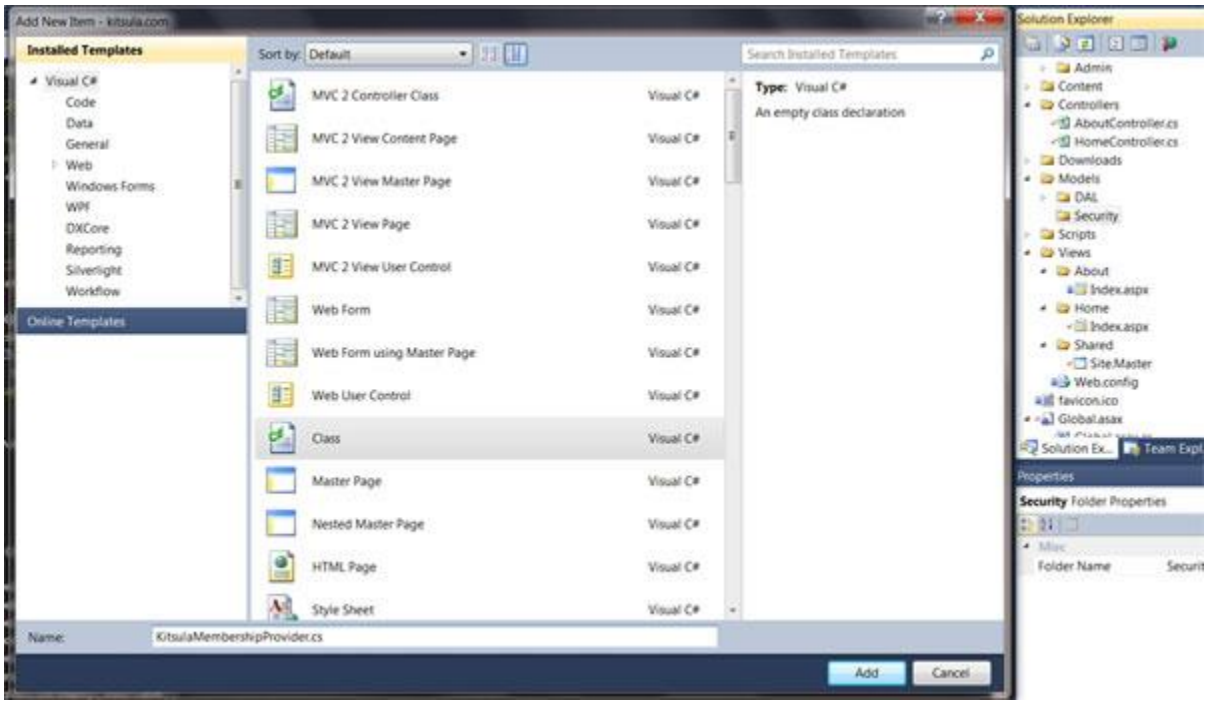
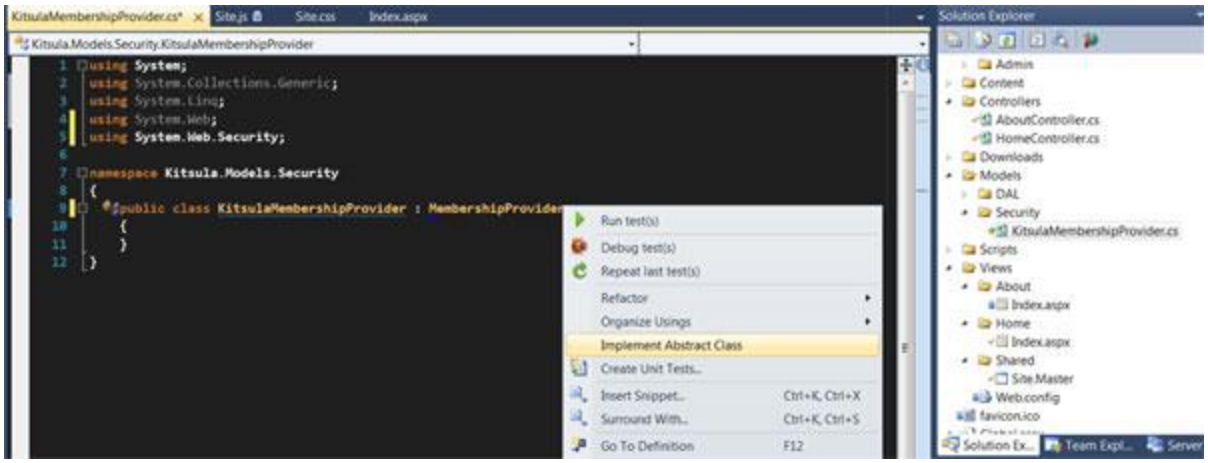
Custom Membership Provider for MVC

