

Improving ASP.NET Core Security By Putting Your Cookies On A Diet



© Tore Nestenius Datakonsult AB. All Rights Reserved.

<https://www.tn-data.se>

1

About Your Instructor

- Name: Tore Nestenius
- Programming for over 40 years
- **1996, www.programmersheaven.com**
A popular website for programmers with over 750,000 monthly visitors.
- **2010, Cofounded of Edument AB**
A consulting and training company.
- **2020, Stack Overflow**
Started to help others!



2

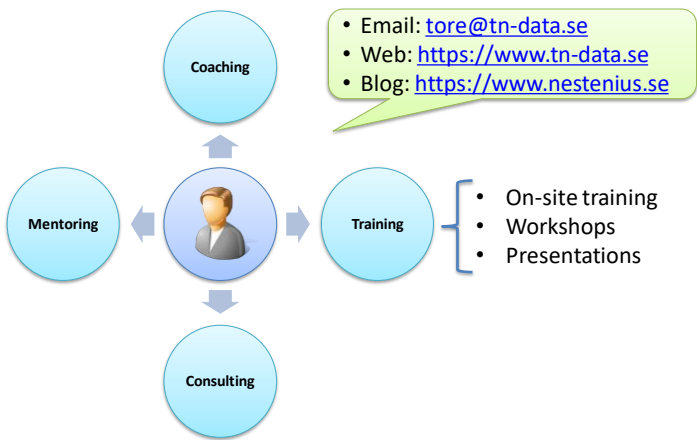
Past Projects – 1987-1993



3

My current occupation

Today, I am self-employed at T.N. Datakonsult AB.



What topics do I focus on?



<https://www.tn-data.se>

4

My training courses

Fundamentals

- Containers and **Kubernetes**
- Introduction to **Git** and **GitHub**

Architecture

- Modern Application **Architecture**
- Service Communication - **REST** vs. **GraphQL** vs. **gRPC**

OpenID Connect and OAuth

- Introduction to **OIDC** and **OAuth**
- Securing ASP.NET Using **OIDC** and **IdentityServer**
- **IdentityServer** in Production

Web

- **Web Security** Fundamentals

.NET

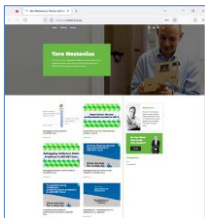
- Applied **Domain-Driven Design** in .NET
- **CQRS** and **Event Sourcing** in .NET
- Building **ASP.NET Core** APIs
- **Asynchronous** Programming in C#
- **C# Expert**
- C# Fundamentals
- TDD.NET

For course details
<https://tn-data.se>

5

My blog

I blog at <https://nstenius.se/>



- Default Azure Credentials Under the Hood
- Improving ASP.NET Core Security By Putting Your Cookies On A Diet
- Demystifying OpenID Connect's State and Nonce Parameters
- Exploring what is inside the ASP.NET Core cookies
- Debugging cookie problems in ASP.NET Core
- BearerToken: The new Authentication handler in .NET 8
- Debugging JwtBearer Claim Problems in ASP.NET Core
- Debugging OpenID Connect Claim Problems in ASP.NET Core
- Troubleshooting JwtBearer authentication problems in ASP.NET Core
- IdentityServer – IdentityResource vs. ApiResource vs. ApiScope
- ASP.NET Core JwtBearer library: what's new?
- How I built my own Sega Mega Drive hardware dev kit from scratch
- .NET 5 Source Generators – MediatR – CQRS – OMG!
- Storing the ASP.NET Core Data Protection Key Ring in Azure Key Vault
- Exploring the non-nullable type warnings in C# 8

<https://www.tn-data.se>

6

Important Notes!

