

ASP.NET WEBAPI INTRO



AARHUS
UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING

ST3ITS3
26 OCTOBER 2023

HENRIK BITSCH KIRK
ASSOCIATE PROFESSOR



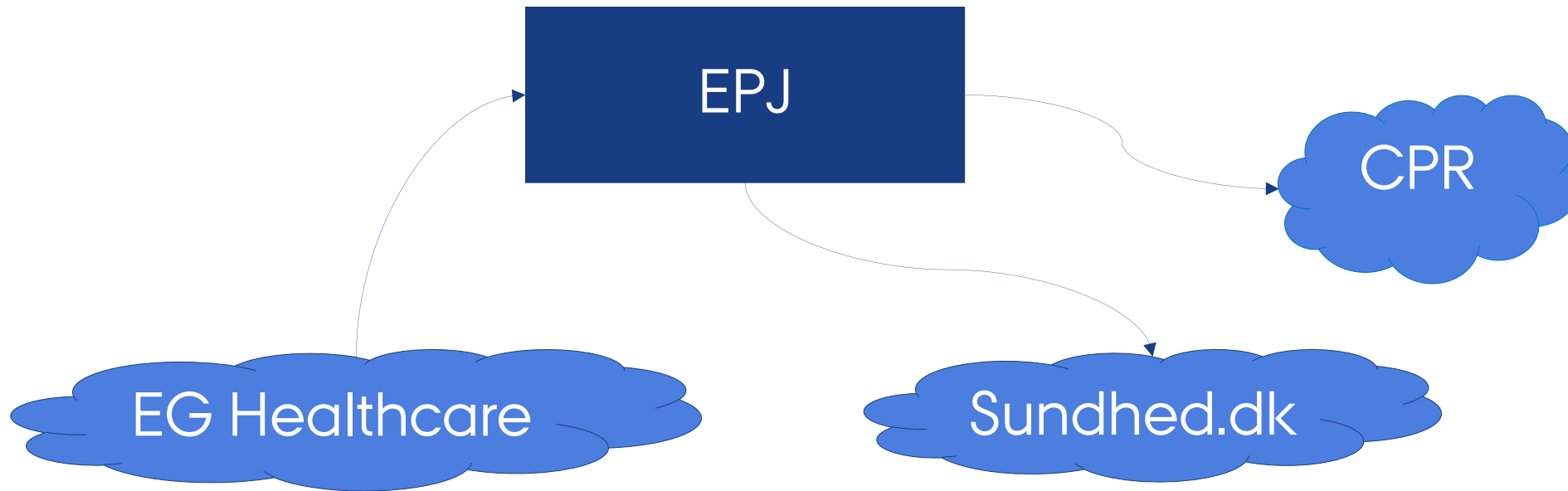
AGENDA

- Problem
- XML / JSON
- API
- ASP.NET

WHERE DO DATA COME FROM

Any one system can't contain all the needed data

So how does an application get data from *another* application?



WEBPAGES AND HTML

Søg i

nov

Vi fandt 631 resultater

Vis alle

Virksomheder (182)

P-enheder (434)

Personer (15)

Novo Holdings A/S ⓘ Tuborg Havnevej 19 2900 Hellerup	CVR-nummer: 24257630		
NOVO INVEST ApS ⓘ Gammel Strandvej 402 3060 Espergærde	CVR-nummer: 35679146		
Novo Revision ApS ⓘ Blomsterhaven 46 4300 Holbæk	CVR-nummer: 40100377	Status: Normal	Virksomhedsform: Anpartsselskab
NOVO LECO ApS C/O Mads Leth Christiansen Søbakken 11D 2920 Charlottenlund	CVR-nummer: 34691460	Status: Normal	Virksomhedsform: Anpartsselskab

```
<html lang="da">
  <head> ... </head>
  <body> flex
    <noscript> ... </noscript>
    <div id="app" data-v-app>
      <div id="cvr-paa-virk">
        <div class="hide-base-svg"> ... </div>
        <div class="layout">
          <header class="header"> ... </header>
          <div class="stage">
            <div class="sprogvaelger" role="complementary" aria-label="Sprogvaelger" data-v-54211c0a> ... </div>
            <div class="login-dropdown" role="complementary"> ... </div>
            <div class="container">
              <div class="broedkrumme" role="navigation" aria-label="Brødkrumme navigation" data-v-959af90c> ... </div>
              <main id="main-content" class>
                <div> ... </div>
                <!-->
                <div class="soegeresultater">
                  <div class="soegefelt" data-v-2b88c0cb> ... </div>
                  <h2>Vi fandt 631 resultater</h2>
                  <div class="row mb-4"> ... </div> flex
                  <div class="soegeresultaterTabel" data-cy="soegeresultater-tabel" data-v-295c8b5e>
                    <div class="row" data-v-295c8b5e> flex
                      <div class="col-12 col-lg-4" data-v-295c8b5e>
                        <span class="bold value" data-v-295c8b5e> == $0
                          "Novo Holdings A/S "
                        <!-->
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </body>
</html>
```



