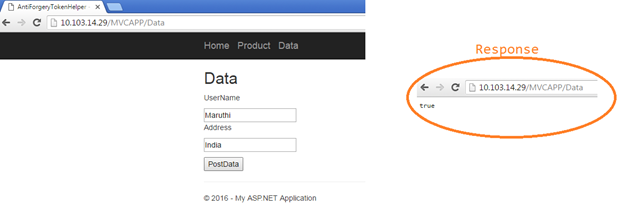
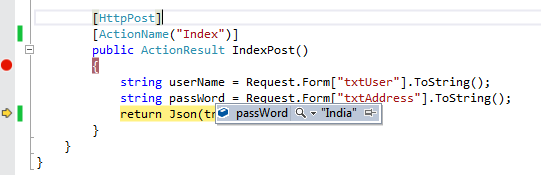
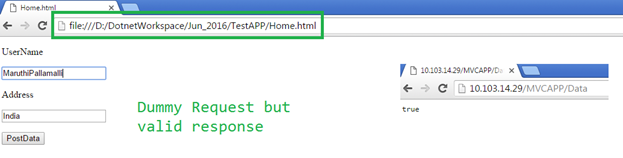
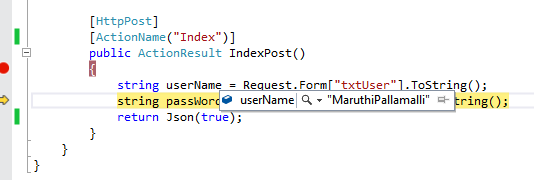
# ValidateAntiForgeryToken Attribute in ASP.NET MVC

# Web applications are exposed to several security threats such as cross-site scripting attacks and cross-site request forgery. In an attempt to assist developers protect their web applications from these attacks ASP.NET provides various techniques. To that end this article shows how an ASP.NET MVC web site can be secured from Cross Site Request Forgery (CSRF).

# Before beginning, you need some information about attacks which is a very sensible thing. A server may have a number of applications and receive a number of requests to an application, but the server can’t find which request is valid.

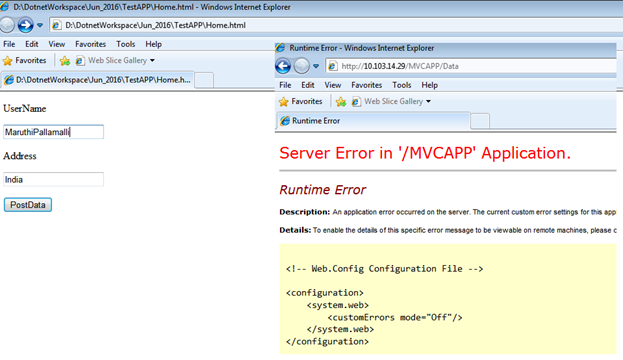
I developed one sample application and ran it in the browser.  
  
  
  
See here, the request and response are both coming from your application only.  
  
  
  
I just tried to change request scenarios and I sent the request to controller from one third party application.  
  
  
  
Here I made a request from one dummy html page and it got a valid response from server.  
  
  
  
**Preventions**  
You can prevent it by using identity values with the help of the ValidateAntiForgeryToken in Asp.Net MVC.

Just add an attribute to your code.

1. [HttpPost]
2. [ValidateAntiForgeryToken]
3. [ActionName("Index")]
4. **public** ActionResult IndexPost()
5. {
6. **string** userName = Request.Form["txtUser"].ToString();
7. **string** passWord = Request.Form["txtAddress"].ToString();
8. **return** Json(**true**);
9. }

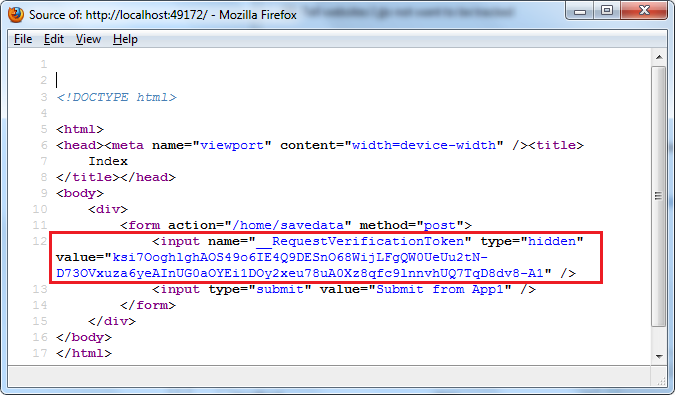
**Code**

1. @Html.AntiForgeryToken()
2. <p>UserName</p><input type="text" id="txtUser" name="txtUser" value="Maruthi" /><br/>
3. <p>Address</p><input type="text" id="txtAddress" name="txtAddress" value="India" /><br/>
4. <p></p><input type="submit" value="PostData" />

Now let’s try again after adding ValidateAntiForgeryToken and check the output  
  
  
  
This time you will not get any response from your server and the server can validate your request with forgery value.

