ASP. NET MVC Roles Demo

**Amit Ahlawat**

**CSE 686 – Internet Programming**

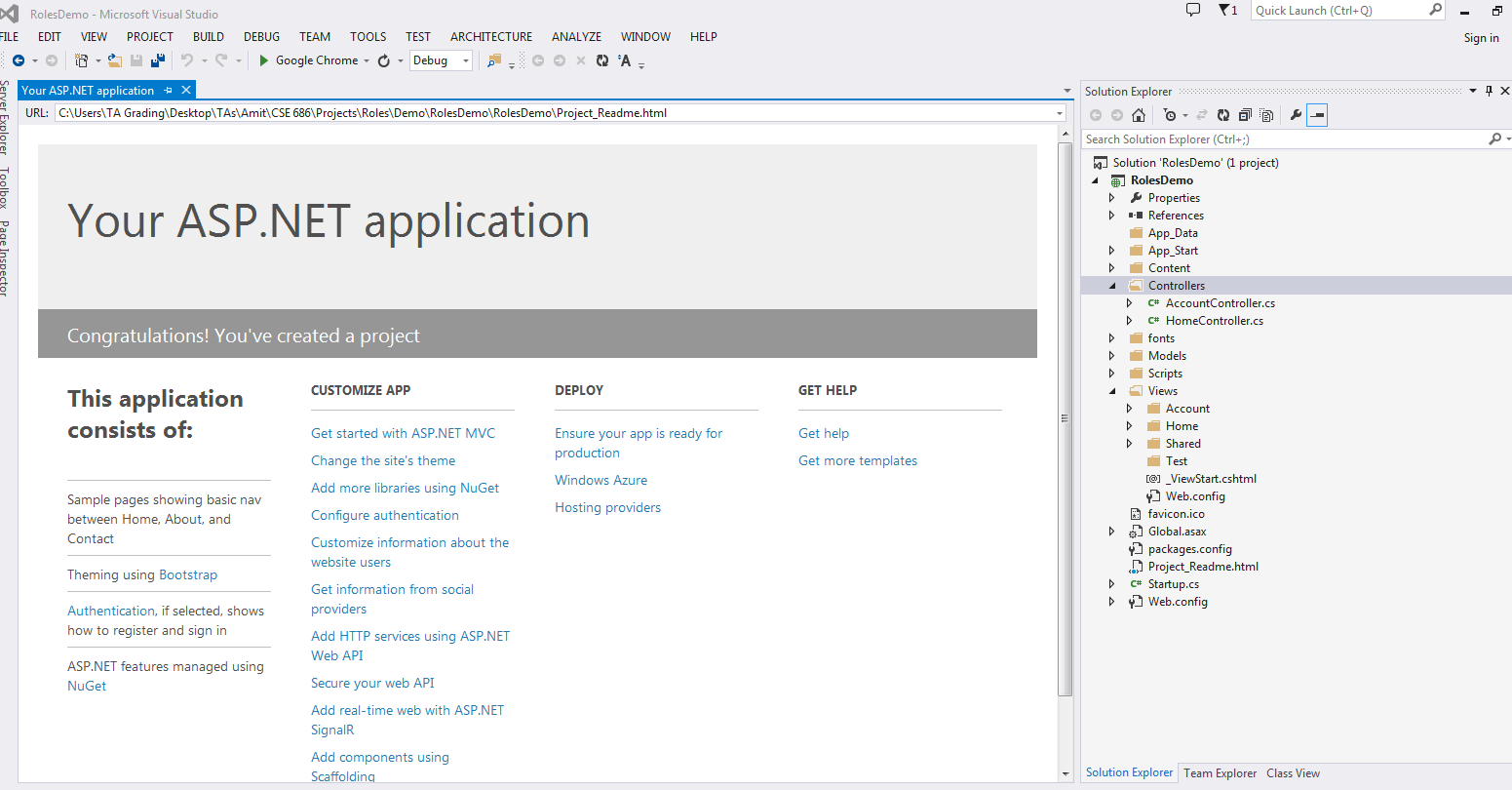
**Spring 2014**

Goals of document

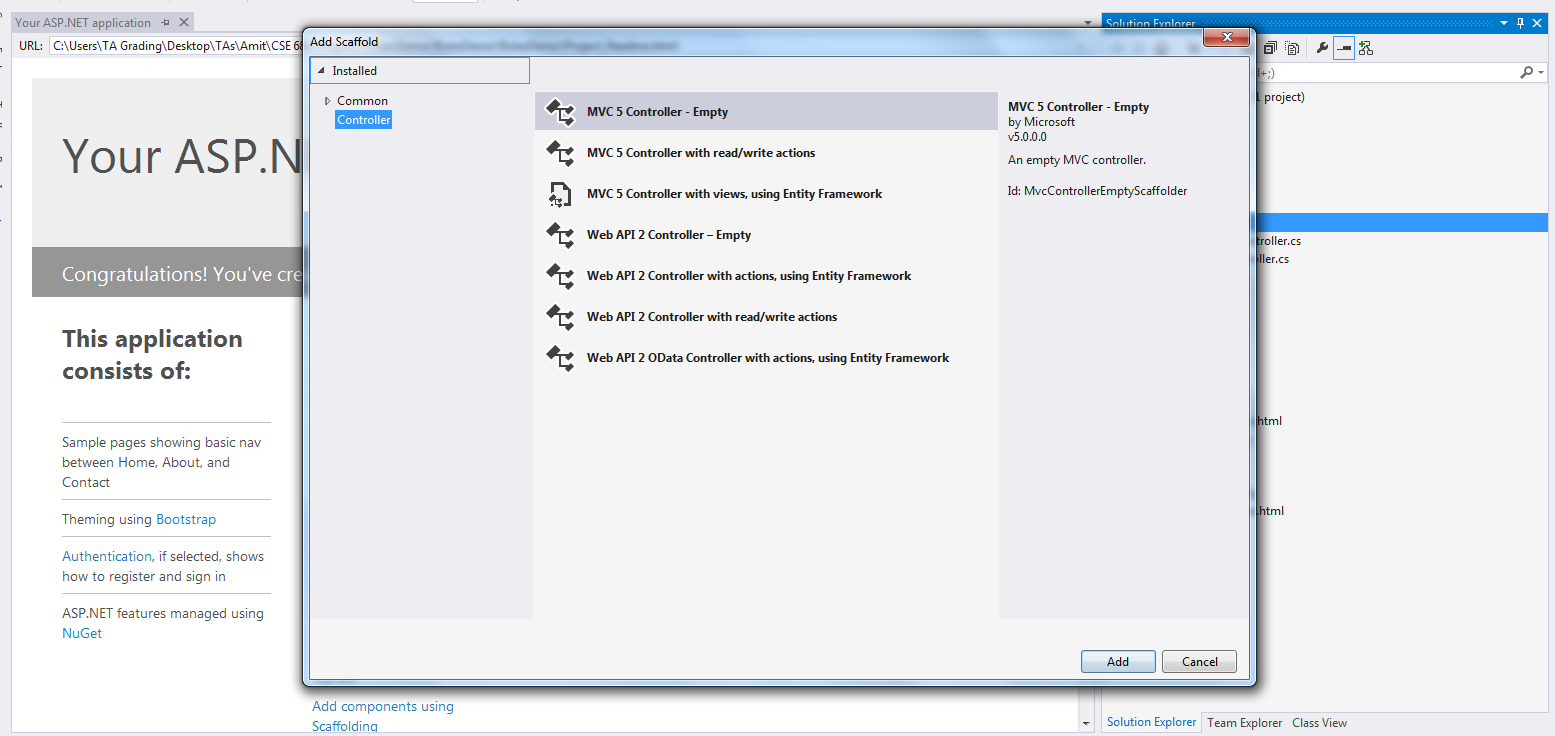
1. Show how to make a group of pages available only after registration with built-in authentication and registration mechanisms
2. Show to make specific pages available only to select users
3. Show to make specific pages available only to select roles

Using Built-in Registration and Authentication Mechanisms

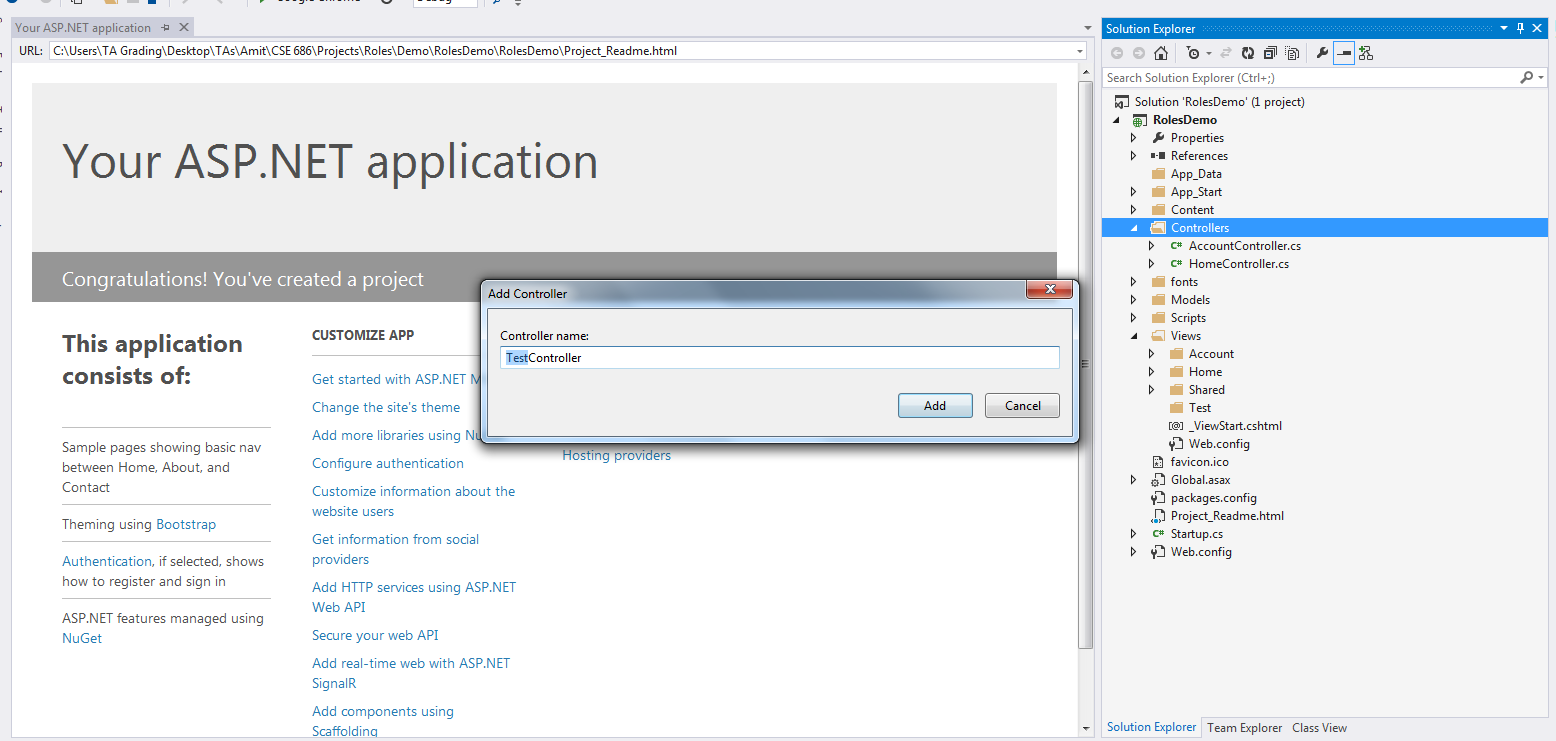
Create an ASP .NET MVC application.



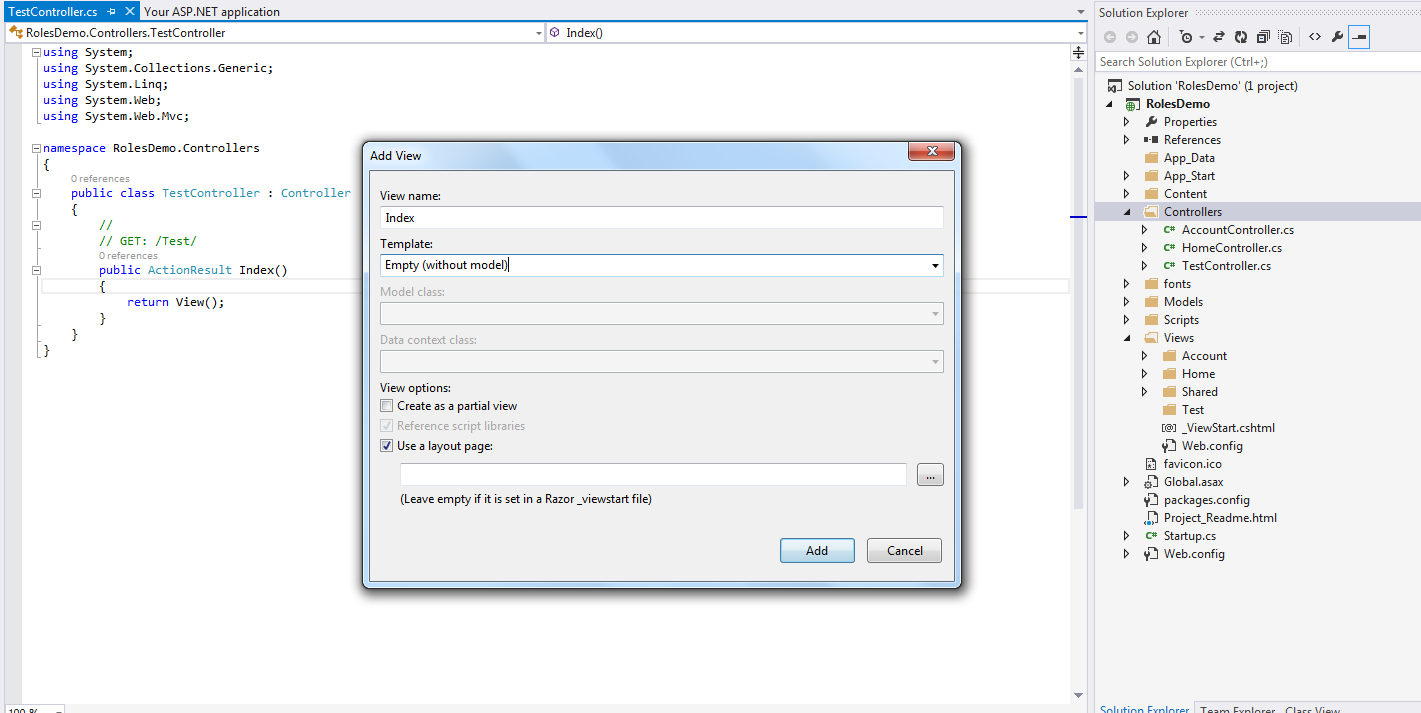
Add a new controller to the application.



