Показване на MVC Filter attributes. Създаване на authorize атрибут за ограничване достъп до определени actions и controllers според ролята на логнатия потребител.

Създаване на Action attribute който измерва времето за изпълнение на всеки метод.

Отваряме проекта MVCFilters.

Този проект е същия като проекта FormLogin .Само сме добавили 2 нови екрана:

Home/About и Admin/Index. Също така само 2 потребителя могат да се логнат - user и admin (с произволни пароли)

Това може да го ограничим като сложим проверка в ~/Views/Shared/\_Layout.cshtml

@if (LoginUserSession.Current.IsAdministrator)

{

<li>

@Html.ActionLink("Admins", "Index", "Admin")

</li>

}

Сега ако сме влзезли с “user” не виждаме Admin меню-то, но ако го напишем в URL-a: “/Admin” то можем да влезем. Как да забраним това?

За целта се използват допълнителни атрибути и MVC filters.

<http://www.dotnet-stuff.com/tutorials/aspnet-mvc/understanding-asp-net-mvc-filters-and-attributes>

-----------------

Създаваме клас “CustomAuthorizeAttribute : FilterAttribute, IAuthorizationFilter”  и имплементираме IauthorizationFilter

public class CustomAuthorizeAttribute : FilterAttribute, IAuthorizationFilter

{

public bool AllowAccessToUser { get; set; }

public void OnAuthorization(AuthorizationContext filterContext)

{

// check if the current session is not authenticated - then redirect to login

if (LoginUserSession.Current.IsAuthenticated == false)

{

filterContext.Result = new RedirectToRouteResult(

new RouteValueDictionary(new { controller = "Home", action = "Login" }));

}

// then check if the current user should be able to access the action

else if (AllowAccessToUser == false && LoginUserSession.Current.IsAdministrator == false)

{

filterContext.Result = new ViewResult { ViewName = "AccessDenied" };

}

}

}

--------------

Добавяме AccessDenied.cshtml в Views/Shared

-----------------

Слагаме [CustomAuthorize] на AdminController/ Index

[CustomAuthorize]

public ActionResult Index()

{

return View();

}

И пробваме,

Но е по-добре да се сложи на целия AdminController

[CustomAuthorize]

public class AdminController : Controller

{

// GET: Admin

public ActionResult Index()

{

return View();

}

}

---------------

Добавяме атрибут “[CustomAuthorize(AllowAccessToUser = true)]” на Home/Index() за да не се вижда ако няма активен(логнат) потребител.

[CustomAuthorize(AllowAccessToUser = true)]

public ActionResult Index()

{

return View();

}

Пробваме и след това го махаме, т.к.

За да не го добавяме атрибута на всеки action, може да го добавим на HomeController-a и слагаме [AllowAnonymous] на Login() и LoginPost(). Този атрибут може да се сложи и в Application\_Start() който се намира във файла Global.asax.cs

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

RouteConfig.RegisterRoutes(RouteTable.Routes);

CustomAuthorizeAttribute authorizeAttribute = new CustomAuthorizeAttribute();

authorizeAttribute.AllowAccessToUser = true;

GlobalFilters.Filters.Add(authorizeAttribute);

}

Но за да работи трябва в класа CustomAuthorizeAttribute, в началото на метода OnAuthorization() да се сложи проверката:

if (filterContext.ActionDescriptor.GetCustomAttributes(typeof(AllowAnonymousAttribute), true).Any()

|| filterContext.ActionDescriptor.ControllerDescriptor.GetCustomAttributes(typeof(AllowAnonymousAttribute), true).Any())

{

return;

}

============================================

Добавяме следващ вид атрибут - Action attribute

Показваме OutputCache атрибута как работи като го слагаме на Home/Index():

[HttpGet]

[AllowAnonymous]

[OutputCache(Duration = 1, VaryByParam = "none")]

public ActionResult Login()

{

return View();

