Michiel van Otegem is werkzaam bij The Vision Web als Lead Architect

Doe meer met ASP.NET 2.0 Membership en Role Management

BEVEILIGDE APPLICATIES MET MEMBERSHIP PROVIDERS EN ROLE MANAGEMENT PROVIDERS

ASP.NET 2.0 biedt met het Membership en Role Manager-systeem een handige basis voor het maken van beveiligde applicaties. Hoewel die basis in veel gevallen voldoet, kan het nodig zijn om verder te gaan dan wat ASP.NET 2.0 'out-of-the-box' verzorgt.

Om een webapplicatie te beveiligen kun je gebruikmaken van het Membership en Role Management-systeem van ASP.NET 2.0. Dit systeem stelt je in staat om met weinig inspanning (gedeeltes van) een site te beveiligen, omdat er sprake is van een standaard API en daaraan gekoppelde servercontrols. Dit is in *Autorisatie en authenticatie met ASP.NET 2.0 en ADAM* (.NET Magazine #9) en *Inloggen met ASP.NET 2.0* (.NET Magazine #10) al uitgebreid aan de orde gekomen.
Een van de grote voordelen van het Membership en Role Management-systeem is dat de API's zijn opgezet volgens het Provider Model Design Pattern. Dit design pattern scheidt de API van de implementatie daarvan, waardoor code die gebruikmaakt van de API niet beïnvloed wordt door de keuze voor een andere implementatie, ook wel een provider genoemd. Dit is ook waarom servercontrols zoals de Login-control zonder aanpassing blijven werken, ongeacht de gekozen provider.
Standaard wordt ASP.NET 2.0 geleverd met Membership Providers voor SQL Server en Active Directory, en Role Management Providers voor SQL Server, Authorization Store, en Windows-tokens. Wanneer welke provider van toepassing is, zie je in tabel 1. Zoals je kunt zien is de Membership Provider voor Active Directory van toepassing op zowel een Active Directory die gebruikt wordt voor Windows-domeinen, als Active Directory in Application Mode (ADAM). Wanneer je Role Management hierop wilt baseren, gaat dit via de Authorization Store Provider die samenwerkt met Authorization Manager (AzMan). AzMan biedt zonodig ook de mogelijkheid gebruik te maken van een XML-bestand, hoewel dat wel het schema moet aanhouden dat AzMan hanteert. Je kunt niet een eigen XML-formaat gebruiken, tenzij je zelf een provider implementeert (later meer hierover).

Afbeelding 1. Provider class-hiërarchie

Merk op dat in het geval van Windows Security er geen Membership Provider gebruikt wordt. Deze functionaliteit wordt immers door Internet Information Server verzorgd, hetzij via Integrated Security, hetzij via Basic of Digest Authentication. De WindowsTokenRoleProvider is in dat geval te gebruiken voor Role Management, omdat de token van de gebruiker de groepen bevat waartoe de gebruiker behoort. Met deze provider kun je geen nieuwe groepen definiëren en ook niet aanpassen welke groepen tot een gebruiker behoren. Dit omdat het token per definitie read-only is. Bij de overige Role Management-providers is het afhankelijk van de instellingen van de onderliggende gegevensbron of groepsinstellingen en dergelijke gewijzigd kunnen worden.

Naast de meegeleverde providers zijn er ook third-party providers voor onder andere Oracle en MySQL (zie referenties). Je kunt ook zelf providers schrijven door te erven van de juiste (abstracte) base class, waardoor het een (wat uitgebreide) invuloefening is. Zoals je kunt zien in de class-hiërarchie in afbeelding 1 stammen alle providers van ProviderBase. Dit is een base class die de basis biedt voor het Provider Model Design Pattern. Deze kun je dus gebruiken om zelf iets te bouwen op basis van dit design pattern (zie *Build Your Own ASP.NET Provider-Based Feature* op MSDN TV: http://msdn.microsoft.com/msdntv/episode.aspx?xml=episodes/en/20060518aspnetss/manifest.xml). Wil je een eigen Membership Provider maken, dan moet je erven van System.Web.Security.MembershipProvider, zoals de meegeleverde providers doen.