- **Interrupt me!**
- **Discuss!**
- **Ask questions!**

<https://www.tn-data.se>

7

Challenging the user

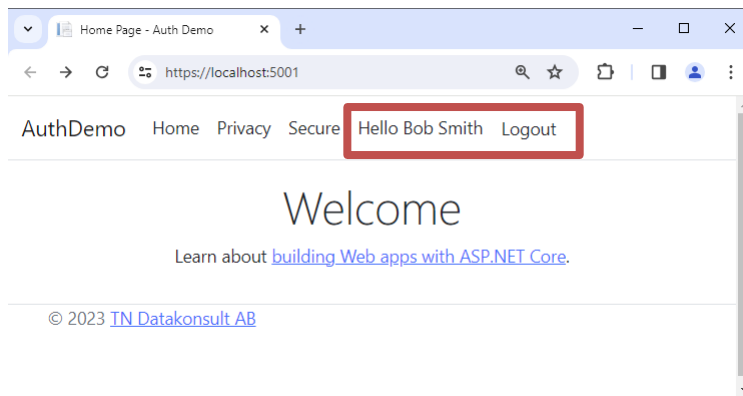
Module #1

© Tore Nestenius Datakonsult AB. All Rights Reserved.

<https://www.tn-data.se>

8

What do we want to achieve? ?

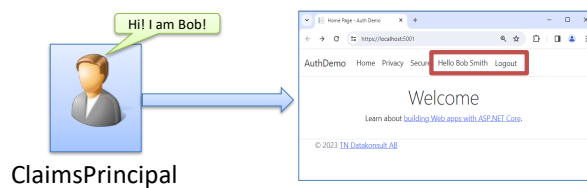


What does a page need to support this? ?

<https://www.tn-data.se>

9

We need a **ClaimsPrincipal** object!



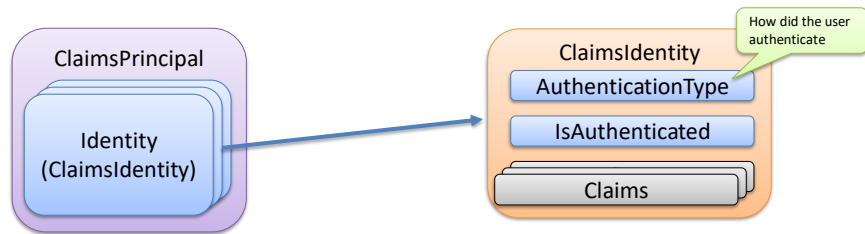
What is a **ClaimsPrincipal**? ?

<https://www.tn-data.se>

10

ClaimsPrincipal

The **ClaimsPrincipal** is the core **user object** in ASP.NET Core




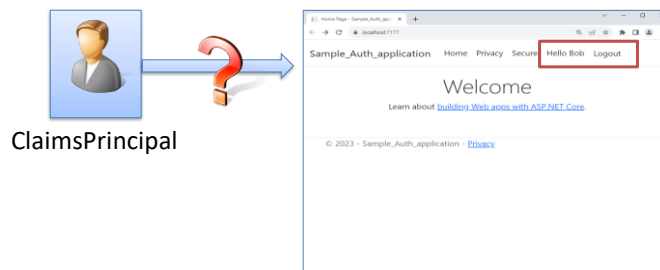
```
public class ClaimsPrincipal : IPrincipal
{
    //List of identities
    List<ClaimsIdentity> _identities = new();
    ...
}

public class ClaimsIdentity : IIdentity
{
    //List of claims
    List<Claim> _instanceClaims = new();
    ...
}
```

<https://www.tn-data.se>

11

How does the page get the **ClaimsPrincipal**? 



