Articles » Web Development » ASP.NET » General

ASP.NET MVC 4 Forms Authentication Customized

By **sanjivksingh**, 4 Jun 2013

4.45 (5 votes)

Introduction

I am doing a POC in ASP.Net MVC 4 project in which i had to customize User Authentication requirement so that it can validate an active directory user which should also exist in my project database. I have googled a lot and looked into many articles but have not found any straight forward solution. So thought to take it up and share with you all.

Initially i thought i will go with Windows Authentication and in `Application_AuthenticateRequest` event of *global.asax* I will write my custom code to validate the user in my project database. But the main issue is, if the user validation is failed in the database it is difficult (I didn't find a way) to reset the authentication cookie to mark the user as unauthenticated user. In simple words the user is always authenticated with Windows Authentication. You can see `Request.IsAuthenticated` flag is always `TRUE`.

So I decided to go with Forms Authentication in which I will bypass the login page and will read the windows logged in user from `Request.ServerVariables["LOGON_USER"]` and will validate the user in database.

Background

This article is helpful for those who is looking for single signon in MVC 4 application as well as they want the user should also be verified in users table in your project database. Please note that this article will be based on MVC 4 and i am using VS2010 SP1 to create sample code.

Understanding Forms Authentication in MVC

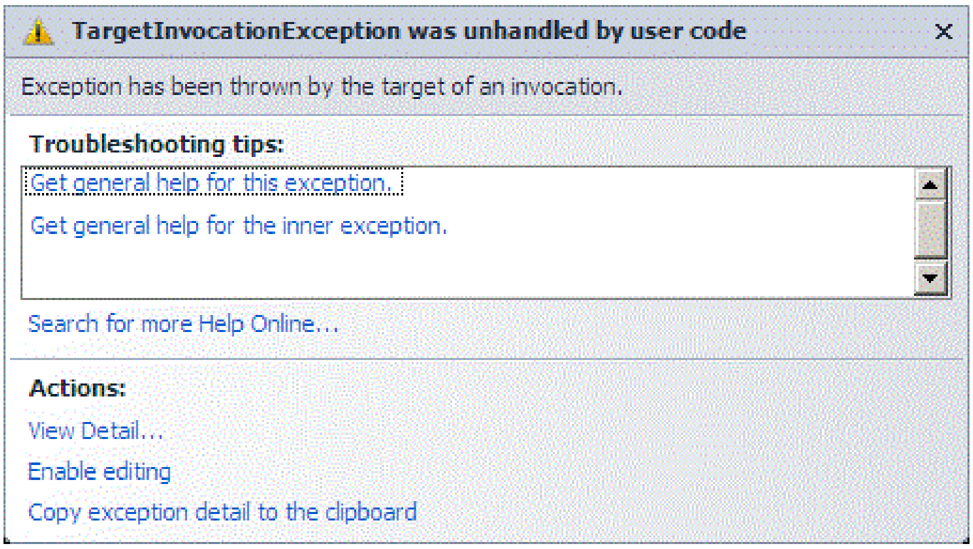
You can find tons of article on Forms Authentication. So i am not going to write the repeated boring stuff but will demystify the abstraction you see in your project when you select "*Internet Application*" template while creating your MVC project. Lets get into step by step.

1. File -> New -> Project -> ASP.NET MVC 4 Application Click on "OK" button
2. Project Template -> Internet Application
3. View Engine -> Razor, now click on "OK" button to create the project

Now the project is created. If you look at the web.config file you can find the following Authentication tag which claims that the project will use "Forms" authentication.

```
<authentication mode="Forms" /><forms loginUrl="~/Account/Login" timeout="2880" />
</authentication>
```

Now if you try to compile the project, you will be able to compile it and you shall be able to run the project successfully. The first issue comes when you click on "Login" button. You can see the following error in the "*InitialiseSimpleMembershipAttribute.cs*" file.


TargetInvocationException was unhandled by user code

Exception has been thrown by the target of an invocation.

Troubleshooting tips:

Get general help for this exception.

Get general help for the inner exception.

Search for more Help Online...

Actions:

View Detail...

Enable editing

Copy exception detail to the clipboard

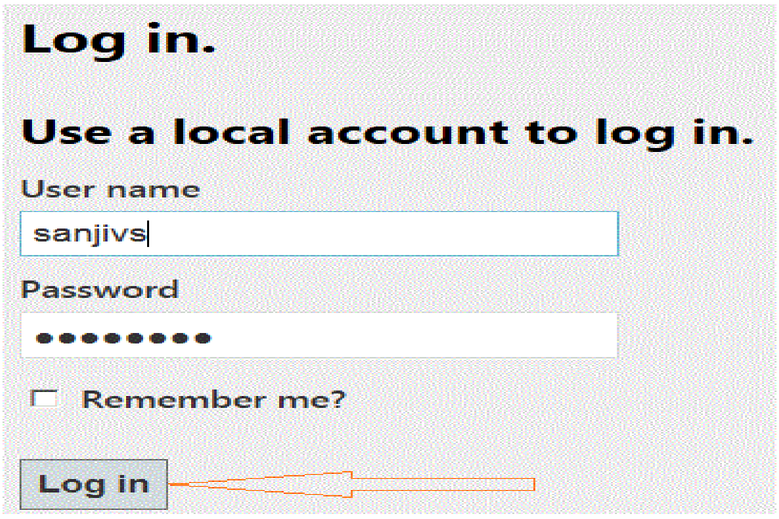
The above error occures because the code is not able to initialise/create membership table in the SQL express. you can find the following *connectionstring* in the *web.config* file.

```
<add name="DefaultConnection" connectionstring="Data Source=.\SQLEXPRESS;
Initial Catalog=aspnet-MvcApplication1-20130603132719;Integrated Security=SSPI"
 providername="System.Data.SqlClient" />
```

