### *Agenda*

1. *What is Authentication and Authorization*
2. *Types of Authentication*
3. *Forms Authentication*
4. *Role based Authentication*
5. *Windows and Basic Authentication*
6. *What is ASP.NET Impersonation*
7. *Using location section in web.config*

## Authentication and Authorization

**Authentication:** The act of determining the identity of the requesting entity is known as authentication. The user must present the credentials such as a name / password pair in order to be authenticated.

**Authorization:** Once the request is authenticated, it must be determined whether that identity can access the given resource or not. This process is known as authorization.

**Types of Authentication:**

1. Forms Authentication
2. Windows Authentication
    a. Integrated Windows Authentication
    b. Basic Authentication

## Forms Authentication

Forms authentication uses an authentication ticket that is created when a user logs on to a site, and then it tracks the user throughout the site. The forms authentication ticket is usually contained inside a cookie.

Default configuration settings for Forms Authentication

```
<system.web>
 <authentication mode="Forms">
  <forms loginUrl="Login.aspx"
       protection="All"
       timeout="30"
       name=".ASPXAUTH"
       path="/"
       requireSSL="false"
       slidingExpiration="true"
       defaultUrl="default.aspx"
       cookieless="UseDeviceProfile"
       enableCrossAppRedirects="false" />
 </authentication>
</system.web>
```

**Step 1:** Add New Item → Web Configuration File (web.config) - (If the file does not exist):

/Web.Config

```
<system.web>
 <authentication mode="Forms">
  <forms name="__AUTHCOOKIE" defaultUrl="~/" loginUrl="~/LoginForm.aspx">
   <credentials passwordFormat ="Clear">
    <user name ="u1" password ="p1"/>
    <user name ="u2" password ="p2"/>
   </credentials>
  </forms>
 </authentication>
</system.web>
```

**Step 2:** To the project add a NewFolder by name "Secured" and in this folder add "web.config" file with following content.

**/Secured/Web.config**

```
<system.web>
 <authorization> <!--- This section can be present in any directory web.config file-->
  <deny users ="?,u1"/> <!--All Anonymous users and "u1 is denied access-->
  <allow users="*"/> <!--Allow access to all users except anonymous and u1.-->
 </authorization>
</system.web>
```

**Interpretation:** All the WebForms in the "Secured" folder are not accessible to Anonymous Users.

**Step3:** To the project, in secured folder add a webform by name UserDetails.aspx

**/Secured/UserDetails.aspx**

```
Username: <%= User.Identity.Name %> <br />
Authentication Type: <%= User.Identity.AuthenticationType.ToString() %>
```

**Step 4:** Add to the root directory of the application– LoginForm.aspx

**/LoginForm.aspx**

```
<div><asp:Label runat="server" ID="lblError" ForeColor="Red" /></div>
Login:  <asp:TextBox ID="txtLogin" runat="server"></asp:TextBox><br />
Password:  <asp:TextBox ID="txtPassword" runat="server"></asp:TextBox><br />
<asp:Button ID="btnLogin" runat="server" Text="Login" />
<asp:CheckBox ID="chkRememberMe" runat="server" Text="Remember Me" /><br />
```

**/LoginForm.aspx.cs: Handle btnLogin Click event.**

```
protected void btnLogin_Click(object sender, EventArgs e)
{
   string log = txtLogin.Text;
   string pwd = txtPassword.Text;
   if (FormsAuthentication.Authenticate(log, pwd)) //If log/pwd has to verified with our database, this line
should be  replaced with the required ADO.NET code.
   {
      FormsAuthentication.RedirectFromLoginPage(log, chkRememberMe.Checked);
      /* FormsAuthentication.SetAuthCookie(log,chkRememberMe.Checked);
      string returl = Request.QueryString["ReturnUrl"];
      if (returl == null)
         Response.Redirect(FormsAuthentication.DefaultUrl); //The user has directly come to the login page.
      else
```

```
            Response.Redirect(returl); */
    }
    else
        lblError.Text = "Invalid Login or Password";
}
```

**Step 5:** In **Default.aspx** of root directory add hyperlink for LoginForm.aspx and UserDetails.aspx

**Default.aspx**

```
<a href="Secured/UserDetails.aspx">UserDetails.aspx</a><br />
<a href="LoginForm.aspx">LoginForm.aspx</a>
```

