

ASP.NET 4.5

Lesson 9: ASP.NET Authentication
and Authorization

Lesson Objectives

- In this lesson, you will learn:
 - Securing Web Applications
 - Authentication and Authorization
 - Security Schemes
 - Membership Provider in ASP.NET
 - Role Provider in ASP.NET



9.1: Securing Web Applications

Securing Web Applications

- Securing web applications involves displaying content of the web pages based on User Identity
- Some pages should be presented only to Authenticated users or some content should be presented to the users belonging to specific roles
- To secure Web applications ASP.NET provides three security schemes
 - Windows based, Forms based and Passport based

9.1: Securing Web Applications

Securing Web Applications (Contd...)

- Any security scheme revolves around the concept of Users and Roles
- ASP.NET provides set of classes, called as Membership APIs to work with users and roles in the system

9.2: Authentication and Authorization

Authentication and Authorization

- Authentication is a process wherein the user trying to access the system proves that he is the one who he claims to be
- Authorization is a process where the system grants the user access rights and privileges
- Authorization is done only when the authentication is successful



Copyright © Capgemini 2015. All Rights Reserved. 5

The ASP.NET Account

Some of the subtlest issues with ASP.NET deployment involve security. When the web server runs your web application, it performs all its work under a specific Windows user account that has a carefully limited set of privileges. The actual account that's used depends on the web server you're using:

If you're using the integrated test server in Visual Studio, the server runs under your account. That means it has all your permissions, and as a result, you generally won't run into permission problems while you're testing your application. This can be very misleading because you might not realize that there are potential permission problems waiting for you once you deploy the application.

- If you're using IIS 7, it's the *network service account*. *This is a special account that* Windows creates when you first install it. It's not allowed to do much, but it can access network locations.

- If you're using IIS 7.5 or IIS 8, it's an account that's based on the application pool. For example, an application pool named ASP.NET v4.5 will use an account named *IIS*

AppPool\ASP.NET v4.5, which IIS generates automatically.

9.3: Security Schemes

Security Schemes

- Windows based security
 - The credentials of the Windows OS are used to authenticate the user
 - This scheme is often used for company intranet or portals
- Forms based security
 - Presents a login page to the user
 - Once the user enters the user id and password the details are verified against a database
- Passport
 - A security scheme with single sign on feature, with the help of passport network

9.3: Security Schemes

Security and IIS

- When the request goes to IIS, IIS first checks if the requesting IP is allowed to access the resource
- If IP is allowed then IIS attempts to authenticate the request using windows authentication scheme
- If the user credentials are not found in the Windows Access Control List (ACL) the requesting user is denied the access, otherwise the request is forwarded to ASP.NET for further processing

9.3: Security Schemes

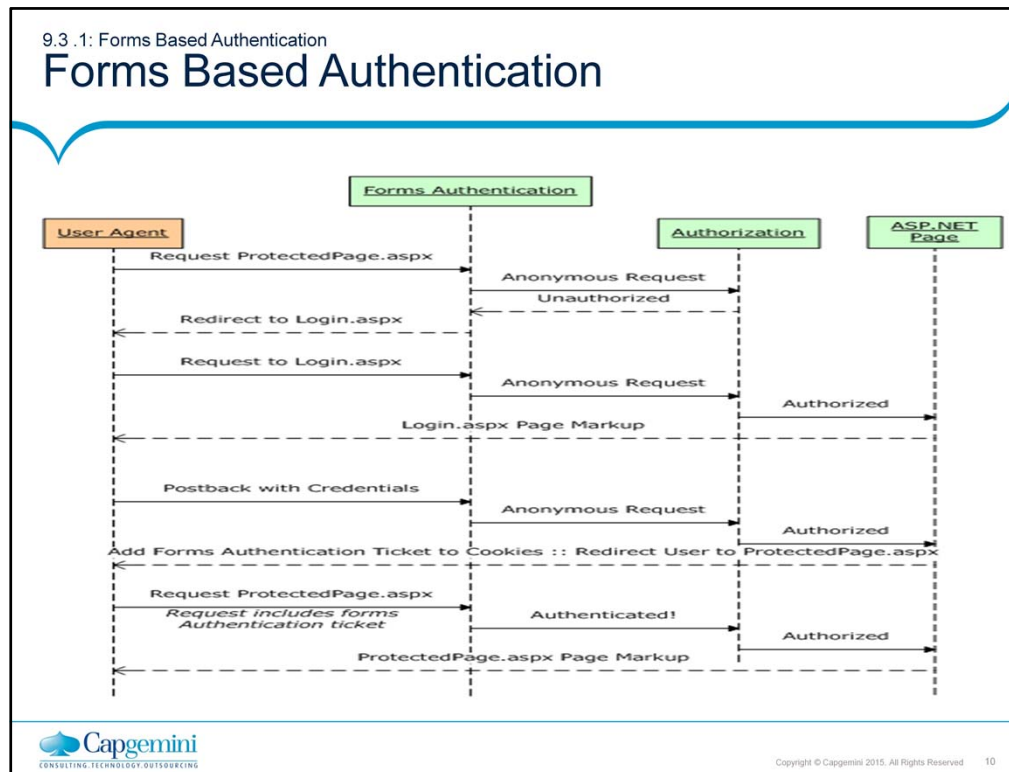
Security and ASP.NET

- Once IIS forwards the request to ASP.NET, it will perform checks on the request depending on the type of security scheme enabled for the application
- If the application is configured to use Windows based security then ASP.NET will first check if impersonation is enabled
- If yes then ASP.NET will try to run the code assuming the identity of the logged in user, otherwise it will run the code on behalf of ASP.NET inbuilt user

9.3: Security Schemes

Security and ASP.NET (Contd...)

