Održavanje stanja

Održavanje stanja

- Kada korisnik pristupi web sajtu, web server šalje stranu koju je korisnik zahtevao
- Ako korisnik zatraži drugu stranu, web server je locira i vraća je korisniku
- Sa stanovišta servera, ova dva zahteva su potpuno nezavisna
- Ovo predstavlja problem ukoliko su na drugoj strani potrebni podaci uneti na prvoj strain
- Potreban je mehanizam za čuvanje informacija koje predstavljaju stanje korisnika
- Browser može slati te informacije kao deo svakog zahteva, čime server može prepoznati korisnika

Cookie

- Cookie je tekstualni fajl koji se čuva na računaru klijenta u folderu pridruženom web browseru dok korisnik pregleda web sajt
- Svaki browser ima svoje kolačiće
- Najčešća primena kolačića je identifikacija korisnika prilikom posete više strana web aplikacije
- Cookie se automatski šalje web serveru kao deo svakog HTTP zahteva
- Postoje dve vrste cookie-ja:
  - privremeni cookie – vreme života je ograničeno na korisničku sesiju
  - stalni (perzistentni) cookie – vreme života je unapred definisano

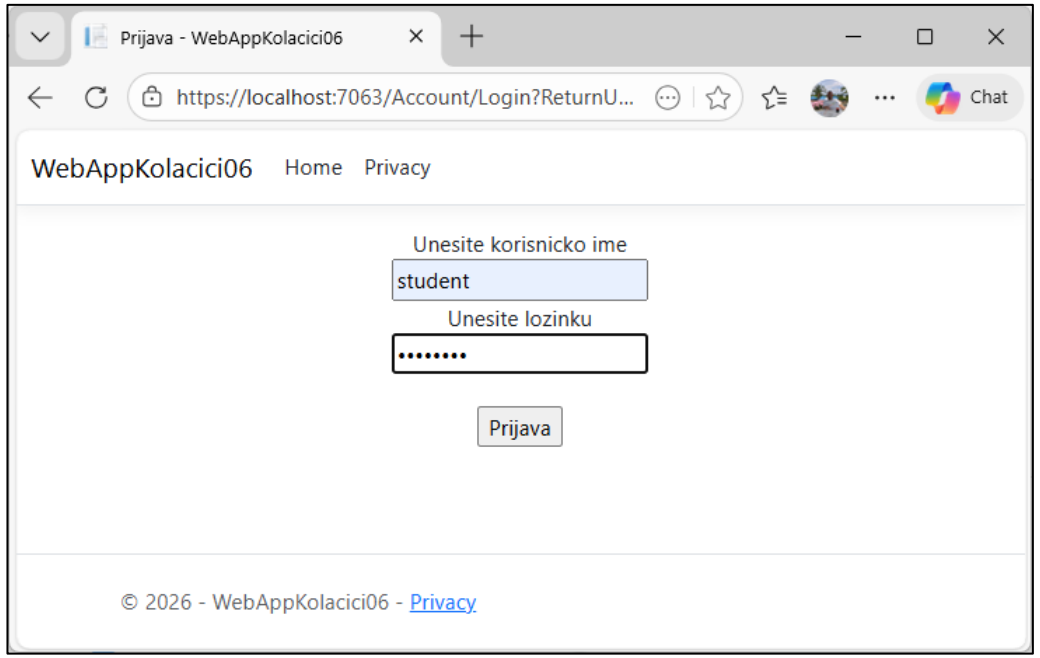
Cookie

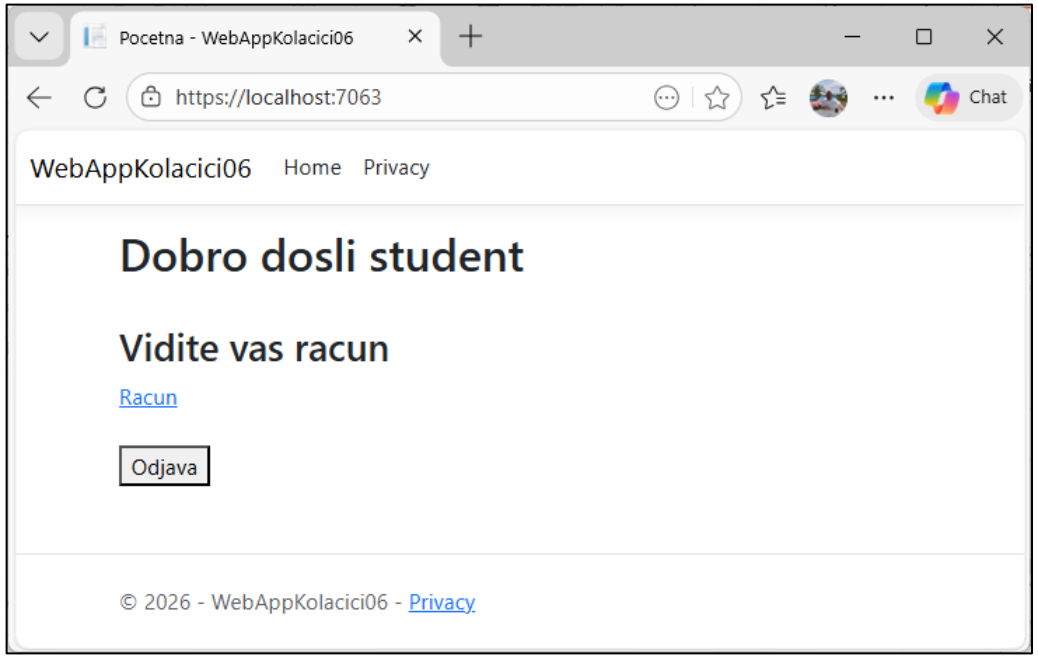
- Maksimalna veličina cookie-ja je 4 KB
- Kolačići se šalju između browsera i web servera kao deo svakog HTTP zahteva i odgovora
- Podaci sačuvani u kolačićima dele se između tabova istog browsera
- Kolačić se može kreirati i čitati pomoću klijentskog koda
- **HttpOnly** cookie se ne može čitati pomoću klijentskog koda i kreira se uz pomoć serverskog koda

Praćenje klijenta uz korišćenje kolačića

- U koraku 1 klijent zahteva stranu od web servera, i pošto ranije nije posećivao sajt, ne poseduje kolačić
- U koraku 2 web server, pored odgovora, šalje korisniku kolačić
- Klijent šalje kolačić uz svaki sledeći zahtev, čime biva identifikovan (koraci 3 i 4)

