Security Fundamentals

Today you will learn

- Understanding Security
- Authentication and Authorization
- Forms Authentication
- Windows Authentication

Understanding Security

- In e-commerce example, you need to make sure that a user who successfully logs in can’t view another user’s recent orders.
- Use query string argument named userID, like this:

  http://localhost/InsecureStore/ViewOrders.aspx?userID=4191

- This is not secure, because any user can easily modify the userID parameter by editing the URL to see another user’s information.

Testing and Deploying Security Settings

Difference between Website in Visual Studio and Web Server:

- The website user account
- Security configuration:
  - Using Two ways:
    - By editing the web.config file
    - By running the Website Administration Tool (WAT)
- Other Security settings
  - Configure the specific protocols that IIS uses for Windows authentication

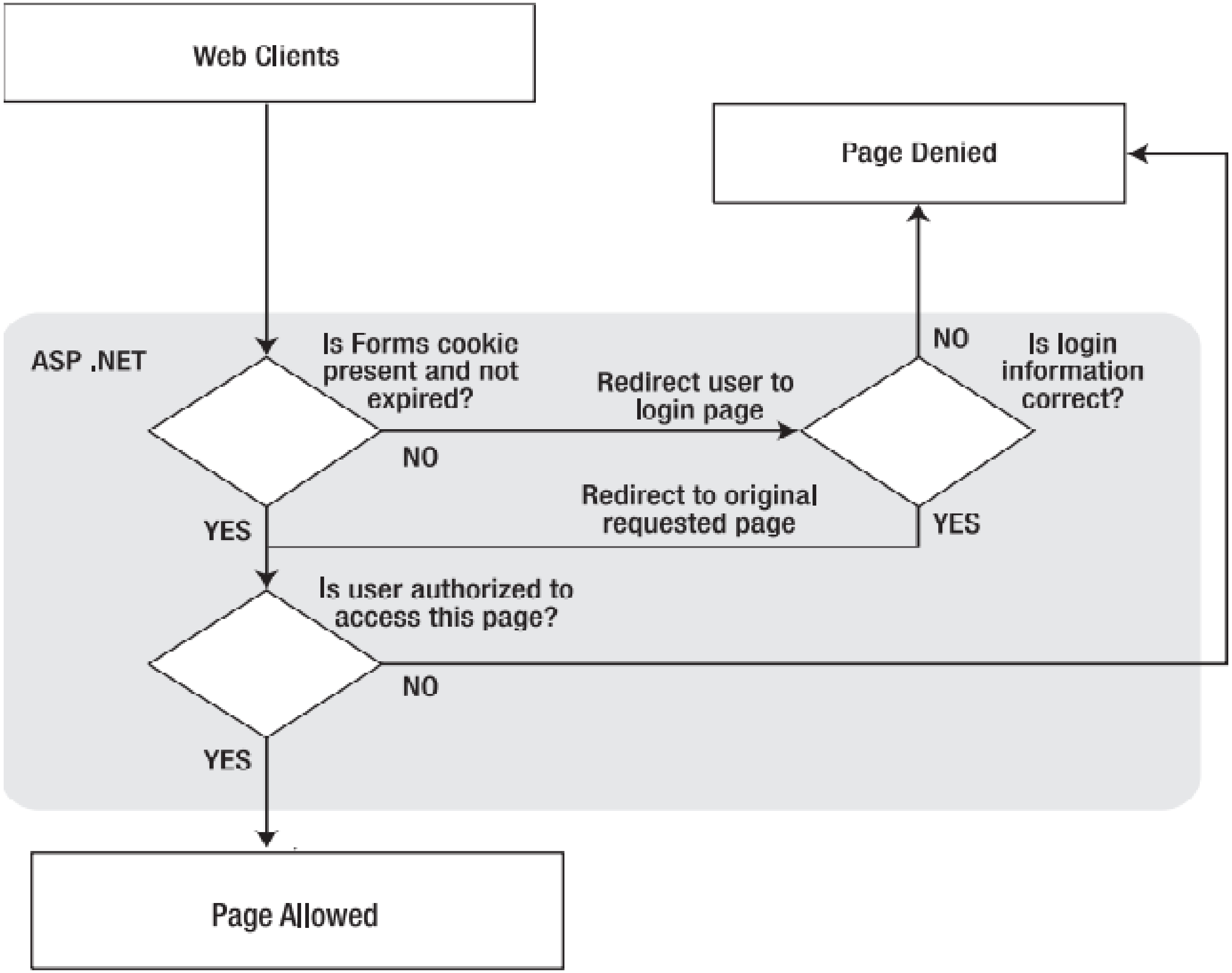
Authentication and Authorization

- **Authentication:** This is the process of determining a user’s identity and forcing users to prove they are who they claim to be.
- **Authorization:** Once a user is authenticated, authorization is the process of determining whether that user has sufficient permissions to perform a given action.