Name the controller ‘TestController’.



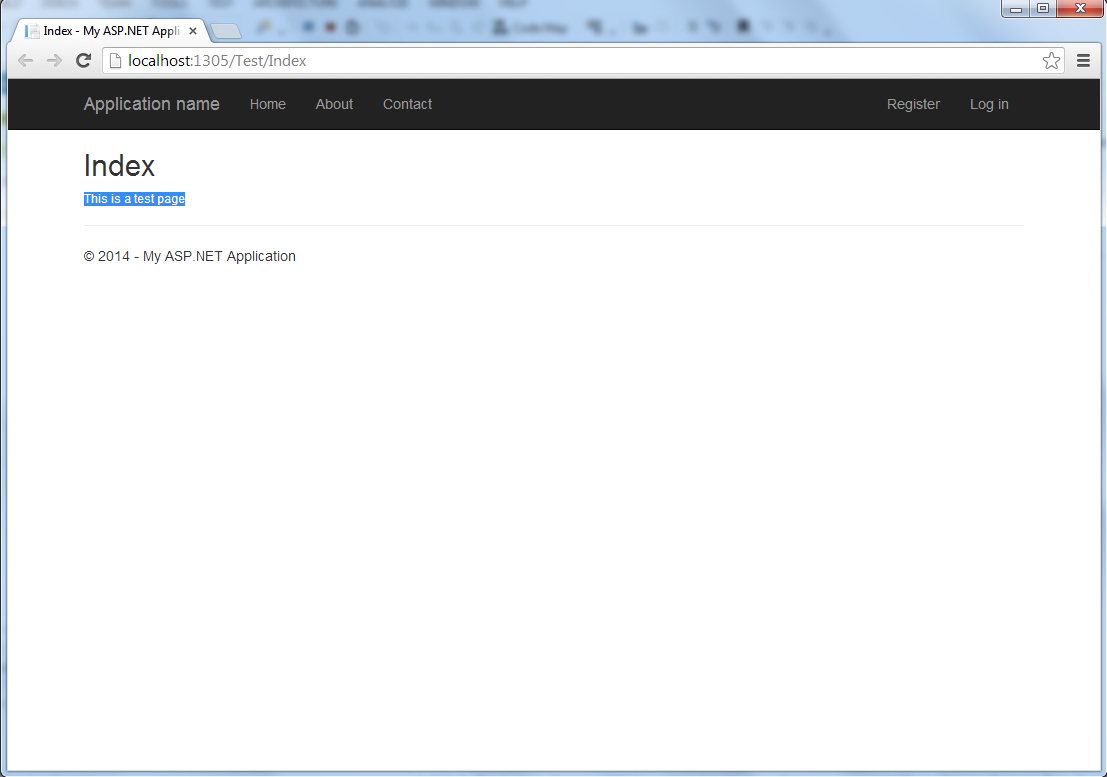
Add a view for the Index method of the TestController.



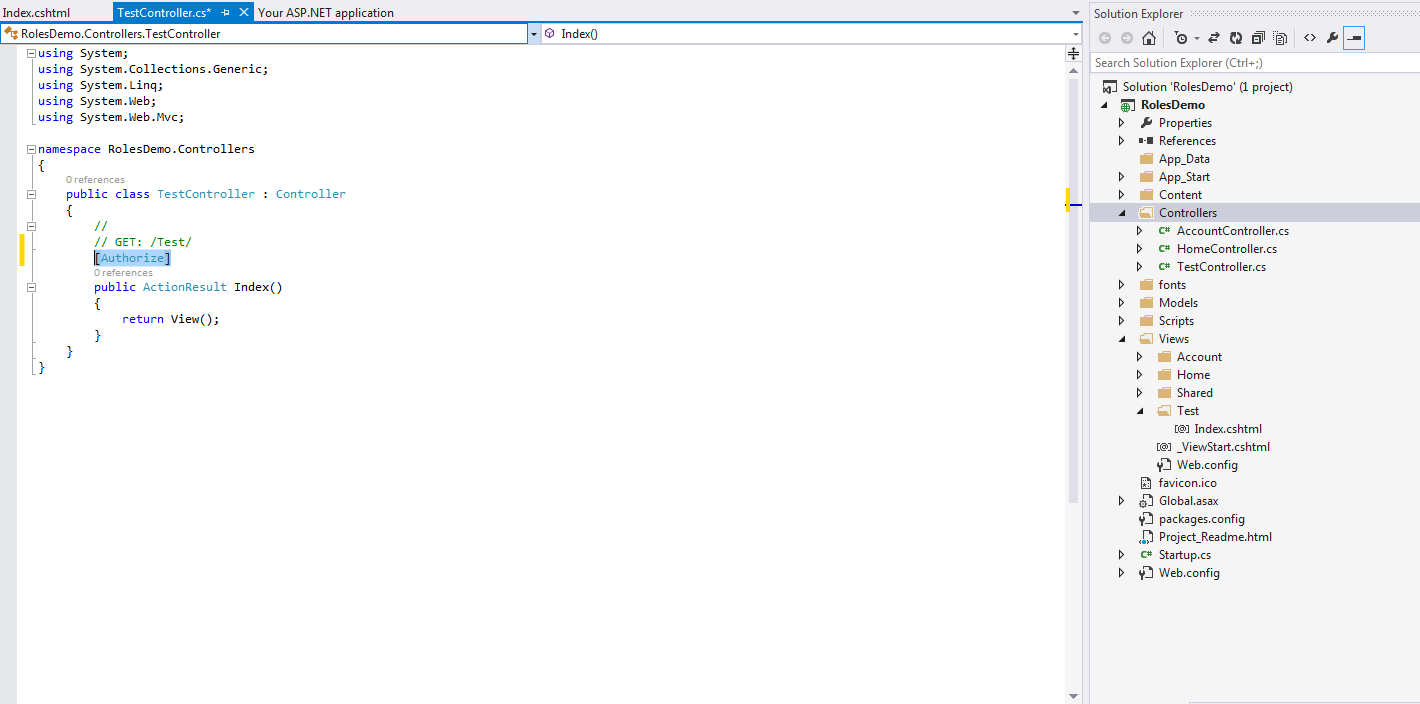
Here the view is shown. An additional heading has been added (This is a test page).



Run the application and navigate to /Test/Index as shown:

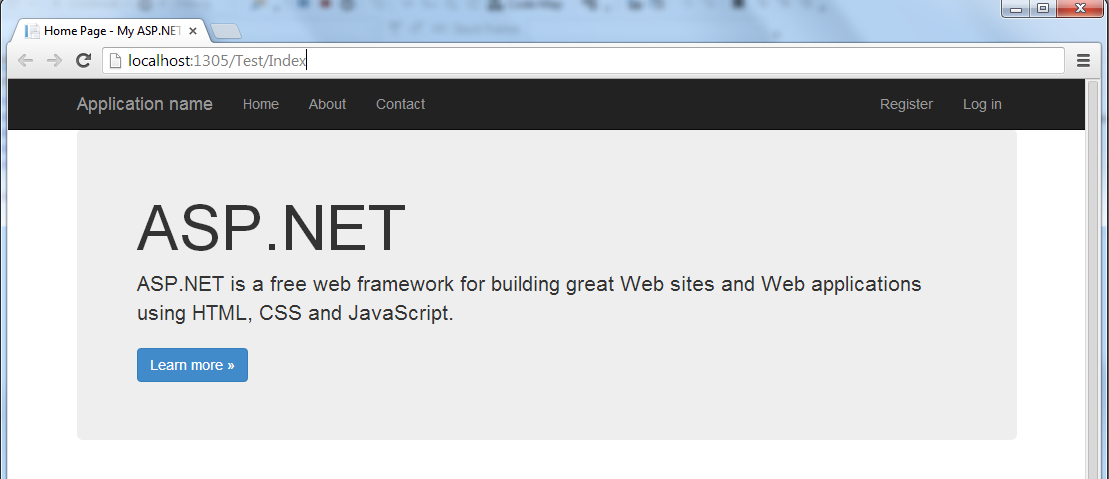


To add the requirement of user authentication to view a page, we will use the [Authorize] attribute as shown :

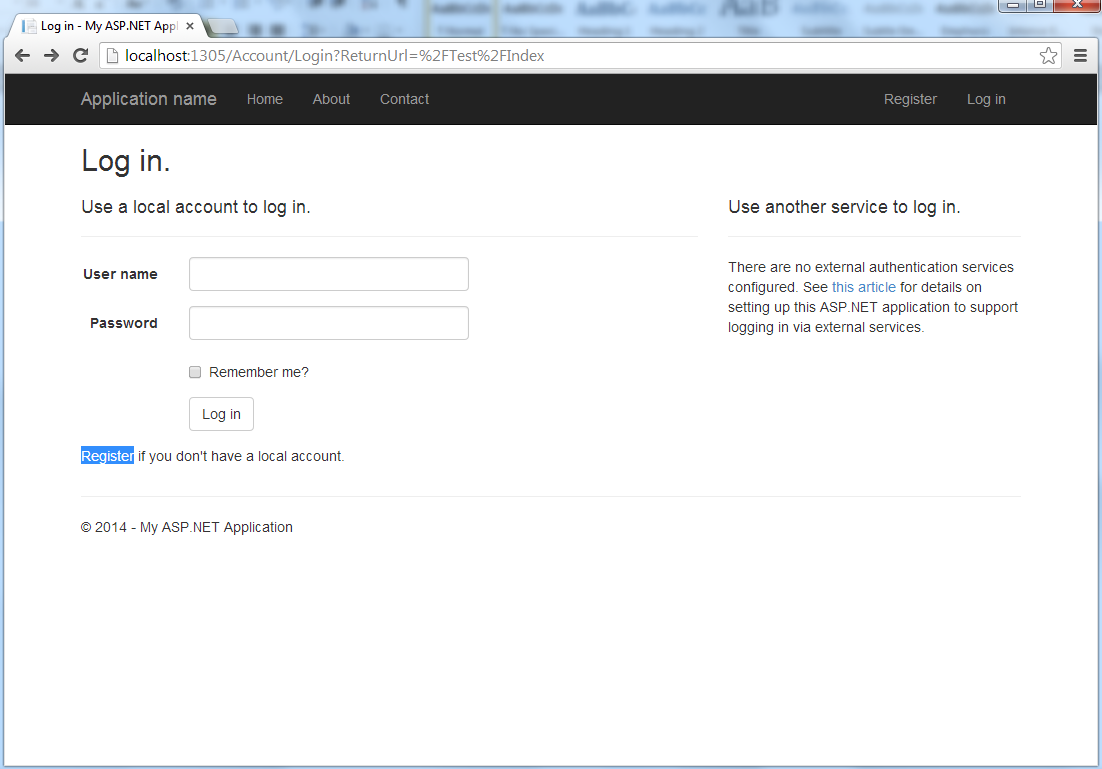


The above should not allow any user to view page /Test/Index unless logged in.

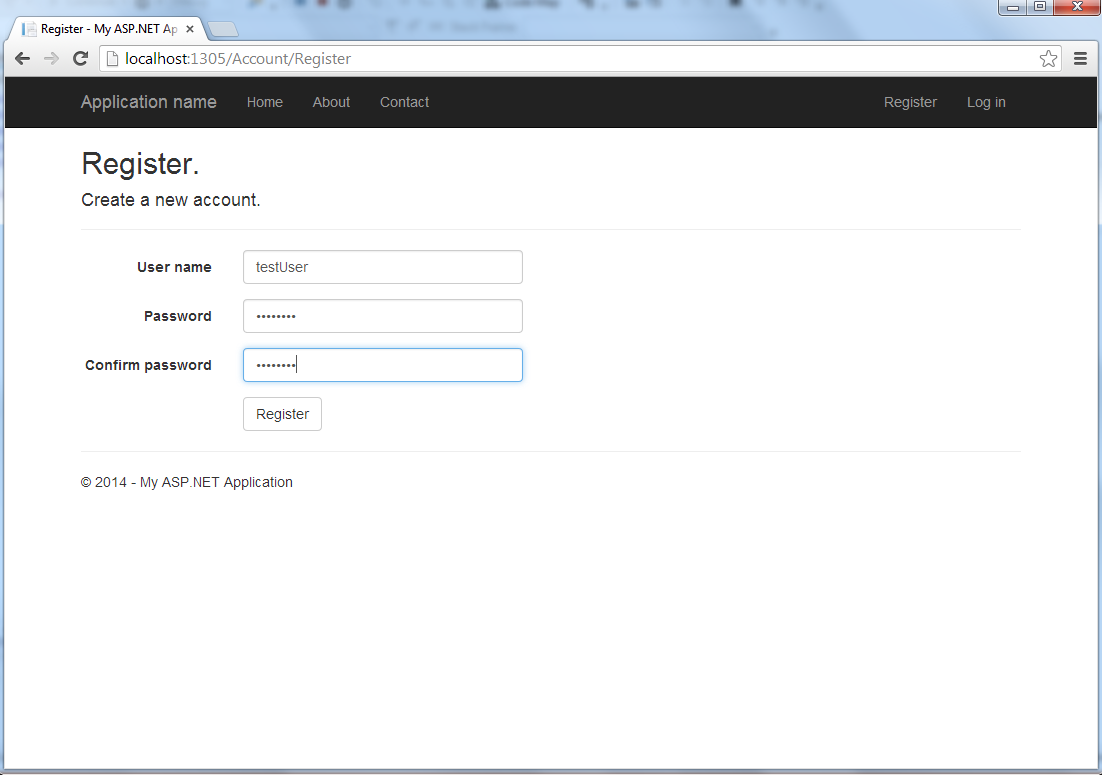
Run the application and navigate to /Test/Index.



