Universitatea Tehnica a Moldovei
Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator Nr. 5
la Tehnologii Web
Tema: Accesul utilizatorilor

A efectuat — St. gr. TI-216
Rosca Dorin

A verificat — Asist. univ.
Gaidarji Alina

Chișinău 2023

Sarcina lucrării de laborator: Implementarea restricțiilor privind accesul utilizatorilor la anumite pagini ale site-ului (admin, utilizator).

Teorie:

Pentru a implementa mecanismul de restricționare a anumitor grupuri de utilizatori la anumite acțiuni disponibile pe site, este necesar să se utilizeze mecanisme de atribute, și anume atributul ActionFilterAttribute. Atributul Admin Mod va permite să restricționăm accesul nu la toate acțiunile controlerului și numai la anumite acțiuni ale controlorului. Pentru a utiliza atributul Admin Mod, trebuie să adăugați o adnotare înaintea numelui metodei sau clasei. În același timp, deoarece metodele OnActionExecuting ale acestor atribute trebuie să fie executate secvențial, mai întâi de către AdminMod.

Realizarea sarcinii:

Adaugarea filtrelor în mapa Filters ca în figura 1. Aceste filtre restrictioneaza accesul utilizatorilor la diverse acțiuni din controlere. Filtrul UserMod restrictioneaza accesul utilizatorilor neautentificați.