- You can use two types of authentication to secure an ASP.NET website:
  - **Forms authentication :** With forms authentication, ASP.NET is in charge of authenticating users, tracking them, and authorizing every request.
  - **Windows authentication:** With Windows authentication, the web server forces every user to log in as a Windows user.

Forms Authentication

- A common approach was to insert a little snippet of code at the beginning of every secure page. This code would check for the existence of a **custom cookie**. If the cookie didn’t exist, the user would be redirected to a login page, where the cookie would be created after a successful login.

Forms Authentication


Web Clients
Page Denied
ASP .NET
Is Forms cookie present and not expired?
Redirect user to login page
NO
Is login information correct?
NO
YES
Redirect to original requested page
YES
Is user authorized to access this page?
NO
YES
Page Allowed

Forms Authentication

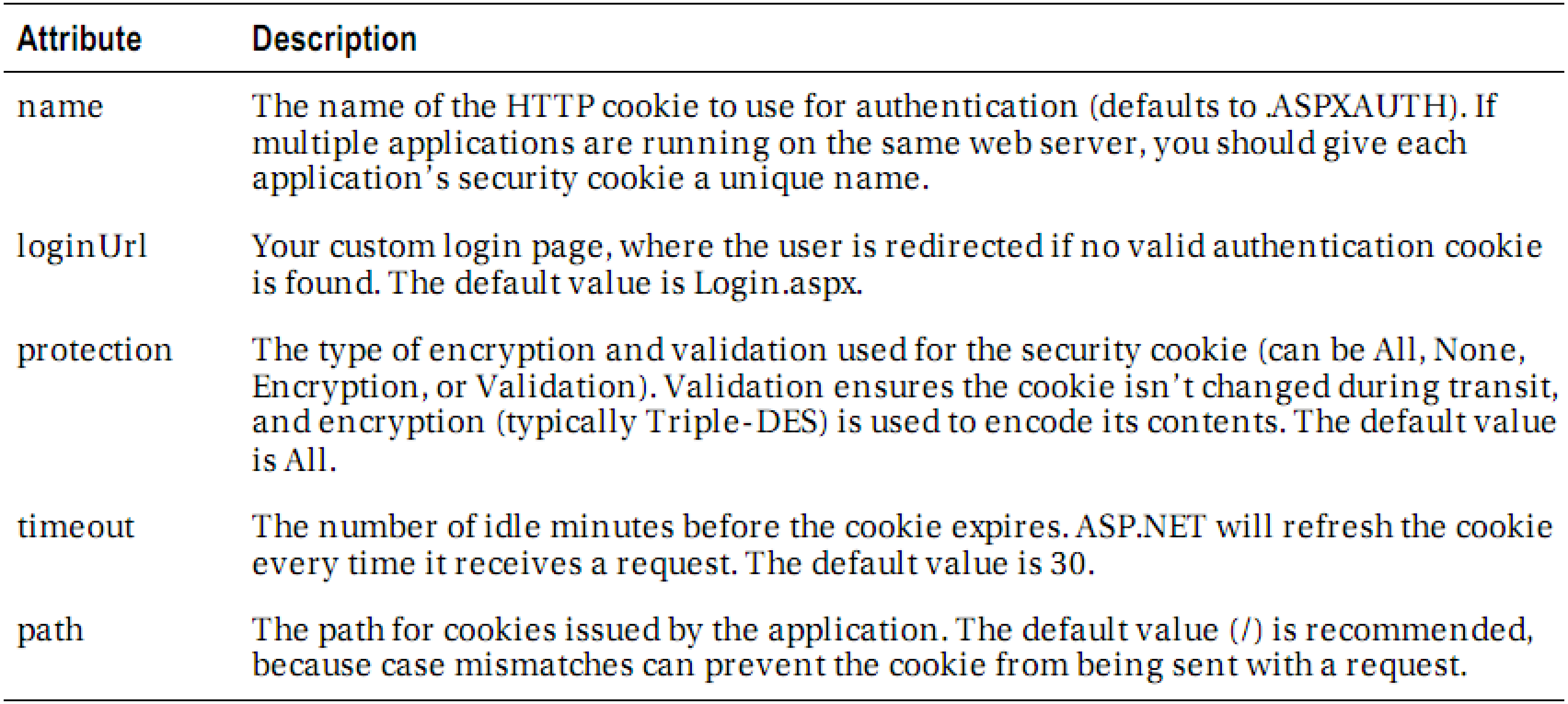
- **To implement forms-based security, you need to follow three steps:**
   - Set the authentication mode to forms authentication in the web.config file.
   - Restrict anonymous users from a specific page or directory in your Application.
   - Create the login page.

