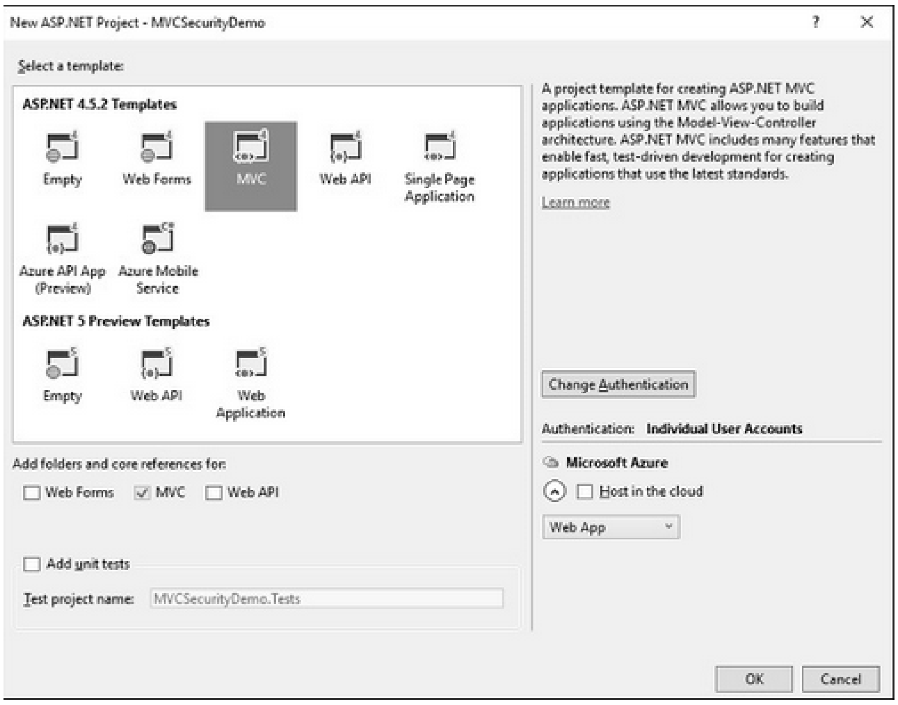
ASP.NET MVC – Security

Authentication

Authentication of user means verifying the identity of the user. This is really important. You might need to present your application only to the authenticated users for obvious reasons.

1. Create a new ASP.NET Web Application -> Select the Select MVC template and you will see that the Change Authentication button is now enabled.



A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Authentication: Individual User Accounts

Add folders and core references for: ☐ Web Forms ☑ MVC ☐ Web API

Test project name: MVCSecurityDemo.Tests

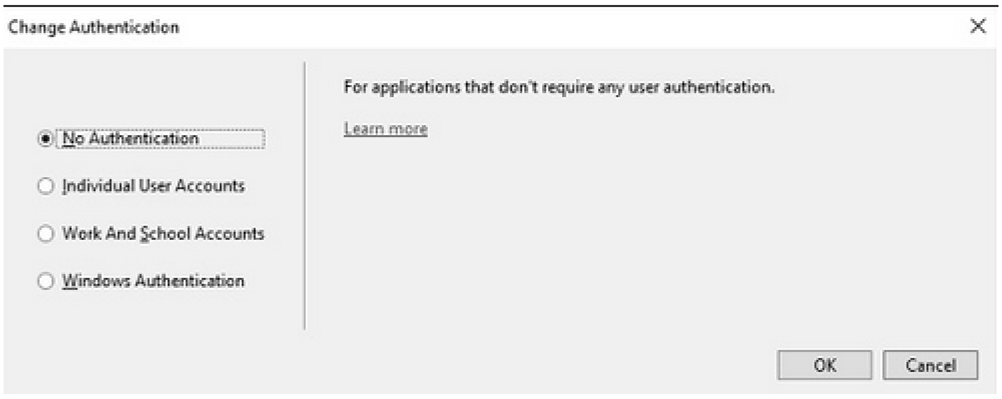
2. Keep the default Authentication – Individual User Accounts -> OK

Authentication Options

When you click the Change button, you will see a dialog with four options, which are as follows.