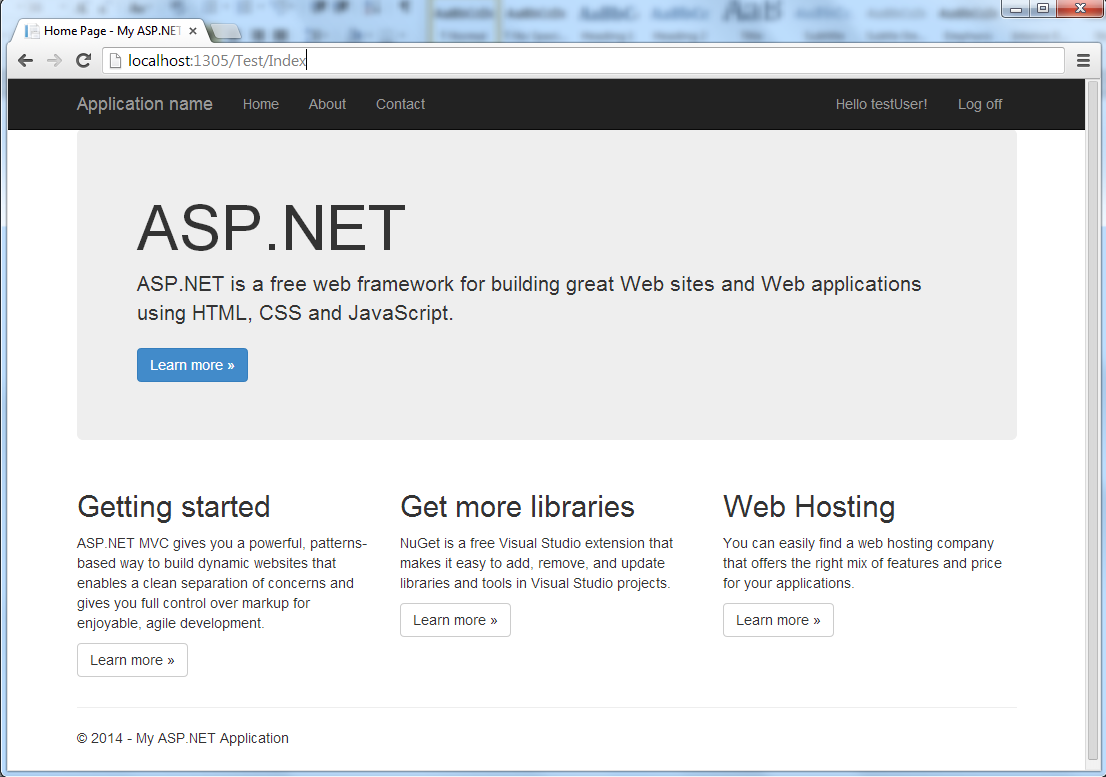
The page is redirected to the login page.



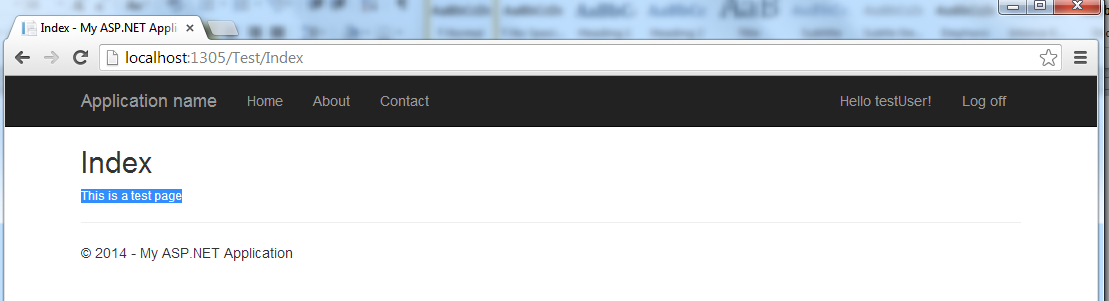
Create a new account called ‘testUser’.



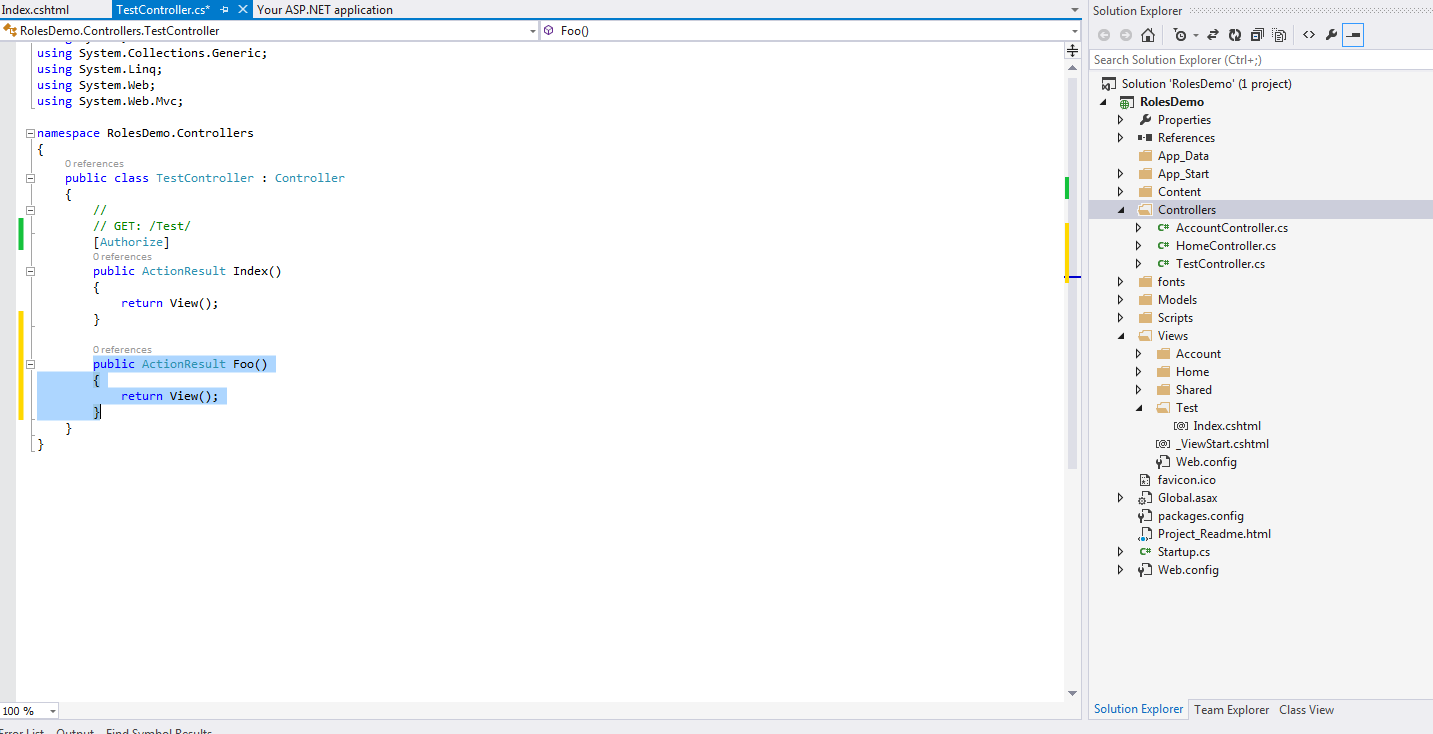
After registration, navigate to /Test/Index



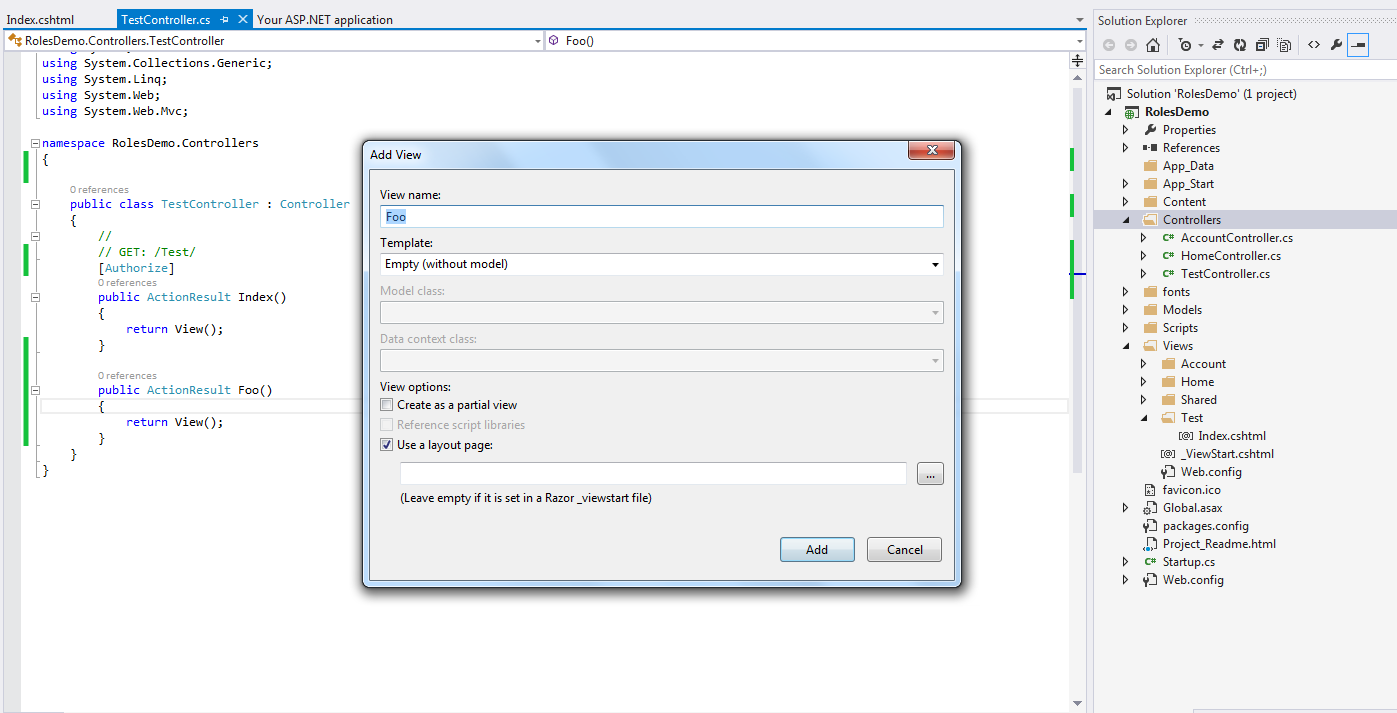
Now the user is logged in, /Test/Index can be viewed.



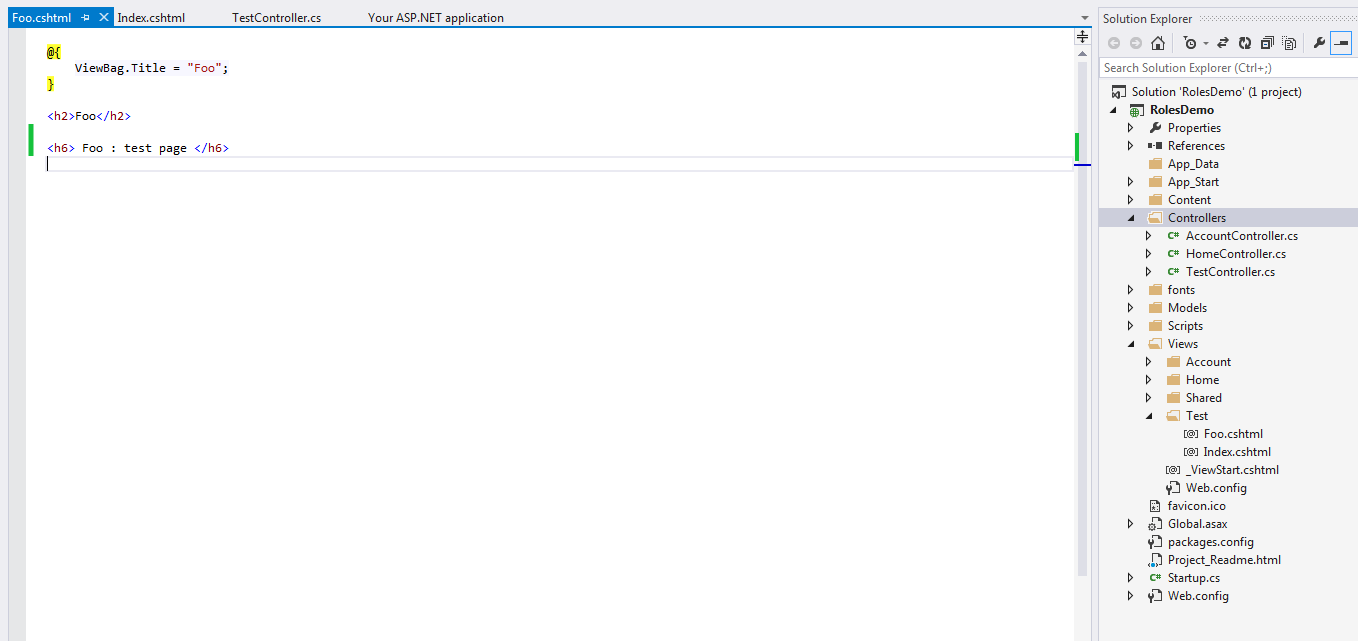
Add another action to the TestController called ‘Foo’.



Add a view to the action ‘Foo’.

