ARTICLE

Custom Login in ASP.NET MVC

Posted by Krishna Garad in Articles | ASP.NET MVC on January 19, 2012

In this article we will learn how to provide a custom login against our own database in ASP.NET MVC.

Download Files: CustomLogin.rar

Introduction

In this article we will learn how to provide a custom login against our own database in ASP.NET MVC. In every business application we need to provide login functionality in our application using a custom database where we are storing the user's details. In my last article you have seen how to start the ASP.NET MVC application. In this article we will see how you can provide the login functionality in ASP.NET MVC.

Step 1:

Start a new project as MVC3 Web Application from the New Project Dialog and select Empty Application. In the next screen we will create one empty ASP.NET MVC3 Web Application.

Step 2:

In this application we will see how to use a master page in ASP.Net MVC. So let's create one master page in Views/Shared folder in your application. Design your master page as you want. Here I'm designing it very simply just giving two links on the master page i.e. Home and About for navigation as well for displaying the user login status creating the UserControl. So Add one user control to Views/Shared folder and write the following code in that.

```
<%
   if (Request.IsAuthenticated) {
%>
       Welcome <strong><%: Page.User.Identity.Name %></strong>!
       [ <%: Html.ActionLink("Logout", "Logout", "Account") %> ]
<%
    }
   else {
%>
       [ <%: Html.ActionLink("LogIn", "Login", "Account") %> ]
<%
    }
%>
```

In the above lines of code we are just checking whether the user is logged in or not and with respect to their status we are displaying the link like if the user is not logged in the link for login page and if user is logged in then a greeting message with username and a logout link. Now our user control is ready but we need to load this control onto master page so write the following code in the master page.

```
<body>
 <header>
          <div id="title">
             <h1>Custom Login In ASP.Net MVC</h1>
             <div id="logindisplay">
             <% Html.RenderPartial("LoginDisplay"); %>
          </div>
          </div>

          <nav>
             <ul id="menu">
                <li><%: Html.ActionLink("Home", "Index", "Home")%></li>
                <li><%: Html.ActionLink("About", "About", "Home")%></li>
             </ul>
          </nav>

       </header>
    <div>
       <asp:ContentPlaceHolder ID="MainContent" runat="server">

       </asp:ContentPlaceHolder>
    </div>
</body>
```

In the above code you see that we have created two HTML Action links for Home and About as well as above that we are loading our user control using Html.RenderPartial.

Step 3:

Now we will create our Controllers. So go to the Controllers folder and add one controller named as HomeController. In Home controller we have to write two different methods for returning to the Home page and the About page so write the following two methods in HomeController.

```
       public ActionResult Index()
        {

            return View();
        }
        public ActionResult About()
        {
            return View();
        }
```

In the above code you can see we have two methods with index and About but still don't have any associated view with this methods so add two views; one for each and also select Master Page Layout from the Add View Dialog to our master page.

Step 4:

Still now we have our controller with home but now we have to provide the login functionality in again one different controller so add one more controller named as Account. Keep in mind the name of controllers and the methods we used earlier while showing some actionlink in master page and in user control so provide name such as used earlier. In the Login Controller we need two methods; one for checking login another for log out. But for login we need to provide one model, also where our actual logic to check user name and password in database. So add one class named as LoginMadel in Models folder and write the code given below.

```
[Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
        public bool IsValid(string _username, string _pwd)
        {
            string _sql = "Select UserName From TblLogin Where UserName='" + _username + "'
And Password='" + _pwd + "'";
            SqlConnection cn = new SqlConnection("Your connection String Here");
            cn.Open();
            SqlCommand cmd = new SqlCommand(_sql, cn);
            SqlDataReader dr = cmd.ExecuteReader();
            if (dr.Read())
                return true;
            else
                return false;

        }
```

The above code contains some properties for username, password and a remember me option as well it contains one method called IsValid where we are checking our login details in the database.

Still now we only have a model where we have written the login logic to check the user against the database. Now we can use this model in our Account Controllers methods.