Roles apart opslaan

Mocht je het vreemd vinden dat Membership en Role Management niet in één beveiligingssysteem gecombineerd zijn, dan geeft tabel 1

	Membership	Role Management
SQL Server	SqlServerMembership Provider	SqlServerRoleProvider
Active Directory	ActiveDirectoryMember shipProvider	AuthorizationStoreRoleProvider
ADAM	ActiveDirectoryMember shipProvider	AuthorizationStoreRoleProvider
XML	Niet beschikbaar	AuthorizationStoreRoleProvider
Windows Security	Geen (IIS Security)	WindowsTokenRoleProvider

Tabel 1. Providers voor verschillende situaties

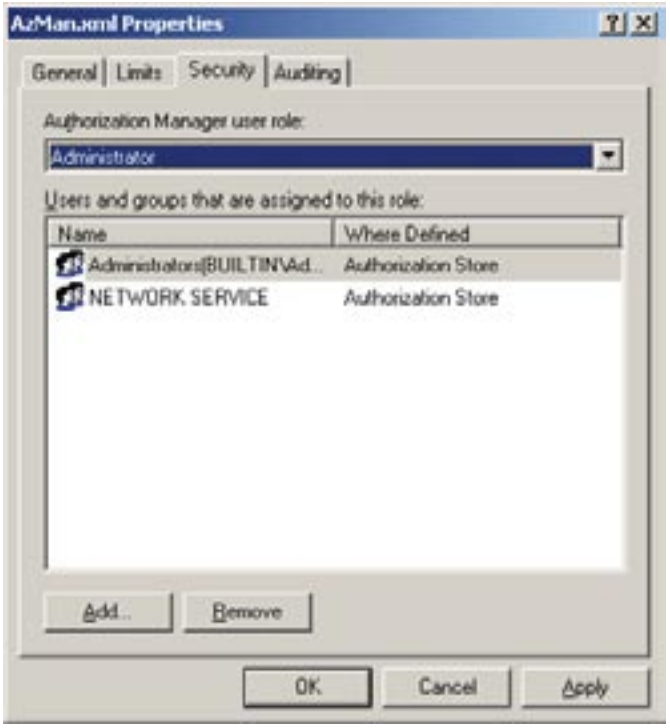
Afbeelding 2. Authorization Store aanmaken in Azman

een hint waarom dit niet zo is. De scheiding tussen deze twee functionaliteiten biedt de mogelijkheid verschillende providers te gebruiken die samen de beveiliging verzorgen. Dit is bijvoorbeeld handig als in een organisatie single sign-on gerealiseerd moet worden op basis van de corporate Active Directory, maar deze Active Directory niet 'bevuild' mag worden met applicatiespecifieke beveiliging. In dat geval kunnen de groepsgegevens bijvoorbeeld in een applicatiespecifieke database opgeslagen worden, of via AzMan in ADAM of een XML-bestand. Codevoorbeeld 1 laat de configuratie zien voor Windows Security met een AzMan XML-bestand voor Role Management.

Voordat je AzMan kunt gebruiken - zoals geconfigureerd in codevoorbeeld 1 - moet je AzMan zelf nog wel instellen. Hiervoor moet je de AzMan Management Console starten door azman.msc in te typen in het Run… window. Standaard staat AzMan ingesteld op Administrator Mode, maar dan kun je geen authorization store aanmaken. Er moet echter een XML authorization store aangemaakt worden en dat kan als je in developer mode werkt. Ga naar Options in het Action-menu en maak een nieuwe authorization store via New Authorization Store. Hierbij moet je kiezen voor een XML-file en het gewenste pad opgeven, zoals in afbeelding 2. In de treeview aan de linkerkant verschijnt nu het aangemaakte XML-bestand. Dat moet toegankelijk gemaakt worden voor de webapplicatie. Normaal gesproken betekent dit dat het account waaronder ASP.NET uitgevoerd wordt (ASP.NET voor IIS5 en Network Service voor IIS6) toegang moet hebben. Als je gebruikmaakt van impersonation moeten alle gebruikers van de website (lees)rechten hebben. Een gebruiker kan als Administrator of als Reader toegevoegd

