Sitecore CMS 6

# Membership Providers

*A Developer's Guide to Integrating Authentication Systems with Sitecore*

## Table of Contents

# Chapter 1

# Introduction

This document provides instructions to integrate authentication system with Sitecore using ASP.NET membership providers.[1] Developers can use these instructions to replace or augment the default Sitecore membership provider with one or more providers that authenticate users against identity management solutions other than Sitecore.

This document contains the following chapters:

- Chapter 1 – Introduction
- Chapter 2 – Implementing an ASP.NET Membership Provider

[1] For more information about ASP.NET membership providers, see http://sdn.sitecore.net/Articles/Security/Low_level_Sitecore_Security_and_Custom_Providers.aspx and http://msdn.microsoft.com/en-us/library/tw292whz.aspx. For more information about Sitecore security, see the Security Reference manual at http://sdn.sitecore.net/Reference/Sitecore%206/Security%20Reference.aspx. For information about relevant APIs, see the Security API Cookbook at http://sdn.sitecore.net/Reference/Sitecore%206/Security%20API%20Cookbook.aspx.

# Chapter 2

# Implementing an ASP.NET Membership Provider

This chapter provides instructions to implement an ASP.NET membership provider.

This chapter contains the following sections:

- Security Providers Overview
- Membership Providers Overview
- Minimal Read-Only Membership Provider Overview
- Read/Write Membership Provider Methods
- Optional Membership Provider Properties
- Configuring Membership Providers

## 2.1 Security Providers Overview

Providers abstract the implementation details of systems that provide similar facilities. There are three types of security providers:

- **Membership providers** abstract user authentication.
- **Role providers** abstract group or role membership.
- **Profile providers** abstract user profiles.

A system used for authentication may also manage roles and profiles, but a membership provider cannot expose those aspects of the system. Role providers expose relationships between users and roles. You can implement a role provider to expose role services, or update the default role provider with user-role and role-role relationship data from the external system. You can implement a profile provider to expose custom profile attributes, or update the default profile provider with profile data from the external system.[2]

**Note**
Membership providers, role providers, and profile providers are security providers. Role providers are not membership providers. Membership providers authenticate users. Role providers associate users and roles with roles.

[2] For more information about custom profile properties, see http://sdn.sitecore.net/Reference/Sitecore%206/Security%20API%20Cookbook.aspx.

## 2.2 Membership Providers Overview

Membership providers abstract user authentication services such as password validation. Membership providers implement methods to manage users in the repository abstracted by the provider, such as creating, updating, and deleting users, as well as validating and optionally recovering or resetting user credentials.

Authentication systems store basic user properties such as username, email address, and password, typically encrypted. Role providers and profile providers expose additional information about users authenticated through membership providers. The default Sitecore membership, role, and profile providers store information in adapted ASP.NET membership tables in the Core database.

**Note**
You can configure the Sitecore security providers to use an alternate database by updating the relevant `connectionStringName` attributes in `web.config`.

To use Sitecore to authenticate users in an existing system, you can implement a read-only membership provider as described in the section Minimal Read-Only Membership Provider Overview. To manage users through the ASP.NET membership provider APIs and the Sitecore User Manager, implement the methods required to support the required operations as described in the section Read/Write Membership Provider Methods.

**Note**
If you use third-party software for authentication, that vendor may provide an ASP.NET membership provider that you can use.

## 2.3 Minimal Read-Only Membership Provider Overview

This section describes the minimal methods that you must override to implement a read-only ASP.NET membership provider.

**Warning**
If a required user does not exist, such as the anonymous user in one or more domains, Sitecore will invoke the `CreateUser()` method of the membership provider. If you do not override the `CreateUser()` method, then you may need to create the required users manually. If all required users exist and you do not create users through the membership provider, then you do not need to implement the `CreateUser()` method. For more information about the `CreateUser()` method, see the section The CreateUser() Method.

**Note**
If accessed, all unimplemented provider methods and properties should throw an exception if accessed. For example:

```
public override string GetUserNameByEmail(string email)
{
  throw new System.NotImplementedException();
}
```

### 2.3.1 The GetUser() Method

Override the `System.Web.Security.MembershipProvider.GetUser()` method to retrieve `System.Web.Security.MemberShipUser` objects through the membership provider.[3]

- The **`username`** parameter indicates the user name, which does not include the domain name or the backslash character (“`\`”).
- The **`userIsOnline`** parameter indicates whether the user is online. You can use this value within the `GetUser()` method to update a record that indicates the date and time of the last activity by the user.

