Aram Smith en Pieter de Bruin zijn werkzaam bij Avanade als respectievelijk architect in system engineering en solution developer

Authenticatie en autorisatie met ASP.NET 2.0 en ADAM

MEMBERSHIPS, PROFILES EN ROLLEN MET ACTIVE DIRECTORY APPLICATION MODE

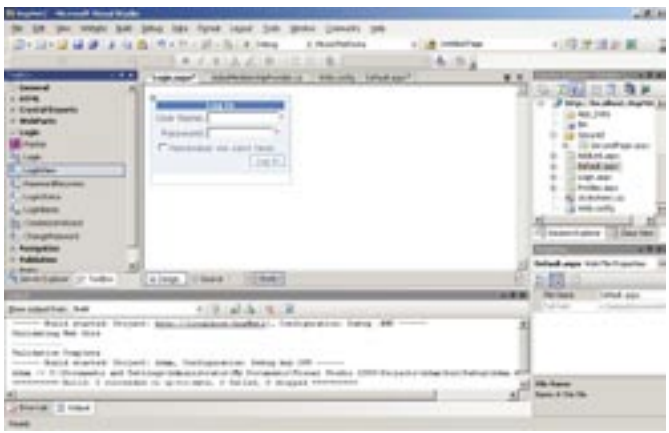
Werken met gebruikersnamen en wachtwoorden is vereenvoudigd in ASP.NET 2.0. Deze vereenvoudiging heeft gezorgd voor nieuwe controls, classes en patronen. Hoe werkt deze nieuwe manier van herkennen van gebruikers en wat zijn de mogelijkheden?

Met de introductie van ASP.NET stelt de Microsoft sterke authenticatie-mechanismes beschikbaar aan ontwikkelaars zoals Windows-, Passport- en forms-authenticatie. Het nadeel was dat ontwikkelaars zelf de logica moesten ontwikkelen om vanaf twee textboxen en een button tot een geauthenticeerde gebruiker met rechten te komen. In ASP.NET 2.0 biedt het .NET Framework nu standaard beheerbaarheid van gebruikers en gebruikersgegevens. In de toolbox in Visual Studio staan, naast de gebruikelijke html-controls, 40 server-controls, waaronder specifieke controls voor het behandelen van gebruikersnamen en wachtwoorden. De betreffende controls zijn Login, LoginView, LoginStatus, LoginName, PasswordRecovery, ChangePassword en CreateUserWizard.

Deze login-controls communiceren met de membership-API die methodes aanbiedt voor het beheren van gebruikers en rollen. Zo zijn er methodes om je wachtwoord te wijzigen, om een wachtwoord te resetten en om een gebruiker te valideren. De membership-API biedt uniforme toegang tot code die verschillende implementaties kan hebben. Bijvoorbeeld de CreateUser-methode is in alle Membership-implementaties aanwezig, maar de onderliggende gegevensbron, een database of bestand, kan in iedere implementatie anders zijn.

Provider design pattern

De membership-API maakt gebruik van de provider-interface. Het Provider Design Pattern is een pattern dat al gebruikt werd



Afbeelding 1. Nieuwe controls in de Visual Studio toolbox

Methode	Beschrijving
ChangePassword	Verwerkt het verzoek om het wachtwoord van een gebruiker te wijzigen
ChangePasswordQuestionAndAnswer	Verwerkt het verzoek om de wachtwoordvraag en –antwoord van een gebruiker te wijzigen
CreateUser	Voegt een nieuwe gebruiker toe aan de data source
DeleteUser	Verwijdert een gebruiker van de data source
FindUsersByEmail	Geeft een collectie van gebruikers op basis van (een deel van) een e-mailadres
FindUsersByName	Geeft een collectie van gebruikers op basis van (een deel van) een naam
GetAllUsers	Geeft alle gebruikers in de data source
GetNumberOfUsersOnline	Geeft het aantal gebruikers dat momenteel online is
GetPassword	Geeft het wachtwoord voor een gebruiker
GetUser	Geeft het MembershipUser-object van een gebruiker
GetUserNameByE-mail	Geeft een MembershipUser-object voor een gebruiker op basis van het e-mail-adres.
ResetPassword	Stel het wachtwoord opnieuw in met een automatisch gegenereerd wachtwoord
UnlockUser	Activeert de gebruiker in de data source
UpdateUser	Biedt de mogelijkheid om diverse attributen van een gebruiker bij te werken
ValidateUser	Valideert het bestaan van de gebruikersnaam met wachtwoord in de data source

