# Gora LEYE ~ Expert C#.NET

## Archives de Tag: *Application Services*

## Creating the Application Services Database for SQL Server

***21** Samedi sept 2013*

PUBLIÉ PAR GORALEYE IN DATABASE, MEMBERSHIP, SECURITY, SQL SERVER

≈ UN COMMENTAIRE

***Tags***
*Application Services, aspnet_regsql, database, MemberShip, Security, sql server*

ASP.NET includes a tool for installing the SQL Server database used by the SQL Server providers, named Aspnet_regsql.exe. The Aspnet_regsql.exe tool is located in the drive:\WINDOWS\Microsoft.NET\Framework\versionNumber folder on your Web server. Aspnet_regsql.exe is used to both create the SQL Server database and add or remove options from an existing database.

You can run Aspnet_regsql.exe without any command line arguments to run a wizard that will walk you through specifying connection information for the computer running SQL Server and installing or removing the database elements for all the supported features. You can also run Aspnet_regsql.exe as a command-line tool to specify database elements for individual features to add or remove.

| Note |
|---|
| The database elements that are installed in the feature database will always be owned by the SQL Server database owner account (dbo). In order to install the feature database, a SQL Server login must be permitted to the db_ddladmin and dd_securityadmin roles for the SQL Server database. However, you do not need to be a system administrator for the SQL Server in order to install the feature database. |

To run the Aspnet_regsql.exe wizard, run Aspnet_regsql.exe without any command line arguments, as shown in the following example:

C:\WINDOWS\Microsoft.NET\Framework\<versionNumber>\aspnet_regsql.exe

You can also run the Aspnet_regsql.exe tool as a command-line utility. For example, the following command installs the database elements for membership and role management on the local computer running SQL Server:

aspnet_regsql.exe -E -S localhost -A mr

The following table describes the command line options supported by the Aspnet_regsql.exe tool.

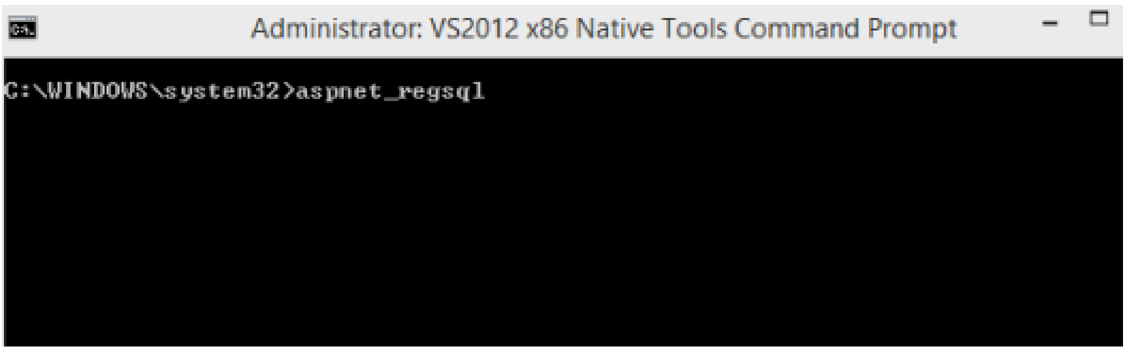
| **Option** | **Description** |
|---|---|
| -? | Prints Aspnet_regsql.exe tool Help text in the command window. |
| -W | Runs the tool in wizard mode. This is the default if no command line arguments are specified. |
| -C connection string | The connection string to the computer running SQL Server where the database will be installed, or is already installed. This option is not necessary if you only specify the server (-S) and login (-U and -P, or -E) information. |
| -S server | The name of the computer running SQL Server where the database will be installed, or is already installed. The server name can also include an instance name, such as .\INSTANCENAME. |
| -U login id | The SQL Server user id to log in with. This option also requires the password (-P) option. This option is not necessary if you are authenticating using Windows credentials (-E). |
| -P password | The SQL Server password to log in with. This option also requires the login id (-U) option. This option is not necessary if authenticating using Windows credentials (-E). |
| -E | Authenticates using the Windows credentials of the currently logged-in user. |
| -d database | The name of the database to create or modify. If the database is not specified, the default database name of "aspnetdb" is used. |
| -sqlexportonlyfilename | Generates a SQL script file that can be used to add or remove the specified features. The specified actions are not performed. |
| -A all\|m\|r\|p\|c\|w | Adds support for one or more features. The following identifiers are used for ASP.NET features.<br><br>IdentifierAffects<br>allAll features<br>mMembership<br>rRole management<br>pProfile<br>cWeb Parts personalization<br>wWeb events<br>Feature identifiers can be specified together or separately, as shown in the following examples.<br><br>aspnet_regsql.exe -E -S localhost -A mp |

| | |
|---|---|
| | aspnet_regsql.exe -E -S localhost -A m -A p |
| -R all\|m\|r\|p\|c\|w | Removes support for one or more features. The following identifiers are used for ASP.NET features.<br><br>IdentifierAffects<br>allAll features<br>mMembership<br>rRole management<br>pProfile<br>cWeb Parts personalization<br>wWeb events<br>Feature identifiers can be specified together or separately, as shown in the following examples.<br><br>aspnet_regsql.exe -E -S localhost -R mp<br><br>aspnet_regsql.exe -E -S localhost -R m -R p |
| -Q | Runs the tool in quiet mode and does not confirm before removing a feature. |

In this section , we can create our security database according to our business model and store it on sql server, oracle, mysql or other.

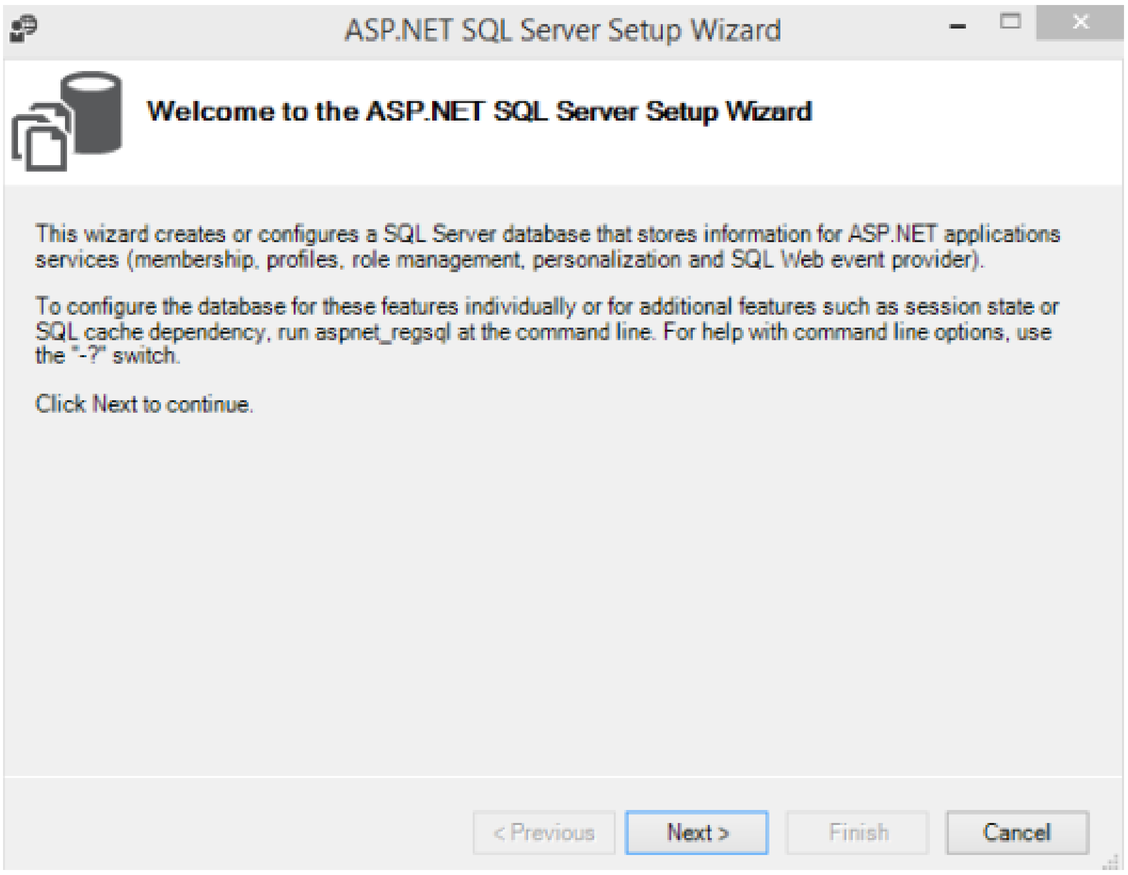
*If you have already a security database, go to the next section.*

Open visual studio prompt command tool and run command line **aspnet_regsql** as following



```
C:\WINDOWS\system32>aspnet_regsql
```

The screen explain the wizard scenario  so Click next


ASP.NET SQL Server Setup Wizard

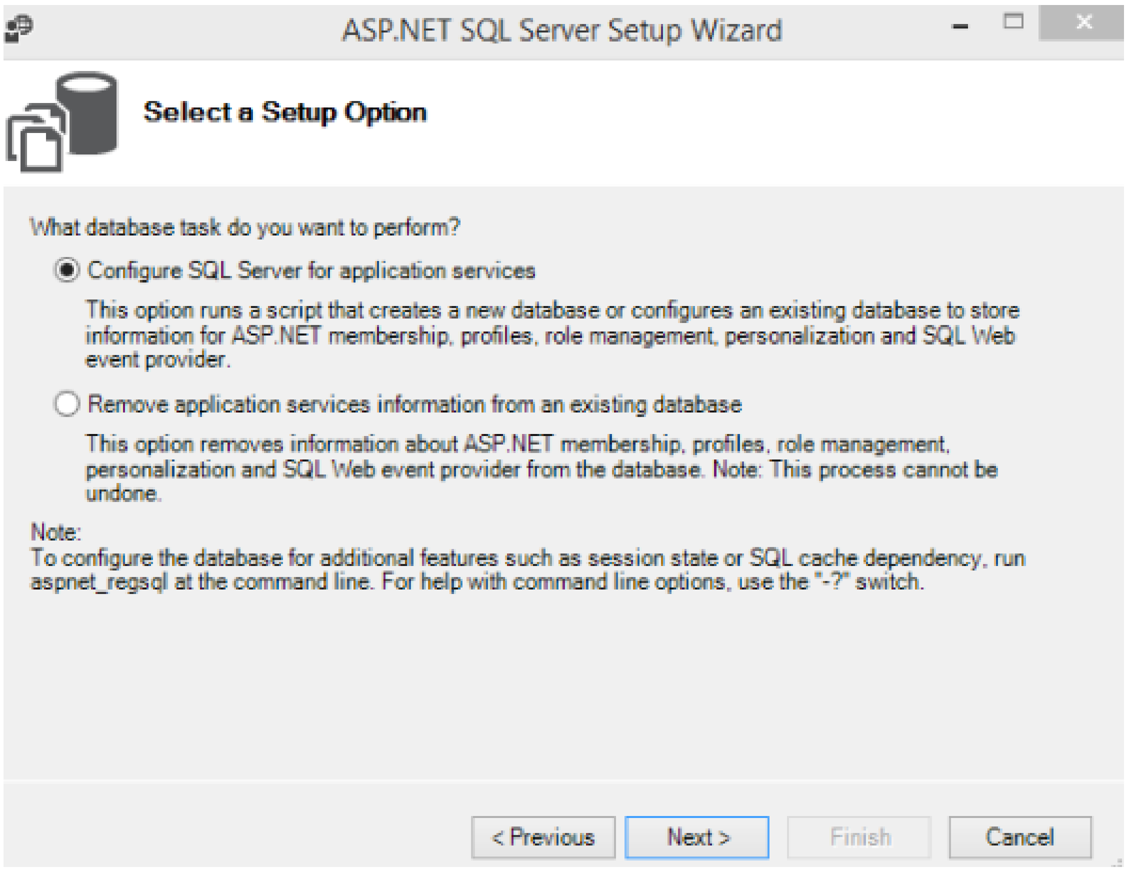
**Welcome to the ASP.NET SQL Server Setup Wizard**

This wizard creates or configures a SQL Server database that stores information for ASP.NET applications services (membership, profiles, role management, personalization and SQL Web event provider).

To configure the database for these features individually or for additional features such as session state or SQL cache dependency, run aspnet_regsql at the command line. For help with command line options, use the "-?" switch.

Click Next to continue.

Here we can remove existing security database or create a new one. We want to create a new security database. So check the first option and click next


ASP.NET SQL Server Setup Wizard

**Select a Setup Option**

What database task do you want to perform?

(●) Configure SQL Server for application services

This option runs a script that creates a new database or configures an existing database to store information for ASP.NET membership, profiles, role management, personalization and SQL Web event provider.

