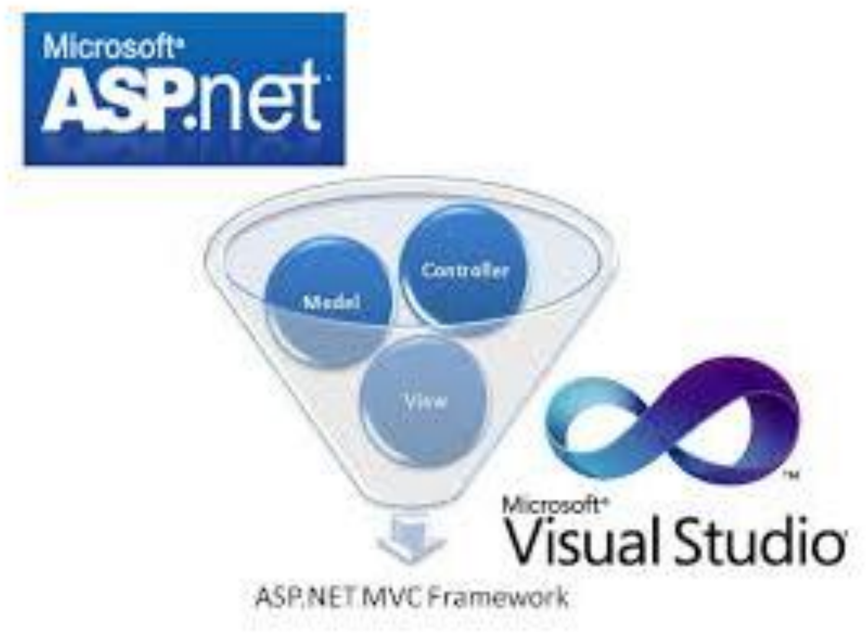


C#

6. SECURITY AND ASP.NET MVC

Overview

- Authentication
  - Authentication Attributes
  - Windows Authentication
  - Forms Authentication
- Authorization
- XSS Cross Site Scripting
- CSRF – Cross Site Request Forgery

Authentication

Forms

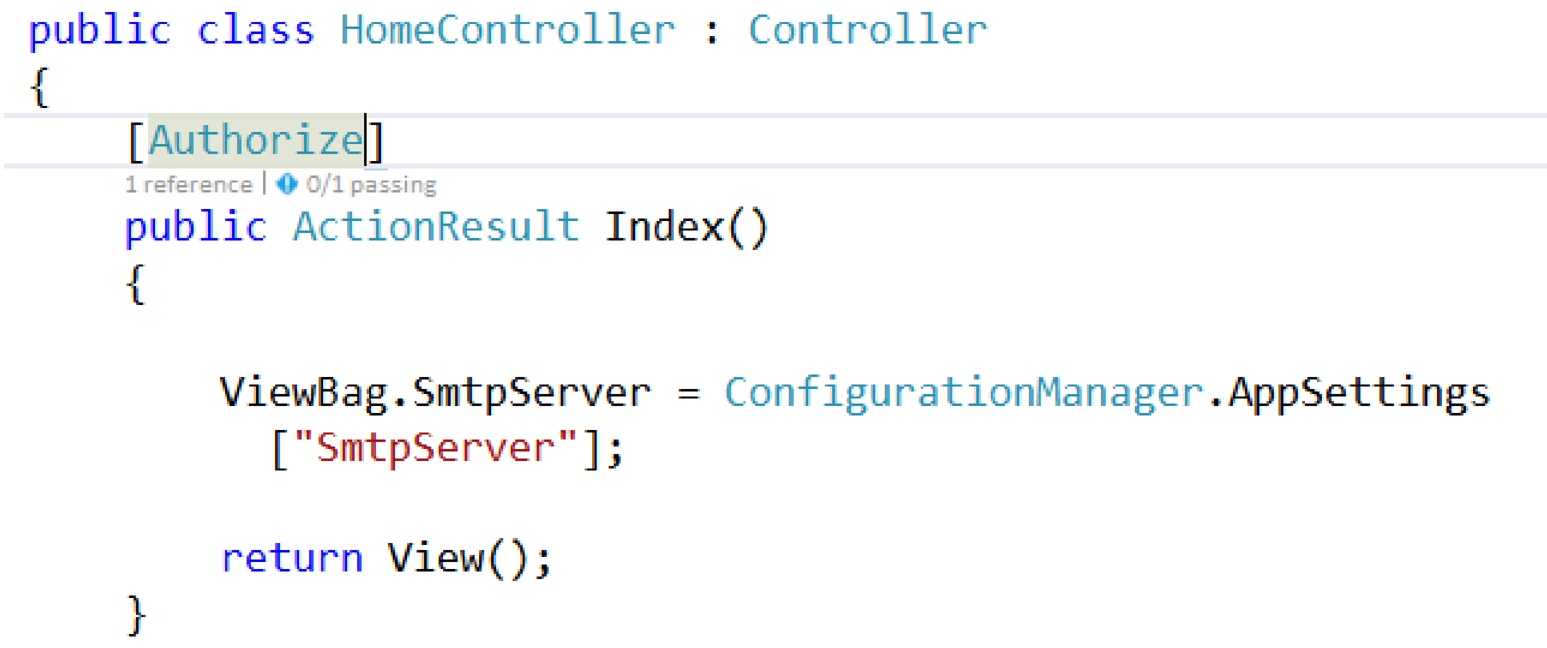
- Internet applications
- Customizable
- Typically relies on cookies
- Some SSL required