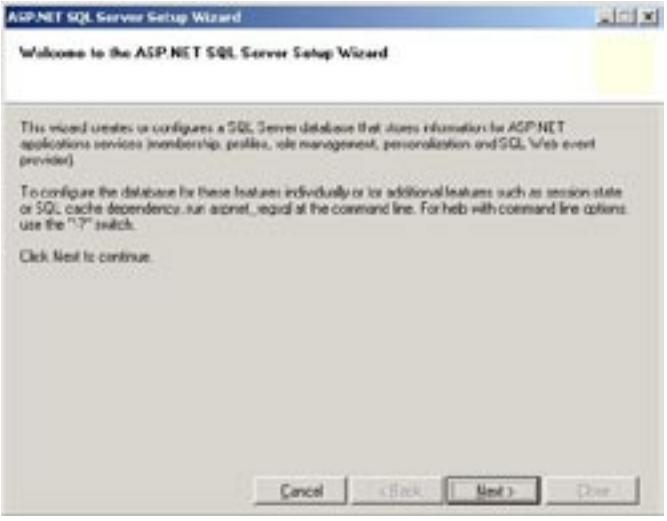
Tabel 1. MembershipProvider API-methodes

Afbeelding 2. Provider-model

in enkele ASP.NET Starter Kits en dat officieel aan het .NET Framework is toegevoegd in ASP.NET Whidbey. De theorie van de provider is dat het een goed gedocumenteerde en eenvoudig te gebruiken API moet zijn die ontwikkelaars alle vrijheid over de intern gebruikte procedures moet geven. De provider is meermalen geïmplementeerd in het .NET Framework, onder andere in de RoleManagerProvider, voor het beheer van rollen en gebruikerskoppelingen en in de ProtectedConfigurationProvider voor het beveiligd opslaan van configuratie-instellingen. Afbeelding 2 laat zien hoe dezelfde set server-controls voor login gebruikt kan worden bij verschillende implementaties van de MembershipProvider-functionaliteit.

Bij Visual Studio 2005 (Community Technology Preview February 2005) wordt automatisch een Access-database gebruikt voor opslag door de System.Web.Security.AccessMembershipProvider. Deze database staat in de App_Data-folder van het project en heet aspnetdb.mdb. De database bevat gegevens van gebruikers, rollen, profiles en applicaties. Of de Access-variant aanwezig zal zijn in de RTM-versie later dit jaar is niet duidelijk. Een wizard kan Microsoft SQL Server instellen voor het opslaan van deze gegevens. De wizard installeert de betreffende SQL Server-database en registreert de SqlMembershipProvider.

Start de wizard in 'C:\WINDOWS\Microsoft.NET\Framework\<versie>\aspnet_regsql.exe'. Dan worden AspNetDB.mdb en -.ldb in de App_Data folder geplaatst. Daarnaast kan ook gebruik gemaakt worden van Active Directory en Active Directory Application Mode door gebruik van de ActiveDirectoryMembershipProvider.
Van oudsher worden gebruikersdata in een relationele database opgeslagen. Dat is nog steeds mogelijk, maar Active Directory en Active Directory Application Mode zijn er juist voor gemaakt om deze gegevens te behandelen. Een database blijft het ideale middel om transactioneel geüpdate data te verwerken en ADAM is een prima plek om meer statische data, die weinig geschreven en veel gelezen wordt, in op te slaan.



Afbeelding 3. De installatiewizard voor het opslaan van service data in SQL Server

Active Directory Application Mode (ADAM)

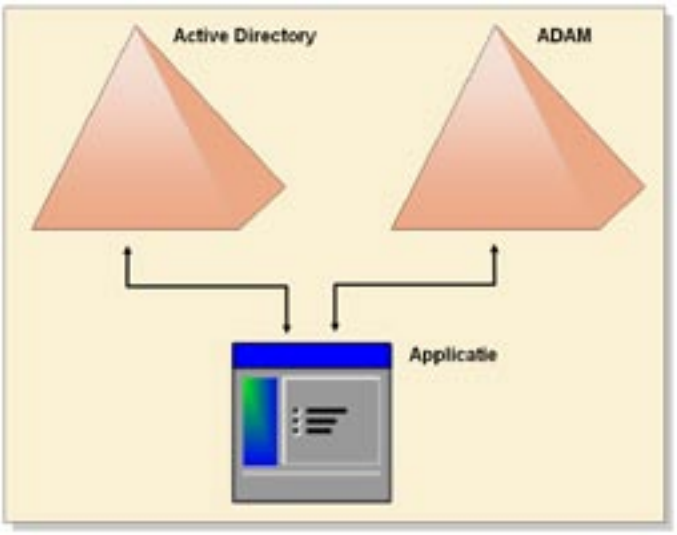
Veel bedrijven hebben hun diensten en het beheer in hun infrastructuur gebaseerd op een directory-service waarbij het LDAP- of X.500-protocol vaak de grondslag vormt. Veel van deze directory-diensten zoals Microsoft Active Directory zijn geïntegreerd in het netwerkbesturingssysteem en hebben veelal een focus op het beveiligen en beheren van een infrastructuur.
Een directory-service gebruikt een schema waarin de definitie is gespecificeerd die informatie bevat zoals objecten en attributen en hun relatie in de directory service. Een schema is bepalend voor een gehele directory-structuur (Forest-niveau), wat inhoudt dat een verandering in het schema een forse impact kan hebben op de gehele infrastructuur. Om nieuwe diensten en applicaties te integreren met een dergelijke directory-service is er vaak een schemawijziging of -uitbreiding noodzakelijk. Hier wringt dan ook de schoen. Veel bedrijven zijn huiverig voor dergelijke schema-extensies. Het resultaat is dat als de schemawijziging fout gaat de Active Directory beschadigd kan worden.
Active Directory Application Mode, kortweg ADAM, is een directory-service die is gebaseerd op het LDAP-protocol en op Active Directory, maar niet is geïntegreerd in een besturingssysteem. Het kan gescheiden op diverse systemen draaien. De application-mode in ADAM geeft aan dat deze directory bedoeld is om applicatieve data en structuren in op te slaan en daarbij gebruik kan maken van diensten die in Active Directory ook geïmplementeerd zijn, zoals replicatie, authenticatie en autorisatie. Het schema in ADAM is zeer klein tegenover het enorme schema van Active Directory en is daardoor eenvoudig en flexibel uit te breiden. ADAM kan geïnstalleerd worden op Windows 2003 Server Standard, Enterprise en Datacenter Edition en op Windows XP. Bovendien is ADAM gratis! Om te kunnen bepalen voor welke toepassingen ADAM ingezet kan worden, is het wenselijk de verschillen tussen ADAM en Active Directory en ADAM en relationele databases zoals SQL Server eens naast elkaar te zetten.

