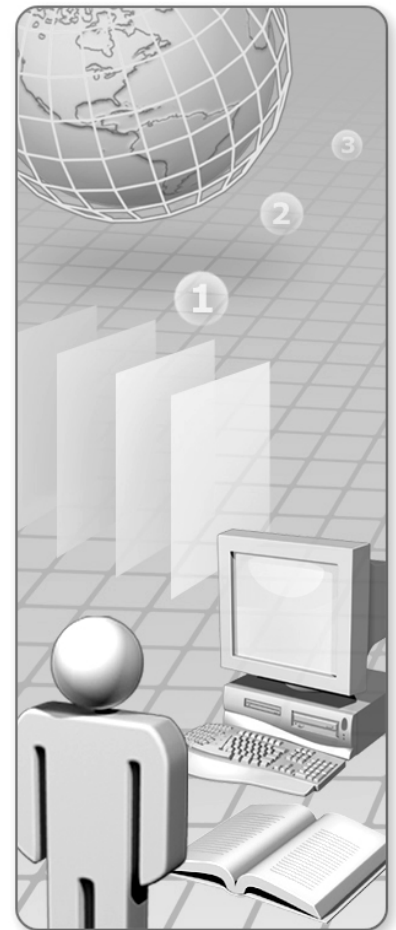


# Unit 7: Controlling Access to a Web Application

---

## Contents

Overview	1
Authentication for Web Applications	2
Authorization for Web Applications	4
Site Membership Systems Using the Membership Class	6
Web Site Security Administration Using the Roles Class	8
Lab Scenario	10
Lab Tasks and Objectives	11
Lab: Controlling Access to a Web Application	13
Lab Discussion	25



# Overview



- Authentication for Web Applications
- Authorization for Web Applications
- Site Membership Systems Using the Membership Class
- Web Site Security Administration Using the Roles Class
- Lab Scenario
- Lab Tasks and Objectives
- Lab: Controlling Access to a Web Application

## Introduction

Microsoft® ASP.NET 2.0 includes several features that make it easy to control access to your Web site. For example, you can use ASP.NET authentication to identify a user with a user name and password. It can store these credentials, along with other information about that user, in a variety of places, such as a database. You can use authorization to specify the pages in your application that a particular user has access to. You can configure ASP.NET authentication and authorization by using the Web.config file for the Web site.

You can use the ASP.NET login controls to build login pages for user authentication and authorization. These controls enable you to build an access control system and require very little custom coding. The **Membership** class provides methods for validating users' credentials and managing users' settings. You can use the **Roles** class to group users, depending on their role in the application, and assign access to parts of the system on that basis.

## Objectives

After completing this unit, you will be able to:

- Describe the authentication methods for Web applications.
- Describe the authorization methods for Web applications.
- Describe the main components of a membership system.
- Describe how to build a security administration interface.
- Configure authentication and authorization for a Web application.
- Implement a membership registration page.
- Implement a login page.
- Create a membership management administrative user interface.

# Authentication for Web Applications



- Windows Authentication
- Forms Authentication
- Passport Authentication

## Introduction

*Authentication* is the process by which users prove their identity. This usually involves users entering a user name and password, or credentials, onto a login page. ASP.NET 2.0 provides three authentication mechanisms:

- Windows authentication
- Forms authentication
- Passport authentication

## Windows Authentication

If you configure your application to use Microsoft Windows® authentication, Internet Information Services (IIS) identifies the user by comparing the credentials entered by the user against the user's Windows account. Windows authentication provides three possible login methods:

- Basic authentication. In this method the credentials are passed across the network in plain text.
- Digest authentication. In this method the password is irreversibly hashed before it is transmitted.
- Windows Integrated authentication. In this method, the user is authenticated in the same way as when logging on to a Microsoft Active Directory® directory service domain.

The method IIS chooses for a particular user will depend on the browser being used and the configuration of IIS. For example, only Microsoft Internet Explorer supports the Windows Integrated method; other browsers will use Basic authentication.



**Important** When using Basic authentication, the credentials are passed across the network in plain text. It is essential to use an encryption protocol, usually Secure Sockets Layer (SSL), to protect the credentials.

---

## Forms Authentication

Windows authentication is useful only if all users have Microsoft Windows accounts. If you are building an Internet application, using Windows authentication will not be feasible or desirable. Consequently, you might prefer to store user accounts somewhere other than the Windows security system. For example, you can elect to store user credentials in a database hosted on a computer running Microsoft SQL Server™, and you can include other properties not present in a Windows account. The Forms authentication mode makes it easy for you to create such a customized security regime and to do so securely.

When you configure Forms authentication, you can specify a login page. When users request any page in your application, if they are not authenticated, they are redirected to the login page where they can enter their credentials. You must write code to check these credentials. After they have been authenticated, users are redirected to the page they originally requested.

If you have a small number of users, you can store credentials in the Web.config file. However, a more scalable storage location, such as a database, is recommended for most situations.

By default, Forms authentication creates a cookie when the user has logged on, and it stores the cookie on the user's computer. This cookie is submitted with each request. However, Forms authentication can also be configured to use the query string for browsers that have had cookie support disabled.