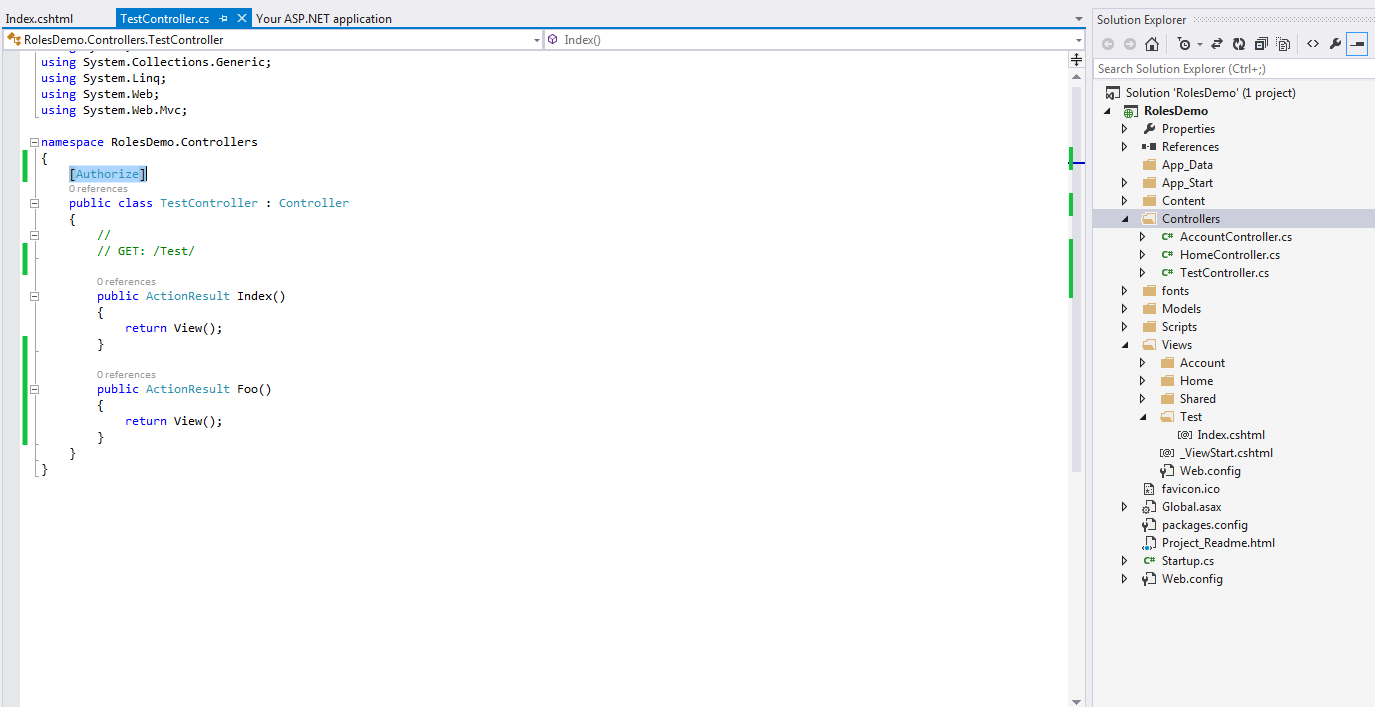


An additional heading has been added to the new page (Foo : test page).

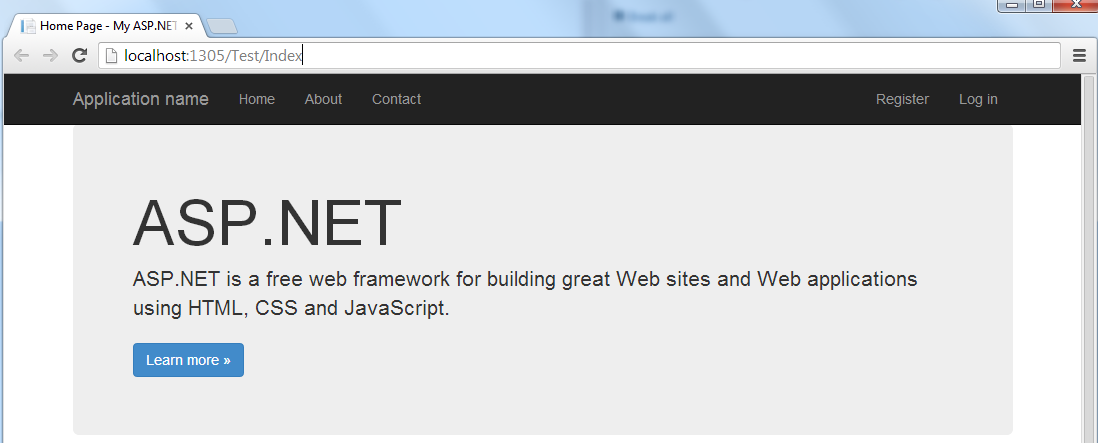


The [Authorize] attribute can be applied to the Controller as well. That will affect every action in that Controller.

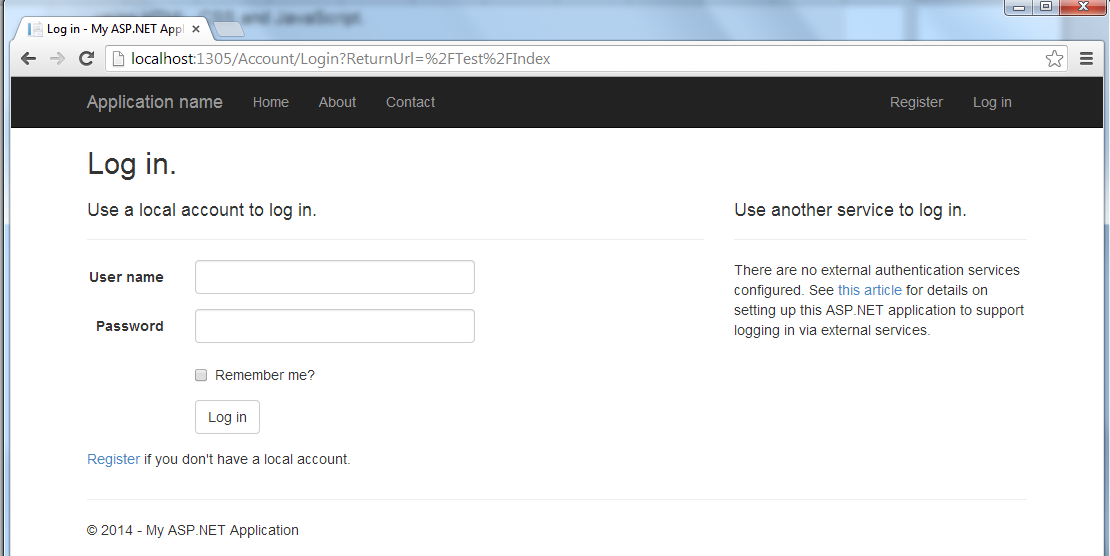
Remove the [Authorize] attribute from the Index action and add it to the TestController as follows:



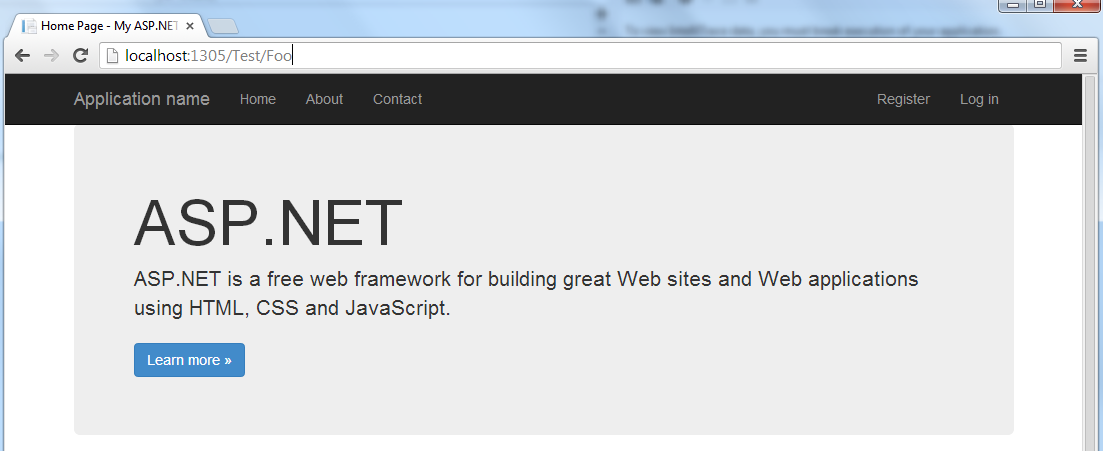
Run the application, and navigate to /Test/Index :



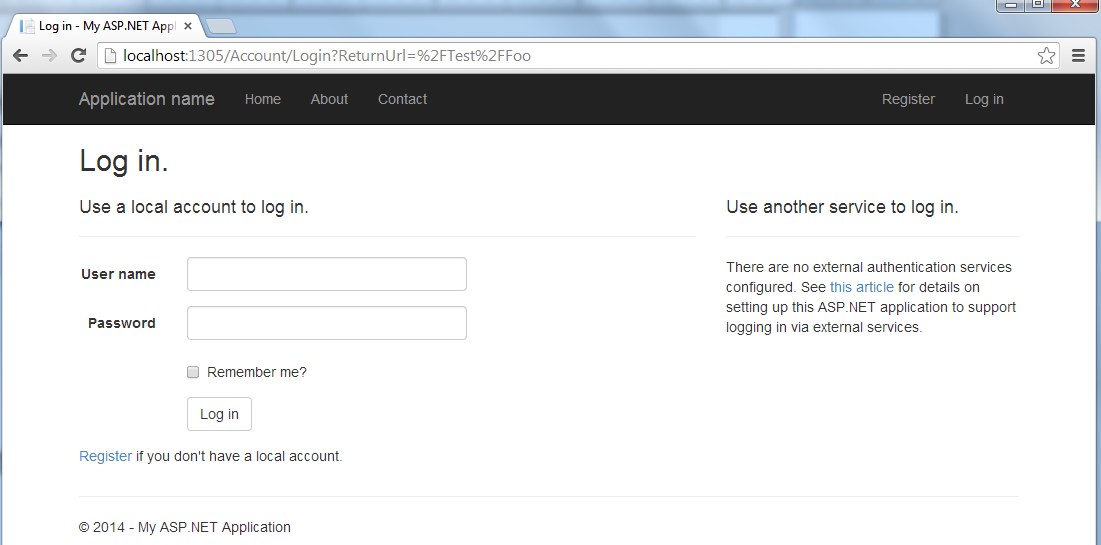
Navigation to /Test/Index redirects the page to login page.



Now try navigating to /Test/Foo. Since the attribute was applied to the controller, this navigation should also redirect to login page if user is not logged in.

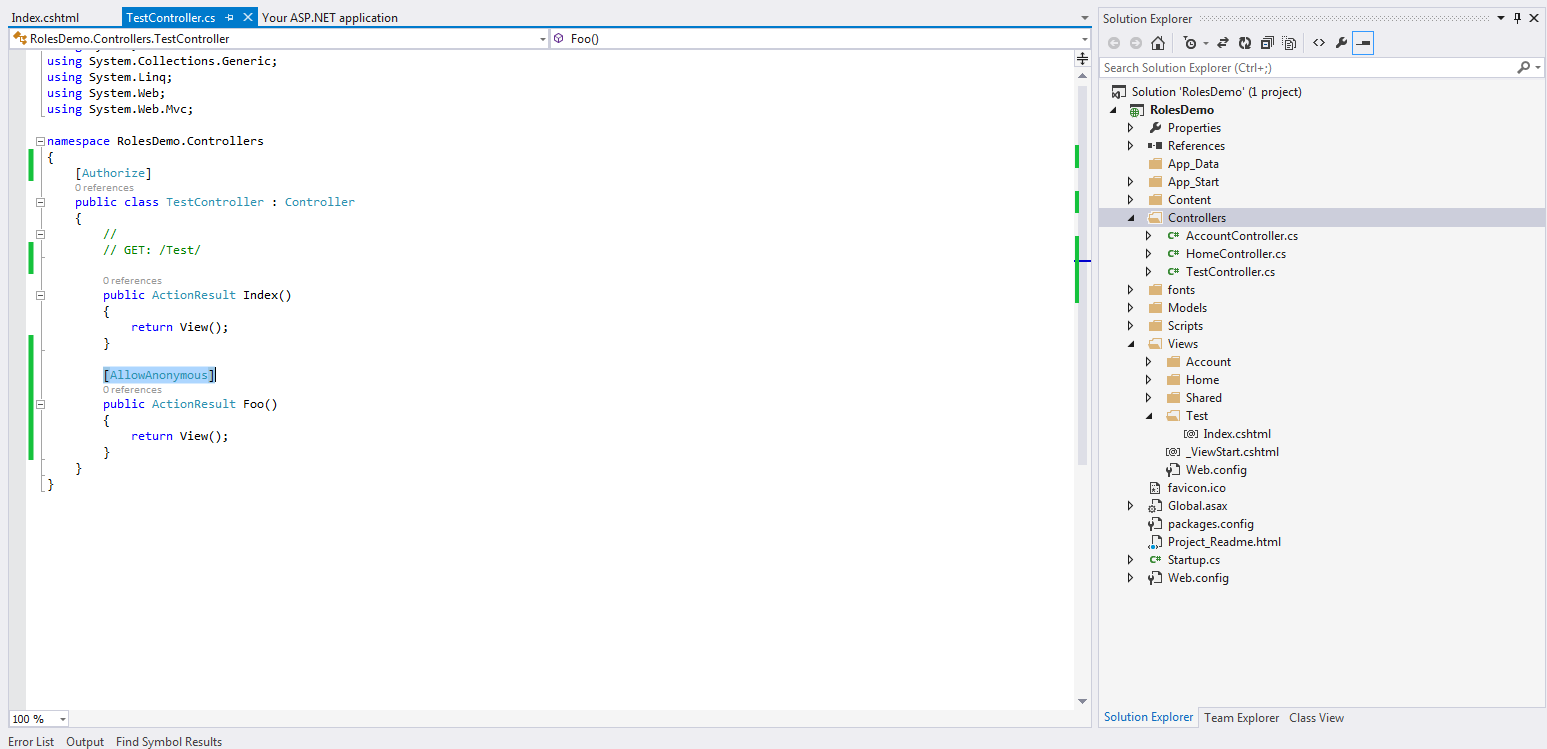


Navigation to /Test/Foo redirects to login page.



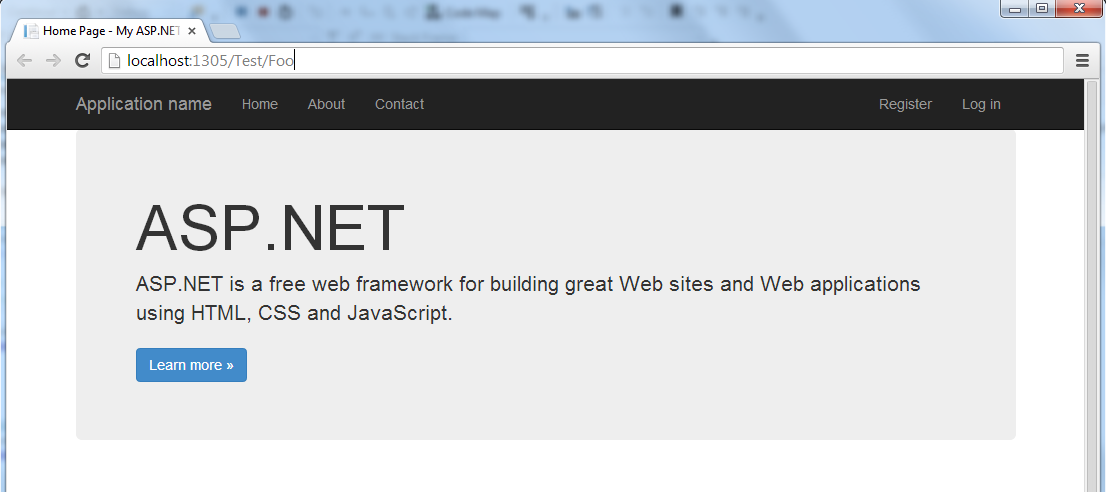
There are situations when you want to exclude some of actions, but still apply the [Authorize] attribute to the controller. For this, the [AllowAnonymous] attribute is used.