ADAM versus Active Directory

Hoewel ADAM gebaseerd is op de broncode van Active Directory zijn er toch significante verschillen. Zo is ADAM geoptimaliseerd voor het ondersteunen van applicaties die hun data hiërarchisch in een directory willen opslaan. Ook ADAM maakt gebruik van een schema. Deze is echter veel kleiner dan het schema van een Active Directory. Dit is logisch, de scope van een Active Directory is infrastructuurgerelateerd, terwijl het schema van ADAM is geoptimaliseerd om uitbreidingen en aanpassingen van applicaties te ondersteunen.
Bij het partitioneren van een Directory Service wordt een logische partitie van de Directory Data Store toegewezen aan een bepaalde naam-instance. Active Directory heeft een sterke integratie met en afhankelijkheid van het Domain Name System (DNS). Deze is noodzakelijk om diensten in een Active Directory-infrastructuur door name-resolving terug te vinden. Hierdoor ondersteunt Active Directory alleen maar DNS-naamtypering (DC=) voor de directory-partities. In Active Directory worden deze partities dan

Distinguished Name Component	Omschrijving
C=	Country/region
CN=	Common name
DC=	Domain component
L=	Location
O=	Organization
OU=	Organizational unit

Tabel 2. Ondersteunde namen in ADAM



Afbeelding 4. Authenticatie in de Active Directory en gebruikersgegevens in ADAM

ook domeinen genoemd. In Active Directory kan alleen een Distinguished Name, zoals DC=Avanade,DC=NL, voorkomen. Een echte X.500 DN wordt niet ondersteund. In ADAM worden beide stijlen, DNS en X.500, voor namen ondersteund. Hierdoor kunnen de volgende distinguished names worden gebruikt:

In tegenstelling tot Active Directory kan ADAM welzeker de DN DC=Avanade,C=NL ondersteunen. Door de opbouw en beperkingen in de vorm van Forests en Domeinen kan een Active Directory Domein Controller maar één instance van de Active Directory behelzen, dus ook maar één Domein. Omdat ADAM niet geïntegreerd is met een besturingssysteem en infrastructuur kent het deze beperkingen niet en kan op één server dus meer instances van ADAM draaien. Dit kan kosten besparen en mogelijkheden vergroten.

Net als Active Directory kan ADAM zijn datastore repliceren. In Active Directory repliceren domein-controllers met elkaar op basis van het domein-membership. Bij ADAM wordt de datastore gerepliceerd op basis van een configuratieset. Een configuratieset is een groep van ADAM-instances die een gezamenlijk schema en configuratiepartitie delen. Ook al zou ADAM in een Active Directory Forest geïnstalleerd zijn, dan nog is zijn replicatietopologie onafhankelijk van de replicatietopologie van Active Directory. Zowel ADAM als Active Directory zijn multimaster, wat wil zeggen dat er wijzigingen kunnen plaatsvinden op elke Active Directory Domein Controller en dus ook elke ADAM-instance. Als er op twee plaatsen hetzelfde object of attribuut wordt aangepast, kan dit leiden tot conflicten tijdens replicatie. Er zal maar één van de twee wijzigingen na replicatie overblijven. De vraag is natuurlijk welke. De wijziging met de recentste timestamp zal overblijven, dus degene die het laatst is ingevoerd. Hierbij kunnen we de regel hanteren Last Writer Wins.
Bij het ontwerpen van een Active Directory is het ontwerp van een replicatietopologie en -strategie dus uiterst belangrijk. Dit geld dus ook voor ADAM: de beschikbaarheid van data staat of valt met een goede replicatietopologie en -strategie.

