Pieter Gheysens is .NET-consultant bij Compuware België

Bepaal zelf het registratieproces bij ASP.NET Membership

HET REGISTRATIEPROCES IN DE PRAKTIJK

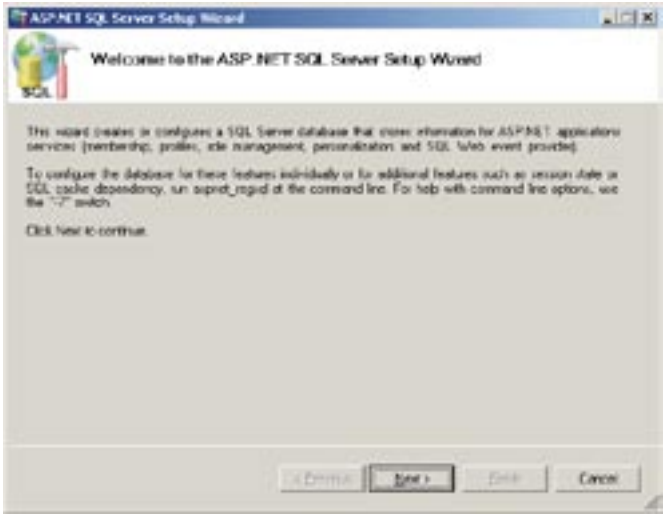
Door het afstellen van de login-controls en de configuratie van ASP.NET Membership kun je eenvoudig zelf het registratieproces van je gebruikers bepalen en verfijnen. In het uitgewerkte voorbeeld wordt ook gebruikgemaakt van ASP.NETRole Management en User Profiles.

Wat is ASP.NET Membership? ASP.NET Membership geeft je de mogelijkheid bij het ontwikkelen van een website op een eenvoudige manier gebruikersgegevens te valideren, op te slaan en op die manier toegang te verlenen tot bepaalde afgeschermde delen van je website. Met andere woorden, het is bedoeld om de authenticatie van gebruikers te beheren zonder dat je als ontwikkelaar alle code hiervoor dient te schrijven. Met de beschikbaarheid van specifieke login-controls in ASP.NET 2.0 kun je een volledig authenticatiesysteem opzetten. Een gedetailleerd overzicht van alle beschikbare login-controls kun je terugvinden in de link aan het eind van het artikel. In plaats van zo'n authenticatiesysteem telkens opnieuw te laten ontwikkelen door verschillende individuele ontwikkelaars, biedt Microsoft dit nu out-of-the-box aan met de mogelijkheid tot uitbreiding. ASP.NET Membership kan bijvoorbeeld uitgebreid worden met extra eigenschappen (gebruikersprofielen/personalisatie) voor individuele gebruikers. Het Membership-verhaal kan natuurlijk ook gecombineerd worden met ASP.NET Role Membership om toegang tot je website aan de hand van verschillende rollen te beheren. Je kunt dus verschillende autorisatiemogelijkheden aanbieden.