### 2.3.2 The ValidateUser() Method

Override the `System.Web.Security.MembershipProvider.ValidateUser()` method to validate a password in order to authenticate a user.[4]

- The **`username`** parameter indicates the user name, which includes the domain name and the backslash character (`domain\username`).
- The **`password`** parameter indicates the password attempted by the user.

[3] For more information about the `System.Web.Security.MembershipProvider.GetUser()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.getuser.aspx.
[4] For more information about the `System.Web.Security.MembershipProvider.ValidateUser()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.validateuser.aspx.

## 2.4 Read/Write Membership Provider Methods

You can implement additional membership provider methods to support the features described in the following sections.

### 2.4.1 The CreateUser() Method

You can override the membership provider `CreateUser()` method to create users in the authentication system through the membership provider.[5]

- The **`username`** parameter specifies the username for the new user, including the domain and backslash character (`domain\username`).
- The **`password`** parameter specifies the password for the new user in plain text.
- The **`email`** parameter specifies the email address of the user.
- The **`passwordQuestion`** and **`passwordAnswer`** parameters specify the security question or the identifier of a security question) and answer you can use to partially validate the user when resetting a password.
- This **`isApproved`** parameter is True.
- The **`providerUserKey`** parameter is the identifier of the membership provider.
- The **`status`** parameter provides information to the calling method if Sitecore is not able to create the user.

### The Sitecore.Security.Accounts.User.Profile.IsAdministrator Property

The `Sitecore.Security.Accounts.User.Profile.IsAdministrator` property indicates whether a user is a Sitecore administrator. If your membership provider authenticates CMS users and can indicate that some users are CMS administrators, the `GetUser()` method of your membership provider can set the `Sitecore.Security.Accounts.User.Profile.IsAdministrator` property accordingly. For example:

```
string username = @"domain\\username"; // TODO: set from external system
bool isAdmin = true; // TODO: set from external system
Sitecore.Security.Accounts.User sitecoreUser =
  Sitecore.Security.Accounts.User.FromName(username, true);

if (sitecoreUser.Profile.IsAdministrator != isAdmin)
{
  sitecoreUser.Profile.IsAdministrator = isAdmin;
  sitecoreUser.Profile.Save();
}
```

**Warning**
Ensure the System.Web.Security.Membership.MembershipUser exists before setting the `Sitecore.Security.Accounts.User.Profile.IsAdministrator` property.

[5] For information about the `System.Web.Security.MembershipProvider.CreateUser()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.createuser.aspx.

### 2.4.2 The Initialize() Method

You can override the membership provider `Initialize()` method to initialize the membership provider.[6] The `EnsureAnonymousUsers` processor in the `initialize` pipeline defined in `web.config` is typically responsible for causing ASP.NET to invoke the `Initialize()` method of the membership provider.

### 2.4.3 The GetAllUsers() Method

You can implement the membership provider `GetAllUsers()` method to return groups of users, such as to support a user interface that pages through the entire list of users.[7]

### 2.4.4 The ChangePasswordQuestionAndAnswer() method

You can implement the membership provider `ChangePasswordQuestionAndAnswer()` method to update the security question and answer associated with the specified user if the specified password is valid for that user.[8]

### 2.4.5 The GetPassword() Method

If your membership provider supports password recovery, you can implement the membership provider `GetPassword()` method to return the password of the specified user if the specified answer matches the answer to the security question associated with the user.[9]

### 2.4.6 The ChangePassword() Method

You can implement the membership provider `ChangePassword()` method to change the password associated with the specified user if the specified old password matches the password associated with the specified user.[10]

[6] For more information about the `System.Web.Security.MembershipProvider.Initalize()` method, see http://msdn.microsoft.com/en-us/library/system.configuration.provider.providerbase.initialize.aspx.
[7] For more information about the `System.Web.Security.MembershipProvider.GetAllUsers()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.getallusers.aspx.
[8] For more information about the `Sytem.Web.Security.MembershipProvider.ChangePasswordQuestionAndAnswer()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.changepasswordquestionandanswer.aspx.
[9] For more information about the `System.Web.Security.MembershipProvider.GetPassword()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.getpassword.aspx.
[10] For more information about the `Sytem.Web.Security.MembershipProvider.ChangePassword()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.changepassword.aspx.