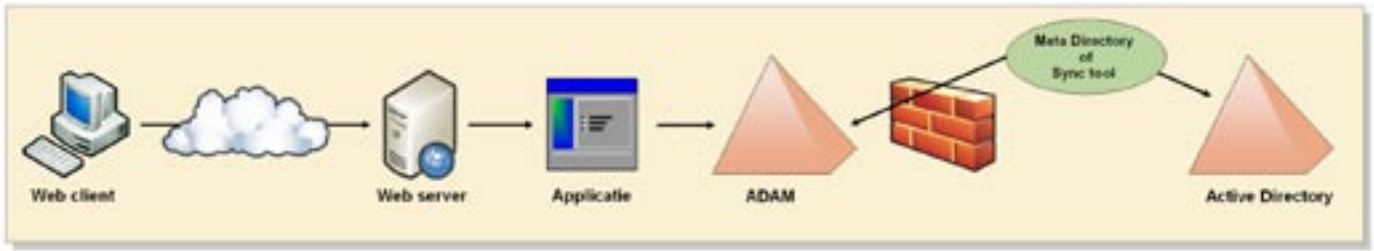
Directory Services (ADAM) versus een relationele database (SQL)
Beide databasetypes zijn ontworpen voor verschillende doeleinden. Een Directory Service zoals ADAM is sterk geoptimaliseerd voor lees- en zoekopdrachten, de informatie is hiërarchisch opgeslagen. Een relationele database zoals SQL Server is juist geoptimaliseerd voor schrijfopdrachten. Door de hiërarchische structuur van een Directory Service kunnen objecten hiërarchisch in containers worden opgeslagen in tegenstelling tot een relationele database, waarbij de data in tabellen en kolommen wordt gerangschikt. Door de hiërarchische opbouw van een Directory Service kan een fijnmazige beveiliging op object- of attribuutniveau worden toegepast, terwijl in een relationele database de beveiliging op rij- of kolomniveau kan worden gezet. Een Directory Service is geoptimaliseerd voor het repliceren van data en is dus loosely coupled. Dit betekent dat er geen consistentie wordt gewaarborgd tussen de replicatiepartners. Een relationele database kenmerkt zich door transactionele verwerking en garandeert een optimale consistentie van de data. Dit wordt onder andere afgedwongen door file- en record-locking door de database server.

Een Directory Service kwalificeert zich voor applicaties waarbij de data snel over een infrastructuur beschikbaar moet zijn voor servers en clients; applicaties waarbij de nadruk ligt op veel of snelle zoek en leesacties; en applicaties waarbij een hiërarchische opbouw gewenst is om bijvoorbeeld attributen en rechten van objecten op andere niveaus te overerven.
Voorbeelden van Directory Service-applicaties zijn: relatiedatabases waarbij een contactobject met attributen zoals telefoon nummer en adresgegevens als attributen worden bewaard; Resource Management-applicaties; applicaties voor infrastructurele toepassingen.

Relationele databases kwalificeren zich voor applicaties met complexe transactiegebaseerde gegevensverwerking met meer objecten en tabellen. Dit wordt niet ondersteund in LDAP Directory-services. Ook kwalificeren zij zich voor dataverwerking waarbij een ACID, Atomicity, Consistency, Isolation and Durability verwerking van belang is. Ten derde voor dataverwerking waarbij de veranderingen en historie in een log-file opgeslagen moeten zijn voor playback- of audit-doelen. En tot slot voor applicaties die intolerant zijn met betrekking tot datareplicatie. Bij LDAP Directories is dit gebaseerd op 'Store and forward' waarbij de data tussen replicatievensters van replicatiepartners niet consistent hoeven te zijn. Data waarbij consistentie is vereist tussen alle replica's moeten gebaseerd zijn op een relationele database. Voorbeelden van databaseapplicaties zijn reservering- en betalingssystemen, en procesgebaseerde applicaties.

Applicatieontwikkeling

Ontwikkelaars die applicaties vervaardigen die met een Directory Service moeten integreren kunnen ADAM goed gebruiken. ADAM biedt dezelfde ervaringen als een Active Directory en kan lokaal zonder DNS geïnstalleerd worden. ADAM kan bijvoorbeeld gebruikt worden voor een Single Sign On (SSO)-toepassing. Een applicatie wil graag eigen (dynamische) data opslaan in een LDAP store. Bij het gebruik van Active Directory zou dit onherroepelijk leiden tot een schemawijziging, wat mogelijk ongewenst is. Door het gebruik van ADAM voor de applicatiespecifieke data - dus alle



Afbeelding 5. Scenario met AD-integratie

Afbeelding 6. Set up-opties

Afbeelding 7. Instance-name

objecten en attributen die niet in Active Directory voorkomen en de Active Directory voor authenticatie en eventueel autorisatie - kan de applicatie de gebruiker valideren tegen de Active Directory, zonder dat deze opnieuw moet inloggen. Ondertussen kan de applicatie de applicatiespecifieke data uit ADAM halen.

Extranet-toegang

Als een bedrijf een applicatie wil aanbieden via een extranet kan hiervoor natuurlijk een Active Directory worden ingericht. Hier moeten dan schema-extensies worden uitgevoerd wanneer een applicatie dat vereist. Daarnaast is een Active Directory niet applicatiegeoptimaliseerd. De inzet van ADAM maakt het in een dergelijk scenario mogelijk om zowel de gebruikersgegevens als de applicatieve data op te slaan. Dit scenario kan worden uitgebreid mochten medewerkers van het bedrijf vanaf andere locaties via het extranet willen aanloggen. Door de gebruikersgegevens vanuit de Active Directory te synchroniseren met ADAM in het extranet, voorkom je mogelijke beveiligingsproblemen doordat Active Directory zelf niet in de DMZ aanwezig hoeft te zijn.

