Gora LEYE ~ Expert C#.NET

How to configure Custom Membership and Role Provider using ASP.NET MVC4

***29** Jeudi août 2013*

PUBLIÉ PAR GORALEYE IN ARCHITECTURE, ASP.NET, C#.NET, MEMBERSHIP, ROLE PROVIDER, SECURITY

≈ 3 COMMENTAIRES

Tags

Application Services, Architecture, ASP.NET, C#.NET, Custom Membership Provider, Role provider, Security

ASP.NET membership is designed to enable you to easily use a number of different membership providers for your ASP.NET applications. You can use the supplied membership providers that are included with the .NET Framework, or you can implement your own providers.

There are two primary reasons for creating a custom membership provider.

- You need to store membership information in a data source that is not supported by the membership providers included with the .NET Framework, such as a MysQL database, an Oracle database, or other data sources.
- You need to manage membership information using a database schema that is different from the database schema used by the providers that ship with the .NET Framework. A common example of this would be membership data that already exists in a SQL Server database for a company or Web site.

In tis tutorial, we are going to implement and configure a custom Membership Provider using ASP.NET MVC4

Let's go

A. Create a Custom MemberShip Application class Library

1. Create a class Library Project (our sample Projet name is **LogCorner.SoftwareStore.Security**)
2. Reference the assembly **System.Web.ApplicationServices** (Right Click Reference è Add reference => Select Assemblies => navigate to System.Web.ApplicationServices and add it)

3. Create a Class **CustomMembershipProvider** and derive it from **MembershipProvider**
4. Override ValidateUser as follow

```
using System;
using System.Collections.Generic;
using System.Web.Security;

namespace LogCorner.SoftwareStore.Security.Infrastructure
{
    public class User
    {
        public string Username { get; set; }
        public string Password { get; set; }
    }

    public class CustomMembershipProvider : MembershipProvider
    {
        #region Private Fields
       // For simplicity, just working with a static in-memory collection
        // In any real app you'd need to fetch credentials from a database

        private static readonly List<User> Users = new List<User> {
            new User { Username = "Yves", Password = "123456" },
            new User { Username = "Jean", Password = "123456" },
            new User { Username = "Georges", Password = "123456" }
        };

        public override string ApplicationName
        {
            get
            {
                throw new NotImplementedException();
            }
            set
            {
                throw new NotImplementedException();
            }
        }

        public override bool ValidateUser(string username, string password)
        {
            return Users.Exists(m => m.Username == username && m.Password == password);
        }

        public override bool ChangePassword(string username, string oldPassword, string newPassword)
        {
            throw new NotImplementedException();
        }

        public override bool ChangePasswordQuestionAndAnswer(string username, string password, string newPasswor
        {
            throw new NotImplementedException();
        }
```

For now we have what we need for our application security. To go further in the implementation of Custom Membership Provider, please see our tutorial *Mastering Custum ASP.NET MemberShip Provider using ASP.NET MVC*

B. Create an ASP.NET MVC4 application Client

1. Create an ASP.NET MVC4 application Client (Add New projet è ASP.NET MVC4 Web Application è Select Template Internet Web Appliction and Click OK)
2. Open Web.config file
3. Add or Replace membership section as follow