Web.config Settings

```
<configuration>
    <system.web>
        <authentication mode="Forms">
            <forms name="MyAppCookie"
                    loginUrl="~/Login.aspx"
                    protection="All"
                    timeout="30" path="/" />
        </authentication>
        ...
    </system.web>
</configuration>
```

Forms Authentication

Forms Authentication Settings



Attribute	Description
name	The name of the HTTP cookie to use for authentication (defaults to .ASPXAUTH). If multiple applications are running on the same web server, you should give each application's security cookie a unique name.
loginUrl	Your custom login page, where the user is redirected if no valid authentication cookie is found. The default value is Login.aspx.
protection	The type of encryption and validation used for the security cookie (can be All, None, Encryption, or Validation). Validation ensures the cookie isn't changed during transit, and encryption (typically Triple-DES) is used to encode its contents. The default value is All.
timeout	The number of idle minutes before the cookie expires. ASP.NET will refresh the cookie every time it receives a request. The default value is 30.
path	The path for cookies issued by the application. The default value (/) is recommended, because case mismatches can prevent the cookie from being sent with a request.

Authorization Rules

- To control who can and can't access your website, you need to add access control rules to the <authorization> section of your web.config file.

Forms Authentication

```
<configuration>
    <system.web>
        <authentication mode="Forms">
            <forms loginUrl="~/Login.aspx" />
        </authentication>

        <authorization>
            <allow users="*" />
        </authorization>
        ...
    </system.web>
</configuration>
```

- ASP.NET’s default settings allow all users.

Controlling Access to Specific Directories

```
<authorization>
    <deny users="?" />
</authorization>
```

Controlling Access to Specific Files

- Use the <location> tag to specify the control access to specific file.

Forms Authentication

```
<configuration>
    <system.web>
        <authentication mode="Forms">
            <forms loginUrl="~/Login.aspx" />
        </authentication>
         <authorization>
            <allow users="*" />
        </authorization>
        ...
    </system.web>

     <location path="SecuredPage.aspx">
        <system.web>
            <authorization>
                <deny users="?" />
            </authorization>
     </system.web>
    </location>

    <location path="AnotherSecuredPage.aspx">
        <system.web>
            <authorization>
                <deny users="?" />
            </authorization>
        </system.web>
    </location>
</configuration>
```

