podíamos realizar alguna acción mas si queremos. En el caso de que sea correcto "e" le asigna el valor de autenticado. Recuerda que "e" es el parámetro de entrada genérico que si ves en la declaración es:

```
e As System.Web.UI.WebControls.AuthenticateEventArgs
```

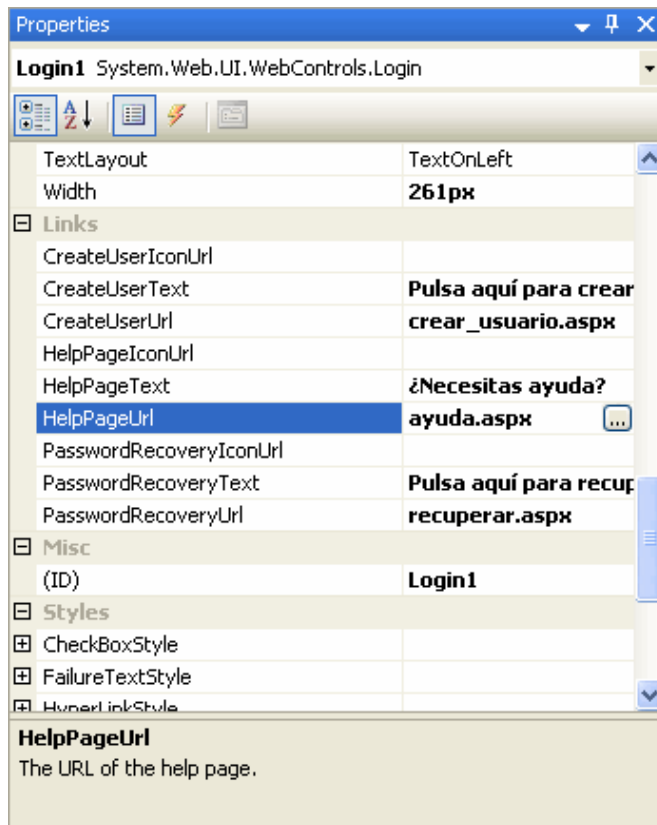
Seguridad en ASP.NET

Por tanto al ponerle "e.authenticated=true" le estamos validando y aceptando el login. El usuario será redirigido a la página indicada en "Login.DestinationUrl" o si está en blanco a la página de donde venía el usuario. Nosotros pondremos la "default.aspx" para que pueda acceder ya a la página de nuestra aplicación Web. Si ponemos "e.authenticated=false" denegamos pues el acceso y se dispara el evento "LoginError" mostrando el mensaje de error "Login.FailureText"

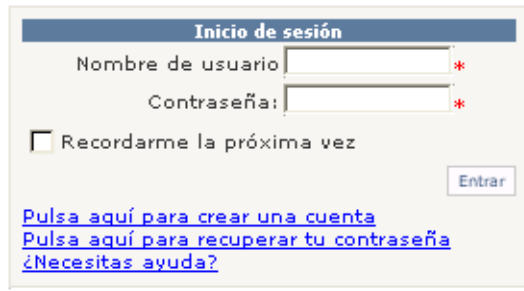
Este control no es estrictamente necesario ya que sin él todo funcionaría correctamente pero la idea es que podamos incluir algo de código extra por si necesitamos realizar alguna acción. Por ejemplo si queremos que se haga login en un intervalo determinado de horas... en caso de que no sea ese horario pondremos siempre "false". Además si todavía no estamos utilizando la base de datos de usuario o utilizamos otra pondremos aquí las validaciones necesarias.

Ya has visto que tienes el autoformato para mejorar el aspecto del control pero curioseas también con todas las propiedades que terminan en "style": TitleStyle, LabelStyle, ValidatorStyle, LoginButtonStyle...

También tenemos unas propiedades para cuando ha olvidado la contraseña, para poner así de forma automática lo que hemos escrito antes a mano y también para indicarle la página para cuando quiera crear una contraseña, o para obtener ayuda. Al ponerlas:



Se muestran en la ventana de login:



Inicio de sesión

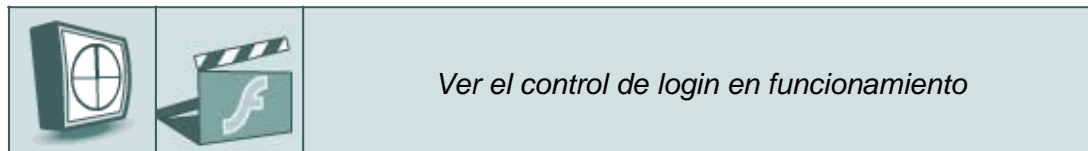
Nombre de usuario: *

Contraseña: *

☐ Recordarme la próxima vez

[Pulsa aquí para crear una cuenta](#)
[Pulsa aquí para recuperar tu contraseña](#)
[¿Necesitas ayuda?](#)

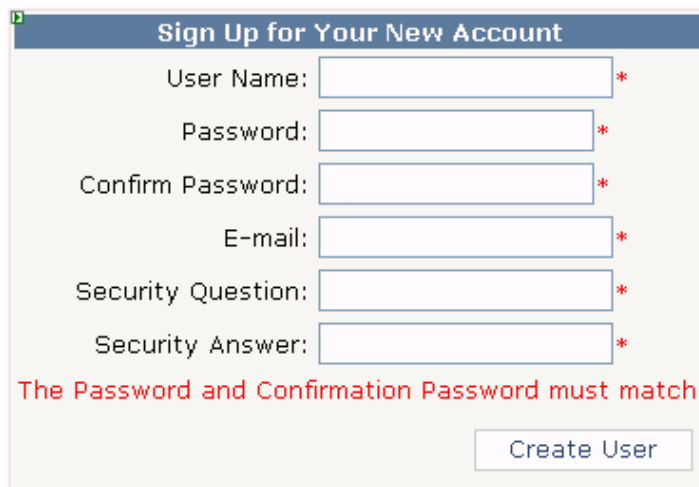
Explora todas las propiedades porque la verdad es que permiten una versatilidad completa. Vamos ahora a permitir al usuario crear una cuenta de usuario que según hemos puesto en el control anterior lo haremos en la página "crear_usuario.aspx"



5.2 Permitir a los usuarios crear una cuenta

En nuestro sitio Web permitiremos que los usuarios creen sus propias cuentas de usuario así que vamos a utilizar ahora los controles necesarios para esto.

El control "CreateUserWizard" está diseñado para este proceso y nos va a mostrar en pantalla todos los elementos necesarios para hacer este proceso de creación de cuenta:



Sign Up for Your New Account

User Name: *

Password: *

Confirm Password: *

E-mail: *

Security Question: *

Security Answer: *

The Password and Confirmation Password must match.

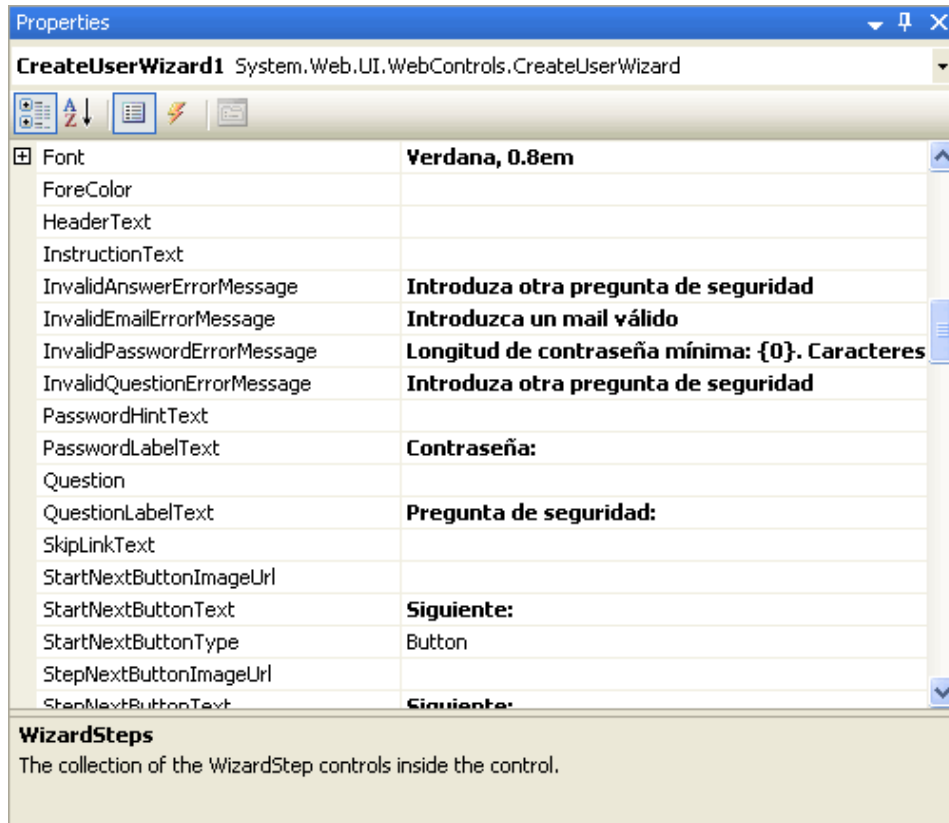
Vamos a añadir una nueva página en nuestro Web de pruebas que se llame "crear_usuario.aspx". Una vez creada arrastraremos este control a nuestra nueva página:

Tenemos varios grupos de propiedades:

- Propiedades de estilos que dan formato al control
- Propiedades de los textos del control
- Propiedades para las partes ocultas del control

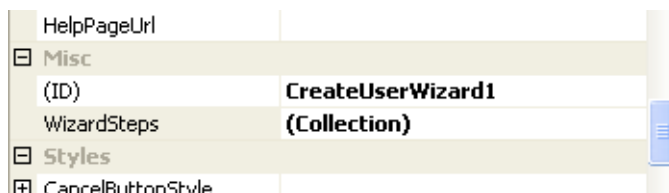
Seguridad en ASP.NET

Tenemos muchas propiedades para personalizarlo. Para empezar vete explorando las de los mensajes en ingles para que los vayas traduciendo y así hacerte una idea de la cantidad que tiene:

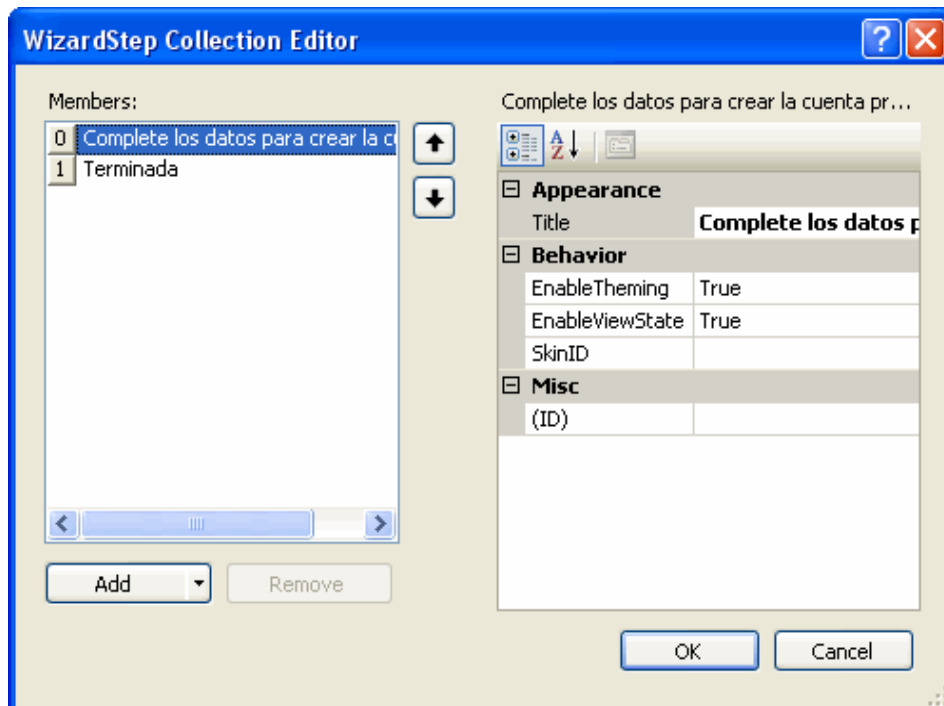


Este control se comporta como un asistente, con una pantalla para la introducción de datos y otra para cuando ha ido todo correcto, ahí es donde pondremos los títulos de las dos ventanas, en la propiedad "WizardSteps":

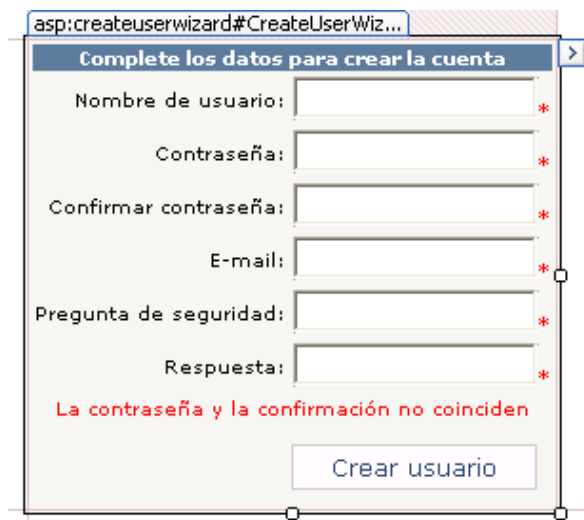
que nos muestra:



Donde le pondremos esos datos:



Para dar un aspecto final como por ejemplo este:



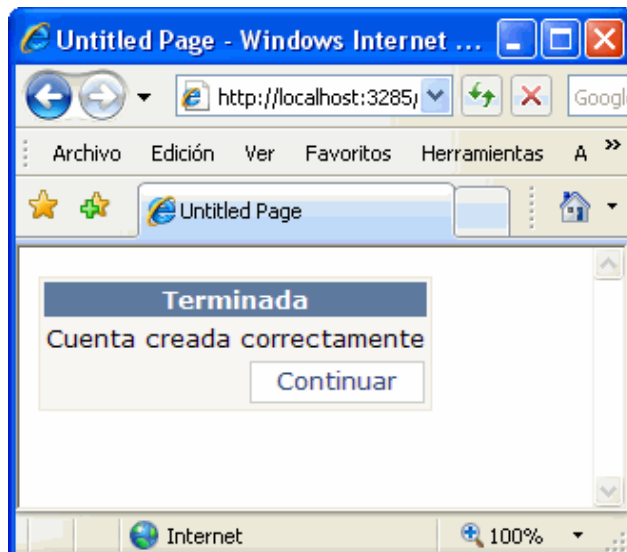
Tendremos como eventos los habituales y los ya conocidos de: CreatingUser, CreatedUser y CreateUserError. Que están sincronizados con todo el entorno. Si recuerdas, cuando vimos este tema pusimos un control de este tipo que ya funcionaba completamente, luego si lo ejecutamos lo tendremos totalmente operativo. Veamos produciendo un error:

The screenshot shows a Windows Internet Explorer window titled "Untitled Page - Windows Internet Explorer". The address bar displays "http://localhost:3285/WebSite2/crear_usua". The menu bar includes "Archivo", "Edición", "Ver", "Favoritos", "Herramientas", and "Ayuda". The toolbar shows various icons including a star, a plus sign, a home button, a RSS feed icon, a printer icon, a document icon, and a settings icon. The main content area displays a registration form titled "Complete los datos para crear la cuenta". The form contains the following fields and values:

- Nombre de usuario: MiguelR
- Contraseña: (empty)
- Confirmar contraseña: (empty)
- E-mail: miguel@mimail.com
- Pregunta de seguridad: pregunta...
- Respuesta: respuesta...

Below the form, there is a red text message: "Longitud de contraseña mínima: 7. Caracteres no alfanuméricos requeridos: 1." and a button labeled "Crear usuario". The status bar at the bottom shows "Listo" and "Internet" with a zoom level of 100%.

Y si es todo correcto:



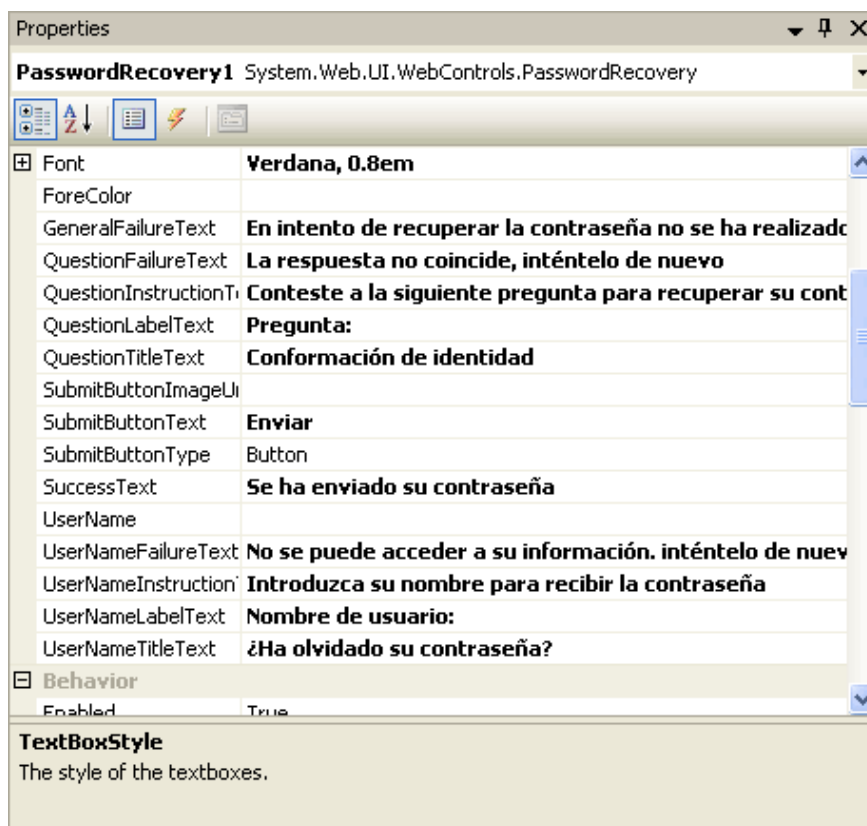
Nos habrá introducido al usuario en la base de datos con lo que tenemos el objetivo cumplido.