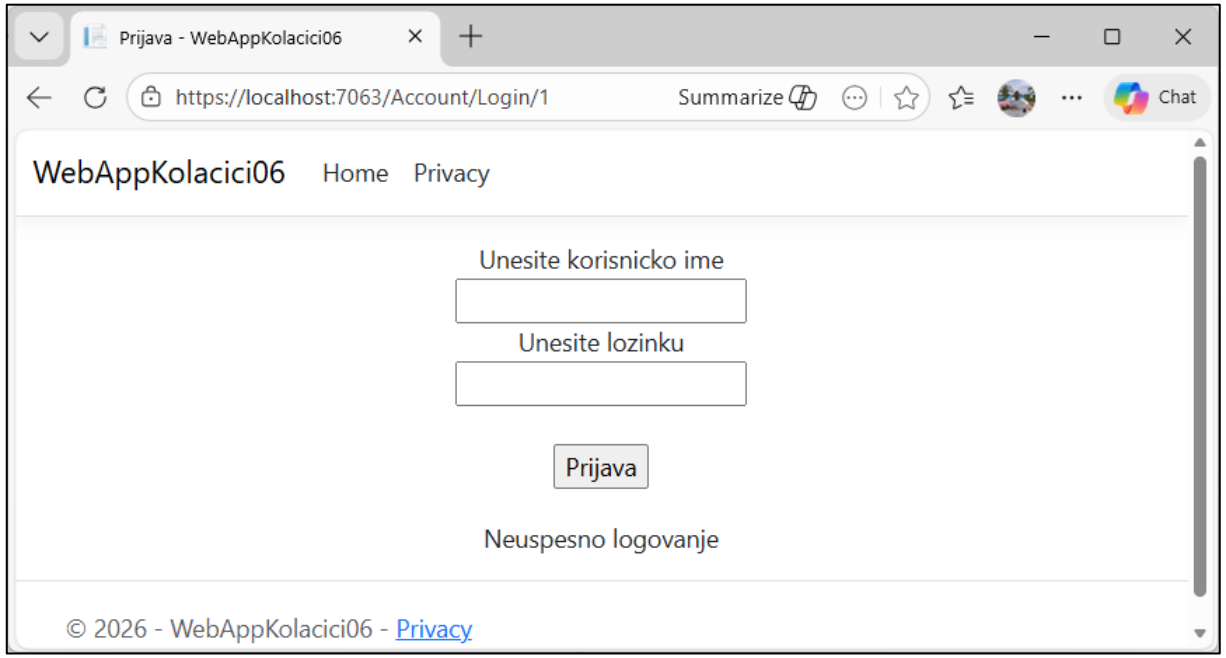
Prijava na sistem





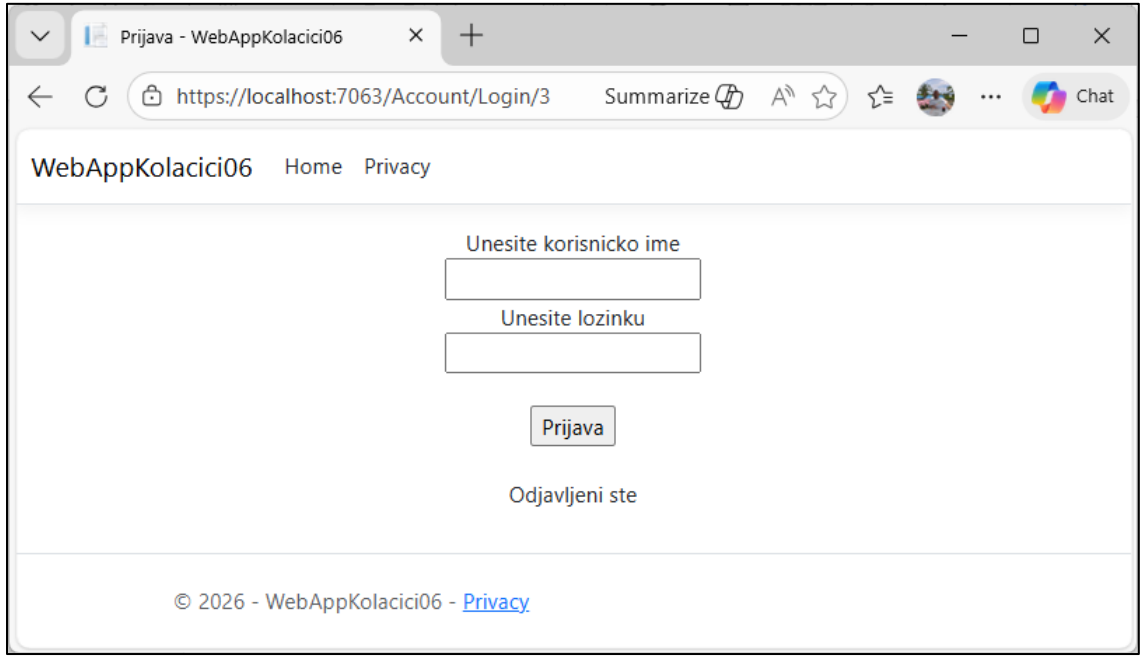
Prikazana je **Login** strana za unos korisničkih podataka, a nakon uspešne prijave prikazuje se zaštićena strana **Index (HomeController)**

Neuspesna prijava



Nakon neuspešne prijave korisnik ostaje na **Login** strani, a u URL adresi (/Account/Login/1) parametar 1 označava neuspešno logovanje

Odjava



Nakon odjave korisnik se preusmerava na Login stranu, što se vidi u URL adresi: /Account/Login/3

Baza podataka LogovanjeMVC

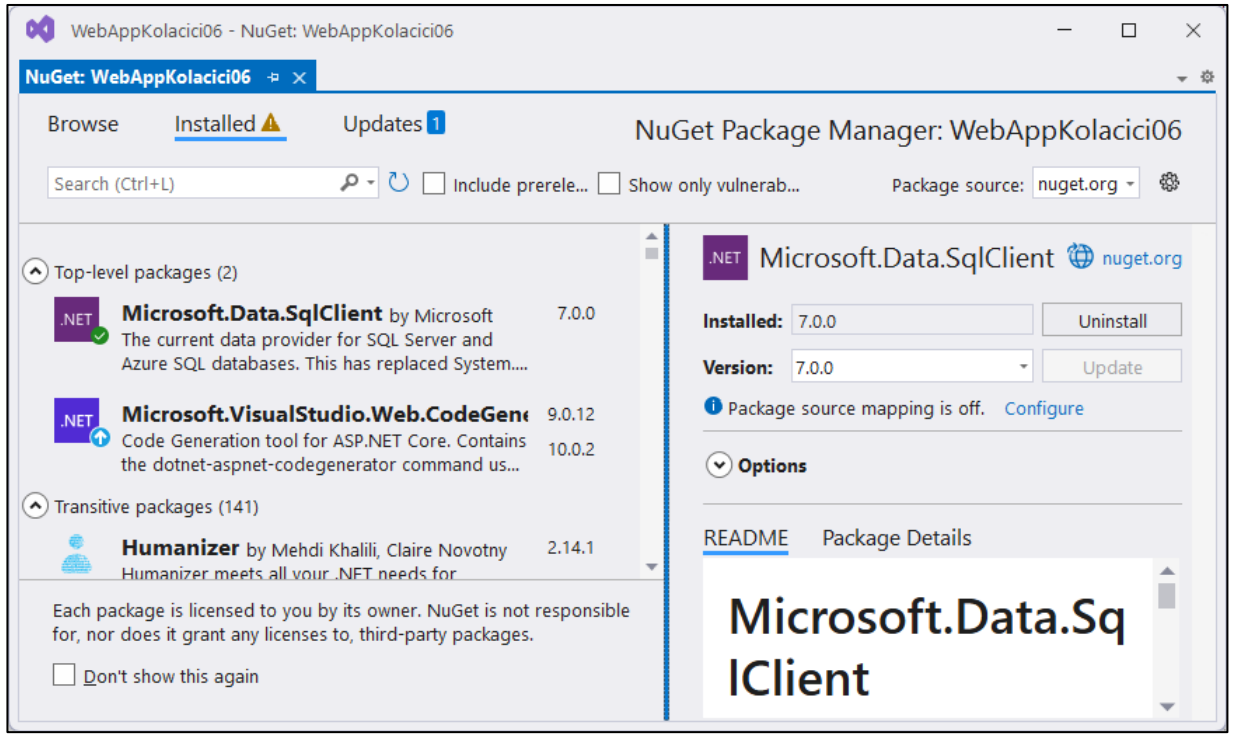
```
CREATE DATABASE LogovanjeMVC
COLLATE Serbian_Latin_100_CS_AS
GO

USE LogovanjeMVC
GO

CREATE TABLE Korisnik(
KorisnikId int NOT NULL IDENTITY(1,1) PRIMARY KEY,
KorisnickoIme nvarchar(50) NOT NULL  UNIQUE,
Lozinka nvarchar(20) NOT NULL,
Provera UNIQUEIDENTIFIER  DEFAULT NEWID()
);

INSERT INTO Korisnik(KorisnickoIme,Lozinka)
VALUES('student','student1');
```

KorisnikId	KorisnickoIme	Lozinka	Provera
1	student	student1	AC761011-C027-48B7-A2F2-BF72EA959805