Controlling Access for Specific Users

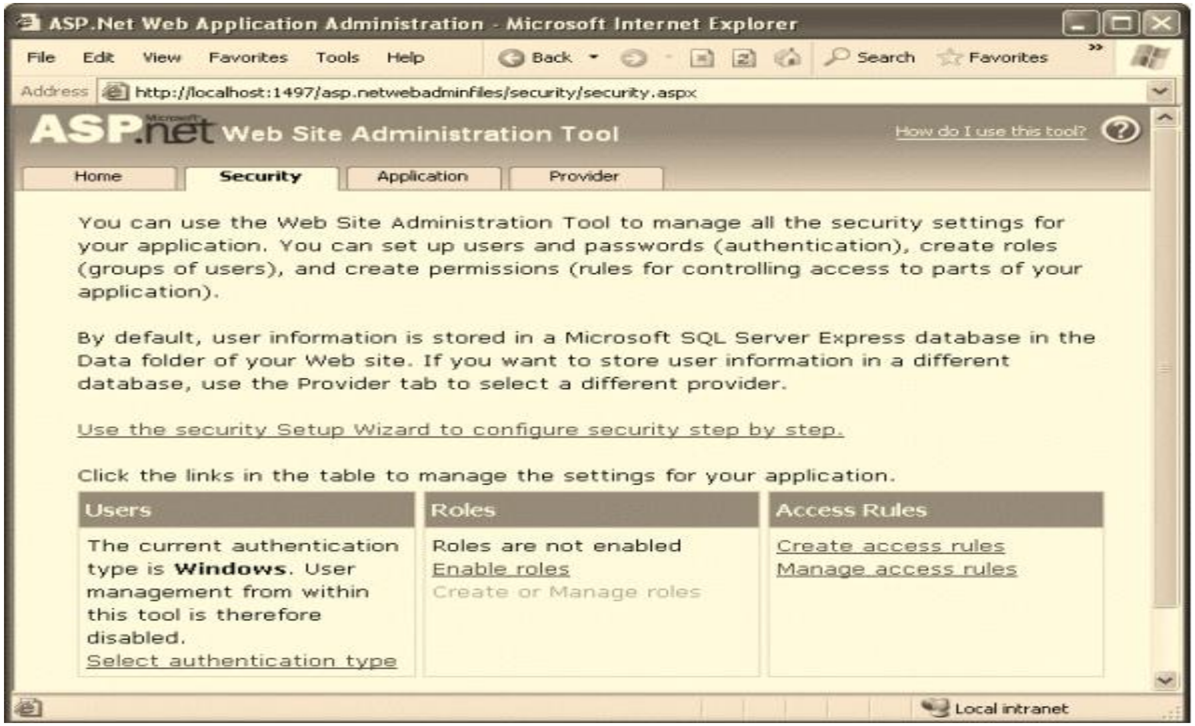
- The <allow> and <deny> rules can specifically identify a user name or a list of comma-separated user names.

```
<authorization>
        <deny users="?" />
        <deny users="matthew,sarah" />
        <deny users="john" />
        <allow users="*" />
</authorization>
```

The WAT

- **To set an application to use forms authentication, follow these steps:**
  - ○ Click Select Authentication Type.
  - ○ Choose the From the Internet option.
  - ○ Click Done. The appropriate <authorization> tag will be created in the web.config file.

To define the authorization rules, click the **Create Access Rules** link.


ASP.Net Web Application Administration - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost:1497/asp.netwebadminfiles/security/security.aspx
ASP.net Web Site Administration Tool
How do I use this tool?
Home
Security
Application
Provider
Local intranet

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.

