Membership & Role Management

Introduction

- Forms authentication identifies users & control access to web pages with authorization.

- Not complete solution

  - One need to maintain a user list and check it during the authentication process.

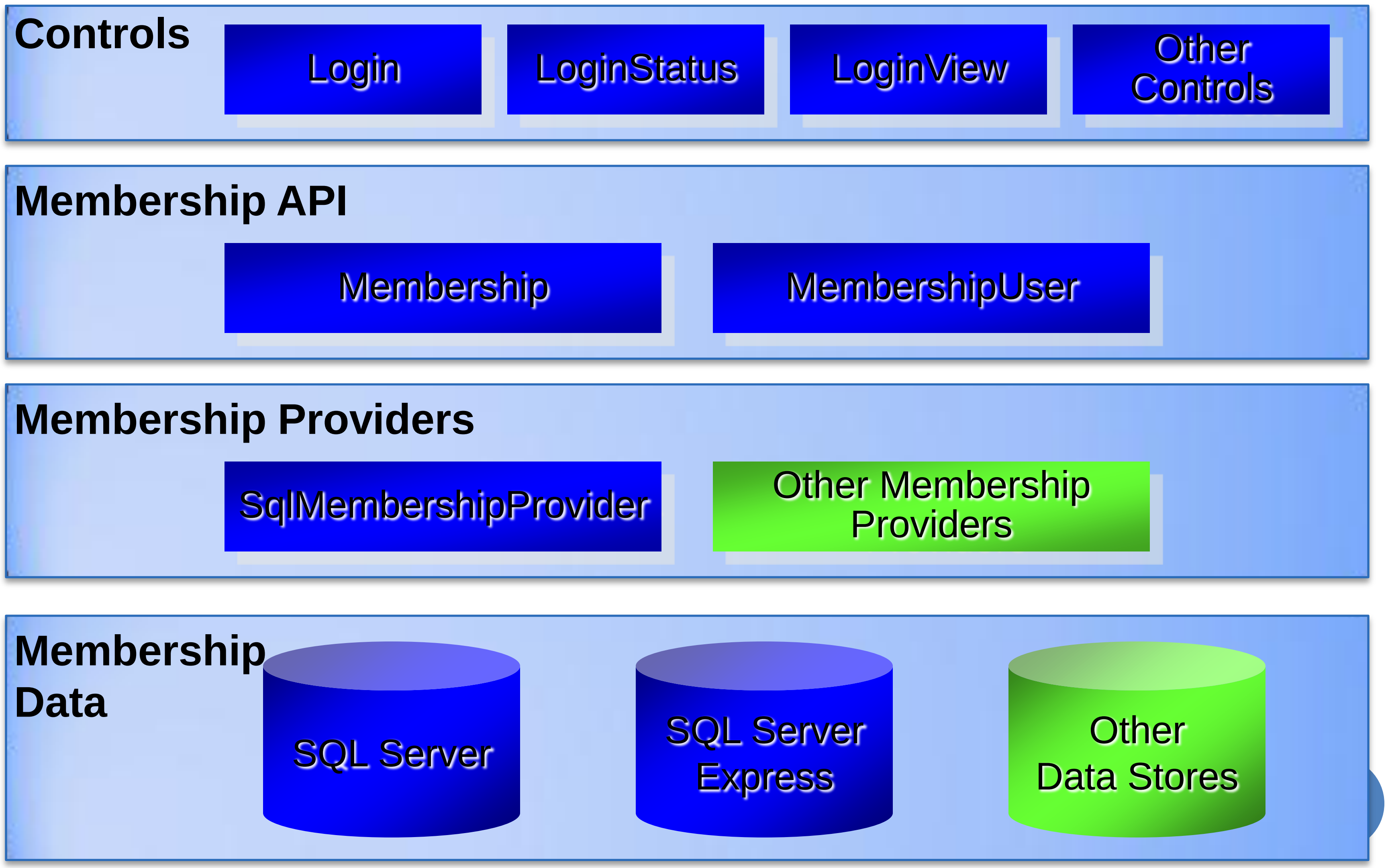
  - Developer required to build & manage the back-end mechanics of the overall system.

- ASP.NET 2.0 has *membership and role management service,* to take care of the login, authentication, authorization, and management of users.

MEMEBERSHIP SERVICE

- Manages users and credentials
  - Declarative access via WS Admin Tool
  - Programmatic access via Membership API
- Simplifies forms authentication
  - Provides logic for validating user names and passwords, creating users, and more
  - Manages data store for credentials, e-mail addresses, and other membership data
- Provider-based for flexible data storage

Membership Schema



Membership Providers

- Membership is provider-based
  - Provider provides interface between Membership service and data store
- Ships with one membership provider
  - SqlMembershipProvider (SQL Server and SQL Server Express)
- Use custom providers for other Membership data stores

Membership Provider

- By default, membership is enabled for every new website one creates.

- The default membership provider assumes

  - Membership database in SQL Server2K5 Express Edition

  - SQL Server2K5 instance name is SQLEXPRESS

  - Membership data store file aspnetdb.mdf is in App_Data folder of web application.

Membership Provider Configuration

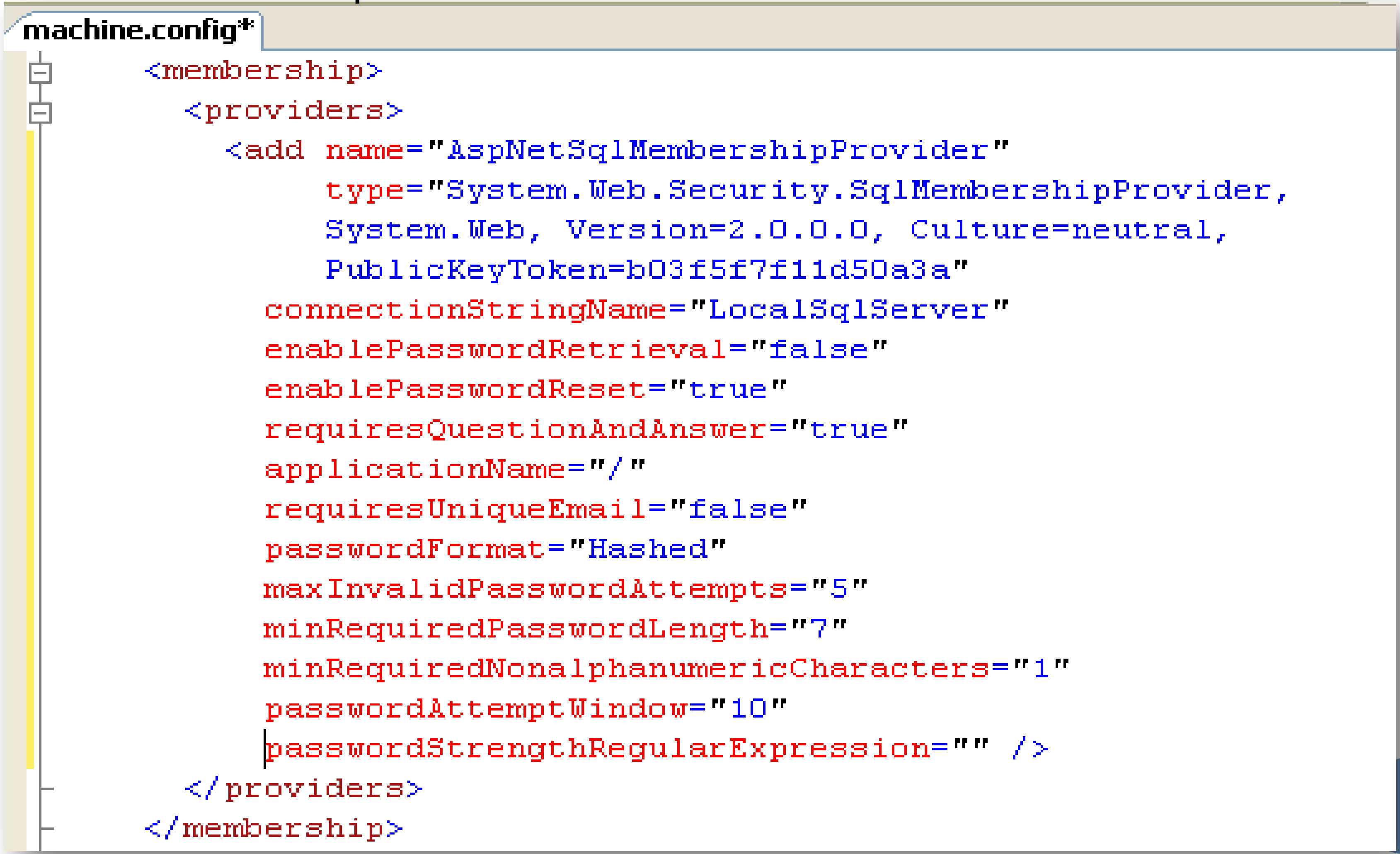
- For single web application, edit web.config instead of machine.config.