Step 5:

In the Account controller we need to create two methods as we said earlier. So create two methods, one for calling model IsValid method and providing authentication and returning the view and the second method for logging the user out like bellow.

```
[HttpGet]
        public ActionResult Login()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Login(CustomLogin.Models.LoginModel model)
        {
            if (ModelState.IsValid)
            {
                if (model.IsValid(model.UserName, model.Password))
                {
                    FormsAuthentication.SetAuthCookie(model.UserName, model.RememberMe);

                    return RedirectToAction("Index", "Home");

                }
                else
                {
                    ModelState.AddModelError("", "The user name or password provided is incorrect.");
                }
            }


            return View(model);
        }
        public ActionResult Logout()
        {
            FormsAuthentication.SignOut();
            return RedirectToAction("Index", "Home");
        }
```

In the above code you see we wrote three methods but previously I said write two methods only but I'm showing three methods because here when the user makes the first request to login then simply we have to return the view and remember both methods with different parameters return the same view. While the first parameter-less login method is used with attribute [HttpGet] to return only view and second method with parameter to check the login and then return the action to index if valid else returns the same view with message.

In the second login method it takes a LoginModel as a parameter where it will check the username and password in our database. In the last method we are just providing signout and returning to action Index of Home controller.

Step 6:

Still now we have controller and methods but don't have any associated view. So add one view by clicking on the first login method and provide the textboxes and one button for taking input from the user.

```
<div>
     <% using (Html.BeginForm()) { %>
       <%: Html.ValidationSummary(true, "Login was unsuccessful. Please correct the errors and
try again.") %>
       <div>
          <fieldset>
              <legend>Account Information</legend>

              <div class="editor-label">
                  <%: Html.LabelFor(m => m.UserName) %>
              </div>
              <div class="editor-field">
                  <%: Html.TextBoxFor(m => m.UserName) %>
                  <%: Html.ValidationMessageFor(m => m.UserName) %>
              </div>

              <div class="editor-label">
                  <%: Html.LabelFor(m => m.Password) %>
              </div>
              <div class="editor-field">
                  <%: Html.PasswordFor(m => m.Password) %>
                  <%: Html.ValidationMessageFor(m => m.Password) %>
              </div>

              <div class="editor-label">
                  <%: Html.CheckBoxFor(m => m.RememberMe) %>
                  <%: Html.LabelFor(m => m.RememberMe) %>

              </div>

              <p>
                  <input type="submit" value="Log In" />
              </p>
          </fieldset>
       </div>
    <% } %>
    </div>
```

In the above code we are designing our login page with input fields and with validation which we wrote in our model i.e. in LoginModel. In the above code you were not able to access the member of LoginModel until and up to where you are inheriting the page from that model like bellow.

```
Inherits="System.Web.Mvc.ViewPage<CustomLogin.Models.LoginModel>
```

Now our login page is also ready. Now you can run the application and see the results. I think you might still be in confusion; we have done the login but how we are going to be logged out? So in the User Control you can see we have kept one action link for logout and calling the Logout method of Account Controller where we are going to logout.

One more thing about the login is to change the configuration so go to root web.config file and change the login url in of forms tag to our login page. Still we don't have any secure directory. It is as usual like in ASP.Net you are using when you are adding any new secure directory then just add one web.config file in that and add the authorization tag to deny unauthenticated user.

Conclusion

In this way we can provide our own login functionality to our application in ASP.NET MVC.

COMMENT USING

Add a comment...

☑ Also post on Facebook Posting as Asaduzzaman Arif (Change) Comment

Singh Dilip · Works at Unisys Infosolutions Pvt Ltd
how will I do login without validating with database ?
Reply · Like · Follow Post · January 22 at 1:04pm

Satheesh Madupathi · Follow · Developer at Strategic Mobitech Solutions
good....
Reply · Like · Follow Post · January 24 at 2:23pm

Arif Pahat · Software Developer at Currently Working in Noida
thanks mem
Reply · Like · Follow Post · February 16, 2013 at 9:33pm

Facebook social plugin