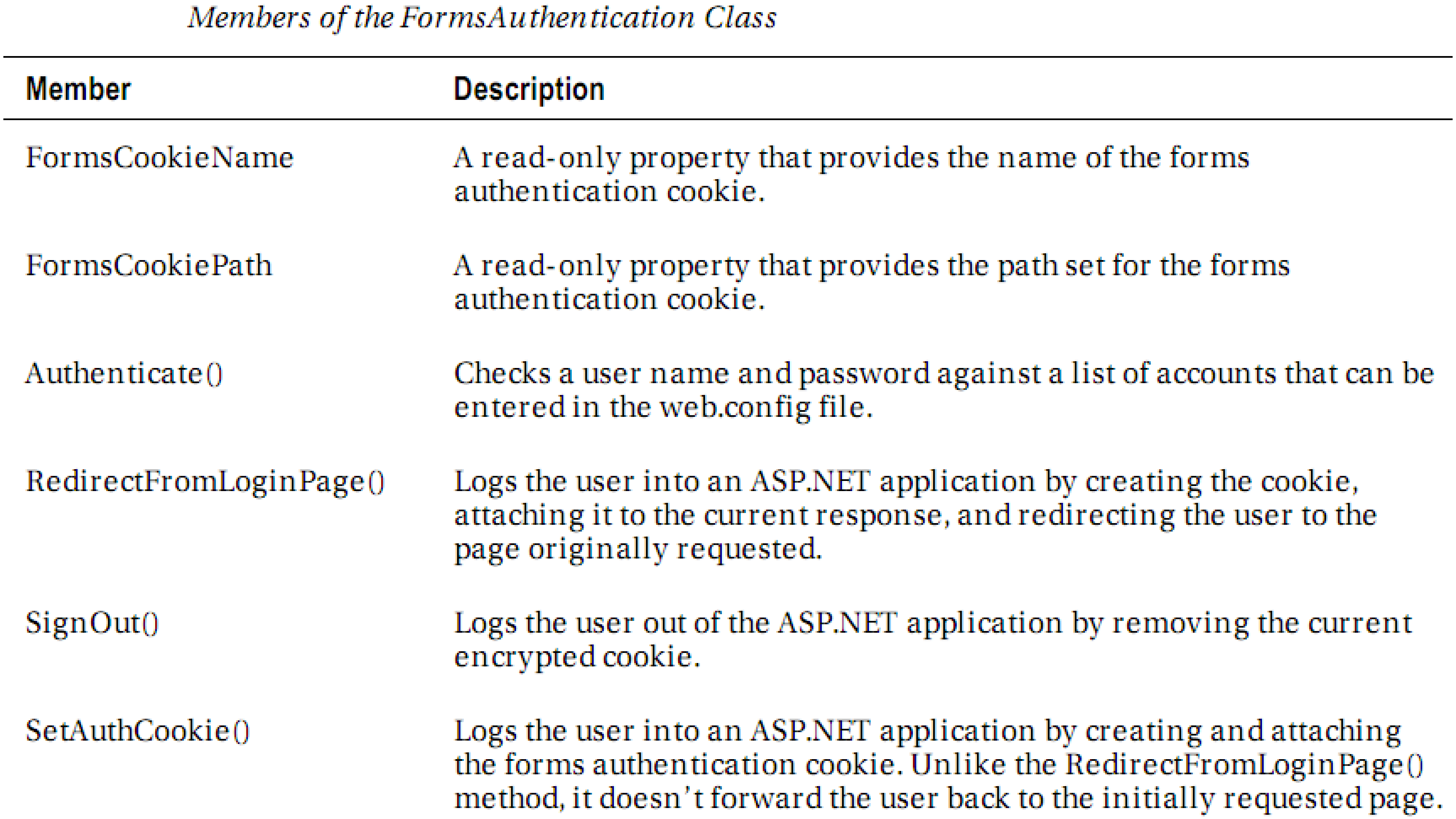
Use the security Setup Wizard to configure security step by step.

