Implementing a Membership Provider

.NET Framework 4 Other Versions

ASP.NET membership is designed to enable you to easily use a number of different membership providers for your ASP.NET applications. You can use the supplied membership providers that are included with the .NET Framework, or you can implement your own providers.

There are two primary reasons for creating a custom membership provider.

- You need to store membership information in a data source that is not supported by the membership providers included with the .NET Framework, such as a FoxPro database, an Oracle database, or other data sources.
- You need to manage membership information using a database schema that is different from the database schema used by the providers that ship with the .NET Framework. A common example of this would be membership data that already exists in a SQL Server database for a company or Web site.

Required Classes

To implement a membership provider, you create a class that inherits the MembershipProvider abstract class from the System.Web.Security namespace. The MembershipProvider abstract class inherits the ProviderBase abstract class from the System.Configuration.Provider namespace, so you must implement the required members of the ProviderBase class as well. The following tables list the required properties and methods that you must implement from the ProviderBase and MembershipProvider abstract classes and a description of each. To review an

implementation of each member, see the code supplied for the Sample Membership Provider Implementation.

Required ProviderBase Members

Member	Description
Initialize method	Takes, as input, the name of the provider and a NameValueCollection of configuration settings. Used to set property values for the provider instance including implementation-specific values and options specified in the configuration file (Machine.config or Web.config) supplied in the configuration.

Required MembershipProvider Members

Member	Description
EnablePasswordReset property	A **Boolean** valu configuration fil The EnablePass indicates wheth ResetPassword current passwo generated pass This property is
EnablePasswordRetrieval property	A **Boolean** valu configuration fil The EnablePass indicates wheth password using This property is
RequiresQuestionAndAnswer property	A **Boolean** valu configuration fil The RequiresQu indicates wheth password answ password using or reset their pa ResetPassword This property is
RequiresUniqueEmail property	A **Boolean** valu configuration fil The RequiresUr

	indicates wheth unique e-mail a a user. If a user source for the c CreateUser met status value of I This property is
PasswordFormat property	A MembershipF specified in the (Web.config). The PasswordFc format that pas Passwords can **Encrypted**, and **Clear** password which improves password stora secure, as pass data source is c passwords are can be decrypte or password re additional proc and retrieval bu passwords are data source is c passwords are hash algorithm salt value when When a passwo with the salt val verification. Has retrieved. You can use the DecryptPasswor MembershipPrc decrypt passwo your own encry EncryptPasswor virtual methods class, **Encrypte** using the key in machineKey ele This property is
MaxInvalidPasswordAttempts property	An **Integer** valu configuration fil The MaxInvalidF conjunction with PasswordAttem an unwanted so or password ar through repeate

	of invalid passw questions suppl exceeds the Ma within the numb the PasswordAt membership us the IsLockedOu user is unlocked method. If a val answer is suppl MaxInvalidPassv the counter that invalid attempts If the RequiresQ is set to **false**, i attempts are nc Invalid passwor attempts are tra ChangePasswor ChangePasswor GetPassword, a This property is
PasswordAttemptWindow property	An **Integer** valu configuration fil For a descriptio MaxInvalidPassv This property is
ApplicationName property	The name of the membership inf configuration fil ApplicationNam source with rela used when quer See the section later in this topi This property is the ApplicationN explicitly.
MembershipProvider.CreateUser method	Takes, as input, password, and inserts a new us the data source returns a Memb populated with newly created u also defines an Basic, you can u MembershipCre indicates wheth created, or a re successfully cre

	The CreateUser ValidatingPassw MembershipVal has been specif cancels the crea results of the ev OnValidatingPas execute the spe MembershipVal
UpdateUser method	Takes, as input, populated with updates the dat values.
DeleteUser method	Takes, as input, deletes that use data source. Th returns **true** if t deleted; otherw **Boolean** param whether related such as role or deleted.
ValidateUser method	Takes, as input, password and v those in the dat method returns name and pass **false**.
GetUser method	Takes, as input, a **Boolean** valu update the Last user to show th online. The Getl MembershipUs current values f specified user. I in the data sour returns **null** (**No**
GetUser method	Takes, as input, value indicating LastActivityDate that the user is GetUser metho object populate the data source user name is nc the GetUser me in Visual Basic).