Let's look under the hood of ASP.NET Core

<https://www.tn-data.se>

12

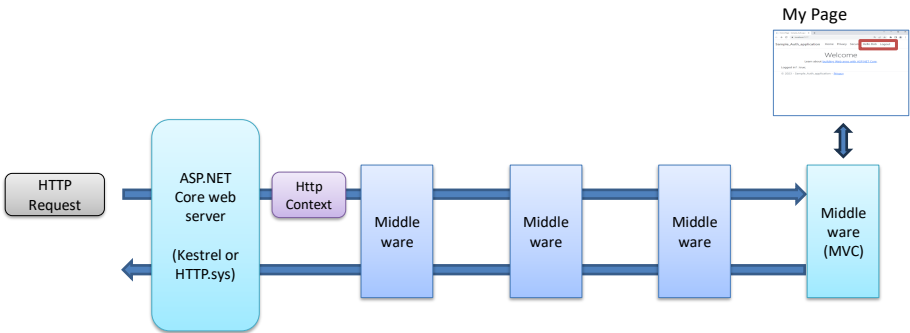
ASP.NET Core under the hood

<https://www.tn-data.se>

13

Authenticating the user

ASP.NET Core consists of a modular **request pipeline**

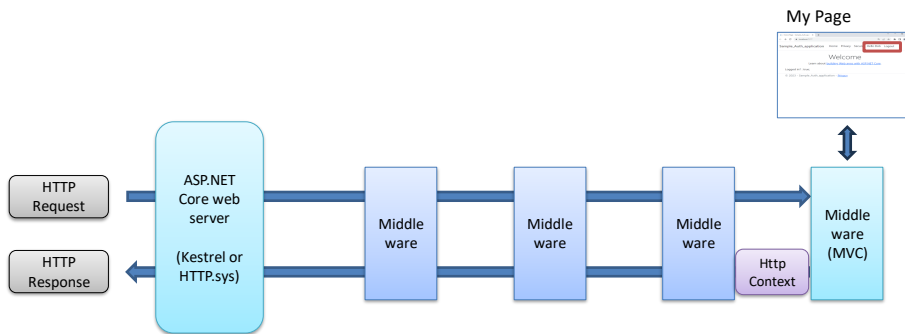


<https://www.tn-data.se>


14

Authenticating the user

ASP.NET Core consists of a modular **request pipeline**



The user is initially an **anonymous, unauthenticated** user

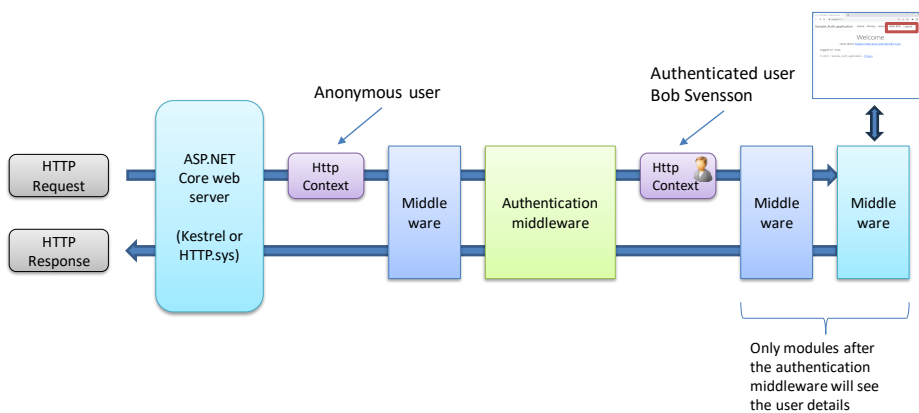
Who sets the user? 


<https://www.tn-data.se>

15

Setting the ClaimsPrincipal

The **Authentication middleware** sets the user



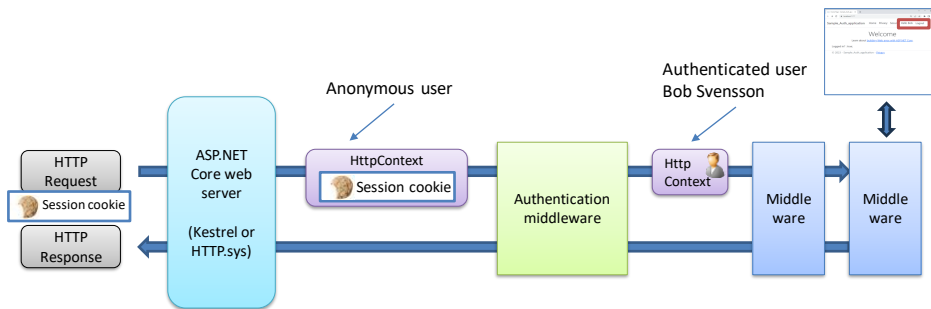
Where does the user data come from? 