Apply the [AllowAnonymous] attribute to the Foo action as follows:

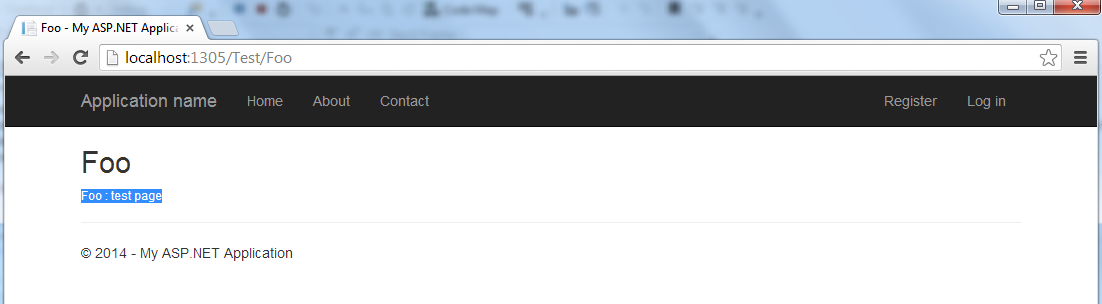


From the above, navigation to action methods other than ‘Foo’ should require authentication.

Run the application and navigate to /Test/Foo:

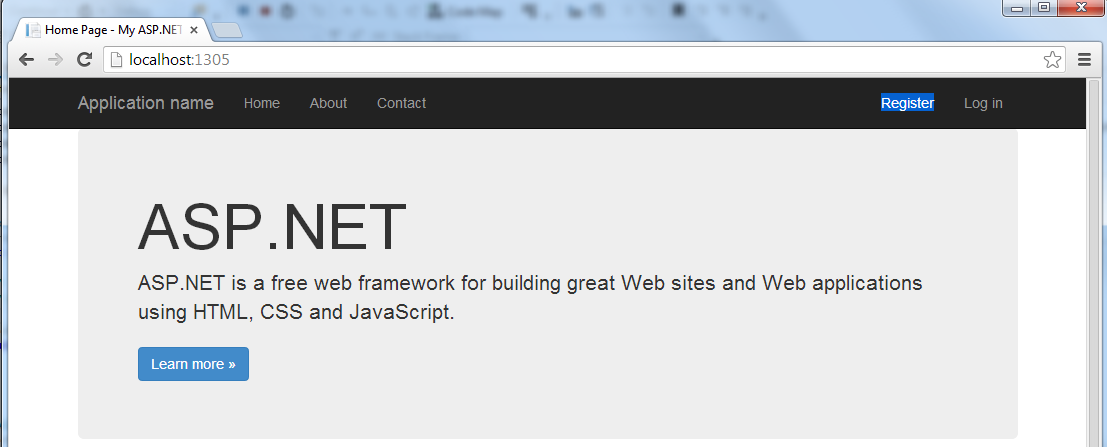


The page is accessible even though user is logged out, because of the [AllowAnonymous] attribute.

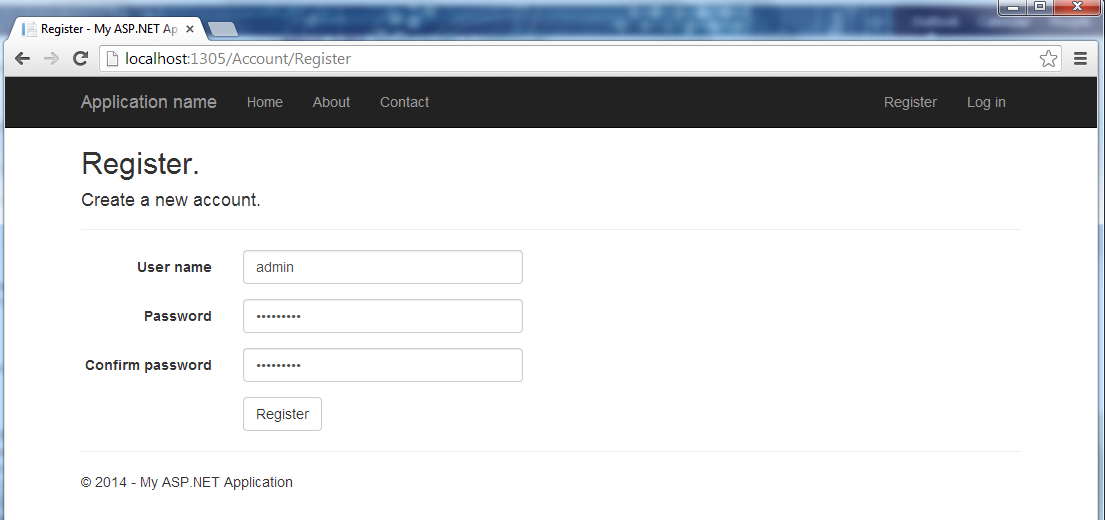


How to make specific pages available only to select users

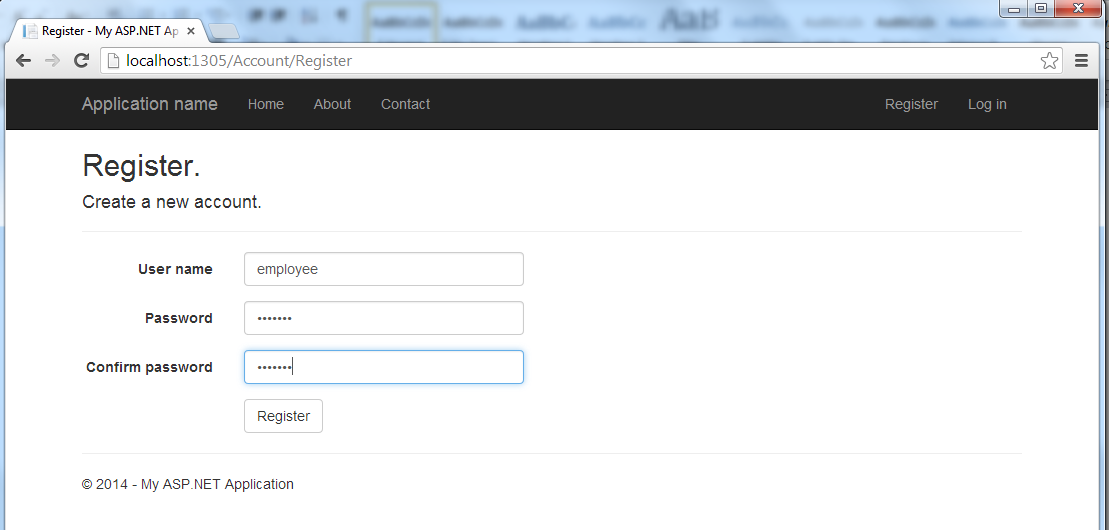
For this demo, we need to create two test users. Run the application and click Register.



Register a user named ‘admin’.



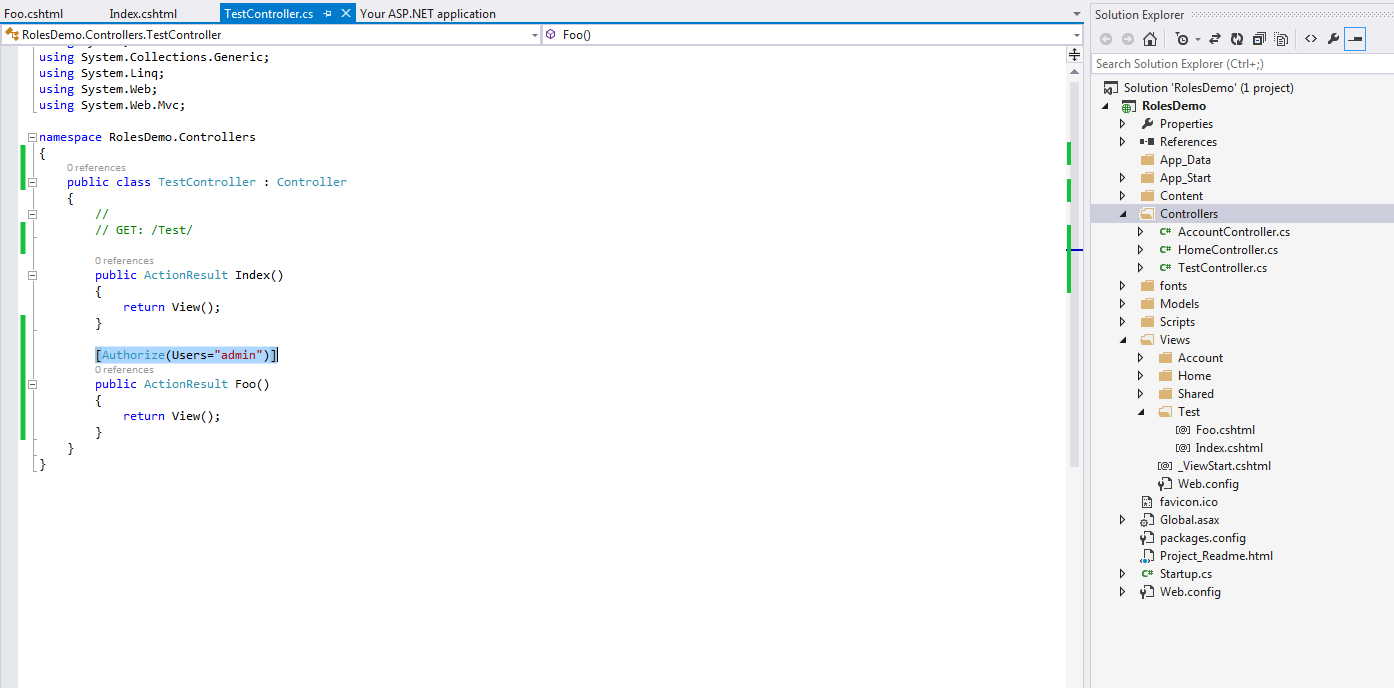
After registering ‘admin’ user, register another user named ‘employee’.



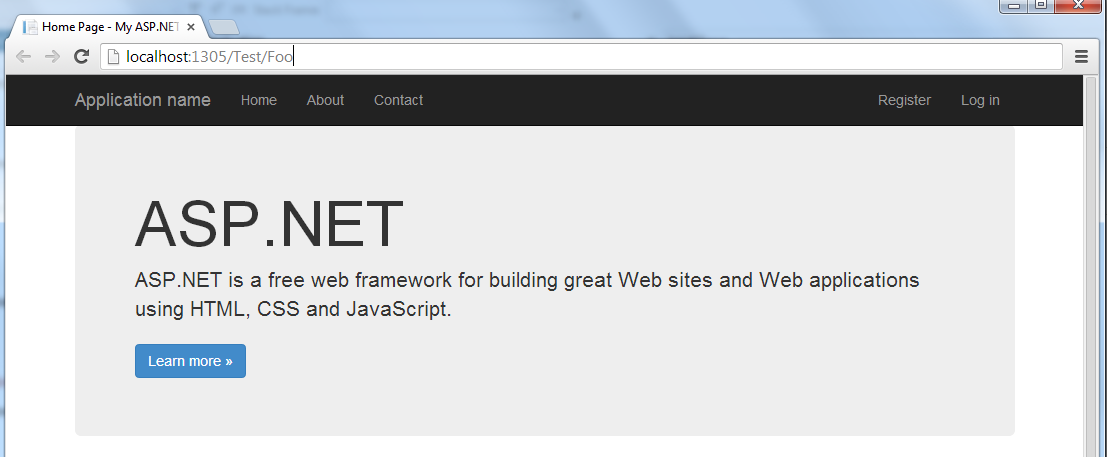
To specify which users are allowed to view a page after authentication, we use the [Authorize] attribute, and specify a comma separated list of users.

Remove the [Authorize] attribute from the TestController.

Add the [Authorize (Users=”admin”)] attribute to Foo action method to specify only ‘admin’ user can view that page after logging in.

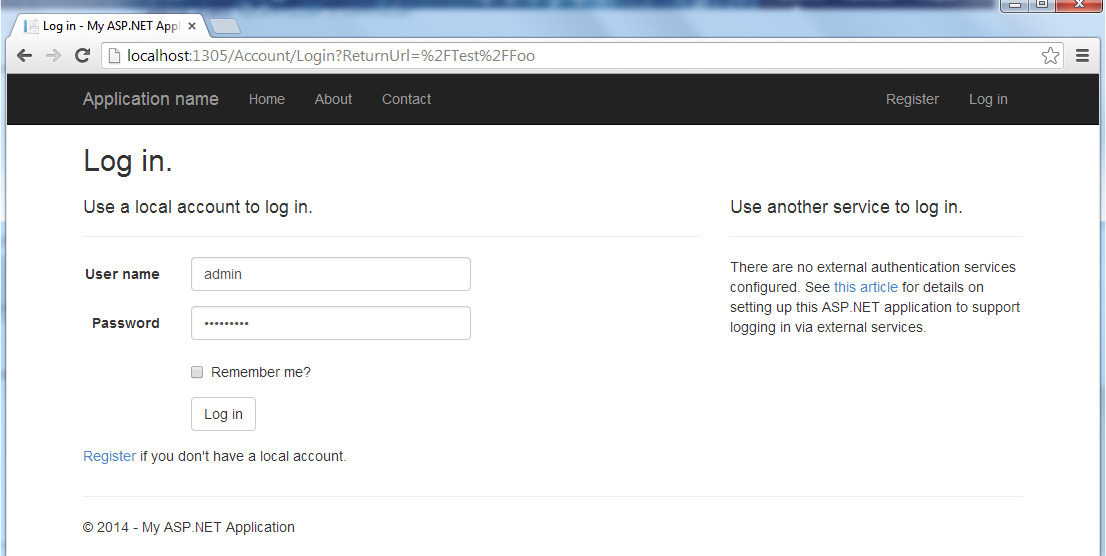


Run the application and navigate to /Test/Foo.