I suggest you to change default connection string to point to your SQL server project database as as shown below:

```
<add name="DefaultConnection"
  connectionString=quot;server=SQl Server machine name Integrated Security=false;
  User ID=sa; Password=password; database= your project database;
  providerName="System.Data.SqlClient" /&gt;</p>
```

Now you again run the application and click on the "Login" button, you will be able to see the Login Page as follows:



Well, now if you examine the database you will be able to see 5 new tables have been created automatically by the membership provider. Those are as follows:

1. dbo.UserProfile (this table contains the users name when you try to register a user from application)
2. dbo.webpages_ Membership (this table maintains the password of users created during registering a user from application)
3. dbo.webpages_OAuthMembership (this table contains user id if you using OAuth. Out of scope for this article)
4. dbo.webpages_Roles (You can create the roles in this table which can be used in `[Authorize]` attribute during authorization in the application)
5. dbo.webpages_UsersInRoles (this table contains the user and their role relationship)

Now let's talk about what happens when you click on the "Log in" button. First. Here is the code written in *Account* Controller which gets fire when you click on "Log in" button.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Login(LoginModel model, string returnUrl)
{
if (ModelState.IsValid && WebSecurity.Login(model.UserName,
          model.Password, persistCookie: model.RememberMe))
{
return RedirectToLocal(returnUrl);
}

// If we got this far, something failed, redisplay form
ModelState.AddModelError("", "The user name or password provided is incorrect.");
return View(model);
}
```

The fairly simple code. Magic happens with `WebSecurity.Login()` method. this method first checks if supplied user exists in *dbo.userprofile* table. If exists, then this method checks if supplied password matches in the *dbo.webpages_ Membership table*. If password also matches then this method reads the roles associated with this user and set in the request context and returns true to indicate successful login. It internally also sets the `Request.IsAuthenticated` flag to true. That's all about forms authentication side in MVC 4 project.

Resolution of authentication customization problem

Now since you understand the logic behind forms authentication lets talk about resolution of single sign on problem with forms authentication in which user will also be validated in project database.

Step -1

Change the <Forms> tag in *Web.Confile* file with following tag.

```
<forms loginUrl="errors/InvalidUser" timeout="2880" />
```

Step -2

Add errors Controller with following Action methods.

```
public class ErrorsController : Controller
{
    //
    // GET: /Errors/

    public ActionResult InvalidUser()
    {
        return PartialView("_InvalidUser");
    }
    public ActionResult UnAuthorizedUser()
    {
        return PartialView("_UNAuthorizedUser");
    }

}
```

Step -3

Add a shared view "*_InvalidUser.cshtml*" with following Razor.

```
@{
    ViewBag.Title = "Invalid User";
 }

<hgroup class="title">
    <h1 class="error">Invalid User.</h1>
    <h2 class="error">Please contact your Administrator.</h2>
</hgroup;>
```

Add another shared view "*_UNAuthorizedUser.cshtml*" with following Razor.

```
@{
    ViewBag.Title = "Invalid User";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<hgroup class="title">
    <h1 class="error">You are not authorized to view this page.</h1>
    <h2 class="error">Please contact your Administrator.</h2>
</hgroup>
```

Step -4

change the existing *_LoginPartial.cshtml* partial view with following Razor.

```
@if (Request.IsAuthenticated)
{
    <text>
```

```
        Hello, @User.Identity.Name

    </text>
}
```

Step -5

Lets create our own User table in the Application Database.

```
CREATE TABLE [dbo].[AppUser](  [USerID] [int] IDENTITY(1,1) NOT NULL,

[LoginName1] [nvarchar]
 (50) NULL, )
```

Step -6

Create user data in user table manually.

```
Insert Into dbo.AppUser values(1,'<your windows user name>');
```

Step 7

Create data in Membership table.

```
insert into dbo.webpages_Membership(UserId,PasswordFailuresSinceLastSuccess,Password)
values
(1,0,'AJfhqOHKFeLY8aHVGCAwf0dnN6QkGPv09Hj5sQaG2FQsdIk9p7zniTJmb6tMQK/HIQ==')
```

Please note : the Encrypted value shown in the above query for password is "1", since `WebSecurity.Login` can not receive a blank password so I am hard-coding this password with "1" and I am planning to create every user with the same encrypted password. Since we are going with single signon password doesn't make much sense here.

Step -8

Create role as per your need in role table.

Step -9

Go to the *InitializeSimpleMembershipAttribute.cs* file and change the following code in which "*User*" is newly created table.