<https://www.tn-data.se>

16

Setting the ClaimsPrincipal

By default, the data is taken from the **session cookie**

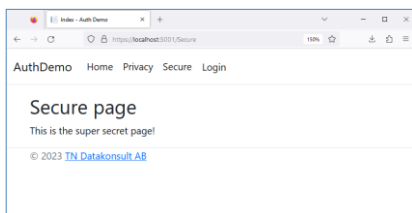


<https://www.tn-data.se>

17

DEMO TIME!

1 We secure this page



```
public async Task<IActionResult> Index()
{
    if (User.Identity.IsAuthenticated == false)
    {
        await HttpContext.ChallengeAsync();
    }
    return View();
}
```

2 We add the Authentication middleware

```
builder.Services.AddAuthentication();
```

```
...
```

```
app.UseAuthentication();
```

```
...
```

<https://www.tn-data.se>

18

Handlers

Module #2

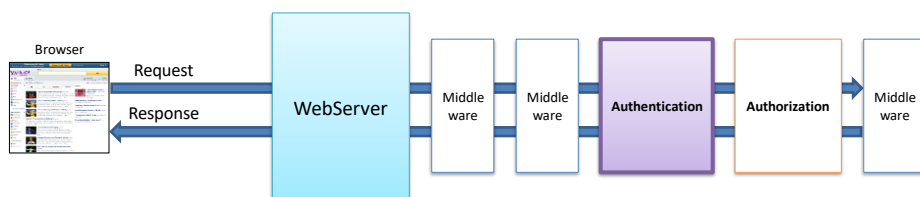
© Tore Nestenius Datakonsult AB. All Rights Reserved.

<https://www.tn-data.se>

19



What is inside the **Authentication sub-system**? 

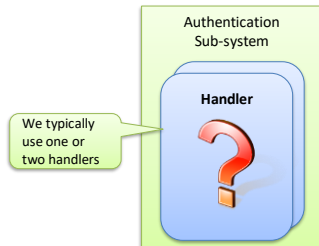


<https://www.tn-data.se>

20

The authentication sub-system

Inside, it contains a set of **authentication handlers**



```
//Add the Cookie and OpenID-Connect handler
builder.Services.AddAuthentication()
    .AddCookie()
    .AddOpenIdConnect();
```

```
// Add the JwtBearer handler (for APIs)
builder.Services.AddAuthentication()
    .AddJwtBearer();
```

They implement different **authentication functionalities**

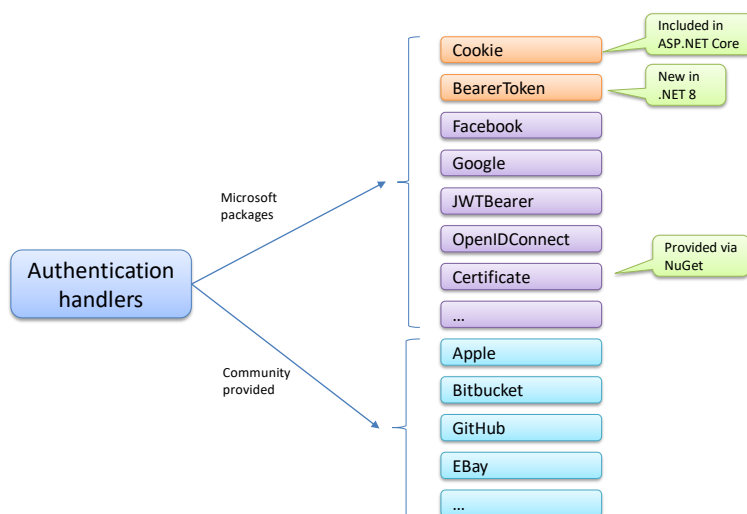
What handlers exist? 