.NET Framework allow you authorization engine based on two layer security: users and roles. Most popular is SQL Membership Provider and it contains methods and properties specific to using SQL as a data store for membership information. But for standard Membership provider you should create separate database. To use users or contacts information from your DB for authentication users with standard .NET form authorization you can create Custom Membership Provider and Custom Role Provider. Today we will create simple Custom Membership Provider and I'll show how to configure it for using in MVC.

First we need create class inherits from the MembershipProvider class, which provides data store-agnostic methods for storing membership information and checking login info.





Class have ValidateUser function where we can put our logic for user login validating:

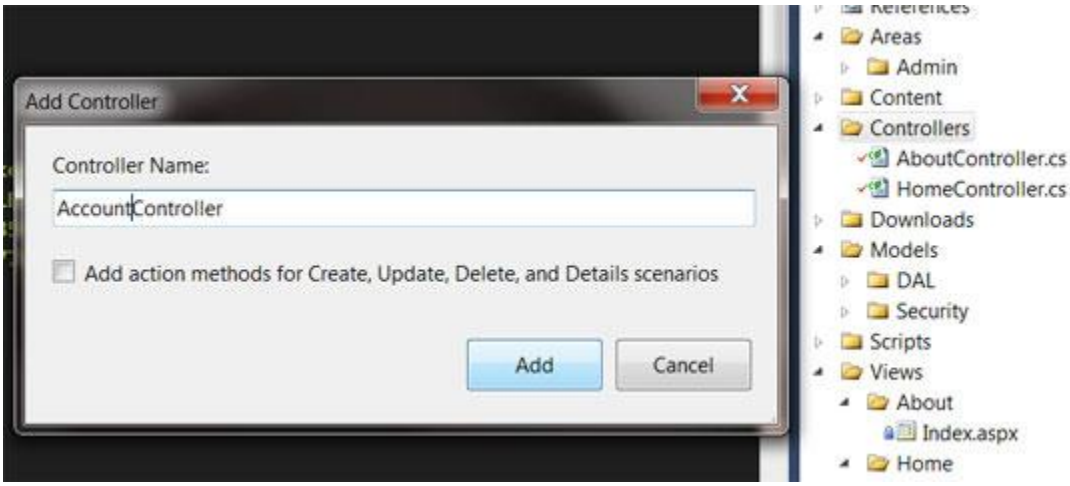
```
public override bool ValidateUser(string username, string password)
{
    if (username.Equals("Shaman", StringComparison.CurrentCultureIgnoreCase)
        && password.Equals("password"))
        return true;
    else
        return false;
}
```