```
WebSecurity.InitializeDatabaseConnection("DefaultConnection",
   "User", "UserId", "UserName", autoCreateTables: true);
```

Step 10

Add the following Attribute to the Home Controller.

```
[Authorize]
[InitializeSimpleMembership]
```

Step -11

Change the Index Action of Home Controller with following code.

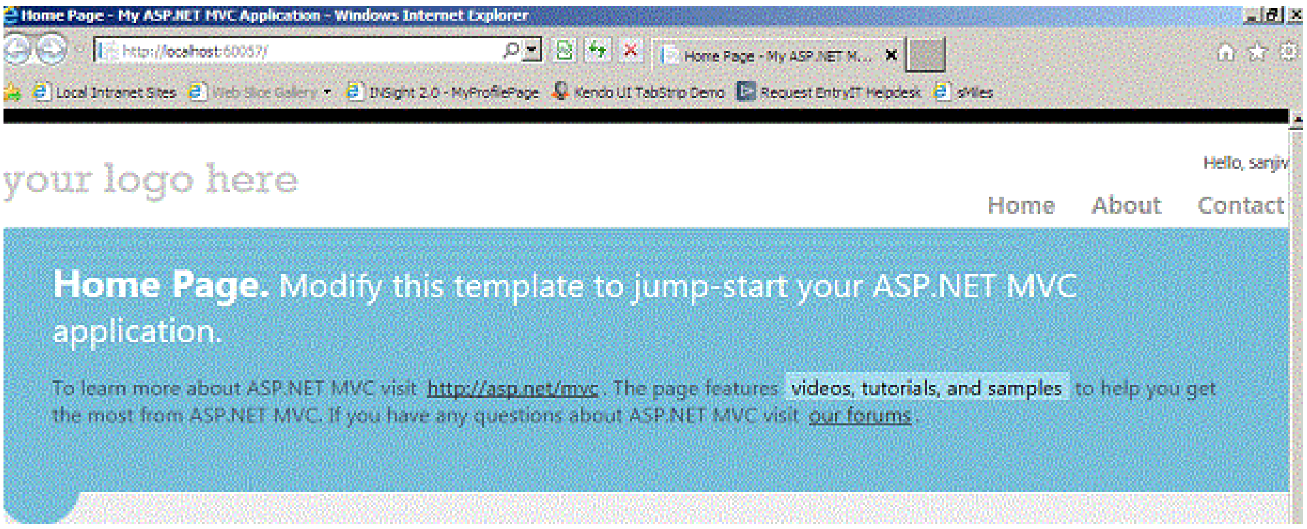
```
[AllowAnonymous]
public ActionResult Index()
{
    string username;
    var logonUser = Request.ServerVariables["LOGON_USER"];
    username = logonUser.Split('\\')[1].ToString();
    if (!Request.IsAuthenticated)
    {
        username = "sanjiv";
        if (WebSecurity.Login(username, "2", persistCookie: false))
        {
            return RedirectToAction("Index", "Home");
        }
        else
        {
            return RedirectToAction("InvalidUser", "Errors");
        }
    }
    ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC application.";
    return View();
}
```

Step - 12

Remove account controller (*AccountController.cs*) completely from the project.

Step -13

Now you run the application and you will be able to see the following screen with logged in user information.



Home Page. Modify this template to jump-start your ASP.NET MVC application.

To learn more about ASP.NET MVC visit http://asp.net/mvc . The page features videos, tutorials, and samples to help you get the most from ASP.NET MVC. If you have any questions about ASP.NET MVC visit our forums .

Summary

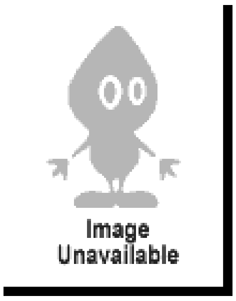
I will recap the activities performed in this article.

1. We had removed the Account Controller completely from the project and managed the authentication from "Index" action of "Home" controller. That's the reason we had to provide `[AllowAnonymous]` attribute to "Index" action. In this authentication process we are reading the logged in user name from Server variables and not providing the user to input username hence it behaves like single sing on application.
2. Secondly we have changed the "`InitializedSimpleMembership`" provider to connect to our Local DB. We also managed the user data in our own AppUSer table in project database.

3. Third and importantly we had changed the `_LoginPartial` partial view and also created our own error view to display "Invalid user ." error message.

License

About the Author



sanjivksingh

United States

No Biography provided

Comments and Discussions

6 messages have been posted for this article Visit **http://www.codeproject.com/Articles/601687/ASP-NET-MVC-4-Forms-Authentication-Customized** to post and view comments on this article, or click **here** to get a print view with messages.