**Step 6:** Make Default Page as the Startup Page and run the application.

From Default Page goto UserDetails.aspx

From Default Page goto Login.aspx

**Step 7:** Add Logout LinkButton to UserDetails.aspx.

**lbtnLogout**

```
protected void lbtnLogout_Click(object sender, EventArgs e)
{
    FormsAuthentication.SignOut();   //Adds authorization cookie to the response with an expired date
    FormsAuthentication.RedirectToLoginPage();
}
```

**Notes:**

1. If the request includes the cookie for authentication (**"__AUTHCOOKIE")** then ASP.NET treats the client as authenticated otherwise it is treated as Anonymous.
2. If anonymous users are denied access (<deny users="?"/>) and if the request is not authenticated then ASP.Net redirects the users to login URL ("<forms loginUrl=…").
3. If a user directly comes to Login Page and provides valid credentials (username/password), the user is then redirected to default url ("<forms **DefaultUrl**=…")
4. If an anonymous user tries to visit a secured page, the user is redirected to the login page and the url of the current secured page is appended as query string parameter "**ReturnUrl**" to the Url of LoginPage.
5. A user is redirected to Login page either if not authenticated or not authorized
6. If the User is authenticated, then based on the value of authentication cookie (__AUTHCOOKIE) the user is authorized and for this it looks in "<authorization>" section in web.config and the value of authentication cookie.
7. The value of **User.Identity.Name** is set from the value of __AUTHCOOKIE which mostly contains the username used for login.

**FormsAuthentication.Authenticate(string log, string pwd)**

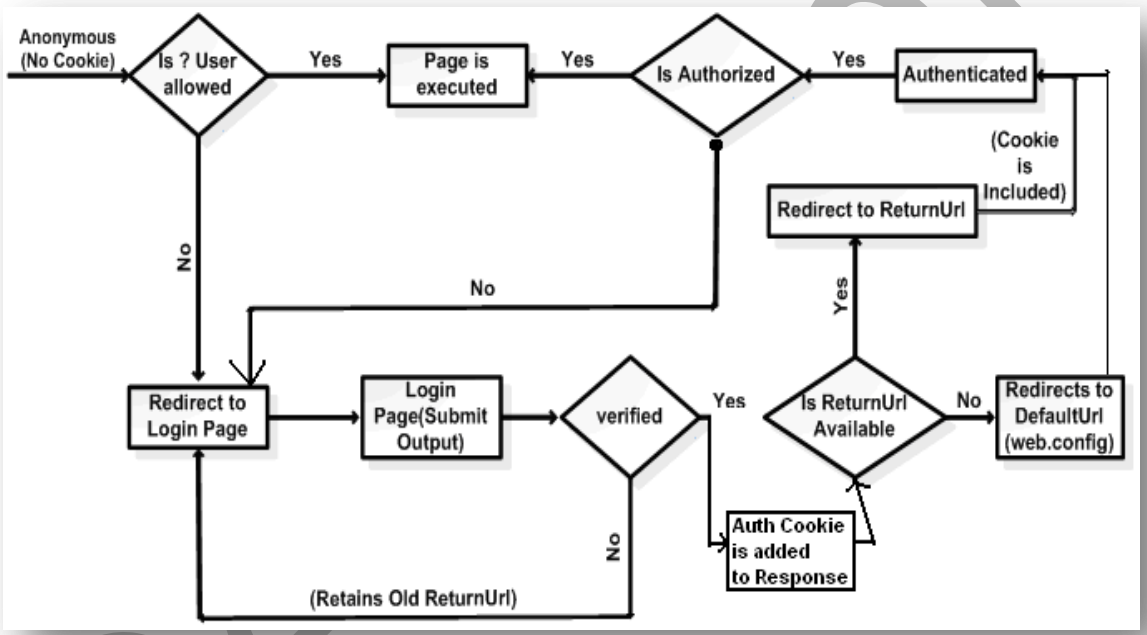
Verifies if the specified log and password exists in the credentical section of web.config or not. If present returns true otherwise returns false.

**FormsAuthentication.SetAuthCookie(string loginname, bool isPersistentCookie)**

This function adds to the response of the authentication cookie, the name of the cookie is taken from forms tag in web.config and the value of the cookie is the first parameter. If the second parameter is true the cookie created is persistant cookie otherwise created a non-persistent cookie.