Windows

- Also called as Integrated Authentication
- Intranet Applications
- Single Sign On

C#

Controller

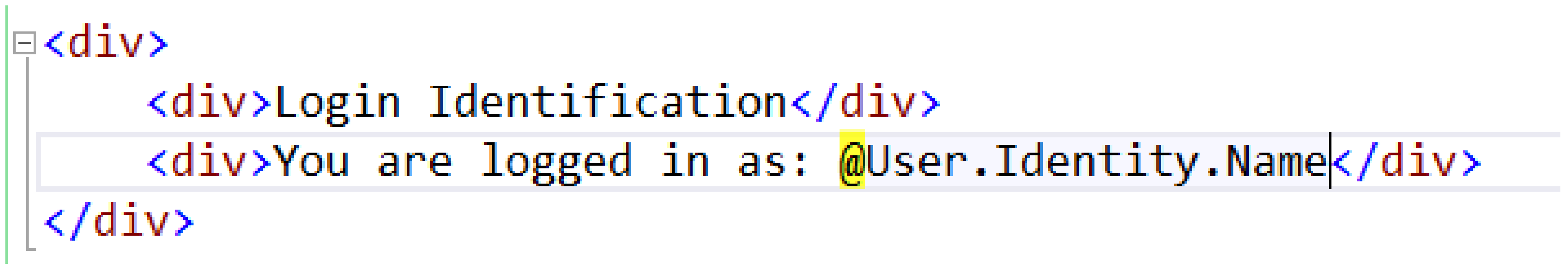


```
public class HomeController : Controller
{
    [Authorize]
    1 reference | 0/1 passing
    public ActionResult Index()
    {

        ViewBag.SmtpServer = ConfigurationManager.AppSettings
          ["SmtpServer"];

        return View();
    }
```

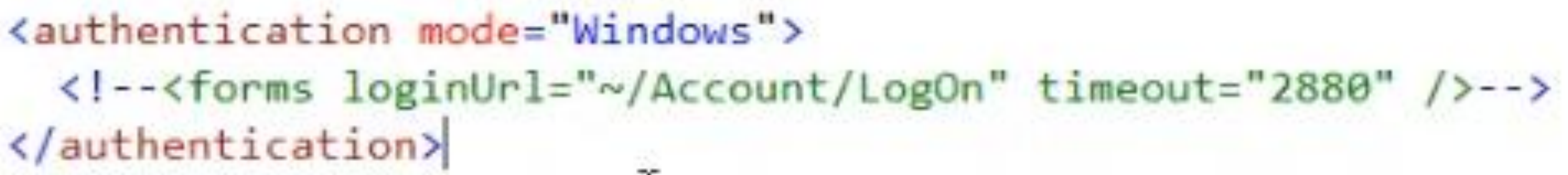
View



```
<div>
    <div>Login Identification</div>
    <div>You are logged in as: @User.Identity.Name</div>
</div>
```

Windows Authentication

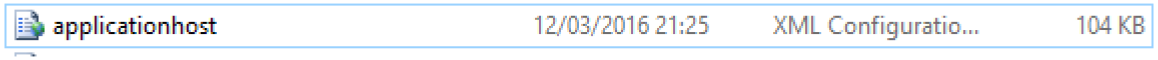
1.Web.Config -> Authentication mode= "Windows"