JSON OR XML

Data format better suited for applications

JSON

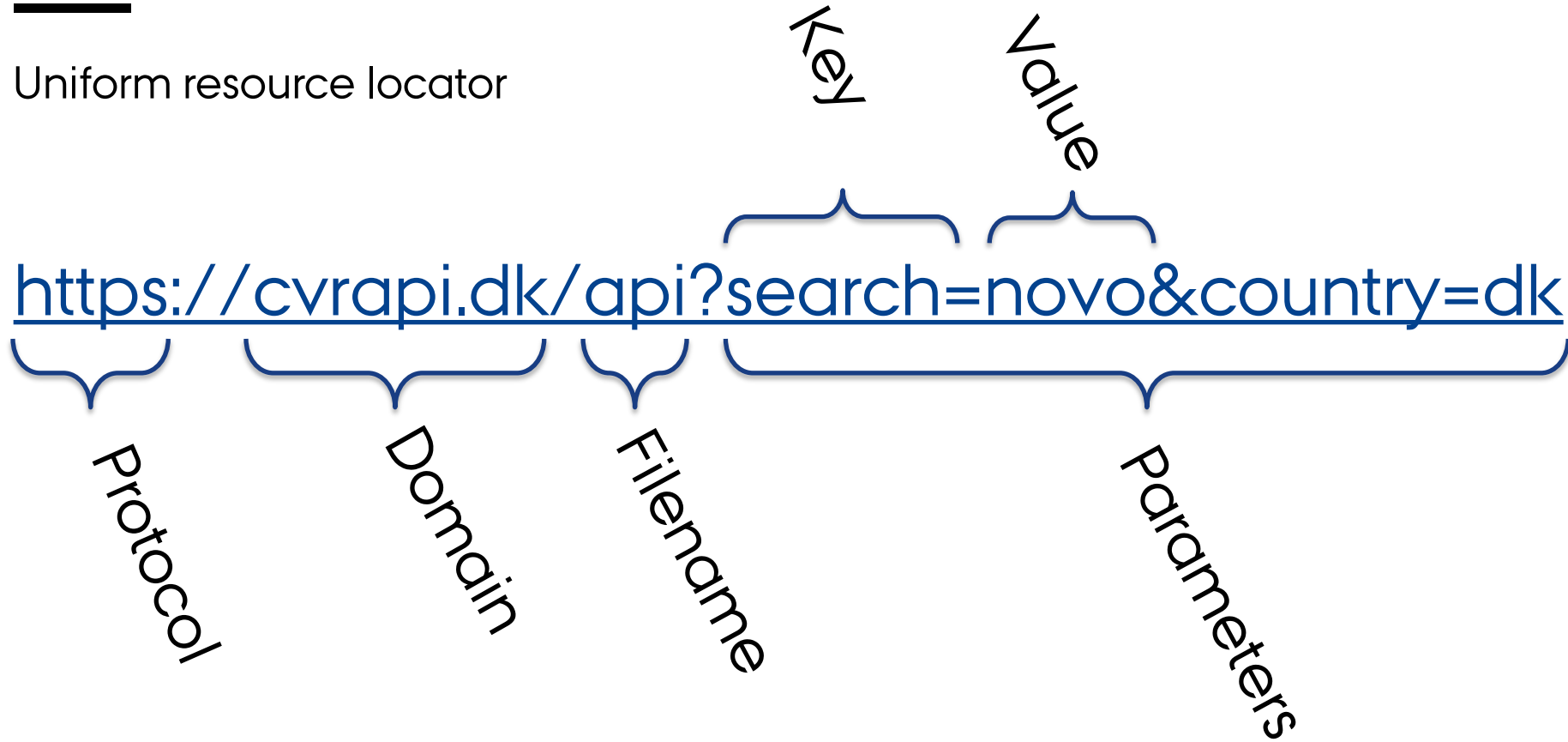
- E.g.
<https://cvrapi.dk/api?search=novo&country=dk>
- JavaScript Object Notation
- Lightweight
- Simple to parse with
<https://www.newtonsoft.com/json>