**FormsAuthentication.RedirectFromLoginPage(string loginname, bool isPersistentCookie)**

1. Add the authentication cookie to the response so that for all subsequent request the client can be treated as authenticated.
2. Redirects the user to the Url as mentioned in the QueryString parameter "**ReturnUrl**". If ReturnUrl is not present, the request is redirected to "**DefaultUrl**" as in forms tag of root directory web.config




**To protect the password in web.config file (credential section)**

**Step 8:** Add Encrypt Button (btnEncrypt) and TextBox (txtPassword) to Default.aspx

**/LoginForm.aspx.cs: Handle btnLogin Click event.**

```
protected void btnEncrypt_Click(object sender, EventArgs e)
{
   Response.Write(FormsAuthentication.HashPasswordForStoringInConfigFile(txtPassword.Text, "SHA1"));
}
```

**Step 9:** Run the application and replace "p1", "p2' and "p3" in web.config with their hashed values.

**Step 10:** Change in /web.config

Replace <credentials passwordFormat ="**Clear**"> to <credentials passwordFormat ="**SHA1**">

**SHA1 (Secured Hash Algorithm):** 160 bit encryption & **MD5(Message Digest):** 128 bit encryption

– Using both we can only encrypt data as they are One Way Hash Algorithm.

## Role Based Authentication

Roles should be assigned to the request based on identity of the authenticated user and this should be done in "**Application_AuthenticateRequest**" in Global.asax (Global Application Class).

**Step 11:** Project → Add New Item → Global Application Class **(Global.asax)**

```
protected void Application_AuthenticateRequest(object sender, EventArgs e)
{
   if (Request.IsAuthenticated)
   {
      string[] arRoles = new string[2];
      switch (User.Identity.Name)
      {
         case "u1":
            arRoles[0] = "r1"; arRoles[1] = "r4";
            break;
         case "u2":
            arRoles[0] = "r2"; arRoles[1] = "r4";
            break;
         default:
            arRoles[0] = "r3"; arRoles[1] = "r4";
            break;
      }
      Context.User = new System.Security.Principal.GenericPrincipal(User.Identity, arRoles);
   }
}
```

**Step 12:** Based on Role we can allow or deny access to the resources of the Secured folder.

**Default.aspx**

```
<authorization>
 <deny users ="?"/>
 <deny roles="r1"/> - Deny access to all users with Role "r1".
</authorization>
```

**Step 13:** Run the application try to access secured page with u/p as "u1" and "p1". Even though ASP.NET

authenticates the request, the user is redirected back to Login page because for identity "u1" role assigned is "r1" & "r4" and "r1" is denied access to secured folder.

**Step 14:** To show or hide a portion of a page based on role of the current user.

Add a button with id "**btnForR2"** to Userdetails.aspx

***In /SecurityDemo/UserDetails.aspx***

***Page_Load***

```
protected void Page_Load(object sender, EventArgs e)
{
   btnForR2.Visible = User.IsInRole("r2"); //Checks if the current user has the role "r2" or not?
}
```

## Windows Authentication

1. You can use Windows authentication when your IIS 7 server runs on a corporate network that is using Microsoft Active Directory service domain identities or other Windows accounts to identify users .
2. When you enable Windows authentication, the client browser sends a strongly hashed version of the password in a cryptographic exchange with your Web server.
3. Windows authentication is best suited for an intranet environment for the following reasons:
    a. Client computers and Web servers are in the same domain.
    b. Administrators can make sure that every client browser is Internet Explorer.
    c. HTTP proxy connections, which are not supported by NTLM, are not required.
    d. Kerberos v5 requires a connection to Active Directory, which is not feasible in an Internet environment
4. By default the web browser along with the request does not include any identity of the client. In such cases web server treats such request as **Anonymous** request. If anonymous users are allowed access authentication is not required else Authentication has to be performed by Web server.
5. **To configure Windows Authentication in IIS visit the following link:** http://www.iis.net/configreference/system.webserver/security/authentication/windowsauthentication
6. It is not supported by Windows Vista and Windows 7 Home editions.

## Basic Authentication

- In Basic authentication requires that the user provides the username and password to access the content
- Credentials submitted as plain text unless HTTPS is used.
- It works with all the web browsers.
- It is a good choice when you want to restrict access to some, but not all, content on a server.