- One need to remove all the existing connection strings using the <clear> element.

- Add the connection string.

Membership Provider Configuration

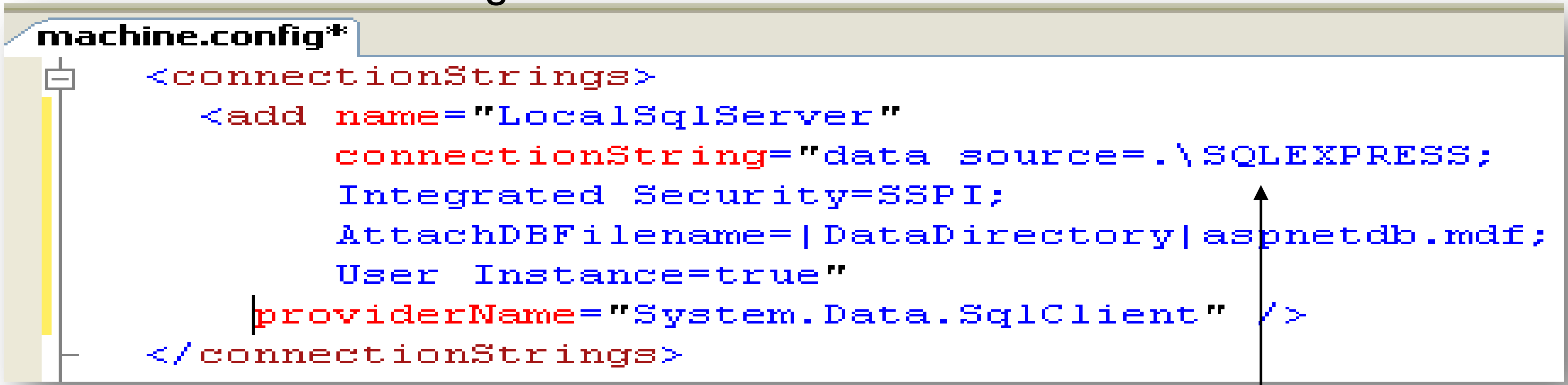
Default Membership Provider



```
<membership>
  <providers>
    <add name="AspNetSqlMembershipProvider"
         type="System.Web.Security.SqlMembershipProvider,
         System.Web, Version=2.0.0.0, Culture=neutral,
         PublicKeyToken=b03f5f7f11d50a3a"
      connectionStringName="LocalSqlServer"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="true"
      applicationName="/"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      maxInvalidPasswordAttempts="5"
      minRequiredPasswordLength="7"
      minRequiredNonalphanumericCharacters="1"
      passwordAttemptWindow="10"
      passwordStrengthRegularExpression="" />
  </providers>
</membership>
```

Membership Provider Configuration

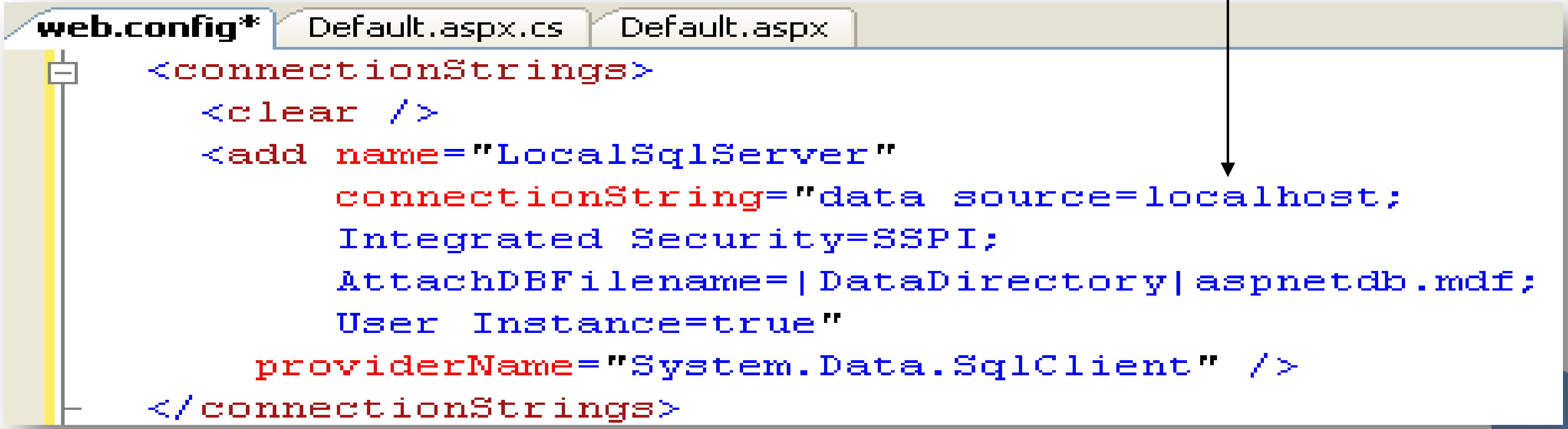
Default Connection String



machine.config*

```
<connectionStrings>
  <add name="LocalSqlServer"
       connectionString="data source=.\SQLEXPRESS;
       Integrated Security=SSPI;
       AttachDBFilename=|DataDirectory|aspnetdb.mdf;
       User Instance=true"
     providerName="System.Data.SqlClient" />
</connectionStrings>
```

Connection String Modified for SQL Server full version



web.config* | Default.aspx.cs | Default.aspx

```
<connectionStrings>
  <clear />
  <add name="LocalSqlServer"
       connectionString="data source=localhost;
       Integrated Security=SSPI;
       AttachDBFilename=|DataDirectory|aspnetdb.mdf;
       User Instance=true"
     providerName="System.Data.SqlClient" />
</connectionStrings>
```