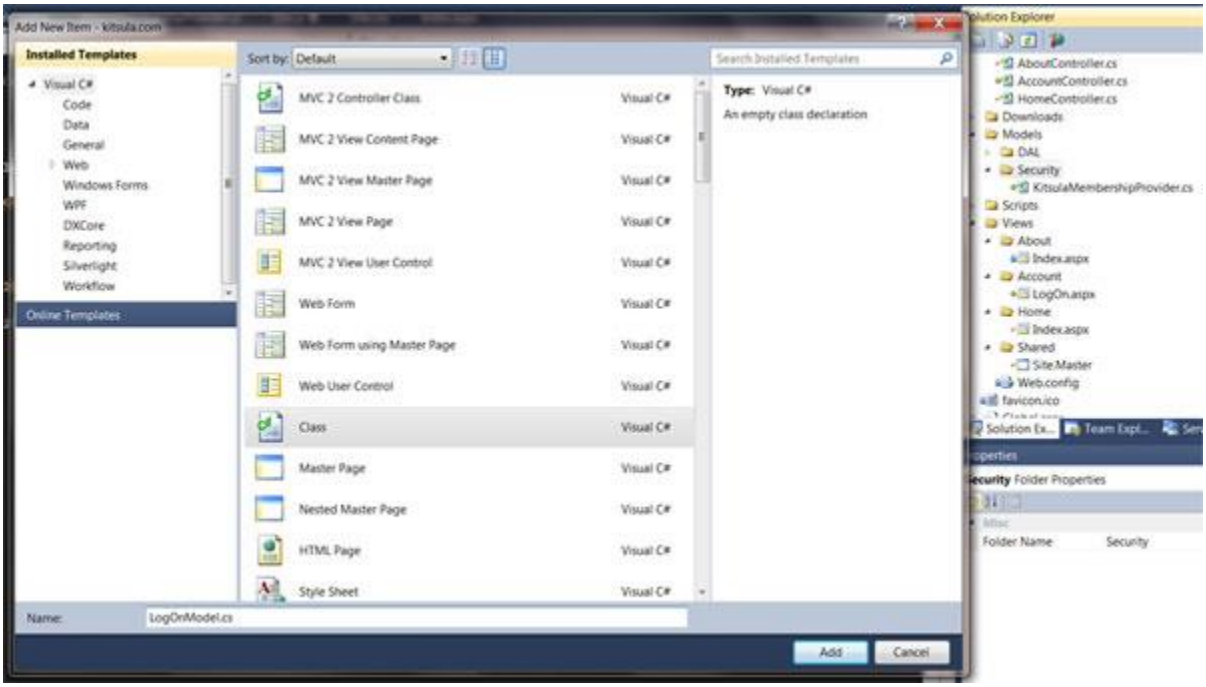
Of course here you can query DB for checking user info. Then we need configure solution to use created membership provider (web.config file):

```
<system.web>
    ...
    <authentication mode="Forms">
      <forms loginUrl="~/Account/LogOn" timeout="2880" />
    </authentication>
    <membership defaultProvider="KitsulaMembershipProviter">
      <providers>
          <clear />
          <add name="KitsulaMembershipProviter"
type="Kitsula.Security.KitsulaMembershipProviter"
enablePasswordRetrieval="false" enablePasswordReset="true"
requiresQuestionAndAnswer="false" requiresUniqueEmail="false"
maxInvalidPasswordAttempts="5" minRequiredPasswordLength="5"
minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10"
applicationName="/" />
      </providers>
    </membership>
    ...
</system.web>
```

We set settings redirect to ~/Account/LogOn page if user is trying open protected page. Let’s create this login page.

Controller:



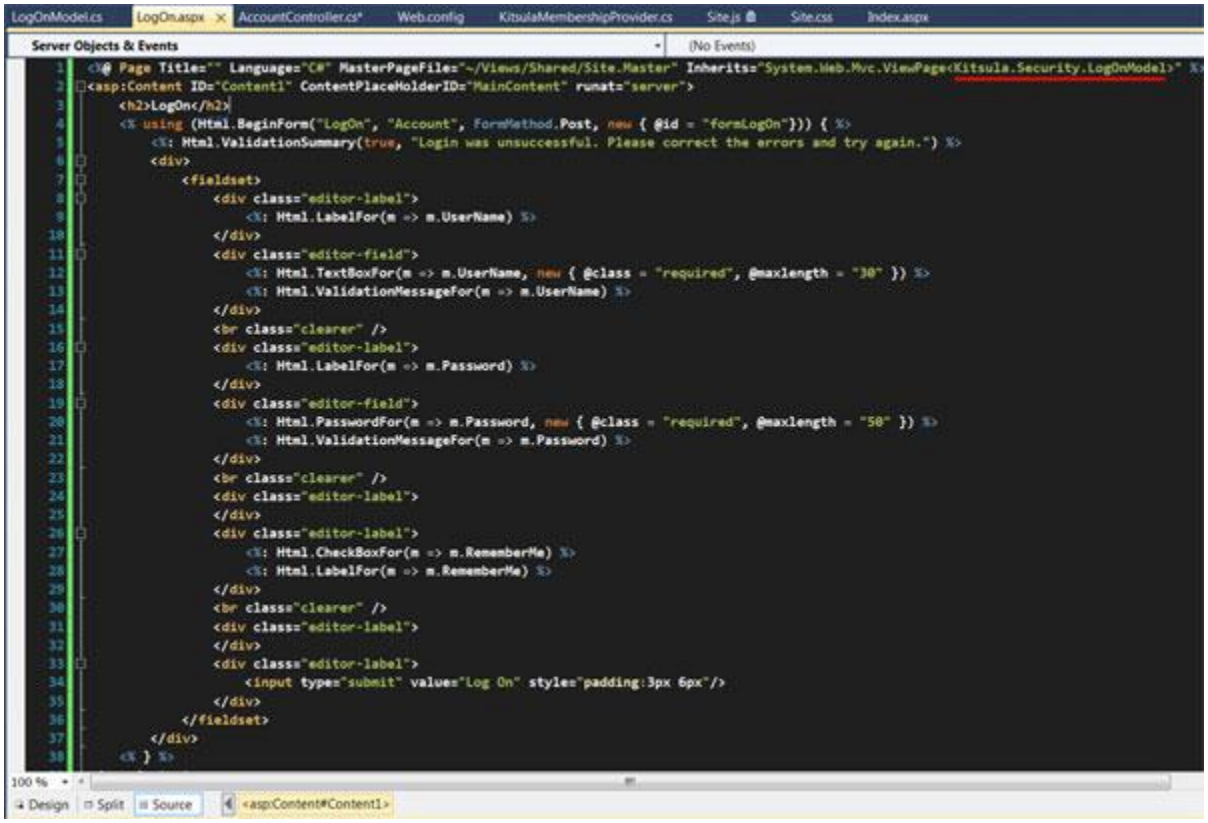


Model:

```
/// LogOnModel.cs
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel;

namespace Kitsula.Security
{
```

```
    public class LogOnModel
    {
        [Required]
        [DisplayName("User name")]
        public string UserName { get; set; }

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

        [DisplayName("Remember me?")]
        public bool RememberMe { get; set; }
    }
}
```

View:



Back to the controller handle post command from view:

```
/// AccountController.cs
using System;
using System.Web.Mvc;
using Kitsula.Security;
using System.Web.Security;
```

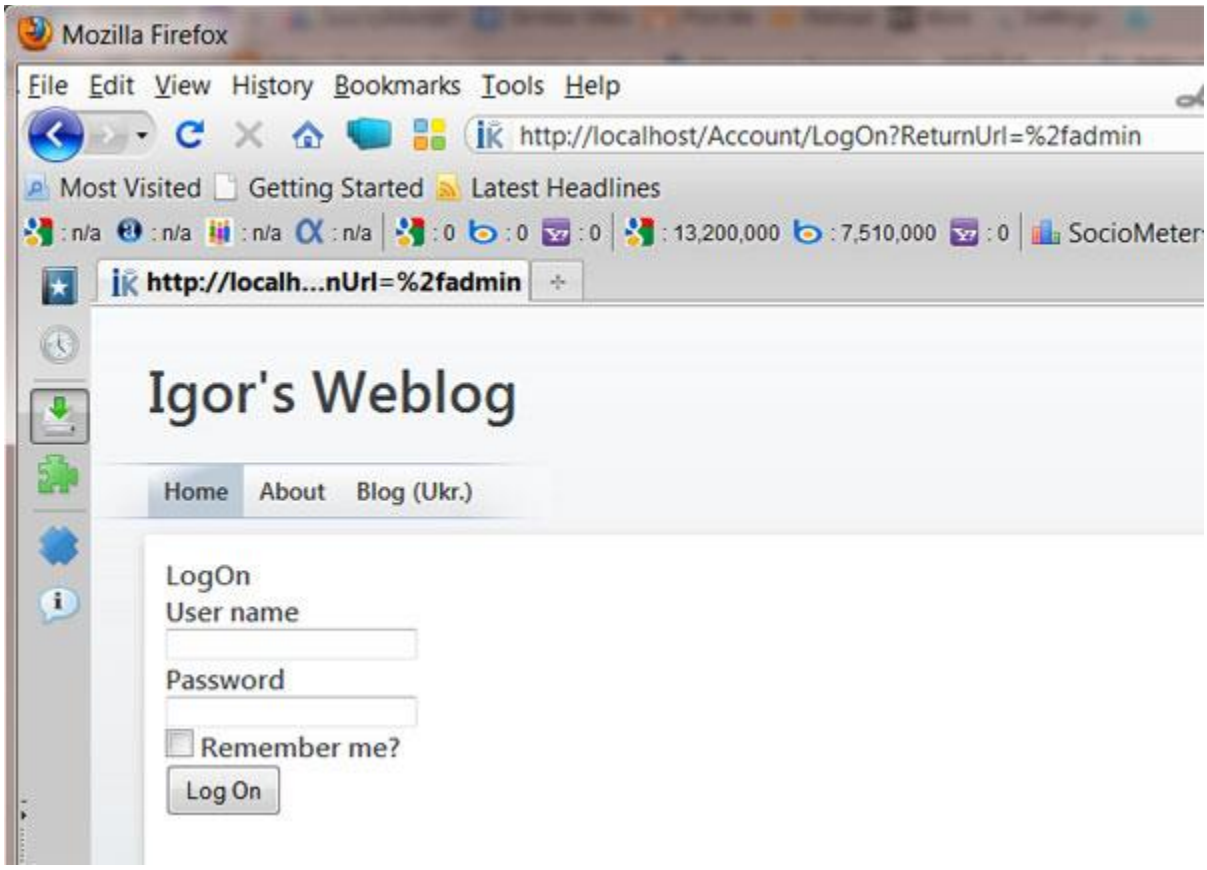
```
namespace Kitsula.Controllers
{
    public class AccountController : Controller
    {
        //
        // GET: /Account/LogOn

        public ActionResult LogOn()
        {
            return View();
        }

        [HttpPost]
        public ActionResult LogOn(LogOnModel model, string returnUrl)
        {
            if (ModelState.IsValid)
            {
                if (new
KitsulaMembershipProvider().ValidateUser(model.UserName, model.Password))
                {
                    FormsAuthentication.SetAuthCookie(model.UserName,
model.RememberMe);
                    if (!String.IsNullOrEmpty(returnUrl))
                    {
                        return Redirect(returnUrl);
                    }
                    else
                    {
                        return RedirectToAction("Index", "Home");
                    }
                }
                else
                {
                    ModelState.AddModelError("", "The user name or password
provided is incorrect.");
                }
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

    }
}
```