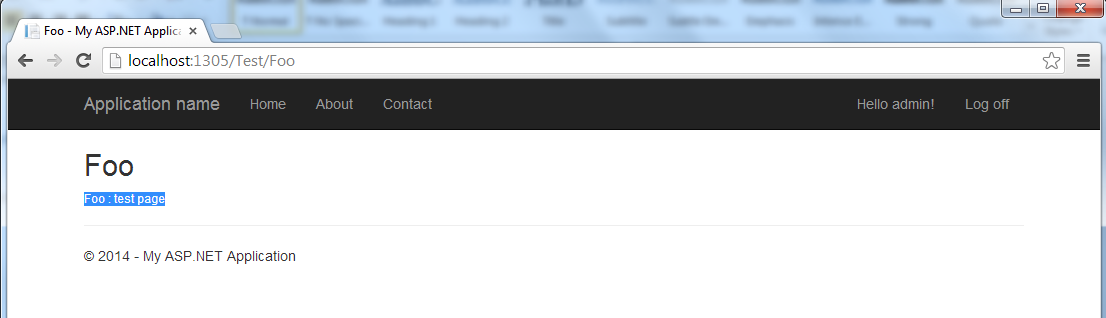


Because of the [Authorize] attribute, the page is redirected to login page.

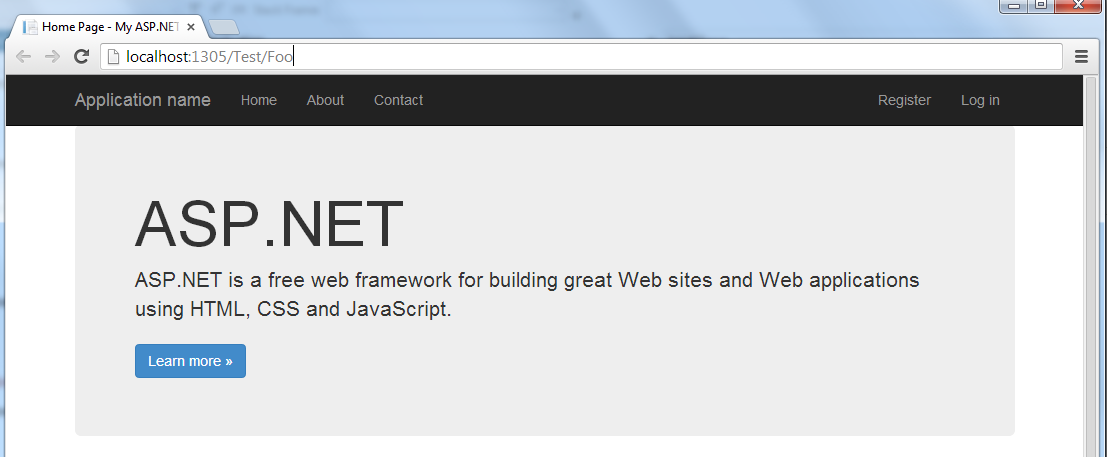
Login as the ‘admin’ user.



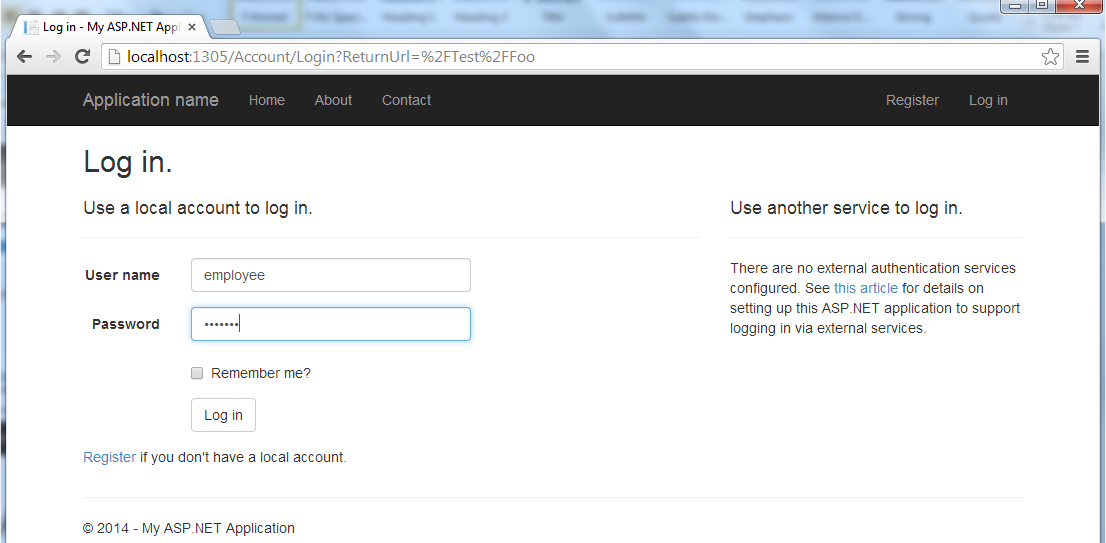
After logging in, the ‘admin’ user can view the page.



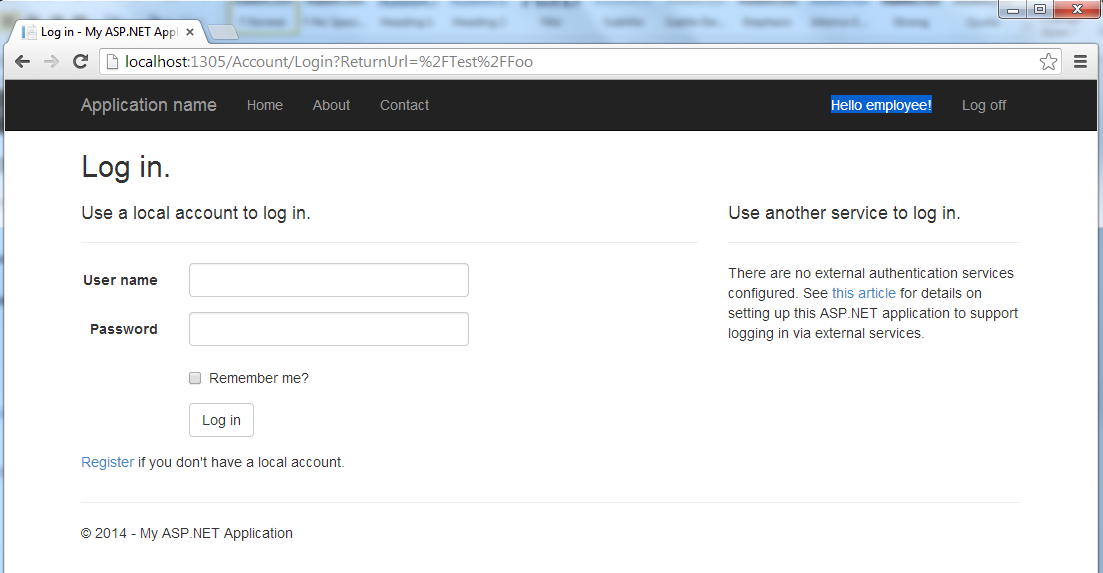
Log out the ‘admin’ user. Navigate to /Test/Foo as follows:



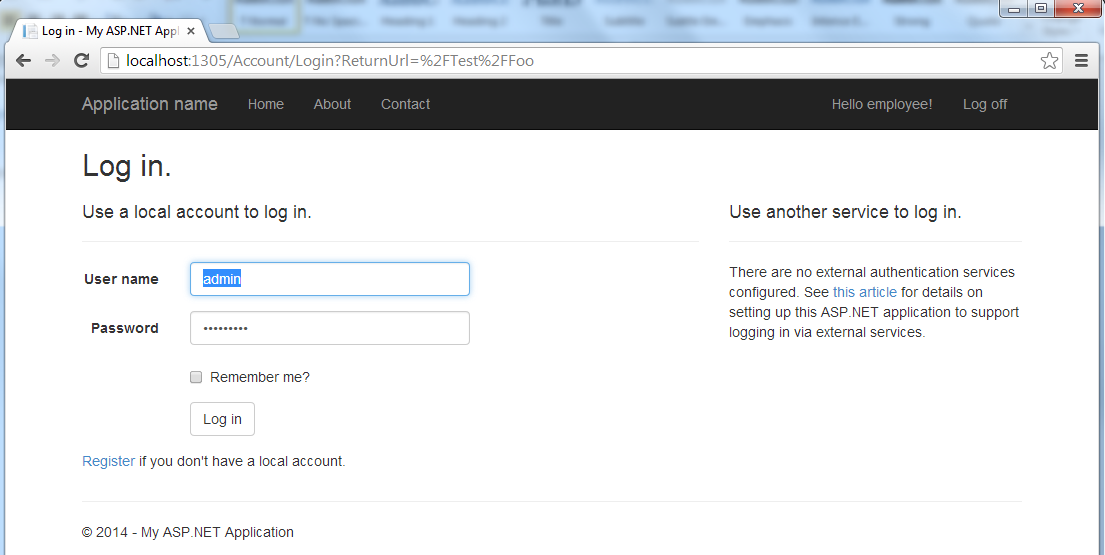
The page will be redirected. Now try logging in as ‘employee’. Since ‘employee’ is not in the list of allowed users, even after logging in, ‘employee’ must not be able to navigate to /Test/Foo.



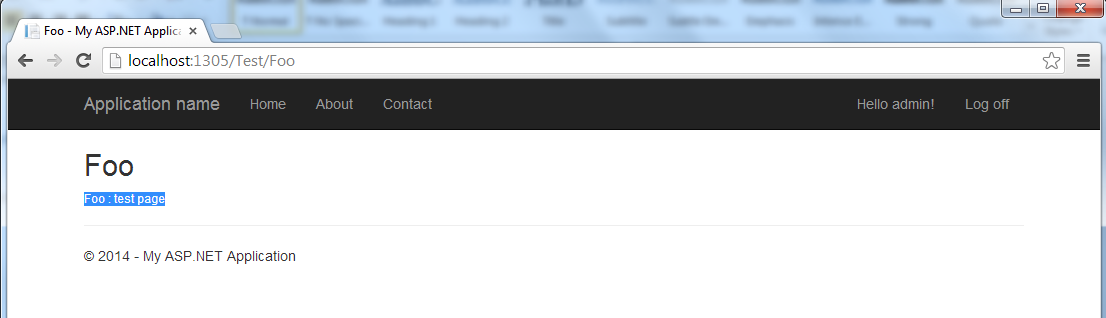
Now try navigating to /Test/Foo. You will observe that the page is navigated to the login page again, even though the ‘employee’ user is logged in. It won’t allow the navigation to /Test/Foo unless a user in the [Authorize] attribute list is logged in.



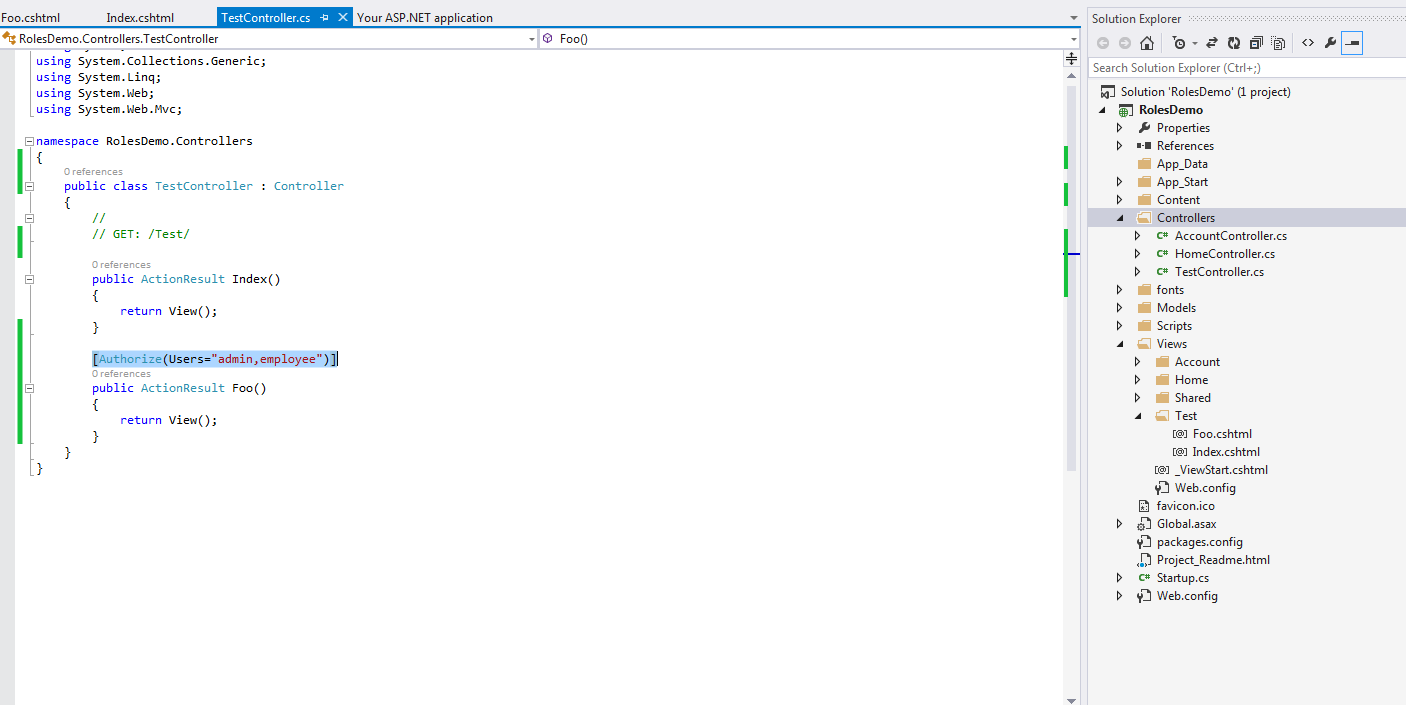
Log in as the ‘admin’ user again.