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

Users	Roles	Access Rules
The current authentication type is **Windows**. User management from within this tool is therefore disabled. Select authentication type	Roles are not enabled Enable roles Create or Manage roles	Create access rules Manage access rules

The Login Page

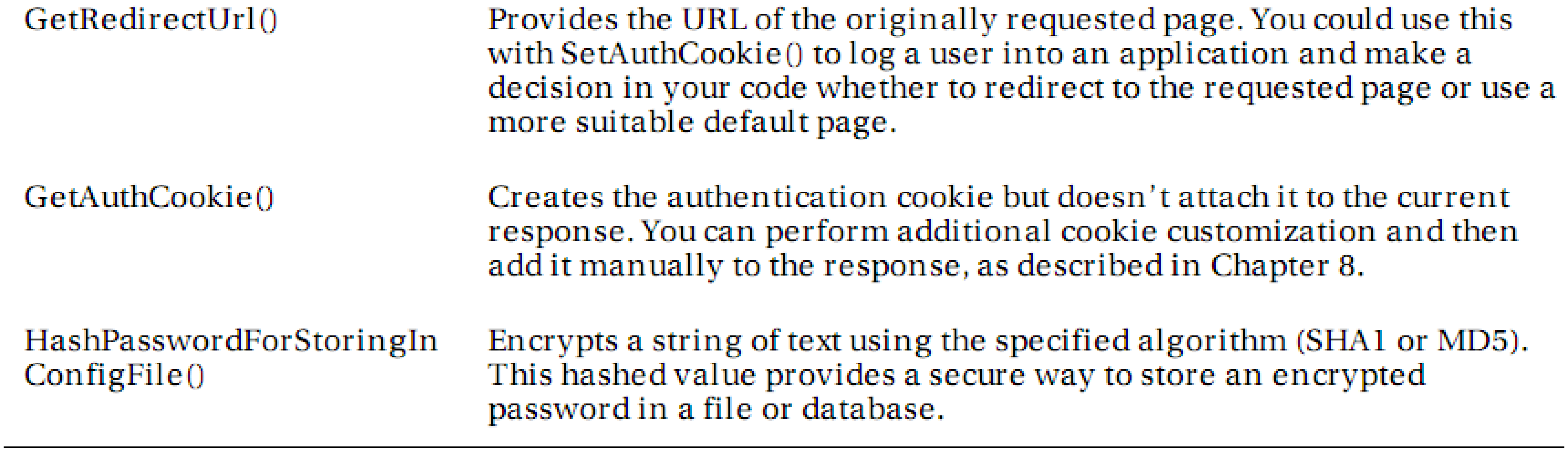
- FormsAuthentication class in the System.Web.Security namespace



Members of the FormsAuthentication Class

Member	Description
FormsCookieName	A read-only property that provides the name of the forms authentication cookie.
FormsCookiePath	A read-only property that provides the path set for the forms authentication cookie.
Authenticate()	Checks a user name and password against a list of accounts that can be entered in the web.config file.
RedirectFromLoginPage()	Logs the user into an ASP.NET application by creating the cookie, attaching it to the current response, and redirecting the user to the page originally requested.
SignOut()	Logs the user out of the ASP.NET application by removing the current encrypted cookie.
SetAuthCookie()	Logs the user into an ASP.NET application by creating and attaching the forms authentication cookie. Unlike the RedirectFromLoginPage() method, it doesn't forward the user back to the initially requested page.

The Login Page



GetRedirectUrl()	Provides the URL of the originally requested page. You could use this with SetAuthCookie() to log a user into an application and make a decision in your code whether to redirect to the requested page or use a more suitable default page.
GetAuthCookie()	Creates the authentication cookie but doesn't attach it to the current response. You can perform additional cookie customization and then add it manually to the response, as described in Chapter 8.
HashPasswordForStoringIn ConfigFile()	Encrypts a string of text using the specified algorithm (SHA1 or MD5). This hashed value provides a secure way to store an encrypted password in a file or database.

- The RedirectFromLoginPage() method requires two parameters.
  - The first sets the name of the user.
  - The second is a Boolean variable that specifies whether you want to create a persistent cookie.

Retrieving the User's Identity

User provides only one property and one method:

- The Identity property lets you retrieve the name of the logged-in user and the type of authentication that was used.

The Login Page

- The IsInRole() method lets you determine whether a user is a member of a given role.