- If the application is configured to use Forms based security then ASP.NET checks the incoming request for presence of a cookie called authentication cookie
- This cookie acts like a ticket or license for the user
- If the cookie is not found, then the user is redirected to the login page
- Login details are verified and an authentication cookie is issued
- If cookie is found, the user will be allowed to go ahead



The login page's responsibility is to determine if the user's credentials are valid and, if so, to create a forms authentication ticket and redirect the user back to the page they were attempting to visit. The authentication ticket is included in subsequent requests to the pages on the website, which the FormsAuthenticationModule uses to identify the user.

The **Forms Authentication** class is defined in the **System.Web.Security** namespace and is used to implement the form authentication when the configuration file stores the user credentials.

The commonly used methods are,

Authenticate:

Validated the user credentials passed to it and returns a boolean indicating whether the authentication process is successful or not.

RedirectFromLoginPage:

Redirects an authenticated user back to the specific URL.

SignOut:

Removes the authentication cookies created on the computer of an authenticated user.

```

protected void Button1_Click(object sender, EventArgs e)
{
    if (FormsAuthentication.Authenticate(txtUserName.Text,txtPassword.Text))
    {
        FormsAuthentication.RedirectFromLoginPage(txtUserName.Text, saveMe.Checked);
    }
}
  
```

Configuring an Application for Forms Authentication

Change the mode attribute of the <authentication> element to Forms

If a user tries to access the restricted page without logging on, the user should be redirected to the login page. The settings for redirecting the anonymous users to the logon page are specified by using the <forms> element.

The <forms> element has the following attributes.

1. name:

Specifies the HTTP cookie that will be used for authentication.

.ASPXAUTH is the default value of this attribute.

2. loginurl:

Specifies the URL to which the user requests are redirected for logon when the authentication cookie is not found.

The default value is default.aspx

3. protection:

Specifies the type of protection to protect the cookie.

The valid values for this attribute are All, None, Encryption and Validation.

The default value is All.

4. timeout:

Specifies the life span of the cookie used for authentication.

After the time specified (in minutes) in this attribute, the cookie expires.

Examples:

Assuming the user credentials are stored in DB

```
<authentication mode="Forms">
  <forms loginUrl="LogIn.aspx" cookieless="UseUri"
name=".ASPXAUTH" protection="All" timeout="60"/>
</authentication>
```

In the example given below, the user credentials are stored in the configuration file itself

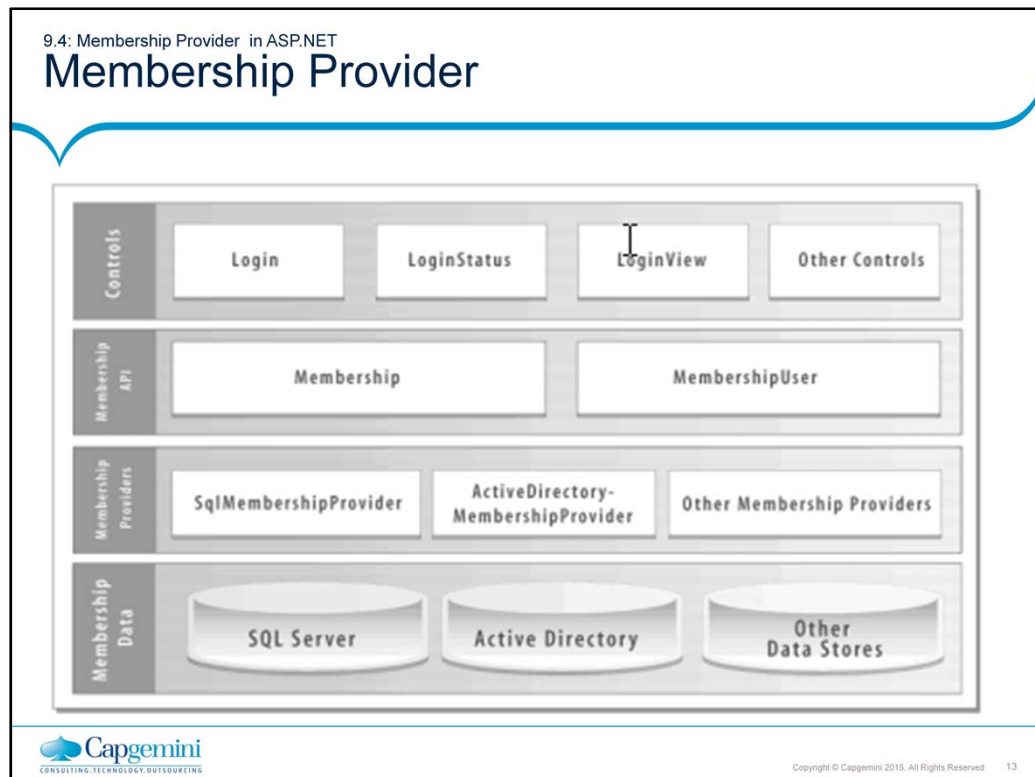
```
<authentication mode="Forms">
  <forms loginUrl="LogIn.aspx" name=".ASPXFORMAUTH"
protection="All">
    <credentials passwordFormat="Clear">
      <user name="John" password="J0hn@123"/>
      <user name="Grace" password="P@ss123"/>
    </credentials>
  </forms>
</authentication>
```