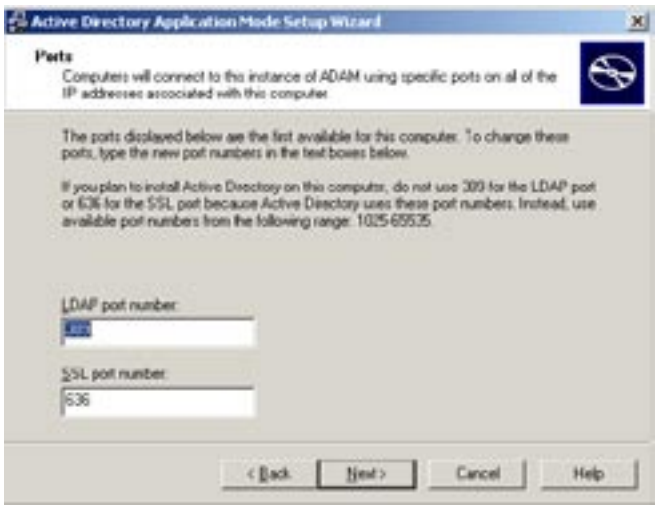
Installatie en beheertooling

Het installeren van ADAM is redelijk eenvoudig. Microsoft heeft ADAM uitgerust met de Active Directory Application Mode Setup Wizard . Zie de stappen van de setup-wizard in afbeelding 6 tot en met 9. De eerste keus waar je bij de installatie van ADAM voor staat is: een nieuwe instance of een replica van een bestaande instance? Hierna moet een instance-naam opgegeven worden. Vervolgens moeten de poorten worden opgegeven waarover deze instance te bereiken is. Let er op dat als er een bestaande Active Directory aanwezig is, je de voorgestelde poorten niet kunt gebruiken. Hierna (afbeelding 9) kan een partitie ingegeven worden. Hierbij is het belangrijk de juiste DN-naamgeving te gebruiken. Deze partitienaam vormt samen met de instance-naam en poortnummer de connection-string om vanuit de code met de instance te communiceren.

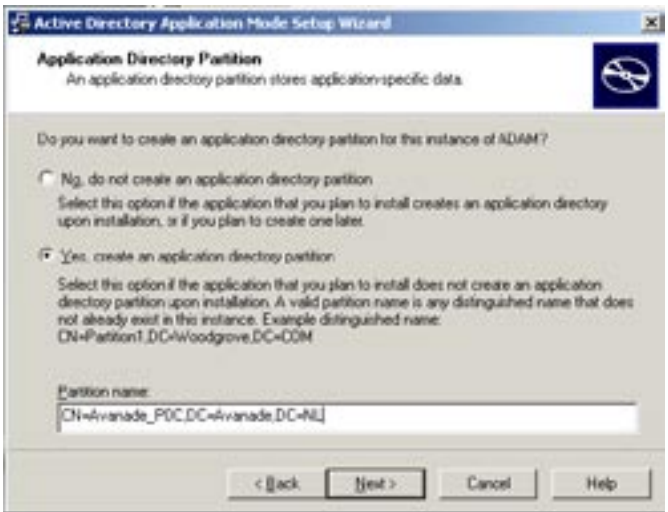
Hierna zal ADAM nog vragen om het pad van de databestanden en het security-account waaronder hij moet draaien. Om ADAM te beheren worden diverse tools meegeleverd, waaronder ADSI Edit, LDP en REPADMIN. Het nadeel van ADAM is dat de beheertools nogal Spartaans aandoen, zeker voor diegenen die gewend zijn aan de luxe GUI-tools van Active Directory. En bedenk goed dat hoewel je bepaalde objecten in ADAM niet kunt weggooien, zoals het schema (CN=Enterprise Schema), je met ADSI EDIT de ADAM-instance snel om zeep kunt helpen. ADAM is uitgerust met een zeer goed helpbestand dat een goede uitleg geeft hoe bijvoorbeeld het schema door middel van Ldifde.exe aangepast kan worden.

Single Sign On

De heilige graal voor applicaties is de transparantie voor gebruikers, ofwel Single Sign On (SSO). Het kan zeer wenselijk zijn de Active Directory-gebruikersgegevens met ADAM te synchroniseren. Hiervoor zijn drie tools beschikbaar met elk hun eigen kenmerken. Voorheen waren alleen Microsoft Identity and Integration Server (MIIS) en Identity and Integration Featurepack beschikbaar (IIFP). Later heeft Microsoft de 'Active Directory to ADAM synchronizer' beschikbaar gesteld. Het MIIS-product is een dure, maar ook complexe aangelegenheid. Het voordeel van MIIS is dat het ook andere directory- en databasebronnen kan uitlezen. MIIS beheert ook de status van de directory-services en houdt deze gesynchroniseerd.