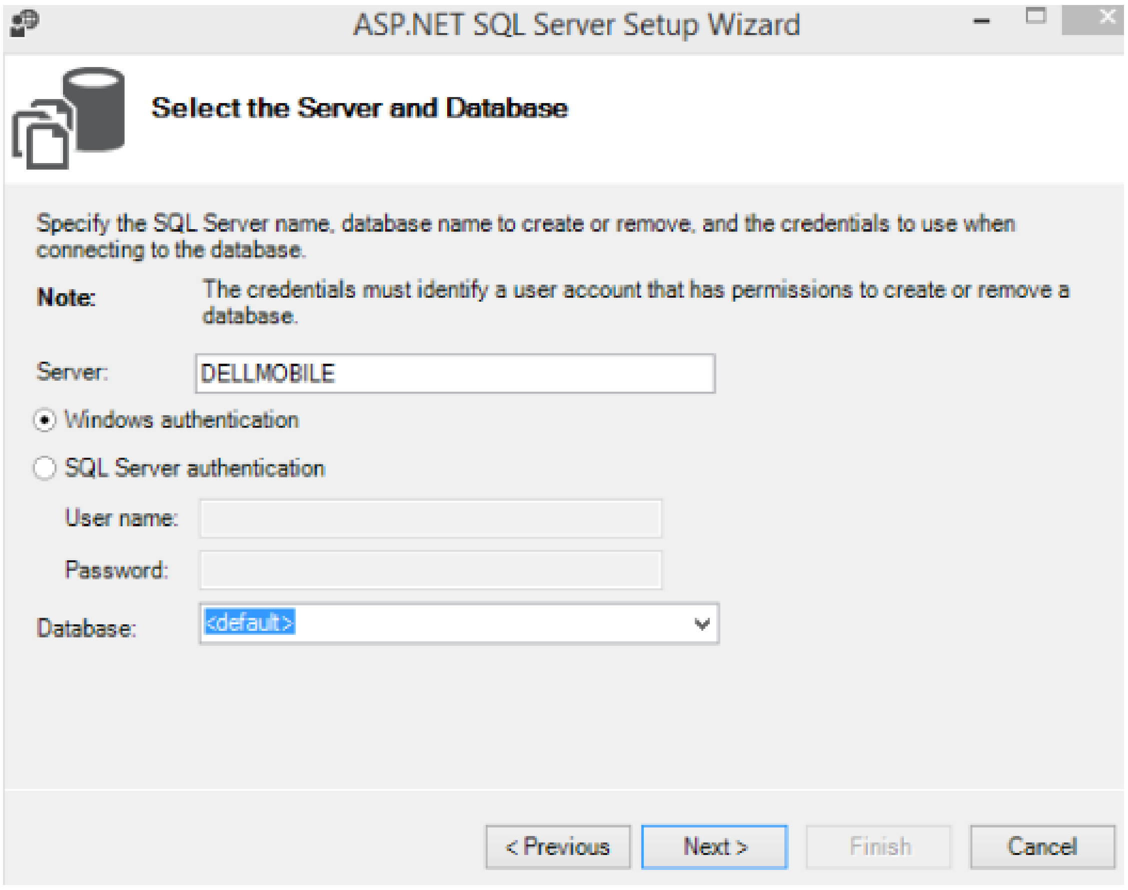
( ) Remove application services information from an existing database

This option removes information about ASP.NET membership, profiles, role management, personalization and SQL Web event provider from the database. Note: This process cannot be undone.

Note:
To configure the database for additional features such as session state or SQL cache dependency, run aspnet_regsql at the command line. For help with command line options, use the "-?" switch.

Enter our database server name .\SQLEXPRESS ( enter the appropriate server name). if you already an existing database, you can select it. So the wizard will create the security tables on the selected datase.

If you do not have a database, let default. the default database name that will be created is aspnetdb


ASP.NET SQL Server Setup Wizard

**Select the Server and Database**

Specify the SQL Server name, database name to create or remove, and the credentials to use when connecting to the database.

**Note:** The credentials must identify a user account that has permissions to create or remove a database.

Server: DELLMOBILE

[✓ Windows authentication]

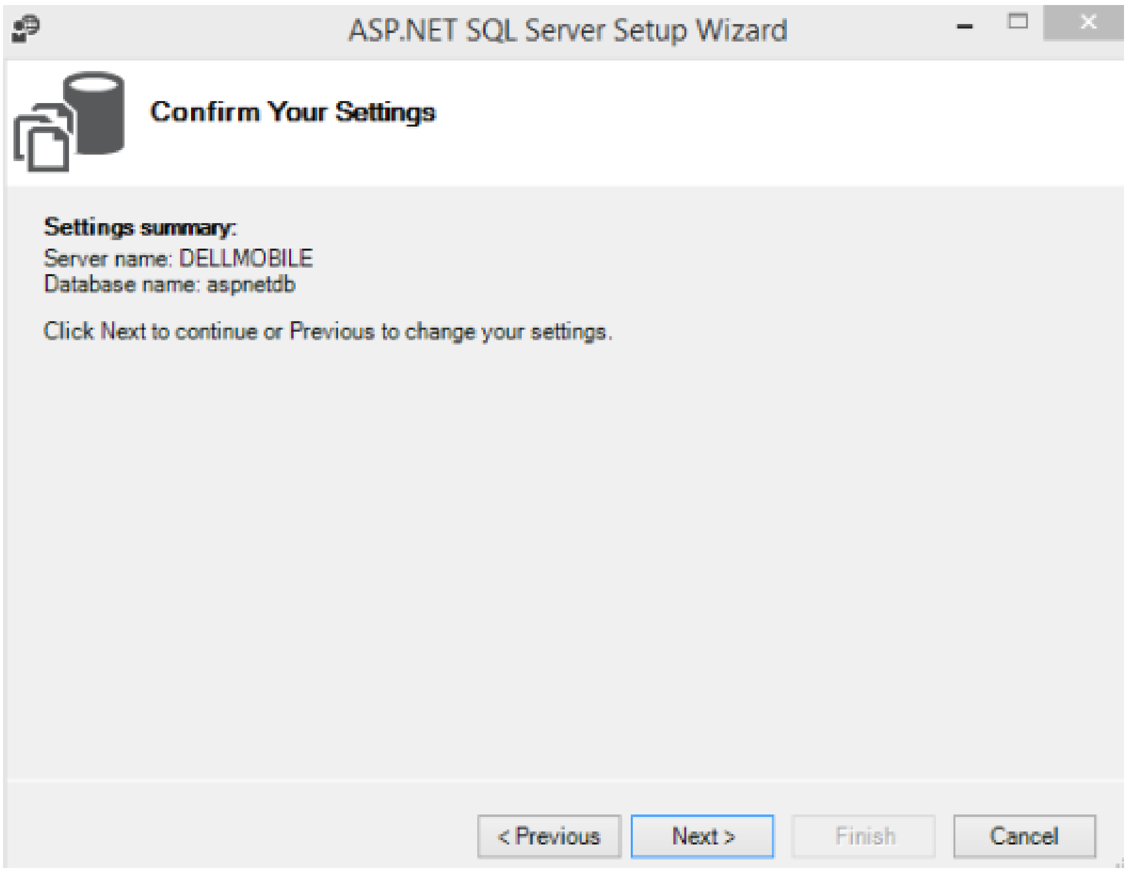
SQL Server authentication

User name:

Password:

Database: <default>

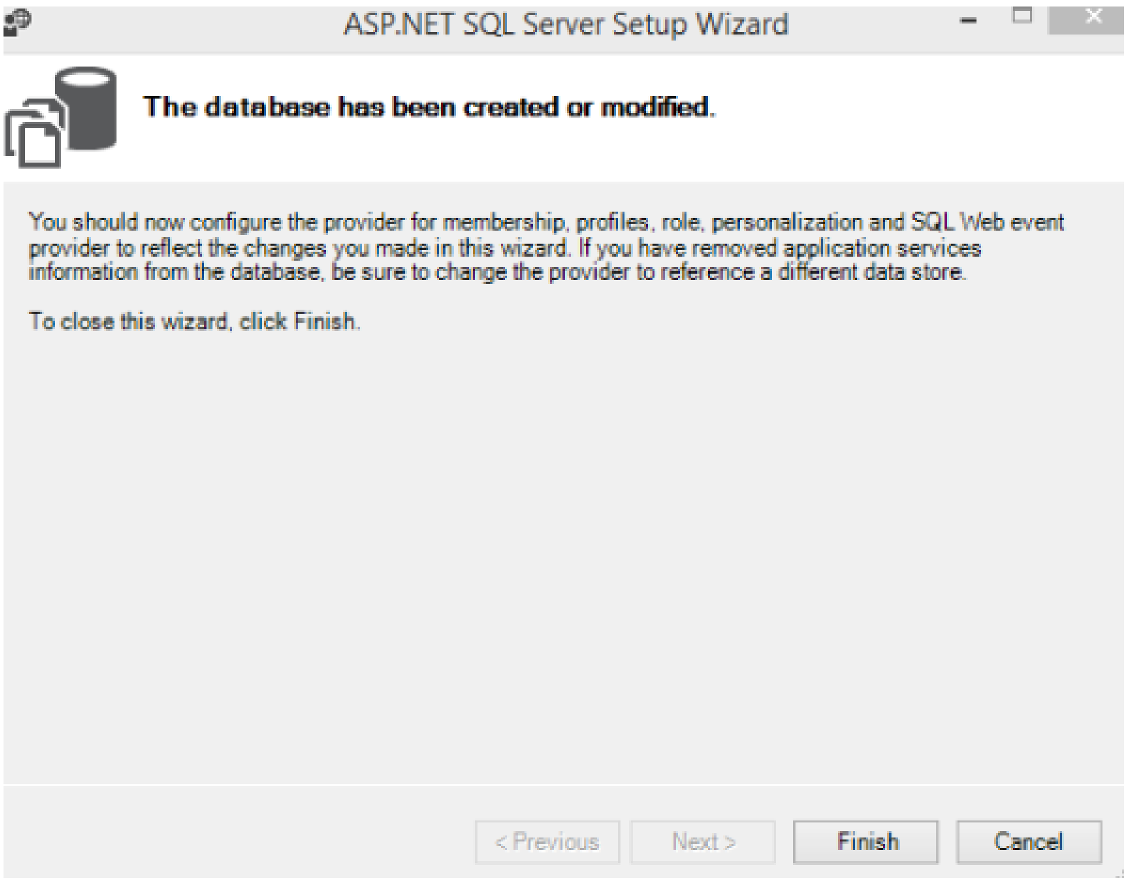
The wizard dispays the summary action, so click next to confirm  and finish the action



**Confirm Your Settings**

**Settings summary:**
Server name: DELLMOBILE
Database name: aspnetdb

Click Next to continue or Previous to change your settings.

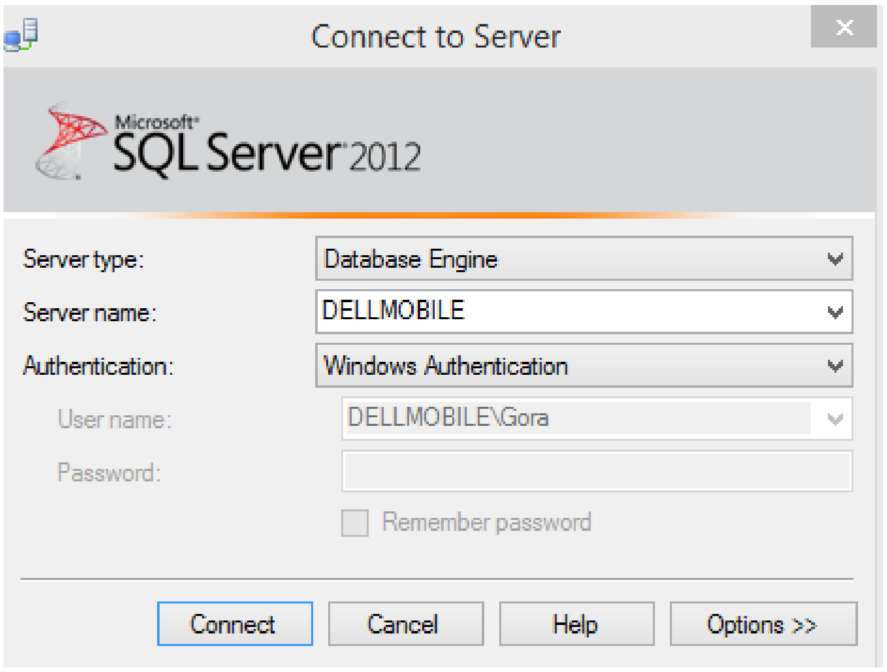


**The database has been created or modified.**

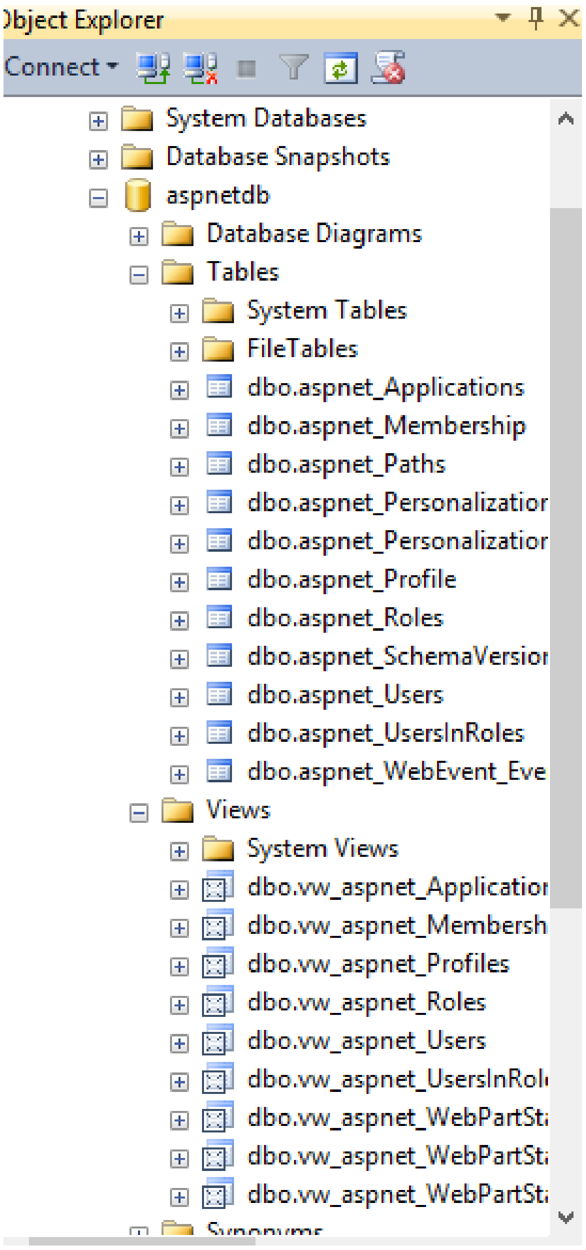
You should now configure the provider for membership, profiles, role, personalization and SQL Web event provider to reflect the changes you made in this wizard. If you have removed application services information from the database, be sure to change the provider to reference a different data store.