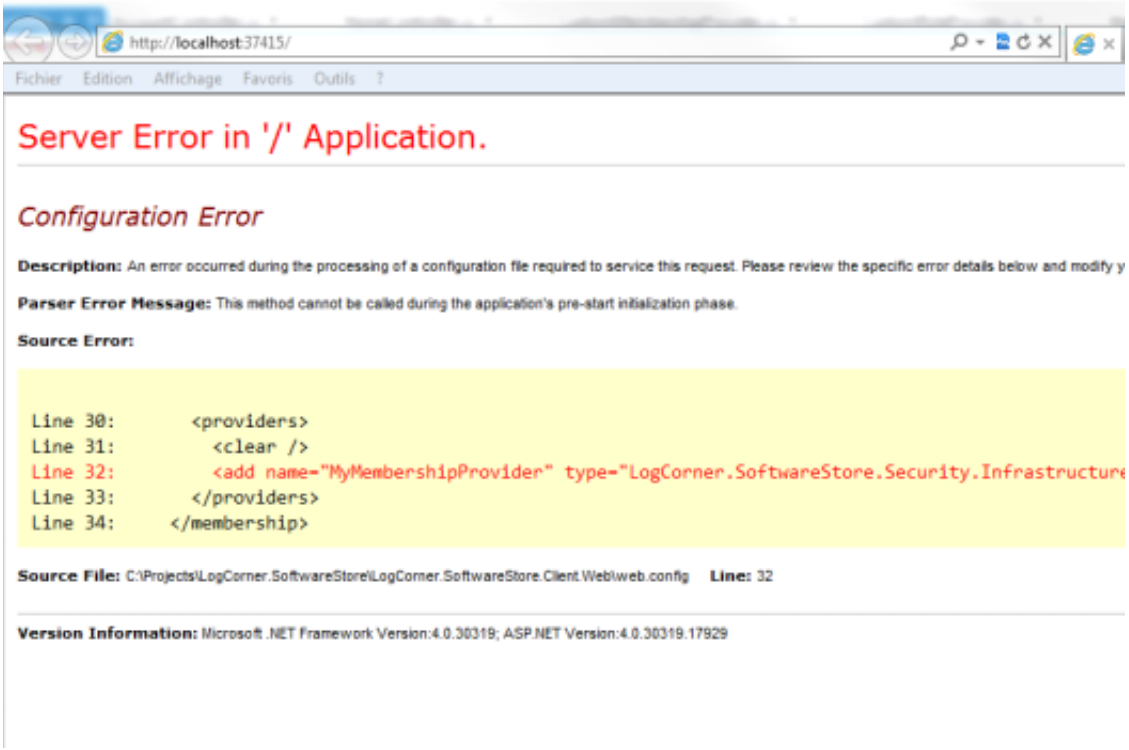
```
<authentication mode="Forms">
  <forms loginUrl="~/Account/Login" timeout="2880" />
</authentication>


<membership defaultProvider="MyMembershipProvider">
  <providers>
    <clear />
    <add name="MyMembershipProvider" type="LogCorner.SoftwareStore.Security.Infrastructure.CustomMembershipProvider"
         connectionStringName="ApplicationServices"
         enablePasswordRetrieval="false"
         enablePasswordReset="true"
         requiresQuestionAndAnswer="false"
         requiresUniqueEmail="false"
         maxInvalidPasswordAttempts="5"
         minRequiredPasswordLength="6"
         minRequiredNonalphanumericCharacters="0"
         passwordAttemptWindow="10"
         applicationName="LogCorner.SoftwareStore.Client.Web" />
  </providers>
</membership>
```

4. Open **HomeController** and **Authorize** Attribute to Index **ActionResult**

```
namespace LogCorner.SoftwareStore.Client.Web.Controllers
{
    public class HomeController : Controller
    {
        [Authorize]
        public ActionResult Index()
        {
```

5. Run the application ASP.NET MVC4 application Client, you ll have the errors below



6. do not panic, proceed as follows :

Add this in your web.config (in the appSettings section):