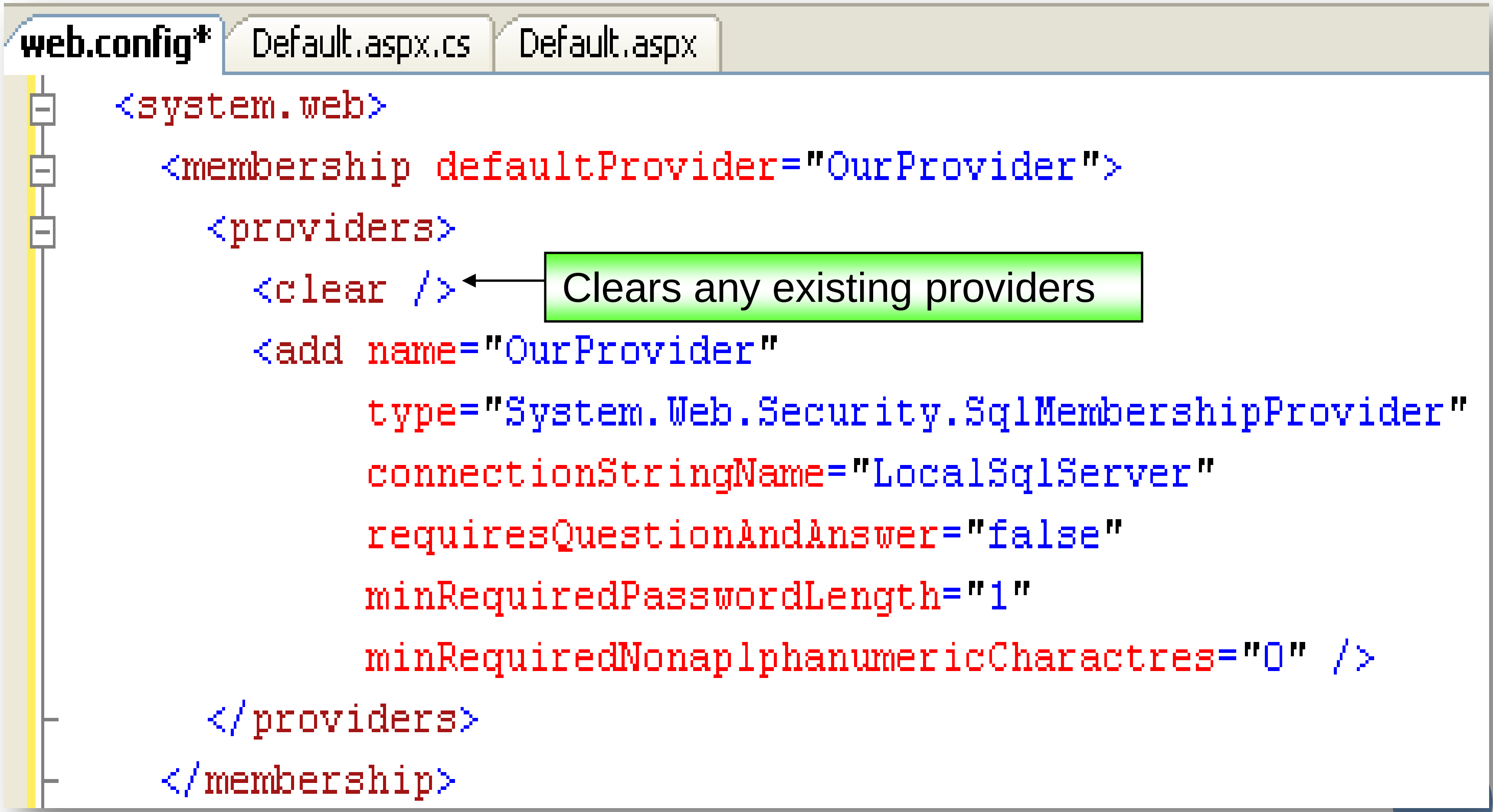
Older Version of SQL Server

- *AttachDbFileName* option in connection string is not available.
- Supply name of database from server.
- Run aspnet_regsql.exe command line tool to generate aspnetdb database.

Custom Membership Provider Config

- Define a new membership provider with our custom settings using <membership> element in web.config.
- Set the defaultProvider attribute of the <membership> element so it refers to our membership provider by name.

Custom Membership Provider Config



```
<system.web>
  <membership defaultProvider="OurProvider">
    <providers>
      <clear />
      <add name="OurProvider"
           type="System.Web.Security.SqlMembershipProvider"
           connectionStringName="LocalSqlServer"
           requiresQuestionAndAnswer="false"
           minRequiredPasswordLength="1"
           minRequiredNonaplphanumericCharactres="0" />
    </providers>
  </membership>
```

Membership Configuration Attributes

* denotes must attributes

- name*
  - Specifies a name for the membership provider.
- type*
  - The type of membership provider.
- connectionStringName*
  - The name of the connection string referring to a connection string defined in the <connectionStrings> section of web.config or machine.config.

Membership Configuration Attributes

- passwordFormat
  - Defines the format in which the password is stored in the data store.
  - The possible values include *Hashed* (SHA1), *Clear*, and *Encrypted* (3DES).
- minRequiredPassordLength
  - Specifies the minimum length of a password.
- minNonAlphanumericCharacters
  - Specifies the number of nonalphanumeric characters (characters other than numbers and letters) the password needs to have.

Membership Configuration Attributes

- maxInvalidPasswordAttempts
  - Specifies the number of times a user can supply an invalid password for their login before the user account is locked and made inaccessible.
- passwordAttemptWindow
  - The internal time in which maxInvalidPasswordAttempts is measured. e.g. if you set a window of 30 minutes, after 30 minutes the number of invalid password attempts is reset.

Membership Configuration Attributes

- enablePasswordRetrieval
  - Determines whether a password can be requested (and e-mailed to the user), which is useful if a user forgets a password. This feature is never supported if passwordFormat is set to Hashed.
- enablePasswordReset
  - Determines whether a password can be reset, which is useful if a password is forgotten.

Membership Configuration Attributes

- requiresQuestionAndAnswer
  - Determines whether the membership security answer will be required when you request or reset a user password.
- requiresUniqueEmail
  - If false, more than one user can have the same e-mail address.

Manual Creation of Tables for Membership

- By default, the *aspnet_regsql* tool creates database named *aspnetdb* & installs tables that can be used for user authentication, role-based authorization, profiles, and Web Parts personalization.

- One can specify exactly what database name should be & what tables to install using command-line switches.

ASPNET_REGSQL SWITCHES

- -S <servername>
  - Specifies the location of the SQL Server instance where you want to install the database.
- -E
  - Connects to the server through Windows authentication.
- -U <user> -P <pwd>
  - Specifies the user name and password needed to connect to the SQL Server database.

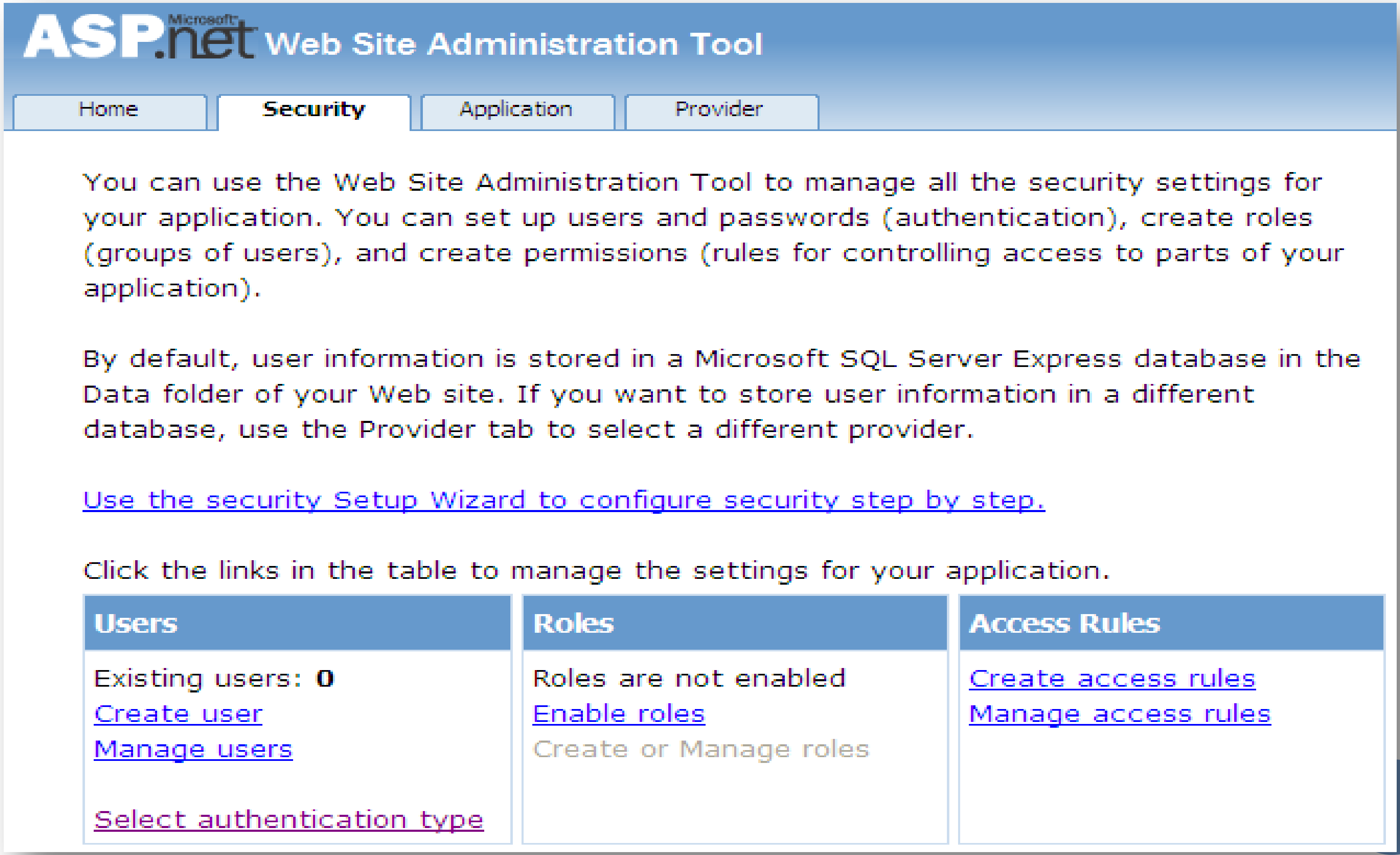
ASPNET_REGSQL SWITCHES

- -A
  - Specifies the features onw want to use.
  - Valid options for this switch are all, m (membership), r (role-based security), p (profiles), c (Web Part personalization), and w (for database cache dependencies with SQL Server 2000).
- -R
  - Removes the databases specified by the -A switch.

ASPNET_REGSQL SWITCHES

- -d <databasename>
  - Allows one to specify the name of the database.
- -sqlexportonly
  - Creates SQL scripts for the specified options but doesn’t actually create the tables in the database.

