|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Forms Authentication Tutorial   **[** [**printer friendly**](http://devhood.com/tutorials/tutorial_details.aspx?tutorial_id=85&printer=t) **]** | |  | | --- | | Stats | | |  | | --- | | Rating: 4.55 out of 5 by 49 users | | Submitted: 12/02/01 | | | | [Andrew Ma](http://devhood.com/public.aspx?user_id=00014A50872078CB) ([ajmaonline@hotmail.com](mailto:ajmaonline@hotmail.com)) |  | |
|  |
| |  |  |  | | --- | --- | --- | |  |  |  | | What is authentication? "To establish the authenticity of; prove genuine" (The American Heritage® Dictionary of the English Language). ASP.NET has buildt in authentication support for web page development making it really easy for any web developer to add personalized webpages or password-protect areas.  This tutorial is aimed at people who already have a good understanding of ASP.NET. If you want to learn ASP.NET, I have a beginner tutorial here: [Guestbook.NET - a beginner tutorial to ASP.NET](http://www.devhood.com/Tutorials/tutorial_details.aspx?tutorial_id=96)  There are three types of autentication in ASP.NET: forms authentication, windows authentication, and Passport authentication.  **Windows Authentication** - This is provided so that web pages can make use of the local Windows User and Groups.  **Passport Authentication** - This is Microsoft's initiative for a single authentication point. It is used for such things as Hotmail, MSN and Devhood. Find out more at [www.passport.com](http://www.passport.com/)  **Forms Authentication** - This is a cookie based authentication system where the username and passport is stored in a text file or a database. We will be focusing on this authentication model in this tutorial.  Let's start!  **web.config file**  The web.config file is an XML based configuration file which exists for every web application. The web.config file typical resides in the application root directory although it is possible to have multiple web.config files. If there is another web.config file placed in a directory below the application root, it will use those setting instead. The web.config file is where you will tell a web application to use either of the three types of autentication types.  Here we show you a basic example of what a web.config file looks like when it has be set to use form authentication. I will go in further detail and explain the tags.   |  | | --- | | <configuration>  <system.web>  <authentication mode="Forms">  <forms loginUrl="login.aspx" protection="All" timeout="30">  <credentials passwordFormat="Clear">  <user name="devhood" password="password"/>  <user name="someguy" password="password"/>  </credentials>  </forms>  </authentication>  <authorization>  <allow users="\*" />  </authorization>  </system.web>  <location path="admin/">  <system.web>  <authorization>  <allow users="devhood" />  <deny users="someguy" />  </authorization>  </system.web>  </location>  <location path="usersonly/">  <system.web>  <authorization>  <deny users="?" />  </authorization>  </system.web>  </location>  <location path="public/">  <system.web>  <authorization>  <allow users="\*" />  </authorization>  </system.web>  </location>  </configuration> |   The first tag in the web.config file is the <configuration> tag. It is the base tag for the web.config file and will contain all your configuration settings in here. The first <system.web> tag specifies the settings that will apply to all the file in the same directory and the files below this directory.  **<authentication> tag**  Here we come to our first tag for authentication, which is thence called <authentication>. We see that there is one attribute for this tag and it specifies the type of authentication that will be applied to this site. The choices are Windows|Forms|Passport|None. This tutorial focuses on Forms authentication so that's what we got in there.  **<forms> tag**  Next we move to the <forms> tag. This tag is unique to the Forms authentication mode and will specify things such as the loginUrl, the type of protection and the timeout of inactivity.  **loginUrl** attribute - when a user does not have the correct credentials to view a page, the user will be forwarded to this url.  **protection** attribute - there are four(4) types of protection modes, All|None|Encryption|Validation. For simplicity sake, we're not going to go into this now, but if you want to know more, consult the MSDN documentation.  **timeout** attribute - this is an integer value specifying the number of minutes a user's cookie will be valid for. After this period of inactivity, the user will be forced to re-authenticate.  **<credentials> tag**  This is an optional section if you want to specify the username/password combinations in here. We will first discuss authentication with passwords in the web.config file and I will later highlight how you can store the usernames and passwords in a database or XML file. The credentials tag also has an attribute called passwordFormat. Your choices for password format are: Clear|SHA1|MD5. We still stick with clear text passwords for now and talk about encrypting the passwords further down.  **<user> tag**  This is also an optional tag (since it resides in the optional credentials tag). This tag is pretty straight forward, name attribute for the username and password attribute for the password.  **<authorization> tag**  Now that we have specified our authentication type and the user accounts, we have to specify how the authentication is to be applied to our website. We used the authorization tag to mark this. The autorization tag is placed between the system.web tags. In the example above, we see that the authorization tag contains the <allow> tag. This allow tag will (as you can guess) specify which users have access to the website. And there is also a <deny> tag which will specify which users are denied access. The format of the users attributes is pretty simple. It's just a comma-delimited list of user names (i.e. users="jsmith, jdoe, blah"). There are also two special values that can be used in the users attribute. The first one is the \* (asterix) character. This is used to denote "all users". So the example above allows access to all users. The second one is the ? (question mark) character. This is used to denote "anonymous" users. You can use this to deny anonymous access which will force users to authenticate before getting into some webpages (see the examples in the locations tags).  **<location> tag**  Now what happens when we want some parts of the website to be protected and others to not be protected? ASP.NET did think of that and handles that by the <locations> tags. The location tag has one attribute, path, which is the path to apply a different set of security rules to. Inside the location tag, we have the system.web tag once again. The authorization tag is placed inside the system.web tag (just like the in first usage of <authorization>).  **login.aspx file**  Now that we have our web application all configured, we tackle the task of getting a user to authenticate themself by sending his/her username and password. In our <forms> tag, we specified that the loginUrl attribute is login.aspx and here is an example of a login page:   |  | | --- | | <html>  <head>  <title>Login</title>  <script language="C#" runat="server">  void Login\_Click(Object sender, EventArgs e) {  if (FormsAuthentication.Authenticate(username.Text, password.Text))  FormsAuthentication.RedirectFromLoginPage(username.Text, true);  else  status.InnerHtml += "Invalid Login";  }  </script>  </head>  <body>  <p class=title>Login</p>  <span id="status" class="text" runat="Server"/>  <form runat="server">  Username: <asp:textbox id=username cssclass="text" runat="Server"/><br />  Password: <asp:textbox id=password textmode=Password cssclass="text" runat="Server"/><br />  <asp:button id=login\_button onclick="Login\_Click" text=" Login " cssclass="button" runat="Server"/>  </form>  </body>  </html> |   First, let's look at what the user sees. Our simple webpage example has two textboxes and a button. This webpage will be shown to the user anytime a request is made for a page and the user is does not have the proper credentials. This is a simple example, which you'll probably want to modify. Now we look at the code, this is where the authentication is done and the cookies are sent to the browser.  FormsAuthentication.Authenticate  The single login button on the webpage calls the Login\_Click method when clicked. In this method, we use the FormsAuthentication.Authenticate(username,password) to get ASP.NET to check the credentials of the user. The parameters for this method is pretty straightforward and it just returns a boolean value.  FormsAuthentication.RedirectFromLoginPage  If the user is providing proper credentials, then we'll use the FormsAuthentication.RedirectFromLoginPage method. The parameters of this method are a username string and a boolean value. The first parameter is a username string and it is a name of the user for cookie authentication purposes. The value you put in there will be the user name that the client is associated with. It does not need to match the username used in the FormsAuthentication.Authenticate method but it is advisable to set the cookie to the username that was used to log in. The second parameter is a boolean value and it specifies whether or not a durable cookie (one that is saved across browser sessions) should be issued.  If you remember, when a user requests a page without proper authentication, they are redirected to the login page. After setting the cookie, the RedirectFromLoginPage will then send the user back to the page they came from.  There you go. You should have everything you need for a basic Forms based authentication system. Give it a try. If you want to extend the usage of authentication more, read on!  Advanced Part  Now we move to the more advanced topics on authenication. First we'll talk about encrypting your passwords then I'll tell you how to store your user accounts outside the web.config file.  **Encrypting passwords**  If you look at one of my tips, I say "Storing them as cleartext is asking for trouble." Why? Well, if your system gets comprimised, then someone can steal all your passwords. That's trouble. Also, sometimes people tend to use the same password in different sites because they don't want to memorize so many passwords. If you comprimse your password lists, then someone's going to be unhappy. That's trouble.  So how will you encrypt your passwords. ASP.NET has included some hashing functions to encrypt your passwords. What is a hashing function? Well, simply put, it sends your password to a function which spits out your password all garbled. The two hashing functions that SHA1 or MD5. Both hashing functions are suppose to not let malious users take the hashed password and get the original password. So how does this all work in ASP.NET? Passwords in the web.config file are stored as their hashed values. Then when a user tries to login, they will send they're password to the webserver and the webserver will hash the password and compare it to the hased password in the web.config file. If the two hashes match, then the password is correct. To hash a password, you can use the built in method called HashPasswordForStoringInConfigFile. You can call it like this: FormsAuthentication.HashPasswordForStoringInConfigFile("password","md5"). The first parameter is the password to be hashed. The second parameter is either "md5" or "sha1" depending on which hashing function you use. See your MSDN documentation for more detailed instructions. Make sure to update the passwordFormat attribute in the <credentials> tag.  **Users accounts stored elsewhere**  Storing user accounts in the web.config file has it's limitations. A lot of larger website will prefer to store the passwords in an SQL database or maybe a seperate XML/text file. So in this case, we won't have those optional <credential> and <user> tags. This tutorial does not concentrate on either of those technologies but we will show you an example of user accounts stored in an SQL database (so I'll assume you have some knowledge of reading data from a database.).  First, our example will assume the existence of a tabled called Users with the fields username and password.  Let's go back to the login.aspx page. We'll change the Login\_Click method to look like this:   |  | | --- | | void Login\_Click(Object sender, EventArgs e) {  String sHashedPassword = FormsAuthentication.HashPasswordForStoringInConfigFile(password.Text,"MD5");  String sqlStmt = "Select username from Users where username='" + UserName.Text + "' and password='" + sHashedPassword + "'";  SqlConnection sqlConn = new SqlConnection("server=localhost;uid=sa;pwd=password;database=master;");  SqlCommand sqlCmd = new SqlCommand(sqlStmt, sqlConn);  sqlCmd.Connection.Open();  SqlDataReader sqlReader = sqlCmd.ExecuteReader(CommandBehavior.CloseConnection);  if (sqlReader.Read())  FormsAuthentication.RedirectFromLoginPage(username.Text, true);  else  status.InnerHtml += "Invalid Login";  } |   In this function, we've replaced the FormsAuthentication.Authenticate function with some SQL code that will query the database for a username/password pair. If such a pair if found, then the credentials are correct and the user can be issued a cookie and redirected back to the webpage. Also, notice in this example, I've used the HashPasswordForStoringInConfigFile method (see the section above for encrypting password). This function is used because the database will have the passwords hashed. The code above could easily be changed to look in an XML file or a text file. Just remember the key methods that you will have to use is "FormsAuthentication.RedirectFromLoginPage".  **Tips**   * web.config file is cAsE-sEnSiTivE. Be careful how you type the tags and attributes. * Since the web.config file is an XML file, all opened tags must have a closing tag. Or a single tag must look like this: <user name="devhood" password="password"/> (notice the slash near the end). * Encrypt your passwords! Storing them as cleartext is asking for trouble. * Make sure you hash your passwords with the right hash algorithm and that you don't get them mixed up. * loginUrl has to be absolute. not relative * don't forget to make the password box of type password **Links** [Devhood ASP.NET Message Forum](http://www.devhood.com/messages/message_threads-2.aspx?category_id=3) - Feel free to ask questions to me or other Devhood members. [http://www.devhood.com](http://www.devhood.com/) - Can't forget to mention this great resource. :-) [http://www.asp.net](http://www.asp.net/) - Good portal to other ASP.NET resources. [http://www.brinkster.com](http://www.brinkster.com/) - Free ASP.NET hosting <http://hosting.msugs.ch/> - Free ASP.NET hosting [Guestbook.NET - a beginner tutorial to ASP.NET](http://www.devhood.com/Tutorials/tutorial_details.aspx?tutorial_id=96) [Authentication in ASP.NET: .NET Security Guidance](http://www.msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/authaspdotnet.asp?frame%3Dtrue) - very detailed MSDN info about ASP.NET authentication.  Copyright © 2001 Andrew Ma. | |