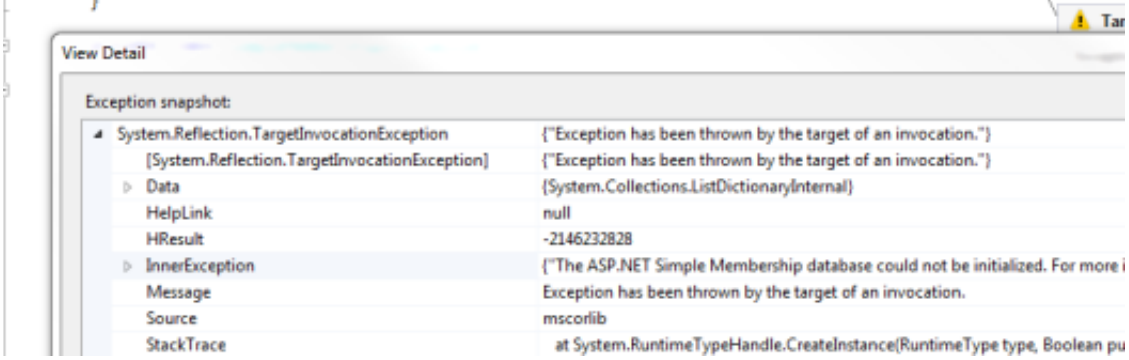
<add key="enableSimpleMembership" value="false"/>

<add key="autoFormsAuthentication" value="false"/>

7. Run the application ASP.NET MVC4 application Client, you ll have another error

```
namespace LogCorner.SoftwareStore.Client.Web.Filters
{
    [AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = false, Inherited = true)]
    public sealed class InitializeSimpleMembershipAttribute : ActionFilterAttribute
    {
        private static SimpleMembershipInitializer _initializer;
        private static object _initializerLock = new object();
        private static bool _isInitialized;

        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            // Ensure ASP.NET Simple Membership is initialized only once per app start
            LazyInitializer.EnsureInitialized(ref _initializer, ref _isInitialized, ref _initializerLock);
        }
```



8. To fix it Open **AccountController** and comment **InitializeSimpleMembership** , because we using Custom Membership Provider instead of Simple Membership

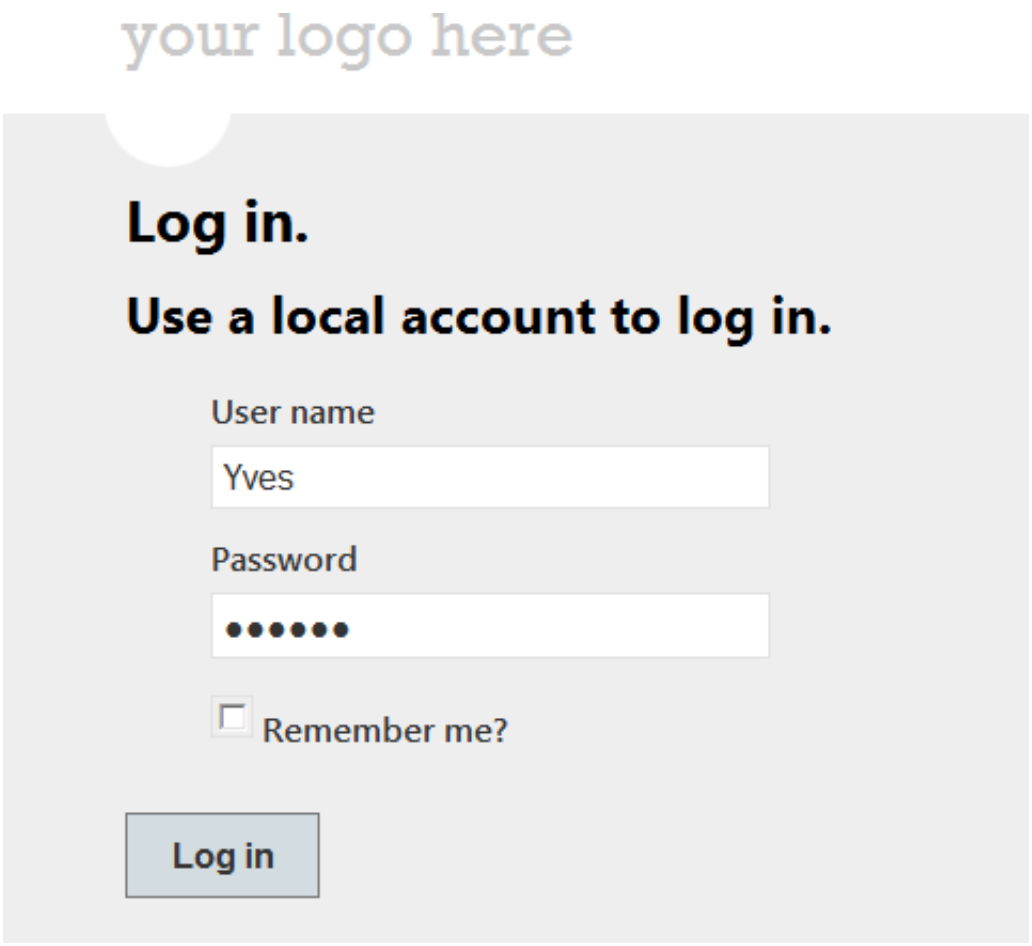
9. Override **Login** Action of **AccountController** as follow :

```
//
// POST: /Account/Login

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Login(LoginModel model, string returnUrl)
{
    if (ModelState.IsValid && Membership.ValidateUser(model.UserName, model.Password))
    {
        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);
}
```