Website Administration Tool (WAT)



You can use the Web Site Administration Tool to manage all the security settings for your application. You can set up users and passwords (authentication), create roles (groups of users), and create permissions (rules for controlling access to parts of your application).

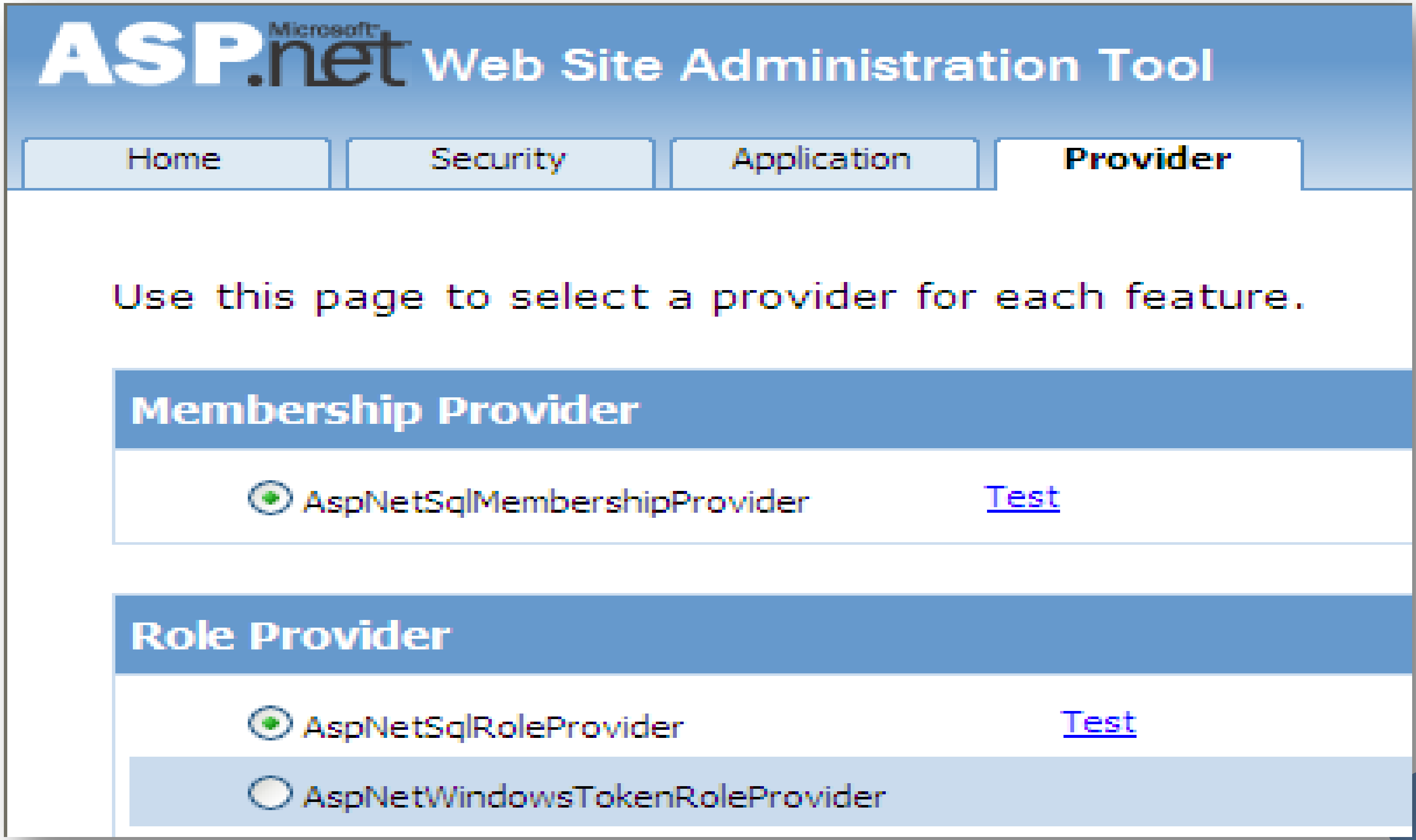
By default, user information is stored in a Microsoft SQL Server Express database in the Data folder of your Web site. If you want to store user information in a different database, use the Provider tab to select a different provider.

Use the security Setup Wizard to configure security step by step.

Click the links in the table to manage the settings for your application.

Users	Roles	Access Rules
Existing users: **0** Create user Manage users Select authentication type	Roles are not enabled Enable roles Create or Manage roles	Create access rules Manage access rules

Website Administration Tool (WAT)



Use this page to select a provider for each feature.

Membership Provider

[✓ AspNetSqlMembershipProvider] Test

Role Provider

[✓ AspNetSqlRoleProvider] Test

AspNetWindowsTokenRoleProvider

Membership Class

- Provides static methods for performing key membership tasks
  - Creating and deleting users
  - Retrieving information about users
  - Generating random passwords
  - Validating logins
- Includes read-only static properties for acquiring data about provider settings.

Key Membership Methods

Name	Description
CreateUser	Adds a user to the membership data store
DeleteUser	Removes a user from the membership data store
GeneratePassword	Generates a random password of a specified length
GetAllUsers	Retrieves a collection of MembershipUser objects representing all currently registered users
GetUser	Retrieves a MembershipUser object representing a user
UpdateUser	Updates information for a specified user
ValidateUser	Validates logins based on user names and passwords

Creating New User

```
try {
    Membership.CreateUser (“Ram", "iconnect", “ram@iconnectgroup.com");
}
catch (MembershipCreateUserException e) {
    // Find out why CreateUser failed
    switch (e.StatusCode) {

    case MembershipCreateStatus.DuplicateUsername:
      ...
    case MembershipCreateStatus.DuplicateEmail:
      ...
    case MembershipCreateStatus.InvalidPassword:
      ...
    default:
      ...
    }
}
```

Authenticating User

```
if (Membership.ValidateUser (UserName.Text, Password.Text))
    FormsAuthentication.RedirectFromLoginPage (UserName.Text,
        RememberMe.Checked);
```

MembershipUser Class

- Represents individual users registered in the membership data store
- Includes numerous properties for getting and setting user info
- Includes methods for retrieving, changing, and resetting passwords
- Returned by Membership methods such as *GetUser* and *CreateUser*.

Key MembershipUser Properties

Name	Description
Comment	Storage for user-defined data
CreationDate	Date user was added to the membership data store
Email	User's e-mail address
LastLoginDate	Date user last logged in successfully
LastPassword-ChangedDate	Date user's password was last changed
ProviderUserKey	Unique user ID generated by membership provider
UserName	User's registered user name

Key MembershipUser Methods

Name	Description
ChangePassword	Changes user's password
ChangePassword-QuestionAndAnswer	Changes question and answer used for password recovery
GetPassword*	Retrieves a password
ResetPassword**	Resets a password by setting it to a new random password
UnlockUser	Restores suspended login privileges

** Works if Membership.EnablePasswordRetrieval is true*

*** Works if Membership.EnablePasswordReset is true*

Restoring Login Privileges

```
MembershipUser user = Membership.GetUser ("Jeff");

if (user != null) {
    if (user.IsLockedOut) {
        user.UnlockUser ();

        // TODO: Optionally use MembershipUser.ResetPassword
        // to reset Jeff's password

    }
}
```

Security Controls

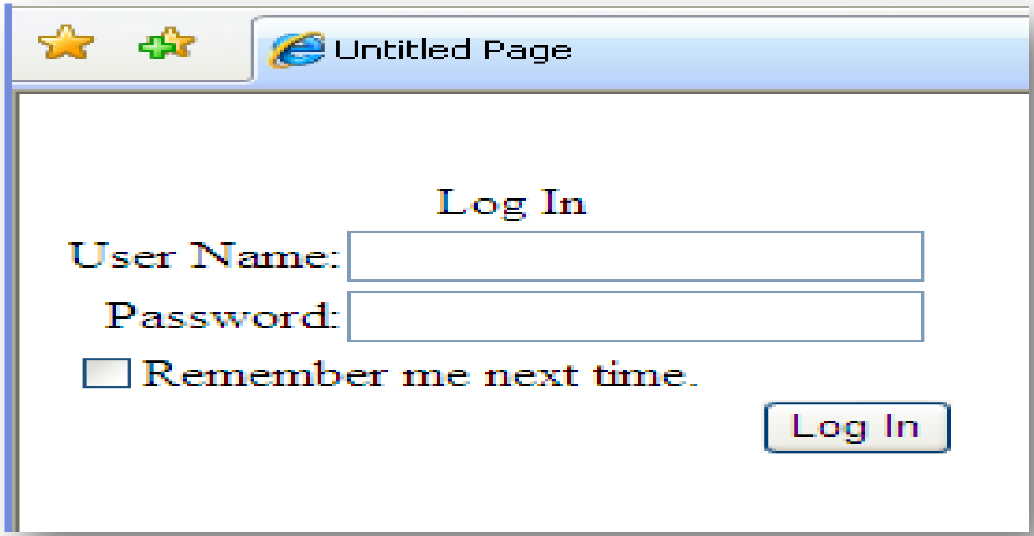
- *Membership* class can be used by components & controls to integrate them with ASP.NET Security.
- Login Control
- LoginStatus Control
- LoginName Control
- LoginView Control
- CreateUserWizard Control
- PasswordRecovery Control
- ChangePassword Control