9.3.2: Demo: Implementing Security

Demo: Implementing Security

- Implementing Windows based security
- Implementing Forms based security



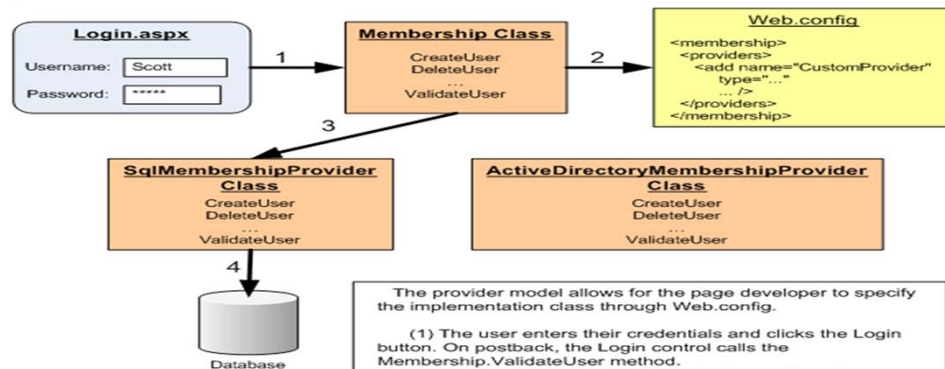


Membership Provider:

- The fundamental job of a membership provider is to interface with data sources containing data regarding a site's registered users, and to provide methods for creating users, deleting users, verifying login credentials, changing passwords, and so on.
- The .NET Framework's **System.Web.Security** namespace includes a class named **MembershipUser** that defines the basic attributes of a membership user and that a membership provider uses to represent individual users.

9.4: Membership Provider in ASP.NET

Membership Provider



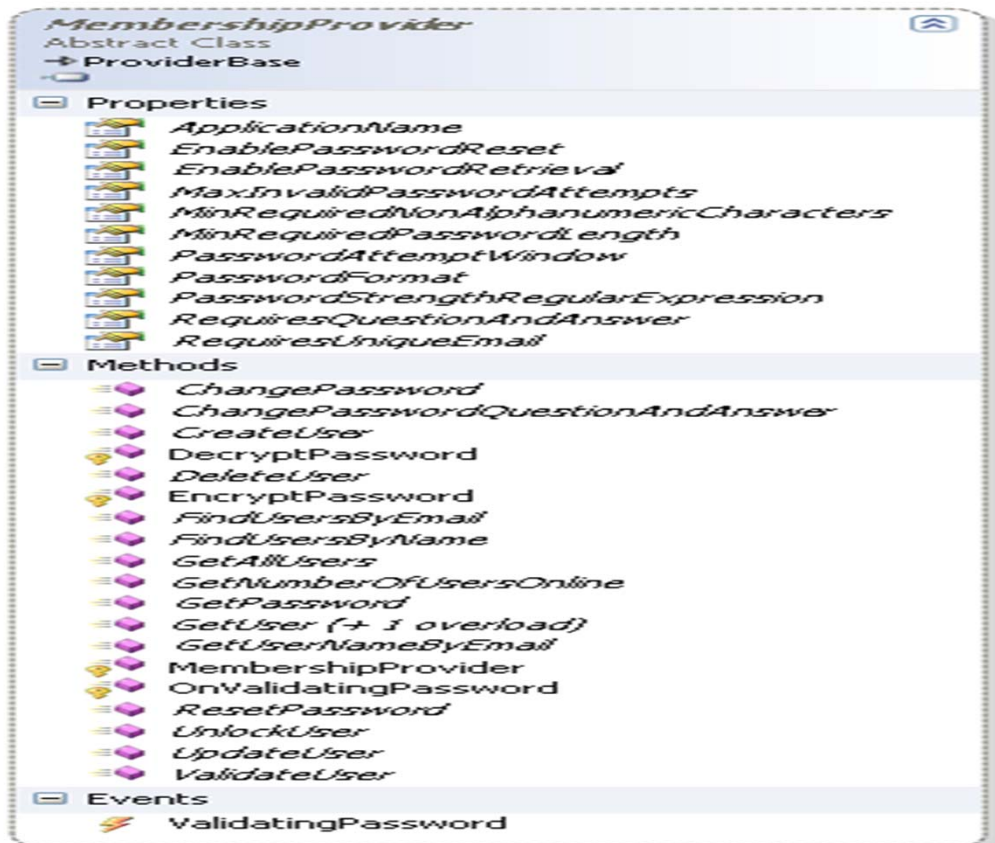
The provider model allows for the page developer to specify the implementation class through Web.config.

(1) The user enters their credentials and clicks the Login button. On postback, the Login control calls the `Membership.ValidateUser` method.

(2) The Membership class consults the configuration settings to determine what Membership provider to use.

(3) The Membership class delegates the call to the specified Membership provider.

(4) In this case of the `SqlMembershipProvider`, a query is executed against the user store SQL Server database to determine if the credentials are valid. The result – a Boolean value – is returned to the caller. Depending on the return value the Login control either informs the user their credentials are invalid or logs them in to the system via forms authentication.



9.4: Membership Provider in ASP.NET

What does ASP.NET Membership Offer?

