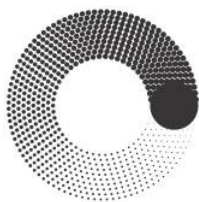


ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Факультет Информационных технологий  
Кафедра Информатики и информационных технологий*

направление подготовки 09.03.02  
«Информационные системы и технологии»

## ЛАБОРАТОРНАЯ РАБОТА № 15

Дисциплина: BackEnd-разработка

---

Тема: контроль доступа к адресам в веб-приложение на основе ASP.NET Core

---

Выполнил(а): студент(ка) группы 221-3711

Костоваров А. С.

(Фамилия И.О.)

Дата, подпись \_\_\_\_\_

(Дата)

(Подпись)

Проверил: \_\_\_\_\_

(Фамилия И.О., степень, звание)

(Оценка)

Дата, подпись \_\_\_\_\_

(Дата)

(Подпись)

Замечания: \_\_\_\_\_

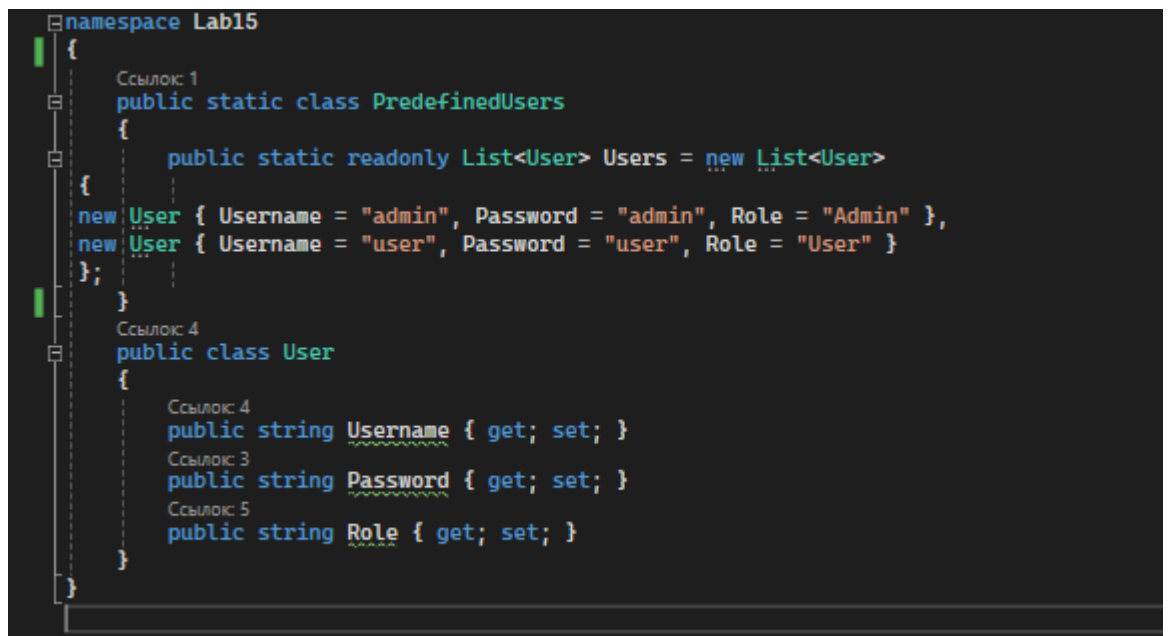
---

Москва

2024

В данной работе мы реализуем механизм перенаправления запросов для контроля доступа через middleware в веб-приложении на платформе ASP.NET Core. Создаем новое веб приложение.

Были созданы две роли: Admin и User. Для каждой роли были определены права доступа:



```
namespace Lab15
{
    // Ссылка 1
    public static class PredefinedUsers
    {
        public static readonly List<User> Users = new List<User>
        {
            new User { Username = "admin", Password = "admin", Role = "Admin" },
            new User { Username = "user", Password = "user", Role = "User" }
        };
    }

    // Ссылка 4
    public class User
    {
        // Ссылка 4
        public string Username { get; set; }
        // Ссылка 3
        public string Password { get; set; }
        // Ссылка 5
        public string Role { get; set; }
    }
}
```

Admin: Пользователи с этой ролью имеют доступ к разделу администратора, расположенный по адресу /Admin.