Done. We create Custom Membership Provider. Instead of protect folders or files (like it is in Web Forms) in MVC model you can protect any controller by

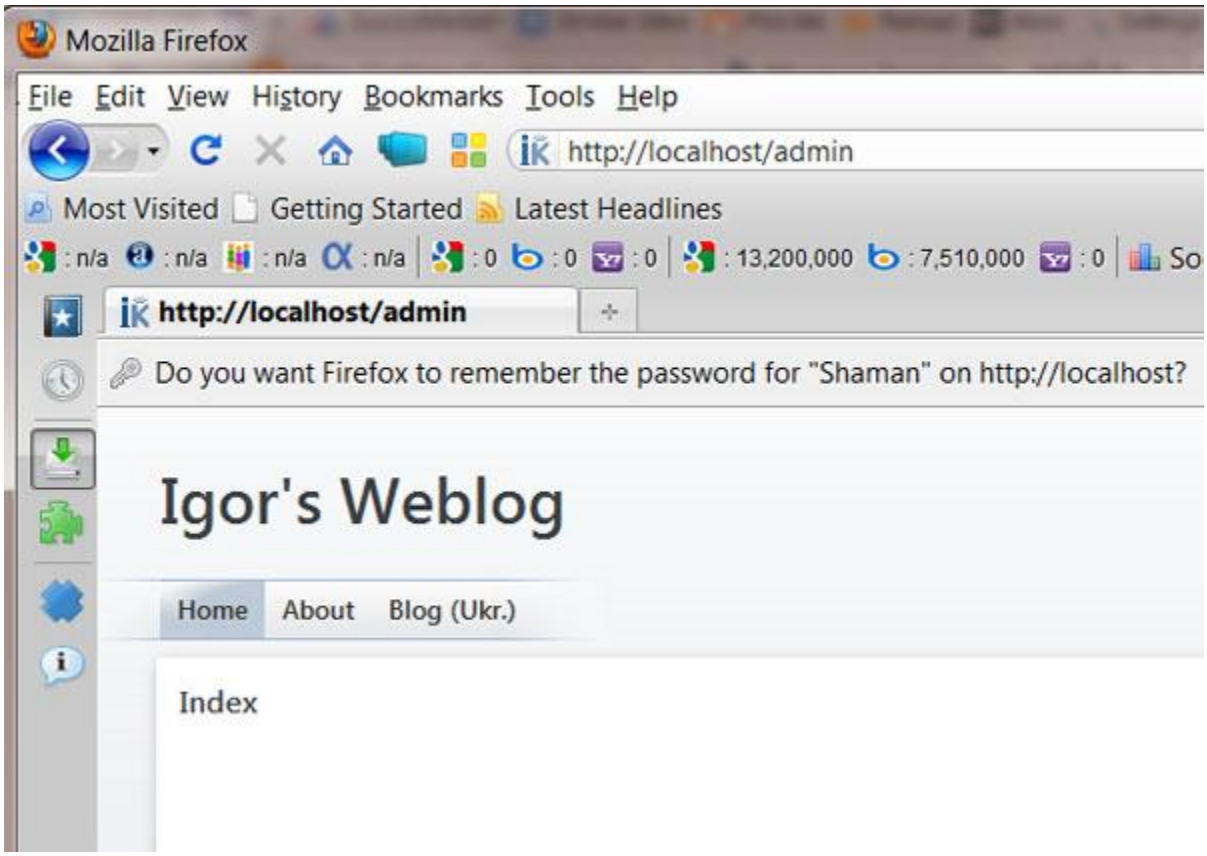
putting [Authorize()] attribute to the controller class definition (in my case I put this attribute for all controllers in Admin area):

```
using System;
using System.Web.Mvc;

namespace Kitsula.Areas.Admin.Controllers
{
    [Authorize()]
    public class HomeController : Controller
    {
        //
        // GET: /Admin/Home/

        public ActionResult Index()
        {
            return View();
        }

    }
}
```