Entitetska klasa Korisnik (Models)

```
public class Korisnik
    {
        public int KorisnikId { get; set; }
        public string KorisnickoIme { get; set; }
        public string Lozinka { get; set; }

        public Guid Provera { get; set; }
    }
```

Biblioteka Microsoft.Data.SqlClient

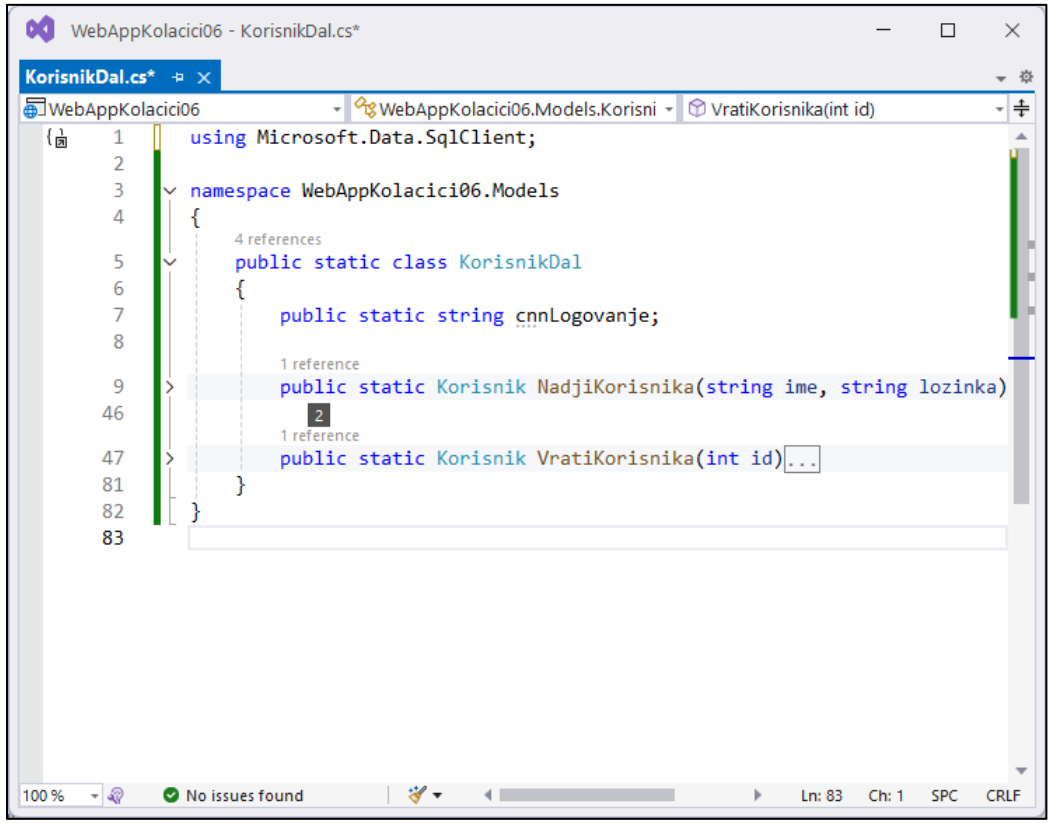


Biblioteka **Microsoft.Data.SqlClient** koristi se za povezivanje .NET aplikacije sa SQL Server bazom podataka i izvršavanje SQL upita

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Server=GORAN-HP;Initial Catalog=LogovanjeMVC;
     Integrated Security=true;Encrypt=False"
  }
}
```

Klasa KorisnikDal(Models)



```
public static Korisnik NadjiKorisnika(string ime, string lozinka)
{
    string upit = @"SELECT KorisnikId, KorisnickoIme, Lozinka, Provera
                    FROM Korisnik
                    WHERE KorisnickoIme = @ime
                        AND Lozinka = @lozinka
                    COLLATE Serbian_Latin_100_CS_AS";

    using SqlConnection konekcija = new SqlConnection(cnnLogovanje);
    using SqlCommand komanda = new SqlCommand(upit, konekcija);
    komanda.Parameters.AddWithValue("@ime", ime);
    komanda.Parameters.AddWithValue("@lozinka", lozinka);

    try
    {
        konekcija.Open();
        using SqlDataReader dr = komanda.ExecuteReader();

        if (dr.Read())
        {
            return new Korisnik
            {
                KorisnikId = dr.GetInt32(0),
                KorisnickoIme = dr.GetString(1),
                Lozinka = dr.GetString(2),
                Provera = dr.GetGuid(3)
            };
        }
        return null;
    }
    catch
    {
        return null;
    }
}
```

```
public static bool PostojiKorisnik(int id, Guid provera)
{
    string upit = @"SELECT COUNT(*) FROM Korisnik
            WHERE KorisnikId = @KorisnikId AND Provera = @Provera";

    using (SqlConnection konekcija = new SqlConnection(cnnLogovanje))
    {
        SqlCommand komanda = new SqlCommand(upit, konekcija);
        komanda.Parameters.AddWithValue("@KorisnikId", id);
        komanda.Parameters.AddWithValue("@Provera", provera);

        try
        {
            konekcija.Open();
            int postojiKorisnik = (int)komanda.ExecuteScalar();

            if (postojiKorisnik == 1)
            {
                return true;
            }

            return false;
        }
        catch
        {
            return false;
        }
    }
}
```

```
public static Korisnik VratiKorisnika(int id)
{
    string upit = @"SELECT KorisnikId, KorisnickoIme, Lozinka, Provera
                    FROM Korisnik
                    WHERE KorisnikId = @id";
    using SqlConnection konekcija = new SqlConnection(cnnLogovanje);
    using SqlCommand komanda = new SqlCommand(upit, konekcija);
    komanda.Parameters.AddWithValue("@id", id);

    try
    {
        konekcija.Open();
        using SqlDataReader dr = komanda.ExecuteReader();
        if (dr.Read())
        {
            return new Korisnik
            {
                KorisnikId = dr.GetInt32(0),
                KorisnickoIme = dr.GetString(1),
                Lozinka = dr.GetString(2),
                Provera = dr.GetGuid(3)
            };
        }
        return null;
    }
    catch
    {
        return null;
    }
}
```

Kontroler za autentifikaciju

```
public class AccountController : Controller
{
    public AccountController(IConfiguration configuration)
    {
        KorisnikDal.cnnLogovanje =
        configuration.GetConnectionString("DefaultConnection");
    }
}
```

Svojstvo HttpContext

- Može mu se pristupiti unutar akcione metode kontrolera
- Vraća **HttpContext** objekat koji sadrži sve informacije o jednom HTTP zahtevu
- Svojstvo HttpContext.User daje informaciju o korisniku koji upućuje zahtev
- Svojstvo **Request** ovog objekta vraća **HttpRequest** objekat i sadrži informacije o korisničkom zahtevu
- Svojstvo **Response** vraća **HttpResponse** objekat i sadrži informacije koje server šalje klijentu kao odgovor
- Svojstva **Request** i **Response** mogu se koristiti i bez prethodnog pozivanja HttpContext svojstva
- Svojstvo **HttpContext.Session** vraća objekat koji omogućava rad sa korisničkom sesijom

Metoda KreirajKolacic AccountController

```
private void KreirajKolacic(string kljuc, string vrednost, int? vremeVazenja = null)
    {
        CookieOptions opcije = new CookieOptions();

        opcije.IsEssential = true;
        opcije.HttpOnly = true;

        if (vremeVazenja.HasValue)
        {
            opcije.Expires = DateTime.Now.AddMinutes(vremeVazenja.Value);
        }

        Response.Cookies.Append(kljuc, vrednost, opcije);
    }
```

- Metoda kreira kolačić i dodaje ga u HTTP odgovor, pri čemu se kroz **CookieOptions** definišu njegova svojstva (npr. trajanje i bezbednost)
- IsEssential = true znači da je kolačić neophodan za osnovno funkcionisanje aplikacije

Metoda ObrisiKolacic AccountController