You can configure Forms authentication by using the Web.config file. The following example specifies that users should be directed to a logon page called logon.aspx, and it has the details of two users. The passwords have been irreversibly hashed:

```
<authentication mode="Forms">
  <forms name="SavingsPlan" loginUrl="/logon.aspx">
    <credentials passwordFormat="SHA1">
      <user name="Kim" password="07B7F3EE06F278DB966BE960E7CBBBD103DF30CA6"/>
      <user name="John"
password="BA56E5E0366D003E98EA1C7F04ABF8FCB3753889"/>
    </credentials>
  </forms>
</authentication>
```

## Passport Authentication

Passport authentication is a centralized authentication service provided by Microsoft. Users can use their Microsoft .NET Passport to access services such as Microsoft Hotmail® and Microsoft Windows Messenger/MSN® Messenger. If you register your site with the Passport service, users can use the same Passport to access your site: They do not need to remember a separate set of credentials. To use Passport authentication, you must complete the following steps:

1. Obtain the .NET Passport software development kit (SDK). This is included with Microsoft Windows Server™ 2003, or it can be downloaded from the Microsoft Passport Network.
2. Configure Passport authentication in the Web.config file as follows:

```
<authentication mode= "Passport"/>
```

3. Implement authentication and authorization by using the functionality in the .NET Passport SDK.

# Authorization for Web Applications



- File Authorization
- URL Authorization

## Introduction

After a user has been authenticated and has gained access to your Web site, the application must determine the pages and resources that the user has access to. This process is known as *authorization*.

## File Authorization

Windows provides its own authorization mechanism. You can set permissions on any file or folder stored on a disk formatted with the NTFS file system. These permissions are stored in the access control list (ACL), which is stored with the file. The ASP.NET File authorization module enables you to use these permissions to control access to resources, pages, and folders in your Web application. You must use it in conjunction with Windows authentication.

To use File authorization, configure your application to use Windows authentication, and then assign permissions to the files and folders in your Web site.

## URL Authorization

You can use the URL authorization module to control access to each virtual directory within your Web site hierarchy. You can use URL authorization with any of the authentication modules.

To establish permissions for a particular directory, create a Web.config file within that directory. Add to the file an <authorization> section that contains <allow> and <deny> tags for each user or role. There are two special values that you can also use as wildcard identities:

1. “\*” This wildcard entry applies to everyone who visits the directory.
2. “?” This entry applies to anonymous users.

In the following example, Kim is permitted to access the current directory, but John and anonymous users are denied access:

```
<authorization>
  <allow users="Kim"/>
  <deny users="John"/>
  <deny users="?" />
</authorization>
```

# Site Membership Systems Using the Membership Class



- The Components of a Typical Membership System
- The Membership Class
- Login Controls

## Introduction

Many Web sites implement a membership system. ASP.NET 2.0 includes a set of classes that enable you to build such a system, using a database or other location for storing member details.

## The Components of a Typical Membership System

To manage user accounts on your Web site, you must do more than authenticate and authorize users. The following features are typically required:

- User account creation and deletion. New users to your Web site will need to add a new account for themselves and be able to modify their personal data. You might also want to remove unused accounts.
- User account storage. You can store user accounts in SQL Server, Microsoft Office Access, or an alternative data store.
- Authentication. If you are using Forms authentication mode, you can use ASP.NET login controls to create a login page with little or no custom code.
- Password management. You should provide a means of creating and resetting passwords, setting password expiration, and recovering or resetting lost passwords.

The ASP.NET Membership system includes classes to help you implement all of these features. You can use these classes to create a versatile membership system by using a very small amount of code. ASP.NET Membership can be used on its own or in conjunction with Forms authentication and ASP.NET Roles.

## The Membership Class

You can use the **Membership** class to configure a membership system. The **Membership** class provides methods for creating, deleting, and updating user accounts, authenticating users, and managing passwords.

## Login Controls

The ASP.NET login controls are a set of Web server controls that provide the common user interface elements of a membership system. The following controls are available in the Login section of the Microsoft Visual Studio® Toolbox:

- **CreateUserWizard** control. This control collects information from the user and creates a new account in the membership system.
- **Login** control. This control consists of user name and password text boxes and a check box that users can use to indicate that they would like to be remembered the next time they visit. User credentials are authenticated by the membership system.
- **LoginStatus** control. This control displays either a Login link or a Logout link, depending on whether the user is currently logged in.
- **LoginView** control. Use this control to display different information to logged-in users and anonymous users. For example, you might want to display links to enhanced content for members, but not for anonymous users.
- **PasswordRecovery** control. This control enables users to recover a forgotten password if they can answer a security question. If the password is stored with reversible encryption, it is sent to the user via e-mail; otherwise a new password is issued.
- **ChangePassword** control. This control enables the user to change his or her password.



# Web Site Security Administration Using the Roles Class



- Roles and Authorization
- Role Management Configuration
- Creating and Populating Roles
- Checking Role Membership

## Introduction

You can use roles to reduce the administrative overhead of managing permissions for large numbers of users. You can group users into roles and assign permissions to each role. All users belonging to a role automatically gain the permissions associated with that role when they log in. For example, you could have roles for Administrators, Moderators, Power Users, and so on. If you had to assign permission to every single user individually, it would take a long time and require very long `<authorization>` tags in the Web.config files. By grouping users into roles, you can assign permissions once for many users; the system becomes far more manageable.