### 2.4.7 The ResetPassword() Method

You can implement the membership provider `ResetPassword()` method to reset the password associated with the specified user to a random password if the specified answer matches the answer to the security question associated with the user.[11]

### 2.4.8 The UpdateUser() Method

You can implement the membership provider `UpdateUser()` method to update the user record in the authentication system as specified by the `System.Web.Security.MembershipUser`.[12]

### 2.4.9 The UnlockUser() Method

If the membership provider supports locking users out after some number of invalid password attempts, you can implement the membership provider `UnlockUser()` method to unlock the specified user.[13]

### 2.4.10 The GetUser() Method

You can implement an additional signature of the membership provider `GetUser()` method.[14]

### 2.4.11 The GetUserNameByEmail() Method

You can implement the `GetUserNameByEmail()` method to return the username associated with an email address.[15]

### 2.4.12 The DeleteUser() Method

You can implement the membership provider `DeleteUser()` method to delete the specified user from the authentication system through the provider.[16]

[11] For more information about the `System.Web.Security.MembershipProvider.ResetPassword()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.resetpassword.aspx.
[12] For more information about the `System.Web.Security.MembershipProvider.UpdateUser()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.updateuser.aspx.
[13] For more information about the `System.Web.Security.MembershipProvider.UnlockUser()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.unlockuser.aspx.
[14] For more information about the `System.Web.Security.MembershipProvider.GetUser()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.aspx.
[15] For more information about the `System.Web.Security.Membership.GetUserNameByEmail()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.getusernamebyemail.aspx.
[16] For more information about the `System.Web.Security.MembershipProvider.DeleteUser()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.deleteuser.aspx.

### 2.4.13 The GetNumberOfUsersOnline() Method

You can implement the membership provider `GetNumberOfUsersOnline()` method to return the number of users currently online.[17]

### 2.4.14 The FindUsersByEmail() Method

You can implement the membership provider `FindUsersByEmail()` method to return groups of users with a specific email address, such as to implement a user interface that pages through the such as list of users.[18]

[17] For more information about the `System.Web.Security.MembershipProvider.GetNumberOfUsersOnline()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.getnumberofusersonline.aspx.
[18] For more information about the `System.Web.Security.MembershipProvider.GetAllUsers()` method, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.getallusers.aspx.

## 2.5 Optional Membership Provider Properties

You can override the following membership provider properties to support relevant functionality.

### 2.5.1 The ApplicationName Property

You can override the membership provider `ApplicationName` property to return the `applicationName` attribute of the membership provider.[19]

### 2.5.2 The EnablePasswordRetrieval Property

You can implement the membership provider `EnablePasswordRetrieval` property to indicate whether the membership provider supports password retrieval.[20]

### 2.5.3 The EnablePasswordReset Property

You can implement the membership provider `EnablePasswordReset` property to indicate whether the membership provider supports resetting passwords.[21]

### 2.5.4 The RequiresQuestionAndAnswer Property

You can implement the membership provider `RequiresQuestionAndAnswer` property to indicate whether the membership provider requires a security question and answer in order to reset the password for a user.[22]

### 2.5.5 The MaxInvalidPasswordAttempts Property

You can implement the membership provider `MaxInvalidPasswordAttempts` property to indicate the maximum invalid password attempts allowed before locking out a user.[23] A locked user cannot log in to the system.

---

[19] For more information about the `System.Web.Security.MembershipProvider.ApplicationName` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.applicationname.aspx.

[20] For more information about the `System.Web.Security.MembershipProvider.EnablePasswordRetrieval` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.enablepasswordretrieval.aspx.

[21] For more information about the `System.Web.Security.MembershipProvider.EnablePasswordReset` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.enablepasswordreset.aspx.

[22] For more information about the `System.Web.Security.MembershipProvider.RequiresQuestionAndAnswer` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.requiresquestionandanswer.aspx.

[23] For more information about the `System.Web.Security.MembershipProvider.MaxInvalidPasswordAttempts` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.maxinvalidpasswordattempts.aspx.

### 2.5.6 The PasswordAttemptWindow Property

You can implement the membership provider `PasswordAttemptWindow` property to indicate the number of minutes in which a maximum number of invalid password or password-answer attempts are allowed before the system locks the user out.[24]

### 2.5.7 The RequiresUniqueEmail Property

You can implement the membership provider `RequiresUniqueEmail` property to indicate whether the authentication system requires that each user have a unique email address.[25]

**Tip**
In the membership provider `CreateUser()` method, if the `RequiresUniqueEmail` property is True, you can use the `System.Web.Security.GetUserNameByEmail()` method in the to determine whether a user with the specified email address exists.