```
private void ObrisiKolacic(string kljuc)
{
    Response.Cookies.Delete(kljuc);
}
```

Metoda briše kolačić iz HTTP odgovora i koristi se prilikom odjave korisnika

Akcija Login (GET) kontroler Account

```
[HttpGet]
public IActionResult Login(int id = 0)
{
    ViewBag.id = id;
    return View();
}
```

- Akcija se poziva kada korisnik pristupi Login stranici putem HTTP GET zahteva
- Parametar id se koristi za prosleđivanje poruke o statusu (npr. neuspešna prijava, odjava)
- Vrednost parametra se prosleđuje View-u preko ViewBag objekta
- Metoda vraća odgovarajući View za unos korisničkih podataka

Login.cshtml

```
@{
    ViewData["Title"] = "Prijava";
}
<div class="text-center">
    <form  asp-action="Login" method="post">
        @Html.AntiForgeryToken()

        <label for="TextIme">Unesite korisnicko ime</label>
        <br />
        <input type="text" name="ime" id="TextIme" />
        <br />
        <label for="TextLozinka">Unesite lozinku</label>
        <br />
        <input type="password" name="lozinka" id="TextLozinka" />
        <br />
        <input type="submit" value="Prijava" />
    </form>
    <div style="margin-top:20px;">
        @if (ViewBag.id == 1)
        {
            <span>Neuspesno logovanje</span>
        }
        @if (ViewBag.id == 2)
        {
            <span>Morate se prijaviti na sistem</span>
        }
        @if (ViewBag.id == 3)
        {
            <span>Odjavljeni ste</span>
        }
    </div>
</div>
```

- Stranica omogućava unos korisničkog imena i lozinke i slanje podataka na Login akciju kontrolera
- Na osnovu vrednosti prosleđene kroz ViewBag.id prikazuje se odgovarajuća poruka (neuspešna prijava, obavezna prijava ili odjava)

Metoda Login POST

```
[HttpPost]
public IActionResult Login(string ime, string lozinka)
{
    Korisnik k = KorisnikDal.NadjiKorisnika(ime, lozinka);

    if (k != null)
    {
        // kreiranje kolačića
        KreirajKolacic("id", k.KorisnikId.ToString());
        KreirajKolacic("provera", k.Provera.ToString());

        // preusmeravanje na zaštićenu stranu
        return RedirectToAction("Index", "Home");
    }
    else
    {
        // neuspešna prijava → vraća na Login sa porukom
        return RedirectToAction("Login", new { id = 1 });
    }
}
```

Metoda Odjava kontroler Account

```
[HttpPost]
public ActionResult Odjava()
{
    ObrisiKolacic("id");
    ObrisiKolacic("provera");

    return RedirectToAction("Login", new { id = 3 });
}
```

Metoda Autentifikacija AuthHelper

```
namespace WebAppKolacici06.Helpers
{
    public static class AuthHelper
    {
        public static bool Autentifikacija(HttpRequest request)
        {
            string id1 = request.Cookies["id"];
            string provera1 = request.Cookies["provera"];

            if (id1 != null && provera1 != null)
            {
                Guid provera = Guid.Parse(provera1);
                int id = int.Parse(id1);

                return KorisnikDal.PostojiKorisnik(id, provera);
            }

            return false;
        }
    }
}
```

- Čitanje podataka iz kolačića
- Provera korisnika u bazi
- Vraćanje rezultata autentifikacije

Metoda Index kontroler Home

```
public IActionResult Index()
{
    if (!AuthHelper.Autentifikacija(Request))
    {
        return RedirectToAction("Login", "Account", new { id = 2 });
    }

    int id = int.Parse(Request.Cookies["id"]);
    Korisnik k1 = KorisnikDal.VratiKorisnika(id);

    return View(k1);
}
```

- Metoda proverava da li je korisnik autentifikovan na osnovu kolačića
- U slučaju da korisnik nije prijavljen, vrši se preusmeravanje na Login stranu
- Ako je autentifikacija uspešna, podaci o korisniku se učitavaju iz baze i prosleđuju pogledu

Pogled Index.cshtml

```
@model WebAppKolacici06.Models.Korisnik

@{
    ViewData["Title"] = "Početna";
}

<h1>Dobro došli @Model.KorisnickoIme</h1>

<br />

<h2>Vidite vaš račun</h2>
<a asp-action="Racun">Račun</a>

<br />
<br />

<form asp-controller="Account" asp-action="Odjava" method="post">
    <button type="submit">Odjava</button>
</form>
```

Metoda Racun kontroler Home

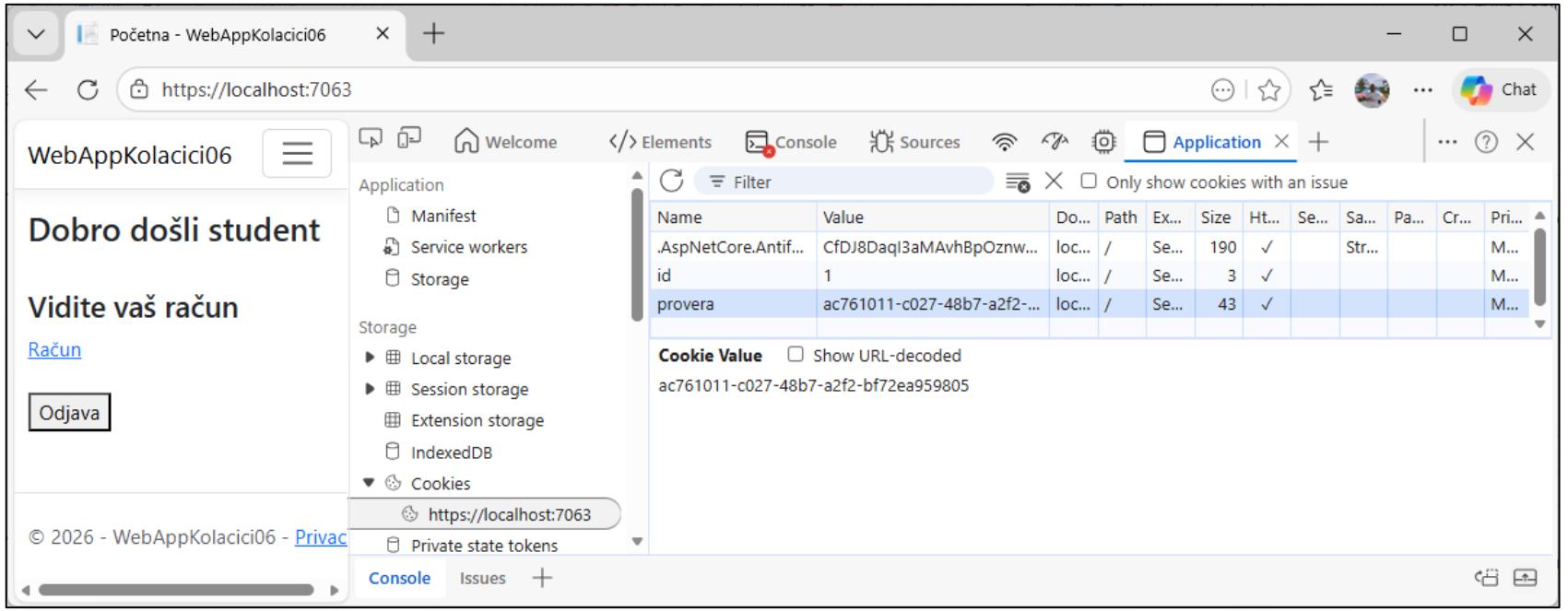
```
public IActionResult Racun()
{
    if (!AuthHelper.Autentifikacija(Request))
    {
        return RedirectToAction("Login", "Account", new { id = 2 });
    }

    return View();
}
```

- Metoda predstavlja zaštićenu stranu i pristup joj je dozvoljen samo autentifikovanim korisnicima
- U slučaju da korisnik nije prijavljen, vrši se preusmeravanje na Login stranu

Pogled Racun.cshtml