To close this wizard, click Finish.

Connect to sql server , locate aspnetdb database, expand tables, views, stored procedure.  you can now explore the default microsof security database business model.



Server type: Database Engine
Server name: DELLMOBILE
Authentication: Windows Authentication
User name: DELLMOBILE\Gora
Password:
Remember password



```
System Databases
Database Snapshots
aspnetdb
    Database Diagrams
    Tables
        System Tables
        FileTables
        dbo.aspnet_Applications
        dbo.aspnet_Membership
        dbo.aspnet_Paths
        dbo.aspnet_Personalizatior
        dbo.aspnet_Personalizatior
        dbo.aspnet_Profile
        dbo.aspnet_Roles
        dbo.aspnet_SchemaVersior
        dbo.aspnet_Users
        dbo.aspnet_UsersInRoles
        dbo.aspnet_WebEvent_Eve
    Views
        System Views
        dbo.vw_aspnet_Applicatior
        dbo.vw_aspnet_Membersh
        dbo.vw_aspnet_Profiles
        dbo.vw_aspnet_Roles
        dbo.vw_aspnet_Users
        dbo.vw_aspnet_UsersInRol
        dbo.vw_aspnet_WebPartSt
        dbo.vw_aspnet_WebPartSt
        dbo.vw_aspnet_WebPartSt
    Synonyms
```

for more informations please visit msdn web site

This tutorial is the first of a series Mastering Custum ASP.NET MemberShip Provider using ASP.NET MVC , please see next Introduction to entity framework database first

## 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

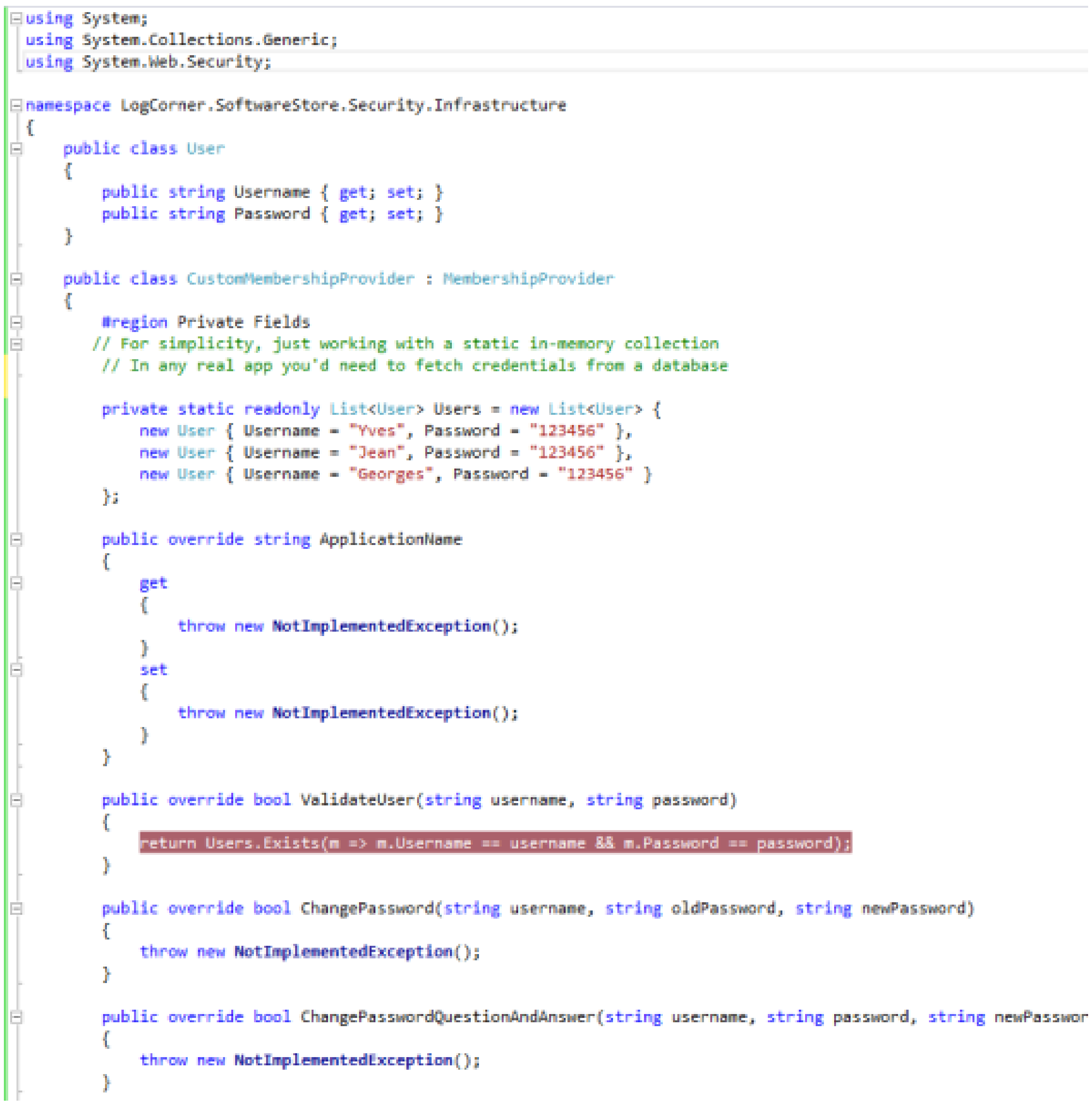
Let's go

## 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

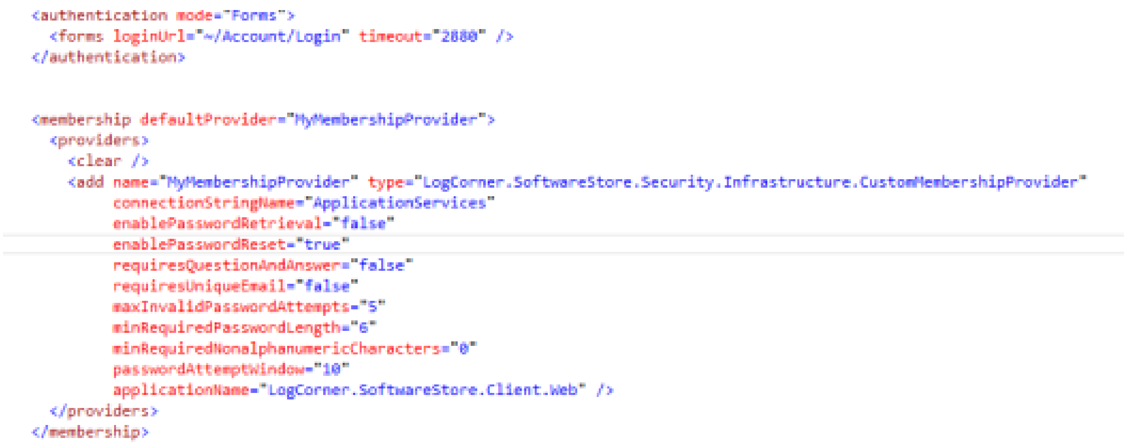


```
using System;
using System.Collections.Generic;
using System.Web.Security;

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*

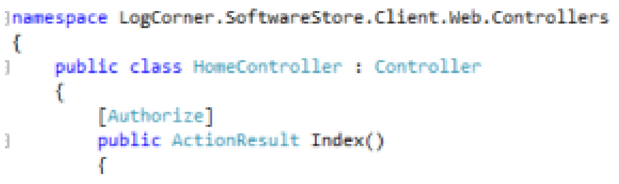
## B. Create an ASP.NET MVC4 application Client

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



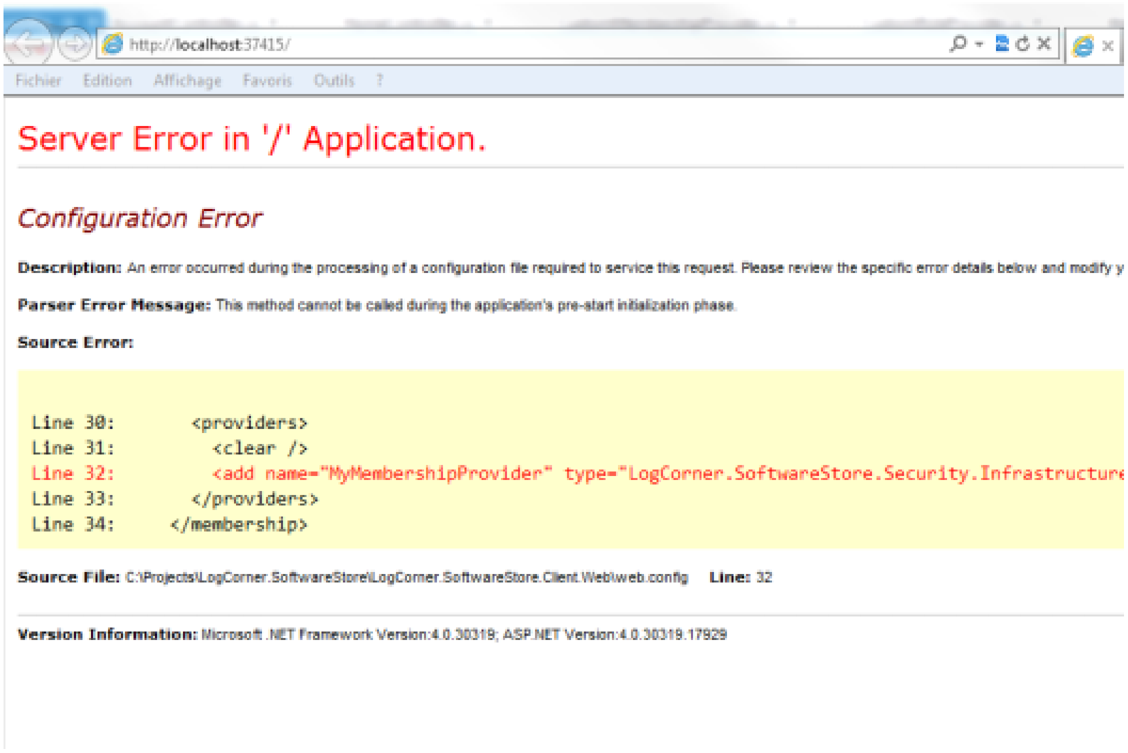
```
<authentication mode="Forms">
  <forms loginUrl="~/Account/Login" timeout="2880" />
</authentication>


<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



Server Error in '/' Application.