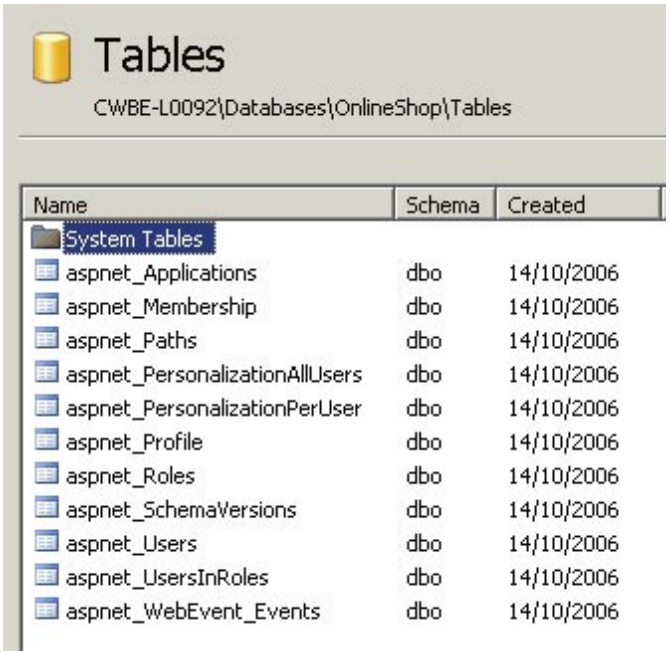
Starten met ASP.NET Membership

Om gebruik te maken van ASP.NET Membership, moeten er wel enkele stappen uitgevoerd worden. In de eerste plaats moet er beslist worden waar alle gebruikersgegevens worden opgeslagen. Er moet dus een keuze gemaakt worden welke bron de membership-gegevens zal bevatten. Dit bepaal je met behulp van de MembershipProvider. In het globale configuratiebestand, machine.config (%systemroot%\Microsoft.NET\Framework\%versionnumber%\CONFIG) vind je een AspNetSqlMembershipProvider terug, die verwijst naar een connectionstring van een locale SQL Express- database, genaamd aspnetdb. De eerste keer dat een ASP.NET 2.0-applicatie opstart en gebruik maakt van Membership-functionaliteit wordt deze database automatisch aangemaakt met de nodige tabellen voor gegevensopslag. In het voorbeeld dat volgt, wil ik echter gebruikmaken van een bestaande SQL Server 2005-database OnlineShop en moet ik zelf de nodige tabellen aanmaken in deze nieuwe database. Hiervoor kan ik gebruikmaken van de Aspnet_regsql.exe executable die teruggevonden kan worden in de folder %systemroot%\Microsoft.NET\Framework\%versionNumber%.

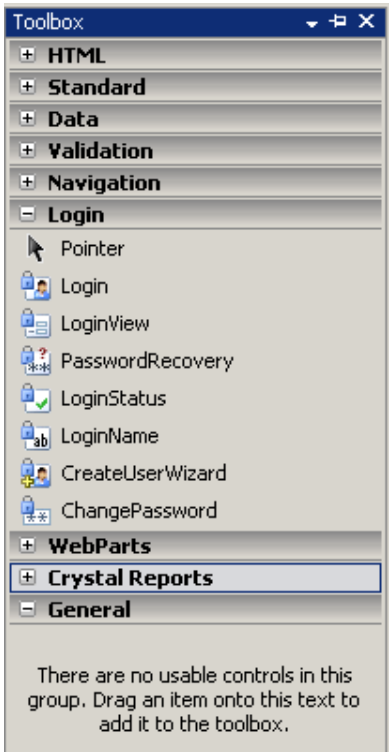
Met de wizard (zie afbeelding 1) kun je gemakkelijk alle noodzakelijke tabelscripts voor alle application services (Membership, Role Manager, Profiles, Personalization) uitvoeren op een bestaande database. Via de command-line is het ook mogelijk deze onderdelen apart te configureren. De tabellen die net aangemaakt zijn via de wizard (zie afbeelding 2) worden gebruikt om alle gebruikersgegevens voor de applicatie op te slaan. Neem beslist zelf een kijkje



Afbeelding 1. De wizard ASP.NET SQL Server Setup



Afbeelding 2. De tabellen die zijn aangemaakt via de wizard

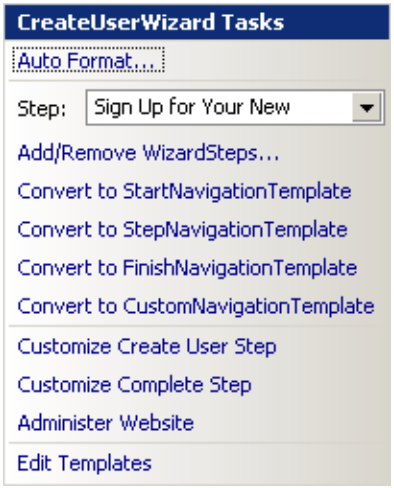


Afbeelding 3. De toolbox

in SQL Server 2005 om te zien hoe deze tabellen aan elkaar gekoppeld zijn en welke informatie ze kunnen bevatten. Er is ook een aantal stored procedures aangemaakt in de database voor het wegschrijven en ophalen van al deze informatie vanuit de applicatie.