Signing Out

Any web application that uses forms authentication should also feature a prominent logout button that destroys the forms authentication cookie:

```
protected void cmdLogin_Click(Object sender, EventArgs e)
{
        if (txtPassword.Text == "secret")
        {
                FormsAuthentication.RedirectFromLoginPage(txtName.Text, false);
        }
        else
        {
                lblStatus.Text = "Try again.";
        }
}
```

Persistent Cookies

- A persistent authentication cookie remains on the user’s hard drive and keeps the user signed in for hours, days, or weeks—even if the user closes and reopens the browser.
- Instead of RedirectFromLoginPage() method, manually create the authentication ticket, set its expiration time, encrypt it, attach it to the request, and then redirect the user to the requested page.

Windows Authentication

- Windows authentication, the web server takes care of the authentication process.
- ASP.NET simply makes this identity available to your code for your security checks.
- To implement Windows-based security with known users, you need to follow three steps:
    1. Set the authentication mode to Windows authentication in the web.config file.
    2. Disable anonymous access for a directory by using an authorization rule.
    3. Configure the Windows user accounts on your web server.

Web.config Settings

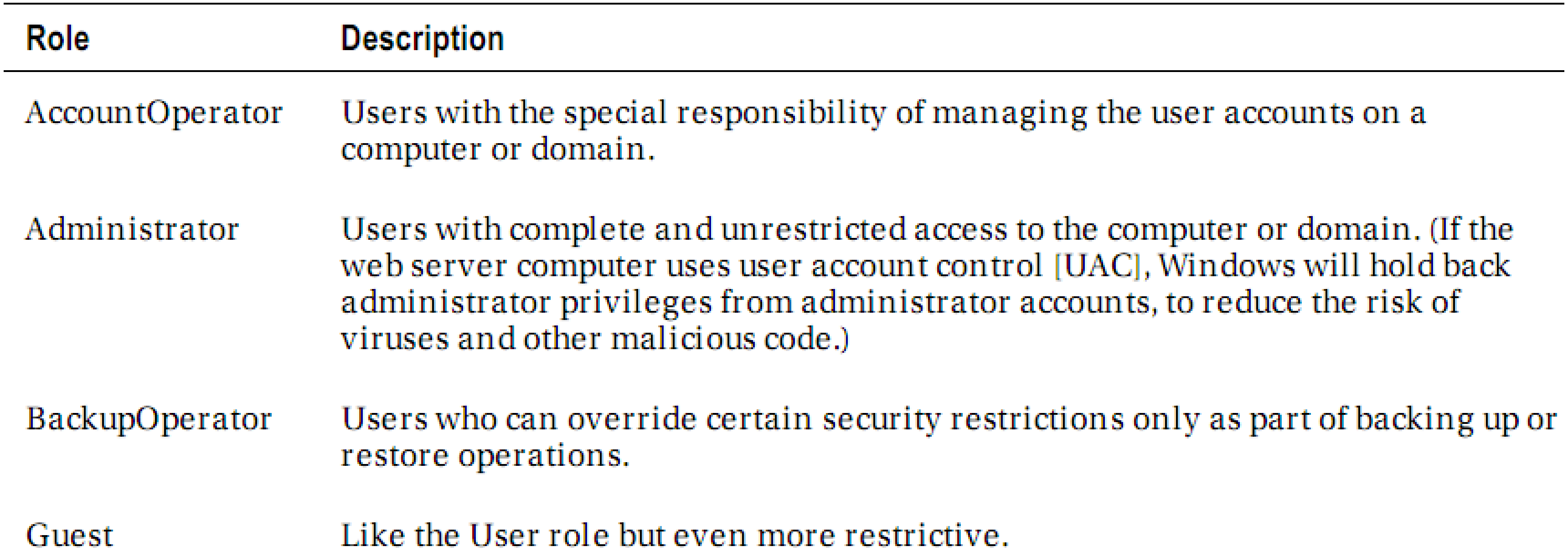
```
<configuration>
        <system.web>
                <authentication mode="Windows" />
                        <authorization>
                        <deny users="?" />
                </authorization>
        </system.web>
</configuration>
```