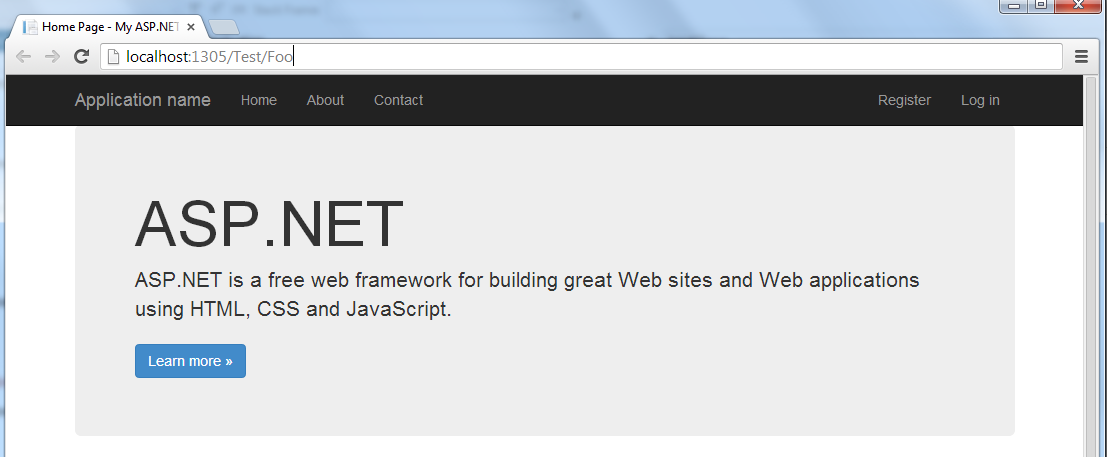
It can be observed that the navigation was allowed, once a user in the list logged in.



To allows both ‘admin’ and ‘employee’ users to view that page, edit the ‘Users’ list in the [Authorize] attribute of the Foo action method and add ‘employee’ to that list as follows:

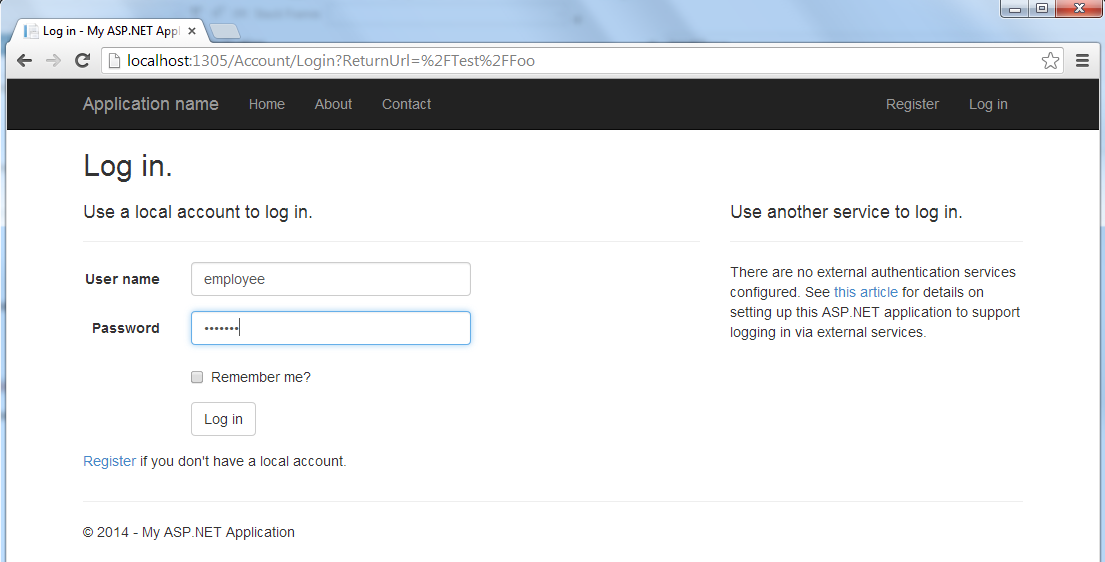


Run the application and navigate to /Test/Foo.

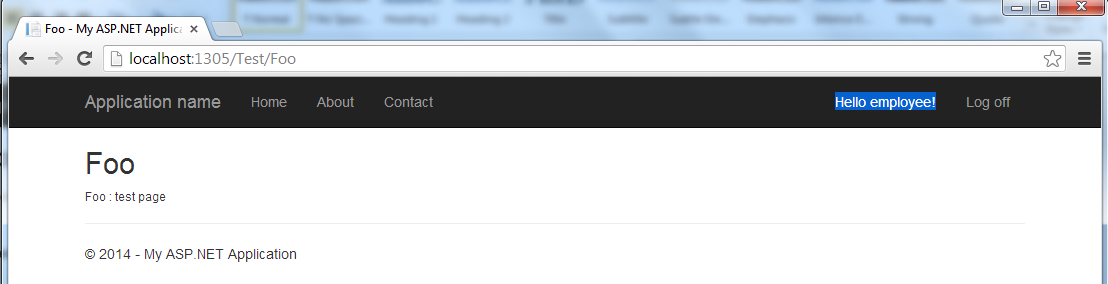


It will be redirected to login page.

Login as the ‘employee’ user.



Navigate to /Test/Foo, and the page will be displayed for ‘employee’ user.



How to make specific pages available only to select roles

In Visual Studio 2013, the ASP .NET Web configuration tool was removed which assisted in managing roles.

To complete this task, we have to manage roles by using an alternative method, by editing some tables using SQL Server Explorer.

The tables we are interested in are:

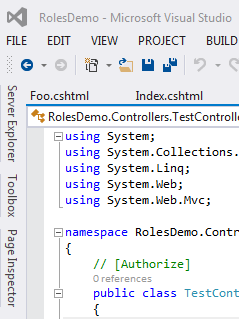
1. AspNetRoles – stores role id and name
2. AspNetUserRoles – stores mapping of user id to role id
3. AspNetUsers – stores user id and user name

The first step is to create some users in the website. For the remainder of this task, it will be assumed that the following users exist:

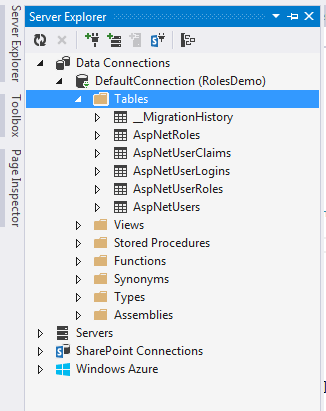
1. admin
2. employee
3. employee2

The first step is to open Server Explorer and get to the window where we can run some queries in the tables.

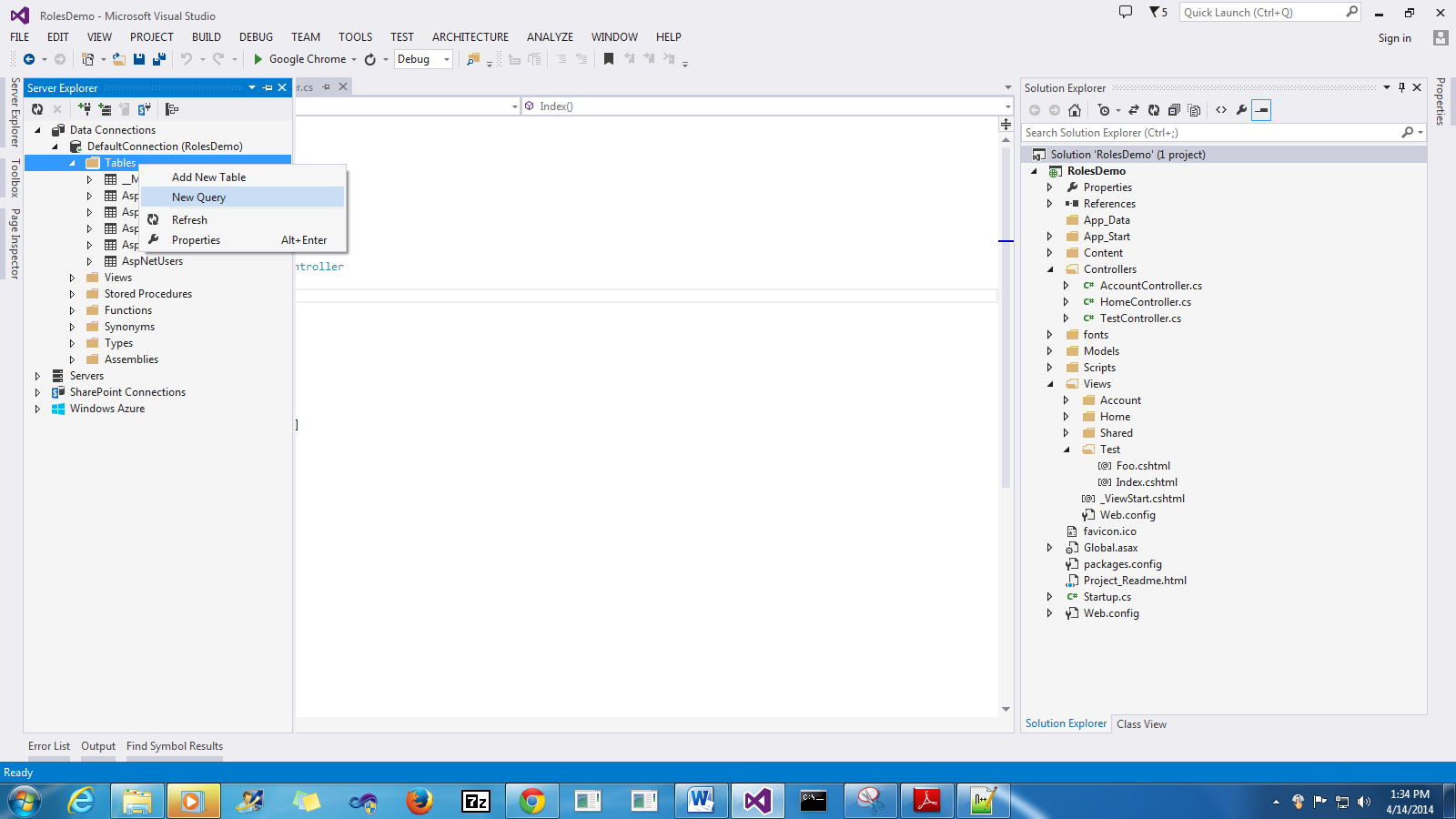
Server Explorer appears as an option in the left column as shown:



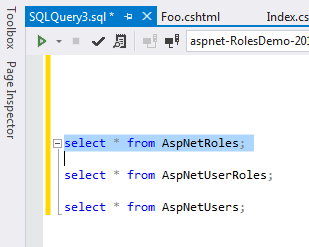
Click and navigate to the Server Explorer and open the tables menu as shown:



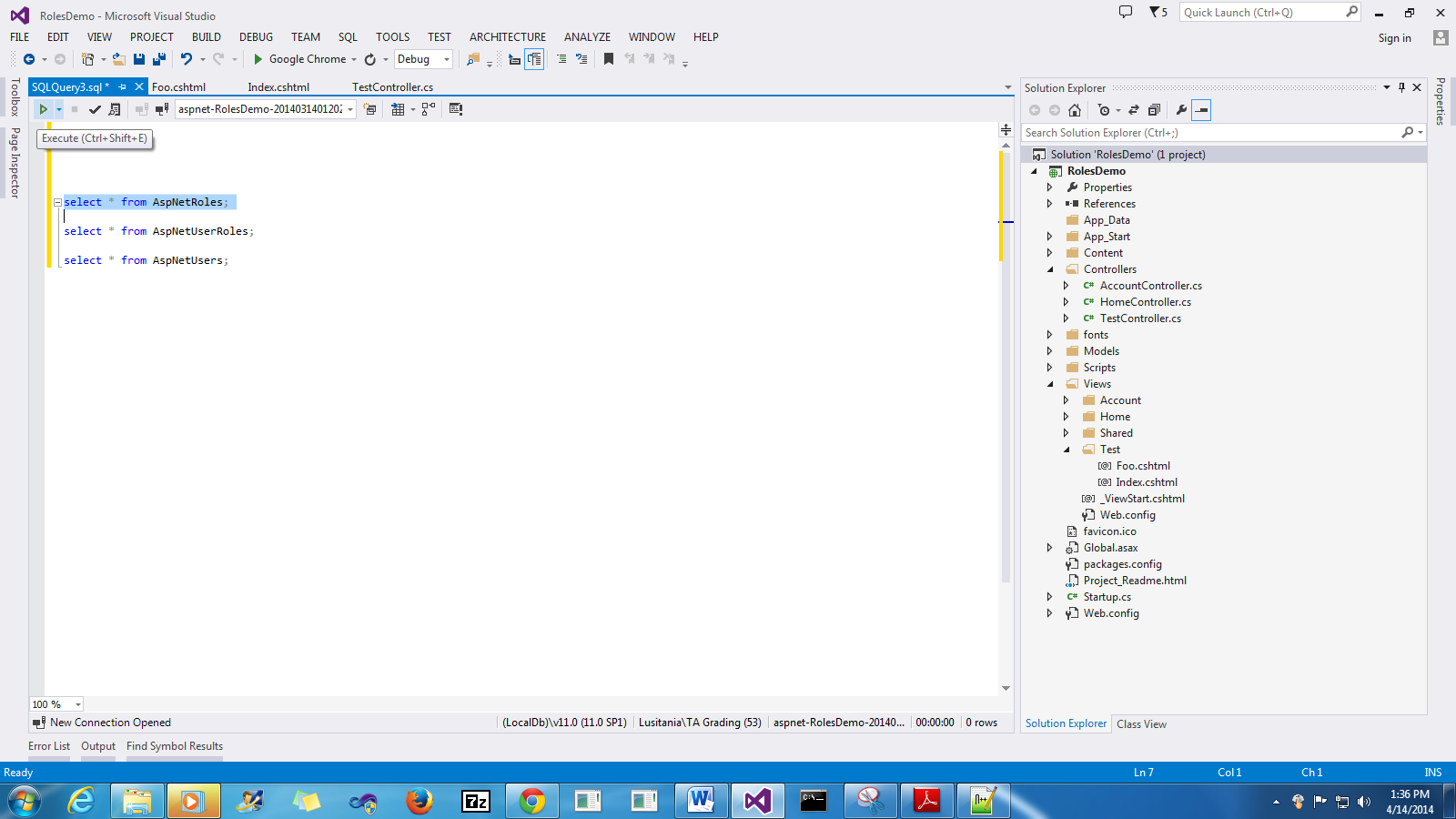
Right click and select new query.