User: Пользователи с этой ролью имеют доступ к разделу пользователя, расположенный по адресу /User. Доступ к разделу администратора для них ограничен и контролируется с помощью middleware перенаправляющее на страницу запрета доступа.

```

using Lab15.Middleware;
using Microsoft.AspNetCore.Rewrite;
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews();
builder.Services.AddAuthentication("CookieAuth")
    .AddCookie("CookieAuth", options =>
    {
        options.Cookie.Name = "UserLoginCookie";
        options.LoginPath = "/Account/Login";
        options.AccessDeniedPath = "/Account/AccessDenied";
    });
builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("AdminPolicy", policy => policy.RequireRole("Admin"));
    options.AddPolicy("UserPolicy", policy => policy.RequireRole("User"));
});
var app = builder.Build();
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
var rewriteOptions = new RewriteOptions()
    .AddRedirect("^Admin$", "Admin/Index")
    .AddRedirect("^User$", "User/Index");
app.UseRewriter(rewriteOptions);
app.UseRoleCheck();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Account}/{action=Login}/{id?}");
app.Run();

```

В файле `program` включаем аутентификацию при помощи куки, а также авторизацию, добавляем 2 политики доступа для админа и пользователя. После этого добавляем созданный нами `middleware` для контроля доступа и так же промежуточное ПО для перенаправления запросов к контроллерам юзера и админа на их главные страницы.

```

namespace Lab15.Middleware
{
    Ссылка: 2
    public class RoleCheckMiddleware
    {
        private readonly RequestDelegate _next;
        Ссылка: 0
        public RoleCheckMiddleware(RequestDelegate next)
        {
            _next = next;
        }
        Ссылка: 0
        public async Task InvokeAsync(HttpContext context)
        {
            if (context.User.Identity.IsAuthenticated)
            {
                if (!context.User.IsInRole("Admin") &&
                    context.Request.Path.StartsWithSegments("/Admin"))
                {
                    context.Response.Redirect("/Account/AccessDenied");
                    return;
                }
            }
            await _next(context);
        }
    }
    Ссылка: 0
    public static class RoleCheckMiddlewareExtensions
    {
        Ссылка: 1
        public static IApplicationBuilder UseRoleCheck(this IApplicationBuilder
            builder)
        {
            return builder.UseMiddleware<RoleCheckMiddleware>();
        }
    }
}

```

Далее разберем промежуточное ПО для ограничения доступа, для каждого запроса мы получаем аутентификацию пользователя и проверяем его роль, если пользователь не имеет достаточно прав доступа, то он будет перенаправлен на страницу запрета доступа.

```

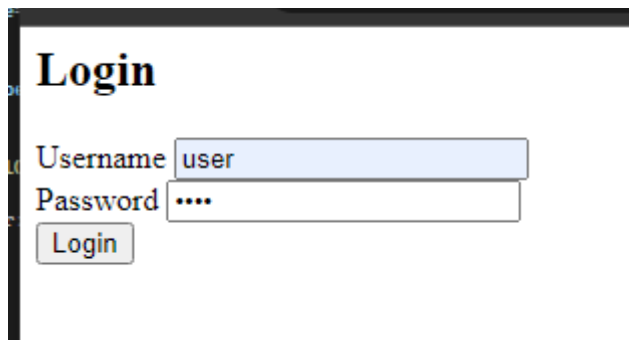
namespace Lab15.Controllers
{
    Ссылка: 0
    public class AccountController : Controller
    {
        [HttpGet]
        Ссылка: 0
        public IActionResult Login()
        {
            return View();
        }
        [HttpPost]
        Ссылка: 0
        public async Task Login(string username, string password)
        {
            var user = PredefinedUsers.Users.FirstOrDefault(u => u.Username ==
            username && u.Password == password);
            if (user != null)
            {
                var claims = new List<Claim>
                {
                    new Claim(ClaimTypes.Name, user.Username),
                    new Claim(ClaimTypes.Role, user.Role)
                };
                var claimsIdentity = new ClaimsIdentity(claims, "CookieAuth");
                var claimsPrincipal = new ClaimsPrincipal(claimsIdentity);
                await HttpContext.SignInAsync("CookieAuth", claimsPrincipal);
                if (user.Role == "Admin")
                {
                    return RedirectToAction("Index", "Admin");
                }
                else if (user.Role == "User")
                {
                    return RedirectToAction("Index", "User");
                }
            }
            ViewBag.Message = "Неверное имя пользователя или пароль.";
            return View();
        }
        [Authorize]
        Ссылка: 0
        public async Task Logout()
        {
            await HttpContext.SignOutAsync("CookieAuth");
            return RedirectToAction("Login");
        }
        Ссылка: 0
        public IActionResult AccessDenied()
        {
            return View();
        }
    }
}