*Configuration Error*

**Description:** An error occurred during the processing of a configuration file required to service this request. Please review the specific error details below and modify y

**Parser Error Message:** This method cannot be called during the application's pre-start initialization phase.

**Source Error:**

```
Line 30:       <providers>
Line 31:         <clear />
Line 32:         <add name="MyMembershipProvider" type="LogCorner.SoftwareStore.Security.Infrastructure
Line 33:       </providers>
Line 34:     </membership>
```

**Source File:** C:\Projects\LogCorner.SoftwareStore\LogCorner.SoftwareStore.Client.Web\web.config    **Line:** 32

**Version Information:** Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.17929

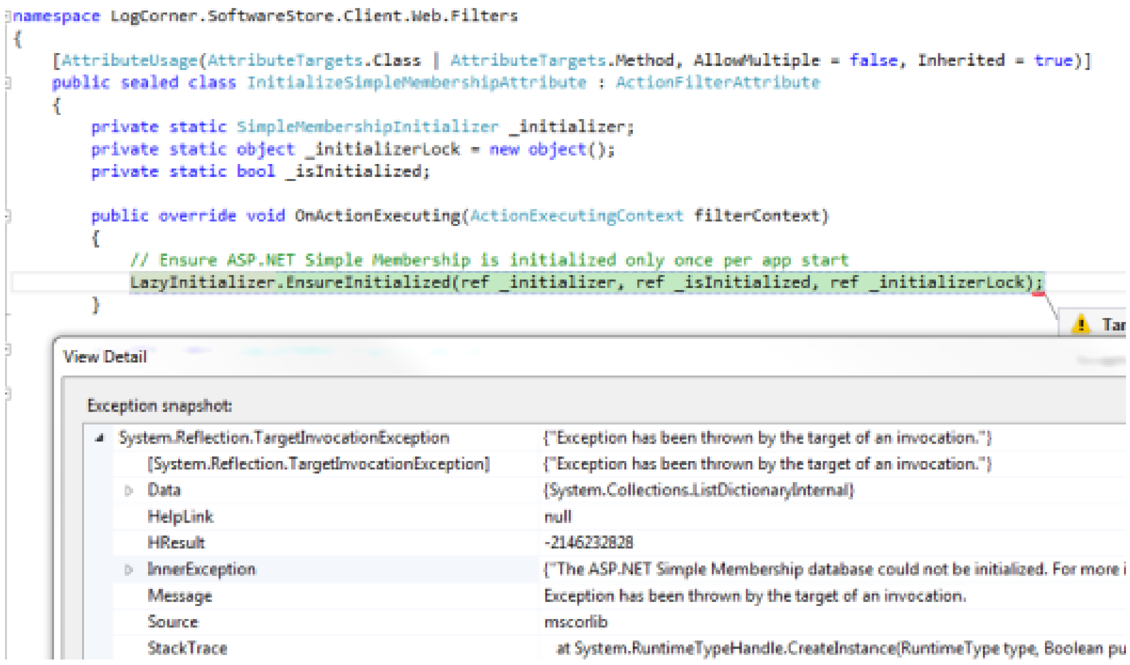
6.   do not panic, proceed as follows :

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);
        }
```

Tar

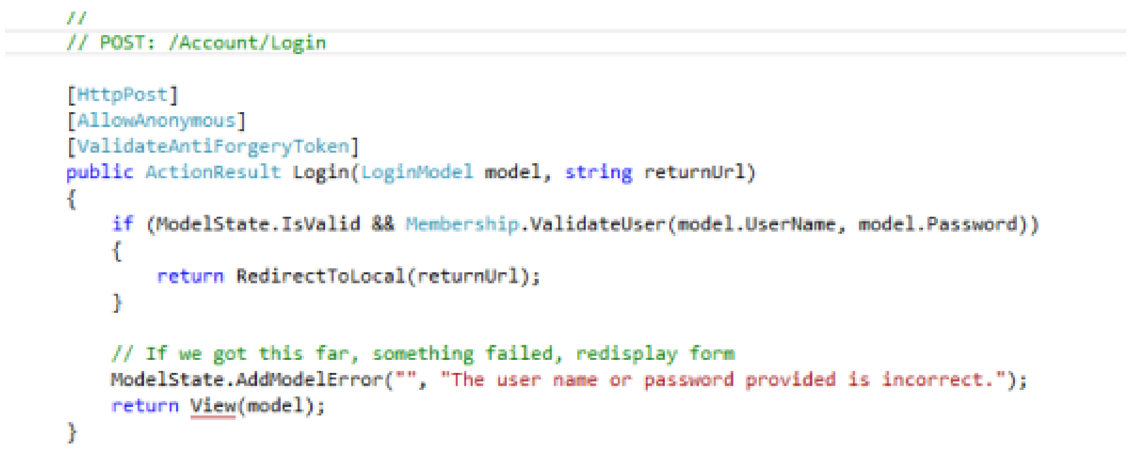
View Detail

Exception snapshot:

| | |
|---|---|
| System.Reflection.TargetInvocationException | {"Exception has been thrown by the target of an invocation."} |
| [System.Reflection.TargetInvocationException] | {"Exception has been thrown by the target of an invocation."} |
| Data | {System.Collections.ListDictionaryInternal} |
| HelpLink | null |
| HResult | -2146232828 |
| InnerException | {"The ASP.NET Simple Membership database could not be initialized. For more i |
| Message | Exception has been thrown by the target of an invocation. |
| Source | mscorlib |
| StackTrace | at System.RuntimeTypeHandle.CreateInstance(RuntimeType type, Boolean pu |

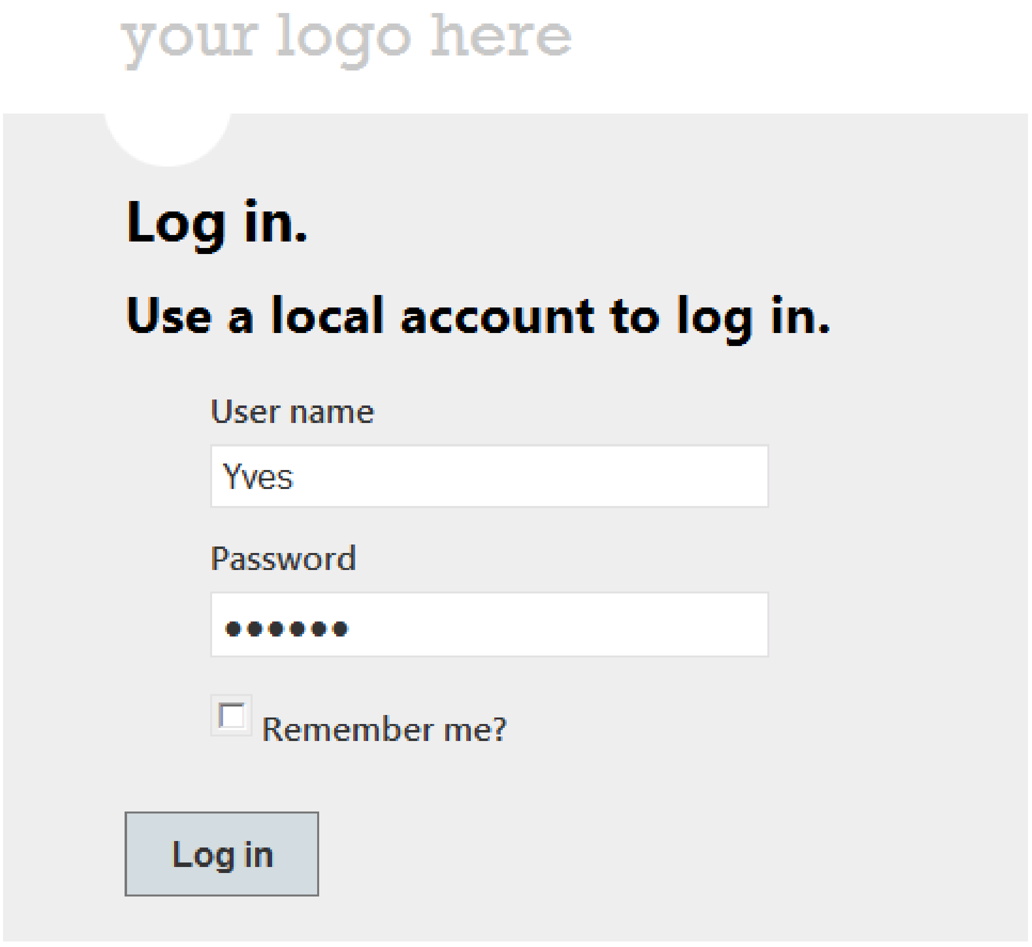
8. To fix it Open **AccountController** and comment **InitializeSimpleMembership** , because we using Custom Membership Provider instead of Simple Membership

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


your logo here

Log in.

Use a local account to log in.

User name

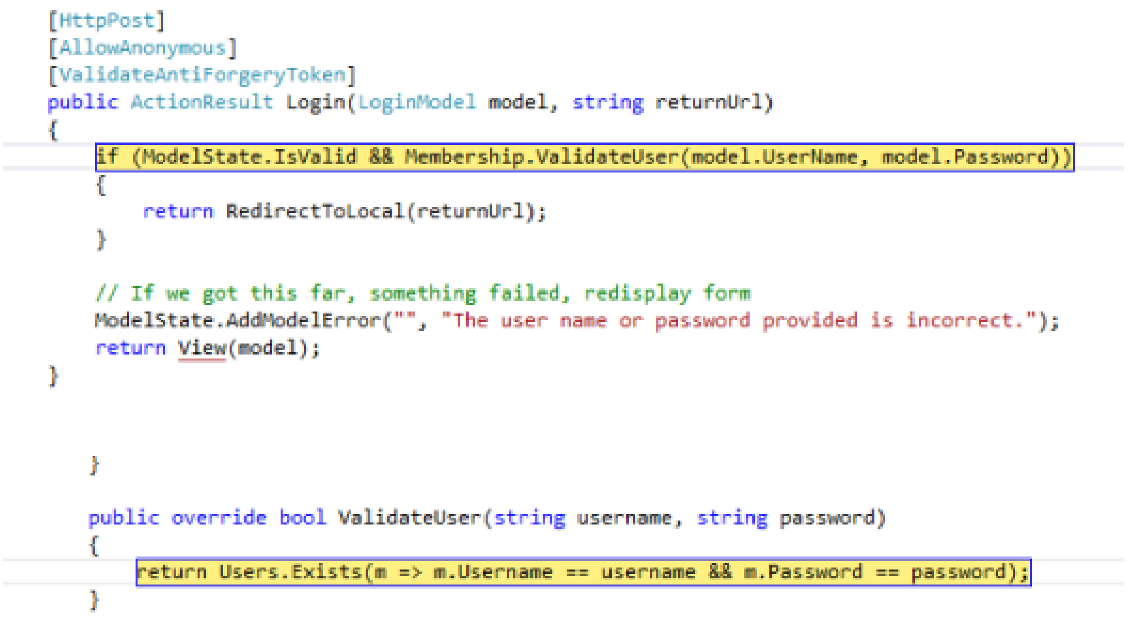
Yves

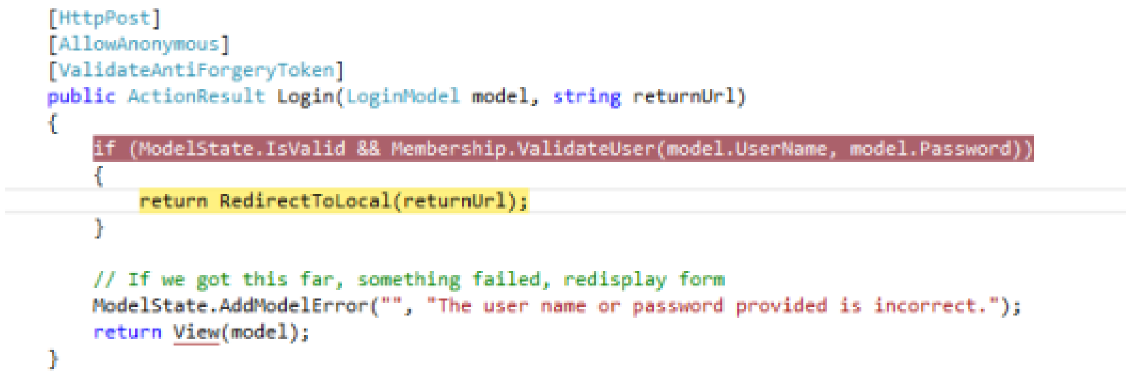
Password

●●●●●●

☐ Remember me?

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);
    }
```