5.3 Control para recuperación de contraseña

Este control permite al usuario recuperar la contraseña si la ha olvidado. Fundamentalmente lo que hará será pedirnos la pregunta secreta. El proceso se realiza en tres pasos utilizando un sencillo asistente. Vamos a crear una página que llamaremos recuperar.aspx y donde pondremos un control de tipo:

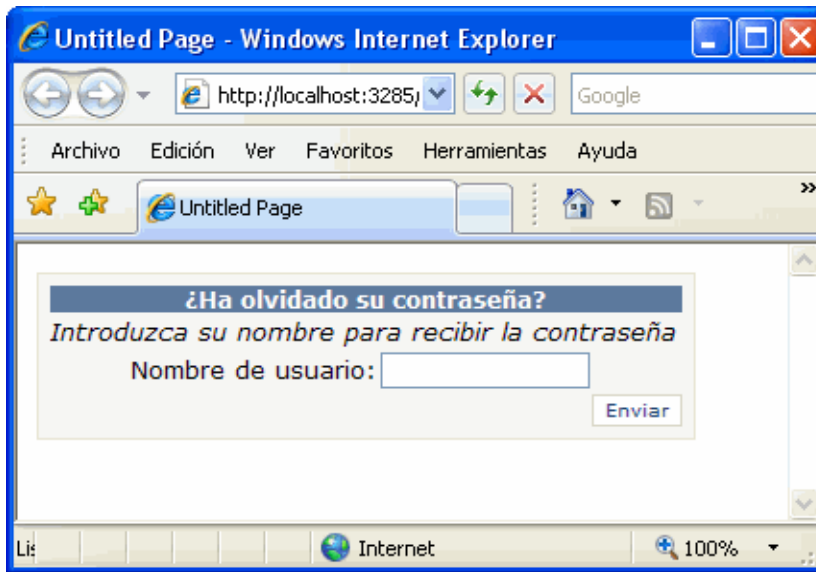


Los pasos son los siguientes... primero se solicita el nombre del usuario, luego se solicitará la pregunta de seguridad y si es correcta se mandará un mail al usuario con la contraseña. Existen varios formatos de la contraseña, si se utilizan las de tipo "Encrypted" o "Clear" se mandará la original pero si se utiliza el formato predefinido, llamado "hash" se envía una nueva contraseña. De momento vamos a poner un estilo de autoformato e iremos traduciendo los textos del control para así explorar las propiedades:

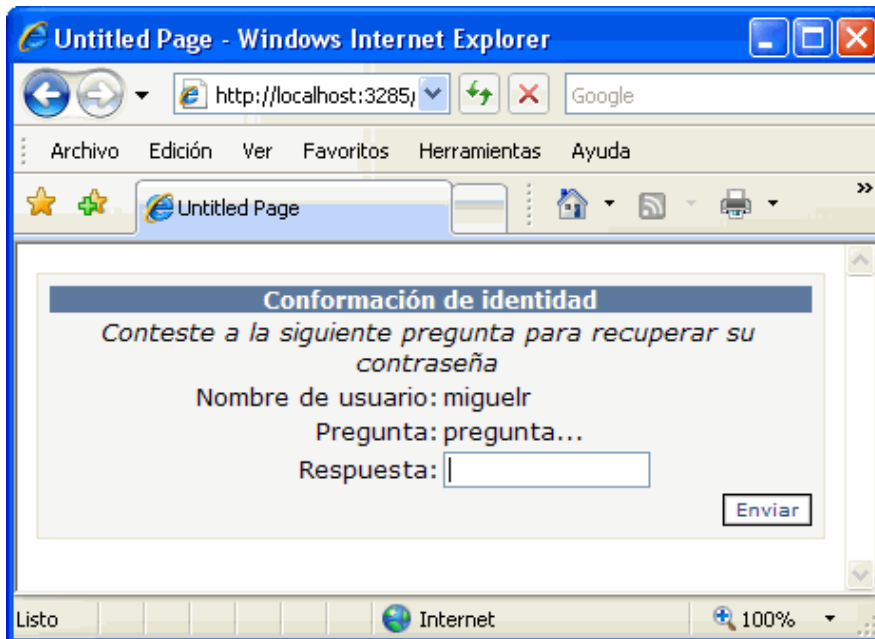


Vamos a ejecutar la página:

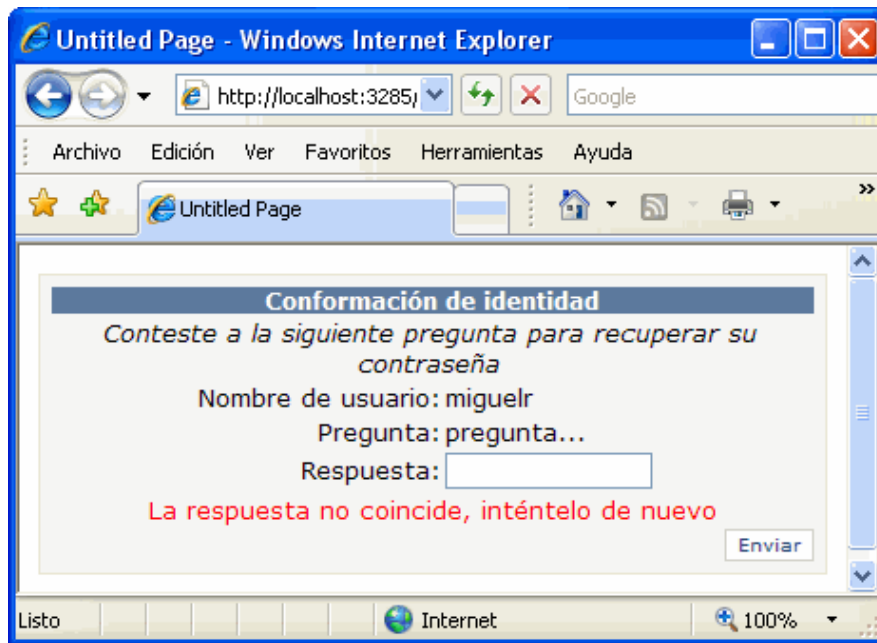
Seguridad en ASP.NET



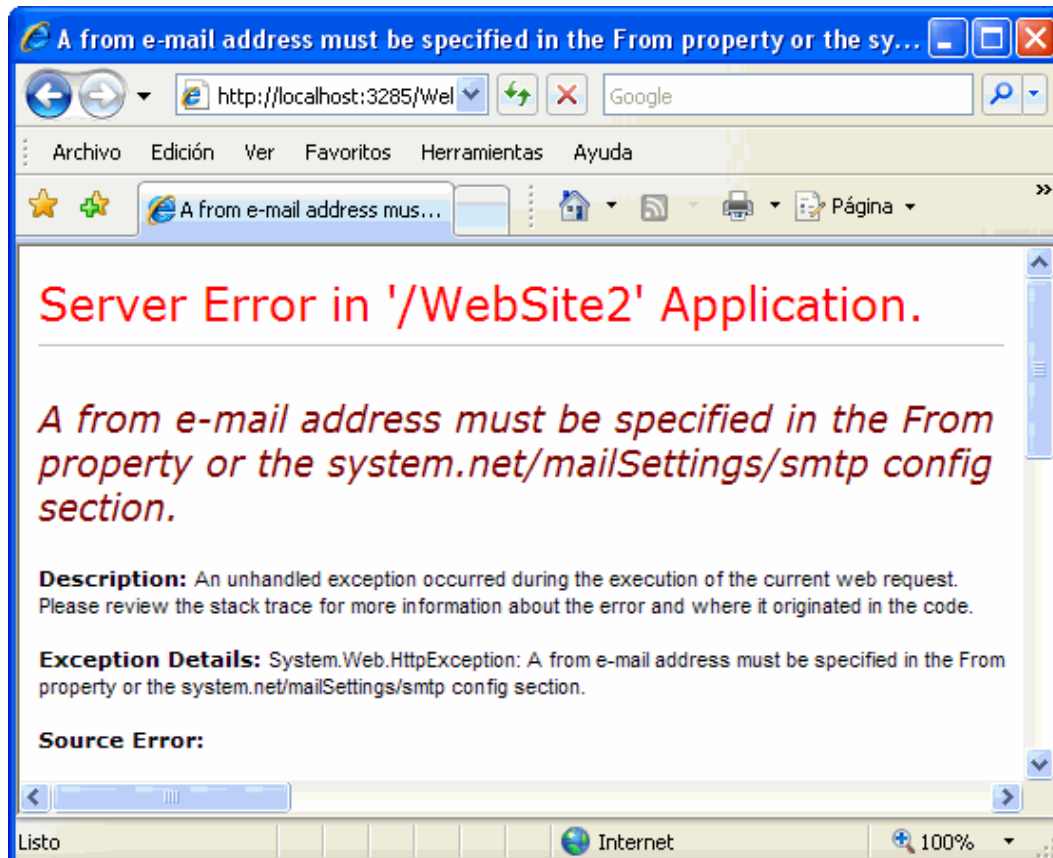
Ponemos un usuario válido y pulsamos en "enviar":



El usuario existe y nos pide la pregunta de seguridad, si no es correcta:

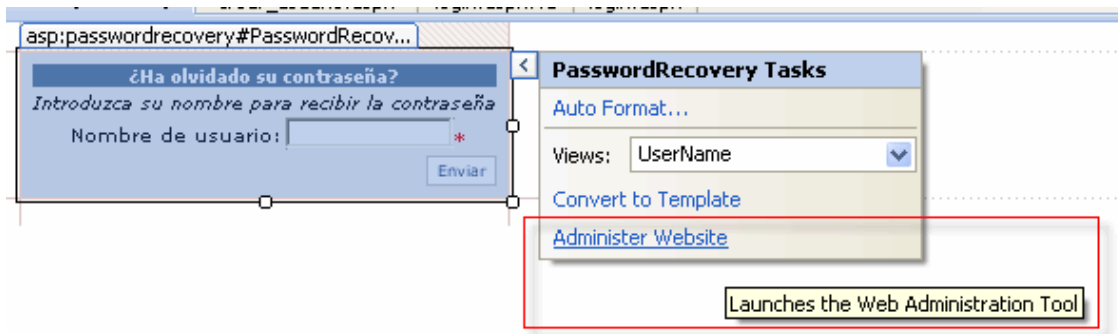


Y si todo es correcto nos enviará un mail:

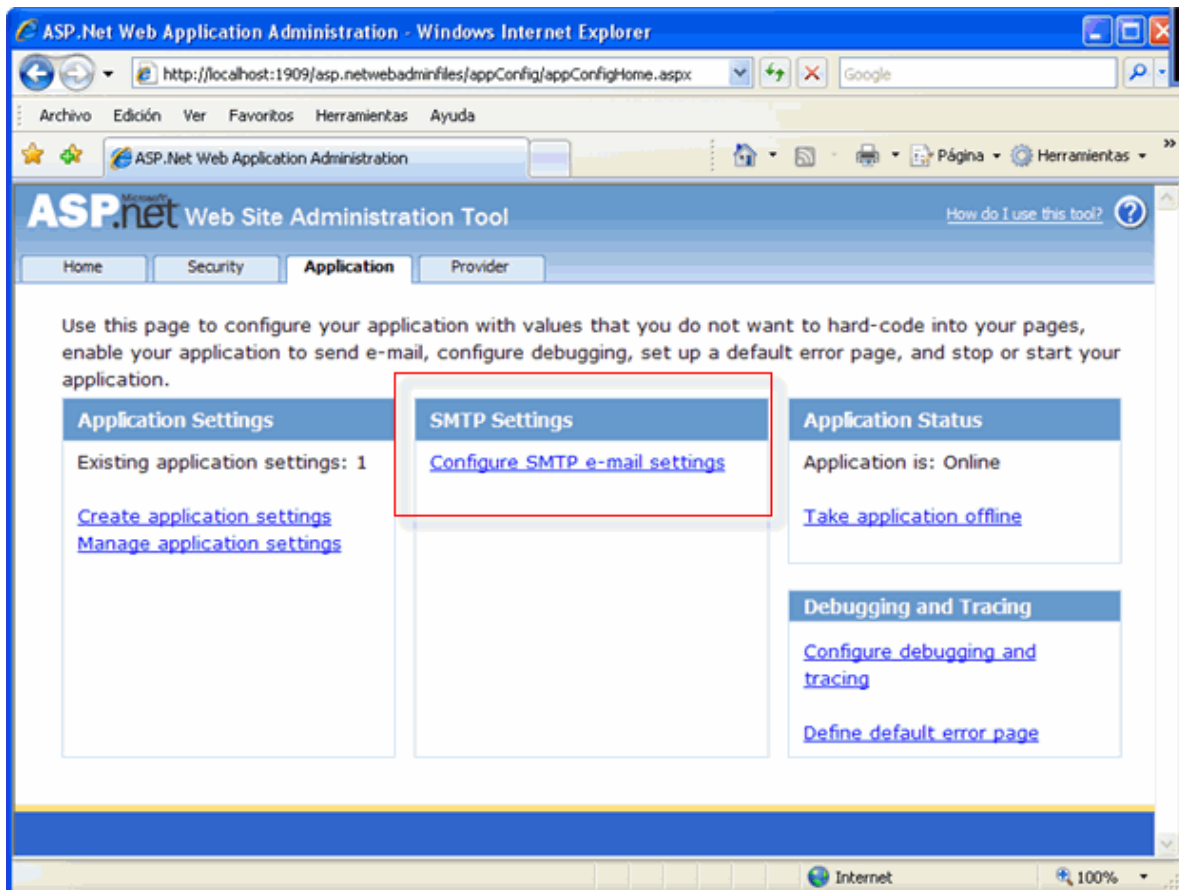


Pero nos ha dado error en el último paso. Si vemos el mensaje nos pide que configuremos un servidor de correo en el fichero de configuración. Para esto nos vamos al servidor web de administración pulsando:

Seguridad en ASP.NET



Nos mostrará nuestra conocida página de configuración del web.



Elegimos la opción del centro para configurar un servidor de correo SMTP:

Configure SMTP Settings	
Server Name:	<input type="text" value="servidor_de_correo"/>
Server Port:	<input type="text" value="25"/>
From:	<input type="text" value="Aplicación Web"/>

Estos datos si no los conoces puedes pedirselos a tu administrador de red o a tu proveedor de Internet para indicar por ejemplo, el servidor de correo de tu empresa.

5.4 Cambiar contraseña

Este control se mostrará en una de las páginas en las que el usuario tienen permiso y se haya "logineado ya". Crea una página que se llame "cambiar_pwd.aspx" y le pones el control:



Change Your Password

Password: *

New Password: *

Confirm New Password: *

The Confirm New Password must match the New Password entry.

Change Password Cancel

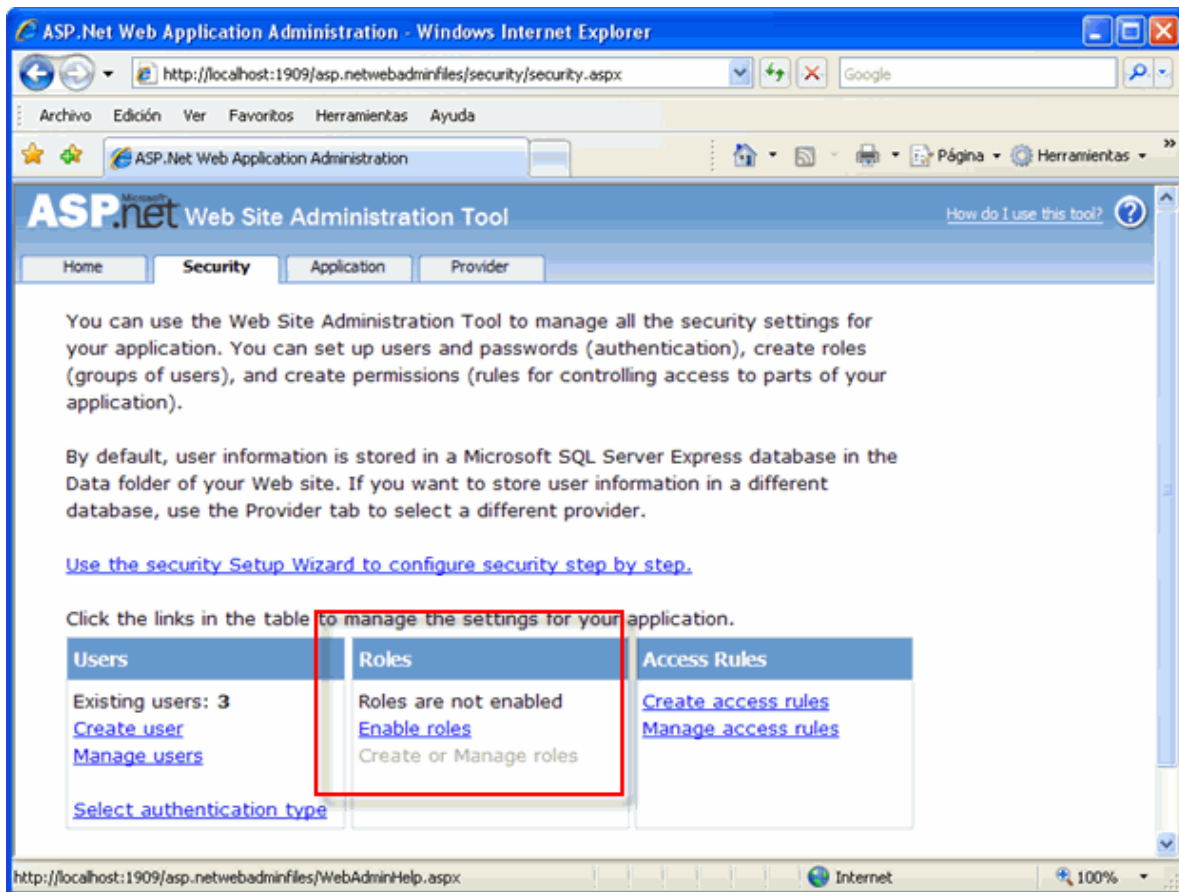
Haz lo mismo que en los otros casos para poner en castellano los textos y así explorar las opciones. Es un control muy sencillo y obligatorio para que el usuario pueda actualizar su contraseña, sobre todo si se le ha proporcionado una aleatoria que es imposible de recordar.