## Roles and Authorization

If you are using URL authorization mode, you configure access to a directory by using the Web.config file in each directory. You can add roles to the `<authorization>` section. The following example shows how to grant access to members of the Admin and PowerUsers roles but deny access to members of the Customers role and anonymous users:

```
<authorization>
  <allow roles="Admin"/>
  <allow roles="PowerUsers" />
  <deny roles="Customers"/>
  <deny users="?" />
</authorization>
```

## Role Management Configuration

You must configure role management in the Web.config file in the root folder of the Web application. The following example shows how to enable role management and specifies that a cookie will be downloaded to the client to store role membership details:

```
<roleManager
  enabled="true"
  cacheRolesInCookie="true" >
</roleManager>
```

## Creating and Populating Roles

The easiest way to create roles and specify their membership is to use the ASP.NET Web Site Administration Tool. However, you can also create and populate roles programmatically by using the **Roles** class. The following example creates a role called Subscribers and adds two users to it:

### [Visual Basic]

```
Roles.CreateRole("Subscribers")
Roles.AddUsersToRole("Anatoly Sabantsev", "Subscribers")
Roles.AddUsersToRole("Bobby Moore", "Subscribers")
```

### [C#]

```
Roles.CreateRole("Subscribers");
Roles.AddUsersToRole("Anatoly Sabantsev", "Subscribers");
Roles.AddUsersToRole("Bobby Moore", "Subscribers");
```

## Checking Role Membership

You can use the **User** object to check whether the current user is a member of a particular role. The code in the following example hides the btnDownloadFile button if the user is not a member of the Subscribers role:

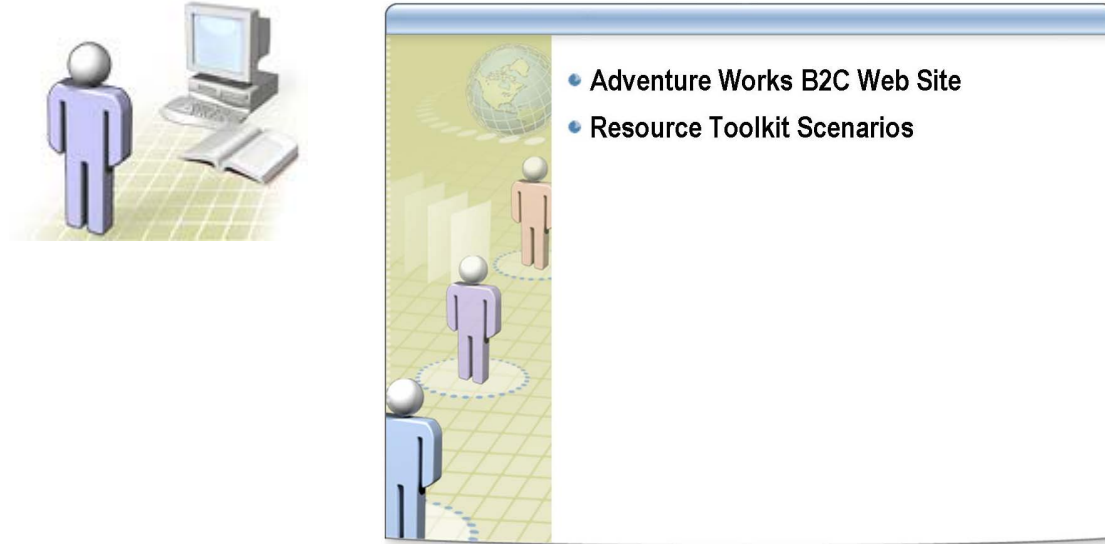
### [Visual Basic]

```
If Not User.IsInRole("Subscribers") Then
  btnDownloadFile.Visible = False
End If
```

### [C#]

```
if (! User.IsInRole("Subscribers"))
  btnDownloadFile.Visible = false;
```

## Lab Scenario



### Adventure Works B2C Web Site

You are a developer in the Adventure Works organization, a fictitious bicycle manufacturer. You have been asked to assist in the development of the Business-to-Consumer (B2C) Web application and a related Business-to-Employee (B2E) extranet portal.

Decisions on the design of the application have already been made. You have been asked to carry out a number of specific tasks in order to implement various elements of this design. As part of the first phase of the B2C development, you have been asked to complete the prototypes for the following pages:

- *MembersLogin.aspx*. This page collects and checks credentials to identify the user.
- *Register.aspx*. This page enables users to become members of the site.
- *Employees.aspx*. This page shows sales figures for the Adventure Works staff, and it should be viewable only by employees.
- *MemberUpdate.aspx*. This page enables users to change the e-mail address and password stored for their account.
- *Admin.aspx*. This page enables site administrators to change the role membership on the site.

You will also ensure that several pages are secured properly.

### Resource Toolkit Scenarios

Before you start work on the lab, you should review the Scenario tab in the Resource Toolkit. The instructor will lead a group discussion of the scenarios for this lab.

## Lab Tasks and Objectives



Task	Objectives
Configure authentication and authorization for a Web application	<ul style="list-style-type: none"> <li>Implement Forms authentication</li> <li>Configure authorization for anonymous users</li> <li>Implement Windows authentication</li> </ul>
Implement a membership registration page	<ul style="list-style-type: none"> <li>Configure the Membership provider</li> <li>Create a membership registration page</li> <li>Create a membership update page</li> </ul>
Implement a login page	<ul style="list-style-type: none"> <li>Create a login page</li> <li>Add login Web server controls</li> <li>Create templates for login controls</li> </ul>
Create a membership management administrative user interface	<ul style="list-style-type: none"> <li>Create a secure page for administration</li> <li>Manipulate role-membership for members</li> <li>Add and delete roles</li> </ul>