```
@{
    ViewData["Title"] = "Račun";
}

<h1>Vaš račun</h1>

<br />

<p>Ovo je zaštićena stranica dostupna samo prijavljenim
korisnicima.</p>

<br />

<form asp-controller="Account" asp-action="Odjava"
method="post">
    <button type="submit">Odjava</button>
</form>

<a asp-action="Index">Nazad na početnu</a>
```

Prikaz kolačića u browseru



Autentifikacija pomoću kolačića (ASP.NET Core)

- ASP.NET Core omogućava automatsko logovanje korisnika pomoću kolačića
- Nije potrebno ručno kreirati i proveravati kolačiće
- Framework preuzima brigu o autentifikaciji korisnika
- Podaci o korisniku čuvaju se u kolačiću na bezbedan način

Konfigurisanje autentifikacije

- Da bi se omogućila autentifikacija, potrebno je konfigurisati odgovarajući servis u aplikaciji
- Autentifikacija se podešava u fajlu Program.cs
- Definiše se način autentifikacije pomoću kolačića
- Konfigurišu se osnovna svojstva kolačića (naziv, trajanje, putanja za login)

Registracija servisa za autentifikaciju i autorizaciju

```
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Account/Login";
        options.AccessDeniedPath = "/Account/Login";
        options.Cookie.Name = "LogovanjeMVC.Auth";
        options.Cookie.HttpOnly = true;
        options.Cookie.IsEssential = true;
        options.Cookie.SameSite = SameSiteMode.Lax;
        options.ExpireTimeSpan = TimeSpan.FromMinutes(30);
        options.SlidingExpiration = true;
    });

builder.Services.AddAuthorization();
```

Konfigurisanje middleware-a za autentifikaciju

```
app.UseStaticFiles();
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}")
    .WithStaticAssets();


app.Run();
```

- Middleware definiše redosled obrade HTTP zahteva u aplikaciji
- Metoda UseAuthentication() vrši identifikaciju korisnika na osnovu kolačića
- Metoda UseAuthorization() proverava prava pristupa korisnika
- Redosled pozivanja middleware-a je važan za ispravno funkcionisanje aplikacije

Osnovne klase za autentifikaciju

- Klasa **Claim** predstavlja pojedinačnu informaciju o korisniku
- Klasa **ClaimTypes** sadrži unapred definisane tipove podataka (npr. ime, identifikator)
- Klasa **ClaimsIdentity** predstavlja identitet korisnika
- Klasa **ClaimsPrincipal** predstavlja korisnika u aplikaciji
- Navedene klase nalaze se u okviru .NET biblioteke **System.Security.Claims**

Definisanje informacija o korisniku (claims)

```
var claims = new List<Claim>
{
    new Claim(ClaimTypes.NameIdentifier, k.KorisnikId.ToString()),
    new Claim(ClaimTypes.Name, k.KorisnickoIme)
};
```

- Kreira se kolekcija claims koja sadrži podatke o korisniku
- Svaki **claim** predstavlja jednu informaciju o korisniku
- Najčešće se koriste unapred definisani tipovi iz klase **ClaimTypes**
- Podaci iz claims koriste se za identifikaciju korisnika u aplikaciji

Kreiranje identiteta i prijava

```
var identity = new ClaimsIdentity(
    claims,
    CookieAuthenticationDefaults.AuthenticationScheme);

var principal = new ClaimsPrincipal(identity);

await HttpContext.SignInAsync(
    CookieAuthenticationDefaults.AuthenticationScheme,
    principal);
```

- **ClaimsIdentity** opisuje identitet korisnika
- **AuthenticationScheme** definiše tip autentifikacije (cookie)
- **ClaimsPrincipal** predstavlja korisnika u aplikaciji
- **SignInAsync** kreira autentifikacioni cookie
- Cookie se koristi u svakom sledećem zahtevu

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Login(string ime, string lozinka)
{
    Korisnik k = KorisnikDal.NadjiKorisnika(ime, lozinka);

    if (k == null)
    {
        return RedirectToAction("Login", new { id = 1 });
    }

    var claims = new List<Claim>
    {
    new Claim(ClaimTypes.NameIdentifier, k.KorisnikId.ToString()),
    new Claim(ClaimTypes.Name, k.KorisnickoIme)
    };

    var identity = new ClaimsIdentity(
        claims,
        CookieAuthenticationDefaults.AuthenticationScheme);

    var principal = new ClaimsPrincipal(identity);

    await HttpContext.SignInAsync(
        CookieAuthenticationDefaults.AuthenticationScheme,
        principal);

    return RedirectToAction("Index", "Home");
}
```

- Nakon uspešnog logina kreira se identitet korisnika
- Poziva se SignInAsync()
- .NET automatski kreira autentifikacioni kolačić
- Nema potrebe za ručnim radom sa kolacicima
- Podaci o korisniku dostupni su kroz User svojstvo (HttpContext.User)
- Korisnik je autentifikovan

Odjava korisnika (Logout)

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Odjava()
{
    await HttpContext.SignOutAsync(
        CookieAuthenticationDefaults.AuthenticationScheme);

    return RedirectToAction("Login", new { id = 3 });
}
```

- SignOutAsync uklanja autentifikacioni cookie
- Prekida se važeća autentifikaciona sesija
- [HttpPost] ograničava poziv akcije na POST zahtev
- [ValidateAntiForgeryToken] obezbeđuje zaštitu od falsifikovanja zahteva
- Nakon odjave vrši se redirekcija na login stranicu

Autorizacija Index metode Home kontrolera

```
[Authorize]
public IActionResult Index()
{
    string idTekst = User.FindFirstValue(ClaimTypes.NameIdentifier);

    if (string.IsNullOrEmpty(idTekst))
    {
        return RedirectToAction("Login", "Account", new { id = 2 });
    }

    int id = int.Parse(idTekst);
    Korisnik k1 = KorisnikDal.VratiKorisnika(id);

    if (k1 == null)
    {
        return RedirectToAction("Login", "Account", new { id = 2 });
    }

    return View(k1);
}
```

- **User** (svojstvo HttpContext klase) predstavlja trenutno autentifikovanog korisnika
- Podaci o korisniku čuvaju se u claim-ovima
- FindFirstValue pronalazi vrednost po tipu claim-a
- NameIdentifier sadrži ID korisnikaVrednost se vraća kao string
- [**Authorize**] zahteva postojanje validnog autentifikacionog cookie-ja, ako ne postoji radi se redirekcija na Login stranicu
- [**Authorize**] popunjava User objekat na osnovu cookie-ja

Sesija

Sesija u ASP.NET Core web aplikacijama

- Predstavlja **stanje korisnika** koji posećuje web aplikaciju
- Sesija započinje kada korisnik prvi put pristupi serverskoj metodi koja koristi **session** objekat
- Sesija iščezava kada protekne određeni period vremena (koji se može konfigurisati korišćenjem svojstva IdleTimeout) od poslednjeg zahteva
- ASP.NET Core dodeljuje korisniku kolačić koji sadrži **ID sesije**
- Sesija koja koristi kolačiće ne zauzima memoriju sve dok se podaci ne sačuvaju unutar session objekta
- Podrazumevano se vrednosti sesije čuvaju u **memoriji servera**
- Sesija omogućava deljenje vrednosti između zahteva i čuva informacije specifične za konkretnog korisnika
- Stanje sesije se implementira kao kolekcija **ključ–vrednost** parova

Konfigurisanje sesije

- Da bi se omogućio middleware za sesiju, potrebno je u fajlu Program.cs izvršiti:
  - Implementaciju skladišta za sesiju (sesija se čuva kao objekat na serveru) – **AddDistributedMemoryCache**
  - Pozvati metodu **AddSession**, čime se registruje servis za sesiju
  - Pozvati metodu **UseSession**, čime se uključuje middleware komponenta za rad sa sesijom

Konfigurisanje sesije