No Authentication

The first option is No Authentication and this option is used when you want to build a website that doesn't care who the visitors are.


Change Authentication

(●) No Authentication
( ) Individual User Accounts
( ) Work And School Accounts
( ) Windows Authentication

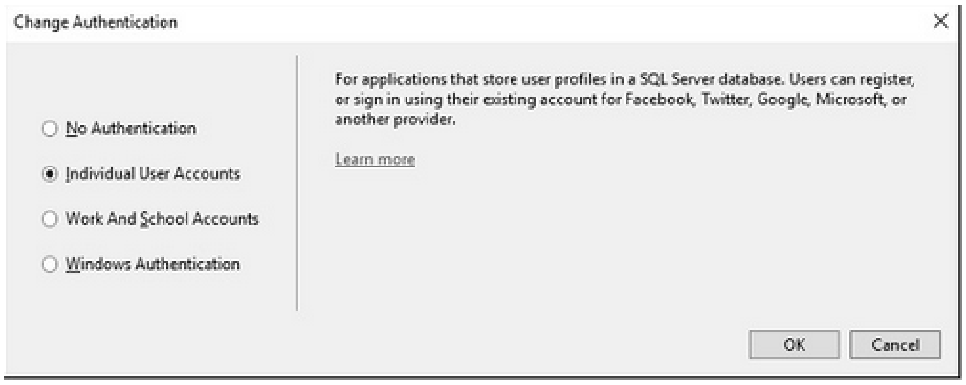
For applications that don't require any user authentication.

Learn more

It is open to anyone and every person connects as every single page. You can always change that later, but the No Authentication option means there will not be any features to identify users coming to the website.

Individual User Accounts

The second option is Individual User Accounts and this is the traditional forms-based authentication where users can visit a website. They can register, create a login, and by default their username is stored in a SQL Server database using some new ASP.NET identity features, which we'll look at.


Change Authentication

( ) No Authentication
(●) Individual User Accounts
( ) Work And School Accounts
( ) Windows Authentication

For applications that store user profiles in a SQL Server database. Users can register, or sign in using their existing account for Facebook, Twitter, Google, Microsoft, or another provider.

Learn more

The password is also stored in the database, but it is hashed first. Since the password is hashed, you don't have to worry about plain-text passwords sitting in a database.

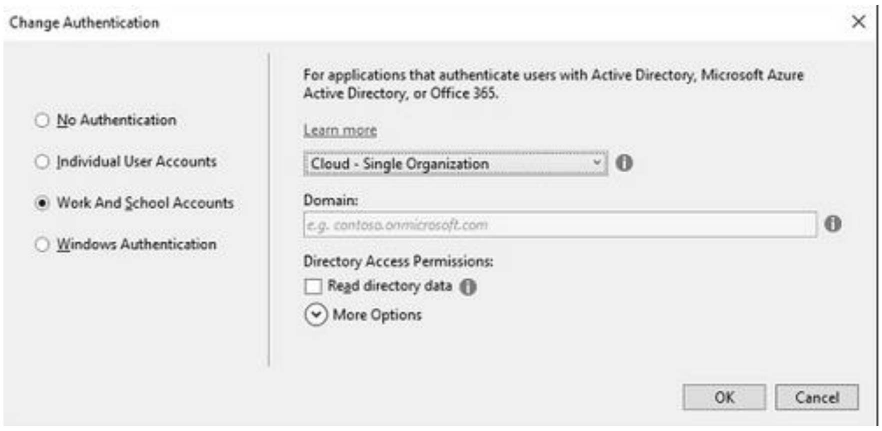
This option is typically used for internet sites where you want to establish the identity of a user. In addition to letting a user create a local login with a password for your site, you can also enable logins from third parties like Microsoft, Google, Facebook, and Twitter.

This allows a user to log into your site using their Live account or their Twitter account and they can select a local username, but you don't need to store any passwords.

This is the option that we'll spend some time with in this module; the individual user accounts option.

Work and School Accounts

The third option is to use organizational accounts and this is typically used for business applications where you will be using active directory federation services.