```

В контроллере для входа создаем действие по входу, получаем заранее созданных пользователь со своими ролями и если введенные данные корректны, мы аутентифицируем пользователя. А так же перенаправляем на соответствующую роли страницу.

Так же создадим действия для выхода из аккаунта и действие, на которое будем перенаправлять при попытке получить доступ к запрещенному действию.

Протестируем работу приложения.

A screenshot of a web application's login page. The page has a dark header bar. Below it, the word "Login" is displayed in a large, bold, serif font. Underneath, there are two input fields: "Username" with the text "user" and "Password" with four dots. A "Login" button is positioned below the password field.

**Login**

Username user

Password ....

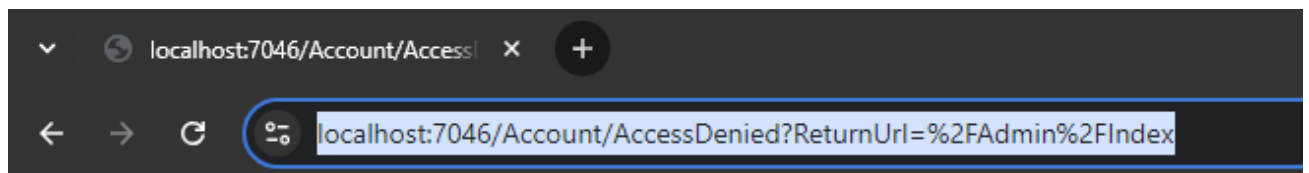
Login

Войдем в аккаунт юзера

**User**

К админу нельзя /Admin

Попадаем на нужную страницу, попробуем перейти на страницу админа.



**Access Denied**

У вас нет прав для доступа к этой странице.

При попытке перейти на запрещенную страницу, мы попадаем на страницу запрета доступа.

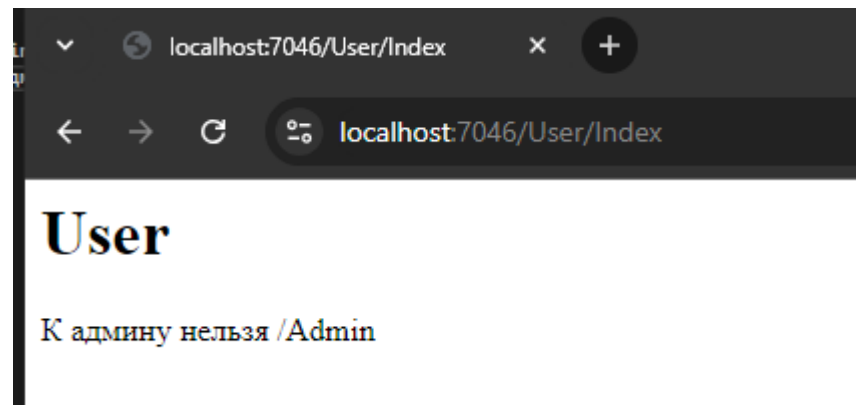
Попробуем зайти под админом, у которого нет ограничений в доступе.

A screenshot of a web application's admin page. The page has a dark header bar. Below it, the word "Admin" is displayed in a large, bold, serif font. Underneath, the text "Вы Админ" is displayed in a smaller, bold, serif font.