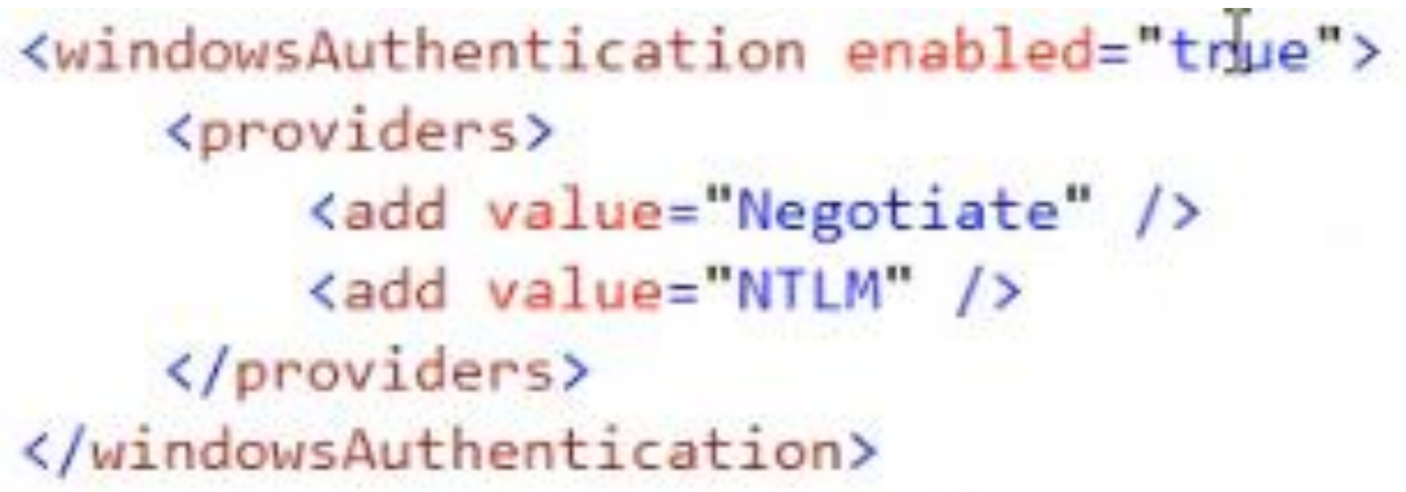
```
<authentication mode="Windows">
  <!--<forms loginUrl="~/Account/LogOn" timeout="2880" />-->
</authentication>
```

2. C:\Users\SyedAwase\Documents\IISExpress\config

Applicationhost



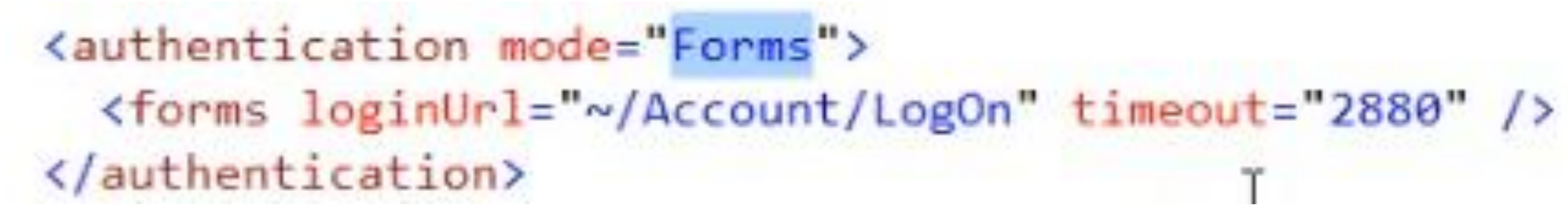
```
applicationhost    12/03/2016 21:25    XML Configuratio...    104 KB
```



```
<windowsAuthentication enabled="true">
    <providers>
        <add value="Negotiate" />
        <add value="NTLM" />
    </providers>
</windowsAuthentication>
```