Het tweede gedeelte van de voorbereiding is het aanpassen van het website-specifieke configuratiebestand web.config om zo de link te leggen vanuit de applicatie naar de Membership-tabellen. Dit gebeurt met behulp van Forms-authenticatie in plaats van de standaard Windows-authenticatie. In de web.config (zie codevoorbeeld 1) heb ik hiervoor een nieuwe SqlMembershipProvider toegevoegd (OnlineShopAspNetSqlMembershipProvider), die verwijst naar de OnlineShop-database connectionString. Om ASP.NET Membership te configureren voor een applicatie kun je ook nog gebruikmaken van de Web Site Administration Tool waarmee je de configuratie van de applicatie kunt beheren via een wizard (zie de link in de referenties). Ik kom later nog terug op verschillende eigenschappen van de SqlMembershipProvider.

Afbeelding 4. Aanmaken van een nieuwe gebruiker



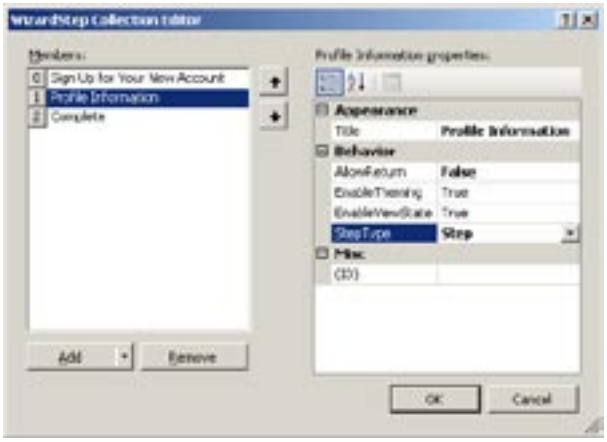
Afbeelding 5. De smart tag op de CreateUserWizard

Aanmaken nieuwe gebruikers

Laten we nu van dichterbij de control CreateUserWizard bekijken die ik zal gebruiken voor het aanmaken van nieuwe gebruikers op de OnlineShop-website. Deze control maakt achter de schermen gebruik van de gekozen MembershipProvider in de web.config om gebruikers op te slaan. Je kunt deze control eenvoudig op een nieuwe webform slepen vanuit de toolbox (zie afbeelding 3) in Visual Studio 2005 en zonder extra aanpassingen is deze control eigenlijk al onmiddellijk bruikbaar om nieuwe gebruikers aan te maken.

In de web.config (zie codevoorbeeld 1) heb ik ervoor gekozen de optie QuestionAndAnswer niet te gebruiken tijdens de registratie. Bij het opstarten van de nieuwe webpagina met de CreateUserWizard- control (zie afbeelding 4) kun je zien dat er met deze opties rekening gehouden wordt. Als er wel voor de optie QuestionAndAnswer was gekozen, dan zouden er twee extra velden zichtbaar zijn op het formulier om extra informatie op te nemen van de gebruikers indien zij later hun paswoord zouden zijn vergeten (Question & Answer). Ook andere opties - zoals de sterkte van het paswoord (lengte, aantal niet-alfanumerieke tekens, ...) – kunnen op deze manier aangepast worden.

Via de smart tag op de CreateUserWizard (zie afbeelding 5) zie je nog meer aanpassingsmogelijkheden voor de control. In afbeelding 4 is te zien dat ik de formattering (autoformat) van de control al heb aangepast aan het Professional Schema. Er kan bijvoorbeeld nog een extra stap aan de wizard toegevoegd worden om extra informatie te weten te komen van de geregistreerde gebruikers. Om dit mogelijk te maken heb ik de gewenste profieleigenschappen toegevoegd (voornaam, achternaam, geboortedatum) in de web.config (zie codevoorbeeld 1). Hiervoor definieer ik ook een