6. Seguridad basada en roles

Las aplicaciones necesitan a menudo distintos grupos de acceso para poder establecer distintos niveles de usuarios. Por ejemplo una carpeta para los usuarios de administración. Podríamos crear un grupo que se llame "administración" y asignar a los usuarios a ese departamento mediante lo que en ASP.NET llama "roles". Luego en las carpetas de mi web le daremos permisos a esos roles en lugar de a los usuarios individuales. Además si hacemos una seguridad basada en roles podemos personalizar los menús para que muestre unas opciones u otras dependiendo del rol. Por ejemplo el menú "informes" solo se le mostrará a los usuarios que tengan el "rol" de "administración"

Para administrar los roles utilizaremos la administración web:

Seguridad en ASP.NET



Pulsamos en "Enable roles":



Y ahora en "Create or Manage roles" para crearlos:

Create New Role	
New role name:	<input type="text" value="Administracion"/> <input type="button" value="Add Role"/>

Y después de crear unos cuantos tenemos:

Seguridad en ASP.NET

Create New Role

New role name:

Role Name	Add/Remove Users
Administracion	Manage Delete
Compras	Manage Delete
Informática	Manage Delete

Ahora asignaremos usuarios a los roles desde la administración de usuarios:

Search for Users

Search by: for:

Wildcard characters * and ? are permitted.

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#)

Active	User name			Roles
<input checked="" type="checkbox"/>	Ana	Edit user	Delete user	Edit roles
<input checked="" type="checkbox"/>	Jose	Edit user	Delete user	Edit roles Edit roles [Ana]
<input checked="" type="checkbox"/>	MiguelR	Edit user	Delete user	Edit roles

[Create new user](#)

Para añadir al usuario a los roles que queremos:

Active	User name			Roles
<input checked="" type="checkbox"/>	Ana	Edit user	Delete user	Edit roles
<input checked="" type="checkbox"/>	Jose	Edit user	Delete user	Edit roles
<input checked="" type="checkbox"/>	MiguelR	Edit user	Delete user	Edit roles

Add "Jose" to roles:

☒ Administracion

☐ Compras

☒ Informática

Donde iremos marcando los roles que puede tener cada usuario. Pero también podemos realizar esto por programación utilizando los objetos adecuados con la clase "Roles". Los métodos que tendremos serán:

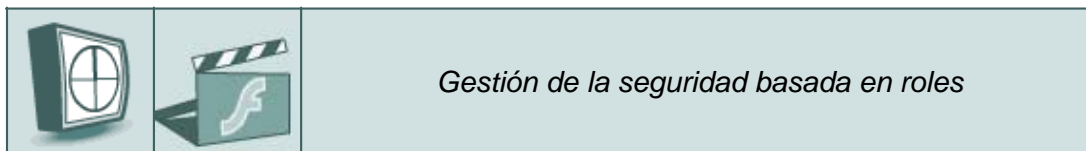
Método	Descripción
CreateRole()	Crea un rol en la base de datos
DeleteRole()	Elimina un rol
RoleExists()	Comprueba si el nombre del rol especificado existe en la base de datos

Seguridad en ASP.NET

GetAllRoles()	Recupera una lista de todos los roles de la aplicación
AddUserToRole() AddUserToRoles() AddUsersToRole() AddUsersToRoles()	Asigna un rol a un usuario, varios roles a un usuarios, un role a varios usuarios o varios roles a varios usuarios.
RemoveUserFromRole() RemoveUserFromRoles() RemoveUsersFromRole() RemoveUsersFromRoles()	Permite borrar un usuario de un rol, de varios roles, varios usuarios de un rol y varios usuarios de varios roles
InUserRole()	Comprueba si un usuario forma parte de un rol
GetRolesForUser()	Obtiene todos los roles de un usuario especificado
GetUsersInRole()	Obtiene los usuarios de un role
FindUserInRole()	Recupera los usuarios de un role. Filtrando por una parte del nombre especificado

Por ejemplo, podemos crear el siguiente controlador de eventos con el control "CreateUserWizard" para asignar al usuario recién creado a un rol:

```
Protected Sub CreateUserWizard_CreatedUser (Byval sender as Object,
                                           Byval e as System.EventArgs) Handles CreateUserWizard1.CreatedUser
    Roles.AddUserToRole (CreatedUserWizard1.UserName, "Administracion")
End Sub
```



6.1 Restricciones de acceso basado en roles

Una vez que hemos creado los roles necesitamos ajustar nuestra aplicación para que trabaje con la información del rol mediante varias técnicas:

- Podemos escribir reglas de autorización para negar a varios roles el acceso a determinadas páginas o carpetas. Lo podemos hacer escribiéndolos en la sección <authorization> en el fichero web.config. o administrando desde la gestión web
- Podemos utilizar el método User.IsInRole() en el código para decidir si tiene acceso o no
- Podemos utilizar el control "LoginView" para establecer distintos contenidos según los roles.

Por ejemplo podemos dar o quitar permisos a roles en lugar de a usuarios:

```
<authorization>

    <deny users="?">

    <deny roles="Invitados">
```

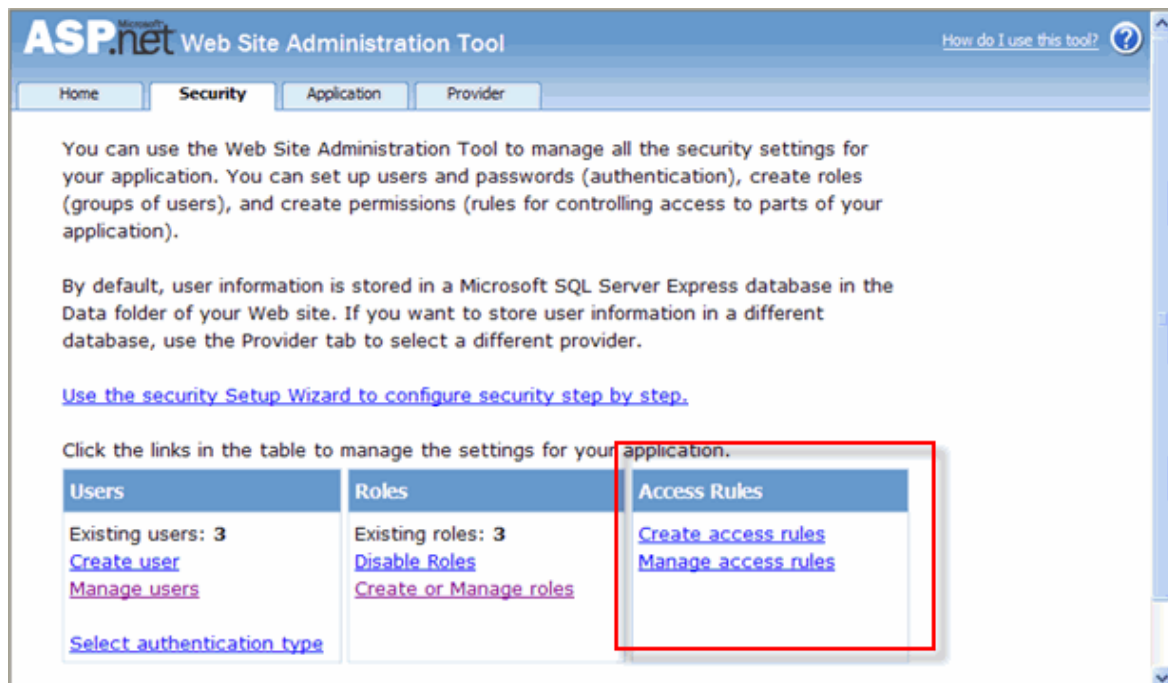
Seguridad en ASP.NET

```
<allow users="*">  
  
</authorization>
```

Deniega a todos los anónimos y de los autenticados deniega a los que son del rol "Invitados". Esto lo podemos aplicar a las carpeta que queremos. Mediante programación podemos ver si está en algún rol para decidir las acciones que queramos:

```
Protected Sub Page_Load ()  
  
    lb_mensaje.Text="Estás en una página segura, "  
  
    lb_mensaje.Text &= User.Identityu.Name  
  
If User.IsInRole ("administracion") then  
  
    lb_mensaje.Text &= "<br> Felicidades, eres administrador.: "  
  
End If  
  
End Sub
```

Mediante el administrador Web podemos poner estos permisos de esta forma tan sencilla:



Creamos una regla de acceso:

El propósito de crear estos perfiles es distinguir a los distintos usuarios que pueden acceder a nuestro sitio Web y así mediante estos grupos darles permisos a una zonas u otras mediante las "reglas de acceso" (access rules).