### Lab Tasks

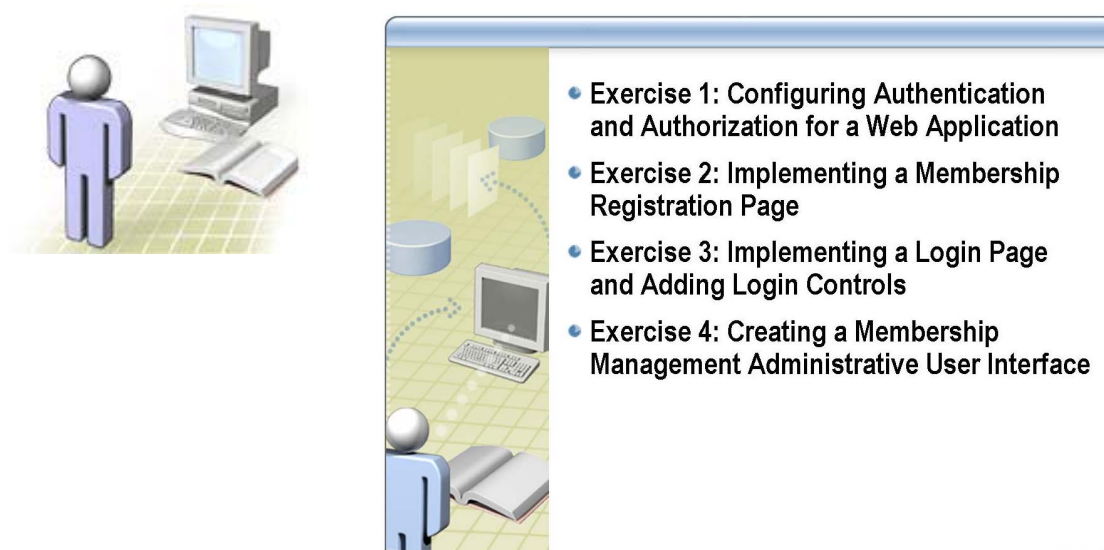
There are four tasks in this lab. Each one is designed to help you achieve one or more learning objectives. Resources are provided in the Resource Toolkit to help you complete the tasks. You should try to complete all of the tasks.

The tasks in this lab are:

- **Configure authentication and authorization for a Web application.** In this task, you will implement Forms authentication for restricted portions of the Web application. You will then configure authorization to control access for anonymous users, and you will implement an employee-only section of the Web site that will use Windows authentication. The resources for this task are titled:
  - *How to: Configure Authentication for a Web Application*
  - *How to: Configure Authorization for a Web Application*
- **Implement a membership registration page.** In this task, you will configure the Membership provider and create a page that enables users to register with the site. You will also create a page that existing registered users can use to update their personal information. The resources for this task are titled
  - *Introduction to Membership*
  - *How to: Configure an ASP.NET Application to Use Membership*
  - *How to: Manage Users by Using Membership*

- *Implement a login page.* In this task, you will create a login page that includes login Web server controls. You will also configure templates for the login Web server controls. The resources for this task are titled:
  - *Login and Membership Page Design*
  - *How to: Implement a Login Page and Add Login Controls*
- *Create a membership management administrative user interface.* In this task, you will create a secure page that administrators can use to manage role-membership for the Web application. The page will allow administrators to add users to existing roles and to add and delete roles. The resource for this task is titled *How to: Create a Membership Management Administrative User Interface*.

## Lab: Controlling Access to a Web Application



After completing this lab, you will be able to:

- Configure authentication and authorization for a Web application.
- Implement a membership registration page.
- Implement a login page.
- Create a membership management administrative user interface.

Estimated time to complete this lab: **90 minutes**

### Lab Solutions

There are Microsoft Visual Basic® and Microsoft Visual C#® solution files associated with the labs in this workshop. You can find the lab solution files in the folder E:\Labfiles\Solution on the virtual machines.

### Lab Setup

For this lab, you will use the available Microsoft Virtual PC environment. Before you begin the lab, you must:

1. Start the **2543B-LON-DEV-01-07** virtual machine.
2. Log on to the **2543B-LON-DEV-01-07** virtual machine with the user name **Student** and the password **Pa\$\$w0rd**.


No additional setup is required for this lab.

## Exercise 1



### Configuring Authentication and Authorization for a Web Application

In this exercise, you will configure the Adventure Works Web site to enable users to log in to the restricted parts of the application. Authentication will be required to access the Bike Reviews Trail Reports and the Employees pages.

#### Scenario



Tasks	Supporting information
<ol style="list-style-type: none"> <li>1. Open the Adventure Works Web site for editing in Visual Studio.</li> </ol>	<ol style="list-style-type: none"> <li>a. If you want to use Visual Basic to complete this lab:               <ul style="list-style-type: none"> <li>• Use Windows Explorer to browse to the <b>E:\Labfiles\Starter\VB</b> folder, and then double-click the <b>VB.sln</b> file.</li> </ul> </li> <li>b. If you want to use Visual C# to complete this lab:               <ul style="list-style-type: none"> <li>• Use Windows Explorer to browse to the <b>E:\Labfiles\Starter\CS</b> folder, and then double-click the <b>CS.sln</b> file.</li> </ul> </li> </ol> <p>The solution contains the Web application and two Web services.</p>
<ol style="list-style-type: none"> <li>2. Implement Forms authentication for the Web application.</li> </ol>	<ol style="list-style-type: none"> <li>a. If you are working with Visual Basic, in Solution Explorer, expand the E:\...\VB\ project.</li> <li>b. If you are working with Visual C#, in Solution Explorer, expand the E:\...\CS\ project.</li> <li>c. Open the Web.Config file in the root folder of the Web application.</li> <li>d. Modify the Web.Config file to configure the application to use Forms authentication. Specify a login page called <b>MembersLogin.aspx</b> in the root folder of the Web site. Do not store any user credentials in Web.Config.</li> </ol> <div style="display: flex; align-items: center;">  <p>See the resource in the Resource Toolkit, “How to: Configure Authentication for a Web Application.”</p> </div>