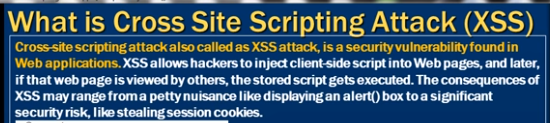
How will  *ValidateAntiForgeryToken* work internally and how it is does it validate your request?

* With the help of *IHtmlString,* a unique value created and stored  in your browser cookies.

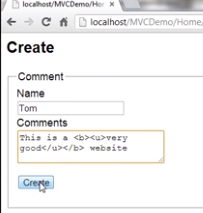


* Every request must carry this value (AntiForgeryToken) under Request. Form[0]
* Internally, the Authorization filter helps to validate this token value.

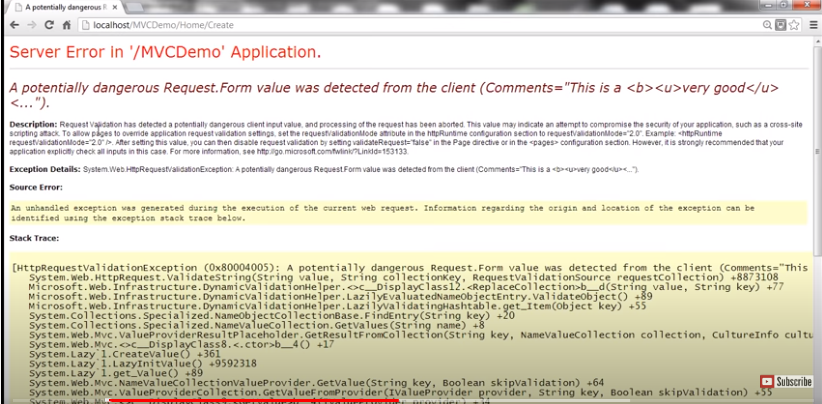
**Cross Site Scripting (XSS)**

****

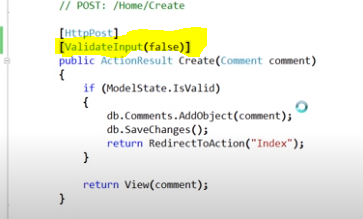
-By default asp.net mvc is checking for the input the value and gave the error if any cross site scripting attack (malicious) input found like below example.



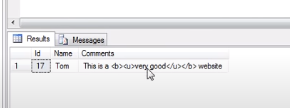
Now if click on the create but we will throw an error because of passing the html tag in comments section.



-Now there is some scenario where we need to allow the user to enter html tag which can be done using the below and disable the checking of the input validation.



-Now after the above change we will able to save the comments in DB.

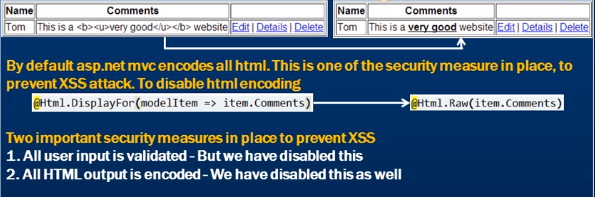


-In UI it look like below.



-As per the above screen shot the word is not showing in bold & underline this is the another security feature which is provided by the asp.net mvc by default all the html tag is encoding format so this will also prevent the cross site scripting.

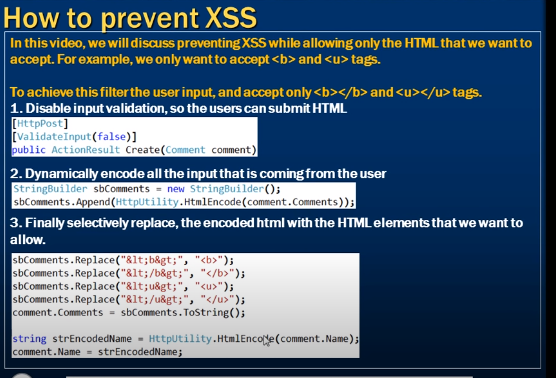
-If we want to show the message in bold and underline we need to do the below changes.