```
builder.Services.AddDistributedMemoryCache();
//builder.Services.AddSession();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(10);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
    options.Cookie.Name = "Login";
});

var app = builder.Build();
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseRouting();
app.UseSession();
app.UseAuthorization();
app.MapStaticAssets();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}")
    .WithStaticAssets();
```

Upis podataka u sesiju

- Unutar kontrolera sesiji se pristupa pomoću svojstva **HttpContext.Session**
- **HttpContext.Session.Id** vraća ID sesije dodeljene korisniku
- Metoda **HttpContext.Session.SetString("key", "value")** upisuje string vrednost u sesiju
- Metoda **HttpContext.Session.SetInt32("key", value)** upisuje celobrojnu vrednost u sesiju
- Metoda **HttpContext.Session.GetString("key")** vraća string vrednost sačuvanu u sesiji
- Metoda **HttpContext.Session.GetInt32("key")** vraća vrednost tipa int? koja sadrži celobrojnu vrednost sačuvanu u sesiji
- Metoda **HttpContext.Session.Clear()** briše sadržaj sesije

Id sesije

```
public IActionResult Index()
{
    var BrojSesije = HttpContext.Session.GetString("BrojSesije");

    if (BrojSesije == null)
    {
        BrojSesije = HttpContext.Session.Id;
        HttpContext.Session.SetString("brojSesije", BrojSesije);
    }

    ViewBag.brojSesije = BrojSesije;
    return View();
}
```

Index.cshtml

```
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <span>Sesija: @ViewBag.BrojSesije</span>
    <br />
    <a asp-action="Detalji">Strana2</a>
</div>
```

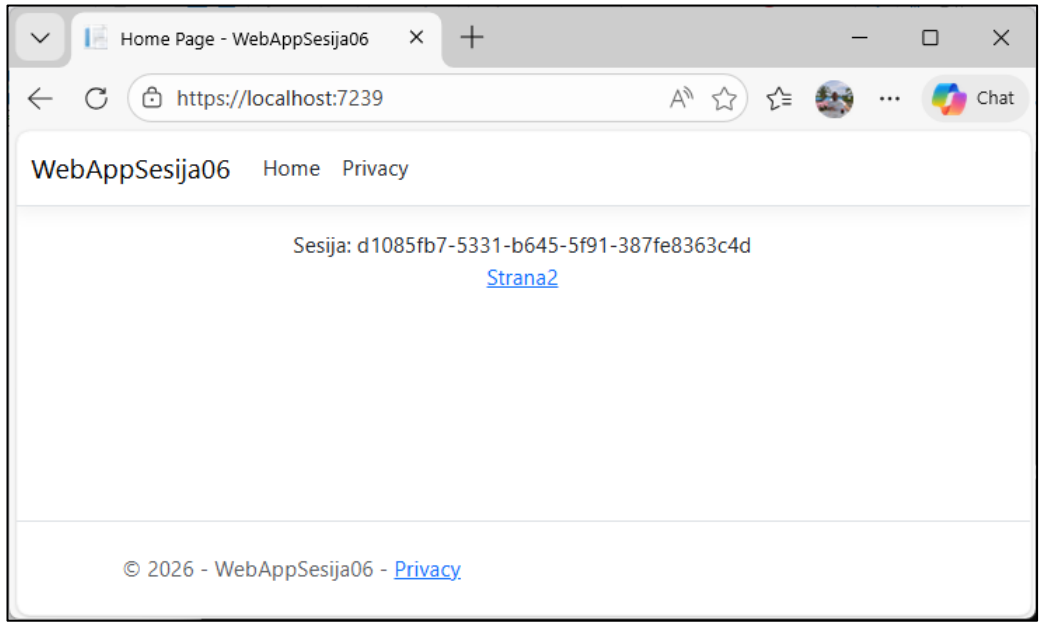
Metoda Detalji

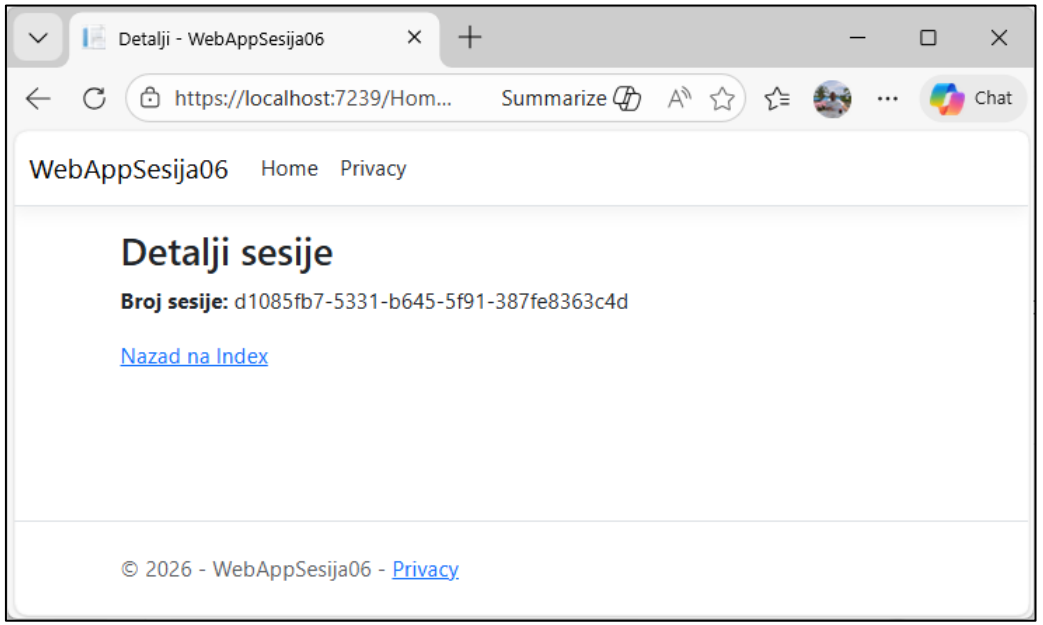
```
public IActionResult Detalji()
{
    var brojSesije = HttpContext.Session.GetString("brojSesije");

    if (brojSesije == null)
        return RedirectToAction("Index");

    ViewBag.BrojSesije = brojSesije;
    return View();
}
```

Pogled Detalji.cshtml

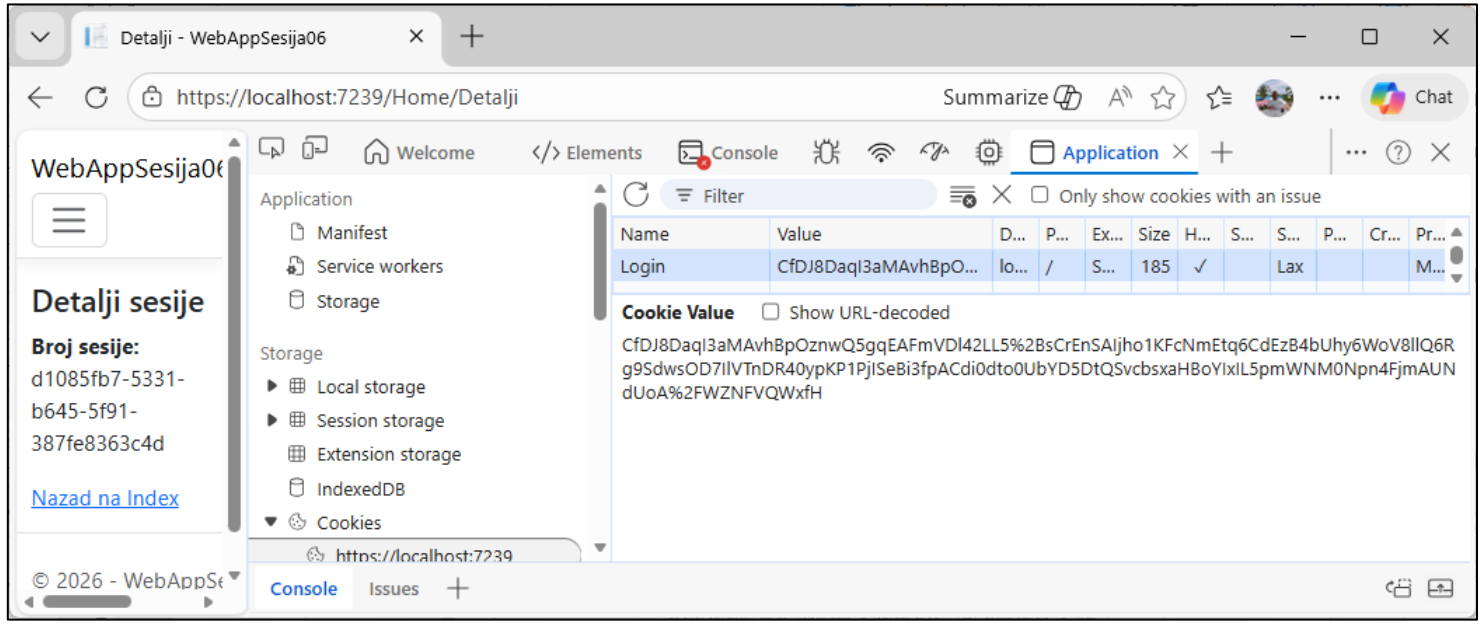
```
@{
    ViewData["Title"] = "Detalji";
}