Change Authentication

No Authentication
Individual User Accounts
[✓ Work And School Accounts]
Windows Authentication

For applications that authenticate users with Active Directory, Microsoft Azure Active Directory, or Office 365.

Learn more

Cloud - Single Organization

Domain: e.g. contoso.onmicrosoft.com

Directory Access Permissions:
Read directory data

More Options

OK Cancel

You will either set up Office 365 or use Azure Active Directory Services, and you have a single sign-on for internal apps and Cloud apps.

You will also need to provide an app ID so your app will need to be registered with the Windows Azure management portal if this is Azure based, and the app ID will uniquely identify this application amongst all the applications that might be registered.

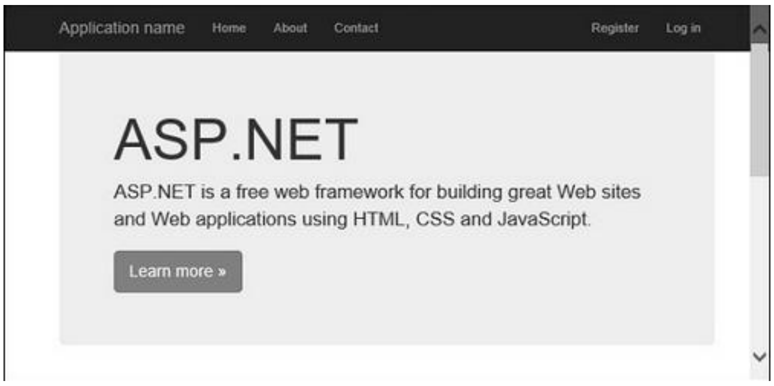
Windows Authentication

The fourth option is Windows authentication, which works well for intranet applications.

A user logs into Windows desktop and can launch a browser to the application that sits inside the same firewall. ASP.NET can automatically pick up the user's identity, the one that was established by active directory. This option does not allow any anonymous access to the site, but again that is a configuration setting that can be changed.

Let's take a look into the forms-based authentication, the one that goes by the name, Individual User Accounts. This application will store usernames and passwords, old passwords in a local SQL Server database, and when this project is created, Visual Studio will also add NuGet packages.

3. Now run this application and when you first come to this application you will be an anonymous user.


Application name Home About Contact Register Log in

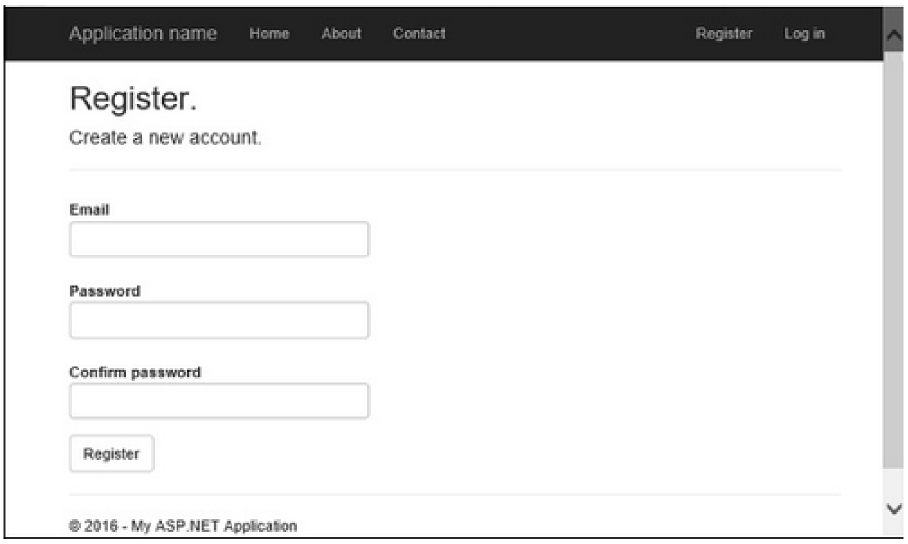
ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

Learn more »

You won't have an account that you can log into yet so you will need to register on this site.

4. Click on the Register link and you will see the following view.



Register.

Create a new account.