Login Control

- Displays the familiar user name and password text boxes, with a login button.
- Standard UI for logging in users
- Integrates with Membership service
  - Calls ValidateUser automatically
  - No-code validation and logins
- Also works without Membership service
- Incorporates RequiredFieldValidators
- Highly customizable UI and behavior

Using Login Control

```
<html>
 <body>
  <form runat="server">
   <asp:Login ID="Login1" RunAt="server" />
  </form>
 </body>
</html>
```



Log In

User Name:

Password:

Remember me next time.

Key Login Properties

- TitleText
  - The text that's displayed in the heading of the control.
- InstrcutionText
  - The text that's displayed just below the heading.
- CreateUserText
  - Sets the text for a link to the user registration page.
  - If not supplied, no link displayed.

Key Login Properties

- CreateUserUrl
  - Supplies a URL to a user registration page.
- FailureText
  - The text that’s displayed when a login attempt fails.
- UserNameLabelText
  - The text that’s displayed before the user name text box.
- PasswordLabelText
  - The text that’s displayed before the password text box.

KEY LOGIN PROPERTIES

- UsernameRequiredErrorMessage
  - Error message to be displayed if the user doesn’t type in a user name. Default is *.
- PasswordRequiredErrorMessage
  - Error message to be displayed if the user doesn’t type in a password. Default is *.
- LoginButtonText
  - The text displayed for the login button.

Key Login Properties

- LoginButtonType
  - The type of button control that’s used as the login button. It can be displayed as Link, Button, or Image.
- LoginButtonImageUrl
  - Specifies image URL.
  - Used in conjunction with LoginButtonStyle.
- DestinationPageUrl
  - The page to which the user is redirected if the login attempt is successful.

Key Login Properties

- DisplayRememberMe
  - Determines whether the Remember Me check box will be shown.
- RememberMeSet
  - Sets the default value for the Remember Me check box. Default is false.
- CreatUserIconUrl
  - Supplies a URL to an image that will be displayed alongside the *CreateUserText* for the user registration link.

Key Login Properties

- PasswordRecoveryUrl
    - Supplies a URL to a password recovery page.
- PasswordRecoveryText
    - Sets the text for the link to the password recovery page.
    - If not supplied, link not displayed.
- PasswordRecoveryIconUrl
    - Supplies a URL to an image that will be displayed alongside the *PasswordRecoveryText*.

Key Login Properties

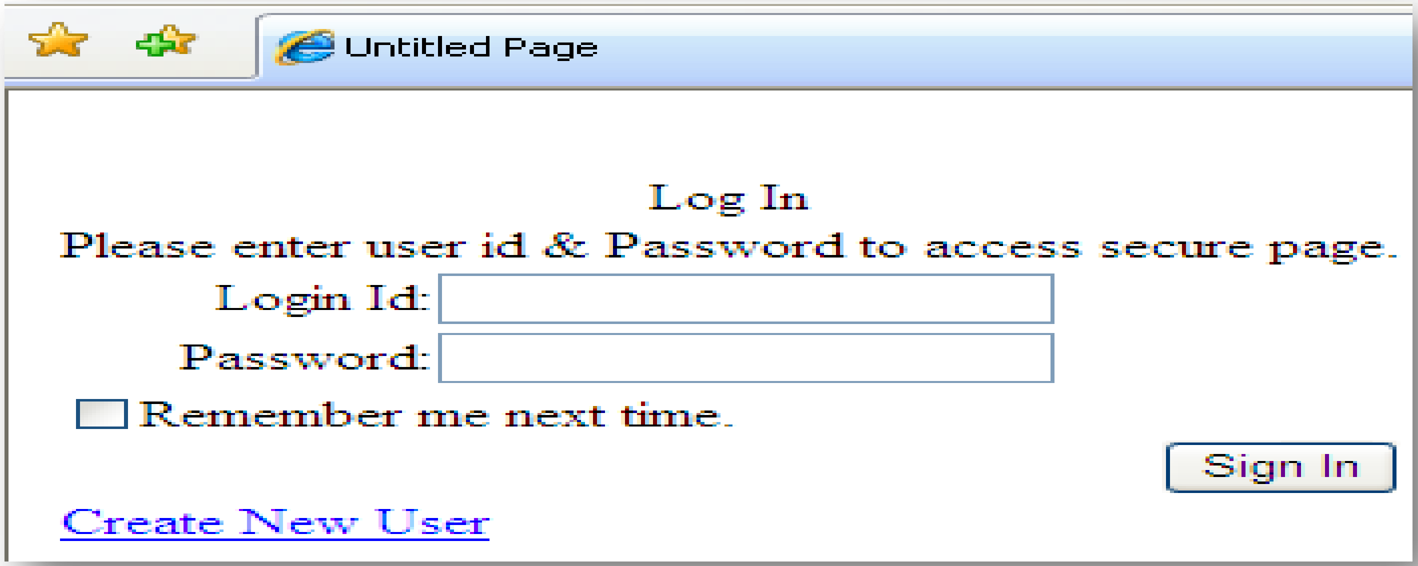
- HelpPageUrl
  - Supplies a URL to a page with help information.
- HelpPageText
  - Sets the text for the link to the help page.
- HelpPageIconUrl
  - Supplies a URL to an image that will be displayed alongside the *HelpPageText*.
- MembershipProvider
  - Specifies the membership provider name.

Login Style Properties

- TitleTextStyle
- LabelStyle
- TextBoxStyle
- LoginButtonStyle
- FailureStyle
- CheckBoxStyle
- ValidatorTextStyle
- HyperLinkStyle
- InstructionTextstyle

Customizing Login Control

```
<asp:Login ID="Login1" runat="server" CreateUserText="Create New User“
CreateUserUrl="~/Register.aspx“
InstructionText="Please enter user id & Password to access secure page.“
LoginButtonText="Sign In" UserNameLabelText="Login Id:">
        </asp:Login>
```


Untitled Page

Log In

Please enter user id & Password to access secure page.

Login Id:

Password:

Remember me next time.

Sign In

Create New User

Login Events

Name	Description
LoggingIn	Fired when the user clicks the Log In button. Purpose: to Prevalidate login credentials (e.g., make sure e-mail address is well-formed)
Authenticate	Fired when the user clicks the Log In button. Purpose: to Authenticate the user by validating his or her login credentials
LoggedIn	Fired following a successful login
LoginError	Fired when an attempted login fails

Authenticating the User

```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
  {
      if (Membership.ValidateUser(Login1.UserName, Login1.Password))
      {
          e.Authenticated = true;
      }
      else
      {
          e.Authenticated = false;
      }
  }
```

AuthenticateEventArgs's ***Authenticated*** property if set to true, fires ***LoggedIn*** event. Else fires ***LoginError*** event.

LoginStatus Control

- Displays links for logging in and out
  - "Login" to unauthenticated users
  - "Logout" to authenticated users
- UI and logout behavior are customizable

```
<asp:LoginStatus ID="LoginStatus1" Runat="server"
  LogoutAction="Redirect" LogoutPageUrl="~/Default.aspx" />
```

LoginStatus Properties

Name	Description
LoginText	Text displayed for login link (default="Login")
LogoutText	Text displayed for logout link (default="Logout")
LoginImageUrl	URL of image used for login link
LogoutAction	Action to take following logout: Redirect, RedirectToLoginPage, or Refresh (default)
LogOutPageUrl	URL of page to go to following logout if LogoutAction="Redirect"

LoginName Control

- Displays authenticated user names

- Use optional FormatString property to control format of output.