Let’s test and open default page in Admin area http://localhost/Admin. We should be redirected to login page (as expected):



After login, we will be redirected to protected page:

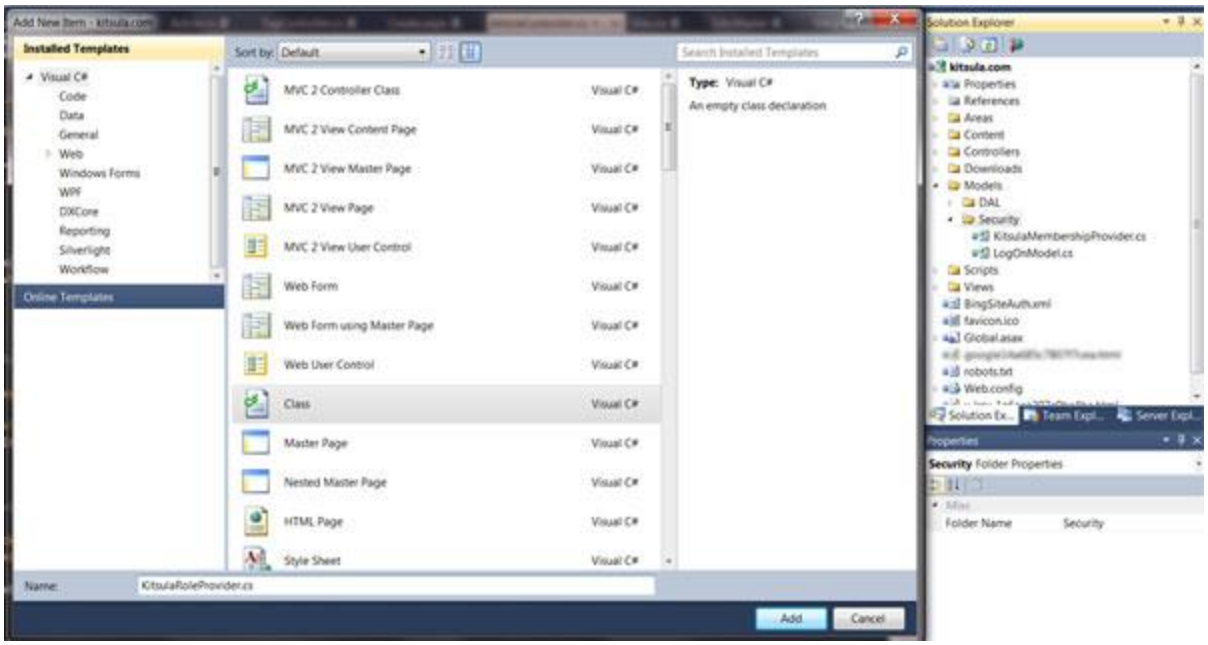


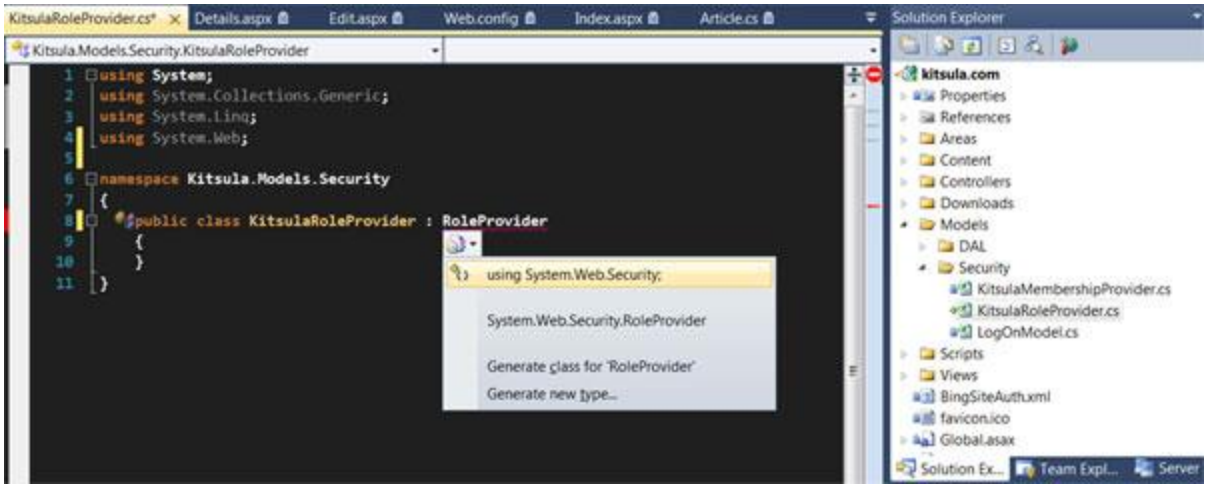
Sources:

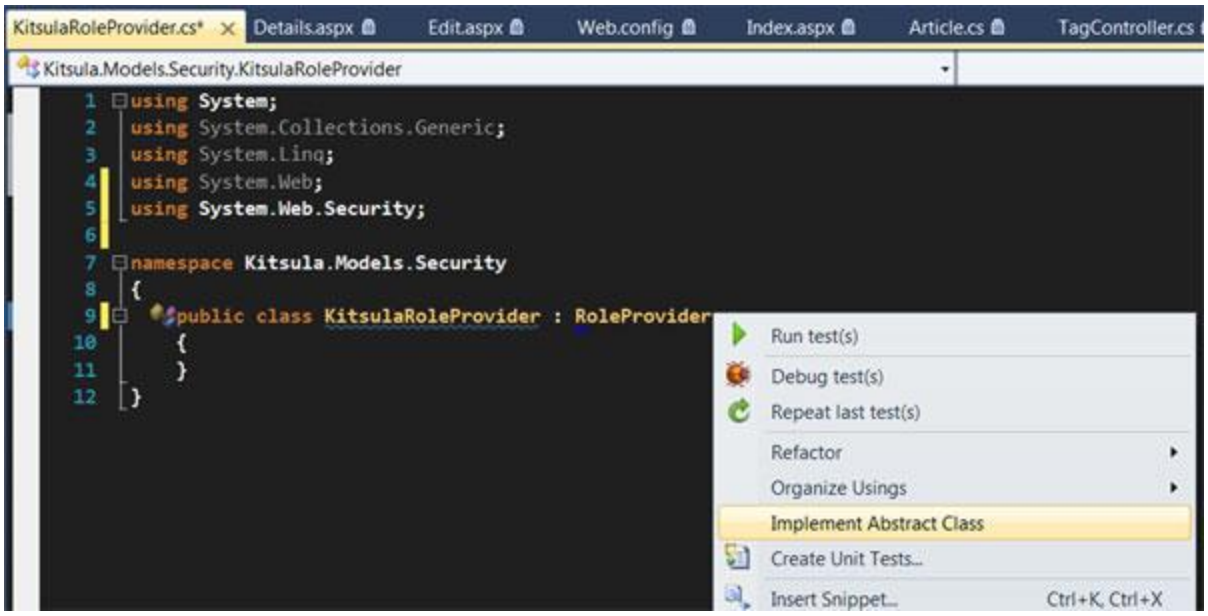
Custom Role Provider for MVC

In previous article I explain how to create Custom Membership Provider to authorize user and protect controls and pages. But what if you want to show or protect some area, controller or page for specific group of users? For example allow access to Admin Panel only for admins.

