# Module 05:

# "Error Handling"

# Agenda

- Introduction
- **RFC 7807 and Problem Details**
- Result Pattern
- Summary

# RFC 7807: Problem Details for HTTP APIs

- Internet standard for errors
  - Described in https://www.rfc-editor.org/rfc/rfc7807.html
  - Blue = standard, Green = custom

- MIME type: **application/problem+json**

```
{ "type": "https://example.com/probs/out-of-credit",
  "title": "You do not have enough credit.",
  "detail": "Your current balance is 30, but that costs 50.",
  "instance": "/account/12345/msgs/abc",
  "balance": 30,
  "accounts": ["/account/12345",
               "/account/67890"]
}
```

# ASP.NET Core Problem Details

- .NET has built-in support for RFC 7807

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddProblemDetails();
...
var app = builder.Build();
app.UseExceptionHandler();
```

- Facilitated through
  - `ProblemDetails` type
  - **`Problem()`** method in `ControllerBase`

# Agenda

- Introduction
- RFC 7807 and Problem Details
- **Result Object Pattern**
- Summary

# Result Object Pattern

- Design approach that encapsulates the outcome of an operation in a way that distinctly separates success from failure.
- Traditionally, methods return null or throw exceptions to signify failure
- Result Object Pattern wraps the outcome in a "Result" object either representing
  - a successful outcome containing the expected data, or
  - an unsuccessful outcome with error information.

# Exceptions vs. Result Object Pattern

- Exceptions
  - Can be hard to analyze and tracks exceptions
  - We are throwing exceptions for ”expected” situations, e.g. NotFound
- Result Object Pattern
  - Errors are explicit, i.e. part of the signature of expected errors
  - Any exceptions are then truly ”exceptional” and unexpected
- Consequently; Result Object Pattern is often preferred in Clean Architecture (but not always!)

# Handling Errors

- Domain Errors
  - Define custom errors
- Application Errors
  - Built-in errors usually suffice

- Both converted to Presentation errors

# ErrorOr

- Many frameworks – a prominent one is the ErrorOr nuget package
  - https://www.nuget.org/packages/ErrorOr

- Defines built-in **`Error`** type and matching functions

```
ErrorOr<CommandResult> result = ...;

return result.Match(
    onValue: commandResult => OK(...),
    onError: Problem
);
```

- Can define further reusable helpers for converting
  - **`ErrorOr.Error`** instances -> **`ProblemDetails`** instances

# Summary

- Introduction
- RFC 7807 and Problem Details
- Result Object Pattern


WINCUBATE
Jesper Gulmann Henriksen
PhD, MCT, MCSD, MCPD
Phone : +45 22 12 36 31
Email : jgh@wincubate.net
WWW : http://www.wincubate.net
Ringgårdsvej 4A
8270 Højbjerg
Denmark