<https://www.tn-data.se>

21

Authentication

Many **handlers** exists:



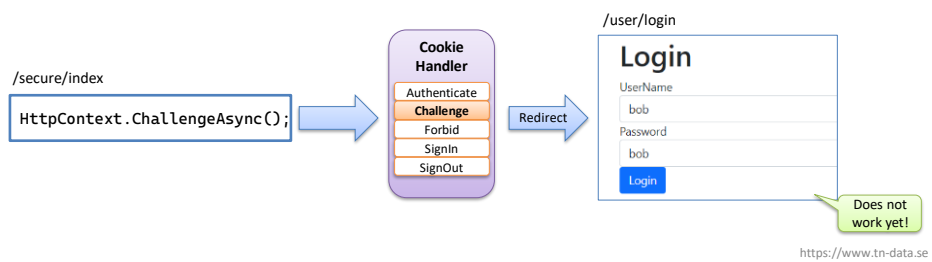
<https://www.tn-data.se>

22

DEMO TIME!

Adding the cookie handler

```
builder.Services.AddAuthentication()  
    .AddCookie("cookie", o =>  
    {  
        o.LoginPath = "/user/login";  
        o.LogoutPath = "/user/logout";  
        o.AccessDeniedPath = "/user/accessdenied";  
    });
```




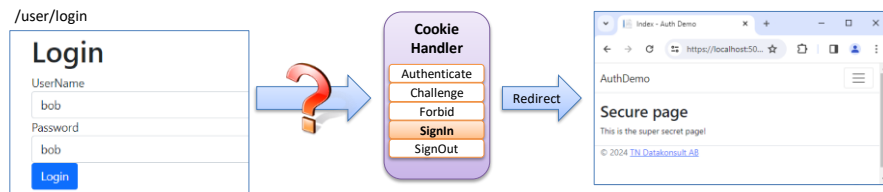
23

Sign in the user

24

Sign in the user

What happens when the user clicks the **Login** button? 



Let's explore this!

<https://www.tn-data.se>

25

Signing in a user

A typical login in page looks like this:

```
[HttpGet]
public IActionResult Login(string returnUrl)
{
    return View(new LoginModel() { ReturnUrl = returnUrl });
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task Login(LoginModel loginCredentials)
{
    //1. Validate username + password
    //...
}
```

```
public class LoginModel
{
    public string Username { get; set; };
    public string Password { get; set; };
    public string ReturnUrl { get; set; };
}
```

First, we validate the **credentials**

<https://www.tn-data.se>

26

Signing in a user

Then, we create a **list of claims** for this user:

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task Login(LoginModel loginCredentials)
{
    //1. Validate username + password

    //2. Load the claims for this user from the DB
    var claims = new List<Claim>()
    {
        new("sub", "1234"),
        new("name", "Bob"),
        new("email", "bob@tn-data.se"),
        new("role", "developer")
    };

    //...
}
```



The **sub** claim is the most important one

<https://www.tn-data.se>

27

Signing in a user

Next, we create an **Identity** based on these claims

```
//1. Validate username + password

//2. Load the claims for this user from the DB
var claims = new List<Claim>()
{
    new("sub", "1234"),
    new("name", "Bob"),
    new("email", "bob@tn-data.se"),
    new("role", "developer")
};

//3. Create an Identity for these claims
var identity = new ClaimsIdentity(claims: claims,
                                authenticationType: "pwd",
                                nameType: "name",
                                roleType: "role");

//...
```

AuthenticationType describes how the user authenticated

<https://www.tn-data.se>

28

Signing in a user

Next, we create a **ClaimsPrincipal** for the user

```
//1. Validate username + password

//2. Load the claims for this user from the DB
var claims = new List<Claim>()
{
    new("sub", "1234"),
    new("name", "Bob"),
    new("email", "bob@tn-data.se"),
    new("role", "developer")
};

//3. Create an Identity for these claims
var identity = new ClaimsIdentity(claims: claims,
    authenticationType: "pwd",
    nameType: "name",
    roleType: "role");

//4. Create the principal based on the users Identity
var principal = new ClaimsPrincipal(identity);

//...
```

It represents the current user and all its identities

<https://www.tn-data.se>

29

Signing in a user

Finally, we **sign in** the **ClaimsPrincipal** user

```
public async Task Login(LoginModel loginCredentials)
{
    //...

    var identity = new ClaimsIdentity(claims: claims,
        authenticationType: "pwd",
        nameType: "name",
        roleType: "role");

    var principal = new ClaimsPrincipal(identity);

    var prop = new AuthenticationProperties()
    {
        RedirectUri = loginCredentials.ReturnUrl,    //Where to redirect to after login
        Items =
        {
            { "IpAddress", "192.168.0.3" },
            { "ComputerName", "MyComputer" },
            { "ApiKey", "Summer2023" }
        }
    };


    await HttpContext.SignInAsync(scheme: "cookie", principal: principal, properties: prop);
}
```

Optional items

Who should do the sign-in

<https://www.tn-data.se>

30

What happens when we call SignInAsync? 