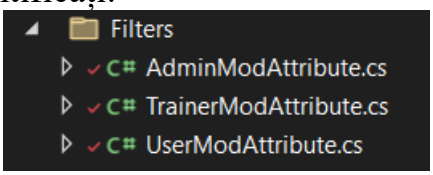


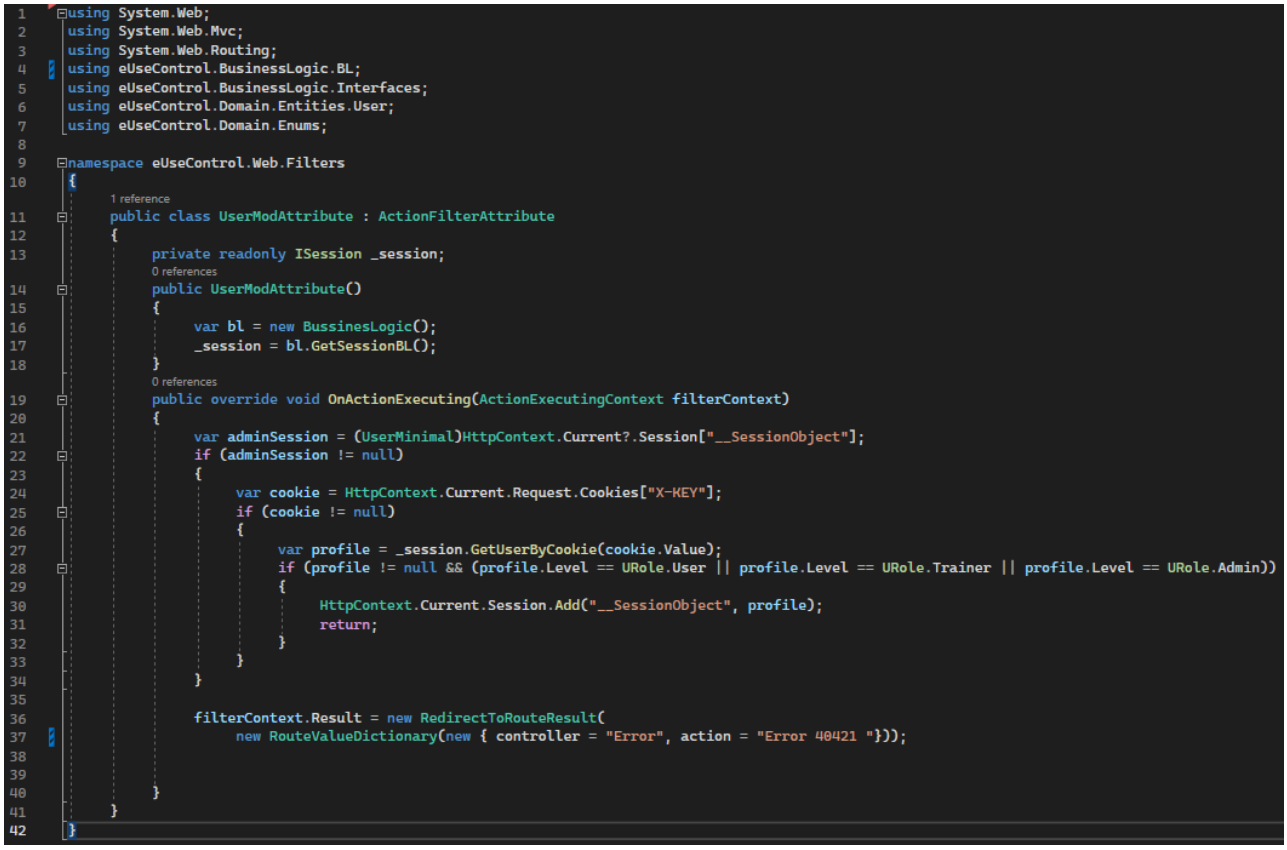
Figura 1. Filters folder

```
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
using eUseControl.BusinessLogic.BL;
using eUseControl.BusinessLogic.Interfaces;
using eUseControl.Domain.Entities.User;
using eUseControl.Domain.Enums;

namespace eUseControl.Web.Filters
{
    1 reference
    public class UserModAttribute : ActionFilterAttribute
    {
        private readonly ISession _session;
        0 references
        public UserModAttribute()
        {
            var bl = new BussinesLogic();
            _session = bl.GetSessionBL();
        }
        0 references
        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            var adminSession = (UserMinimal)HttpContext.Current?.Session["__SessionObject"];
            if (adminSession != null)
            {
                var cookie = HttpContext.Current.Request.Cookies["X-KEY"];
                if (cookie != null)
                {
                    var profile = _session.GetUserByCookie(cookie.Value);
                    if (profile != null && (profile.Level == URole.User || profile.Level == URole.Trainer || profile.Level == URole.Admin))
                    {
                        HttpContext.Current.Session.Add("__SessionObject", profile);
                        return;
                    }
                }
            }

            filterContext.Result = new RedirectToRouteResult(
                new RouteValueDictionary(new { controller = "Error", action = "Error 40421 "}));


        }
    }
}
```

Figura 2. UserModAttribute

Clasa UserModAttribute din figura 2 oferă un mecanism de verificare și validare a permisiunilor utilizatorilor înainte de a permite accesul la anumite acțiuni dintr-un controller.

ActionFilterAttribute este o clasă în limbajul de programare C# care permite aplicarea de filtre asupra acțiunilor (metodelor) dintr-un controller într-o aplicație web. Aceasta este parte a framework-ului ASP.NET și face parte din infrastructura de filtrare a acțiunilor.