10. Run the application ASP.NET MVC4 application Client, you'll have the form authentication below



11. Enter user credentials and click Log In, then you will have the execution workflow below :

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Login(LoginModel model, string returnUrl)
{
    if (ModelState.IsValid && Membership.ValidateUser(model.UserName, model.Password))
    {
        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);
}


    }

    public override bool ValidateUser(string username, string password)
    {
        return Users.Exists(m => m.Username == username && m.Password == password);
    }
```

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Login(LoginModel model, string returnUrl)
{
    if (ModelState.IsValid && Membership.ValidateUser(model.UserName, model.Password))
    {
        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);
}
```

C. Configuration of Custom Role Provider

To configure custom role provider, please proceed as follow :

1. create a class **CustomRoleProvider** that inherits from **RoleProvider**
2. Overrides **GetRolesForUser** method

```
namespace LogCorner.SoftwareStore.Security.Infrastructure
{
    public class CustomRoleProvider : RoleProvider
    {
        public override string[] GetRolesForUser(string username)
        {
            switch (username)
            {
                case "Yves":
                    {
                        return new[] { "Manager", "Administrator" };
                    }
                case "Jean":
                    {
                        return new[] { "Operator", "Manager" };
                    }
                case "Georges":
                    {
                        return new[] { "Customer" };
                    }
                    default:
                    return new string[] { };
            }

        }

        public override void AddUsersToRoles(string[] usernames, string[] roleNames)
        {
            throw new NotImplementedException();
        }

        public override string ApplicationName
        {
            get
            {
                throw new NotImplementedException();
            }
```