<h2>Detalji sesije</h2>

<p><strong>Broj sesije:</strong> @ViewBag.BrojSesije</p>

<a asp-action="Index">Nazad na Index</a>
```

Sesija

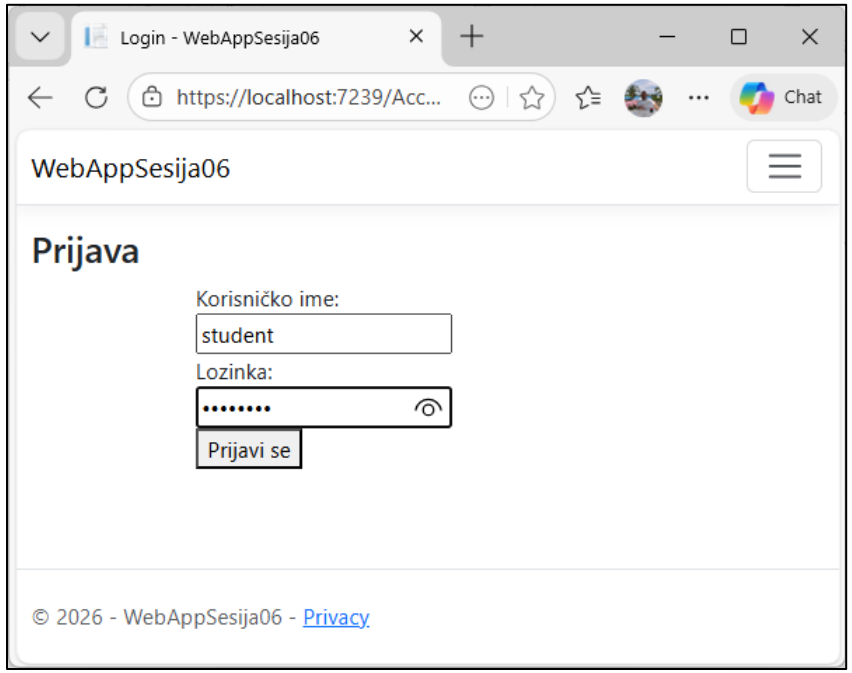

Home Page - WebAppSesija06
https://localhost:7239
Chat
WebAppSesija06
Home
Privacy
Sesija: d1085fb7-5331-b645-5f91-387fe8363c4d
Strana2
© 2026 - WebAppSesija06 - Privacy

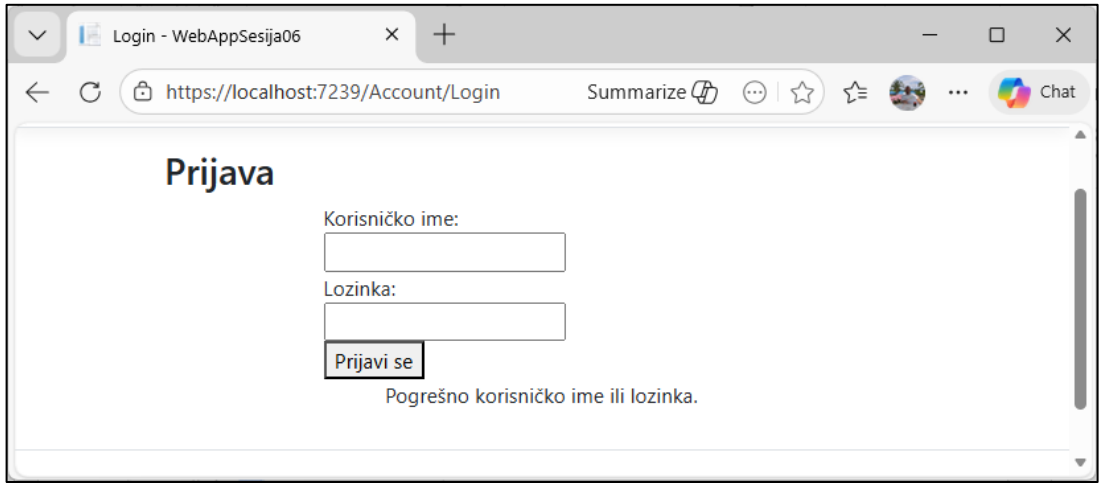

Detalji - WebAppSesija06
https://localhost:7239/Hom...
Summarize
Chat
WebAppSesija06
Home
Privacy
Detalji sesije
Broj sesije: d1085fb7-5331-b645-5f91-387fe8363c4d
Nazad na Index
© 2026 - WebAppSesija06 - Privacy

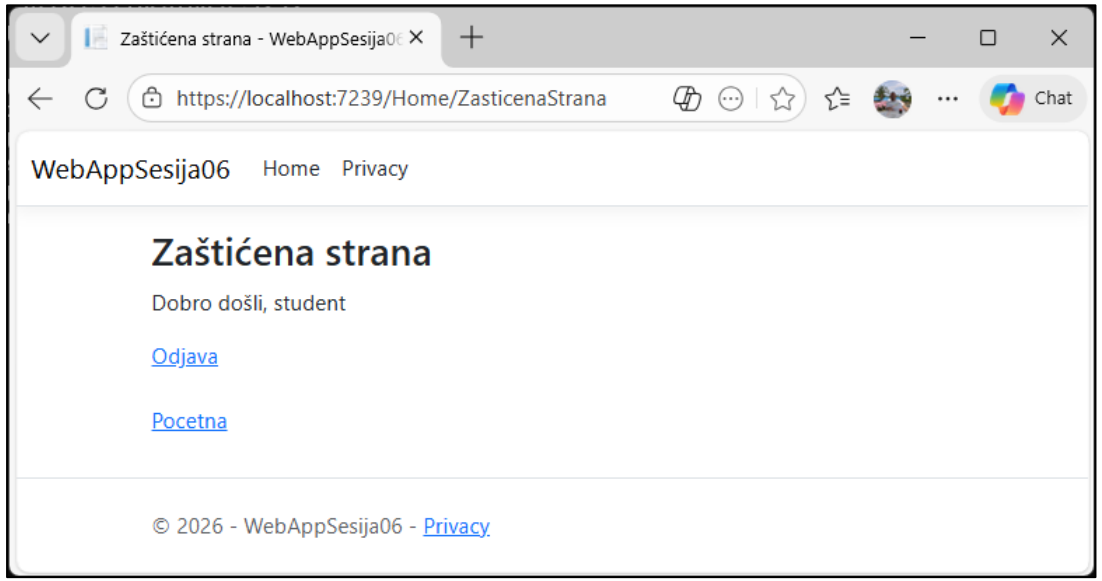
Kolačić sa informacijama o korisničkoj sesiji (sesioni kolačić)



Kreiranje zaštićene strane







AccountController