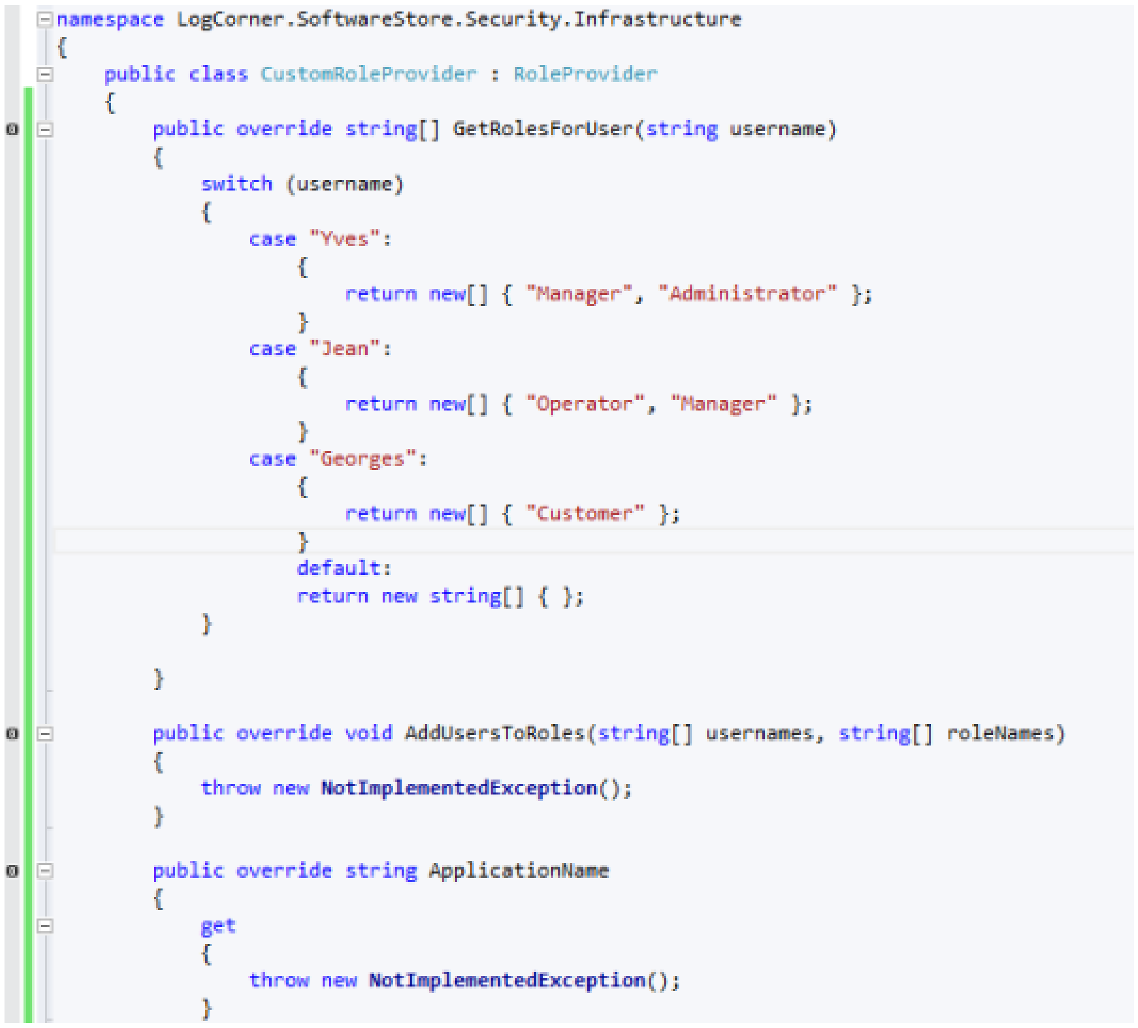


```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public ActionResult Login(LoginModel model, string returnUrl)
{
    if (ModelState.IsValid && Membership.ValidateUser(model.UserName, model.Password))
    {
        return RedirectToLocal(returnUrl);
    }

    // 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

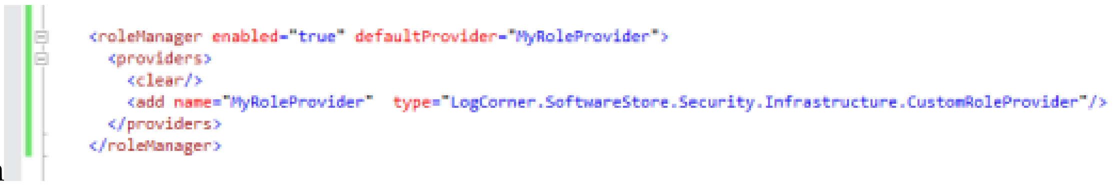
To configure custom role provider, please proceed as follow :

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();
            }
```

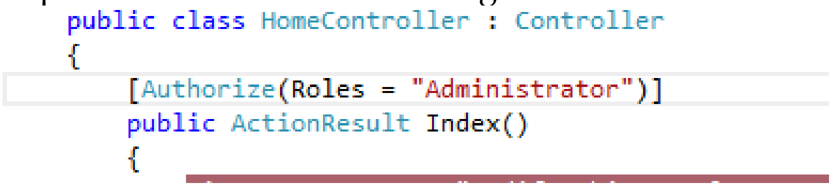
3. Now open web.config file of your client asp.net web application and add a **RoleManager**



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

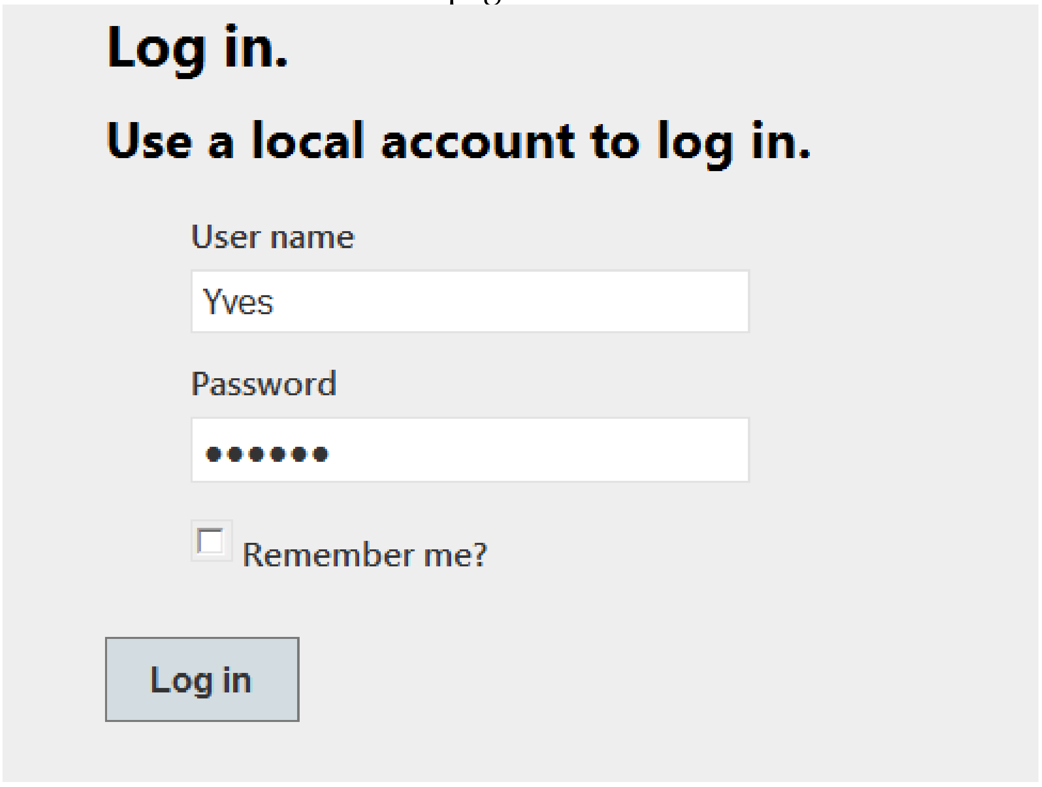
section

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



Log in.

Use a local account to log in.