(continued)

Tasks	Supporting information
<p>3. Configure authorization for anonymous users and members.</p>	<ol style="list-style-type: none"> <li>Examine the Web application in Solution Explorer. Notice that there is a folder in the application called <b>Members</b>.</li> <li>Run the Web application. Use the <b>Members</b> menu to browse to the Bike Reviews page, the Trail Reports page, and the Employees page. Although you are currently an anonymous user, you gain access without any restriction. Close the browser.</li> <li>Add a new Web configuration file called <b>Web.config</b> to the Members folder.</li> <li>Modify the Web.config file to configure authorization for the Members folder as follows: <ul style="list-style-type: none"> <li>Allow a user called <b>Sally</b> access to the folder.</li> <li>Allow users in the role called <b>Members</b> access to the folder.</li> <li>Deny access to all other users.</li> </ul> <p>The user Sally and the Members role will be created and configured later in the lab.</p> </li> <li>Start the application, and then browse to the Bike Reviews page. Notice that the browser cannot locate the resource. This is because the login page has not yet been created.</li> <li>Close the browser.</li> </ol> <p> See the resource in the Resource Toolkit, “How to: Configure Authorization for a Web Application.”</p>
<p>4. Configure IIS.</p>	<p> <b>Note</b> To demonstrate Windows authentication, you will use the IIS Web server and configure it to allow only Basic authentication. Basic authentication always prompts for credentials, and you can easily see when authentication has occurred.</p> <p>To configure IIS to use Basic authentication, take the following steps:</p> <ol style="list-style-type: none"> <li>On the <b>Start</b> menu, click <b>Run</b>, and then in the <b>Open</b> box, type the following command (type <b>Pa\$\$w0rd</b> for the Administrator password when prompted): <pre>Runas /user:administrator "mmc %windir%\system32\inetsrv\iis.msc"</pre> </li> <li>Expand the LON-DEV-01\Web Sites folder.</li> <li>Right-click <b>Default Web Site</b>, and then click <b>Properties</b>.</li> <li>In the <b>Default Web Site Properties</b> dialog box, click the <b>Home Directory</b> tab.</li> <li>If you are working with Visual Basic, in the <b>Local Path</b> box, type the following path: <pre>E:\Labfiles\Starter\VB</pre> </li> </ol>



*(continued)*



Tasks	Supporting information
4. <i>(continued)</i>	<p>f. If you are working with C#, in the <b>Local Path</b> box, type the following path: <b>E:\Labfiles\Starter\CS</b></p> <p>g. Click the <b>Directory Security</b> tab, and then click <b>Edit</b> in the <b>Anonymous access and authentication control</b> section.</p> <p>h. Enable Basic authentication and disable Integrated Windows authentication for the Web site. Permit anonymous access.</p> <p>i. Click <b>OK</b> twice.</p> <p>j. In the Internet Information Services console, expand the Default Web site, right-click the <b>Employees</b> folder, and then click <b>Properties</b>.</p> <p>k. On the <b>Directory</b> tab, in the <b>Application Settings</b> section, click the <b>Create</b> button to create a new application start point. You need to create a second ASP.NET application in order to use a different authentication configuration.</p> <p>l. Click <b>OK</b>, and then close Internet Information Services.</p> <p>m. Open Internet Explorer, and then browse to <b>http://localhost</b>.</p> <p>n. Browse to the Employees page. Notice that you are not prompted for credentials.</p> <p>o. Close Internet Explorer.</p> <p> See the resource in the Resource Toolkit, “How to: Configure Authorization for a Web Application.”</p>
5. Implement Windows authentication for the Employees page.	<p>a. In Visual Studio, add a new Web configuration file to the Employees folder.</p> <p>b. Modify the new Web.config file to restrict access to the Employees folder. Your markup must:</p> <ul style="list-style-type: none"> <li>• Configure the folder to allow only Windows authentication.</li> <li>• Allow access for the Windows account LON-DEV-01\Student.</li> <li>• Deny access for every other user.</li> </ul> <p>c. Click <b>Save All</b> on the <b>File</b> menu.</p> <p>d. Start Internet Explorer, and then browse to <b>http://localhost</b>.</p> <p>e. Browse to the Employee page. Notice that you are now prompted to log on. Verify that you can log on as <b>LON-DEV-01\Student</b> with a password of <b>Pa\$\$w0rd</b>.</p> <p>f. Close Internet Explorer.</p> <p> See the resource in the Resource Toolkit, “How to: Configure Authentication for a Web Application.”</p>

## Exercise 2


### Implementing a Membership Registration Page

In this exercise, you will install the database for the SQL Server membership provider that you will use to store user accounts. You will configure the Web application to use ASP.NET Membership with the SQL Server membership provider. You will also create pages that enable users to create accounts for themselves and update their details.