3. Now open web.config file of your client asp.net web application and add a **RoleManager**

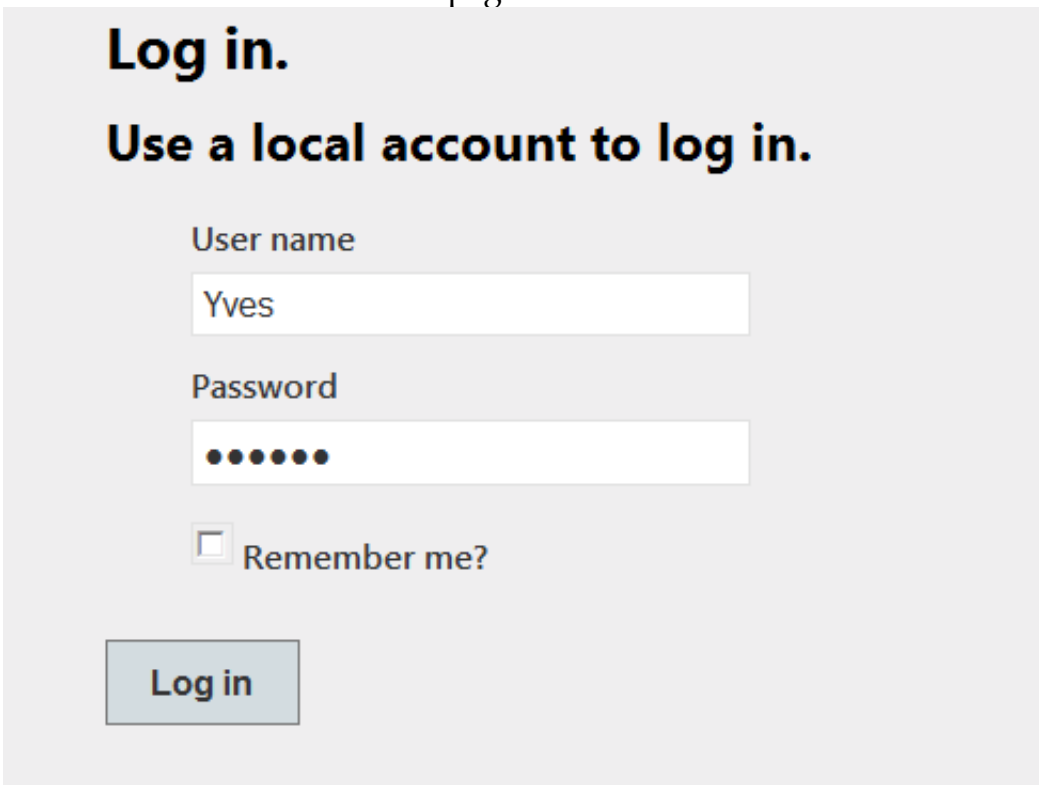
```
<roleManager enabled="true" defaultProvider="MyRoleProvider">
  <providers>
    <clear/>
    <add name="MyRoleProvider"  type="LogCorner.SoftwareStore.Security.Infrastructure.CustomRoleProvider"/>
  </providers>
</roleManager>
```

section

4. Open HomeController and change Authorization as follow :

```
public class HomeController : Controller
{
    [Authorize(Roles = "Administrator")]
    public ActionResult Index()
    {
```

5. Now test your sample. Only users who have approved login credentials and who belong to role Administrator can view Index page




```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Login(LoginModel model, string returnUrl)
{
    if (ModelState.IsValid)
    {
        if (Membership.ValidateUser(model.UserName, model.Password))
        {
            FormsAuthentication.SetAuthCookie(model.UserName, false);
            return Redirect(returnUrl ?? Url.Action("Index", "Admin"));
        }
        else
        {
            ModelState.AddModelError("", "Incorrect username or password");
            return View();
        }
    }
    else
    {
        return View();
    }
}
```

```
namespace LogCorner.SoftwareStore.Security.Infrastructure
{
    public class CustomRoleProvider : RoleProvider
    {
        public override string[] GetRolesForUser(string username)
        {
            switch (username)
            {
                case "Yves":
                    {
                        return new[] { "Manager", "Administrator" };
                    }
                case "Jean":
                    {
                        return new[] { "Operator", "Manager" };
                    }
                case "Georges":
                    {
                        return new[] { "Customer" };
                    }
                    default:
                    return new string[] { };
            }

        }
```

```
public class HomeController : Controller
{
    [Authorize(Roles = "Administrator")]
    public ActionResult Index()
    {
        ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC application.";

        return View();
    }
```

Thank you for reading us, our next tutorial is to configure Custom Membership Provider using ASP.NET MVC4 with external login like facebook, yahoo , google or other relying party accounts.

If you seek information about encoding and decoding password, please read our article ASP.NET Custom Membership Password Encoding and Decoding based on key SALT using SHA-3 algorithm

réflexions à propos de “ How to configure Custom Membership and Role Provider using ASP.NET MVC4 ”

1. Ping: ASP.NET MVC PayPal Integration | Gora LEYE
2. Ping: ASP.NET MVC Tracking User Activity | Gora LEYE
3. *a dit*:Jan

   octobre 15, 2013 à 12:35

   Nice walkthrough of the extensibility of ASP.NET. Thanks!

   RÉPONSE

3