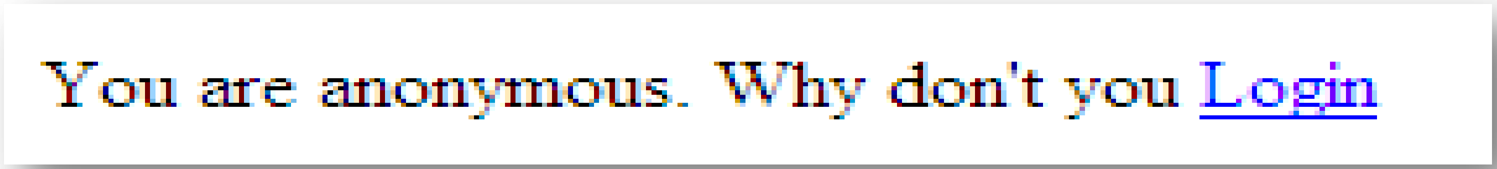
```
<asp:LoginView ID="LoginView1" Runat="server">
  <AnonymousTemplate>
    You are not logged in
  </AnonymousTemplate>
  <LoggedInTemplate>
    <asp:LoginName ID="LoginName1" Runat="server"
      FormatString="You are logged in as {0}" />
  </LoggedInTemplate>
</asp:LoginView>
```

LoginView Control

- Displays content differently to different users depending on:
  - Whether user is authenticated
  - If user is authenticated, the role memberships he or she is assigned
- Template-driven
  - <AnonymousTemplate>
  - <LoggedInTemplate>
  - <RoleGroups> and <ContentTemplate>

Using LoginView Control

```
<asp:LoginView ID="LoginView1" Runat="server">
  <AnonymousTemplate>
    You are anonymous. Why don't you <a href="Login.aspx">Login</a>
  </AnonymousTemplate>
  <LoggedInTemplate>
    <asp:LoginName ID="LoginName1" Runat="server"
      FormatString="You are logged in as {0}" />
  </LoggedInTemplate>
</asp:LoginView>
```

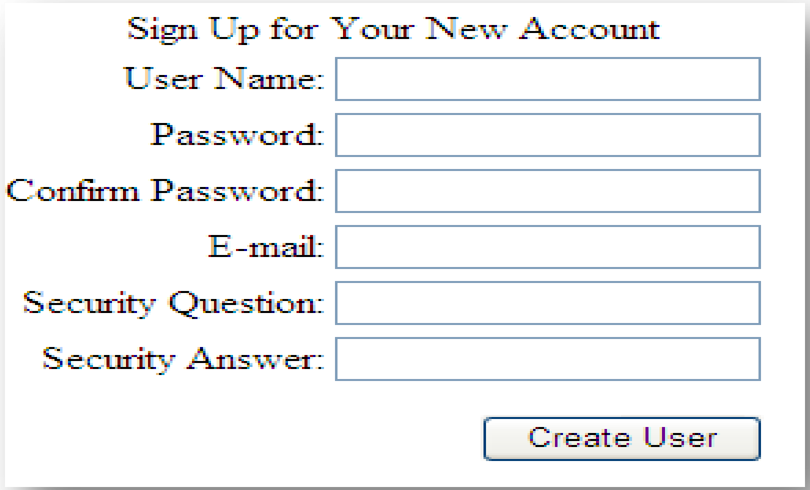

You are anonymous. Why don't you Login

CreateUserWizard Control

- Provides a native functionality for creating and configuring a new user using the membership API.
- If a web application has registration page, so that users can login.
- The registration page uses *CreateUserWizard* control.
- Inherits from *Wizard* control. Hence one can add as many extra wizard steps as required.

Using CreateUserWizard Control

```
<asp:CreateUserWizard ID="CreateUserWizard1" runat="server" >
        <WizardSteps>
          <asp:CreateUserWizardStep runat="server">
          </asp:CreateUserWizardStep>
          <asp:CompleteWizardStep runat="server">
          </asp:CompleteWizardStep>
        </WizardSteps>
    </asp:CreateUserWizard>
```



Sign Up for Your New Account

User Name:	
Password:	
Confirm Password:	
E-mail:	
Security Question:	
Security Answer:	

CreateUserWizard Events

- If someone decides not to use membership features, can use events:
  - CreatingUser
  - CreatedUser
  - CreateUserError
- Some more useful events:
  - NextButtonClick
  - FinshButtonClick
  - ContinueButtonClick
  - PreviousButtonClick
  - SideBarButtonClick
  - ActiveStepChanged

CreateUserWizard Events

- Some more useful events:
  - NextButtonClick
  - FinshButtonClick
  - ContinueButtonClick
  - PreviousButtonClick
  - SideBarButtonClick
  - ActiveStepChanged

CreateUserWizard Properties

- AutoGeneratePassword
  - Determines if the control autogenerates a password for the user.
- CancelDestinationPageUrl
  - The URL to redirect to when the cancel button is clicked.
- CompleteSuccessText
  - The text to be shown after the user has been created.

CreateUserWizard Properties

- ConfirmPasswordCompareErrorMessage
  - The text to be shown in the validation summary when the password and confirm password do not match.
- ContinueDestinationPageUrl
  - The URL to redirect to when the continue button is clicked.
- DisplayCancelButton
  - Determines whether cancel button is displayed.

CreateUserWizard Properties

- DisplaySideBar
  - Indicated whether sidebar is diplayed.
- DuplicateEmailErrorMessage
  - Text to be shown when duplicate e-mail error is returned from create user.
- DuplicateUserNameErrorMessage
  - Text to be shown when duplicate user name error is returned from create user.

PasswordRecoovery Control

- The control represents the form that enables a user to recover or reset a lost password.
- The user will receive the password through an e-mail message.
- Three Views
  - First asks for Username.
  - Second asks for Security Question Answer
  - Third informs success or failure.

PasswordRecovery Control

- The control works only if membership provider supports password retrieval.
- It also requires that the provider defines a *MembershipUser* object and implements the *GetUser* method.
- Control do not work if password is hashed.

<MailDefinition> Element

- The element is child to PasswordRecovery.
- The element configures the e-mail message and indicates the sender as well as the format of the body (text or HTML), priority, subject, and carbon-copy (CC).
- Properly configured SMTP server is required.

Using PasswordRecovery Control

```
<asp:PasswordRecovery ID="PasswordRecovery1" runat="server">
      <MailDefinition From=abc@iconnectgroup.com
            IsBodyHtml="false" Subject="Your Password" />
</asp:PasswordRecovery>
```

If user do not email or SMTP is not properly configures, using *SendingMail* event one can display password on page directly.

```
protected void
PasswordRecovery1_SendingMail(object sender, MailMessageEventArgs e)
   {
      e.Cancel = true;
      PasswordRecovery1.SuccessText = e.Message.Body;
   }
```

PasswordRecovery Events

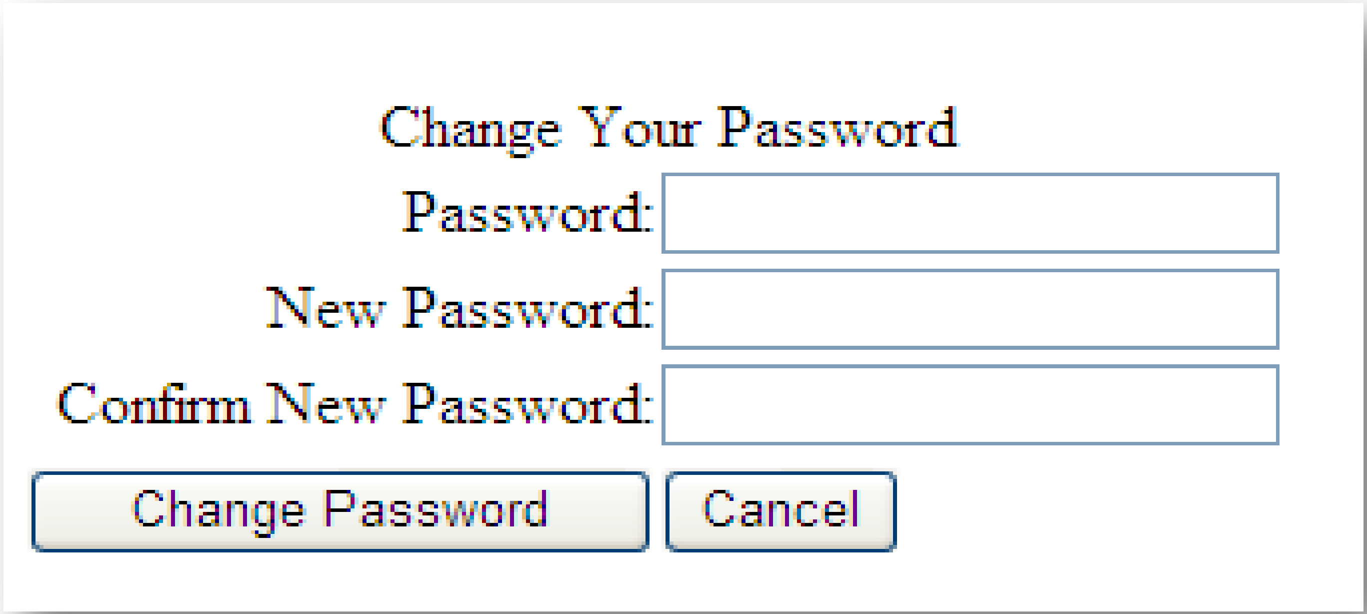
- UserLookupError
  - Raised when user is invalid.
- SendingMail
  - Raised before sending a mail.
- SendingMailError
  - Raised when there is error in sending mail.
- AnswerLookupError
  - Raised when the answer provided is incorrect.

