



Tech Ed

23. - 25. 5. 2023 | PRAHA / ONLINE

Největší československá
konference o IT

GOLD PARTNER

<epam>



Implementing security interfaces in ASP.NET Core Identity

Ákos Nagy | Teched 2023

Tech·Ed

GOPAS

GOLD PARTNER

<epam>



Agenda

- ASP.NET Core Identity – overview
- Implementing defined interfaces to support security-related tasks
- Questions

ASP.NET Core Identity overview

- An API that supports UI login functionality
- Manage users, passwords, profile data, roles, claims token
- Support external login providers
- Support authorization functions

Special requirements related to security

- Extend the built-in user schema with extra information
- Extend information stored in the authentication token/cookie
- Implementing a custom user/scoped session
- Extend the cookie/token validation process
- Extend anti-forgery token/cookie handling
- Add protection to Personally Identifiable Information
- Add custom authentication ticket handling

Why should I care?

- **What is a framework?**

- By definition, a framework provides interfaces that the programmer should implement, then instantiate and pass it back to the framework...
- ... creating custom interfaces not only makes using the framework completely useless
- ... but also starts a chain reaction (no part of the framework know about custom interfaces, so every part of the frameworks must be gradually replaced)

- **How is security “different”?**

- Don't create your own security solutions (unless you're an expert in security solutions)
- Broken authentication workflows
- Information disclosure
- Math ☹️

Extending user information

- Create custom User class
- Add extra properties about the user
- Create custom IdentityDbContext class
- Handles the custom user entites

Using the extended user information for authorization

Using the extended user information for authorization

- **What?**

- A collection of claims
- A claim is a piece of information that the server adds to the token about the user...
- ... so the user can later “claim” something about themselves

- **Why?**

- This way sensitive information that can be used to identify the user don't need to be added to every request
- The server doesn't need to query the information every time from the database
- **The server can use this for authorization**

- **How?**

- The token is digitally signed, so the server can trust the resubmitted token
- In case of an MVC application, the token is sent in a cookie that is added to every subsequent request by the browser

Using the extended user information for authorization

- Create a custom `UserClaimsPrincipalFactory`
 - add any custom claims to the token
- Create a custom `IAuthorizationRequirement`
 - a simple POCO that describes the requirement)
- Create a custom `AuthorizationHandler`
 - receives an instance of the requirement and the http context, so it can validate the claims from the context against the requirement
- Create a custom `AuthorizationAttribute`
 - can be applied to actions/controllers and can relate information to the requirement
- Create a custom `AuthorizationPolicyProvider`
 - processes the attribute, parse the data from the attribute and create the custom requirement

Protecting data

- Implement a custom `ILookupProtector`
 - Protects `UserName`, `NormalizedUserName`, `Email`, `NormalizedEmail`, `PhoneNumber`
 - Also protects any properties marked with `[PersonalData, ProtectedPersonalData]`
 - Supports key rotation
 - Gets the current key id from the keyring, encrypts the data and stores the keyed
 - When decrypting, queries the keyring for the key
- Implement a custom `ILookupProtectorKeyRing`
 - Manages the keys, rotates them using a given algorithm
- Implement `IPersonalDataProtector` (optional)
 - The default implementation uses the previous two to protect the data in the format `keyid:encrypteddata`

Remote sign-in management

- Implement ITicketStore
 - ITicketStore manages the users logins
 - Implementing a custom version helps us to store the logins and later list or delete them
 - AuthenticationTickets store claims, authentication methods and other information
 - Can be serialized and stored in a database

Extending anti-forgery protection

- Implement `IAntiForgeryAdditionalDataProvider`
 - Antiforgery tokens prevent CSRF attacks
 - Antiforgery tokens are scoped to the site and to the user (if there's authentication), but can be reused without time limitation
 - Using this interface, extra information can be added to the token
 - Later this information can be verified

Hooking up callbacks to token validation

- Implement `CookieAuthenticationEvents`
 - Has methods that can be overridden
 - Before and after sign in
 - Before sign out
 - Validating the claims in the token
 - Redirecting when authentication/authorization fails

Thank you for your attention!

- Source code available at: <https://github.com/conwid/AspNetSecurity>
 - Each complete demo is a commit
- Read more from me at: <https://dotnetfalcon.com>
- Available for hire (consultation, training): <https://hireme.dotnetfalcon.com>
- Keep in touch: <https://linkedin.com/in/azakosnagy>



TechEd

23. - 25. 5. 2023 | PRAHA / ONLINE

