Tutorial de Membresía y Roles

Este documento le servirá como una guía para hacer uso de las funciones de Membresía y roles de ASP.NET. Dicho documento se complementa por el proyecto adjunto **EjemploMembresia.zip**. **Para ejecutar el proyecto necesita seguir los pasos de las primeras tres secciones**; el resto ya ha sido añadido al proyecto. El script de creación de la BD se encuentra en *Modelos/script-creacion-bd.sql*

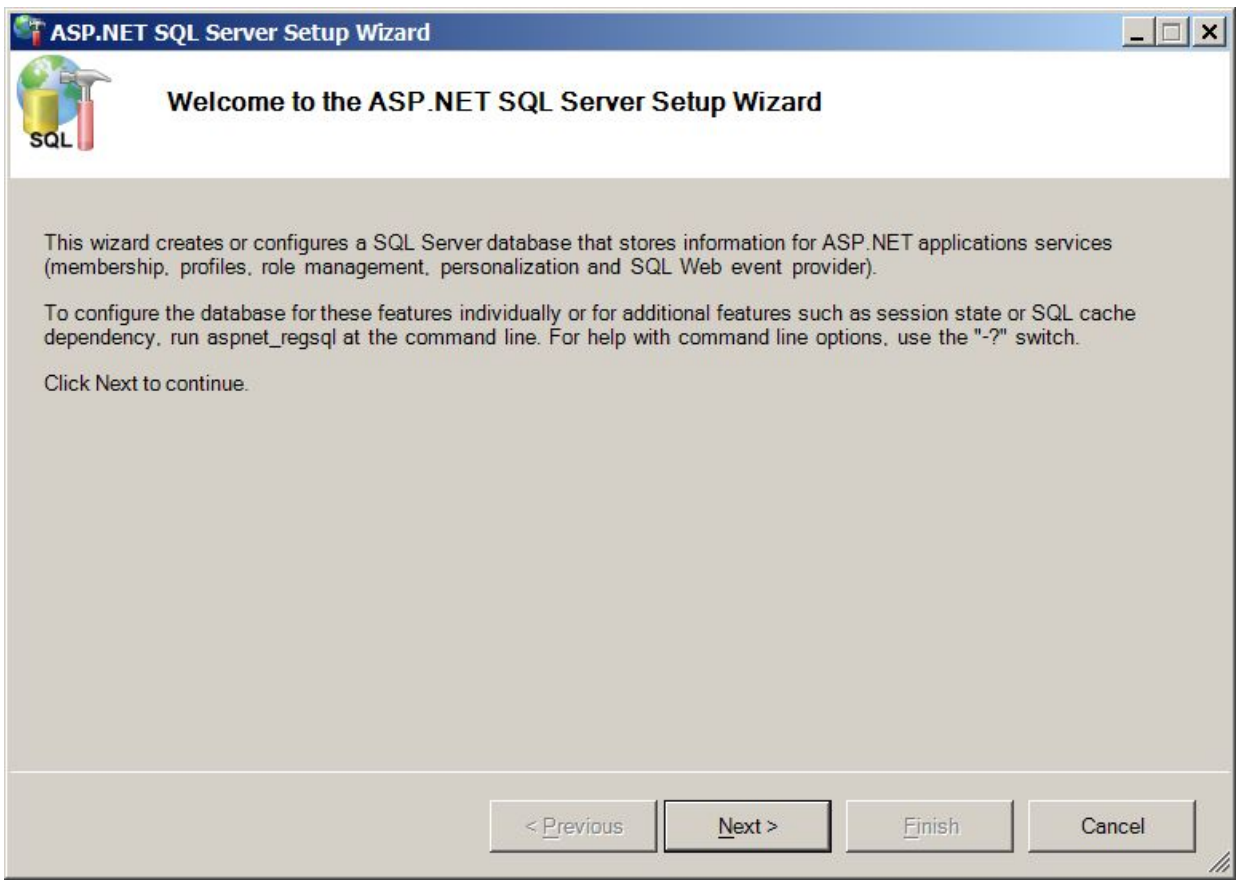
Requerimientos

- Tener una Base de Datos en SQL Server ya creada y desplegada
- Tener un proyecto de ASP.NET Web Forms operativo.

