

## ASP.NET 2.0 Security Enhancements

By Kristen McGregor

ASP.NET 1.1 brought web security to a new level by creating built-in support for user authentication. ASP.NET 2.0 has preserved the concept, but focused on increasing developmental flexibility. With the addition of seven new server-side controls, membership classes to retrieve and update user information within a database, and role management services, ASP.NET 2.0 transformed the user configuration setup process into an easy, non-time-consuming procedure. The foundation of these new security enhancements lie in the provider model.

### Provider Model

The provider model was created to centralize state storage architecture. It offers abstraction to the developer, allowing the developer to determine how and where the actual data storage takes place, rather than being forced to use a particular data model or fixed API behavior. ASP .NET 2.0 has several built-in providers. The two providers related directly to security are the Membership and Role Management providers.

### Membership Provider

The MembershipProvider class is one of the key abstract classes in ASP.NET 2.0 that supplies inherent security of the developmental process. ASP.NET 2.0 provides two built-in membership provider classes: ActiveDirectoryMembershipProvider and SqlMembershipProvider. The provider class is registered in the <providers> section under Membership of the application's configuration file: the web.config file.

The built-in login server controls make the code that is behind user configuration invisible to the developer. The CreateUserWizard control, for example, handles the addition of a new user. Dragging this control onto the web page produces the following:



Sign Up for Your New Account

User Name:

Password:

Confirm Password:

E-mail:

Security Question:

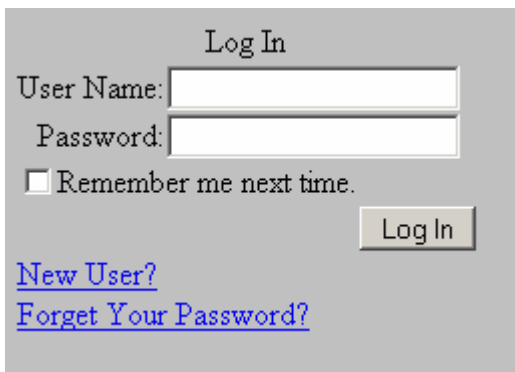
Security Answer:

The CreateUserWizard control offers seven customizable templates. It also provides functionality to send an email when a new user is created, as configured in the following:

```
<asp:CreateUserWizard id="CreateUserWizard1" runat="server">
  <MailDefinition
    BodyFileName="NewUserEmail.txt"
    From="welcome@denimgroup.com"
    Subject="Welcome to the site!"/>
</asp:CreateUserWizard>
```

The email settings are defined in the <smtpMail> mail section of the web.config file.

Once a user is created, the Login control contains functionality to authenticate the user. Optional parameters for the Login control include CreateUserText, CreateUserUrl, and more.



Additional controls are LoginName, which outputs the user name of the currently logged user and LoginStatus, which will display “Login” or “Logout” accordingly. The LoginView control allows for message display depending on the status and/or roles of the user.

The PasswordRecovery control’s function is self-explanatory. It first prompts for a user name, and then prompts for the answer to the security question entered during login. Finally, the ChangePassword control allows users to change their passwords. Both the PasswordRecovery and ChangePassword controls can be configured to send password confirmation emails.

This increased functionality for the user cuts development time in half and results in a more inherently secure application.

### **Membership class**

Though the controls handle the majority of desired functionality, ASP.NET 2.0 provides public methods of the Membership class to expand the developer’s control. A few of these

include `CreateNewUser`, `DeleteUser`, `FindUsersByEmail`, `FindUsersByName`, `GeneratePassword`, `GetAllUsers`, `GetNumberOfUsersOnline`, `GetUser`, `UpdateUser` and `ValidateUser`. These members reference the `MembershipUser` class, which represents a user stored in the Membership data system. Furthermore, the `MembershipProvider` class can be extended to create a custom provider.

### **Role Provider**

Another provider type introduced in ASP.NET 2.0 is the Role provider. Similar to Membership, the Role Provider is defined in the `web.config` file. The Role Management provider offers three built-in providers: `AuthorizationStoreRoleProvider`, `SqlRoleProvider` and `WindowsTokenRoleProvider`. Unlike ASP.NET 1.1, where developers were responsible for implementing custom roles through ASP.NET HTTP pipelines, the built-in Role Manager requires only 2 lines of code:

```
Roles.CreateRole("NewRole");  
Roles.AddUserToRole("UserName", "NewRole");
```

The Roles class offers other methods, such as `GetRolesForUser`, `GetUsersInRoles`, `IsUserInRole`, and `RemoveUserFromRole` to provide the developer with greater control and increase development efficiency.

The Roles class provides a role caching property to optionally store the user's roles in an encrypted HTTP cookie. If the user's roles do not fit in the cookie, the cookie will store the most frequently used roles. When role checking is performed, if the role is not stored in the encrypted cookie, the system uses the Role API to determine if the user belongs to that role. If so, it will replace the last role stored in the cookie with the most recently requested role. This automation of role caching provides efficient role lookup and removes this responsibility from the developer.

### **Additional Security Advancements**

In addition to the provider model, ASP .NET 2.0 introduced other security benefits, the first being cookieless form authentication. ASP.NET 1.1 provided built-in support for forms authentication; however, the process required the browser to support cookies. This slowed down the process of developing by forcing developers to provide code to warn users of browsers that do not support cookies. One of the most common requests for ASP.NET 2.0 was cookieless form authentication. The attribute "cookieless," contained within the `<forms>` section of the `web.config` file can be set to `UseCookies`, `UseUri`, `UseDeviceProfile`, or `AutoDetect`. `UseCookies` and `UseUri` forces the authentication ticket to be stored in a cookie or the URL, respectively. `UseDeviceProfile` determines what to use based on the device profile settings within the `machine.config` file. `AutoDetect` uses cookies if supported; otherwise the ticket is stored in the URL. Due to this improvement, developers can authenticate forms more efficiently with fewer roadblocks.

To increase security, ASP.NET 2.0 introduced the `connectionStringName` property of the `web.config` file. Rather than storing the database connection string in the `web.config` file,

the database connection string is stored within the machine.config file. The connectionStringName property value points to the section within the machine.config file that contains the appropriate connection string.

### **Web Application Administrator**

ASP.NET 2.0 did not just make the application coding process easier for the developer, but took one step further by providing a web browser interface to configure the application. The Web Application Administrator is available through Visual Studio at [http://localhost:\[port\]/asp.netwebadminfiles](http://localhost:[port]/asp.netwebadminfiles). It provides an interface to edit features of the xml configuration file, such as the authentication method, provider and data source. The Web Application Administrator automatically generates a SQL Server Express database containing the necessary tables for user configuration (these tables can be generated in a SQL server through ASP.NET 2.0's packaged aspnet\_regsql.exe tool). The Web Application Administrator also offers a security page to manage roles, users and permissions, thus offering a convenient way to initialize the application during the first stages of development.

### **Conclusion**

ASP.NET 2.0 diminished the time required for user configuration considerably. Though ASP.NET 1.1 set the foundation with some built-in support, ASP.NET 2.0 evolved the user configuration process into minimal step procedures, while still giving the developer flexibility to adapt the components as needed.