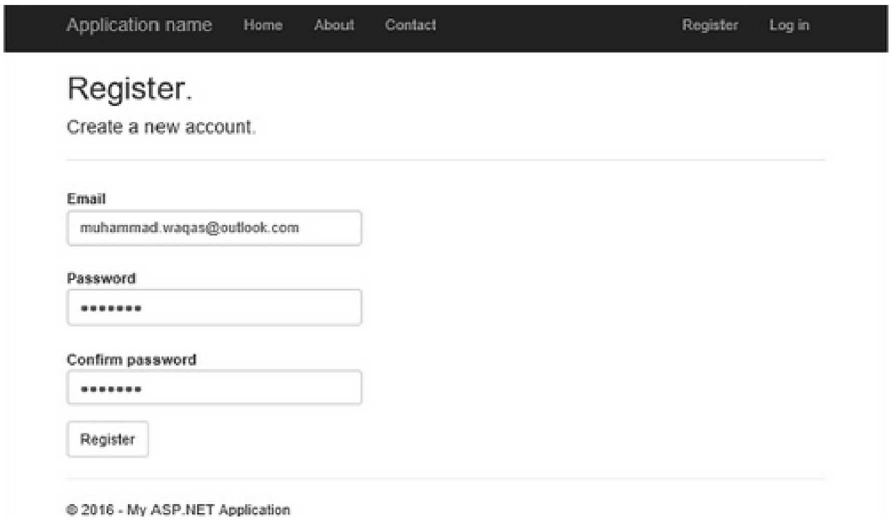
Email

Password

Confirm password

© 2016 - My ASP.NET Application

5. Enter your email and password and click on Register.



Register.

Create a new account.

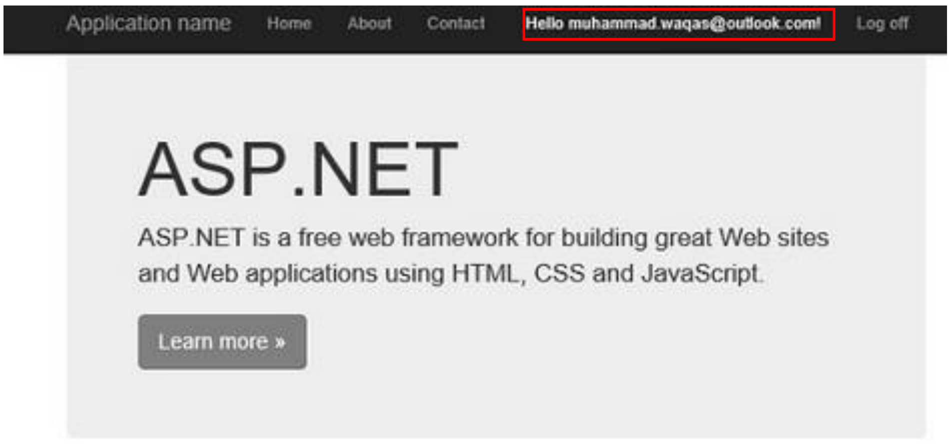
Email: muhammad.waqas@outlook.com

Password: •••••••

Confirm password: •••••••

© 2016 - My ASP.NET Application

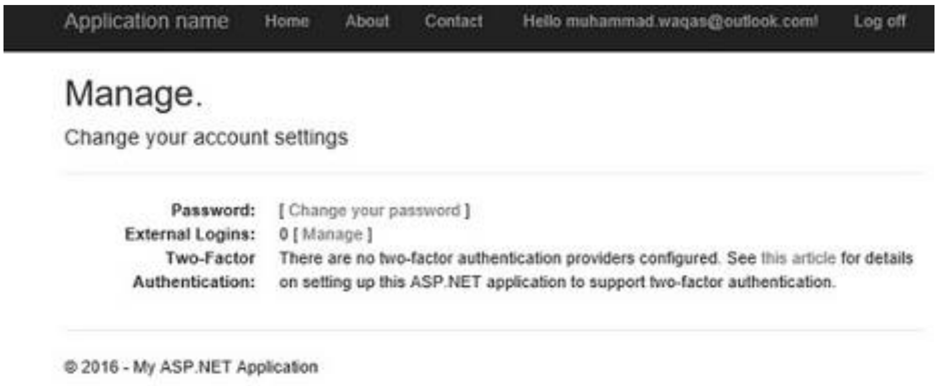
The application will register your credentials on top of the page (see image below)



ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

6. Click on your user name link on this form and it will navigate to the change password page (below)



Manage.