Afbeelding 6. Een extra stap toevoegen aan de CreateUserWizard

custom ProfileProvider die op zijn beurt gebruikmaakt van de OnlineShop connectionString. Deze profieleigenschappen kunnen later strongly-typed vanuit de code aangesproken worden met behulp van de ProfileCommon-klasse.
Ik voeg ook nog een extra stap toe aan de CreateUserWizard (zie afbeelding 6), waarna ik de template voor de nieuwe stap naar eigen goeddunken kan aanpassen. Dit leidt ten slotte tot het resultaat in afbeelding 7. Het aanpassen van de templates kan in het begin vrij stroef overkomen, maar na een tijdje experimenteren schiet je al aardig op. Een interessante link is toegevoegd aan de referenties, zodat je kunt nagaan hoe je hier het best mee aan de slag kunt.

Opslaan profielinformatie

Tot dusver heb ik eigenlijk nog geen letter C#-code geschreven, maar voor het opslaan van de profielinformatie komt daar (eindelijk) verandering in. Wat dient er te gebeuren? Wel, als de profielgegevens zijn ingevuld door de gebruiker (zie afbeelding 7), dient deze informatie ook daadwerkelijk weggeschreven te worden naar de database in de aspnet_Profile-tabel. In codevoorbeeld 2 zie je hoe dit precies gebeurt in het event NextButtonClick. In de eerste plaats zorg ik ervoor dat de profielgegevens enkel en alleen opgeslagen worden wanneer er op de next-knop wordt geklikt in de Profile-stap, die is toegevoegd aan de CreateUserWizard (CreateUser : StepIndex = 0; CreateProfile : StepIndex = 1). Het event (NextButtonClick) kan immers ook voor andere stappen in de wizard uitgevoerd worden. Vervolgens wordt er een nieuw profiel (ProfileCommon) aangemaakt aan de hand van de gebruikersnaam die in de vorige stap door de gebruiker werd ingevoerd. Op dit moment is de gebruiker al geregistreerd in het systeem en moet de profielinformatie alleen nog aan de huidige gebruiker worden gekoppeld.

Via de methode FindControl op de CreateUserWizard-control ga ik verder op zoek naar de controls die mij de nodige gegevens kunnen verschaffen voor het voltooien van het profiel van de gebruiker. Deze controls heb ik zelf toegevoegd aan de template die bij de extra wizard-stap hoort. Hier zie je nogmaals dat de profieleigenschappen onmiddellijk en strongly-typed beschikbaar zijn in de ProfileCommon-klasse. Een groot verschil met bijvoorbeeld het Session-object, waar typefouten gemakkelijker voorkomen en niet altijd even snel opgemerkt worden, omdat je gebruik moet maken van strings om je eigenschappen te identificeren. Er wordt

Afbeelding 7. Het uiteindelijke resultaat

```
<?xml version="1.0"?>
<configuration>
<appSettings/>
  <connectionStrings>
    <add name="OnlineShop"
         connectionString="Data Source=servername;
           Integrated Security=True;   User Instance=False;
           Initial Catalog=OnlineShop;"
         providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
    <compilation debug="true" />
    <authentication mode="Forms">
      <forms loginUrl="Default.aspx"
          defaultUrl="Members/Default.aspx" />
    </authentication>
    <membership
      defaultProvider="OnlineShopAspNetSqlMembershipProvider">
      <providers>
        <add name="OnlineShopAspNetSqlMembershipProvider"
          connectionStringName="OnlineShop"
          enablePasswordRetrieval="false"
          enablePasswordReset="true"
          requiresQuestionAndAnswer="false"
          applicationName="/"
          requiresUniqueEmail="false"
          passwordFormat="Hashed"
          maxInvalidPasswordAttempts="5"
          minRequiredPasswordLength="4"
          minRequiredNonalphanumericCharacters="0"
          passwordAttemptWindow="10"
          passwordStrengthRegularExpression=""
          type="System.Web.Security.SqlMembershipProvider,
            System.Web,  Version=2.0.0.0, Culture=neutral,
            PublicKeyToken=b03f5f7f11d50a3a" />
      </providers>
    </membership>
    <roleManager enabled="true"
                 defaultProvider="OnlineShopRoleProvider">
      <providers>
        <add name="OnlineShopRoleProvider"
          type="System.Web.Security.SqlRoleProvider,   System.Web,
            Version=2.0.0.0, Culture=neutral,
            PublicKeyToken=b03f5f7f11d50a3a"
          connectionStringName="OnlineShop" />
      </providers>
    </roleManager>
    <profile defaultProvider="OnlineShopProfileProvider">
      <providers>
        <add name="OnlineShopProfileProvider"
          type="System.Web.Profile.SqlProfileProvider, System.Web,
            Version=2.0.0.0, Culture=neutral,
            PublicKeyToken=b03f5f7f11d50a3a"
          connectionStringName="OnlineShop"/>
      </providers>
      <properties>
        <add name="FirstName" />
        <add name="LastName" />
        <add name="BirthDate" type="System.DateTime" />
      </properties>
    </profile>
  </system.web>
  <system.net>
    <mailSettings>
      <smtp from="mail@onlineshop.be">
        <network host="localhost" password="" userName=""/>
      </smtp>
    </mailSettings>
  </system.net>
  <location path="Members">
    <system.web>
      <authorization>
        <allow roles="Members" />
        <allow roles="Administrators" />
        <deny users="*" />
      </authorization>
    </system.web>
  </location>
</configuration>
```