### Scenario

Tasks	Supporting information
1. Install the SQL Server provider database.	<p>a. On the <b>Start</b> menu, click <b>Run</b>, and then type the following command to use the Administrator account to open a command prompt (type <b>Pa\$\$w0rd</b> when prompted for the Administrator password):</p> <p><b>Runas /user:administrator cmd</b></p> <p>b. At the command prompt, change to the following folder:</p> <p><b>C:\Windows\Microsoft.NET\Framework\v2.0.50727</b></p> <p>c. Type the following command:</p> <p><b>Aspnet_regsql -E -S localhost -A all</b></p> <p>This command creates a new SQL Server database to host the membership information for ASP.NET Web sites.</p> <p> <b>Note</b> Do not attempt to copy and paste this command from this document or the lab answer key to the command prompt. You must type this command manually at the command prompt.</p> <p>d. When the command has completed, close the command prompt.</p>
2. Configure the ASP.NET SQL Server membership provider.	<p>a. Modify the Web.config file in the root folder of the Web application. Configure the application to use the SQL Server membership provider as follows:</p> <ul style="list-style-type: none"> <li>• Add a connection string to connect to the aspnetdb database on the localhost.</li> <li>• Add an SQL Membership provider using that connection string.</li> <li>• Permit password retrieval.</li> <li>• Permit password reset.</li> <li>• Require each user to have a unique e-mail address.</li> <li>• Store passwords in clear form.</li> </ul> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “Introduction to Membership”</li> <li>▪ “How to: Configure an ASP.NET Application to Use Membership”</li> </ul>

*(continued)*

Tasks	Supporting information
<p>3. Create the membership registration page.</p>	<ul style="list-style-type: none"> <li>a. Add a new Web form to the root folder of the Web application. Call the new page <b>Register.aspx</b>, and then select the <b>Select master page</b> check box.</li> <li>b. Select <b>TopLevel.master</b> for the master page.</li> <li>c. Add a <b>CreateUserWizard</b> control to the <b>Content</b> control on the Web page. Set the <b>ContinueDestinationPageURL</b> property to <b>~/Default.aspx</b> and the <b>ForeColor</b> property to <b>Black</b>.</li> <li>d. Open the Web.sitemap file in the root folder of the Web application. Add the following tag to the Members &lt;siteMapNode&gt; element:  <pre>&lt;siteMapNode url="/Register.aspx"             title="Register" description="" /&gt;</pre> </li> <li>e. Click <b>Save All</b> on the <b>File</b> menu.</li> <li>f. Start Internet Explorer, and then browse to <b>http://localhost</b>.</li> <li>g. Use the <b>Members</b> menu to browse to the Register page.</li> <li>h. Register a new user account for yourself, using the strong password of <b>Pa\$\$w0rd</b>.</li> </ul> <p> <b>Note</b> When you have entered your details for registering as a new user, you must click <b>Create User</b> rather than pressing ENTER on the keyboard.</p> <ul style="list-style-type: none"> <li>i. Close Internet Explorer.</li> </ul>

*(continued)*



Tasks	Supporting information
<p>4. Create the membership update page.</p>	<ol style="list-style-type: none"> <li>a. Add another Web page to the root folder of the Web application. Call the new file <b>MemberUpdate.aspx</b>, and then use <b>TopLevel.master</b> as the master page.</li> <li>b. When the page is displayed, switch to Design view.</li> <li>c. Place a new <b>Label</b> in the <b>Content</b> control, and then set the label's <b>Text</b> property to <b>Username:</b> and its <b>ForeColor</b> property to <b>Black</b>.</li> <li>d. After this, on the same line, place a new <b>TextBox</b>. Set its <b>(ID)</b> property to <b>txtUsername</b> and its <b>ReadOnly</b> property to <b>True</b>.</li> <li>e. On a new line, add a second <b>Label</b>. Set its <b>Text</b> property to <b>E-mail:</b> and its <b>ForeColor</b> property to <b>Black</b>.</li> <li>f. On the same line, add a second <b>TextBox</b>. Set its <b>(ID)</b> property to <b>txtEmail</b>.</li> <li>g. In the <b>Page_Load</b> event handler for the <b>MemberUpdate.aspx</b> Web page, retrieve a <b>MembershipUser</b> object representing the current user. If this object is successfully obtained, display the <b>UserName</b> and <b>Email</b> properties in the text boxes; otherwise, redirect the user to the <b>Default.aspx</b> page, because the user is currently logged on anonymously.</li> <li>h. In the <b>Web.sitemap</b> file for your application, add the following tag to the <b>Members &lt;siteMapNode&gt;</b> element:  <pre>&lt;siteMapNode url="~/MemberUpdate.aspx"     title="Update Account" description="" /&gt;</pre> </li> <li>i. Click <b>Save All</b> on the <b>File</b> menu.</li> <li>j. Start Internet Explorer, and then browse to <b>http://localhost</b>.</li> <li>k. Browse to the <b>Update Account</b> page. Because you have not logged on, you should be redirected to <b>Default.aspx</b>. Close the browser.</li> <li>l. Back on <b>MemberUpdate.aspx</b>, add a new <b>Button</b> on a new line, below the <b>txtEmail</b> text box. Set the <b>(ID)</b> property to <b>btnUpdate</b> and the <b>Text</b> property to <b>Update E-mail</b>.</li> <li>m. Create a click event handler for <b>btnUpdate</b> that saves the e-mail address the user has entered to his or her membership account.</li> <li>n. Add a <b>ChangePassword</b> control to the <b>MemberUpdate.aspx</b> Web page, on a new line below <b>btnUpdate</b>. Set its <b>ForeColor</b> property to <b>Black</b>.</li> <li>o. Save all files.</li> </ol> <p>You will test the functionality on this page after the login page has been created.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “Introduction to Membership”</li> <li>▪ “How to: Manage Users by Using Membership”</li> <li>▪ “Login and Membership Page Design”</li> </ul> </div> </div>

## Exercise 3


### Implementing a Login Page and Adding Login Controls

In this exercise, you will create and complete the login page for members of your Web site. This page will also enable users to recover a password they have forgotten. You will add other types of login controls to other pages in the application.