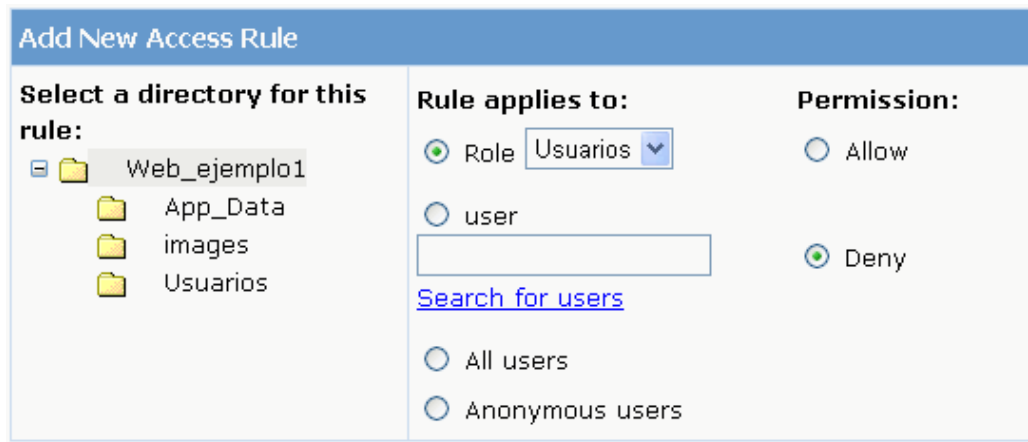
Por ejemplo, en nuestro sitios Web creamos antes una carpeta que se llamaba "Usuarios", he puesto el mismo nombre por claridad pero no tiene porqué serlo. Tranquilamente la carpeta se podía llamar "Informes" y los grupos aquí creados "Finanzas" o "Ventas". Bien, ahora queremos que los usuarios que pertenezcan a este

Seguridad en ASP.NET

perfil tengan acceso a esta carpeta y los demás por supuesto no. Vamos a realizar esto mediante una regla de acceso, en la misma ventana de seguridad pulsamos en:



"Crear access rules":



Select a directory for this rule:	Rule applies to:	Permission:
<ul style="list-style-type: none">Web_ejemplo1<ul style="list-style-type: none">App_DataimagesUsuarios	<input checked="" type="radio"/> Role Usuarios <input type="radio"/> user <input type="text"/> Search for users <input type="radio"/> All users <input type="radio"/> Anonymous users	<input type="radio"/> Allow <input checked="" type="radio"/> Deny

A la izquierda tenemos las carpetas de nuestro Web, en medio los grupos en los que queremos crear la regla y a la derecha el tipo: permitir o denegar. En nuestro ejemplo vamos a eliminar el acceso anónimo, es decir los que no se hayan validado por el formulario de entrada no podrán navegar por la carpeta "usuarios":



Select a directory for this rule:	Rule applies to:	Permission:
<ul style="list-style-type: none">Web_ejemplo1<ul style="list-style-type: none">App_DataimagesUsuarios	<input type="radio"/> Role Usuarios <input type="radio"/> user <input type="text"/> Search for users <input type="radio"/> All users <input checked="" type="radio"/> Anonymous users	<input type="radio"/> Allow <input checked="" type="radio"/> Deny

Date cuenta que muy fácilmente podíamos haber creado otra carpeta dentro de esta de "Usuario" y haber creado un perfil o "rol" nuevo y decirle que "Allow" con lo que permitimos a ese perfil acceder a la carpeta marcada. Pulsamos en "Ok" para que nos guarde esta regla... luego hacemos clic en la parte de administrar estas reglas de acceso (access rules):

Seguridad en ASP.NET

Rules that appear dimmed are inherited from the parent and cannot be changed at this level.

Permission	Users and Roles	Delete
Deny	[anonymous]	Delete
Allow	[all]	Delete

[Add new access rule](#)

Ten en cuenta que estas reglas de acceso se definen para cada carpeta, por eso debes navegar por el árbol de la izquierda para ver los permisos de cada una de ellas. Por ejemplo si hacemos clic en la carpeta "Images":

Permission	Users and Roles	Delete
Allow	[all]	Delete

[Add new access rule](#)

Vemos que se permite a todos el acceso: "Permission: Allow" y "Users and Roles: All". En cambio en la anterior teníamos primero una entrada que restringía el acceso a los que no se autentiquen. Como ves se pueden crear mas de una regla que tienen prioridad según se muestran en la pantalla, de ahí que en la parte derecha tengamos unos botones "Move Up" y "Move Down" que nos va permitir establece la prioridad en la que se deben aplicar.

Esta filosofía de permisos y su administración es una auténtica maravilla comparado con los sistemas que teníamos antes. Es toda una novedad de la versión 2.0 y la verdad es que es realmente cómodo y avanzado.

6.2 Permisos en menús

Si nuestro sitio tiene usuarios u perfiles (roles) puede que según ese perfil les debamos mostrar unas opciones u otras. Por ejemplo habrá usuarios que deban acceder a una opciones del menú para realizar solo consultas a las bases de datos, y otros que tengan que tener todas las opciones. Es decir, nuestros menús deben ser dinámicos de alguna forma para mostrar determinadas opciones.

Solo debemos añadir una etiqueta en nuestro mapa de sitio indicando las opciones de menú que queremos que muestre según los perfiles (roles) de los usuarios que han hecho login. Esto es, cuando inician la sesión estos usuarios pueden pertenecer a un rol u otro según lo hayamos puesto en la administración de nuestro sitio. Pues bien, el que salgan mas o menos opciones en el menú depende de si están en ese perfil o no. Mira esto:

Seguridad en ASP.NET

```
<?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="~/Usuarios/Servicios/Servicios_inicio.aspx"
      title="Servicios de usuarios" description="Pulsa aquí para..."
      roles="Usuarios"/>

    <siteMapNode url="~/Usuarios/Foros/Foros_inicio.aspx"
      title="Foros" description="Pulsa aquí para..."
      roles="Usuarios"/>

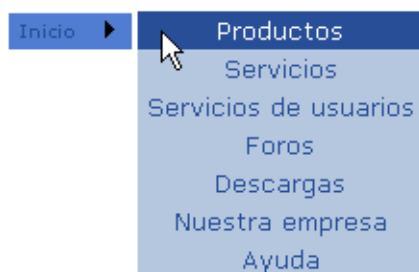
    <siteMapNode url="~/Usuarios/Descargas/Descargas_inicio.aspx"
      title="Descargas" description="Pulsa aquí para..."
      roles="Usuarios"/>

    <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>
```

Hemos añadido un grupo de carpetas dentro de las autenticadas "Usuarios" que son de servicios para estos usuarios: foros y descargas. Y hemos añadido en el mapa del sitio estas tres opciones pero con la etiqueta de "roles" indicando quienes pueden acceder a él. Muy fácil y con muy buen resultado ya que nos mostrará dos tipos de menú:



Para los no autenticados y



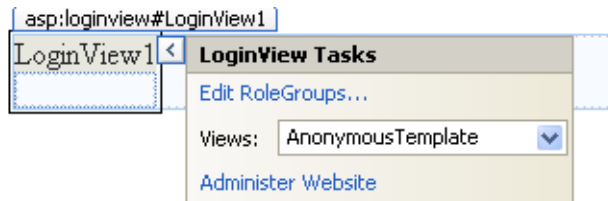
Perfecto, con dos líneas hemos conseguido además crear menús dinámicos. Imagina esta opción con varios perfiles o roles distintos, las opciones serán absolutamente dinámicas sin tener que escribir ni una línea de

código dentro de las páginas Web.

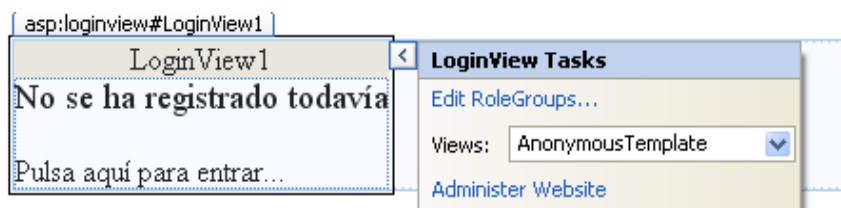
7. Control LoginView

Este control es similar al "PanelView" o al "MultiView" que vimos capítulos atrás, aunque este es mas sencillo, ya que no tenemos vistas para elegir. Lo que nos va a mostrar es el usuario que ha hecho Login, así, si lo colocamos en una parte de la pantalla nos mostrará como información si el usuario está registrado o no.

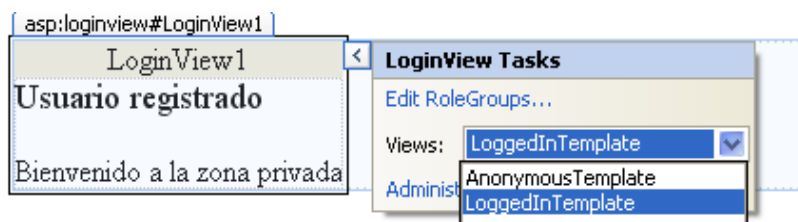
Veamos su uso con un sencillo ejemplo. Crea una página en blanco y pon un control de este tipo:



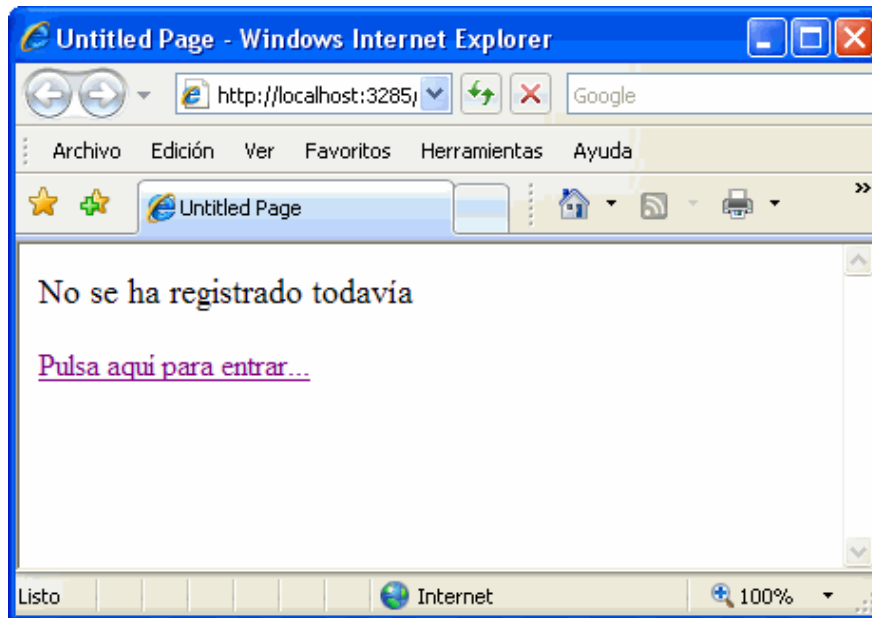
Tenemos dos vista que puedes elegir en "Views", donde nos mostrará el texto que queramos si el usuario no se ha registrado todavía:



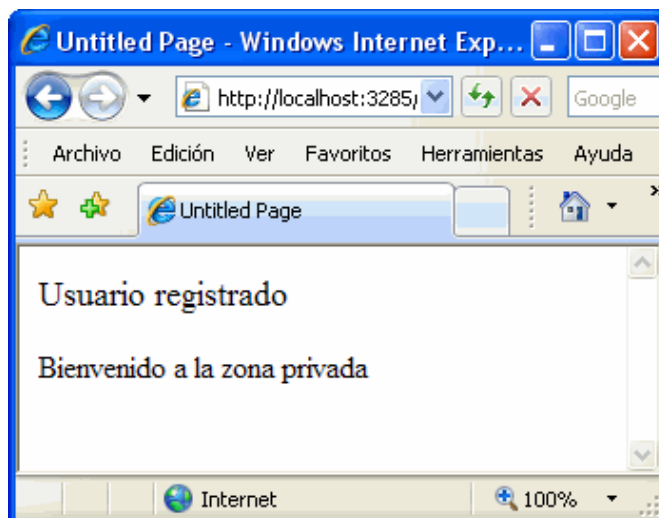
El texto de la segunda línea es un control de tipo hipervínculo que lleva a la página de login.aspx. Y el texto que queremos cuando se ha registrado:



Si ejecutamos la página:



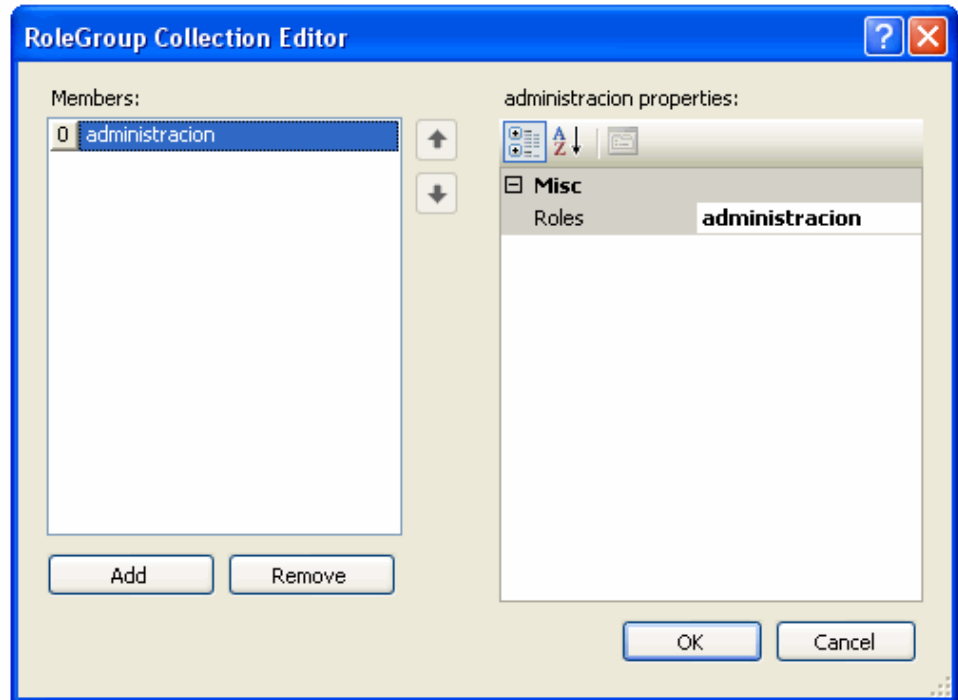
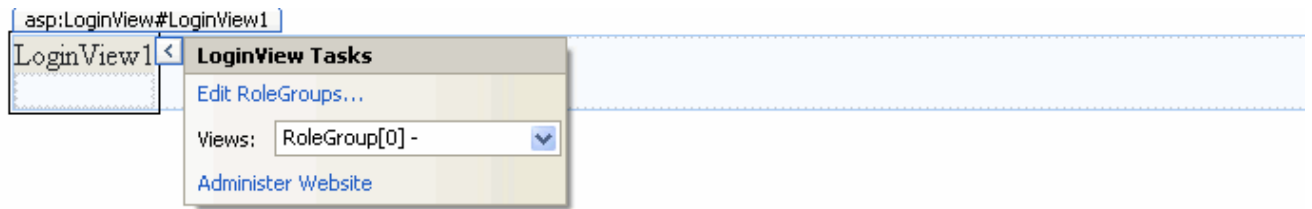
Y nos registramos mediante la pantalla de login que aparece al pulsar el enlace:



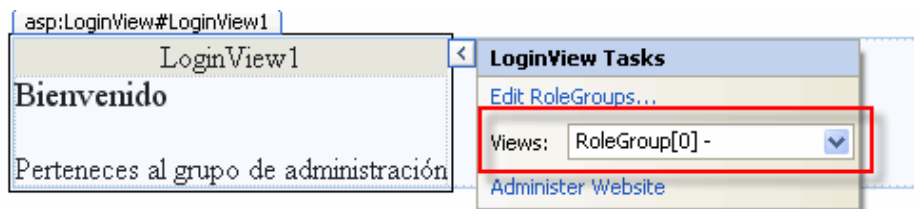
7.1 Información de roles

Podemos completar este control poniendo además la información sobre los roles en los que está el usuario que ha hecho login. Por ejemplo, pulsa en la opción de "Edit RoleGroups":

Seguridad en ASP.NET



Donde podremos tener mas plantillas según los roles que añadamos aquí. Por ejemplo hemos añadido el rol de "administracion". Y ahora podemos editar la plantilla para cuando sea de este rol:



En el código fíjate como queda:

Seguridad en ASP.NET

```
<asp:LoginView ID="LoginView1" runat="server">
  <AnonymousTemplate>
    <span class="style1">No se ha registrado todavía</span><br class="style1" />
    <br />
    <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="login.aspx">Pulsa
    aquí para entrar...</asp:HyperLink>
  </AnonymousTemplate>

  <RoleGroups>
    <asp:RoleGroup Roles="administracion">
      <ContentTemplate>
        <span class="style1">Bienvenido</span><br />
        <br />
        Pertenece al grupo de administración
      </ContentTemplate>
    </asp:RoleGroup>
    <asp:RoleGroup Roles="Usuarios, Invitados">
      <ContentTemplate>
        <span class="style1">Bienvenido</span><br />
        <br />
        Pertenece al grupo de usuarios e invitados
      </ContentTemplate>
    </asp:RoleGroup>
  </RoleGroups>
</asp:LoginView>
```

Hay dos grupos uno para cuando es "anonymoustemplate" que es cuando no se ha registrado y otro para los tipos de roles. Puesto que todos los usuarios pertenecen a un rol hemos quitado la plantilla de antes de cuando los usuarios se habían registrado y hemos puesto dos plantilla según el rol del usuario.