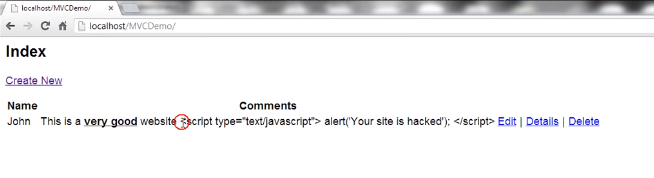


-Still if we write the above comments then we will get the alert the popup once we try to get that comments in UI.

That we can prevent using the below steps.



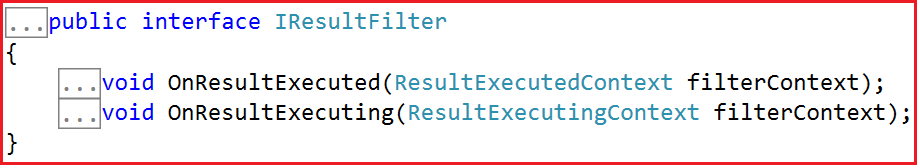
-Below is the final output after the above change.



##### **Result Filters in MVC:**

The Result filters in MVC application are executed before or after generating the result for an action. Action Result type can be ViewResult, PartialViewResult, RedirectToRouteResult, RedirectResult, ContentResult, JsonResult, FileResult and EmptyResult which derives from the ActionResult abstract class. Result filters are called after the Action filters. The in-built [**OutputCacheAttribute**](https://dotnettutorials.net/lesson/outputcache-attribute-mvc/)is an example of Result Filters.

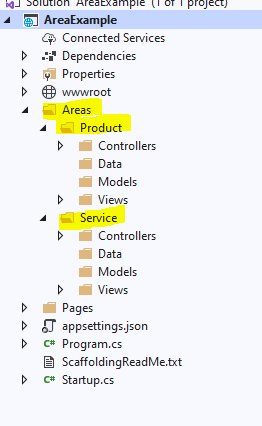
The Result Filters in MVC implements the **IResultFilter** interface. The definition of the IResultFilter interface is given below.



The **IResultFilter** interface provides two methods **OnResultExecuting** and **OnResultExecuted** which will be executed before or after generating the result for an action respectively. If you want to create a Custom Result Filter then you need to implement the **IResultFilter** interface.

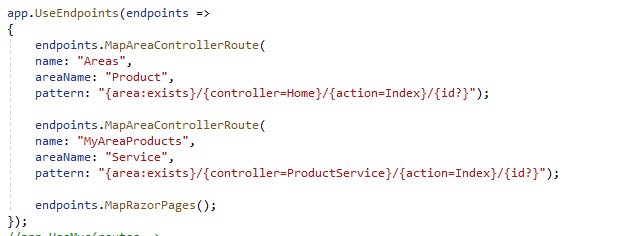
**Area Example with Route in .Net Core**

-First to add the area in project using the VS like below.

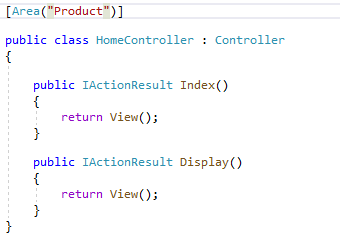


-Now we have two way to access the area.

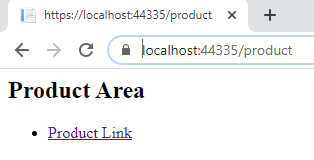
**First way to use the app.UseEndpoints() in Configure() in startup.cs file.**

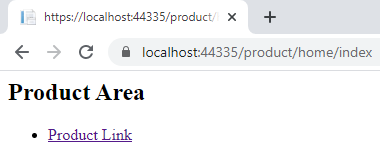


-Then create the controller and define the Area attribute in controller.



-Then when hit the below url it will redirect to the product area controller.

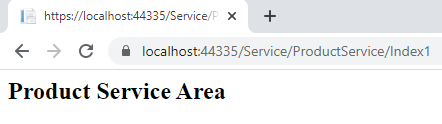




-Also we can call the Area route using the @html.ActionLink() like below



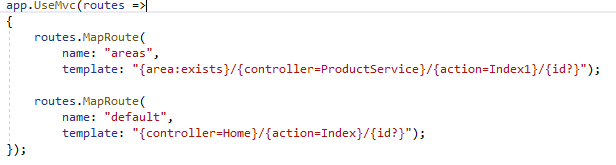
When click on Product Link it will redirect to the Service area.