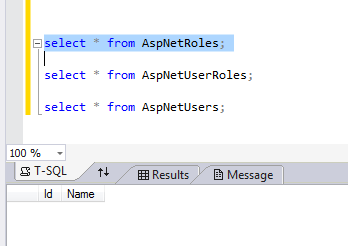


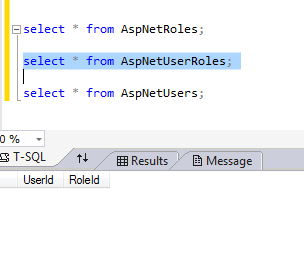
First let’s examine the 3 tables. Use select statements as shown below.

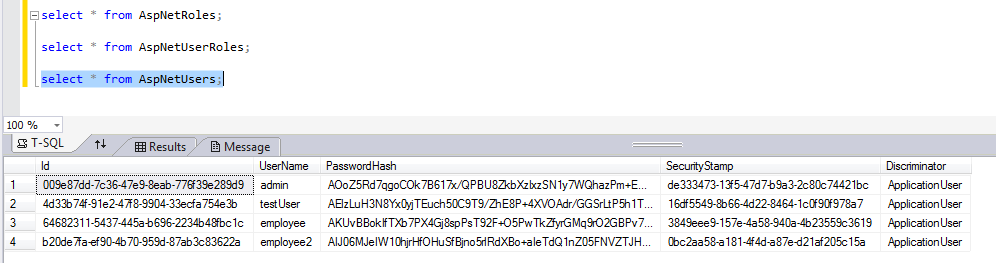


In Server Explorer you can highlight and execute individual lines. The execute button is the green play button in the top left corner.









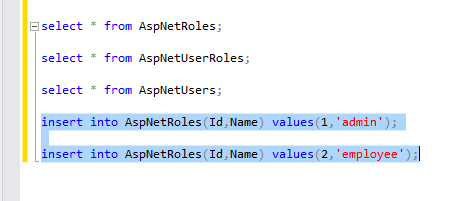
As can be seen, the AspNetRoles, and AspNetUserRoles tables are empty, whereas the AspNetUsers contains details about the current users.

In this task, we will create two roles, namely admin and employee and assign them to users as follows:

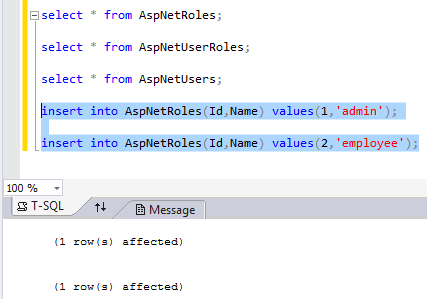
1. admin user -> admin role
2. employee & employee2 user -> employee role

The next step is to create the two roles, admin and employee.

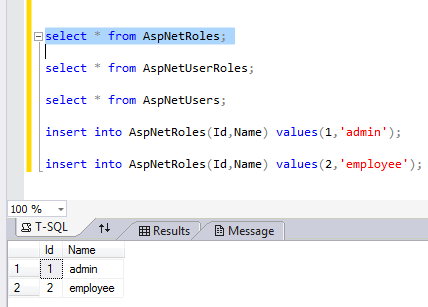
Use insert statements as shown below:



Execute the insert statements (you can highlight them both and click the execute button).

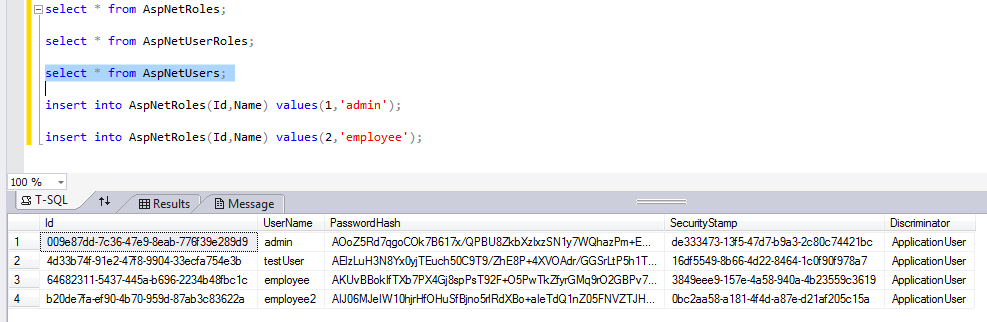


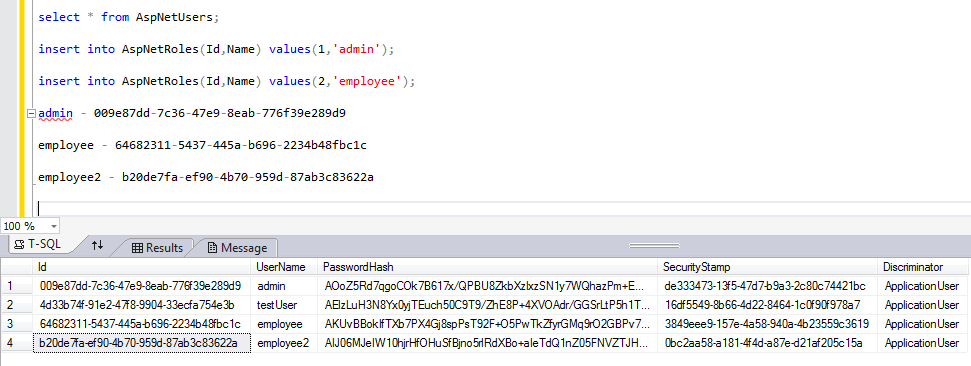
Check the contents of the AspNetRoles table.



The next step is to identify the user ids we want to map.

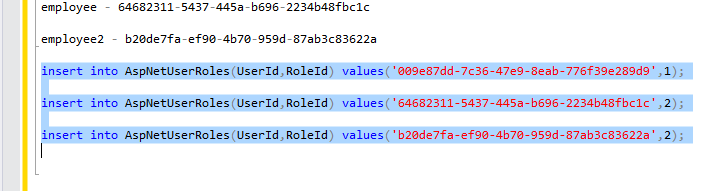
Record the user ids of the desired users somewhere (in this demo, those are just pasted in the same window to keep track of them).



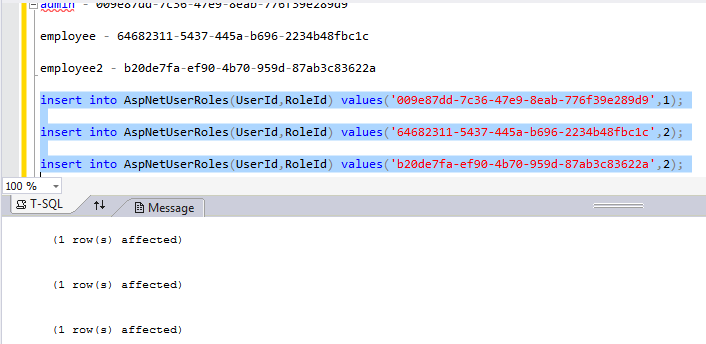


The next step is to insert the mappings between user id and role id. From before, we inserted the role ids:

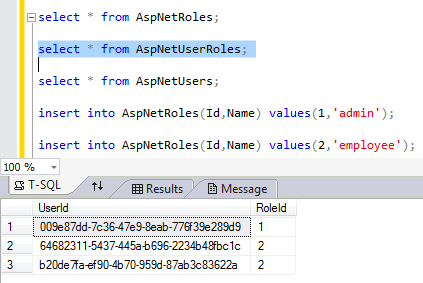
1. admin role -> id 1
2. employee role -> id 2



Execute the insert statements.

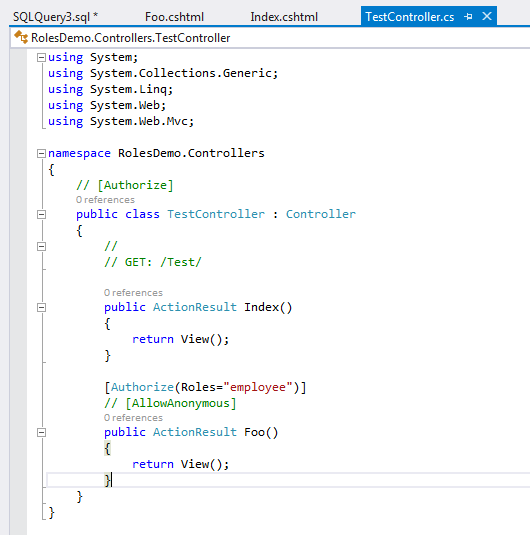


Check the contents of the AspNetUserRoles table.



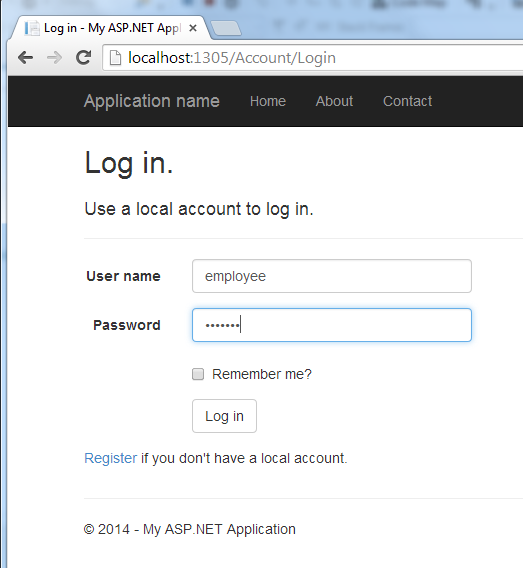
Now our user to role mapping is complete.

The last step is to use these roles in Authorize attribute. Edit the Test Controller as follows (replacing a user list with role list):

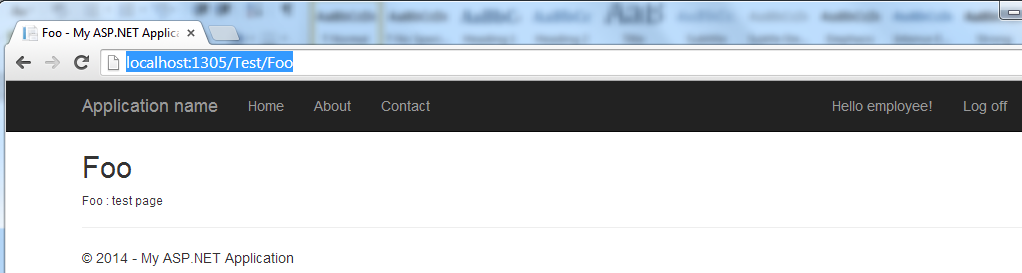


This should only allow users in employee role (employee & employee2) to view the result of Foo Action method.

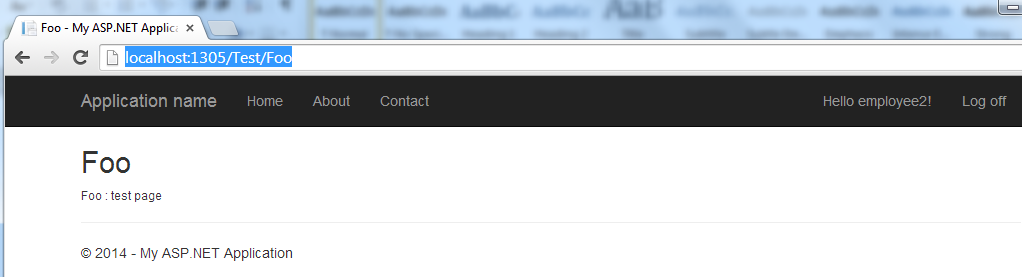
Run the application and login as employee user.



Navigate to Test/Foo as follows:



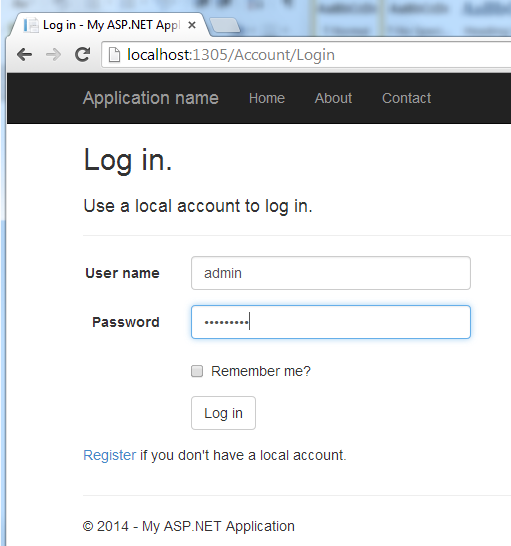
As expected this user can view this page. Repeat the test for employee2 user.



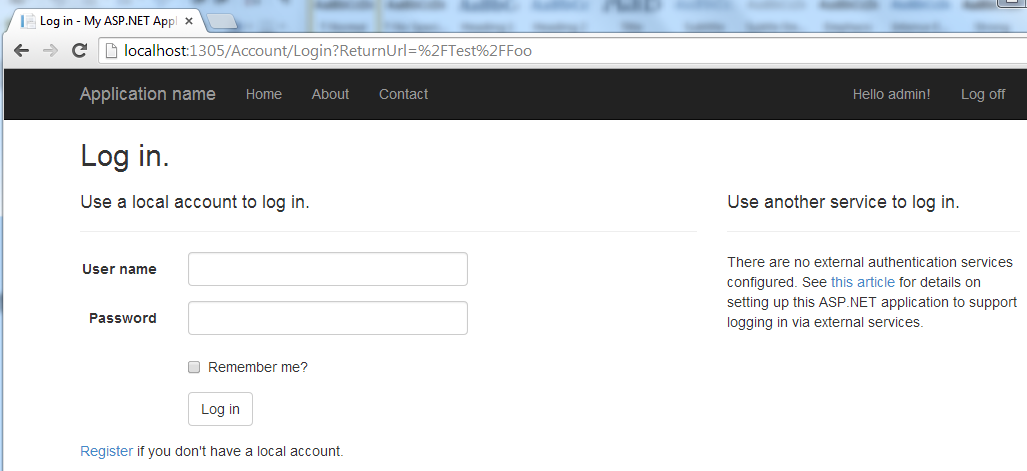
As expected, users in employee role can access the page.

Now let’s test the admin user, which belongs to admin role.

Login as the admin user:

\

Navigate to Test/Foo.



As can be seen, the admin user is not allowed to view the page. This is as we expected, given we only authorized the employee role to access the page.