```
[HttpGet]
public IActionResult Login()
{
    return View();
}

[HttpPost]
public IActionResult Login(string korisnickoIme, string lozinka)
{
    if (korisnickoIme == "student" && lozinka == "student1")
    {
        HttpContext.Session.SetString("ulogovan", "da");
        HttpContext.Session.SetString("korisnik", korisnickoIme);

        return RedirectToAction("ZasticenaStrana","Home");
    }

    ViewBag.Greska = "Pogrešno korisničko ime ili lozinka.";
    return View();
}
```

Login.cshtml

```
<h2>Prijava</h2>

<form asp-action="Login" method="post" style="margin:auto; width: 300px;">
    <div>
        <label>Korisničko ime:</label><br />
        <input type="text" name="korisnickoIme" />
    </div>
    <div>
        <label>Lozinka:</label><br />
        <input type="password" name="lozinka" />
    </div>
    <div>
        <button type="submit">Prijavi se</button>
    </div>
</form>

@if (ViewBag.Greska != null)
{
    <p style="text-align:center;">@ViewBag.Greska</p>
}
```

Kreiranje zaštićene strane

```
public IActionResult ZasticenaStrana()
{
    var ulogovan = HttpContext.Session.GetString("ulogovan");

    if (ulogovan == null)
        return RedirectToAction("Login","Account");

    ViewBag.Korisnik = HttpContext.Session.GetString("korisnik");
    return View();
}
```

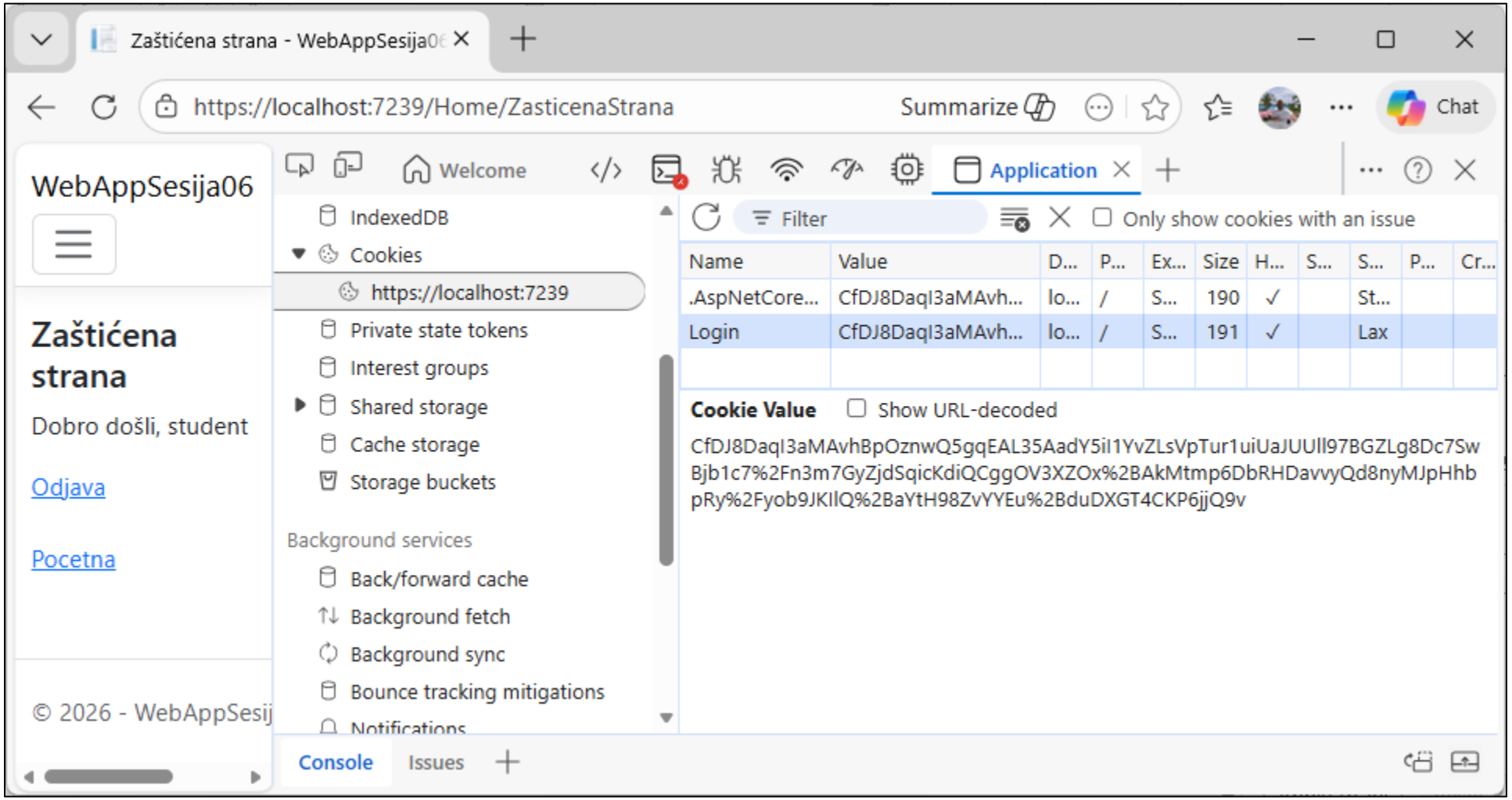
```
@{
    ViewData["Title"] = "Zaštićena strana";
}

<h2>Zaštićena strana</h2>

<p>Dobro došli, @ViewBag.Korisnik</p>

<a asp-controller="Account" asp-action="Logout">Odjava</a> <br /><br />
<a asp-action="Index">Pocetna</a>
```

Uništavanje objekta sesije (AccountController)

```
public IActionResult Logout()
{
    HttpContext.Session.Clear();
    return View();
}
```

```
@{
    ViewData["Title"] = "Odjava";
}

<h2>Uspešno ste se odjavili</h2>

<a asp-controller="Home" asp-action="Index">Pocetna strana</a>
```

Autentifikacioni kolacic



Pitanje 1

Da bismo pristupili objektu sesije unutar metode kontrolera pišemo:

a. Cookie.Session
b. HttpContext.Session
c. Session

Odgovor: b

Pitanje 2

Da bi se sprečilo da klijentski kod pristupi kolačiću, potrebno je postaviti sledeće svojstvo objekta **CookieOptions** na true:

a. ServerOnly
b. HttpOnly
c. Disabled

Odgovor: b

Pitanje 3

Upis string vrednosti u objekat sesije ispravno je izvršen u sledećem primeru:

a. Session.Set("ime","value")
b. HttpContext.Session.SetString(ime,"value")
c. HttpContext.Session.SetString("ime","value")

Odgovor: c

Pitanje 4

Čitanje tekstualne vrednosti koja je upisana u sesiju pod ključem "ime" vrši se na sledeći način:

a. string ime = HttpContext.Session.GetString(ime);
b. sring ime = HttpContext.Session.GetString("ime");
c. string ime = HttpContext.GetString("ime");

Odgovor: b

Pitanje 5

Koja je osnovna uloga kolačića (cookie) u web aplikacijama?

a. Identifikacija korisnika između HTTP zahteva
b. Čuvanje podataka na serveru
c. Ubrzavanje izvršavanja serverskog koda

Odgovor: a

Pitanje 6

Gde se čuvaju podaci sesije u ASP.NET Core aplikaciji, izuzev ID-a sesije?

a. Na klijentu u tekstualnom fajlu
b. U memoriji web servera
c. U headeru htttp zahteva

Odgovor: b