- The ASP.NET Membership Provider provides the following functions:
 - Creating new users and passwords
 - Storing membership information in Microsoft SQL Server, Active Directory, or an alternative data store
 - Authenticating users who visit our site
 - Managing a password which includes creating, changing, and resetting them



Copyright © Capgemini 2015. All Rights Reserved 15

What does ASP.NET Membership Offer?

ASP.NET membership gives us a built-in way to validate and store user credentials. ASP.NET membership therefore helps us to manage user authentication in our Web sites. We can use ASP.NET membership with ASP.NET Forms authentication or with the ASP.NET login controls to create a complete system for authenticating users.

ASP.NET membership supports facilities for:

1. Creating new users and passwords.
2. Storing membership information (user names, passwords, and supporting data) in Microsoft SQL Server, Active Directory, or an alternative data store.
3. Authenticating users who visit our site. We can authenticate users programmatically, or we can use the ASP.NET login controls to create a complete authentication system that requires little or no code.
4. Managing a password, this includes creating, changing, and resetting them. Depending on membership options we choose, the membership system can also provide an automated password-reset system that takes a user-supplied question and response.
5. Exposing a unique identification for authenticated users that we can use in our own applications and that also integrates with the ASP.NET personalization and role-management (authorization) systems.
6. Specifying a custom membership provider, which allows us to substitute our own code to manage membership and maintain membership data in a custom data store

9.4.1: Usage of Membership Provider

Usage of Membership Provider

- Membership providers provide the interface between ASP.NET's membership service and membership data sources
- One can write a custom membership provider for variety of reasons



Copyright © Capgemini 2015. All Rights Reserved 16

Membership Provider:

Membership providers provide the interface between ASP.NET's membership service and membership data sources.

The two most common reasons for writing a custom membership provider are:

1. We wish to store membership information in a data source that is not supported by the membership providers included with the .NET Framework, such as an Oracle database or Web service
2. We wish to store membership information in a SQL Server database whose schema differs from that of the database used by **System.Web.Security.SqlMembershipProvider** – if, for example, we need to integrate ASP. Net's membership service with an existing membership database.

The fundamental job of a membership provider is to interface with data sources containing data regarding a site's registered users, and to provide methods for creating users, deleting users, verifying login credentials, changing passwords, and so on. The .NET Framework's System.Web.Security namespace includes a class named MembershipUser that defines the basic attributes of a membership user and that a membership provider uses to represent individual users.

9.4.1: Usage of Membership Provider

Membership, Roles & User Profiles

- Membership Providers can be integrated with ASP.NET role management to provide authorization services for a site
- Membership can also be integrated with the user profile to provide application-specific customization that can be tailored to individual user requirements

9.4.1: Usage of Membership Provider

User

- Any security scheme revolves around user
- ASP.NET provides an inbuilt object called User that represents the current user accessing the web application
- This object is available irrespective of the security scheme in place
- This User object has some useful properties and methods that allow getting information about the current user

9.4.1: Usage of Membership Provider

User – Properties and methods

- `User.Identity.Name` – Returns the login name of the current user
- `User.Identity.IsAuthenticated` – Returns whether current user is authenticated or not
- `User.IsInRole()` – Returns true if the user belongs to a specified role or else returns false

9.4.2: Configuring Membership Provider

Configuring Membership Provider

- Steps to configure Membership for the site:

1. Specify Membership Options as part of our Web Site Configuration
2. Configure our application to use Forms authentication
3. Define user accounts for membership. Two options are:
 - a. Use the Web Site Administration Tool.
 - b. Create a "new user" ASP.NET Web page where we collect a user name and password (and optionally an e-mail address), and then use a membership function named CreateUser to create a new user in the membership system.

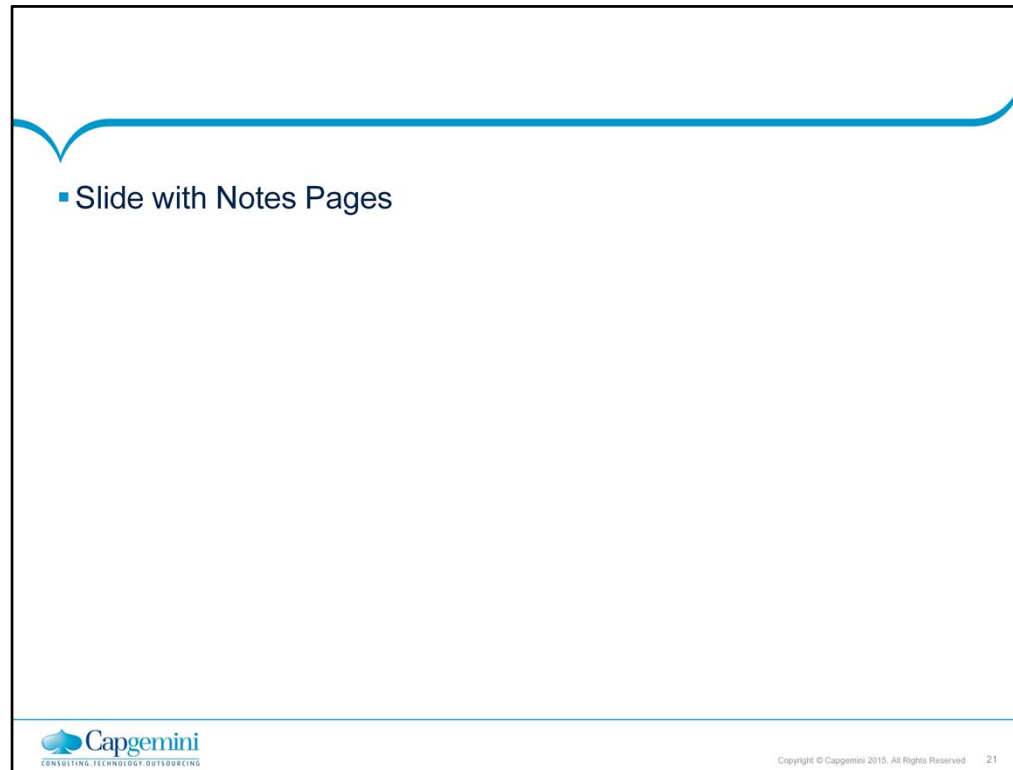


Copyright © Capgemini 2015. All Rights Reserved 20

How Membership Works?