Windows Authentication

- This step is critical for Windows authentication (as it is for forms authentication). Without this step, the user will never be forced to log in.
- Add <allow> and <deny> elements to specifically allow or restrict users from specific files or directories.
- Example:

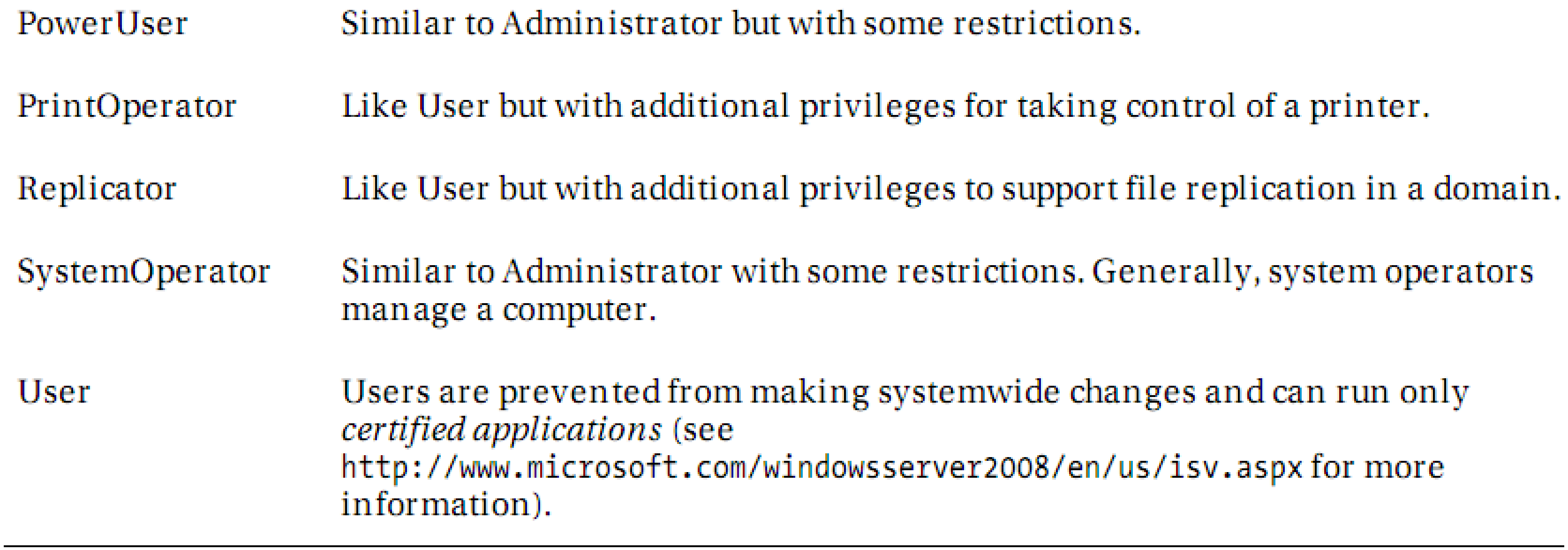
```
<allow users="WebServer\matthew" />

<allow users=".\matthew" />
```

Default Windows Roles



Role	Description
AccountOperator	Users with the special responsibility of managing the user accounts on a computer or domain.
Administrator	Users with complete and unrestricted access to the computer or domain. (If the web server computer uses user account control [UAC], Windows will hold back administrator privileges from administrator accounts, to reduce the risk of viruses and other malicious code.)
BackupOperator	Users who can override certain security restrictions only as part of backing up or restore operations.
Guest	Like the User role but even more restrictive.

Windows Authentication



PowerUser	Similar to Administrator but with some restrictions.
PrintOperator	Like User but with additional privileges for taking control of a printer.
Replicator	Like User but with additional privileges to support file replication in a domain.
SystemOperator	Similar to Administrator with some restrictions. Generally, system operators manage a computer.
User	Users are prevented from making systemwide changes and can run only *certified applications* (see `http://www.microsoft.com/windowsserver2008/en/us/isv.aspx` for more information).

END OF LECTURE