#### Scenario

Tasks	Supporting information
<p>1. Create the login page and add the <b>Login</b> control.</p>	<p>a. Add a new Web Form called <b>MembersLogin.aspx</b> to the root folder of the Web application. Use <b>TopLevel.master</b> for the master page.</p> <p>b. Add a <b>Login</b> control to the Web page.</p> <p>c. Set the <b>ForeColor</b> of the <b>Login</b> control to <b>Black</b>.</p> <p>d. In the Web.sitemap file, add the following tag to the Members <b>&lt;siteMapNode&gt;</b> element:</p> <pre>&lt;siteMapNode url="~/MembersLogin.aspx"     title="Login" description="" /&gt;</pre> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “How to: Implement a Login Page and Add Login Controls”</li> <li>▪ “Login and Membership Page Design”</li> </ul>
<p>2. Add a <b>PasswordRecovery</b> Web server control to the login page.</p>	<p>a. Add a <b>PasswordRecovery</b> control to the MembersLogin.aspx Web page, below the <b>Login</b> control.</p> <p>b. Set the <b>ForeColor</b> property of the <b>PasswordRecovery</b> control to <b>Black</b>. Set the <b>MailDefinition-From</b> property to <b>admin@Adventure-Works.com</b>.</p> <p>c. Modify the Web.config file in the root folder of the application to drop outgoing e-mail messages into the E:\Labfiles folder, by adding the following <b>&lt;system.net&gt;</b> element to the <b>&lt;configuration&gt;</b> element:</p> <pre>&lt;system.net&gt;   &lt;mailSettings&gt;     &lt;smtp       deliveryMethod="SpecifiedPickupDirectory"       &lt;specifiedPickupDirectory         pickupDirectoryLocation "E:\Labfiles"/&gt;     &lt;/smtp&gt;   &lt;/mailSettings&gt; &lt;/system.net&gt;</pre> <p> See the resource in the Resource Toolkit, “How to: Implement a Login Page and Add Login Controls.”</p>

*(continued)*



Tasks	Supporting information
<p>3. Add login controls to other pages.</p>	<ul style="list-style-type: none"> <li>a. Open <b>TopLevel.master</b> in Design view.</li> <li>b. Place a <b>LoginStatus</b> control on a new line at the bottom of the blue table cell on the left of the page.</li> <li>c. Below the <b>LoginStatus</b> control, place a <b>LoginView</b> control on a new line.</li> <li>d. In the Anonymous template for the <b>LoginView</b> control, tell the users that they can access Bike Reviews and Trails Reports if they log in.</li> <li>e. In the Logged In template for the <b>LoginView</b> control, greet the users by name by using a <b>LoginName</b> control, and then remind them that they can access the Bike Reviews and Trail Reports.</li> </ul> <p> See the resource in the Resource Toolkit, “How to: Implement a Login Page and Add Login Controls.”</p>
<p>4. Test the login and membership features.</p>	<ul style="list-style-type: none"> <li>a. Click <b>Save All</b> on the <b>File</b> menu.</li> <li>b. Start Internet Explorer, and then browse to <b>http://localhost</b>. Notice that the <b>&lt;AnonymousTemplate&gt;</b> message appears and that the <b>LoginStatus</b> control displays a hyperlink with the text <b>Login</b>.</li> <li>c. On the <b>Members</b> menu, click <b>Register</b>. Register a new user called <b>Sally</b> with a password of <b>Pa\$\$w0rd</b>. Make a note of the password, e-mail address, security question, and security answer you use. The password must include numbers and symbols.</li> <li>d. Click <b>Continue</b>. You are now logged in as <b>Sally</b>.</li> <li>e. Notice that the <b>&lt;LoggedInTemplate&gt;</b> message now appears and that the <b>LoginStatus</b> control displays the message <b>Logout</b>.</li> <li>f. Access Bike Reviews and Trail Reports. Note that, as Sally, you are permitted access to the files.</li> <li>g. On the <b>Members</b> menu, click <b>Update Account</b>. Change the password. Make a note of the new value.</li> <li>h. Click the <b>LoginStatus</b> control to log out.</li> <li>i. On the <b>Members</b> menu, click <b>Login</b>. Use the password recovery feature to remind Sally of her password.</li> <li>j. Using Windows Explorer, browse to the folder E:\Labfiles. Double-click the mail file in the folder.</li> <li>k. If the Internet Connection Wizard appears, click <b>Cancel</b>. Microsoft Outlook® Express displays the e-mail message.</li> <li>l. Close Outlook Express and Internet Explorer.</li> </ul>

## Exercise 4


### Creating a Membership Management Administrative User Interface

In this exercise, you will configure the application to use ASP.NET Roles by using the SQL Roles provider. You will then build a Web page that Web site administrators can use to manage the membership of two roles, Admins and Members.