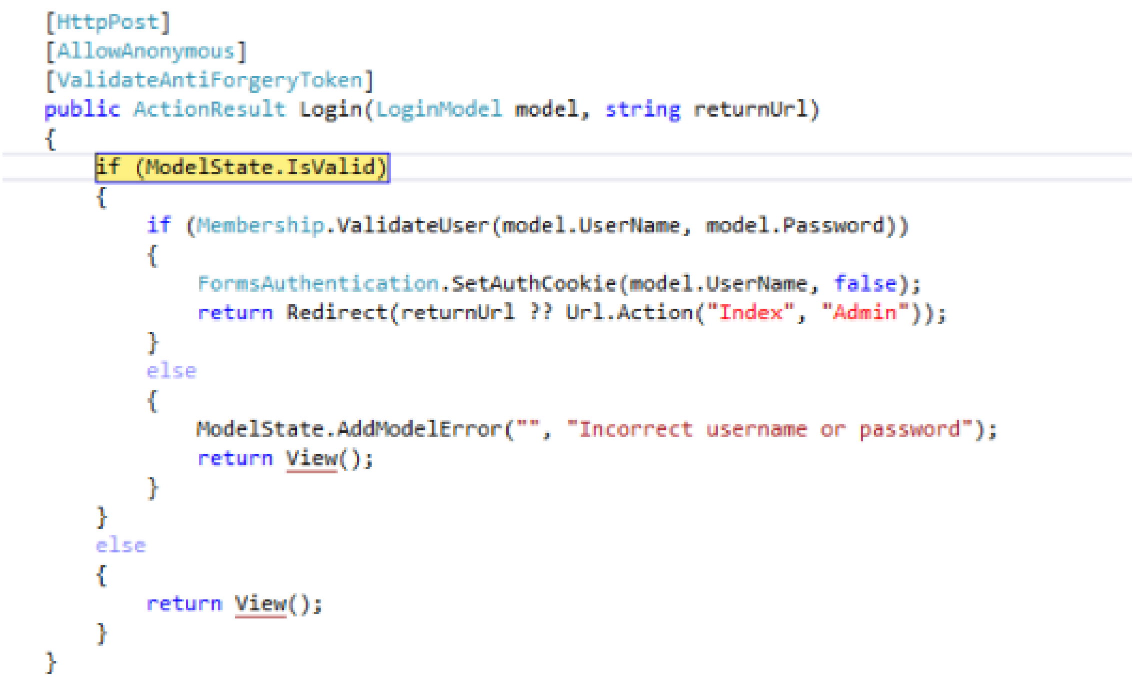
User name

Yves

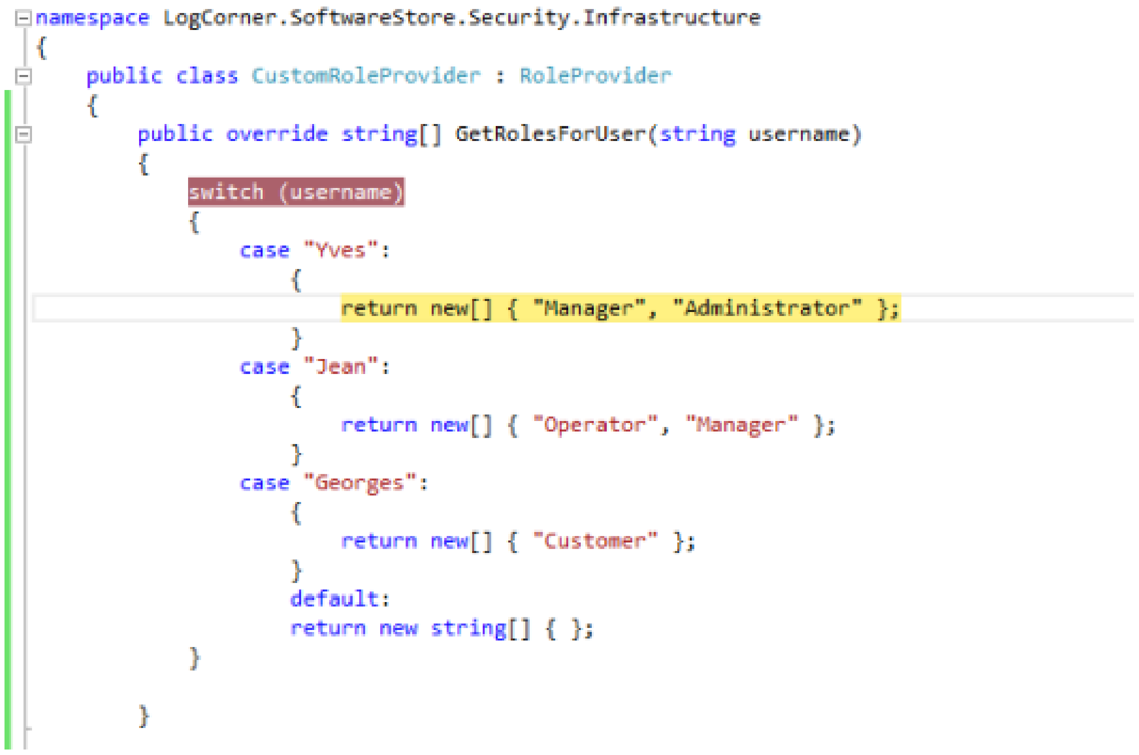
Password

●●●●●●

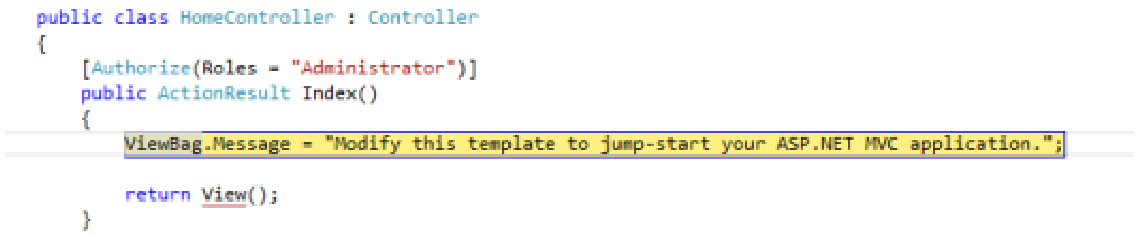
☐ Remember me?



```
[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

## How to configure Custom Membership Provider using ASP.NET MVC4 with external login like facebook, yahoo , google or other relying party accounts.

*28 Mercredi août 2013*

PUBLIÉ PAR GORALEYE IN ARCHITECTURE, ASP.NET, C#.NET, EXTERNAL LOGIN, MEMBERSHIP, OAUTH PROVIDERS, ROLE PROVIDER, SECURITY

≈ 3 COMMENTAIRES

*Tags*
*Application Services, Architecture, ASP.NET, C#.NET, Custom Membership Provider, OAuth Providers, 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.

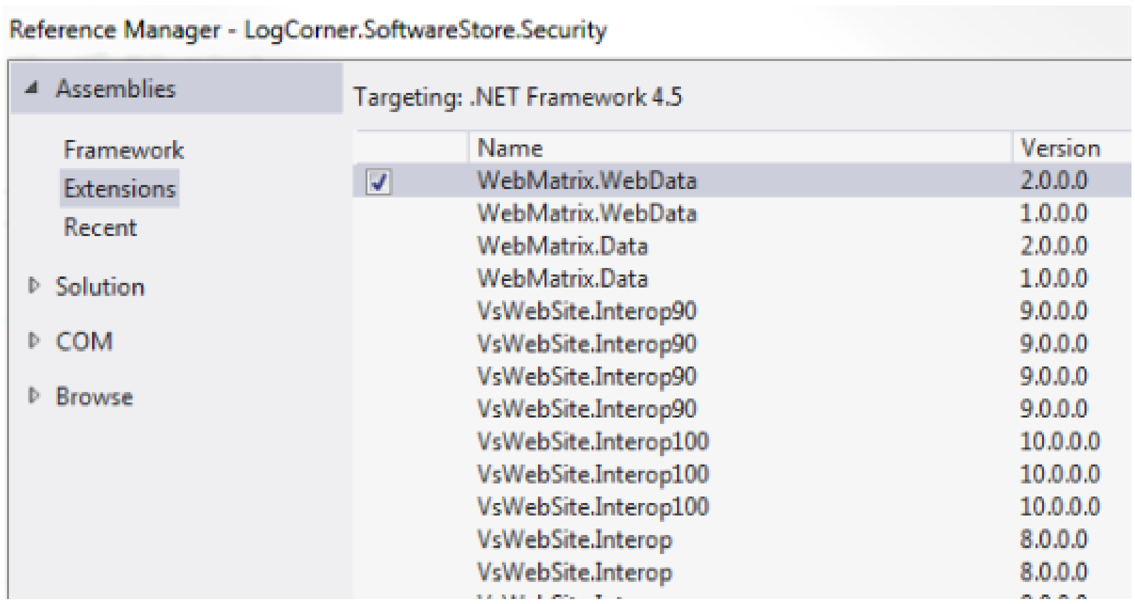
- You need to store membership information in a data source that is not supported by the membership providers included with the .NET Framework, such as a MysQL database, an Oracle database, or other data sources.
- You need to manage membership information using a database schema that is different from 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 that enable external login like facebook, yahoo , google or other relying party accounts.

Let's go

## A. Create a Custom MemberShip Application class Library

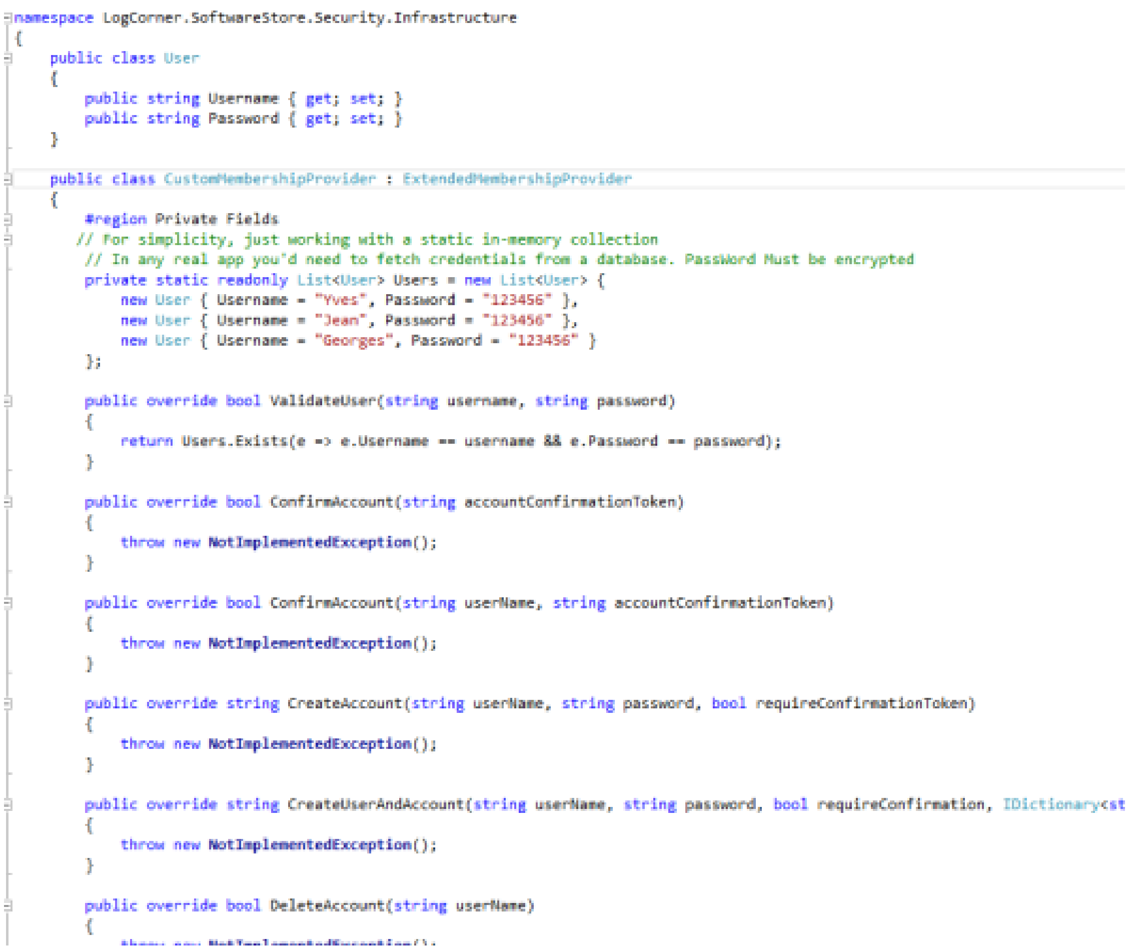
1. Create a class Library Project (our sample Projet name is **LogCorner.SoftwareStore.Security**)
2. Reference the assembly **System.Web.ApplicationServices** (Right Click Reference => Add reference and Select Assemblies => navigate to System.Web.ApplicationServices and add it)
3. Create a Class **CustomMembershipProvider** . Here we are going to enable external login like facebook, yahoo , google or other relying party accounts.So derive our custom class from **ExtendedMembershipProvider** whose base class is **MembershipProvider**
4. So reference the assembly **WebMatrix.WebData** or get it via nuget Packages


Reference Manager - LogCorner.SoftwareStore.Security

Targeting: .NET Framework 4.5

| | Name | Version |
|---|---|---|
| ✓ | WebMatrix.WebData | 2.0.0.0 |
| | WebMatrix.WebData | 1.0.0.0 |
| | WebMatrix.Data | 2.0.0.0 |
| | WebMatrix.Data | 1.0.0.0 |
| | VsWebSite.Interop90 | 9.0.0.0 |
| | VsWebSite.Interop90 | 9.0.0.0 |
| | VsWebSite.Interop90 | 9.0.0.0 |
| | VsWebSite.Interop90 | 9.0.0.0 |
| | VsWebSite.Interop100 | 10.0.0.0 |
| | VsWebSite.Interop100 | 10.0.0.0 |
| | VsWebSite.Interop100 | 10.0.0.0 |
| | VsWebSite.Interop | 8.0.0.0 |
| | VsWebSite.Interop | 8.0.0.0 |

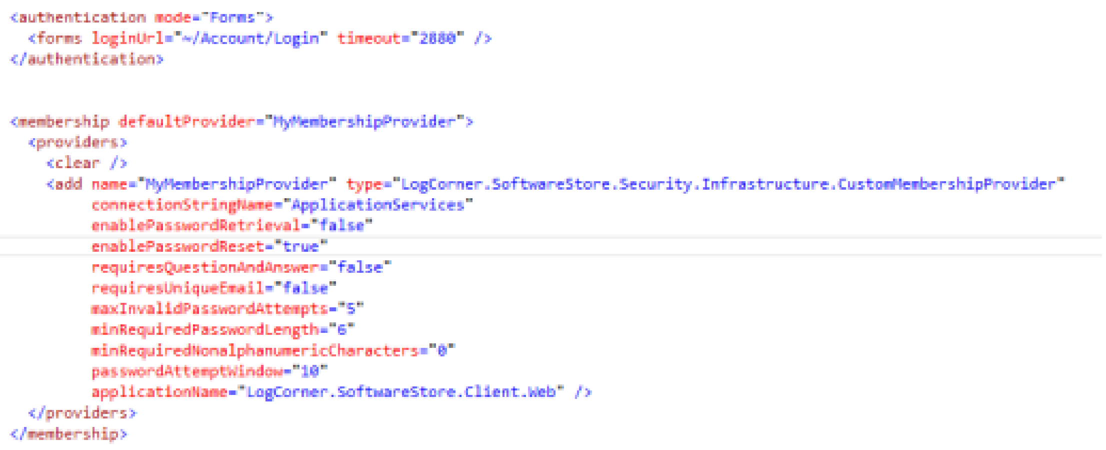
5.
6. Override ValidateUser as follow


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

    public class CustomMembershipProvider : ExtendedMembershipProvider
    {
        #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. PassWord Must be encrypted
        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 bool ValidateUser(string username, string password)
        {
            return Users.Exists(e => e.Username == username && e.Password == password);
        }

        public override bool ConfirmAccount(string accountConfirmationToken)
        {
            throw new NotImplementedException();
        }

        public override bool ConfirmAccount(string userName, string accountConfirmationToken)
        {
            throw new NotImplementedException();
        }

        public override string CreateAccount(string userName, string password, bool requireConfirmationToken)
        {
            throw new NotImplementedException();
        }

        public override string CreateUserAndAccount(string userName, string password, bool requireConfirmation, IDictionary<st
        {
            throw new NotImplementedException();
        }

        public override bool DeleteAccount(string userName)
        {
```

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*

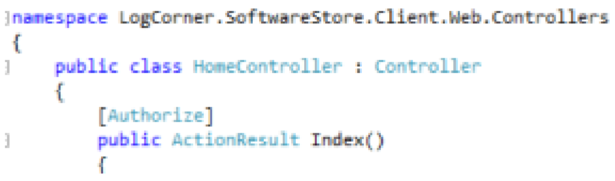
## B. Create an ASP.NET MVC4 application Client

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



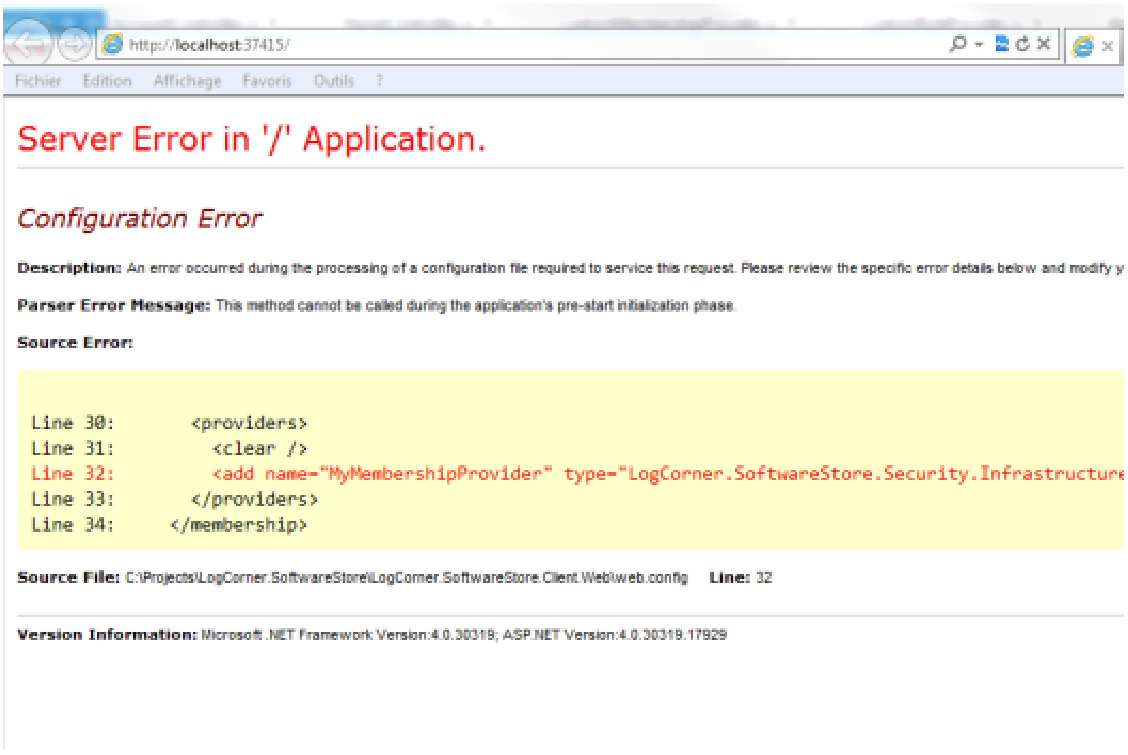
```
<authentication mode="Forms">
  <forms loginUrl="~/Account/Login" timeout="2880" />
</authentication>


<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



Server Error in '/' Application.

*Configuration Error*

**Description:** An error occurred during the processing of a configuration file required to service this request. Please review the specific error details below and modify y

**Parser Error Message:** This method cannot be called during the application's pre-start initialization phase.

**Source Error:**