In .Net Framework for this purpose is Role Provider. But again, it uses own DB for store user roles. So let's create and configure Custom Role Provider which will use our DB or any other storage. As before we should overwrite class from .NET:


Add New Item - kitsula.com
Installed Templates
Visual C#
Code
Data
General
Web
Windows Forms
WPF
DXCore
Reporting
Silverlight
Workflow
Online Templates
Sort by: Default
Search Installed Templates
MVC 2 Controller Class
MVC 2 View Content Page
MVC 2 View Master Page
MVC 2 View Page
MVC 2 View User Control
Web Form
Web Form using Master Page
Web User Control
Class
Master Page
Nested Master Page
HTML Page
Style Sheet
Visual C#
Type: Visual C#
An empty class declaration
Name: KitsulaRoleProvider.cs
Add
Cancel
Solution Explorer
kitsula.com
Properties
References
Areas
Content
Controllers
Downloads
Models
DAL
Security
KitsulaMembershipProvider.cs
LogOnModel.cs
Scripts
Views
BingSiteAuth.xml
favicon.ico
Global.asax
robots.txt
Web.config
Properties
Security Folder Properties
Folder Name
Security


KitsulaRoleProvider.cs*
Details.aspx
Edit.aspx
Web.config
Index.aspx
Article.cs
Kitsula.Models.Security.KitsulaRoleProvider
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace Kitsula.Models.Security
{
public class KitsulaRoleProvider : RoleProvider
{
}
}
using System.Web.Security;
System.Web.Security.RoleProvider
Generate class for 'RoleProvider'
Generate new type...
Solution Explorer
kitsula.com
Properties
References
Areas
Content
Controllers
Downloads
Models
DAL
Security
KitsulaMembershipProvider.cs
KitsulaRoleProvider.cs
LogOnModel.cs
Scripts
Views
BingSiteAuth.xml
favicon.ico
Global.asax
Solution Ex...
Team Expl...
Server



For the minimum functionality we need implement and overwrite two functions GetRolesForUser and IsUserInRole. First one is used to gel list of all user roles (or groups):

```
public override string[] GetRolesForUser(string username)
        {
            using (DatabaseEntities db = new DatabaseEntities())
            {
                User user = db.Users.FirstOrDefault(u =>
u.UserName.Equals(username, StringComparison.CurrentCultureIgnoreCase) ||
u.Email.Equals(username, StringComparison.CurrentCultureIgnoreCase));

                var roles = from ur in user.UserRoles
                            from r in db.Roles
                            where ur.RoleId == r.Id
                            select r.Name;
                if (roles != null)
                    return roles.ToArray();
                else
                    return new string[] {}; ;
            }
        }
```

As you can see I locate user in my DB by username parameter of the function (in my case it’s can be user name or email) and create the string list of user roles.

Second function is to check if user in the role (or group):

```
public override bool IsUserInRole(string username, string roleName)
        {
            using (DatabaseEntities db = new DatabaseEntities())
            {
                User user = db.Users.FirstOrDefault(u =>
u.UserName.Equals(username, StringComparison.CurrentCultureIgnoreCase) ||
u.Email.Equals(username, StringComparison.CurrentCultureIgnoreCase));

                var roles = from ur in user.UserRoles
                            from r in db.Roles
                            where ur.RoleId == r.Id
                            select r.Name;
                if (user != null)
                    return roles.Any(r => r.Equals(roleName,
StringComparison.CurrentCultureIgnoreCase));
                else
                    return false;
            }
        }
```

Then we need to configure in web.config file solution to use created role provider:

```
<system.web>
    ...
    <rolemanager cacherolesincookie="true"
defaultprovider="KitsulaRoleProvider" enabled="true">
        <providers>
            <clear />
            <add name="KitsulaRoleProvider"
type="Kitsula.Security.KitsulaRoleProvider" />
        </providers>
    </rolemanager>
    ...
</system.web>
```

Now you can protect controllers, actions, pages for specific group of user which are in specified roles by set Authorize attribute:

```
using System;
using System.Web.Mvc;

namespace Kitsula.Areas.Admin.Controllers
{
```

```
    [Authorize(Roles = "Administrators")]
    public class HomeController : Controller
    {
        //
        // GET: /Admin/Home/

        public ActionResult Index()
        {
            return View();
        }

    }
}
```