**Admin**

**Вы Админ**

Теперь перейдем к юзеру.



Мы успешно перешли, т.к промежуточное ПО разрешает доступ админу ко всему.

## Код

### Program

```
using Lab15.Middleware;
using Microsoft.AspNetCore.Rewrite;
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllersWithViews();
builder.Services.AddAuthentication("CookieAuth")
    .AddCookie("CookieAuth", options =>
    {
        options.Cookie.Name = "UserLoginCookie";
        options.LoginPath = "/Account/Login";
        options.AccessDeniedPath = "/Account/AccessDenied";
    });
builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("AdminPolicy", policy => policy.RequireRole("Admin"));
    options.AddPolicy("UserPolicy", policy => policy.RequireRole("User"));
});
var app = builder.Build();
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
var rewriteOptions = new RewriteOptions()
    .AddRedirect("^Admin$", "Admin/Index")
    .AddRedirect("^User$", "User/Index");
app.UseRewriter(rewriteOptions);
app.UseRoleCheck();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Account}/{action=Login}/{id?}");
app.Run();
```

### Roles

```
namespace Lab15
{
    public static class PredefinedUsers
    {
        public static readonly List<User> Users = new List<User>
        {
            new User { Username = "admin", Password = "admin", Role = "Admin" },
            new User { Username = "user", Password = "user", Role = "User" }
        };
    }
    public class User
    {
        public string Username { get; set; }
        public string Password { get; set; }
        public string Role { get; set; }
    }
}
```



Middelware

```
namespace Lab15.Middleware
{
    public class RoleCheckMiddleware
    {
        private readonly RequestDelegate _next;
        public RoleCheckMiddleware(RequestDelegate next)
        {
            _next = next;
        }
        public async Task InvokeAsync(HttpContext context)
        {
            if (context.User.Identity.IsAuthenticated)
            {
                if (!context.User.IsInRole("Admin") &&
                    context.Request.Path.StartsWithSegments("/Admin"))
                {
                    context.Response.Redirect("/Account/AccessDenied");
                    return;
                }
            }
            await _next(context);
        }
    }
    public static class RoleCheckMiddlewareExtensions
    {
        public static IApplicationBuilder UseRoleCheck(this IApplicationBuilder
            builder)
        {
            return builder.UseMiddleware<RoleCheckMiddleware>();
        }
    }
}
```

AccountController

```
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace Lab15.Controllers
{
    public class AccountController : Controller
    {
        [HttpGet]
        public IActionResult Login()
        {
            return View();
        }
        [HttpPost]
        public async Task<IActionResult> Login(string username, string password)
        {
            var user = PredefinedUsers.Users.FirstOrDefault(u => u.Username ==
                username && u.Password == password);
            if (user != null)
            {
                var claims = new List<Claim>
                {
                    new Claim(ClaimTypes.Name, user.Username),
                    new Claim(ClaimTypes.Role, user.Role)
                };
                var claimsIdentity = new ClaimsIdentity(claims, "CookieAuth");
                var claimsPrincipal = new ClaimsPrincipal(claimsIdentity);
                await HttpContext.SignInAsync("CookieAuth", claimsPrincipal);
            }
        }
    }
}
```

```

        if (user.Role == "Admin")
        {
            return RedirectToAction("Index", "Admin");
        }
        else if (user.Role == "User")
        {
            return RedirectToAction("Index", "User");
        }
    }
    ViewBag.Message = "Неверное имя пользователя или пароль.";
    return View();
}
[Authorize]
public async Task<IActionResult> Logout()
{
    await HttpContext.SignOutAsync("CookieAuth");
    return RedirectToAction("Login");
}
public IActionResult AccessDenied()
{
    return View();
}
}
}
}

```

AdminController

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
namespace Lab15.Controlles
{
    [Authorize(Policy = "AdminPolicy")]
    public class AdminController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

UserController

```

using Microsoft.AspNetCore.Mvc;
namespace Lab15.Controlles
{
    public class UserController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```