```
Line 30:       <providers>
Line 31:         <clear />
Line 32:         <add name="MyMembershipProvider" type="LogCorner.SoftwareStore.Security.Infrastructure
Line 33:       </providers>
Line 34:     </membership>
```

**Source File:** C:\Projects\LogCorner.SoftwareStore\LogCorner.SoftwareStore.Client.Web\web.config **Line:** 32

**Version Information:** Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.17929

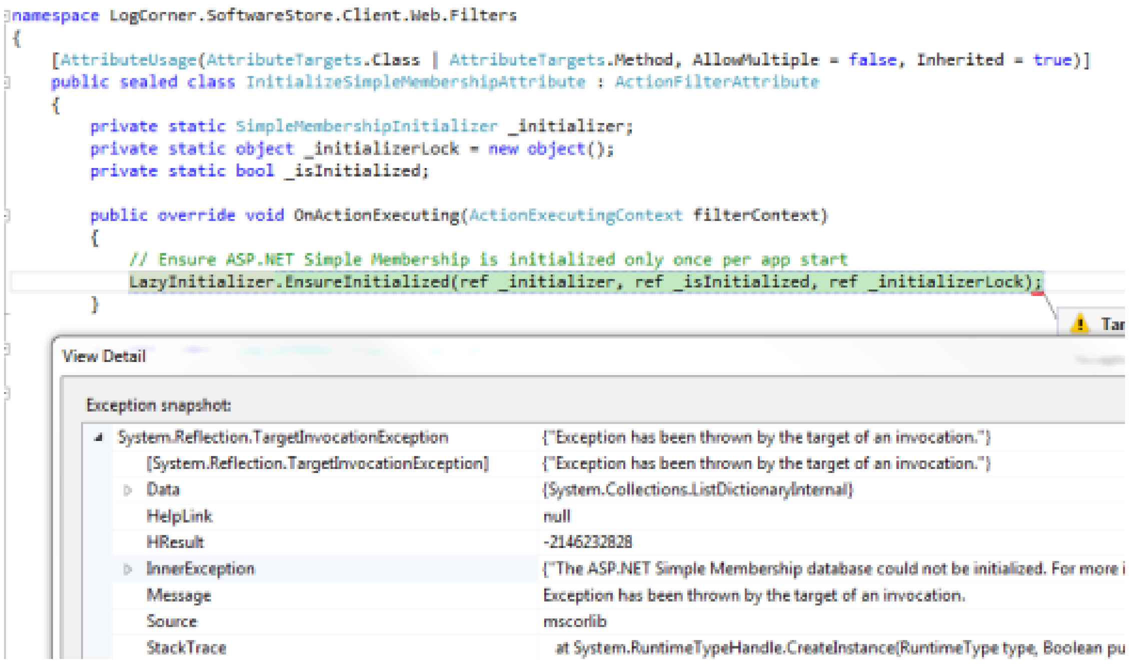
6. do not panic, proceed as follows :

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



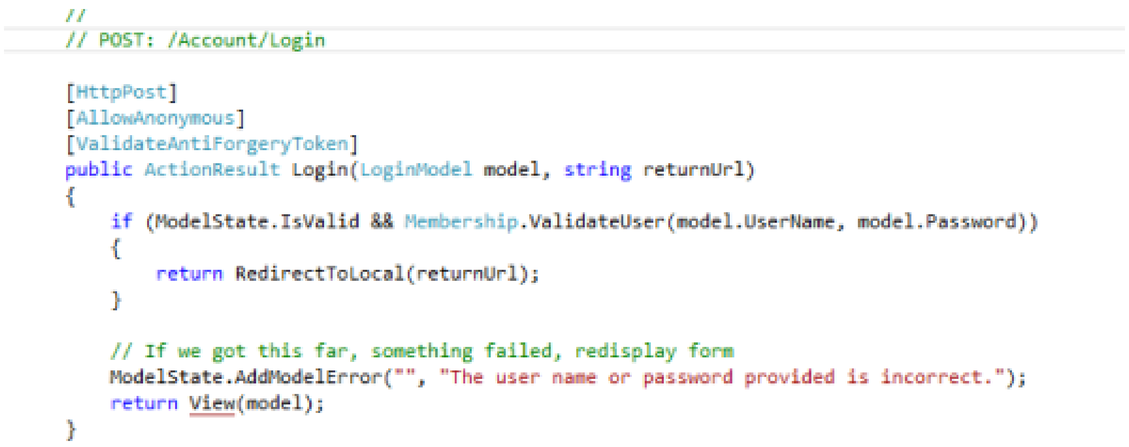
```
namespace LogCorner.SoftwareStore.Client.Web.Filters
{
    [AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = false, Inherited = true)]
    public sealed class InitializeSimpleMembershipAttribute : ActionFilterAttribute
    {
        private static SimpleMembershipInitializer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

Exception snapshot:

| | |
|---|---|
| System.Reflection.TargetInvocationException | {"Exception has been thrown by the target of an invocation."} |
| [System.Reflection.TargetInvocationException] | {"Exception has been thrown by the target of an invocation."} |
| Data | {System.Collections.ListDictionaryInternal} |
| HelpLink | null |
| HResult | -2146232828 |
| InnerException | {"The ASP.NET Simple Membership database could not be initialized. For more i |
| Message | Exception has been thrown by the target of an invocation. |
| Source | mscorlib |
| StackTrace | at System.RuntimeTypeHandle.CreateInstance(RuntimeType type, Boolean pu |

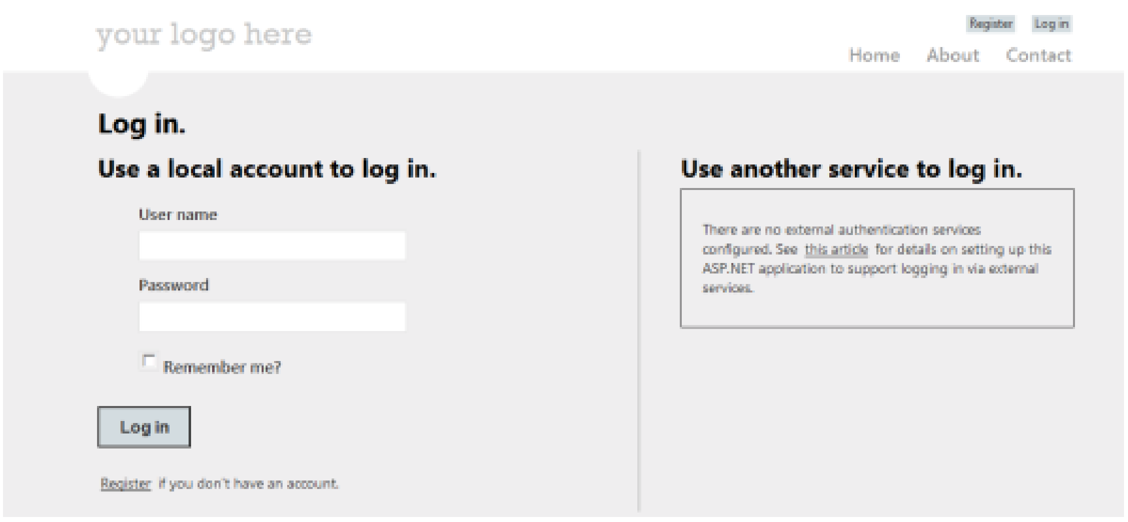
8. To fix it Open **AccountController** and comment **InitializeSimpleMembership** , because we using Custom Membership Provider instead of Simple Membership

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. Note that external logins are disabled. Will going to configure external logins further



Log in.

Use a local account to log in.

User name

Password

Remember me?

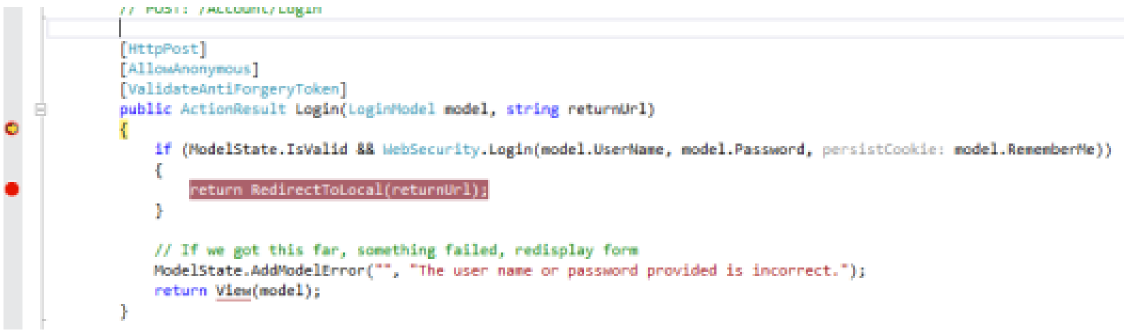
Log in

Register if you don't have an account.

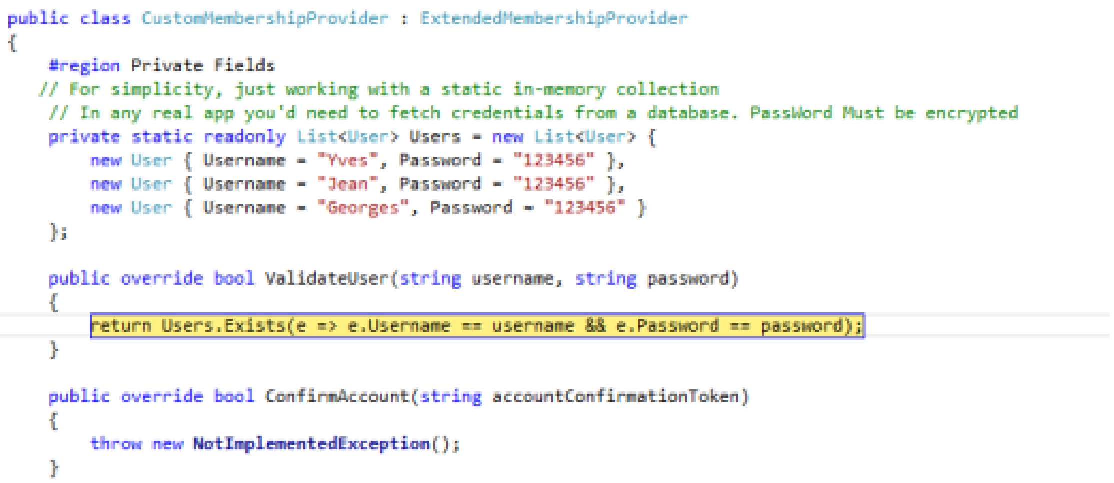
Use another service to log in.

There are no external authentication services configured. See this article for details on setting up this ASP.NET application to support logging in via external services.

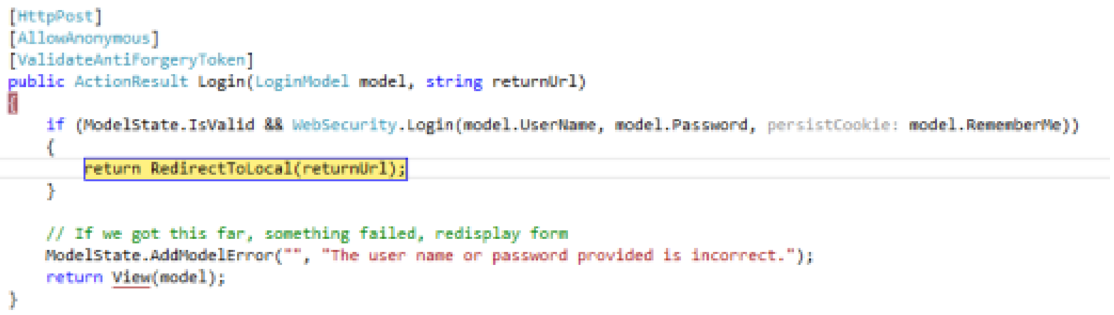
11. Enter user credentials and click Log In, then you will have the execution workflow below :



```
        // POST: /Account/Login

        [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);
        }
```



```
public class CustomMembershipProvider : ExtendedMembershipProvider
{
    #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. PassWord Must be encrypted
    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 bool ValidateUser(string username, string password)
    {
        return Users.Exists(e => e.Username == username && e.Password == password);
    }

    public override bool ConfirmAccount(string accountConfirmationToken)
    {
        throw new NotImplementedException();
    }
```



```
[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);
}
```

# B. Configuration Custom Membership Provider using ASP.NET MVC4 with external login like facebook, yahoo , google or other relying party accounts.