### 2.5.8 The PasswordFormat Property

You can implement the membership provider `PasswordFormat` property to return a `System.Web.Security.MembershipPasswordFormat` value indicating a password storage format such as Clear, Encrypted, or Hashed.[26]

### 2.5.9 The MinRequiredPasswordLength Property

You can implement the membership provider `MinRequiredPasswordLength` property to return the minimum number of characters required for a valid password.[27]

### 2.5.10 The MinRequiredNonAlphanumericCharacters Property

You can implement the membership provider `MinRequiredNonAlphanumericCharacters` property to return the number of non-alphanumeric characters required for a valid password.[28]

---

[24] For more information about the `System.Web.Security.MembershipProvider.PasswordAttemptWindow` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.passwordattemptwindow.aspx.
[25] For more information about the `System.Web.Security.MembershipProvider.RequiresUniqueEmail` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.requiresuniqueemail.aspx.
[26] For more information about the `System.Web.Security.MembershipProvider.PasswordFormat` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.passwordformat.aspx.
[27] For more information about the `System.Web.Security.MembershipProvider.MinRequiredPasswordLength` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.minrequiredpasswordlength.aspx.
[28] For more information about the `System.Web.Security.MembershipProvider.MinRequiredNonAlphanumericCharacters` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.minrequirednonalphanumericcharacters.aspx.

### 2.5.11 The PasswordStrengthRegularExpression Property

You can implement the membership provider `PasswordStrengthRegularExpression` property to specify a regular expression that valid passwords must match.[29]

[29] For more information about the `System.Web.Security.MembershipProvider.PasswordStrengthRegularExpression` property, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.passwordstrengthregularexpression.aspx.

## 2.6 Configuring Membership Providers

You can the procedures in this section to configure ASP.NET membership providers.[30]

### 2.6.1 How to Use the Same Membership Provider for All Domains

To implement an ASP.NET membership provider for all users in all domains:

1. Implement the `System.Web.Security.MembershipProvider` class and the required methods.[31]
2. Add a `/configuration/system.web/membership/providers/add` element to `web.config` that exposes your membership provider class.
3. Change the value of the `realProviderName` attribute of the `/configuration/system.web/membership/providers/add` element in `web.config` with a `name` attribute containing the value `sitecore` to the value of the `name` attribute of your `<provider>` element.

For example:

```
<membership defaultProvider="sitecore">
  <providers>
    <clear />
    <add name="sitecore" realProviderName="namespace" ... />
    <add name="namespace" type="Namespace.Web.Security.NamespaceMembershipProvider"
      applicationName="sitecore" minRequiredPasswordLength="1"
      minRequiredNonalphanumericCharacters="0" requiresQuestionAndAnswer="false"
      requiresUniqueEmail="false" maxInvalidPasswordAttempts="256" />
...
```

**Note**
The `applicationName` attribute must be `sitecore`.

### 2.6.2 How to Use Different Membership Providers for Different Domains

Each membership provider can manage users in multiple domains. In the default Sitecore configuration, a single membership provider manages all users in all domains. A single membership provider must manage all users in an individual domain; two membership providers cannot manage users in a single domain.

You can configure the switching membership provider to use different membership providers for users in different security domains. For example, you can use the default membership provider in the content delivery environment and the Sitecore Active Directory membership to authenticate for Content Management (CM) users.[32] You can authenticate users of each logical site against different security domains using different membership providers.

To use a custom provider for Extranet security and the default provider for Sitecore security:

[30] For more information about configuring membership providers, see http://msdn.microsoft.com/en-us/library/1b9hw62f.aspx.
[31] For more information about the `System.Web.Security.MembershipProvider` class, see http://msdn.microsoft.com/en-us/library/system.web.security.membershipprovider.aspx.
[32] For more information about the Sitecore Active Directory provider, see http://sdn.sitecore.net/products/ad.aspx.

1. Follow the instructions provided in the section How to Use the Same Membership Provider for All Domains.
2. In the `/configuration/system.web/membership` element in `web.config`, change the `defaultProvider` attribute to `switcher`.
3. Within the `/configuration/sitecore/switchingProviders/membership` element in `web.config`, add the following `<provider>` element, where the value of the `providerName` attribute matches the value of the `name` attribute of the `<add>` element created previously:

```
<provider providerName="namespace" domains="extranet" storeFullNames="false"
  wildcard="*" />
```

**Note**
The order of the `<provider>` elements in `web.config` controls the order in which they appear in the User Manager and elsewhere, but has no other effect.