To use membership, we must first configure it for our site. In outline, we follow the given steps:

1. Specify membership options as part of our Web site configuration. By default, membership is enabled. We can also specify what membership provider we want to use. The default provider uses a Microsoft SQL Server database. We can also choose to use Active Directory to store membership information, or we can specify a custom provider.
2. Configure our application to use Forms authentication. We typically specify that some pages or folders in our application are protected and are accessible only to authenticated users.
3. Define user accounts for membership. We can do this in a variety of ways. We can use the Web Site Administration Tool, which provides a wizard-like interface for creating new users. Alternatively, we can create a "new user" ASP.NET Web page where we collect a user name and password (and optionally an e-mail address), and then use a membership function named CreateUser to create a new user in the membership system.



How Membership Works?

We can now use membership to authenticate users in our application. Most often, we will provide a login form, which might be a separate page or a special area on our home page. We can create the login form using ASP.NET TextBox controls, or we can use ASP.NET login controls. Since we have configured the application to use Forms authentication, ASP.NET will automatically display the login page if an unauthenticated user requests a protected page.

If we use login controls, they will automatically use the membership system to validate a user. If we have created a login form using TextBoxes, we can prompt the user for a user name and password and then call the `ValidateUser` method to perform the validation. After the user is validated, information about the user can be persisted (for example, with an encrypted cookie if the user's browser accepts cookies) using Forms Authentication.

The login controls perform this task automatically. If we have created a login form by hand, we can call methods of the `FormsAuthentication` class to create the cookie and write it to the user's computer. If a user has forgotten his or her password, the login page can call membership functions that help the user remember the password or create a new one.

Each time the user requests another protected page, ASP.NET Forms authentication checks whether the user is authenticated and then either allows the user to view the page or redirects the user to the login page.

Steps for Membership to enable SQLServer to serve as RoleProvider.

1. Goto Visual Studio Command Prompt
NDAMSSQL\SQLILEARN is the SQL Server Name
aspnet_regsql.exe -E -S NDAMSSQL\SQLILEARN -A mr -U -P
It will create ASPNetDb in your sqlserver.

To enable membership and roles for Server if you need for a specific database specify the database name with -ed -d Training

2. Do the settings in Web.config file

3. Write the code for Membership.CreateUser

Either specify the connectionstring with ASP.NetDB in web.config or specify a LocalDB so that its local to your application

To Validate the supplied UserName and Password, ValidateUser method of Membership is used.

The method returns a boolean value based on which the page will be either redirected to the requested page using the RedirectFromLoginPage method of FormsAuthentication class or simply prints an error message if the User is not Authenticated

```
public void Login_OnClick(object sender, EventArgs args)
{
    if (Membership.ValidateUser(txtUsername.Text, txtPassword.Text))
        FormsAuthentication.RedirectFromLoginPage(txtUsername.Text,
NotPublicCheckBox.Checked);
    else
        Msg.Text = "Login failed. Please check your user name and password
and try again.";
}
```

9.4.2: Configuring Membership Provider

Configuring Membership Provider

- Let us see an example on Membership Configuration and Management:

```
<configuration>
<system.web>
<membership ...>
<providers>
<!-- Membership providers registered here -->
</providers>
</membership>
...
</system.web>
</configuration>
```



Copyright © Capgemini 2015. All Rights Reserved 23

Note: Providers are registered in **<providers>** configuration sections in the configuration sections of the features and services that they serve. For example, membership providers are registered as shown above.

9.4.2: Configuring Membership Provider

Configuring Membership Provider

- Configure the Membership System using the Web.config file
- The easiest way to configure and manage membership is with the Web Site Administration Tool.
- Web Site Administration Tool provides a wizard-based interface.



Copyright © Capgemini 2015. All Rights Reserved 24

Membership Configuration and Management:

We configure the membership system in our application's **Web.config** file. The easiest way to configure and manage membership is with the Web Site Administration Tool, which provides a wizard-based interface. As part of membership configuration, we specify:

- **What membership provider do you use?**
 - This typically specifies what database to store membership information in.
- **Password options**
 - These options include encryption and whether to support password recovery based on a user-specific question.
- **Users and passwords**
 - If we are using the Web Site Administration Tool, we can create and manage users directly. Otherwise, we must call membership functions to create and manage users programmatically.