ChangePassword Control

- The control enables end users to change their passwords directly in the browser.
- Only a logged-in user can change a password.
- The control works, if the *enablePasswordReset* attribute of the membership provider is set to true.

Using ChangePassword Control

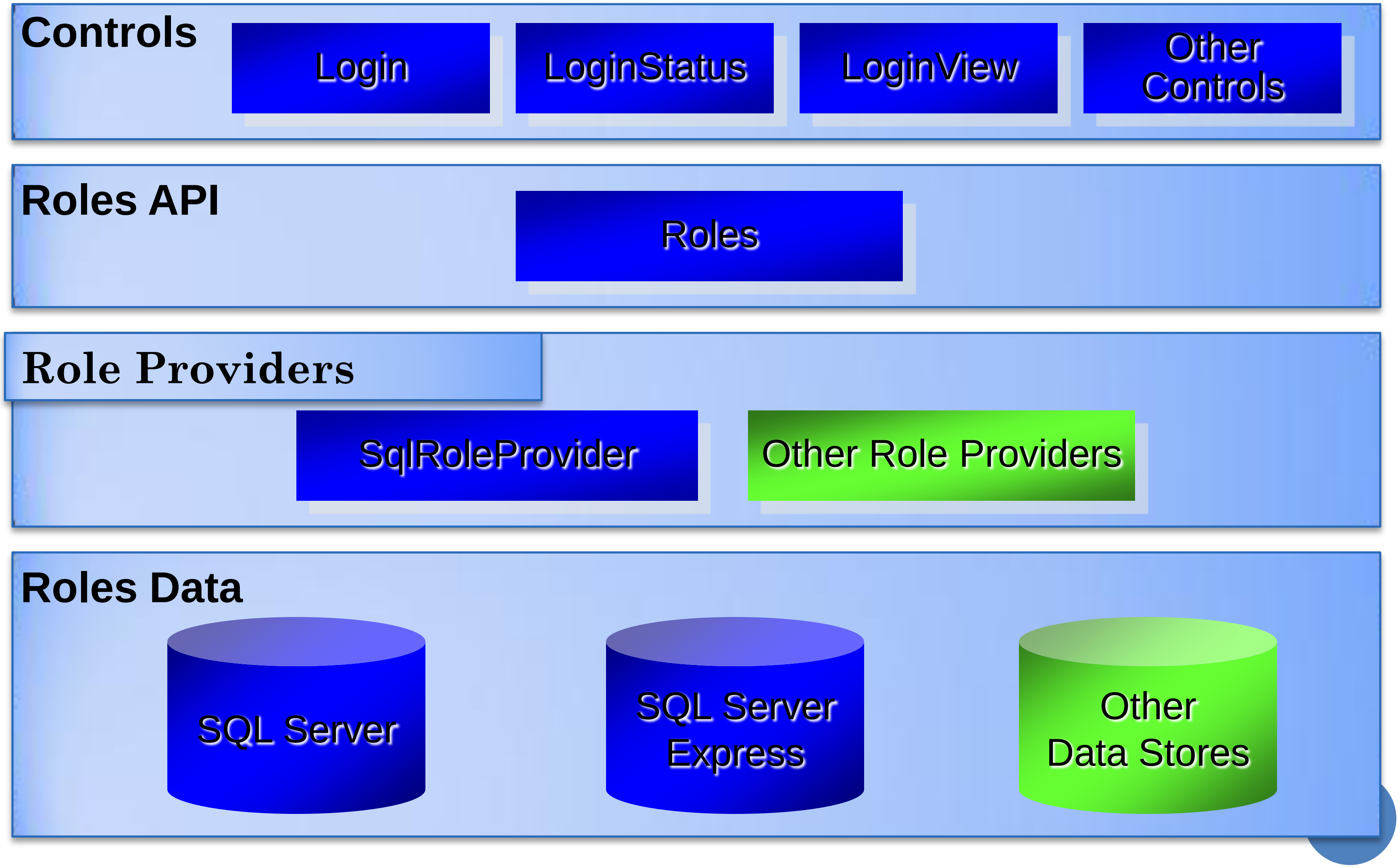
```
<asp:ChangePassword ID="ChangePassword1" runat="server">
        </asp:ChangePassword>
```



Role Management Service

- Role-based security in a box
  - Declarative access via WS Admin Tool
  - Programmatic access via Roles API
- Simplifies adding role-based security to sites that employ forms authentication
  - Maps users to roles on each request
  - Provides data store for role information
- Provider-based for flexible data storage.

Role Management Schema



Enabling Role-based Security

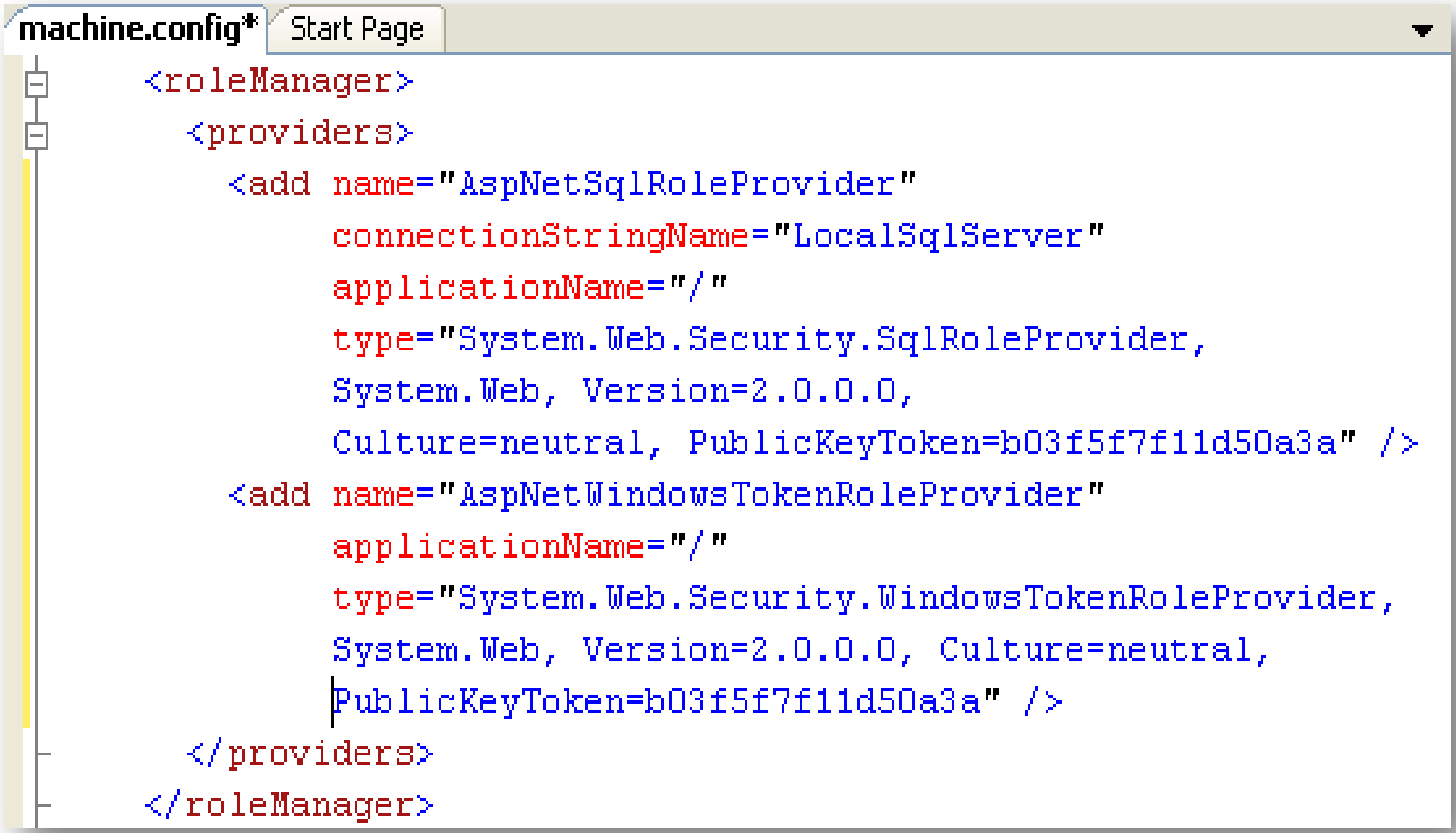
- Role manager is disabled by default.
- One can enable it using WAT.
- Or edit web.config and add line as follows:

```
<system.web>
		<roleManager enabled="true" />
</system.web>
```