Codevoorbeeld 1.

dus achter de schermen een type-safe wrapper gebouwd rondom de profieleigenschappen in de web.config (zie codevoorbeeld 1). Deze eigenschappen hoeven bovendien niet enkel van het type string te zijn, maar kunnen elk geldig serialiseerbaar .NET-datatype aannemen. De profielgegevens worden uiteindelijk weggeschreven naar de database met behulp van gegenereerde stored procedures. Als je geïnteresseerd bent op welke manier deze dynamische data in de database worden weggeschreven, kun je een kijkje nemen in de database (tabel aspnet_Profile).

ASP.NET Role Management

In codevoorbeeld 2 is te zien hoe de geregistreerde gebruiker onmiddellijk wordt toegewezen aan de rol Members. Hier wordt de autorisatie aan de authenticatie gekoppeld. De Role Management Provider werd gedefinieerd in de web.config (codevoorbeeld 1). In de vorige editie van het .NET Magazine (#14) kun je het artikel 'Doe meer met ASP.NET 2.0 Membership en Role Management' nog eens doornemen dat dieper ingaat op deze providers.

Aangepaste registratieprocedure

De standaard registratieprocedure, aangeboden via de CreateUserWizard, laat toe dat nieuwe gebruikers na registratie onmiddellijk toegang krijgen tot de afgeschermde delen van de website. Wat ik nu in dit artikel verder wil uitwerken, is een aangepaste registratieprocedure waarbij nieuwe gebruikers niet onmiddellijk toegang krijgen, maar dat na het registreren en het voltooien van alle stappen er vanuit de applicatie een registratie-e-mail verstuurd wordt naar de gebruiker. Deze dient om de inschrijving te bevestigen via een gepersonaliseerde bevestigingslink in de e-mail. Op deze manier ben je er zeker van dat de personen die op de website een nieuwe account hebben aangevraagd, ook daadwerkelijk over het e-mailadres beschikken dat ze hebben ingevoerd in het formulier. In eerste instantie zet ik hiervoor de property DisableCreatedUser van de CreateUserWizard-control op True, zodat de gebruiker zich niet onmiddellijk na de registratie kan aanmelden via de login-control. Achterliggend wordt in de database, in de tabel aspnet_Membership, een IsApproved-boolean bijgehouden die de authenticatie regelt. Zelf heb ik een extra functie, genaamd sendRegistrationEmail, geschreven (zie codevoorbeeld 3) met de bedoeling een e-mail uit te sturen naar de net geregistreerde gebruiker ter bevestiging van zijn gegevens. Deze functie moet dus aangesproken worden vanuit het event NextButtonClick, na het opslaan van de profielgegevens. In de code kun je zien dat de bevestigingslink wordt opgebouwd met de specifieke identificatiegegevens van de geregistreerde gebruikers. De ProviderUserKey-property op de MembershipUser bevat een Globally Unique Identifier uit de database of GUID, waarmee de gebruiker werd opgeslagen en waarmee deze geïdentificeerd wordt in de applicatie. De gebruikersnaam is ook nog toegevoegd aan de query-string om de verificatie van de gebruikersnaam te koppelen aan de UserId (zie verderop in dit artikel). Voor het versturen van e-mails vanuit je applicatie is het wel belangrijk dat je de SMTP-mailsettings juist configureert in de system.net-namespace (zie codevoorbeeld 1) en dat je ergens een SMTP-server draait.