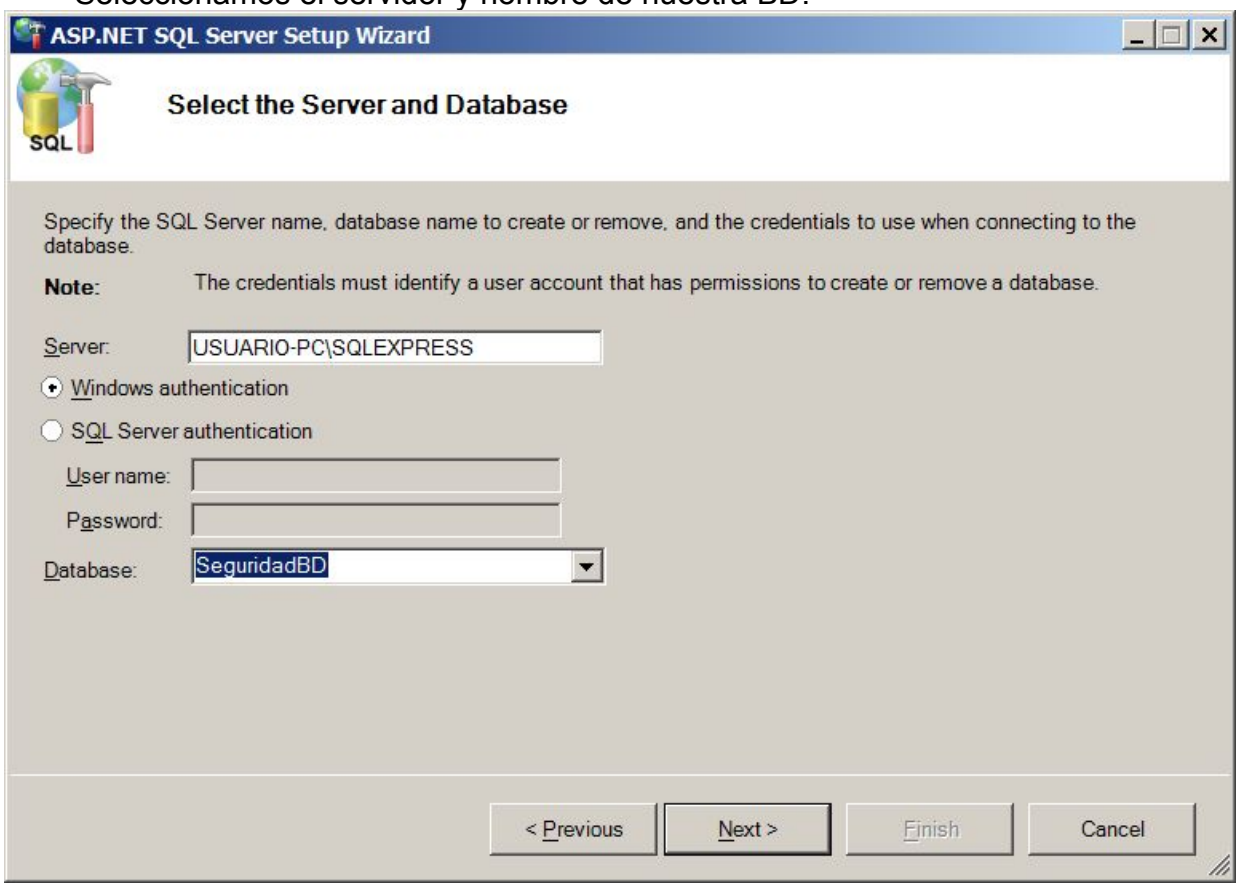
Preparación de nuestra BD para la membresía

Las funciones de Membresía y Roles de ASP.NET requieren un conjunto de tablas, vistas y procedimientos para poder funcionar. Con el fin de poder usar dichas funciones con nuestra BD de datos existente primero tenemos que prepararla. Una vez que tenga su BD creada y desplegada con SQL Server Management Studio (o con la herramienta de su preferencia), siga los siguientes pasos:

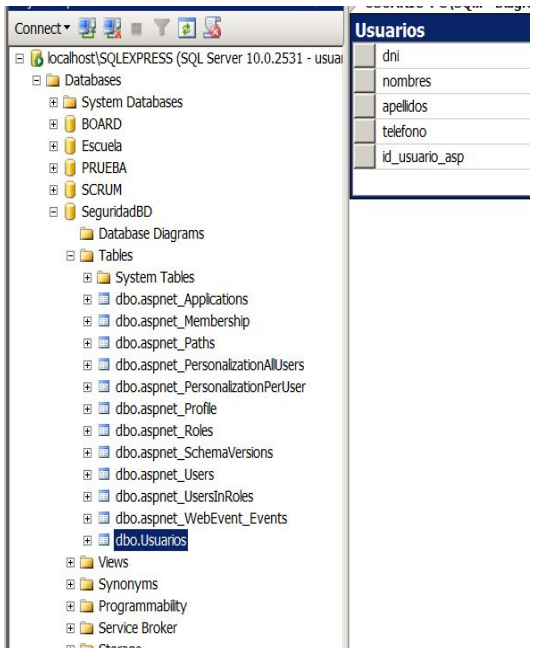
1. Accedemos al siguiente programa en Inicio→ejecutar:
   %WINDIR%\Microsoft.Net\Framework\v4.0.30319\aspnet_regsql.exe
   Si tenemos otra versión del Framework utilizar dicha carpeta
2. Nos debe aparecer la siguiente interfaz:



3. Siguiente→ *Configure SQL Server for application services* → Siguiente→ Seleccionamos el servidor y nombre de nuestra BD:



4. Le damos siguiente hasta finalizar. Luego abrimos nuestra BD en SQL Server Management Studio o la herramienta de nuestra preferencias y podremos ver de que se nos han creado múltiples tablas en adición a la nuestra. En el caso de esta BD, solo tenemos una tabla propia: dbo.Usuarios

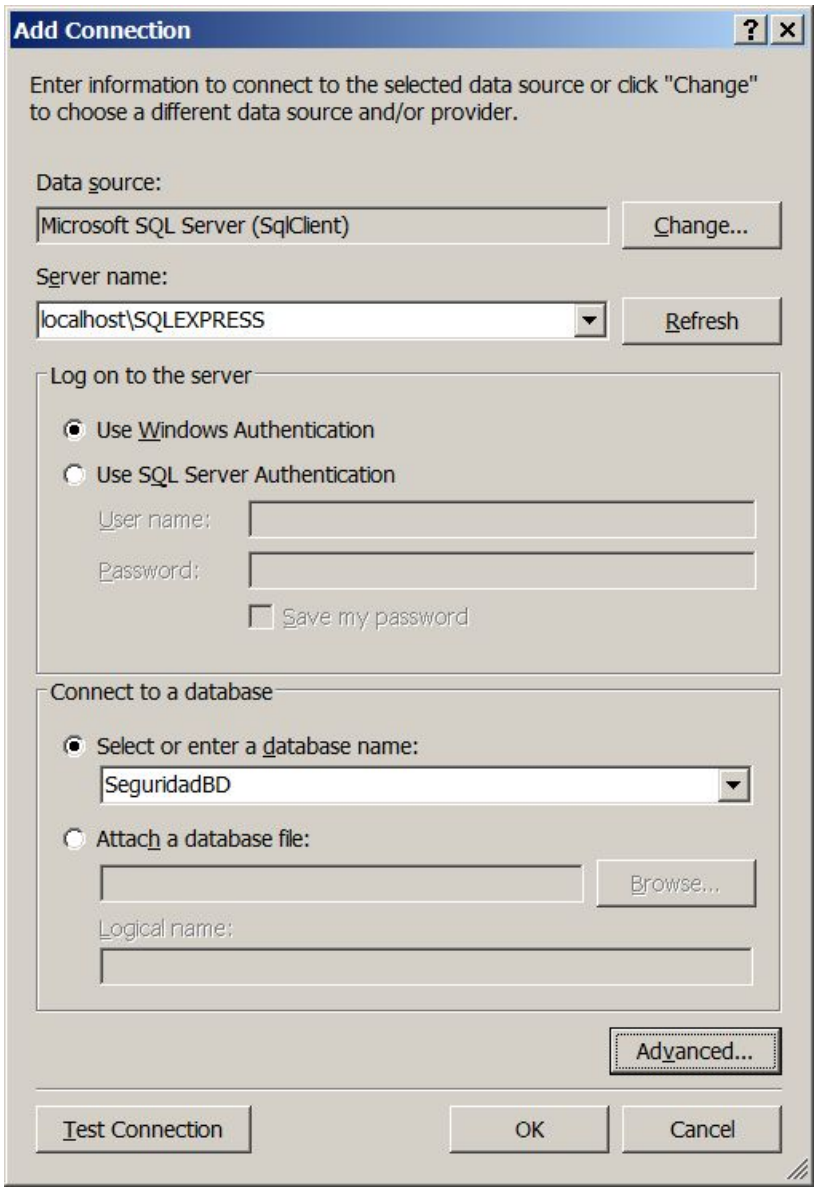


5. (Opcional) podríamos ahora crear una relación (FK) entre nuestra tabla Usuarios y la tabla aspnet_Users