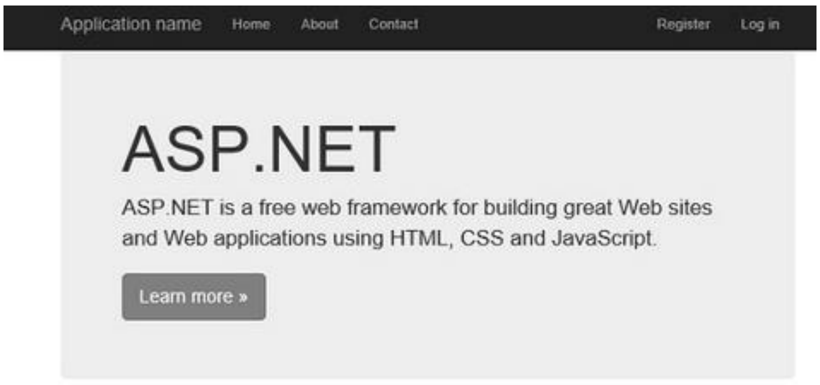
Change your account settings

Password:	[Change your password]
External Logins:	0 [Manage]
Two-Factor Authentication:	There are no two-factor authentication providers configured. See this article for details on setting up this ASP.NET application to support two-factor authentication.

© 2016 - My ASP.NET Application

You can also log off, shut down, reboot, come back a week later, and you should be able to log in with the credentials that you used earlier.

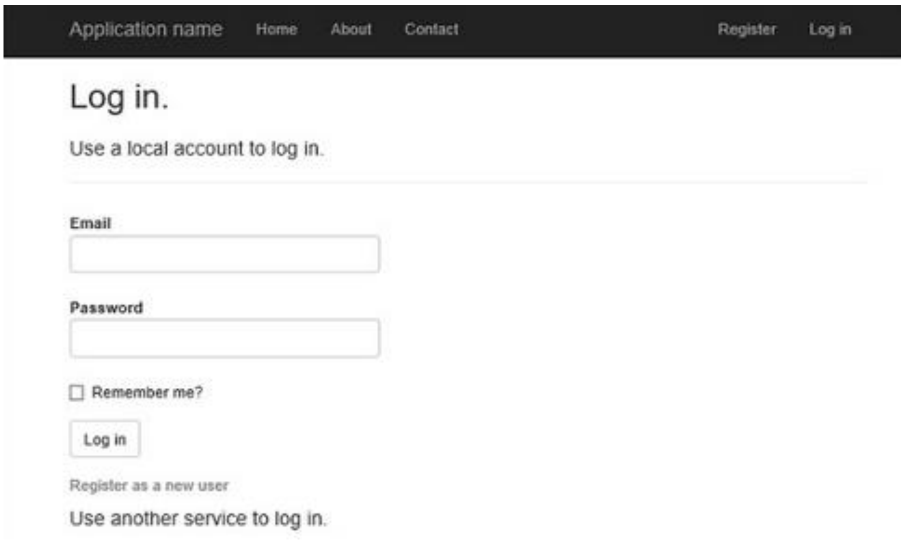
7. Now click on the log off button and it will display the following page.



ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

8. Click again on the Log in link and you will go to the following page



Log in.

Use a local account to log in.