Afbeelding 8. Poorten



Afbeelding 9. Applicatie-partitie

Kenmerk	Active Directory to ADAM Synchronizer	IIFP	MIIS
Synchronisatie endpoints	Alleen Active Directory en ADAM .	Alleen Active Directory, ADAM en Exchange.	Elke directory of nondirectory identity stores, zoals: Active Directory, ADAM, Exchange, Lotus Notes, SQL Server, Oracle, Sun Java System Directory Server 5.2 (voorheen iPlanet Directory Server).
Dataflow direction	Eén richting, alleen van Active Directory (bron) naar ADAM (doel).	Elke richting synchronisatie tussen de endpoints.	Elke richting synchronisatie tussen de endpoints.
Data transformatie	Wijzigingen zijn niet toegestaan in objecten of attributes van bron naar doel.	Wijziging in objecten of attributes van bron naar doel is toegestaan.	Wijziging in objecten of attributes van bron naar doel is toegestaan.
Conflictresolutie	Geen resolutie voor conflict tussen bron en doel. Last writer wins.	Kan geconfigureerd worden om de stroom van data tussen de identity stores te beheren. Bepaald kan worden welke data geïmporteerd moeten worden en bepalen welke directory de authoritative data store is.	Kan geconfigureerd worden om de stroom van data tussen de identity stores te beheren. Bepaald kan worden welke data geïmporteerd moeten worden en bepalen welke directory de authoritative data store is.
Configuratie-vereisten	Vereist een computer met ADAM en een XML-configuratiebestand (kan programmatisch gegenereerd worden).	Eventueel een additionele server vereist met MIIS en installatie van Active Directory management agent voor elke Active Directory forest waarvan de applicatie data nodig heeft.	Eventueel een additionele server vereist met MIIS en installatie van Active Directory management agent voor elke Active Directory forest waarvan de applicatie data nodig heeft.
Intermediary data store	Active Directory naar ADAM Synchronizer benut geen tussentijdse opslag. De data gaan direct van Active Directory naar ADAM.	Vereist een derde store voor alle data die opgevraagd worden. Dit kan SQL Server 2000 Standard Edition of de Enterprise Edition zijn.	Vereist een derde store voor alle data die opgevraagd worden. Dit kan SQL Server 2000 Standard Edition of de Enterprise Edition zijn.
Beheer	Vereist geen additioneel beheer naast het beheer van ADAM.	Vereist een IIFP-beheerder voor het onderhoud van MIIS Server en de SQL Server.	Vereist een MIIS-beheerder voor het onderhoud van MIIS Server en de SQL Server.
Eenvoud van deployment	Active Directory to ADAM Synchronizer richt zich op één taak: datastroom in één richting van Active Directory naar ADAM.	Minder complex dan MIIS omdat IIFP alleen functioneert tussen Active Directory, ADAM en Exchange.	Volledige identity management-oplossing voor heterogene omgeving. Biedt verschillende management agents, object en rules.
Inspanning voor ontwikkelaar	Weinig inspanning voor een ontwikkelaar wanneer een applicatie met ADAM-functionaliteit wordt gebruikt.	Enige inspanning vereist voordat applicatie met ADAM-functionaliteit gebruikt kan worden.	Behoorlijke inspanning voor een ontwikkelaar is vereist voordat ADAM-functionaliteit gebruikt kan worden, in het bijzonder bij heterogene bronnen.

Tabel 3. Verschillen tussen synchronisatietools

Het IIFP is eigenlijk een lichte versie van MIIS en kan alleen synchroniseren tussen Active Directory, ADAM en Exchange Server 200x. De 'Active Directory to ADAM synchronizer' tool is de eenvoudigste vorm en kan dan ook alleen van Active Directory naar ADAM synchroniseren. De belangrijkste verschillen tussen deze synchronisatietools zijn te lezen in tabel 3.

Om de ActiveDirectoryMembershipProvider te gebruiken moet je de provider registreren en aanroepen. De gegevens in code-voorbeeld 1 laten zien hoe de connectionstring voor LDAP moet worden toegevoegd. Ook staat er hoe de provider kan worden toegevoegd aan de membership-sectie.

Wil je meer vrijheid op het gebied van LDAP-benaderingen, gebruik dan de nieuwe System.DirectoryServices.ActiveDirectory.ADAMInstance-class. Deze class erft van de DirectoryServer-class en voorziet in alle gebruikelijke handelingen met een directory service.

Als je gebruikersgegevens in een bestaande database wilt gebruiken of misschien liever een xml-bestand of een andere relationele database op basis van OLEDB, dan kun je je eigen membership-provider ontwikkelen. Custom providers zijn ook het overwegen waard als het te complex wordt om de standaard providers te gebruiken, bijvoorbeeld als er meer gegevensbronnen aangesproken moeten worden.