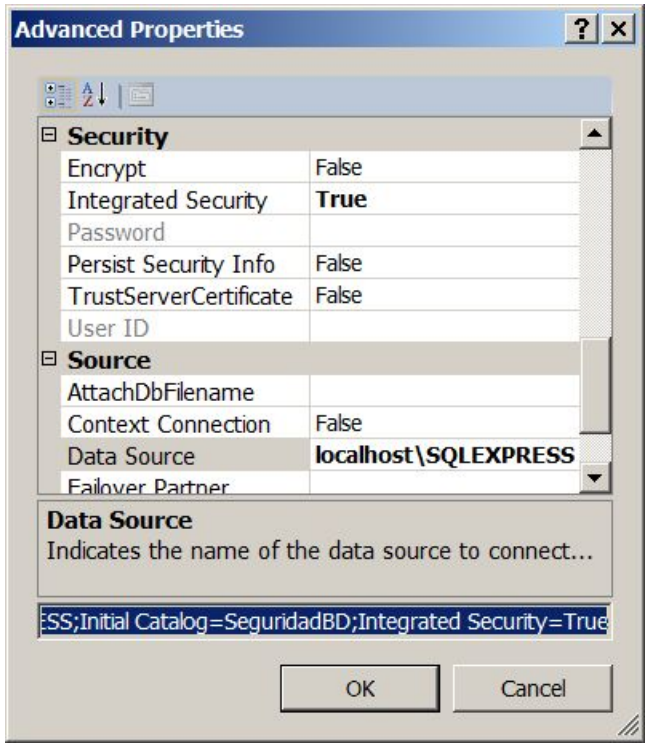
Obteniendo la cadena de conexión a nuestra BD

Como queremos usar nuestra BD existente tenemos que especificar la cadena de conexión de dicha BD en el **web.config.** Para obtener fácilmente la cadena de conexión podemos hacer lo siguiente:

1. En VS2010→View→ Server Explorer (Ctrl+Alt+S)
2. En el Server Explorer → Clic derecho a Data Connections → Add connection… → Microsoft SQL Server → Ingresamos los datos de nuestra BD de manera quede similar a esto:



3. Le damos a Test Connection para probar la conexión. Si está correcta, vamos a Advanced… y seleccionamos la cadena de conexión que se encuentra en el último Textbox:



4. Nos copiamos dicha cadena y la guardamos en algún sitio. Luego podemos darle cancelar a los diálogos que habíamos abierto.

Indicando la cadena de conexión a usar para nuestro proveedor de Membresía

1. Vamos al archivo **web.config** de nuestro proyecto. Si creamos un proyecto vacío deberíamos tener algo así:

```
<?xml version="1.0"?>

<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->

<configuration>
    <system.web>
        <compilation debug="true" targetFramework="4.0" />
    </system.web>

</configuration>
```

2. Ahora lo modificamos para que quede similar a esto:

```
<?xml version="1.0"?>

<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->

<configuration>
  <connectionStrings>
    <add name="CadenaConexionEjemploMembresia"
     connectionString="Data Source=localhost\SQLEXPRESS;Initial Catalog=SeguridadBD;Integrated Security=True"
     providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
        <compilation debug="true" targetFramework="4.0" />
    </system.web>

</configuration>
```

3. En el atributo *connectionString*=… deberíamos pegar la cadena de conexión que obtuvimos en el paso anterior.
4. Vale añadir que esta cadena de conexión puede ser accedida desde nuestra aplicación con la siguiente sintaxis

Implementado nuestro proveedor de Membresía

ASP.NET por defecto crea su propia BD cuando usamos Membresía y, para comunicarse con dicha BD, utiliza una versión **por defecto** del objeto del tipo *SQLMembershipProvider*. Como nosotros queremos usar **nuestra** BD tenemos que crear nuestro propio *Provider.* En este *Provider* podemos especificar distintas configuraciones de seguridad para nuestra aplicación, como el mínimo de caracteres que debería tener una contraseña (*minRequiredPasswordLength),* si los usuarios deben proveer un e-mail único (*requiresUniqueEmail*), etc.. Puede encontrar más detalles sobre cada atributo en: https://msdn.microsoft.com/es-pe/library/f1kyba5e.aspx#Anchor_0

1. Ahora añadimos el código que está seleccionado a nuestro **web.config**, de manera que quede así:

```
<configuration>
  <connectionStrings>
    <add name="CadenaConexionEjemploMembresia"
     connectionString="Data Source=localhost\SQLEXPRESS;Initial Catalog=SeguridadBD;Integrated Security=True"
     providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
    <membership defaultProvider="EjemploMembresiaProveedorSQL">
      <providers>
        <!-- Nuestro proveedor personalizado -->
        <add name="EjemploMembresiaProveedorSQL"
             type="System.Web.Security.SqlMembershipProvider"
             connectionStringName="CadenaConexionEjemploMembresia"
             enablePasswordRetrieval="false"
             enablePasswordReset="true"
             requiresQuestionAndAnswer="false"
             applicationName="EjemploMembresia"
             requiresUniqueEmail="false"
             passwordFormat="Hashed"
             maxInvalidPasswordAttempts="5"
             minRequiredPasswordLength="7"
             passwordStrengthRegularExpression=""/>
      </providers>
    </membership>
    <compilation debug="true" targetFramework="4.0" />
  </system.web>
</configuration>
```