```
HttpContext.SignInAsync(scheme: "cookie", principal: principal, properties: prop);
```

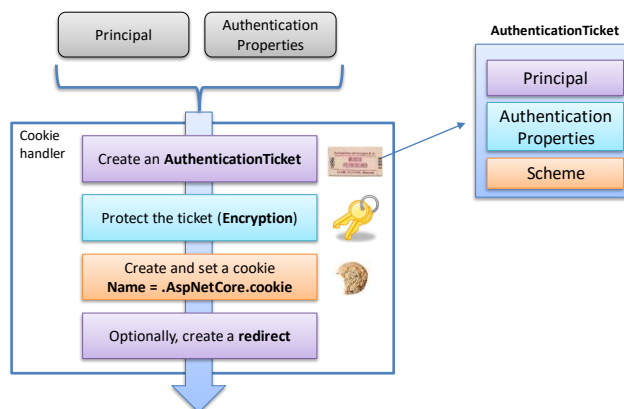
<https://www.tn-data.se>

31

What happens when we call SignIn?

Inside the **cookie handler**, these are the main steps:

```
HttpContext.SignInAsync(scheme: "cookie", principal: principal, properties: prop);
```



<https://www.tn-data.se>

32

DEMO TIME!

Let's implement the Login logic

```
//1. Validate username + password

//2. Load the claims for this user from the DB
var claims = new List<Claim>()
{
    new("sub", "1234"),
    new("name", "Bob"),
    new("email", "bob@tn-data.se"),
    new("role", "developer")
};

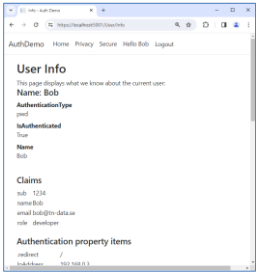
var identity = new ClaimsIdentity(claims,
                                authenticationType: "pwd",
                                nameType: "name",
                                roleType: "role");

var principal = new ClaimsPrincipal(identity);

var prop = new AuthenticationProperties()
{
    RedirectUri = "/",
    Items =
    {
        { "IpAddress", "192.168.0.3" },
        { "ComputerName", "MyComputer" },
        { "ApiKey", "Summer2023" }
    }
};

await HttpContext.SignInAsync(scheme: "cookie",
                              principal: principal,
                              properties: prop);

return LocalRedirect("/");
```



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
.AspNetCore.cookie	CFDj8D...	localhost	/	Session	514	✓	✓	Lax

https://www.tn-data.se

33

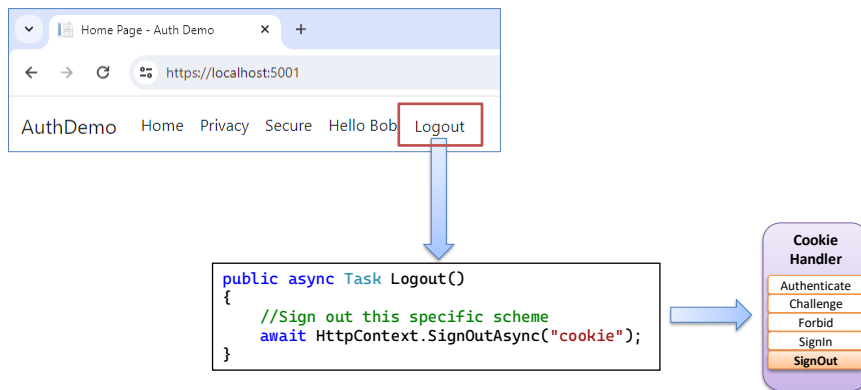
Sign out

Module #4

34

Sign out

Next, we want to be able to sign out users



What happens when we call this method?

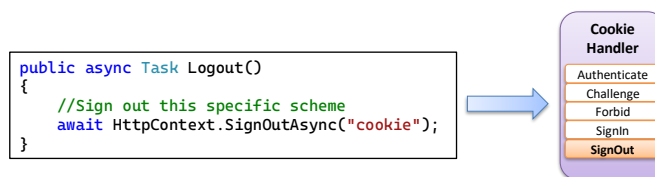


<https://www.tn-data.se>

35

Sign out

What happens when we call this method?



It deletes the authentication cookie

That's all!