Custom MembershipProvider

Wanneer je een nieuwe provider maakt die overerft van de System.Web.Security.MembershipProvider-class dan moet je zelf zorgen voor de implementatie van base-methodes (genoemd in tabel 1) en properties. Daarnaast moet ook de applicatie correct geconfigureerd worden. Configureer de applicatie zodat deze forms-authenticatie gebruikt. Gewoonlijk geef je aan dat enkele of alle pagina's slechts toegankelijk zijn voor bepaalde geauthenticeerde gebruikers. Geef gebruikeraccounts op voor membership. Je kunt dit op een aantal manieren doen. Je kunt de Web Site Administration-tool gebruiken, die je een wizard-achtige interface geeft voor het aanmaken van nieuwe gebruikers. Je kunt ook een registratiepagina maken waarin je gebruikersnaam, wachtwoord en eventueel meer gegevens van de gebruiker ophaalt en met de membership-methode CreateUser een nieuwe gebruiker aanmaakt in het membership-systeem.

Configuration editors

Het instellen van de membership gebeurt in web.config. Waar in versie 1 het wijzigen van instellingen in web.config uitdagend kon zijn, is dat in de tweede versie van het Framework vergemakkelijkt. Er is nu zowel een Windows-interface als een webbased variant van de configuratie-editor. De Windows-interface is op te roepen via Internet Information Services, op het laatste tabblad van eigenschappen van de website of via virtual directory.

Hier heb je mogelijkheden voor het instellen van connection strings, custom errors, autorisatie, authenticatie, application language en encoding en state management.

De web interface benader je via het het ASP.NET Configuration-item in het website-menu van Visual Studio .NET 2003 of via applicatie-url/webadmin.axd. Het security-tabblad van de webinterface gebruiken we om gebruikers, rollen en rechten in te stellen en het provider-tabblad voor provider-eigenschappen. De security setup wizard loodst je door de mogelijkheden heen.

Achtereenvolgens kom je door de Select Access Method, Data Store, Define Roles, Add New Users en Add New Access Rules.

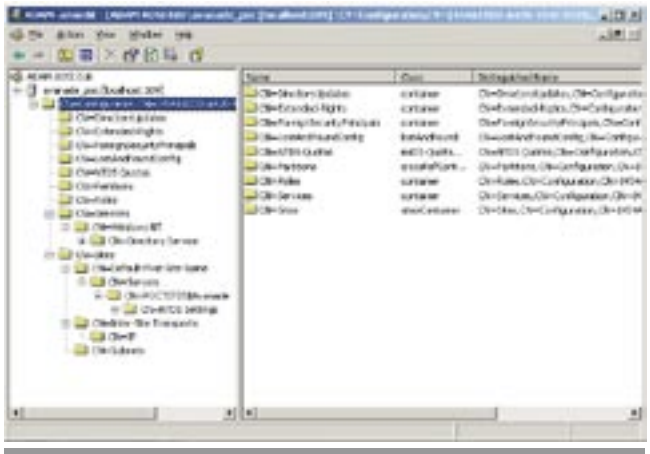
```
<connectionStrings>
   <add name="AdamProviderConnection"
   connectionString="LDAP://<server>:<port>/<CN=..>/<DC=..>" />
</connectionStrings>

<system.web>
   <membership defaultProvider="AdamProvider" userIsOnlineTimeWindow="15">
      <providers>
         <add name="AdamProvider"
         type="System.Web.Security.ActiveDirectoryMembershipProvider,
               System.Web, Version=2.0.0.0, Culture=neutral,
               PublicKeyToken=b03f5f7f11d50a3a"
         connectionStringName="AdamProviderConnection"
         connectionProtection="None"
         connectionUsername="domein\gebruikersnaam"
         connectionPassword="wachtwoord"
         enablePasswordRetrieval="false"
         enablePasswordReset="false"
         requiresQuestionAndAnswer="false"
         requiresUniqueEmail="false"
         passwordFormat="Hashed"
         description="AdamMembership" />
      </providers>
   </membership>
</system.web>
```

Codevoorbeeld 1.
Web.config-instellingen voor het gebruik van ActiveDirectoryMembershipProvider

Ook kun je later handmatig via de Web Site Administration Tool items wijzigen. Gebruikers toevoegen kan zelfs via deze tools. Voor forms-authenticatie is het bijvoorbeeld erg praktisch om ad-hoc accounts op te kunnen geven via deze manier. Rollen, koppelingen van gebruikers en rollen en zelfs toegangsregels voor gebruikers of rollen kunnen allemaal webbased in security-configuratie van de applicatie worden ingesteld.