GetAllUsers method	Returns a Mem populated with for all of the us The results retu constrained by parameters. The identifies the m MembershipUs MembershipUs parameter iden to return, where The *totalRecord* parameter that membership us were in the data and the *pageIna pageSize* of 5, th MembershipUs contain the sixth returned. *totalR*
GetNumberOfUsersOnline method	Returns an integ all the users in t LastActivityDate date and time n UserIsOnlineTim UserIsOnlineTim integer value sp minutes to use user is online.
ResetPassword method	Takes, as input, password answ random passwo The ResetPassw user informatio new password password as a mechanism for password is the of the Members The ResetPassw the EnablePassw **true** before per EnablePasswor NotSupportedE ResetPassword value of the Re property. If the RequiresQuesti **true**, the ResetP value of the sup against the stor

	data source. If t MembershipPas The ResetPassw ValidatingPassw MembershipVal has been specif generated pass cancels the rese the results of th OnValidatingPas execute the spe MembershipVal
GetPassword method	Takes, as input, password answ password for th source and retu **string**. GetPassword er EnablePasswor **true** before per EnablePasswor an ProviderExce The GetPasswo value of the Re property. If the RequiresQuesti **true**, the GetPas value of the sup against the stor data source. If t MembershipPas
GetUserNameByEmail method	Takes, as input, returns the first source where th the supplied *em* If no user name mail address, a If multiple user a particular e-m user name foun
ChangePassword method	Takes, as input, password, and updates the pas the supplied us password are v method returns updated succes The ChangePas ValidatingPassw MembershipVal has been specif

	cancels the cha on the results o OnValidatingPa execute the spe MembershipVa
ChangePasswordQuestionAndAnswer method	Takes, as input, password ques answer, and up question and ar the supplied us valid. The ChangePasswor method returns question and ar successfully; oth If the supplied u are not valid, **fa**
FindUsersByName method	Returns a list of the user name c supplied *userna* configured App if the *username* "user," then the "user3," and so support is inclu source. Users a order by user n The results retu are constrained *pageSize* param parameter iden MembershipUs MembershipUs parameter iden to return, where The *totalRecord* parameter that membership us *usernameToMat* users were four matched part o and the *pageIna* *pageSize* of 5, th MembershipUs the sixth throug *totalRecords* wo
FindUsersByEmail method	Returns a list of the user name c supplied *emailT* ApplicationNam *emailToMatch* p

	"address@exan the e-mail addr "address1@exa "address2@exa returned. Wildc based on the d returned in alph name. The results retu are constrained *pageSize* param parameter iden MembershipUs MembershipUs *pageIndex* parar of results to ret first page. The *t* `out` parameter number of mem the *emailToMat* users were four matched part o and the *pageIna* *pageSize* of 5, th MembershipUs the sixth throug *totalRecords* wo
UnlockUser method	Takes, as input, the field in the d IsLockedOut pr UnlockUser met record for the r successfully; oth

ApplicationName

Membership providers store user information uniquely for each application. This enables multiple ASP.NET applications to use the same data source without running into a conflict if duplicate user names are created. Alternatively, multiple ASP.NET applications can use the same user data source by specifying the same ApplicationName.

Because membership providers store user information uniquely for each application, you will need to ensure that your data schema includes the application name and that queries and updates also include the application name. For example, the following command is used to retrieve a user name from a database, based on the e-mail address, and ensures that the ApplicationName is included in the query.

included in the query.

```
SELECT Username FROM MyUserTable
  WHERE Email = 'someone@example.com' AND
```

Custom Members

You may need to extend the membership provider interfaces with additional functionality not provided by the ProviderBase and MembershipProvider abstract classes. Any public members that you add to your membership provider will be accessible using the Provider or Providers property of the Membership class.

An example of this could be a `LockUser` method that sets the IsLockedOut property to **true**. The following example shows how to cast the Provider property, which exposes the default membership provider for an application, as a custom-provider type in order to call the custom `LockUser` method.

C# VB

```
MyCustomProvider p = (MyCustomProvider)Me
p.LockUser(username);
```

Thread Safety

For each membership provider specified in the configuration for an application, ASP.NET instantiates a single membership provider instance that is used for all of the requests served by an HttpApplication object. As a result, you can have multiple requests executing concurrently. ASP.NET does not ensure the thread safety of calls to your provider. You will need to write your provider code to be thread safe. For example, creating a connection to a database or opening a file for editing should be done within the member that is called, such as MembershipProvider.CreateUser, rather than opening a file or database connection when the Initialize method is called.

See Also

Reference
ValidatePasswordEventArgs
OnValidatingPassword
Concepts
Sample Membership Provider Implementation
Securing ASP.NET Site Navigation
Other Resources
Managing Users by Using Membership
ASP.NET Security