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <connectionStrings>
    <add name="AzManXml"
         connectionString="msxml://D:\NetMagazine\AspNet20Security\Data\
AzMan.xml" />
  </connectionStrings>
  <system.web>
    <authentication mode="Windows" />
    <roleManager defaultProvider="NetMagazineRoleProvider" enabled="true" >
      <providers>
        <add name="NetMagazineRoleProvider"
             type="System.Web.Security.AuthorizationStoreRoleProvider"
             connectionStringName="AzManXml"
             applicationName="NetMagazineAspNet20Demo" />
      </providers>
    </roleManager>
  </system.web>
</configuration>
```

Codevoorbeeld 1. Web.config voor AzMan Role Management



Afbeelding 3. Rechten op de Authorization Store aanpassen

worden aan de authorization store. In het laatste geval kan vanuit de applicatie niets gewijzigd worden aan de rechteninstellingen. In afbeelding 3 is het Network Service-account toegevoegd als Administrator, zodat we groepen kunnen aanmaken en bewerken.
Zoals je misschien hebt opgemerkt verwijst in codevoorbeeld 1 de Role Manager-provider naar een specifieke applicatie. Het is mogelijk om in dezelfde authorization store, bijvoorbeeld in dezelfde SQL Server-database, beveiligingsinformatie op te slaan van verschillende applicaties. Standaard is dit de webroot, maar het kan ook een specifieke naam zijn zoals in codevoorbeeld 1. In AzMan moet je deze applicatie aanmaken en dat doe je via New Application in het Action-menu. Let er wel op dat op dat moment het XML-bestand geselecteerd is. Je krijgt dan het venster in afbeelding 4 te zien.
Nu alle instellingen voltooid zijn, kun je groepen definiëren en gebruikers toevoegen aan groepen door de volgende code uit te voeren:

```
Roles.CreateRole("Finance");
Roles.AddUserToRole("DemoDomain\Michiel", "Finance");
```

Het is daarbij wel van belang dat de gebruiker ook daadwerkelijk bestaat in het domein, want AzMan koppelt gebruikers niet op naam, maar op Security Identifier (SID). Als de gebruiker niet bestaat, is er geen geldige SID en resulteert het toevoegen van de gebruiker in een foutmelding. Dit betekent ook dat AzMan alleen te gebruiken is met Windows Security of als er met Active Directory of ADAM wordt gewerkt.

```
public static void GetEmailAddressesFromRole(string roleName)
{
  string[] usersInRole = Roles.GetUsersInRole(roleName);
  string[] emails = new string[usersInRole.Length];

  for(int i = 0;I < usersInRole.Length; i++)
  {
    MembershipUser user = Membership.GetUser(usersInRole[i]);
    emails[i] = user.Email;
  }