9.4.3: Usage of WebSite Administration Tool

Usage of Web Site Administration Tool

- The Web Site Administration Tool lets you view and manage the Web site configuration through a simple Web interface
- The Web Site Administration Tool lets you change your site configuration without having to manually edit the Web.config file
- The three tabs:
 - **Security** for managing user accounts and roles.
 - **Application** for managing general application settings.
 - **Provider** for testing and assigning providers for membership and role management



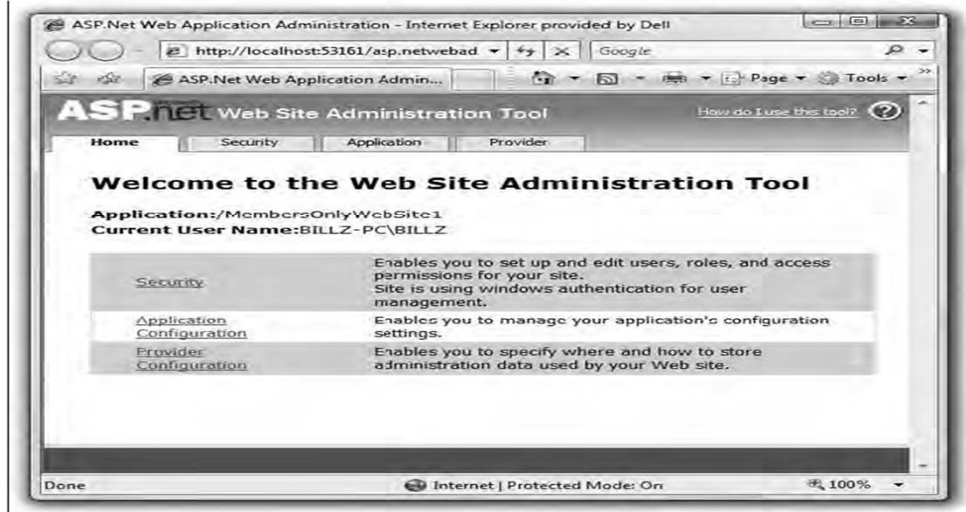
Copyright © Capgemini 2015. All Rights Reserved 25

Web Site Administration Tool:

- One of the strengths of ASP.NET is its configuration system. By centralizing all the settings that control the behavior of your ASP.NET Web site in a single XML file (web.config), ASP.NET makes it easy to be sure that you have not overlooked anything when you are customizing a site.
- Apparently Microsoft's developers too were annoyed by this, because they came up with not one but two different ways to make the ASP.NET configuration process easier in version 2.0. In this article, I will show you the new tools. The Web Site Administration Tool is designed for use by developers who are building ASP.NET 2.0 sites, and the ASP.NET Configuration Tool is smoothly integrated with Internet Information Services Manager for use by administrators.
- The Web Site Administration Tool is available from within Visual Studio 2008 while you are designing an ASP.NET 2.0 Web site. To launch the tool, select ASP.NET Configuration from the Website menu. This will open the tool in a separate browser window.
- The tool breaks its functionality down across three tabs:
 - Security:** To manage access rules to help secure specific resources within the Web site and to manage user accounts and roles
 - Application:** To manage a variety of settings related to the Web site such as Application Settings, SMTP Settings, and so on
 - Provider:** To test or assign providers for membership and role management for the Web site

9.4.3: Usage of WebSite Administration Tool

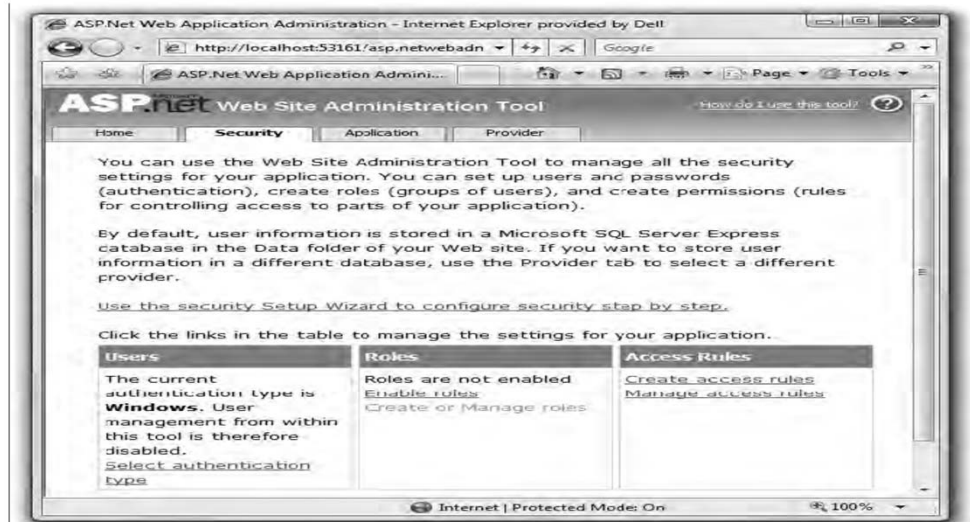
Usage of Web Site Administration Tool (Contd...)



Note: The above slide shows a snapshot of Web Site Administration tool.

9.4.3: Usage of WebSite Administration Tool

Security Tab of Web Site Administration Tool



Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 27

Security Tab of Web Site Administration Tool:

- Use the **Security** tab of the Web Site Administration Tool to manage rules for securing specific resources in the Web application.
- ASP.NET uses a security system that lets you restrict access to specific user accounts or the roles to which the user accounts belong.
- With the **Security** tab, you manage user accounts, roles, and access rules for the Web site. Before using the Security tab for the first time, use the Security Setup Wizard to configure basic security settings for the Web site.
- ASP.NET security is based on the concepts of user accounts, roles, and access rules and lets you restrict access to your Web application resources to only the user accounts that you specify.
- Security settings are established using a combination of configuration settings and data stored in a database (or other data store). User accounts and roles that you create are stored in the database and access rules are stored in the **Web.config** file.
- The above slide shows a snapshot of Web Site Administration tool showing Security tab.