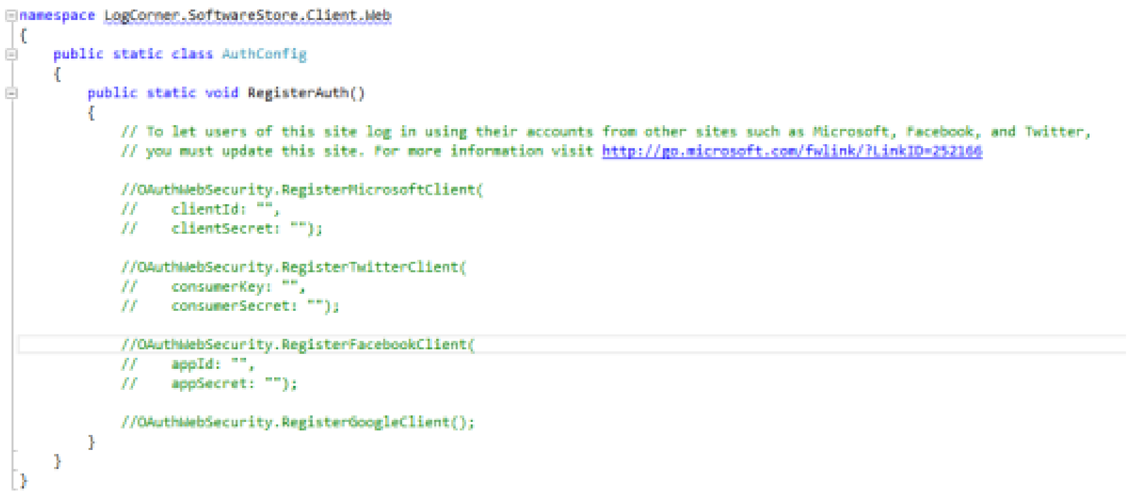
1. **Registering with an external provider**

To authenticate users with credentials from an external provider, you must register your web site with the provider. When you register your site, you will receive the parameters (such as key or id, and secret) to include when registering the client. You must have an account with the providers you wish to use.

To successfully register your site, follow the instructions provided on this sites :

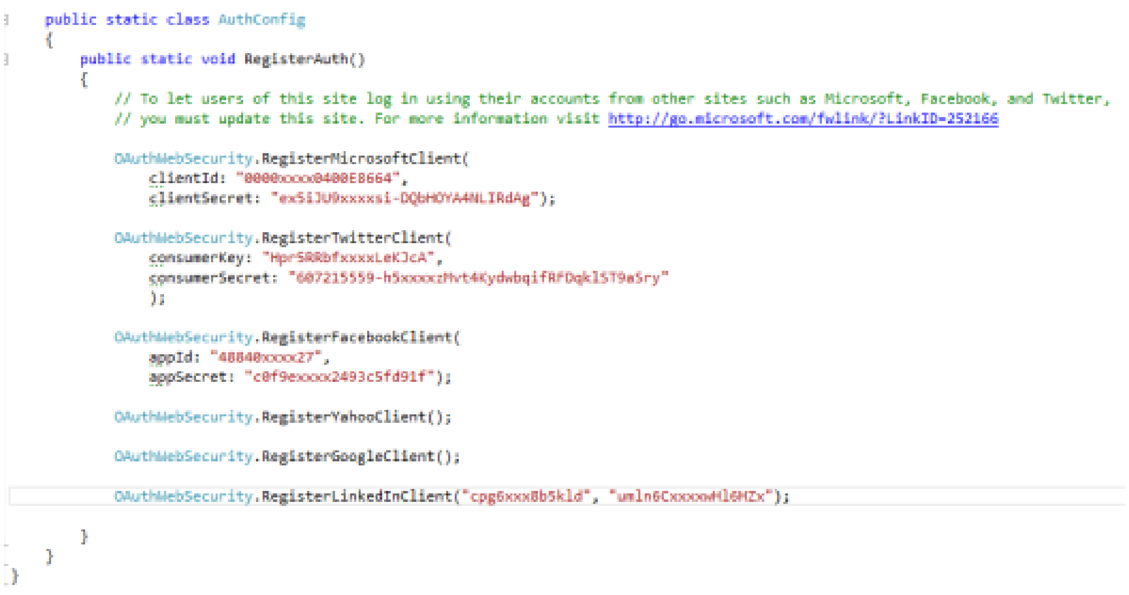
- FaceBook developer
- Google Developer
- Microsoft developer
- Twitter developer

2. navigation to App_Start of your web application ,locate AuthConfig.cs file and open it



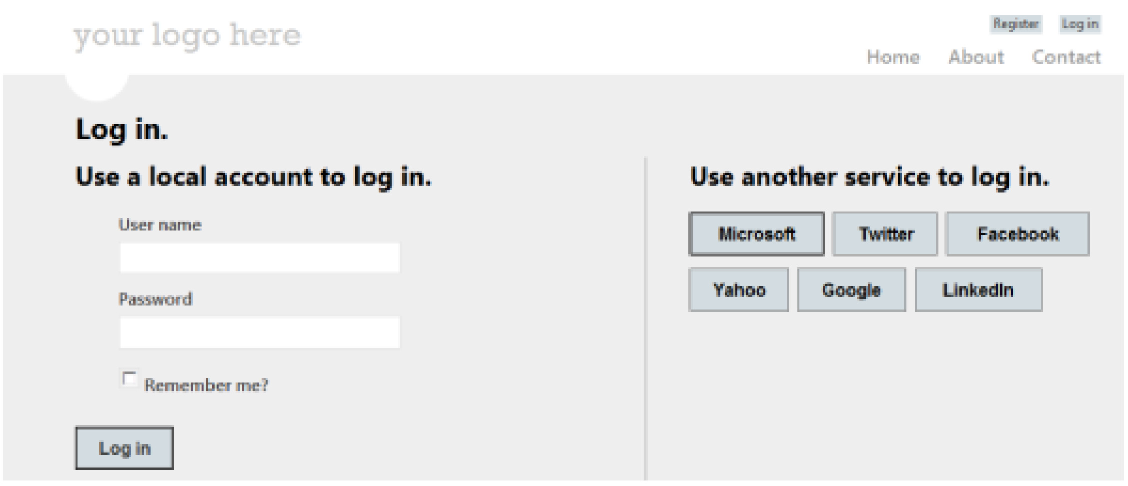
```
namespace LogCorner.SoftwareStore.Client.Web
{
    public static class AuthConfig
    {
        public static void RegisterAuth()
        {
            // To let users of this site log in using their accounts from other sites such as Microsoft, Facebook, and Twitter,
            // you must update this site. For more information visit http://go.microsoft.com/fwlink/?LinkID=252166

            //OAuthWebSecurity.RegisterMicrosoftClient(
            //    clientId: "",
            //    clientSecret: "");

            //OAuthWebSecurity.RegisterTwitterClient(
            //    consumerKey: "",
            //    consumerSecret: "");

            //OAuthWebSecurity.RegisterFacebookClient(
            //    appId: "",
            //    appSecret: "");

            //OAuthWebSecurity.RegisterGoogleClient();
        }
    }
}
```

3. Update **RegisterAuth** method with parameters (such as key or id, and secret) you have received from facebook, twitter, google, yahoo. etc……

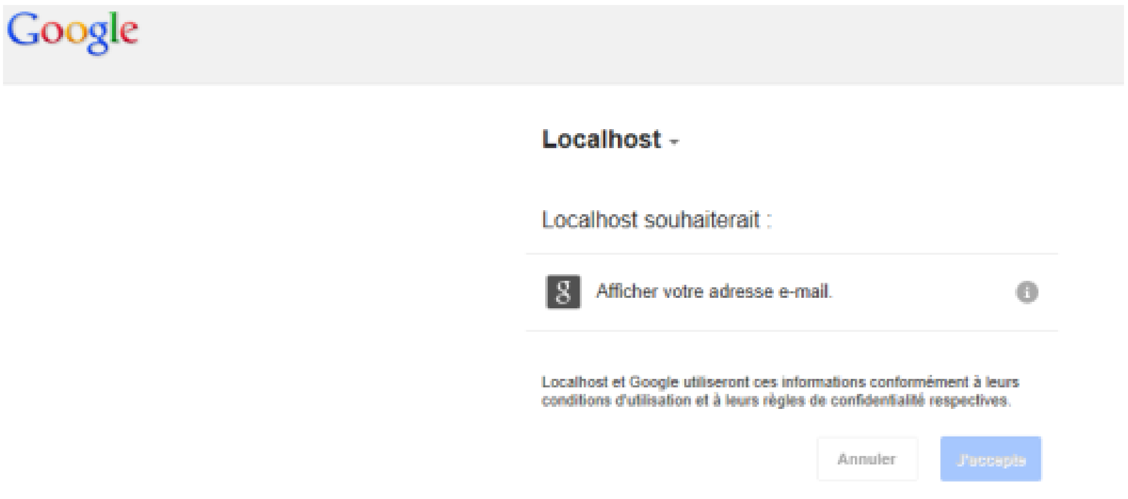


```
public static class AuthConfig
{
    public static void RegisterAuth()
    {
        // To let users of this site log in using their accounts from other sites such as Microsoft, Facebook, and Twitter,
        // you must update this site. For more information visit http://go.microsoft.com/fwlink/?LinkID=252166

        OAuthWebSecurity.RegisterMicrosoftClient(
            clientId: "0000xxxx0400E8664",
            clientSecret: "ex5iJU9xxxxsi-DQbHOYA4NLIRdAg");

        OAuthWebSecurity.RegisterTwitterClient(
            consumerKey: "HprSRRbfxxxxLeKJcA",
            consumerSecret: "607215559-h5xxxxzMvt4KydwbqifRFDqkl5T9a5ry"
            );

        OAuthWebSecurity.RegisterFacebookClient(
            appId: "48840xxxx27",
            appSecret: "c0f9exxxx2493c5fd91f");

        OAuthWebSecurity.RegisterYahooClient();

        OAuthWebSecurity.RegisterGoogleClient();

        OAuthWebSecurity.RegisterLinkedInClient("cpg6xxx8b5kld", "umln6CxxxxwHl6MZx");

    }
}
```

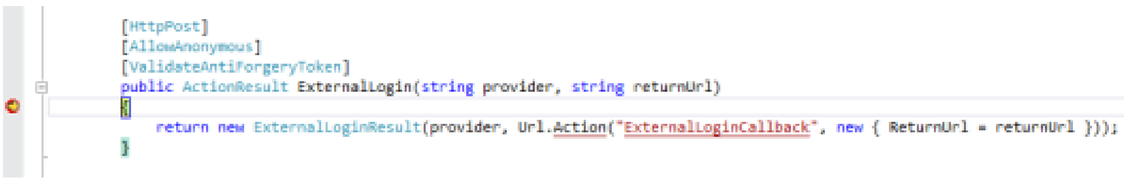
4. Next run your web application, then your logon form must be updated with external logins



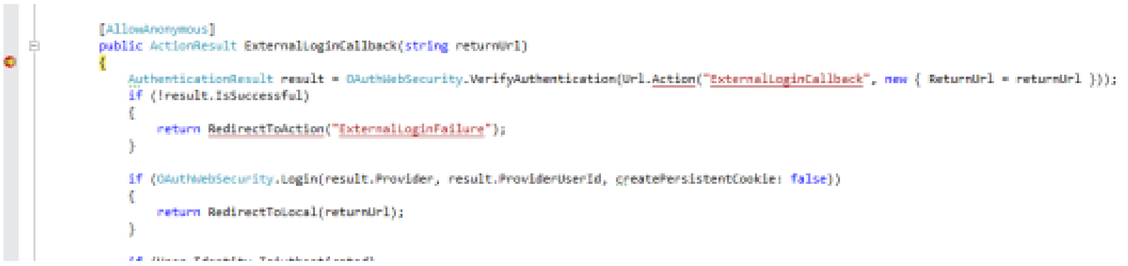
5. Click on Google button (for example )



Localhost et Google utiliseront ces informations conformément à leurs conditions d'utilisation et à leurs règles de confidentialité respectives.



```
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLogin(string provider, string returnUrl)
        {
            return new ExternalLoginResult(provider, Url.Action("ExternalLoginCallback", new { ReturnUrl = returnUrl }));
        }
```



```
[AllowAnonymous]
public ActionResult ExternalLoginCallback(string returnUrl)
{
    AuthenticationResult result = OAuthWebSecurity.VerifyAuthentication(Url.Action("ExternalLoginCallback", new { ReturnUrl = returnUrl }));
    if (!result.IsSuccessful)
    {
        return RedirectToAction("ExternalLoginFailure");
    }

    if (OAuthWebSecurity.Login(result.Provider, result.ProviderUserId, createPersistentCookie: false))
    {
        return RedirectToLocal(returnUrl);
    }
```

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

We have at the end of this tutorial, thank you for feedbacks.