

Øvelse 34 – Håndtering af exceptions i ASP.NET Core Applikationer

Information

Formålet med denne øvelse er at vise den grundlæggende tilgang til overordnet håndtering af exceptions i applikationer.

Øvelsen viser blot den overordnede og generiske tilgang til at sikre, at applikationen ikke u hensigtsmæssigt eksponerer en stack trace. Den beskriver dog ikke, hvordan man gennemgående skal håndtere exceptions, da dette emne vil være for omfattende i forhold til emnets størrelse i faget. Såfremt man ønsker at lære mere om håndtering af uventede fejl (exceptions), kan man referere til programmeringssproget og frameworkets dokumentation, f.eks. til MS docs ASP.NET Core [Handling exceptions in ASP.NET Core](#).

Tilgangen, der skal arbejdes med, refereres ofte til som *Global exception handling*. Dette betyder, at alle exceptions, som ikke bliver grebet (f.eks. med en try/catch statement), altid vil blive håndteret af en generisk exception handler, der typisk returnerer en generisk fejl tilbage til slutbrugeren (f.eks. HTTP statuskode 500 i webapplikationer).

I [WebGoat](#) applikationen er der implementeret en global håndtering af exceptions. Dette kan ses i metoden *Configure* i *Startup*-klassen. I nedenstående udsnit er dette vist.

```
if (env.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}
else
{
    app.UseExceptionHandler($"{StatusCodeController.NAME}?code=500");
}
```

I forrige var det muligt at fremprovokere en stack trace, fordi applikationen eksekverer i udviklingsmiljø. Dette er belejligt, når udvikleren skal ændre på softwaren, for at kunne se de konkrete fejlmeddelelser. Men i et produktionsmiljø er dette ikke hensigtsmæssigt, da det, som vist i forrige opgave, videregiver information til potentielle fjendtlige aktører.

Det er vigtigt at understrege, at global fejl-håndtering er en sidste "nød-bremse" i tilfælde af uforudsete fejl, der ikke bliver håndteret af applikationen. Kan en angriber fremprovokere en status 500-meddelelse, vil dette stadig indikere overfor angriberen, at der er opstået en fejl. Angriberen kan dog ikke se, hvilken fejl der er tale om, eller udlæse informationer fra stack trace.

For yderligere informationer om fejlhåndtering kan du læse MS DOCS [Handling exceptions in ASP.NET Core](#), eller (gen)læse kapitel 9 i bogen *Secure by Design*.

▮ Kobling til sikkerhedsprincipper: *Fail Secure* og *Defense in Depth*

Global exception handling understøtter princippet om **fail secure**: Når noget går galt, skal systemet stadig reagere sikkert. I stedet for at vise intern information (f.eks. stack trace), returneres en generisk fejlmeddelelse til brugeren – og systemets integritet bevares.

Det er også et eksempel på **defense in depth**: Selvom enkelte metoder eller komponenter ikke håndterer fejl korrekt, sikrer den globale fejlhåndtering, at applikationen stadig afslutter sikkert og kontrolleret.

▮ Global exception handling skal dog aldrig stå alene. Den er en *sidste sikkerhedsmekanisme* – ikke en erstatning for lokal, domænespecifik undtagelseshåndtering.

▮ **Fail Secure** En applikation fejler sikkert (fail secure), når den – selv i fejlsituationer – beskytter konfidensielle oplysninger og undgår utilsigtet adfærd.

▮ **Global exception handling** hjælper med at opretholde dette princip, især i webapplikationer, hvor interne fejl ellers kan føre til informationslæk.

▮ Instruktioner

Obs, hvergang du laver ændringer, skal du gen bygge docker filen inden du eksekverer den igen. Kommandoerne er de samme som i forrige øvelse

1. ▮ I klassen *Startup* skal du udkommentere alt koden for global håndtering af exceptions (if/ else der tidligere er vist i afsnittet *Information*) og i stedet indsætte

```
app.UseExceptionHandler($"{StatusCodeController.NAME}?code=500");
```

2. ▮ Eksekver applikationen, og fremprovoker en fejl med ' ' (samme fremgangsmåde som i forrige øvelse).
3. ▮ Notér dig, hvilken fejl der opstår. Giver den fejl mening? Og er den fejl valgt fremfor f.eks. statuskode 500?

1. ▮ I metoden *Configure* i klassen *Startup* skal du udkommentere

```
app.UseExceptionHandler($"{StatusCodeController.NAME}?code=500");
```

2. ▮ Eksekver applikationen, og fremprovoker en fejl med ' '

3. ☐ Notér dig, hvilken fejl der opstår. Hvilken HTTP statuskode bliver der returneret?

1. ☐ I metoden `Configure` i klassen `Startup` skal du indsætte koden

```
app.UseDeveloperExceptionPage();
```

2. ☐ Udkommenter `app.UseExceptionHandler($"{StatusCodeController.NAME}?code=500");` som du tidligere indsat.

3. ☐ Eksekver applikationen, og fremprovoker en fejl med `'`

4. ☐ Notér dig, hvilken fejl der opstår.

☐ Refleksionsspørgsmål

1. ☐ Hvorfor er det vigtigt at skjule detaljerede fejlmeddelelser som stack traces i produktionsmiljøer?
2. ☐ Hvilke risici kan opstå, hvis stack traces bliver eksponeret for slutbrugere eller angribere?
3. ☐ Hvordan kan global exception handling bidrage til en mere sikker applikation?
4. ☐ Hvorfor er global exception handling ikke en erstatning for korrekt exception håndtering i koden?
5. ☐ Hvordan kan du sikre, at fejlmeddelelser er brugervenlige, men samtidig ikke afslører følsomme oplysninger?

Element i øvelsen	Relateret standard / princip	Kommentar
Håndtering af exceptions	ISO/IEC 27002:2022, afsnit 8.28 (Secure coding)	Anbefaler anvendelse af sikre kodningsprincipper, herunder korrekt håndtering af input, output og potentielle fejl, som en del af en sikker udviklingspraksis.
	OWASP ASVS 5.0.0, kontrol 16.5 (Error Handling)	Krav om korrekt håndtering af undtagelser for at undgå informationslækager og uventet adfærd. Fail secure nævnes direkte her.

Element i øvelsen	Relateret standard / princip	Kommentar
	Secure by Design (Johnsson et al.)	Understreger vigtigheden af at undgå at exception flows kompromitterer systemets integritet.

□ *Global exception handling er et vigtigt sidste forsvarslag, men skal suppleres med grundig exception håndtering og sikker design.*

□ *Global exception handling er med til at sikre, at applikationen opretholder princippet om fail secure, og fungerer som et element i en bredere defense in depth strategi.*

Last update: 2025-10-30 12:07:19