2. Las propiedades más importantes a tener en cuenta son **defaultProvider, name, connectionStringName y applicationName.**

Activando roles

Hasta este punto ya nuestra aplicación está lista para crear y modificar usuarios, sin embargo, todavía no podemos aplicar roles: es decir, privilegios distintos según el **rol** o **tipo** de usuario. Para poder usar roles también tenemos que especificar un **Proveedor de Roles**, similar al proveedor de Membresía. Siga los siguientes pasos:

1. Añadimos el siguiente código seleccionado a nuestro **web.config**

```
<system.web>
  <membership defaultProvider="ProveedorMembresiaSQL">
    <providers>
      <!-- Nuestro proveedor personalizado -->
      <add name="ProveedorMembresiaSQL"
           type="System.Web.Security.SqlMembershipProvider"
           connectionStringName="CadenaConexionEjemploMembresia"
           enablePasswordRetrieval="false"
           enablePasswordReset="true"
           requiresQuestionAndAnswer="false"
           applicationName="EjemploMembresia"
           requiresUniqueEmail="false"
           passwordFormat="Hashed"
           maxInvalidPasswordAttempts="5"
           minRequiredPasswordLength="7"
           passwordStrengthRegularExpression=""/>
    </providers>
  </membership>
  <compilation debug="true" targetFramework="4.0" />
  <roleManager enabled="true" defaultProvider="ProveedorRolesSQL">
    <providers>
      <add name="ProveedorRolesSQL"
           type="System.Web.Security.SqlRoleProvider"
           applicationName="EjemploMembresia"
           connectionStringName="CadenaConexionEjemploMembresia" />
    </providers>
  </roleManager>
</system.web>
```

2. Nuestra aplicación ya está lista para hacer uso de los roles y membresía

Métodos, propiedades y tablas de referencia

Autenticación y Membresía

bool *Membership.ValidateUser*(string *nombreUsuario*, string *contraseña*)

- Valida un usuario de membresía, retorna verdadero si las credenciales son correctas.
- Debería llamarse desde una página Login. Ejemplo: Login.aspx

void *FormsAuthentication.RedirectFromLoginPage*(string *nombreUsuario*, bool recordar)

- Redirige al usuario indicado al recurso solicitado. Ejemplo: si quiso acceder a *Mantenimiento.aspx* pero tuvo que loguearse primero, una vez que su login fue exitoso lo redirige a *Mantenimiento.aspx*.
- El segundo parámetro indica si se debería recordar al usuario en el próximo acceso al sitio.
- Debería llamarse desde una página Login. Ejemplo: Login.aspx

string *User.Identity.Name*

- Retorna el nombre del usuario, una vez autenticado. *User.Identity* tiene otras propiedades útiles.

MembershipUser *Membership.GetUser(*string *nombreUsuario)*

- Retorna al usuario indicado, si no existe retorna nulo

MembershipUser *Membership.CreateUser(*string *nombreUsuario,* string *contraseña)*

- Crea al usuario con las credenciales especificadas y retorna al usuario si la creación fue exitosa

bool *Membership.DeleteUser(*string *nombreUsuario)*

- Borra al usuario indicado**.** Retorna verdadero si se borró.

Roles

bool *Roles.RoleExists(*string *nombreRol)*

- Indica si el rol existe

void *Roles.CreateRole(*string *nombreRol)*

- Crea el rol indicado

bool *Roles.DeleteRole(*string *nombreRol)*

- Borrar el rol indicado, retorna verdadero si fue exitoso.

void *Roles.AddUserToRole*(string *nombreUsuario,* string *nombreRol)*

- Añade al usuario indicado al rol indicado

string[] *Roles.GetRolesForUser()*

- Devuelve un arreglo de los roles que tiene asignados el usuario que ha iniciado sesión. Recuerde que un usuario puede tener múltiples roles.

Tablas

- **aspnet_Users:** tabla de los usuarios de membresía
- **aspnet_UsersInRoles:** tabla de los usuarios de membresía con sus roles
- **aspnet_Roles:** tabla de los roles existentes