Role Management Providers

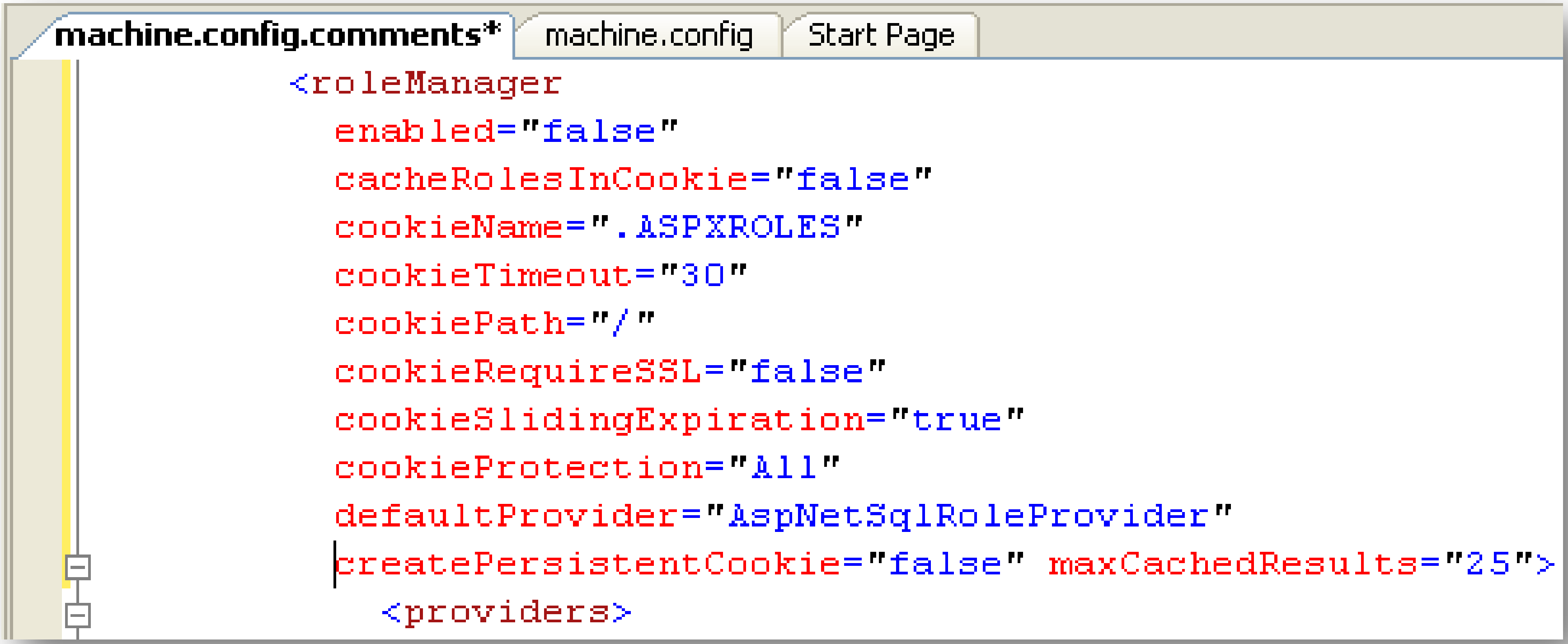
- Role management is provider-based
- Ships with three role providers:
  - AuthorizationStoreRoleProvider (Authorization Manager, or "AzMan")
  - SqlRoleProvider (SQL Server)
  - WindowsTokenRoleProvider (Windows)
- Use custom providers for other data stores.

Role Provider Configuration



```
<roleManager>
  <providers>
    <add name="AspNetSqlRoleProvider"
         connectionStringName="LocalSqlServer"
         applicationName="/"
         type="System.Web.Security.SqlRoleProvider,
         System.Web, Version=2.0.0.0,
         Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" />
    <add name="AspNetWindowsTokenRoleProvider"
         applicationName="/"
         type="System.Web.Security.WindowsTokenRoleProvider,
         System.Web, Version=2.0.0.0, Culture=neutral,
         PublicKeyToken=b03f5f7f11d50a3a" />
  </providers>
</roleManager>
```

Role Manager Configuration



```
<roleManager
  enabled="false"
  cacheRolesInCookie="false"
  cookieName=".ASPXROLES"
  cookieTimeout="30"
  cookiePath="/"
  cookieRequireSSL="false"
  cookieSlidingExpiration="true"
  cookieProtection="All"
  defaultProvider="AspNetSqlRoleProvider"
  createPersistentCookie="false" maxCachedResults="25">
    <providers>
```

ASP.NET 2.0 provides another two files machine.config.default and machine.config.comments. The machine.config.default acts as a backup for the machine.config file. The machine.config.comments file contains a description for each configuration section and explicit settings for the most commonly used values.

Role Manager Config Attributes

- Enabled
  - Defines whether the role management service is enabled for the application. Default is false.

- cacheRolesInCookie
  - Defines whether the roles of the user can be stored within a cookie on the client machine. (default is true).

- cookiName
  - Specifies the name used for the cookie sent to the end user for role management information storage.

Role Manager Config Attributes

- cookieTimeout
  - the amount of time (in minutes) after which the cookie expires. Default is 30.
- cookieRequireSSL
  - Defines whether you require that the role management information be sent over an encrypted connection (SSL).
- cookieSlidingExpiration
  - If set to False, the cookie expires 30 minutes from the first request. True$\rightarrow$ last request.

Role Manager Config Manager

- cookieProtection
  - Specifies the amount of protection you want to apply to the cookie stored on the end user’s machine for management information.
  - Possible values are *All*, *None*, *Encryption*, and *Validation*.
- defaultProvider
  - Defines the provider used for the role management service. (AspNetSqlRoleProvider)

Roles Class

- Gateway to the Role Management API
- Provides static methods for performing key role management tasks
  - Creating and deleting roles
  - Adding users to roles
  - Removing users from roles and more
- Includes read-only static properties for acquiring data about provider settings.

Key Roles Methods

Name	Description
AddUserToRole	Adds a user to a role
CreateRole	Creates a new role
DeleteRole	Deletes an existing role
GetRulesForUser	Gets a collection of roles to which a user belongs
GetUsersInRole	Gets a collection of users belonging to a specified role
IsUserInRole	Indicates whether a user belongs to a specified role
RemoveUserFromRole	Removes a user from the specified role

Programmatic Role Creation & User Membership

```
if (!Roles.RoleExists ("Developers"))
 {
    Roles.CreateRole ("Developers");
}
```

```
string name = Membership.GetUser ().Username; // Get current user
Roles.AddUserToRole (name, "Developers");     // Add current user to role
```

Using LoginView with Roles

```
<asp:LoginView ID="LoginView1" Runat="server">
  <AnonymousTemplate>
    <!-- Content seen by unauthenticated users -->
  </AnonymousTemplate>
  <LoggedInTemplate>
    <!-- Content seen by authenticated users -->
  </LoggedInTemplate>
  <RoleGroups>
    <asp:RoleGroup Roles="Administrators">
      <ContentTemplate>
        <!-- Content seen by authenticated users who are administrators -->
      </ContentTemplate>
    </asp:RoleGroup>
      ...
  </RoleGroups>
</asp:LoginView>
```

Example

```
<asp:LoginView ID="LoginView1" runat="server">
          <AnonymousTemplate>
             U r annonymous. Please Login.
          </AnonymousTemplate>
          <RoleGroups>
             <asp:RoleGroup Roles=”Admins”>
                <ContentTemplate>
                   You are an Admin!
                </ContentTemplate>
             </asp:RoleGroup>
             <asp:RoleGroup Roles=”Developer”>
                <ContentTemplate>
                   You are a developer!
                </ContentTemplate>
             </asp:RoleGroup>
          </RoleGroups>
       </asp:LoginView>
```