**To Configure Basic Authentication**

1. Open IIS Manager and navigate to the level you want to manage → Double click on Authentication (Under IIS Section)
2. Enable Basic Authentication.

**Example:**

**Step 1: Create a New WebApplication using IIS (HTTP)**

File → New → WebSite (**Location=HTTP**) (URL: Http://localhost/DemoApp)

**Step 2:** In "/Web.config" (Applications Root directory Web.Config) verify that the Authentication mode is Windows i.e. Look for <authentication mode="**Windows**" />

**Note: This section can be present only in the root directory web.config and thus for one Web Application only one type of Authentication mode is supported.**

**Step 3:** To the Project add a New Folder by name "Secured"

**Step 4:** In Secured folder add a new webform by name "UserDetails.aspx"

**/Secured/UserDetails.aspx**

```
Is Authenticated: <%= User.Identity.IsAuthenticated.ToString() %> <br />
Username = <%= User.Identity.Name %> <br />
Authentication Type = <%= User.Identity.AuthenticationType.ToString() %> <br />
```

**Step 5:** Run the application, Visit UserDetails.aspx (Startup Page). Following o/p is rendered

Is Authenticated: False

Username =

Authentication Type =

**Note: All users including Anonymous users are allowed access to Web Forms in "Secured" folder**

**Step 6:** To deny access to Anonymous users

a) Add to secured folder "web.config" file

b) To it add the following tags

**web.config**

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

**Step 7:** View Userdetails.aspx in Web Browser and o/p is as below

Is Authenticated: True

Username = BUILTIN\Administrator*

Authentication Type = Negotiate

* <Client Machine Name> \ <Client Identity> with which we have logged in client machine

**Step 8:** Add the following line to "\Secured\web.config"

**web.config**

```
<deny roles="BUILTIN\Administrators"/>
```

**Step 10:** Run the application and if we have logged onto client machine using identity which has role "Administrators", then the secured page is not rendered and browser shows Login dialog asking us to provide another u/p.

**Limitation:**

Windows Authentication is good only for Intranet application because for every identity supported on the server a login has to be created on the domain of the web server**.**

## Impersonation

By default every request for an ASP.NET page when executing on server, uses the identity of the user "**ASPNET 4.5**". Using this code executing during that request can access the all the local and remote resources on the web server.

```
<system.web>
        <identity impersonate="true"/>
</system.web>

<system.webServer>
        <validation validateIntegratedModeConfiguration="false"/>
</system.webServer>
```

With the above tags in **web.config**, the identity of the client (webbrowser) submitted to server is used by ASP.NET to access the local and remote resources.

Note: If the client has not logged in, ASP.NET uses the identity of "IUSER_<MachineName>" for identifying the client and if impersonating the same identity is used for accessing local and remote resources on the server.

**Step1:** Create a new website project using IIS (not ASP.NET Dev. Server) and add the following to /Default.aspx:

//To get the Windows Identity of the user being used by ASP.NET for accessing the resources on the machine:

<%= **Environment.Username %**>

**Step2:** In /web.config add the following

<**identity impersonate**="false"/>

**Step3:** Run the application and see the o/p as below

<MachineName>**ASPNET**

**Step4:** Change from <identity impersonate="false"/>

to <identity impersonate="true"/>

**Step5:** Run the application and see the o/p as below

<MachineName>**IUSR_<MachineName>**

**Step6:** Add the following to web.cofig (deny access to anonymous users).

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

**Step7:** Run the application and see the o/p as below

<MachineName>\<**Identity of client** to login to its machine>

**To use identity of a fixed user:**

<identity **impersonate**="true" **userName**="loginname" **password**="pwd"/>

## Using Location Section

1. In a given directory web.cofig, if we want to provide different configuration settings for a given WebForm, the settings must be enclosed in <location> with path specifying the WebForm.

**Default.aspx**

```
<configuration>
 <location path="Unsecured.aspx">
  <system.web>
   <authorization>
    <allow users="*"/>
   </authorization>
  </system.web>
 </location>
</configuration>
```

2. If any Configuration Setting is enclosed within **<location allowOverride="false">**, it is locked and cannot be overriden in the subdirectory configuration file.

**Note: Location must be nested immediately under <Configuration> tag. (after configSections…)**