Email

Password

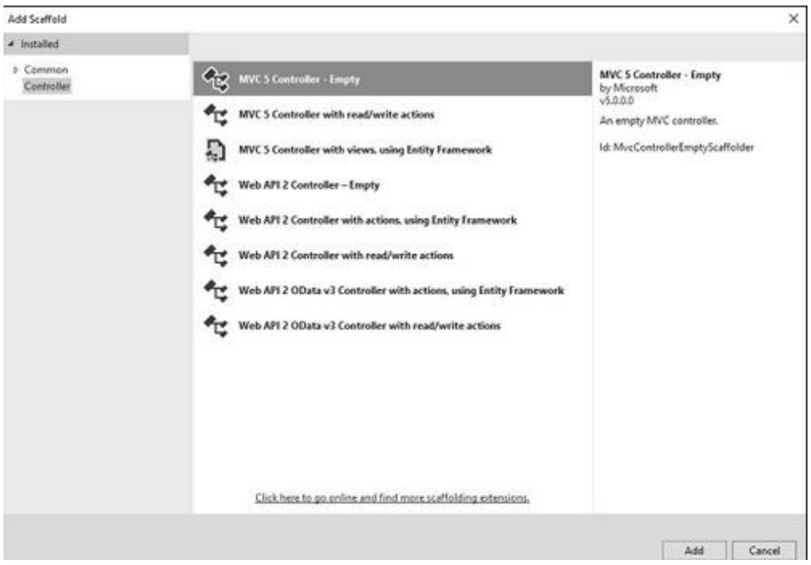
☐ Remember me?

Use another service to log in.

Authorization

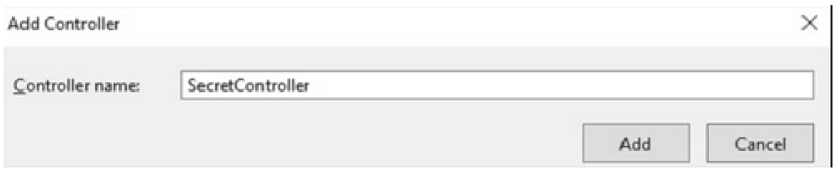
Let's suppose that we have some sort of information which we want to protect from unauthenticated users. So let's create a new controller to display that information, but only when a user is logged in.

5. Right-click on the controller folder and select Add → Controller.


Add Scaffold
Installed
Common
Controller
MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData v3 Controller with actions, using Entity Framework
Web API 2 OData v3 Controller with read/write actions
Click here to go online and find more scaffolding extensions.
MVC 5 Controller - Empty
by Microsoft
v5.0.0.0
An empty MVC controller.
Id: MvcControllerEmptyScaffolder
Add
Cancel

6. Select an MVC 5 (or 4) controller - Empty controller and click 'Add'.

Enter the name SecretController and click 'Add' button. Make this Controller the startup one.


Add Controller

Controller name: SecretController

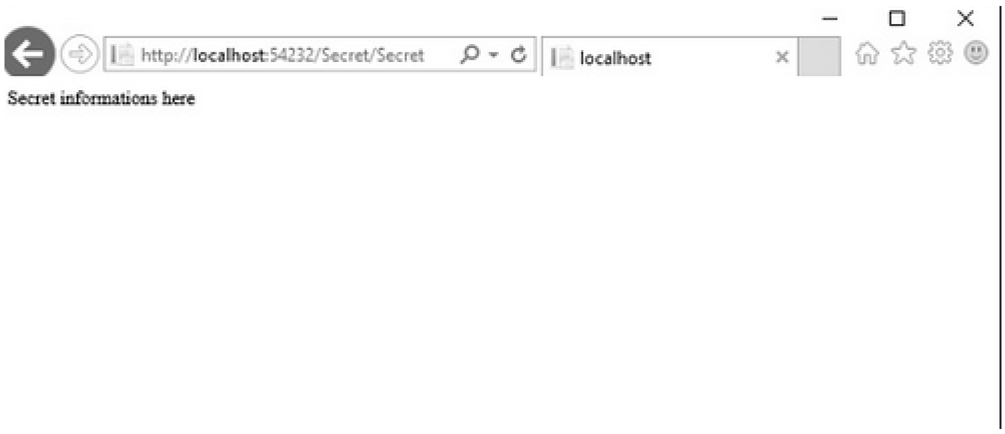
It will have two actions inside as shown in the following code

```
using System.Web.Mvc;

namespace MVCSecurityDemo.Controllers{
   public class SecretController : Controller{
      // GET: Secret
      public ContentResult Secret(){
         return Content("Secret informations here");
      }

      public ContentResult PublicInfo(){
         return Content("Public informations here");
      }
   }
}
```

7. When you run this application, you can access this information without any authentication as shown in the following screenshot.