9.4.3: Usage of WebSite Administration Tool

Demo: Web Site Administration Tool

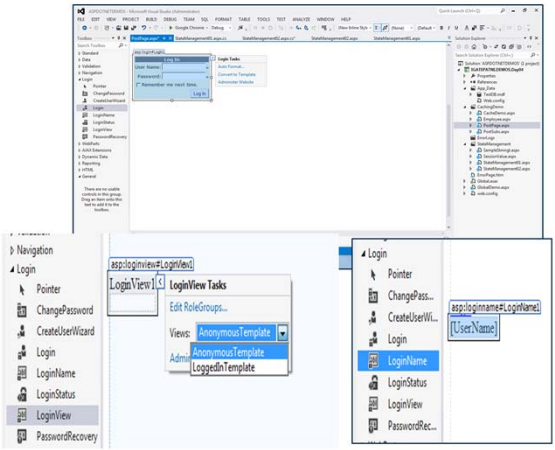
- Creating Membership using Web Site Administration Tool



9.5: ASP.NET Login Controls

ASP.NET Login Controls

- ASP.NET Login Controls available on the screen are:
 - Login
 - LoginView
 - LoginName
 - PasswordRecovery
 - LoginStatus
 - ChangePassword
 - CreateUserWizard



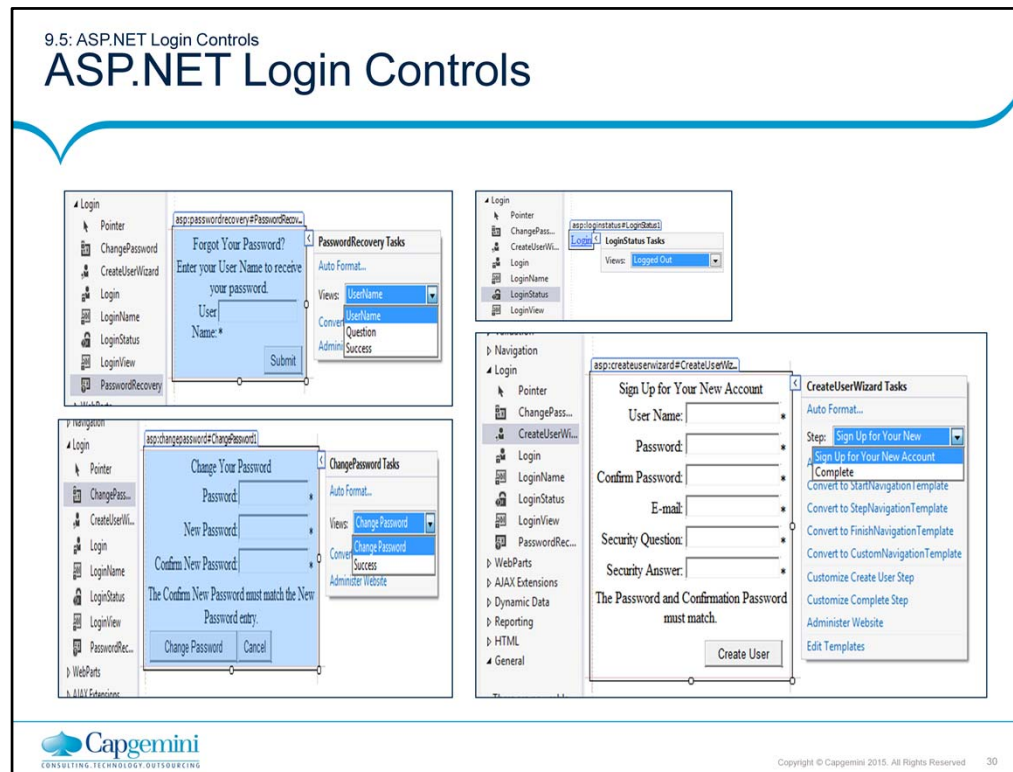
Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 29

ASP.NET Login Controls:

Modern ASP.NET improves things by adding a number of login controls that perform the most common login scenarios you might need for your site. These controls include the Login, LoginView, PasswordRecovery, LoginStatus, LoginName, ChangePassword, and CreateUserWizard controls.

- 1. Login:** The Login control is the simplest login control and supports the most common login scenario—signing in using a user name and password. The control includes user name and password text boxes and a check box for users who want to compromise password security by saving their passwords on the machine. The control exposes properties through which you can change the text and appearance of the control. You may also add links to manage registration or password recovery. The Login control interacts with the ASP.NET membership component for authentication by default. If you want to manage authentication yourself, you may do so by handling the control's Authenticate event.
- 2. LoginView:** The LoginView control is very similar to the optional login page mentioned earlier. It's useful for managing the content you display for authenticated versus nonauthenticated users. The LoginView displays the login status via the display templates AnonymousTemplate and LoggedInTemplate. The control renders a different template depending on the status of the user. The LoginView also lets you manage text and links within each template.
- 3. LoginName:** The LoginName control displays the user's login name.