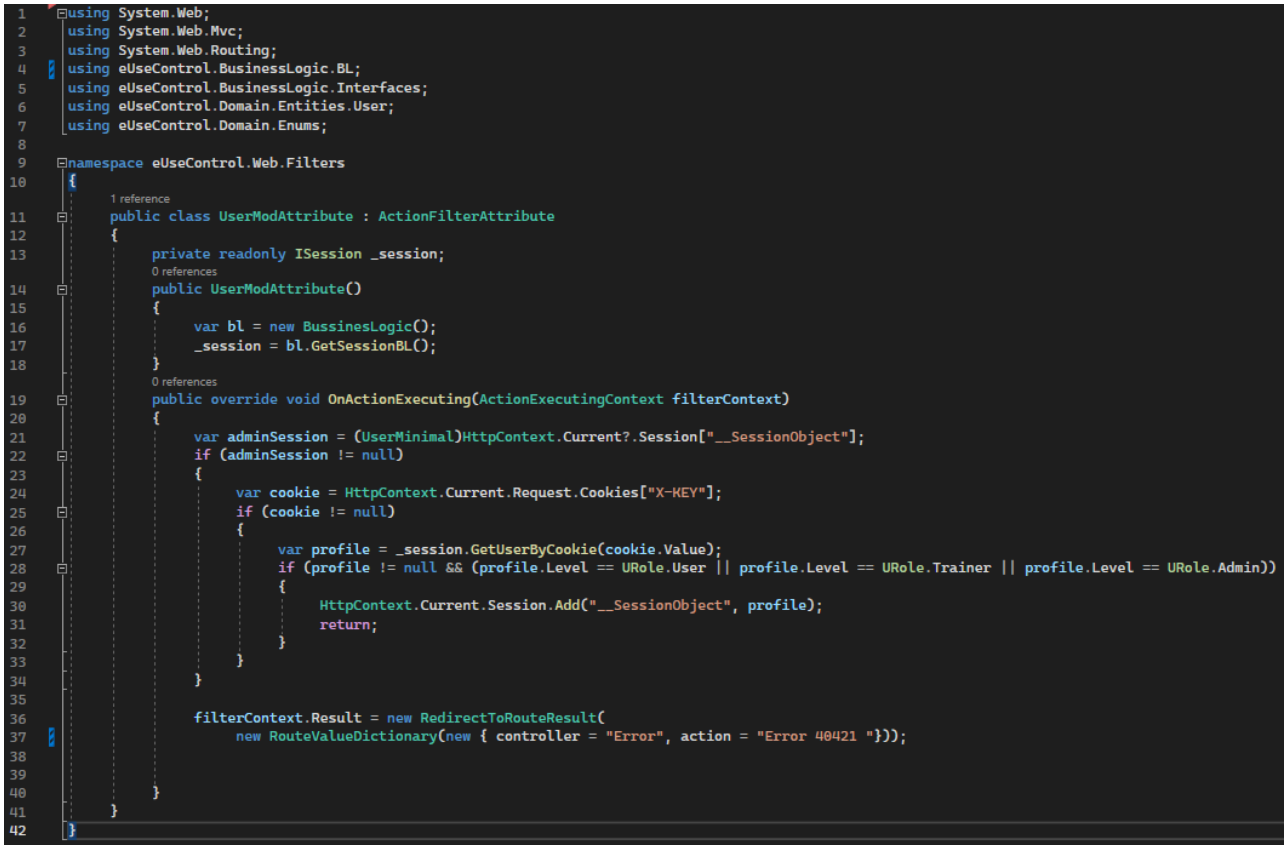
```
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
using eUseControl.BusinessLogic.BL;
using eUseControl.BusinessLogic.Interfaces;
using eUseControl.Domain.Entities.User;
using eUseControl.Domain.Enums;

namespace eUseControl.Web.Filters
{
    1 reference
    public class UserModAttribute : ActionFilterAttribute
    {
        private readonly ISession _session;
        0 references
        public UserModAttribute()
        {
            var bl = new BussinesLogic();
            _session = bl.GetSessionBL();
        }
        0 references
        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            var adminSession = (UserMinimal)HttpContext.Current?.Session["__SessionObject"];
            if (adminSession != null)
            {
                var cookie = HttpContext.Current.Request.Cookies["X-KEY"];
                if (cookie != null)
                {
                    var profile = _session.GetUserByCookie(cookie.Value);
                    if (profile != null && (profile.Level == URole.User || profile.Level == URole.Trainer || profile.Level == URole.Admin))
                    {
                        HttpContext.Current.Session.Add("__SessionObject", profile);
                        return;
                    }
                }
            }

            filterContext.Result = new RedirectToRouteResult(
                new RouteValueDictionary(new { controller = "Error", action = "Error 40421 "}));


        }
    }
}
```

Figura 3. AdminModAttribute

Clasa AdminModAttribute la fel ca și UserModAttribute verifică și validează permisiunile utilizatorului.