#### Scenario


Tasks	Supporting information
1. Configure the Web application to use the SQL Roles provider.	<p>a. Edit the Web.config file in the root folder of your application. Configure the application to use the SQL Server Roles provider, as follows:</p> <ul style="list-style-type: none"> <li>Use the same <code>connectionString</code> value that you used earlier for the Membership provider.</li> <li>Use the same <code>applicationName</code> value that you used earlier for the Membership provider.</li> <li>Cache the user role information in a cookie.</li> </ul> <p>b. Save all files.</p> <p> See the resource in the Resource Toolkit, “How to: Create a Membership Management Administrative User Interface.”</p> <p> <b>Note</b> You do not have to create the database for roles on the computer running SQL Server because the database that was created for membership earlier in the lab can be used for roles as well.</p>
2. Complete the Admin.aspx page.	<p>a. Open the Admin.aspx page in the Admins folder. Run the page by right-clicking it and then clicking <b>View in Browser</b>. Notice that nothing happens when you click the buttons on the page. Close the browser.</p> <p>b. Edit the <b>Page_Load</b> event handler for the Admin.aspx Web page. In this method, add code that performs the following tasks:</p> <ul style="list-style-type: none"> <li>Runs only when the page is not posted back</li> <li>Iterates through all the users of the Web site</li> <li>If a user is in the Members role, adds the user to the <b>lbxMembersRoleMembers</b> list box; otherwise, adds the user to the <b>lbxMembersRoleNonMembers</b> list box</li> <li>If a user is in the Admins role, adds the user to the <b>lbxAdminsRoleMembers</b> list box; otherwise, adds the user to the <b>lbxAdminsRoleNonMembers</b> list box</li> </ul> <p>c. Save all files.</p> <p>d. Run the page by right-clicking it and then clicking <b>View in Browser</b>. You will see the users you added earlier in the lab listed as nonmembers of both roles. Close the browser.</p>

(continued)

Tasks	Supporting information
2. (continued)	<p>e. Create a click event handler for the <b>btnMembersAdd</b> button. Add code to this handler that performs the following tasks:</p> <ul style="list-style-type: none"> <li>• Checks that a user is selected in the <b>lbxMembersRoleNonMembers</b> list box</li> <li>• Adds the user name to the <b>lbxMembersRoleMembers</b> list box</li> <li>• Removes the user name from the <b>lbxMembersRoleNonMembers</b> list box</li> <li>• Creates the Members role if it does not already exist</li> <li>• Adds the selected user name to the Members role</li> </ul> <p>f. Create a click event handler for the <b>btnAdminsAdd</b> button. Add code to this handler to perform the same tasks as in step e for the <i>Admins</i> role.</p> <p>g. Create a click event handler for the <b>btnMembersRemove</b> button. Add code to this handler to perform the following tasks:</p> <ul style="list-style-type: none"> <li>• Checks that a user name is selected in the <b>lbxMembersRoleMembers</b> list box</li> <li>• Adds the user name to the <b>lbxMembersRoleNonMembers</b> list box</li> <li>• Removes the user name from the <b>lbxMembersRoleMembers</b> list box</li> <li>• Removes the selected user name from the Members role</li> </ul> <p>h. Create the default click event handler for the <b>btnAdminsRemove</b> button. Add code to this handler to perform the same tasks as in step g for the <i>Admins</i> role.</p> <p>i. Edit the Web.sitemap file for your application. Add the following tag to the Members <code>&lt;siteMapNode&gt;</code> element:</p> <pre>&lt;siteMapNode url="~/Admins/Admin.aspx"     title="Administration" description="" /&gt;</pre> <p>j. Save all files.</p> <p>k. Start Internet Explorer, and then browse to <b>http://localhost</b>.</p> <p>l. Browse to the <b>Register.aspx</b> page. Add another user account.</p> <p>m. On the <b>Members</b> menu, click <b>Administration</b>. Use the page to add all of the users to the Members role.</p> <p>n. Add your own account to the Admins role.</p> <p>o. Close the browser.</p> <p> See the resource in the Resource Toolkit, “How to: Create a Membership Management Administrative User Interface.”</p>



*(continued)*

Tasks	Supporting information
<p>3. Secure the Administration page.</p>	<ol style="list-style-type: none"> <li>a. Add a new Web configuration file to the Admins folder.</li> <li>b. Edit the new Web.config file to: <ul style="list-style-type: none"> <li>• Allow members of the Admins role.</li> <li>• Deny all other users.</li> </ul> </li> <li>c. Save all files.</li> <li>d. Start Internet Explorer, and then browse to <b>http://localhost</b>.</li> <li>e. Browse to the login page. Log in as the user Sally.</li> <li>f. On the <b>Members</b> menu, click <b>Administration</b>. Verify that you cannot access the Administration page.</li> <li>g. Return to the home page. Click <b>Logout</b>. Log in again as yourself.</li> <li>h. Browse to the Administration page again. Verify that you can access this page because your account is a member of the Admins role.</li> <li>i. Close the browser.</li> </ol> <p> See the resource in the Resource Toolkit, “How to: Configure Authorization for a Web Application.”</p>

## Lab Shutdown

After you complete the lab, you must shut down the **2543B-LON-DEV-01-07** virtual machine and discard any changes.

## Lab Discussion



- Authentication and authorization for a Web application
- Membership registration pages
- Login pages and login controls
- Membership management user interfaces

In this lab, you:

- Configured authentication and authorization for a Web application.
- Implemented a membership registration page.
- Implemented a login page and used other login controls.
- Created a membership management administrative user interface.

The instructor will lead a group discussion of these tasks, and you should be prepared to contribute to the discussion, especially if you encountered problems completing the exercises.