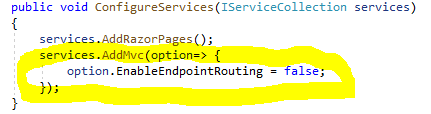
**Second way to use the app.UseMVC() in ConfigureService() of startup.cs file.**

-We can perform the same area route using the app.UseMvc() just only 2 change we need to do all are the same.

1- Replace the app.UseEndpoints() to app.UseMvc() in Configure() like below.

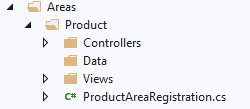


2-Make the enable endpoint route to false in ConfigureService() like below.

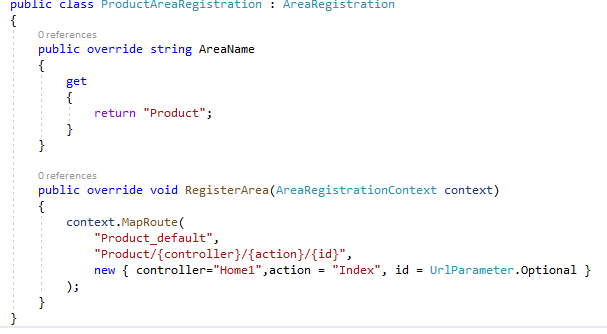


**Area Example in .Net framework**

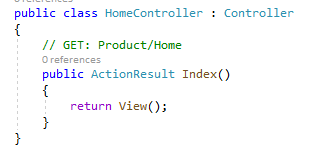
**-**When we create the area in .Net Framework it will create the below structure.



As you can see, each area includes ProductAreaRegistration class in {area name} + AreaRegistration.cs file.

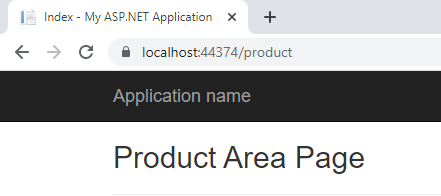


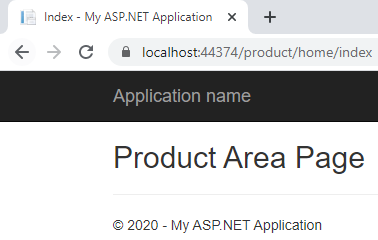
Created Below home controller



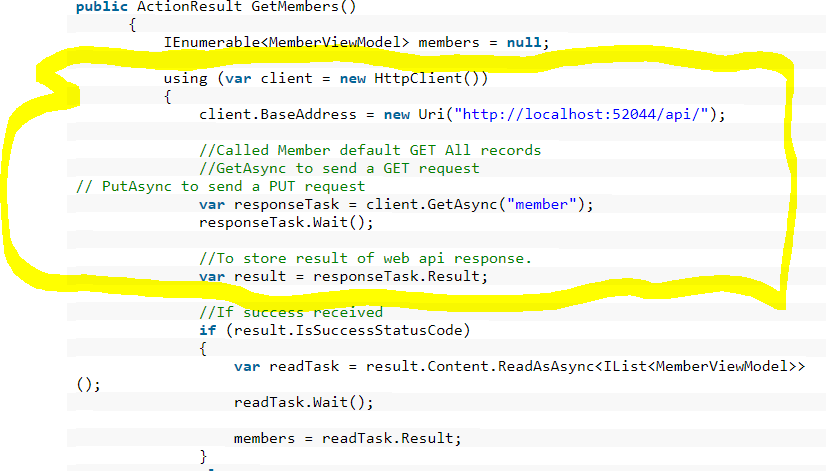
Finally, all the area must be registered in Application\_Start event in Global.asax.cs as AreaRegistration.RegisterAllAreas();

-Now hit the below urls & it will work.





**Consume Web API Get method in ASP.NET MVC**



**Difference Between Keep() and Peak()**

The Peek and Keep methods allow you to read the value without marking it for deletion. Say we get back to the first request where the value was saved to TempData.

With Peek you get the value without marking it for deletion with a single call,

TempData["value"] = "someValueForNextRequest";

//second request, PEEK value so it is not deleted at the end of the request

object value = TempData.Peek("value");

//third request, read value and mark it for deletion

object value = TempData["value"];

With Keep you specify a key that was marked for deletion that you want to keep. Retrieving the object and later on saving it from deletion are 2 different calls.

//second request, get value marking it from deletion

object value = TempData["value"];

//later on decide to keep it

TempData.Keep("value");

//third request, read value and mark it for deletion

object value = TempData["value"];

You can use Peek when you always want to retain the value for another request. Use Keep when retaining the value depends on additional logic.

**Difference between MVC4 and MVC5**