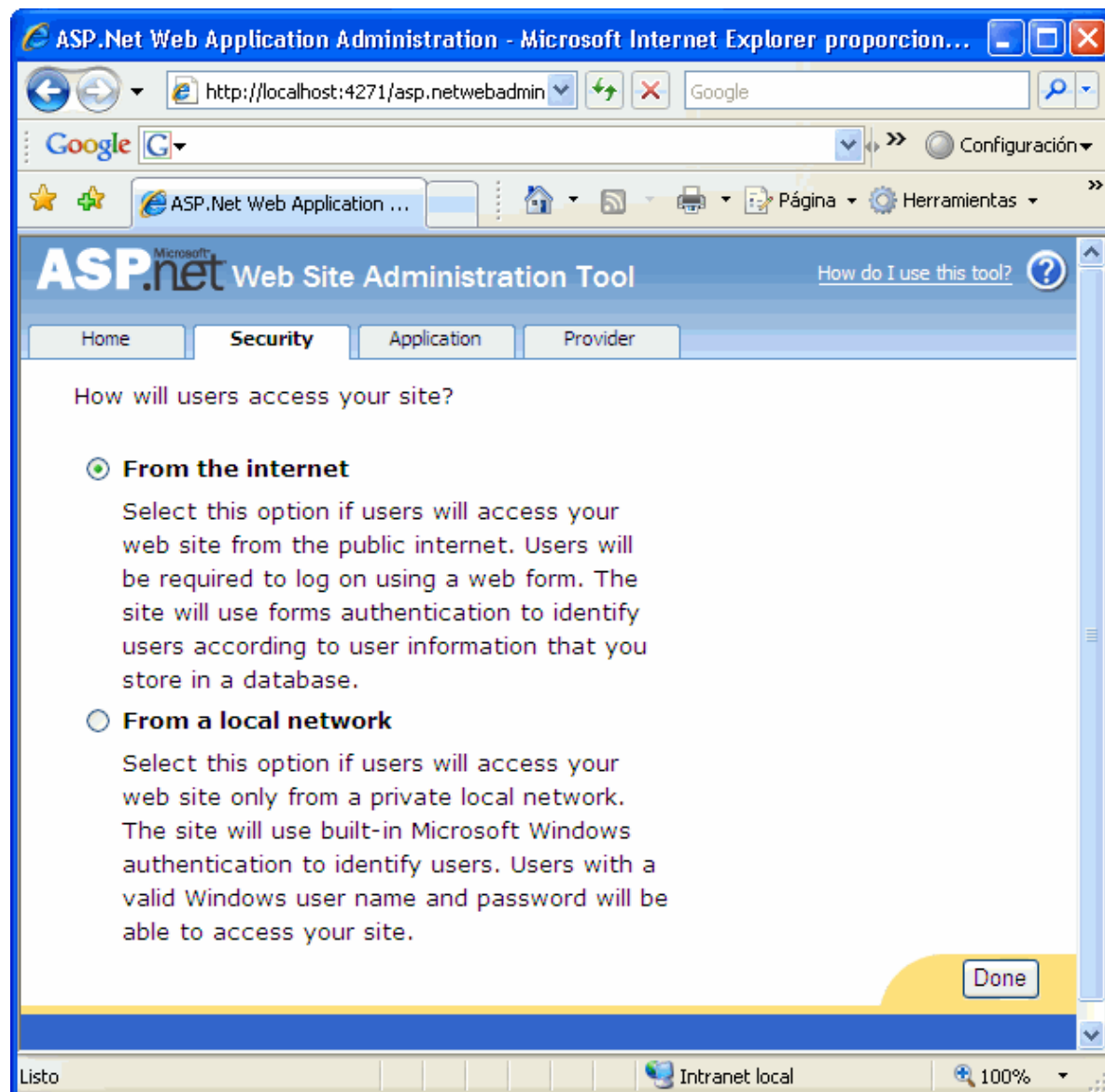
[Pulsa aquí para descargar los ejemplos de este tema](#)

Ejercicios

Partimos de las páginas de ejemplo de hace dos capítulos donde había un par de menús con enlaces a varios departamentos.

Ejercicio 1

Activa la seguridad con formularios y crea dos usuarios de prueba:



Pon un control de login que funcione con la página maestra y que en caso de validarse correctamente acceda a la página principal "default.aspx".

ASP.NET

Web Site Administration Tool

How do I use this tool?

Home

Security

Application

Provider

Use this page to manage access rules for your Web site. Rules are applied in order. The first rule that matches applies, and the permission in each rule overrides the permissions in all following rules. Use the **Move Up** and **Move Down** buttons to change the order of the selected rule.

Rules that appear dimmed are inherited from the parent and cannot be changed at this level.

Manage Access Rules

tema_13

App_Data



Compras

Direccion

Informatica

Publico

Ventas

Permission	Users and Roles	Delete
Deny	 [anonymous]	Delete
Allow	 [all]	Delete

[Add new access rule](#)

Move Up

Move Down

Done

62

Página de inicio de MiEmpresa.Com

[Iniciar sesión](#)

Menú de opciones

- Inicio
- Compras
- Dirección
- Informática
- Ventas
- Público

Mas opciones

Iniciar sesión

Nombre de usuario:

Contraseña:

☐ Recordármelo la próxima vez.

[Inicio de sesión](#)

Muestra los datos del usuario que ha hecho login y pon una opción para cerrar la sesión volviendo a la página de login:

Página de inicio de MiEmpresa.Com

Inicio
jose [Cerrar sesión](#)

Menú de opciones

- Inicio
- Compras
- Dirección
- Informática
- Ventas
- Público

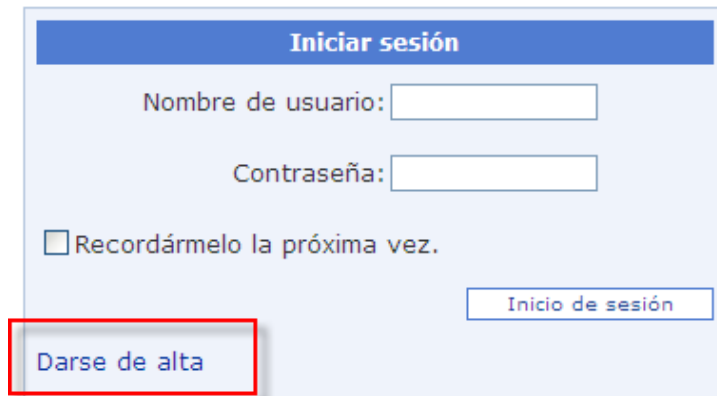
Mas opciones

**Bienvenidos
a la página de inicio de
MIEMPRESA.COM**

Ejercicio 3

En la pantalla de login permite que se puedan dar de alta los usuarios.

Seguridad en ASP.NET



The image shows a web form titled "Iniciar sesión" (Log in) in a blue header bar. Below the header, there are two text input fields: "Nombre de usuario:" (Username) and "Contraseña:" (Password). Under the password field is a checkbox labeled "Recordármelo la próxima vez." (Remember me next time). To the right of the checkbox is a button labeled "Inicio de sesión" (Log in). In the bottom left corner, there is a button labeled "Darse de alta" (Register), which is highlighted with a red rectangular border.

Basta con poner las propiedades adecuadas en el control de login y luego crear una página. Puesto que está prohibido el acceso anónimo no podríamos acceder a esta página de creación de usuario así que cambiaremos los permisos para prohibir el acceso anónimo a las todas las carpeta excepto la pública. Por tanto el usuario anónimo tendrá acceso a todo lo que no hayamos prohibido: la raíz del web y la carpeta "publico".

Por tanto al llegar a la página predeterminado, el usuario intentará hacer login y de allí se podrá dar de alta:

The screenshot shows a Microsoft Internet Explorer window with the title "Página sin título - Microsoft Internet Explorer proporcionado por Metzeler APS Ibérica S.A.". The address bar displays "http://localhost:4225/tema_13/alta.aspx". The page content includes a header "Página de inicio de MiEmpresa.Com" with a link "Iniciar sesión". On the left, there is a "Menú de opciones" with links to Inicio, Compras, Dirección, Informática, Ventas, and Publico, and a "Mas opciones" section with an "Inicio" button and a house icon. On the right, there is a "Regístrese para obtener una nueva cuenta" form with fields for "Nombre de usuario" (felipe), "Contraseña" (masked), "Confirmar contraseña" (masked), "Correo electrónico" (felipe@mimail.com), "Pregunta de seguridad" (nada), and "Respuesta de seguridad" (nada). A "Crear usuario" button is at the bottom right of the form. The status bar at the bottom shows "Listo" and "Intranet local".

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

http://localhost:4225/tema_13/alta.aspx

Google

Google

Ir Configuración

Página sin título

Página Herramientas

Página de inicio de MiEmpresa.Com


Iniciar sesión

Menú de opciones

- Inicio
- Compras
- Dirección
- Informática
- Ventas
- Publico

Mas opciones

Inicio ▶



Regístrese para obtener una nueva cuenta

Nombre de usuario: felipe

Contraseña:

Confirmar contraseña:

Correo electrónico: felipe@mimail.com

Pregunta de seguridad: nada

Respuesta de seguridad: nada

Crear usuario

Listo Intranet local 100%

Comprobar que ya puede entrar a las páginas restringidas:



Ejercicio 4

Crea un rol para dos usuario de prueba que al hacer login tengan acceso a la sección de Informática, sino no podrán acceder a ella. Por tanto determinado rol tendrá acceso a esa carpeta, los demás no.

Activamos los roles:

Users	Roles	Access Rules
Existing users: 3 Create user Manage users Select authentication type	Existing roles: 1 Disable Roles Create or Manage roles	Create access rules Manage access rules

Creamos uno:

Seguridad en ASP.NET







Create New Role		
New role name:	<input type="text"/>	<input type="button" value="Add Role"/>

Role Name	Add/Remove Users
administradores	Manage Delete

Añadiremos los usuarios para ese rol:

User	Roles
User ID: <input type="text" value="jose"/>	Select roles for this user: <input checked="" type="checkbox"/> administradores
* E-mail address: <input type="text" value="jose@mail.com"/> <input checked="" type="checkbox"/> Active user	
Description: <input type="text" value="[not set]"/> <input type="button" value="Save"/>	
(*) Required field	

Y creamos una regla de acceso para el grupo de "administradores":

Manage Access Rules			
	Permission	Users and Roles	Delete
 tema_13 <ul style="list-style-type: none"> App_Data Compras Direccion Informatica Publico Ventas	Deny	 [anonymous]	Delete
	Allow	 administradores	Delete
	Deny	 [all]	Delete
	Allow	 [all]	Delete
	Add new access rule		

Y comprobaremos que los usuarios del rol de administración tienen acceso a la sección de informática y los demás no.

Nota: cuando te descargues las soluciones ten en cuenta que está como un sitio web nuevo para así poder tener su propia bbdd y ficheros xml para el menú