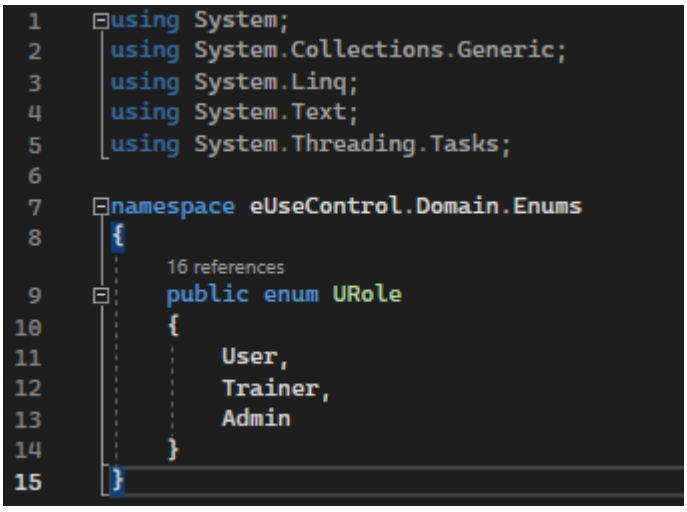
Constructorul clasei AdminModAttribute inițializează o variabilă de instanță _session de tipul ISession folosind o instanță a clasei BussinesLogic și metoda GetSessionBL(). Acesta pare să fie un mecanism personalizat pentru a obține o sesiune.

Metoda OnActionExecuting este suprascrisă pentru a furniza logica de filtrare. Această metodă este apelată înainte de executarea acțiunii în controller.



```
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
using eUseControl.BusinessLogic.BL;
using eUseControl.BusinessLogic.Interfaces;
using eUseControl.Domain.Entities.User;
using eUseControl.Domain.Enums;

namespace eUseControl.Web.Filters
{
    1 reference
    public class UserModAttribute : ActionFilterAttribute
    {
        private readonly ISession _session;
        0 references
        public UserModAttribute()
        {
            var bl = new BussinesLogic();
            _session = bl.GetSessionBL();
        }
        0 references
        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            var adminSession = (UserMinimal)HttpContext.Current?.Session["__SessionObject"];
            if (adminSession != null)
            {
                var cookie = HttpContext.Current.Request.Cookies["X-KEY"];
                if (cookie != null)
                {
                    var profile = _session.GetUserByCookie(cookie.Value);
                    if (profile != null && (profile.Level == URole.User || profile.Level == URole.Trainer || profile.Level == URole.Admin))
                    {
                        HttpContext.Current.Session.Add("__SessionObject", profile);
                        return;
                    }
                }
            }

            filterContext.Result = new RedirectToRouteResult(
                new RouteValueDictionary(new { controller = "Error", action = "Error 40421 "}));


        }
    }
}
```

Figura 4. TrainerModAttribute



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace eUseControl.Domain.Enums
{
    16 references
    public enum URole
    {
        User,
        Trainer,
        Admin
    }
}
```

Figura 5. Urole.cs

Enum-ul URole din figura 5 are trei valori definite: User, Trainer și Admin. Aceste valori reprezintă nivelurile de acces posibile pentru utilizatori în aplicație. Enum-ul poate fi utilizat pentru a specifica și verifica nivelurile de acces ale utilizatorilor în diverse părți ale codului, cum ar fi în logica de autentificare, permisiuni și validări.

Exemplu al utilizarii rolurilor:

```
        //Create a new Subscription
        [AdminMod]

        public ActionResult Create()
        {
            GetUserData();

            return View();
        }
        [AdminMod]
        [HttpPost]

        public ActionResult Create(Subscription subscription)
        {
            GetUserData();