  return emails;
}
```

Codevoorbeeld 2. E-mailadressen opvragen van alle gebruikers in een groep

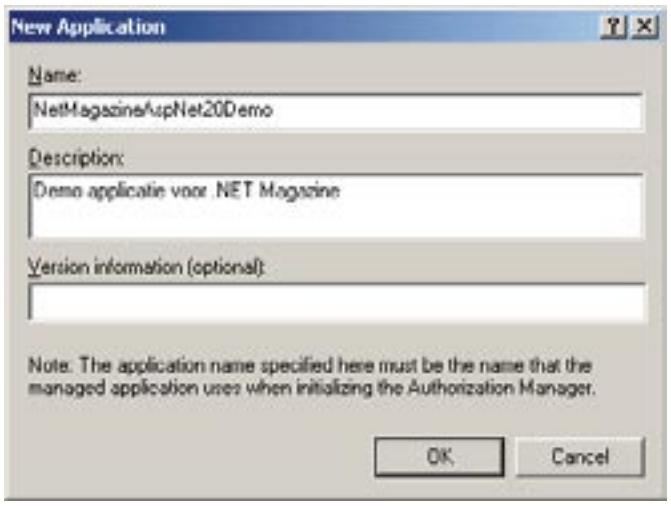
Scheiding tussen Membership en Role Management

Zoals al opgemerkt zijn het Membership en Role Management-systeem twee gescheiden API's. Normaal gezien is dat geen probleem, maar als de samenwerking wat intensiever wordt, kan dit problemen opleveren. Een mooi voorbeeld is wanneer je een e-mailnieuwsbrief wilt versturen naar een bepaalde groep gebruikers. Je zult in dat geval eerst moeten opvragen welke gebruikers in een groep zitten, wat een string array oplevert, en vervolgens de e-mailadressen van alle gebruikers erbij zoeken, zoals in codevoorbeeld 2.

Je hoeft geen genie te zijn om te begrijpen dat de code in codevoorbeeld 2 een kostbare operatie kan worden, omdat je voor iedere gebruiker de Membership.GetUser-methode moet aanroepen. Bij veel gebruikers loopt dat aardig in de papieren. Jammer genoeg is er geen methode op de Membership API waaraan je een array van gebruikersnamen kunt meegeven om een verzameling gebruikers op te vragen. Je zou eventueel wel alle gebruikers kunnen opvragen en dan controleren of de gebruikersnaam voorkomt in de array met gebruikersnamen. Met een grote hoeveelheid gebruikers is dat niet echt wenselijk. In deze situatie kun misschien als beste oplossing extensies schrijven die wel de nodige functionaliteit aanbieden. Al naar gelang de situatie kun je dit op verschillende manieren oplossen. Als je van tevoren weet dat je voor zowel de Membership Provider als de Role Provider dezelfde onderliggende gegevensbron gaat gebruiken, kun je direct tegen de onderliggende bron aan werken. Is dit niet het geval, dan kun je het beste een oplossing maken die ook weer op het Provider Model Design Pattern is gebaseerd. In die situatie moet je er rekening mee houden dat de hoeveelheid providers vrijwel exponentieel groeit met de hoeveelheid gegevensbronnen, tenzij je de extensie zo kunt implementeren dat die maar aan één van de twee gegevensbronnen is gekoppeld.

Een extensie maken

Hoe je een extensie ook opbouwt, kennis van de (interne) werking van de provider is erg handig. De Provider Toolkit (http://msdn.microsoft.com/asp.net/downloads/providers/default.aspx) is zeker de moeite waard, vooral de voorbeeldcode van de providers voor Microsoft Access. De providers voor de verschillende databases wijken namelijk niet gek veel af, behalve dat er databasespecifieke ADO.NET-objecten gebruikt worden en de SQL Server-providers gebruikmaken van stored procedures in plaats van queries. Helaas heb je daarmee nog geen inzicht in de werking van de providers voor Active Directory en AzMan, maar je zou eens een kijkje kunnen nemen met behulp van een tool als Reflector.
Voor het voorbeeld in dit artikel gaan we uit van een SQL Server-database waarbij we een methode willen maken die alle gebruikers uit een bepaalde rol retourneert. Deze methode wijkt



Afbeelding 4. Nieuwe applicatie aanmaken in een authorization store

```
CREATE PROCEDURE dbo.netmag_RolesExtension_GetUsersByRole
   @ApplicationName  nvarchar(256),
   @RoleName      nvarchar(256),
   @PageIndex     int,
   @PageSize      int
AS
BEGIN
   DECLARE @ApplicationId uniqueidentifier
   SELECT  @ApplicationId = NULL
   SELECT  @ApplicationId = ApplicationId FROM dbo.aspnet_Applications
   WHERE LOWER(@ApplicationName) = LoweredApplicationName
   IF (@ApplicationId IS NULL)
      RETURN 0

   -- Pagina berekenen (resultaat moet een specifieke pagina zijn)
   DECLARE @PageLowerBound int
   DECLARE @PageUpperBound int
   DECLARE @TotalRecords   int
   SET @PageLowerBound = @PageSize * @PageIndex
   SET @PageUpperBound = @PageSize - 1 + @PageLowerBound

   -- Id van de rol opzoeken
   DECLARE @RoleId uniqueidentifier
   SELECT  @RoleId = NULL

   SELECT  @RoleId = RoleId
   FROM dbo.aspnet_Roles
   WHERE   LOWER(@RoleName) = LoweredRoleName AND ApplicationId =
 @ApplicationId

   IF (@RoleId IS NULL)
      RETURN(1)

   -- Een TEMP tabel maken voor gebruikers in rol
   CREATE TABLE #PageIndexForUsers
   (
      IndexId int IDENTITY (0, 1) NOT NULL,
      UserId uniqueidentifier
   )