}

---------------

Създаваме клас “CustomActionFilterAttribute : FilterAttribute, IActionFilter”.   
Целта му е да измерваме времето за което ще се изпълни всеки action method и ще го покажем в Layout.cshtml

public class CustomActionFilterAttribute : FilterAttribute, IActionFilter

{

public void OnActionExecuting(ActionExecutingContext filterContext)

{

filterContext.HttpContext.Items.Add("StartTime", DateTime.Now.Ticks);

System.Threading.Thread.Sleep(1); // sleep for at least 1ms - to have some delay for each action

}

public void OnActionExecuted(ActionExecutedContext filterContext)

{

long startTime = (long)filterContext.HttpContext.Items["StartTime"];

TimeSpan period = new TimeSpan(DateTime.Now.Ticks - startTime);

string duration = string.Format("{0}s {1}ms", period.Seconds, period.Milliseconds);

filterContext.Controller.ViewBag.Period = duration;

}

}

-------------------

Добавяме @ViewBag.Period в \_Layout.cshtml

....

</div>

@ViewBag.Period

<div class="container body-content">

@RenderBody()

....

-----------------

В Admin/Index() добавяме 800ms заспиване:

public ActionResult Index()

{

System.Threading.Thread.Sleep(800);

return View();

}

-------------------

* Начини на добавяне на filter attributes
  + GlobalFilters.Filters.Add(new CustomActionFilterAttribute());
  + Като атрибут на action method
  + Като атрибут на action controller

------------

Ние ще добавим в

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

RouteConfig.RegisterRoutes(RouteTable.Routes);

CustomAuthorizeAttribute authorizeAttribute = new CustomAuthorizeAttribute();

authorizeAttribute.AllowAccessToUser = true;

GlobalFilters.Filters.Add(authorizeAttribute);

🡪 GlobalFilters.Filters.Add(new CustomActionFilterAttribute());

}

-------------------

Пробваме като се логнем с admin.

Също може да отидем на изгледа Admins за да видим заспиването с 800ms.

==========================================================================

Ще покажем пример за глобално прихващане на exceptions.

Добавяме линк в Admin/Index.cshtml

@Html.ActionLink("Crash link", "CrashLink", "Admin")

И един action в AdminController:

public ActionResult CrashLink()

{

throw new Exception("CrashLink is not working!");

//return View();

}

Стартираме и виждаме че грешката не е подходяща за потребителя.

Даже може да видим, че не се прихваща и хвърленото изключение. Но за това трябва да се добави във файла Web.config един допълнителен таг:

<system.web>

<.../>

<.../>

--> <customErrors mode="Off" />

</system.web>

Добавяме клас “CustomErrorHandlerAttribute : FilterAttribute, IExceptionFilter” във Helpers и написваме (копираме) кода

public class CustomErrorHandlerAttribute : FilterAttribute, IExceptionFilter

{

public void OnException(ExceptionContext filterContext)

{

// log the exception to a log file

// redirect to the ServerError.cshtml view

filterContext.Result = new ViewResult { ViewName = "ServerError" };

(filterContext.Result as ViewResult).ViewBag.ErrorMessage = filterContext.Exception.Message;

// it is important to mark the exception as handled

filterContext.ExceptionHandled = true;

// this is not needed, just add it for better understanding

filterContext.HttpContext.Response.Clear();

filterContext.HttpContext.Response.StatusCode = 500;

filterContext.HttpContext.Response.TrySkipIisCustomErrors = true;

}

}

-----------------------

Добавяме view ServerError.cshtml във ~/View/Shared

@{

ViewBag.Title = "ServerError";

}

<h2>There was a server error</h2>

<h3>The error message is: @ViewBag.ErrorMessage</h3>

---------------------------------

Добавяме го да се извика при Application\_Start()

protected void Application\_Start()

{

...

-> GlobalFilters.Filters.Add(new CustomErrorHandlerAttribute());

}

и няма нужда да се добавя за всеки метод допълнителен атрибут.