            SubscriptionUDbTable data = new SubscriptionUDbTable()
            {
                Id = subscription.Id,
                Name = subscription.Name,
                Description = subscription.Description,
                Price = subscription.Price,
                ImageUrl = subscription.ImageUrl
            };

            var subscriptionCreate :PostResponse = _subscription.CreateSubscription(data);
            if (subscriptionCreate.Status)
            {
                return RedirectToAction("Index", controllerName: "Subscription");
            }
            else
            {
                ModelState.AddModelError(key: "", errorMessage: subscriptionCreate.StatusMsg);
                return RedirectToAction("Index", controllerName: "Error");
            }
        }
```

Figura 6.
Codul ce reprezinta ActionMethod de creare a unui nou abonament

- In Cazul in care Pagini e accesata de utilizator cu nivelul de acces Admin,rezultatul va fi:

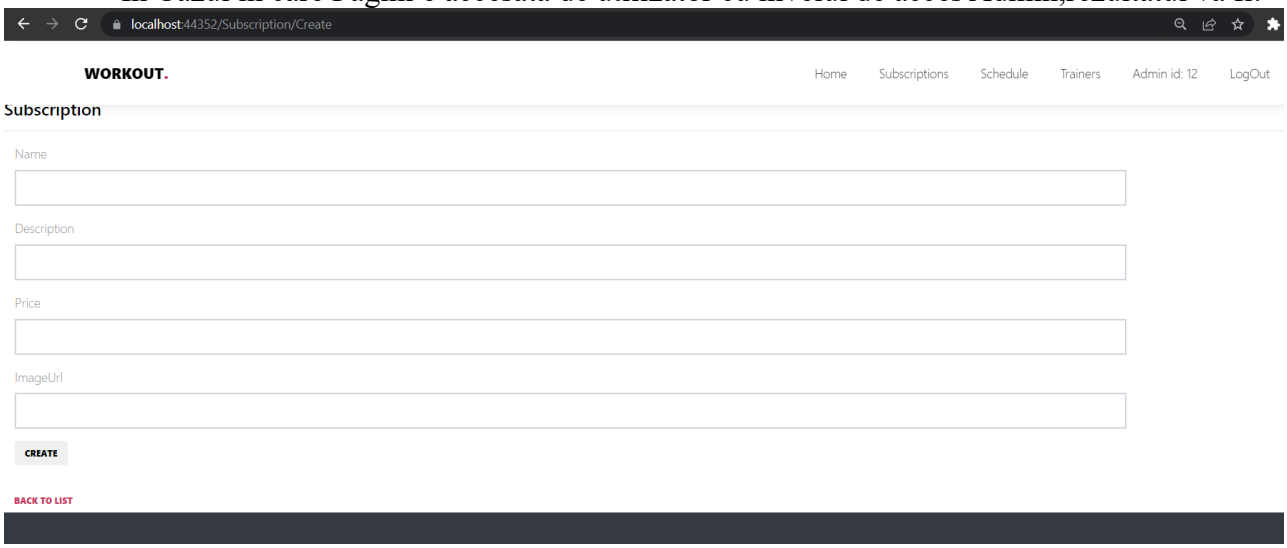


Figura 7.Creare nou Abonament cu nivelul de Acces Admin

- In cazul in Care Nivelul de acces e mai mic decat Admin,Rezultatul va fi

Figura 8. Creare nou Abonament cu nivelul de Acces mai mic decat Admin

Concluzii:

În cadrul acestei lucrări de laborator am creat filtre care vor restricționa accesul utilizatorilor care nu au anumite privilegii la acțiunile respective.

Implementarea restricțiilor privind accesul utilizatorilor la anumite pagini ale unui site poate fi realizată folosind diverse tehnici și funcționalități specifice limbajului și framework-ului utilizat pentru dezvoltarea aplicației web.

Unul dintre modurile comune de implementare a restricțiilor de acces este prin utilizarea atributelor de filtru de acțiune (ActionFilterAttribute). Atributele de filtru de acțiune pot fi aplicate la nivelul acțiunilor dintr-un controller și pot furniza logica de validare și verificare a permisiunilor utilizatorului înainte de a permite accesul la o anumită pagină sau acțiune.