   -- Alle gebruikers in rol in TEMP tabel plaatsen
   INSERT INTO #PageIndexForUsers (UserId)
      SELECT u.UserId
      FROM    dbo.aspnet_Users u, dbo.aspnet_Membership m
      WHERE  u.ApplicationId = @ApplicationId AND m.UserId = u.UserId AND
 u.LoweredUserName IN
   (SELECT LOWER(u.UserName)
      FROM    dbo.aspnet_Users u, dbo.aspnet_UsersInRoles ur
      WHERE  u.UserId = ur.UserId AND @RoleId = ur.RoleId AND
u.ApplicationId = @ApplicationId)
      ORDER BY u.UserName

   -- Juiste pagina uit de TEMP tabel retourneren
   SELECT  u.UserName, m.Email, m.PasswordQuestion, m.Comment,
           m.IsApproved, m.CreateDate, m.LastLoginDate,
           u.LastActivityDate, m.LastPass wordChangedDate,
           u.UserId, m.IsLockedOut, m.LastLockoutDate
   FROM    dbo.aspnet_Membership m, dbo.aspnet_Users u, #PageIndexFor Users p
   WHERE   u.UserId = p.UserId AND u.UserId = m.UserId AND
           p.IndexId >= @PageLowerBound AND p.IndexId <= @PageUpperBound
   ORDER BY u.UserName

   -- Ook totaal aantal gebruikers retourneren
   SELECT @TotalRecords = COUNT(*)
   FROM   #PageIndexForUsers
   RETURN @TotalRecords
END
```

Codevoorbeeld 3. Stored Procedure voor het opvragen van gebruikers in een bepaalde rol

```
public static MembershipUserCollection GetUsersByRole(string roleName,
       int pageIndex, int pageSize, out int totalRecords)
{
   MembershipUserCollection userCollection = new MembershipUserCollec
tion();

   // Maak SQL connection en command voor uitvoeren stored procedure
   Sql Connection connection = new SqlConnection(GetConnection());
   SqlCommand command = new SqlCommand(
            "dbo.netmag_RolesExtension_GetUsersByRole", connection);
   command.CommandType = CommandType.StoredProcedure;
   command.Parameters.Add(CreateInputParam("@ApplicationName",
            SqlDbType.NVarChar, Membership.Provider.ApplicationName));
   command.Parameters.Add(CreateInputParam("@RoleName",
            SqlDbType.NVarChar, roleName));
   command.Parameters.Add(CreateInputParam("@PageIndex",
            SqlDbType.Int, pageIndex));
   command.Parameters.Add(CreateInputParam("@PageSize",
            SqlDbType.Int, pageSize));

   // Initializeer returnwaarde
   totalRecords = 0;
   SqlParameter returnValue = new SqlParameter("@ReturnValue", SqlDbType.Int);
   returnValue.Direction = ParameterDirection.ReturnValue;
   command.Parameters.Add(returnValue);

   // Database openen en gebruikers uitlezen
   connection.Open();
   SqlDataReader reader = command.ExecuteReader(CommandBehavior.SequentialAccess);
   while (reader.Read())
   {
      string userName = GetNullableString(reader, 0);
      string email = GetNullableString(reader, 1);
      string passwordQuestion = GetNullableString(reader, 2);
      string comment = GetNullableString(reader, 3);
      bool isApproved = reader.GetBoolean(4);
      DateTime creationDateGlobal = reader.GetDateTime(5);
      DateTime creationDate = creationDateGlobal.ToLocalTime();
      DateTime lastLoginDateGlobal = reader.GetDateTime(6);
      DateTime lastLoginDate = lastLoginDateGlobal.ToLocalTime();
      DateTime lastActivityDateGlobal = reader.GetDateTime(7);
      DateTime lastActivityDate = lastActivityDateGlobal.ToLocalTime();
      DateTime lastPasswordChangedDateGlobal = reader.GetDateTime(8);
      DateTime lastPasswordChangedDate = lastPasswordChangedDateGlobal.
           ToLocalTime();
      Guid userId = reader.GetGuid(9);
      bool isLockedOut = reader.GetBoolean(10);
      DateTime lastLockoutDateGlobal = reader.GetDateTime(11);
      DateTime lastLockoutDate = lastLockoutDateGlobal.ToLocalTime();

      userCollection.Add(new MembershipUser(Membership.Provider.Name,
           userName, userId, email, passwordQuestion, comment,
           isApproved, isLockedOut, creationDate, lastLoginDate,
           lastActivityDate, lastPasswordChangedDate, lastLockoutDate));
   }