Secret informations here

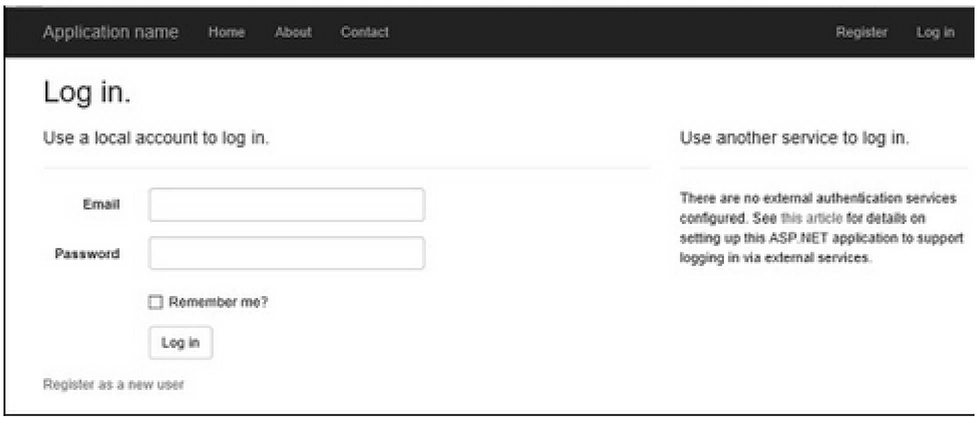
So only authenticated users should be able to get to Secret action method and the PublicInfo can be used by anyone without any authentication.

To protect this particular action and keep unauthenticated users from arriving here, you can use the Authorize attribute. The Authorize attribute without any other parameters will make sure that the identity of the user is known and they're not an anonymous user.

8. Add the [Authorize] attribute to the Secret action method as shown below:

```
// GET: Secret
[Authorize]
public ContentResult Secret(){
   return Content("Secret informations here");
}
```

9. Now run this application again. The MVC application will detect that you do not have access to that particular area of the application and it will redirect you automatically to the login page, where it will give you a chance to log in and try to get back to that area of the application where you were denied.



Log in.

Use a local account to log in.

Email

Password

Remember me?

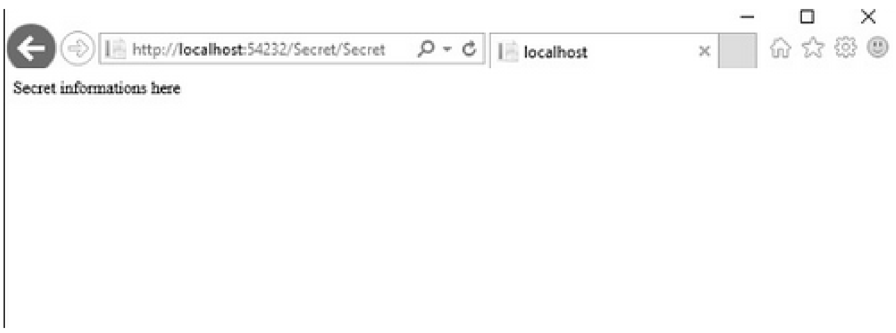
Log in

Register as a new user

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.

10. Enter your credentials and click ‘Log in’ button. You will see that it goes directly to that page.


http://localhost:54232/Secret/Secret

Secret informations here

If you come back to the home page and log off, you cannot get to the secret page. You will be asked again to log in, but if go to /Secret/PublicInfo, you can see that page, even when you are not authenticated.

So, when you don't want to be placing authorization on every action when you're inside a controller where pretty much everything requires authorization. In that case you can always apply this filter to the controller itself and now every action inside of this controller will require the user to be authenticated.