```
Set-Cookie: .AspNetCore.cookie=; expires=Thu, 01 Jan 1970 00:00:00 GMT; path=/; secure; samesite=lax; httponly
```

<https://www.tn-data.se>


36

DEMO TIME!

Let's implement Sign out

```
var prop = new AuthenticationProperties()
{
    RedirectUri = "/"
};

//Sign out from the specific scheme
await HttpContext.SignOutAsync("cookie", prop);
```

But are we really signed out? 



Session cookie

<https://www.tn-data.se>

37

Large cookies




© Tore Nestenius

<https://www.tn-data.se>

38

DEMO TIME!

What happens when the cookie grows? 

```
//2. Load the claims for this user from the DB
var claims = new List<Claim>()
{
    new("sub", "1234"),
    new("name", "Bob"),
    new("email", "bob@tn-data.se"),
    new("role", "developer"),
    new("accesstoken", new string('x', 4000)),
    new("idtoken", new string('y', 4000)),
    new("refresh token", new string('z', 1000))
};
```

Let's find out!

<https://www.tn-data.se>

39

Improving Security By Putting Your Cookies On A Diet



© Tore Nestenius Datakonsult AB. All Rights Reserved.

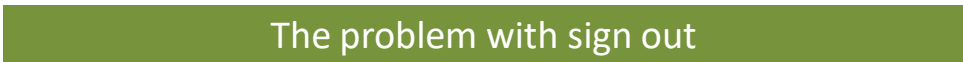
40



The problem with sign out

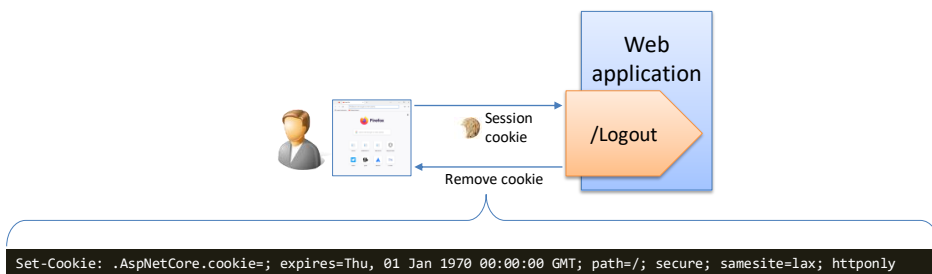
<https://www.tn-data.se>


41



The problem with sign out

The cookie is still valid after signout!



How can we fix this? 

<https://www.tn-data.se>

42

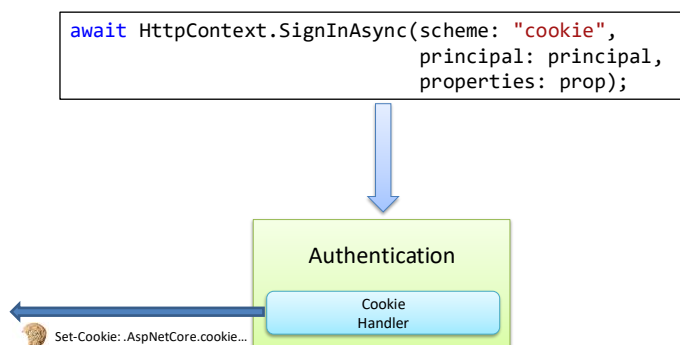
Large cookies


43

<https://www.tn-data.se>

Large cookies

We saw that the cookie can grow in size



How can we reduce the size of the cookie? 

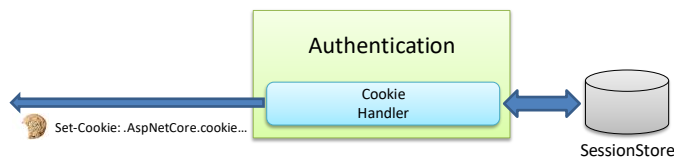
44

<https://www.tn-data.se>

Reducing the cookie size

We can add a **SessionStore** to the cookie handler