   // Klaar, afsluiten, totaal en gebruikers retourneren
   reader.Close();
   totalRecords = (int)returnValue.Value;
   connnection.Close();
   return userCollection;
```

Codevoorbeeld 4. GetUsersByRole-methode van de extensie

niet veel af van de Membership.FindUsersByName-methode die een MembershipUserCollection retourneert met alle gebruikers waarvan de gebruikersnaam overeenkomt met de gegeven gebruikersnaam (in SQL-termen: Username LIKE @username). In de SQL Server Membership Provider roept deze methode de stored procedure dbo.aspnet_Membership_FindUsersByName aan. Als je gebruikmaakt van SQL Server Express wordt deze stored procedure automatisch aangemaakt in de ASPNETDB-database in de App_Code-map. Anders moet je aspnet_regsql.exe in de map van het .NET Framework uitvoeren om deze aan te maken in de door jouw gewenste database. Voor de door ons gewenste functionaliteit hebben we een andere stored procedure nodig, maar we kunnen gelukkig veel overnemen. Codevoorbeeld 3 laat de nieuwe stored procedure zien.

De stored procedure in codevoorbeeld 3 bestaat uit drie stappen. Eerst wordt de ID van de rol opgehaald aan de hand van de rolnaam. Dan worden de id's van de gebruikers in die rol opgevraagd en in een tijdelijke tabel gezet. Als laatste worden uit een combinatie van deze tabel met de aspnet_Membership tabel en aspnet_Users tabel de gebruikers opgevraagd, waarbij rekening wordt gehouden met het feit dat dit per pagina kan. De stored procedure geeft net als dbo.aspnet_Membership_FindUsersByName een resultset terug met rijen, waarin de gegevens staan die nodig zijn voor een MembershipUser object. Er moet daarom per rij een MembershipUser-object gemaakt en toegevoegd worden aan een MembershipUserCollection. Dat is het werk van Membership.FindUsersByName. Wat nu dus nog moet gebeuren is een nieuwe methode maken die daarop gebaseerd is. In die methode veranderen we de stored procedure en parameter (roleName in plaats van username). Dat is gedaan in de GetUsersByRole-methode in codevoorbeeld 4. Voor de duidelijkheid is alle foutafhandeling achterwege gelaten. Alle methodes in de Provider Toolkit zijn allemaal keurig uitgerust met controle van inputparameters en foutafhandeling.
Dit voorbeeld geeft uiteraard maar een glimp van wat er allemaal mogelijk is. De implementatie die hier is gekozen, is helaas gekoppeld aan het gebruik van SQL Server voor zowel Membership als Roles. Hoewel dit waarschijnlijk de meest voorkomende situatie is voor een internetapplicatie, is het niet de meest flexibele oplossing. Het is bijvoorbeeld handiger een stored procedure te maken die een lijst met gebruikernamen als parameter krijgt en voor die gebruikers de nodige gegevens retourneert. Daarmee is de functie alleen gekoppeld aan de Membership Provider. Deze zou je eventueel ook kunnen opzetten volgens het Provider Model Design Pattern. Nadeel hiervan is dat dit meer programmeerwerk vergt, omdat je niet alleen de GetUsersByRole-methode moet uitbreiden om een aanroep naar Roles.GetUsersInRole te doen, maar ook het resultaat moet verwerken dat als input dient voor de stored procedure.

Michiel van Otegem is Lead Architect bij The Vision Web (www.thevisionweb.com), een Microsoft Gold Partner. Ook is Michiel medeoprichter van www.aspnl.com. Voor vragen en opmerkingen kun je terecht op michiel@aspnl.com of zijn blog www.vanotegem.nl.

Referenties

Tutorial over Membersip en Roles:
http://weblogs.asp.net/scottgu/archive/2006/06/19/ASP.NET-2.0-Security_2C00_-Membership-and-Roles-Tutorials.aspx
Oracle Provider: http://msdn.microsoft.com/library/en-us/dnbda/html/bdasamppet4.asp
MySQL Provider: www.codeproject.com/aspnet/MySQLMembershipProvider.asp
Zelf een Provider maken: www.devx.com/asp/Article/29256