ASP.NET Login Controls (contd.):

4. **PasswordRecovery:** The PasswordRecovery control supports Web sites that send user passwords to clients when they forget their passwords. The control collects the user's account name and then follows up with a security question (provided that functionality is set up correctly). The control either e-mails the current password to the user or creates a new one.
5. **LoginStatus:** The LoginStatus control displays whether or not the current user is logged on. Nonlogged-in users are prompted to log in, whereas logged-in users are prompted to log out.
6. **ChangePassword:** The ChangePassword control gives users a chance to change their passwords. An authenticated user may change his or her password by supplying the original password and a new password (along with a confirmation of the new password).
7. **CreateUserWizard:** The CreateUserWizard control collects information from users so it can set up an ASP.NET membership account for each user. Out of the box, the control gathers a user name, a password, an e-mail address, a security question, and a security answer. The CreateUserWizard will collect different information from users, depending on the membership provider used by your application.

9.5.1: Demo: Login Controls

Demo: Login Controls

- Using various Login Controls in ASP.NET:


- Login
- LoginView
- PasswordRecovery
- LoginStatus
- LoginName
- CreateUserWizard
- ChangePassword



9.6: Role Providers in ASP.NET

Role Providers

- Role providers provide the interface between ASP.NET's role management service (the "role manager") and role data sources
- One may create a custom Role Provider for variety of reasons

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 32

Role Providers:

Role providers provide the interface between ASP.NET's role management service (the "role manager") and role data sources.

The two most common reasons for writing a custom role provider are:

- You wish to store role information in a data source that is not supported by the role providers included with the .NET Framework, such as an Oracle database or a Web service.
- You wish to store role information in a SQL Server database whose schema differs from that of the database used by *System.Web.Security.SqlRoleProvider*-if, for example, you need to integrate ASP.NET's role manager with an existing role database.

The fundamental job of a role provider is to interface with data sources containing role data mapping users to roles, and to provide methods for creating roles, deleting roles, adding users to roles, and so on.

Given a user name, the role manager relies on the role provider to determine whether what role or roles the user belongs to. The role manager also implements administrative methods such as *Roles.CreateRole* and *Roles.AddUserToRole* by calling the underlying methods in the provider.

9.6: Role Providers in ASP.NET

Role Providers

- **SqlRoleProvider:**
 - It stores role data in Microsoft SQL Server and Microsoft SQL Server Express databases.
- **AuthorizationStoreRoleProvider**
 - It retrieves role information from Microsoft Authorization Manager ("AzMan").
- **WindowTokenRoleProvider**
 - It retrieves role information from each user's Microsoft Windows authentication token, and returns his or her group membership(s).

```

classDiagram
    class RoleProvider {
        <<MustInherit Class>>
        +ProviderBase
    }
    class AuthorizationStoreRoleProvider {
        +RoleProvider
    }
    class WindowsTokenRoleProvider {
        +RoleProvider
    }
    class SqlRoleProvider {
        +RoleProvider
    }
    RoleProvider <|-- AuthorizationStoreRoleProvider
    RoleProvider <|-- WindowsTokenRoleProvider
    RoleProvider <|-- SqlRoleProvider
  
```

Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 33

Role Providers in ASP.NET:

ASP.NET 4.5 ships with three role providers:

- **SqlRoleProvider** which stores role data in Microsoft SQL Server and Microsoft SQL Server Express databases
- **AuthorizationStoreRoleProvider** which retrieves role information from Microsoft Authorization Manager ("AzMan")
- **WindowTokenRoleProvider** which retrieves role information from each user's Microsoft Windows authentication token, and returns his or her group membership(s).

9.6: Role Providers in ASP.NET

Role Providers

- One can create roles for our application and assign users to these roles
- There are several methods we can use to create and assign roles
- One of the method we can use is ASP.NET Configuration Tool



Copyright © Capgemini 2015. All Rights Reserved 34

Create and Assign Roles:

We can create roles for our application and assign users to those roles. There are several methods we can use to create and assign roles. Using them depends on how our application authenticates its users and which role provider it uses.

The various methods for creating and assigning users to roles include:


- At development time, we can use the ASP.NET configuration tool.
- If we are using the AuthorizationStoreRoleProvider, we can use the AzMan administrator Microsoft Management Console (MMC) snap-in.
- We can create roles programmatically by using either the role management APIs or, if we are using the SqlRoleProvider, by executing SQL scripts to add them to the database directly.
- If we are using the WindowsTokenRoleProvider, we use the Windows Computer Management tool or Active Directory Users and Computers to create Windows groups which are used as roles.

9.6: Role Providers in ASP.NET

Role Providers

- Let us see how role providers are registered.

```
<configuration>
<system.web>
<roleManager ...>
<providers>
<!-- Role providers registered here -->
</providers>
</roleManager>
...
</system.web>
</configuration>
```

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 35

The above example shows how Role Providers are registered in Web.Config

Summary

- In this lesson, you have learnt about:

- Security in ASP.NET
- Membership Provider in ASP.NET
- Web Site Administration Tool
- Login Controls in ASP.NET
- Role Provider in ASP.NET



Review Question

- Question 1: Which of the following are the providers available with ASP.NET:
 - Membership Provider
 - Code Provider
 - Role Provider
 - Profile Provider
 - Web Service Provider
 - Site Map Provider



Review Question

- Question 2: Which of the following Login controls is useful for managing the content displayed for authenticated versus nonauthenticated users:
 - Login View
 - Login Status
 - Login Name
 - Password Recovery



Review Question

- Question 3: Which of the following tools allows you to view and manage the Web site configuration through a simple Web interface
 - GacUtil
 - Web Site Configuration Tool
 - ILDASM
 - Provider Configuration Tool