```
}).AddCookie(options =>
{
    ...
    options.SessionStore = new MySessionStore();
})
```



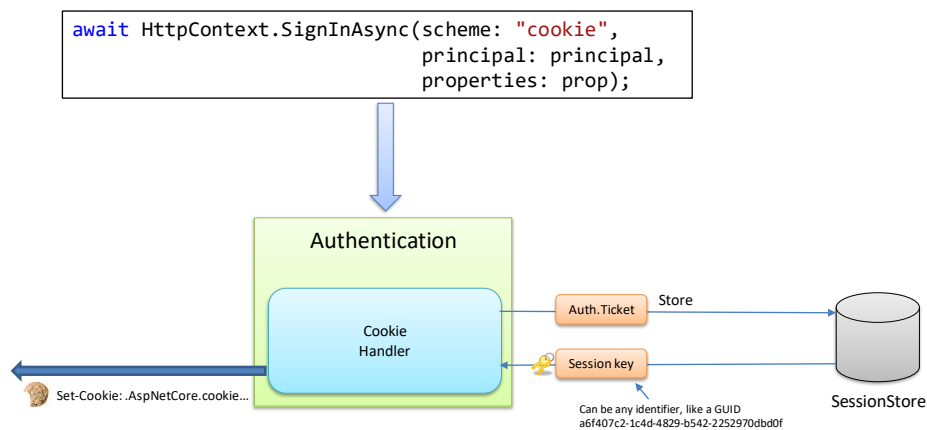
What does the **SessionStore** do? 


<https://www.tn-data.se>

45

Reducing the cookie size

It acts like a **cache** for the tokens



How can we implement a session store? 

<https://www.tn-data.se>

46

Reducing the cookie size

All we need is a class that implements **ITicketStore**



<https://www.tn-data.se>

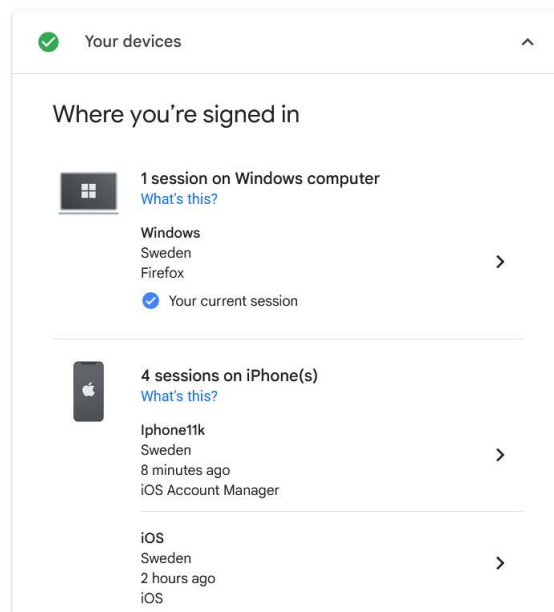
47

Possibilities when using a session store

<https://www.tn-data.se>

48

Possibilities when using a session store



<https://www.tn-data.se>

49

Possibilities when using a session store

You can sign out all or specific users!



<https://www.tn-data.se>

50

Possibilities when using a session store

Metrics!



<https://www.tn-data.se>

51

DEMO TIME!

Let's add a session store

```
.AddCookie("cookie", o =>
{
    ...
    o.SessionStore = new MySessionStore();
});
```

<https://www.tn-data.se>

52

QUESTIONS?



<https://www.tn-data.se>