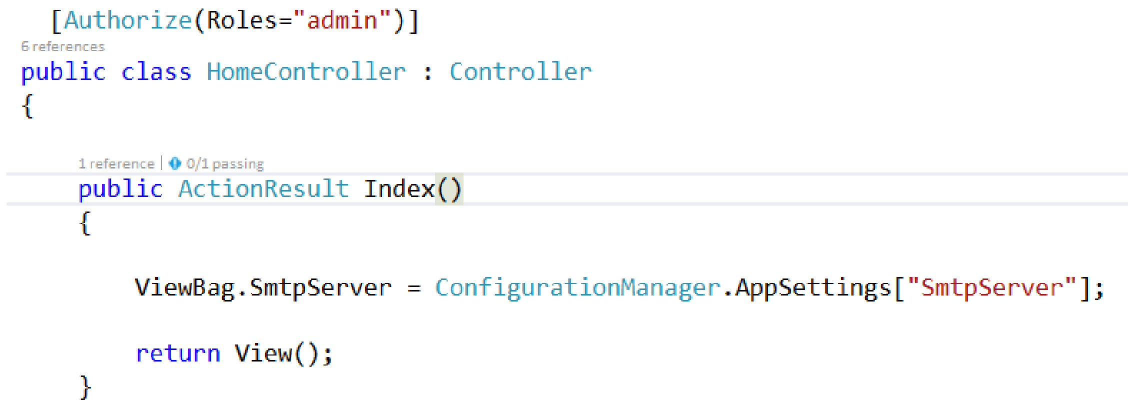
Forms

Web.Config -> Authentication mode= “Forms”



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

Authorization



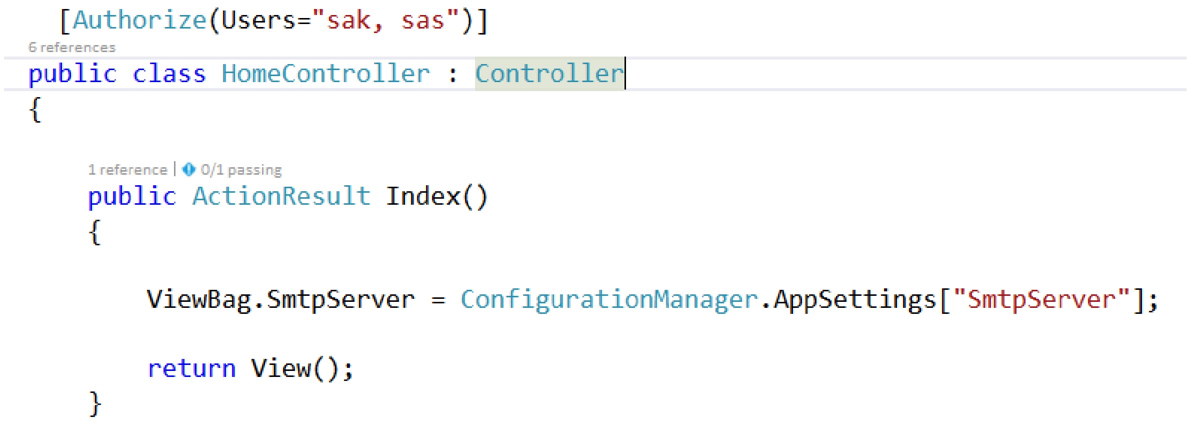
```
  [Authorize(Roles="admin")]
6 references
public class HomeController : Controller
{

    1 reference | 0/1 passing
    public ActionResult Index()
    {

        ViewBag.SmtpServer = ConfigurationManager.AppSettings["SmtpServer"];

        return View();
    }
```

Authorize based on **ROLES**



```
  [Authorize(Users="sak, sas")]
6 references
public class HomeController : Controller
{

    1 reference | 0/1 passing
    public ActionResult Index()
    {

        ViewBag.SmtpServer = ConfigurationManager.AppSettings["SmtpServer"];

        return View();
    }
```

Authorize based to **Specific usernames**

XSS: Cross-Site Scripting

- Cookie theft
- Cross-site Scripting
- Modification of user settings
- Modification of content
- Account Hijacking
- Malware Download

Cross Site Request Forgery

- CSRF is also known as one click attack, sidejacking or session riding.
- It merely transmits unauthorized commands from a user the website trusts.

How CSRF works

- GET requests are the easiest:
  - “src” and “href” attributes
- POST aren’t immune either

```
<body onload=”document.forms[0].submit()”>
<form method=”POST” action=”_url_”>
    <input type=”hidden” name=”amount” value=”$1,000” />
</form>
```

CSRF Prevention

- Avoid Persistent Sessions
- Use GET method properly
- Token-based checks
- Double Authenticate via AJAX(read cookie via JS and submit in the body)
- Code reviews.

Summary

- Various Authentication and Authorization approaches available for ASP.Net MVC Applications
- How to avoid XSS using HTML encode
  - strictly restrict the use of HTML.Raw()
- CSRF