11. To apply the [Authorize] to the entire application, add it at the top of the controller as shown below:

```
using System.Web.Mvc;

namespace MVCSecurityDemo.Controllers{
   [Authorize]
   public class SecretController : Controller{
      // GET: Secret
      public ContentResult Secret(){
         return Content("Secret informations here");
      }

      public ContentResult PublicInfo(){
         return Content("Public informations here");
      }
   }
}
```

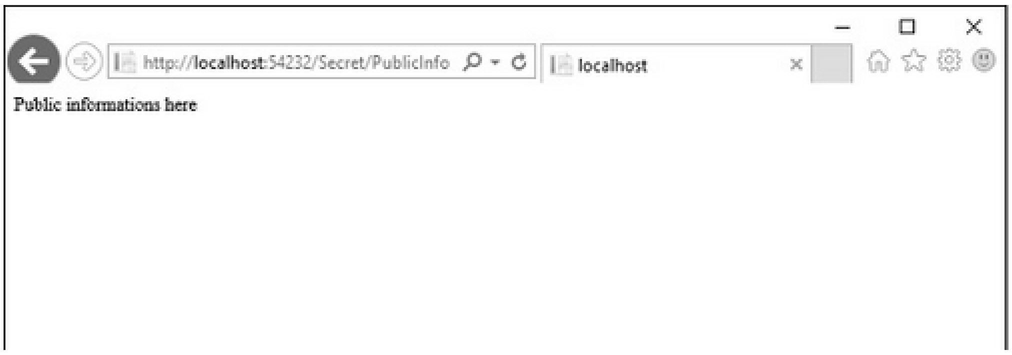
12. But if you really want any action to be open, you can come override this authorization rule with another attribute, which is, AllowAnonymous. (see below)

```
using System.Web.Mvc;

namespace MVCSecurityDemo.Controllers{
   [Authorize]
   public class SecretController : Controller{
      // GET: Secret
      public ContentResult Secret(){
         return Content("Secret informations here");
      }

      [AllowAnonymous]
      public ContentResult PublicInfo(){
         return Content("Public informations here");
      }
   }
}
```

13. Run this application and you can access the /Secret/PublicInfo with logging in but other action will require authentication.



Public informations here

With the Authorize attribute, you can also specify some parameters, like allow some specific users into this action.

14. To allow a specific user to access the controller use the User parameter as follows:

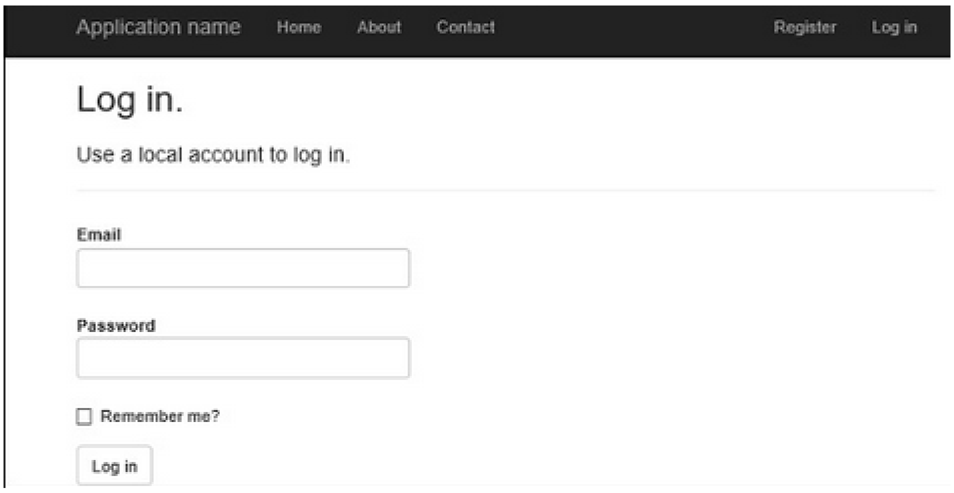
```
using System.Web.Mvc;

namespace MVCSecurityDemo.Controllers{
   [Authorize(Users =                      )]
   public class SecretController : Controller{
      // GET: Secret
      public ContentResult Secret(){
         return Content("Secret informations here");
      }

      [AllowAnonymous]
      public ContentResult PublicInfo(){
         return Content("Public informations here");
      }
   }
}
```

User Name Here

15. Run this application and go to /secret/secret, it will ask you to log as shown below:



Log in.

Use a local account to log in.

Email

Password

Remember me?