Profile

In ASP.NET 1.x-applicaties worden gebruikergegevens vaak opgeslagen in een sessie en opgeslagen indien nodig. Dit proces is vaak inefficiënt en vereist ontwikkeling van generieke code. ASP.NET 2.0 helpt deze activiteiten te verminderen door het aanbieden van een nieuwe profile-dienst. Een profile is een set gegevens die bij een gebruiker hoort. Het profile kan diverse datatypes bevatten zolang het maar serialiseerbaar is. Beheer van het profile wordt voor je gedaan. ASP.NET 2.0 regelt het aanmaken en opslaan van een profile automatisch.
Het hart van de profile-dienst is de profile provider. De profile provider bevat alle code en structuur die nodig zijn om gebruikersgegevens te laden en op te slaan in een data store. Net als de membership provider is de profile provider zodanig ontworpen dat je hem kunt vervangen door een provider die aan je eigen wensen



Afbeelding 10. ADAM beheren met ADSI Edit

Attribuut	Beschrijving
name	Een unieke identificerende naam
type	Een .NET-type of een fully qualified class
serializeAs	Formaat van de waarde als opgeslagen in gegevensbron
allowAnonymous	Sta anoniem gebruik van deze property al of niet toe
provider	De provider die deze property beheert. Deze instelling overschrijft waarden opgegeven in andere .config bestanden
defaultValue	Standaard waarde als hij nog niet is toegekend
readOnly	Alleen lezen waarde

Tabel 4. Profile property-attributen

voldoet. In tabel 4 staan de property-attributen die een custom profile kan hebben, waarvan name en type verplicht zijn. In codevoorbeeld 2 staan een eenvoudige en een complexe definitie van custom properties. Na het opslaan van de gewijzigde web.config zijn de profile-properties direct zichtbaar in de code editor van Visual Studio, inclusief type en onderliggende eigenschappen.

Autorisatie

Autorisatie is het toekennen en beperken van rechten van een geauthenticeerde gebruiker. In het .NET Framework kan autorisatie plaatsvinden op basis van gebruiker, rol of http-actie.

Eerder zagen we dat via de configuration-editors diverse toegangsregels kunnen worden ingesteld. Via het objectmodel kan dit echter veel uitgebreider en dynamischer.
De ASP.NET Role Management Service gebruikt ook het providermodel om functionaliteit te scheiden van uiteindelijke opslag. Rollen kunnen dezelfde opslag gebruiken als die je gebruikt voor de membership service en gebruikers-profile. Toch heb je de mogelijkheid om een afwijkende provider op te geven.

Custom Role Management Providers

Je kunt ook je eigen role management provider maken, die toestaat dat je je eigen opslagmedium gebuikt voor rolinformatie. Ook hier begin je met het overerven van de abstracte RoleProvider-class en het implementeren van methodes die een role manager moet hebben; tabel 5. Nadat je een eigen role provider hebt gemaakt, kan je de applicatie configureren zodat je provider gebruikt wordt. Het role managementsysteem zal dan automatisch jouw provider en zijn methodes gaan gebruiken.

Geen grenzen

Tijdens het behandelen van een aantal nieuwe mogelijkheden in Microsoft ASP.NET 2.0 is duidelijk geworden dat de invoering van

Methode	Beschrijving
AddUsersToRoles	Voegt bepaalde gebruikers aan bepaalde rollen toe
CreateRole	Verwerkt het verzoek om het wachtwoordvraag en –antwoord van een gebruiker te wijzigen
DeleteRole	Voegt een nieuwe gebruiker toe aan de databron
FindUsersInRole	Verwijdert een gebruiker van de databron
GetAllRoles	Geeft een array met alle rollen
GetRolesForUser	Geeft een array met rollen van een gebruiker
GetUsersInRole	Geeft een array van gebruikers met een bepaalde rol
IsUserInRole	Geeft aan of een gebruiker een bepaalde rol heeft
RemoveUsersFromRoles	Verwijdert bepaalde rollen bij bepaalde gebruikers
RoleExists	Geeft aan of een rol bestaat

Tabel 5. RoleProvider API-methodes

```
<anonymousIdentification enabled="true" />
<profile>
  <properties>
    <add name="holding"
    type="string" />
    <add name="nicknames"
    type="System.Collections.Specialized.StringCollection"
    serializeAs="Xml"
    allowAnonymous="true"
    provider="SQL" />
  </properties>
</profile>
```

Codevoorbeeld 2.
Web.config-instellingen van een aangepast profile

```
<authorization>
  <allow verbs="GET" users="*"/>
  <allow verbs="POST" users="pete"/>
  <deny verbs="POST" users="*"/>
</authorization>
<roleManager
  defaultProvider="SQL"
  enabled="true"
  cacheRolesInCookie="true" >
</roleManager>
```

Codevoorbeeld 3.
Web.config-instellingen van autorisatie

het provider-model veel flexibiliteit en uitbreidbaarheid biedt aan ontwikkelaars. Het gebruik van Active Directory Application Mode kan in combinatie met de diverse nieuwe personaliseringsmogelijkheden voordelen opleveren voor stabiliteit, schaalbaarheid en ontwikkeltijd.

Aram Smith is architect in system engineering bij Avanade. Hij heeft ruime ervaring met infrastructurele onderwerpen als active directory en producten als Exchange en Sharepoint.

Pieter de Bruin is solution developer bij Avanade. Sinds het begin van 2002 is hij bezig met ontwikkelen en ontwerpen in en het coachen van .NET. Hij is MCSD.NET en Master CRM Developer gecertificeerd. Zijn e-mail adres is pieterd@avanade.com

Nuttige internetadressen

http://www.microsoft.com/windowsserver2003/adam/default.mspx
http://lab.msdn.microsoft.com/vs2005/
http://forums.asp.net/144/ShowForum.aspx