Bevestigingsregistratie

Wanneer de registratie-e-mail aankomt bij de gebruiker, is het de beurt aan de gebruiker om zelf zijn registratie te bevestigen via de gepersonaliseerde bevestigingslink. Dit biedt het voordeel dat het

```
protected void CreateUserWizard1_NextButtonClick
(object sender, WizardNavigationEventArgs e)
{
  //Create Membership User : StepIndex = 0
  //Profile Information : StepIndex = 1
  //Completion : StepIndex = 2
  if (e.CurrentStepIndex == 1)
  {
    ProfileCommon p = (ProfileCommon)ProfileCommon.Create
      (CreateUserWizard1.UserName, true);

    //FirstName value is stored in textbox with id FirstName
    TextBox txtFirstName =
      (TextBox)CreateUserWizard1.FindControl("FirstName");
    p.FirstName = txtFirstName.Text;

    //LastName value is stored in textbox with id LastName
    TextBox txtLastName =
      (TextBox)CreateUserWizard1.FindControl("LastName");
    p.LastName = txtLastName.Text;

    //BirthDate value is stored in calendar with id BirthDate
    Calendar calendarBirthDate =
      (Calendar)CreateUserWizard1.FindControl("BirthDate");
    p.BirthDate = calendarBirthDate.SelectedDate;

    //Save Profile information
    p.Save();

    //Add role "Members" to registered user
    Roles.AddUserToRole(CreateUserWizard1.UserName, "Members");

    //Send registration mail to user for confirmation
    this.sendRegistrationEmail(
      Membership.GetUser(CreateUserWizard1.UserName));
  }
}
```

Codevoorbeeld 2.

```
private void SendRegistrationEmail(MembershipUser memberShipUser)
{
  SmtpClient client = new SmtpClient();

  using (MailMessage newMailMessage = new MailMessage())
  {
    //From mailAddress
    newMailMessage.From =  new MailAddress("postmaster@onlineShop.be",
      "Online Registration");

    //To mailAddress
    newMailMessage.To.Add (new MailAddress(memberShipUser.Email,
      memberShipUser.UserName));

    newMailMessage.Subject = "Online Registration - Confirmation";
    newMailMessage.IsBodyHtml = true;

    //Create custom hyperlink with UserName & UserId of member
    string confirmationLink =
      "http://www.onlineshop.be/Registration.aspx?UserName="
      + memberShipUser.UserName + "&UserId="
      + memberShipUser.ProviderUserKey.ToString();

    //Create body text for email
    StringBuilder sbContent = new StringBuilder();
    sbContent.Append("Hello " + memberShipUser.UserName +"<br/><br/>");
    sbContent.Append("You were successfully registered.");
    sbContent.Append("<a href='" + Server.UrlEncode(confirmationLink)
                     + "'>Confirm registration.</a>");

    newMailMessage.Body = sbContent.ToString();

    //Send mail to member
    client.Send(newMailMessage);
  }
}
```

Codevoorbeeld 3.

e-mailadres al geverifieerd wordt en dat er geen misbruik wordt gemaakt van een automatische registratie door webrobots. In de web.config (zie codevoorbeeld 1) kan bovendien de property requiresUniqueEmail (SqlMembershipProvider) op true geplaatst worden om te verzekeren dat er slechts één gebruiker aangemaakt kan worden per e-mailadres.

De bevestigingslink verwijst naar een pagina, Registration.aspx, die momenteel nog niet aangemaakt is. Hiervoor kan gewoon een nieuwe webpagina aangemaakt worden die zal reageren op het Page_Load-event (zie codevoorbeeld 4) om enkele verificaties te verrichten. De query-string wordt gecontroleerd op de aanwezigheid van de eigenschappen UserName en UserId (GUID aangemaakt door het systeem). Daarna wordt de member opgehaald via de GetUser-operatie op het Membership-object en wordt de onontbeerlijke UserId vergeleken. Indien de UserIds overeenstemmen, wordt de boolean IsApproved op True geplaatst en wordt dit ook doorgevoerd in de database (tabel aspnet_Membership) door de UpdateUser-methode.

Login

Op dit moment houdt niets de gebruiker nog tegen om in te loggen via een login-formulier (aangemaakt met de Login-control – zie afbeelding 8) en doorverwezen te worden naar de standaard url van de member-sectie (zie defaultUrl in Forms-element in codevoorbeeld 1).

Deze member-sectie kan tevens beveiligd worden in de web.config (zie location-element onderaan codevoorbeeld 1). Alleen gebruikers die deel uitmaken van de rollen Administrators of Members worden toegelaten, alle andere gebruikers wordt de toegang ontzegd.

```
protected void Page_Load(object sender, EventArgs e)
{
   if (Request.QueryString["UserName"] != null
      && Request.QueryString["UserId"] != null)
   {
      //Get member with given username
      MembershipUser member =
         Membership.GetUser(Request.QueryString["UserName"]);

      //Verification of ID
      if (member.ProviderUserKey.ToString() ==
         Request.QueryString["UserId"])
      {
         //approve member and update
         member.IsApproved = true;
         Membership.UpdateUser(member);
      }
   }
}
```

Codevoorbeeld 4.

```
protected void Login1_LoginError(object sender, EventArgs e)
{
   MembershipUser member = Membership.GetUser(Login1.UserName);
   if (member != null)
   {
      if (!member.IsApproved)
      {
         //Notify member to confirm his registration
         Login1.FailureText = "Please confirm registration-email.";
      }
   }
}
```

Codevoorbeeld 5.

Afbeelding 8. De gebruiker kan inloggen

De login-control heeft een event LoginError waar men een mogelijke foutboodschap voor de gebruiker nog kan toelichten of aanpassen. In codevoorbeeld 5 vang ik bijvoorbeeld de login op van een gebruiker die zich al geregistreerd heeft via de website, maar die de registratie-e-mail nog niet heeft bevestigd. Wanneer deze specifieke situatie niet zelf wordt opgevangen in de code, krijgt de gebruiker geen specifieke foutboodschap terug. Andere situaties zoals een lock-out kunnen hier bijvoorbeeld ook opgevangen worden.

Minder code

ASP.NET 2.0 is vooral gericht op het verminderen van de hoeveelheid programmacode die nodig is om een applicatie te maken. In ASP.NET 1.x vergt Forms-beveiliging vrij veel werk voor de ontwikkelaar, aangezien alles nog uitgewerkt dient te worden. In ASP.NET 2.0 daarentegen is dit alles nu veel eenvoudiger geworden met behulp van de Membership API en de daaraan verwante Role Manager API. De login-controls zijn vrij intuïtief te gebruiken en bieden voldoende uitbreidingsmogelijkheden voor meer specifieke vereisten.

Pieter Gheysens is .NET-consultant bij Compuware België (www.compuware.be) Zijn blog is te lezen op http://kinnie.blogspot.com en u kunt hem bereiken via zijn e-mailadres Pieter.Gheysens@gmail.com

Referenties

Microsoft ASP.NET 2.0 Providers : http://msdn2.microsoft.com/en-us/library/aa478948.aspx

ASP.NET Login Controls Overview : http://msdn2.microsoft.com/en-us/library/ms178329.aspx

Website Administration Tool : http://msdn2.microsoft.com/en-us/library/yy40ytx0.aspx

How to customize the ASP.NET CreateUserWizard Control : http://msdn2.microsoft.com/en-us/library/ms178342.aspx