XML

- E.g.
<https://cvrapi.dk/api?search=novo&country=dk&format=xml>
- Extensible Markup language
- Verbose
- Parse with <https://learn.microsoft.com/en-us/dotnet/api/system.xml.xmldocument?view=net-6.0>

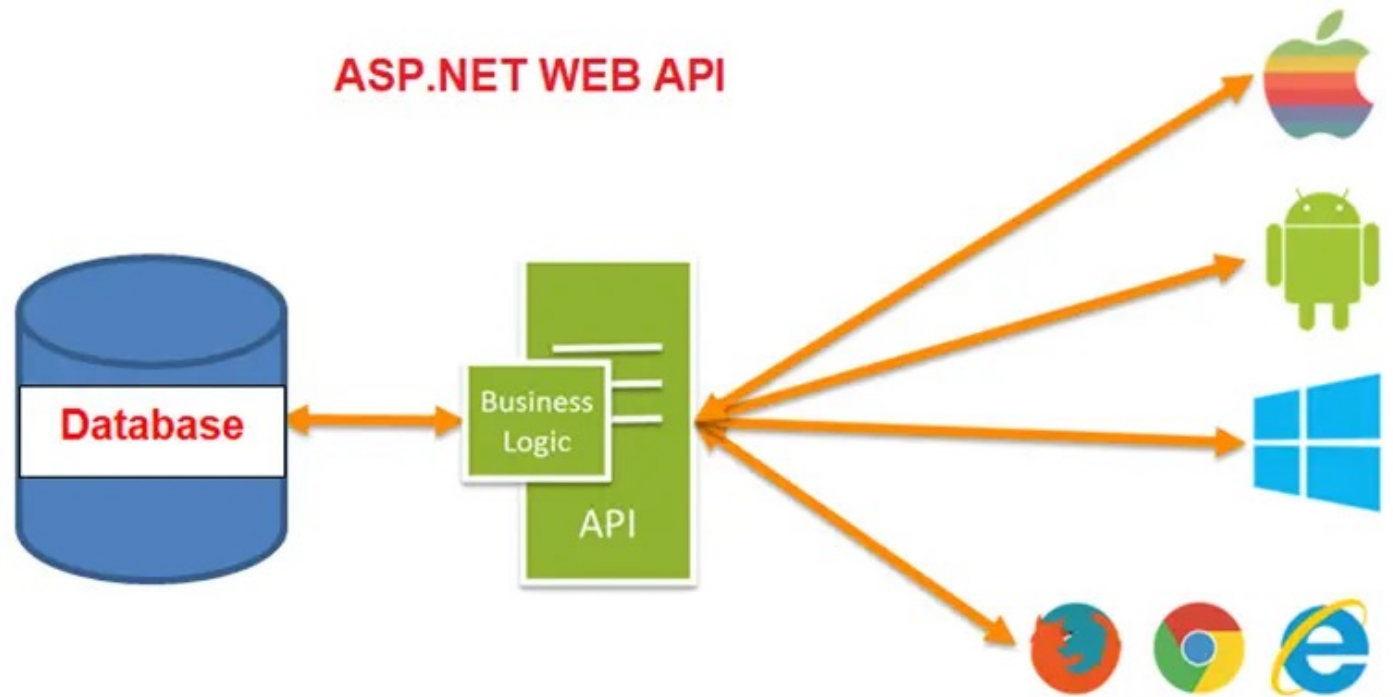
URL

Uniform resource locator



ASP.NET WEB API

- This is Microsoft's recommended way of creating an API
 - Application Programming Interface

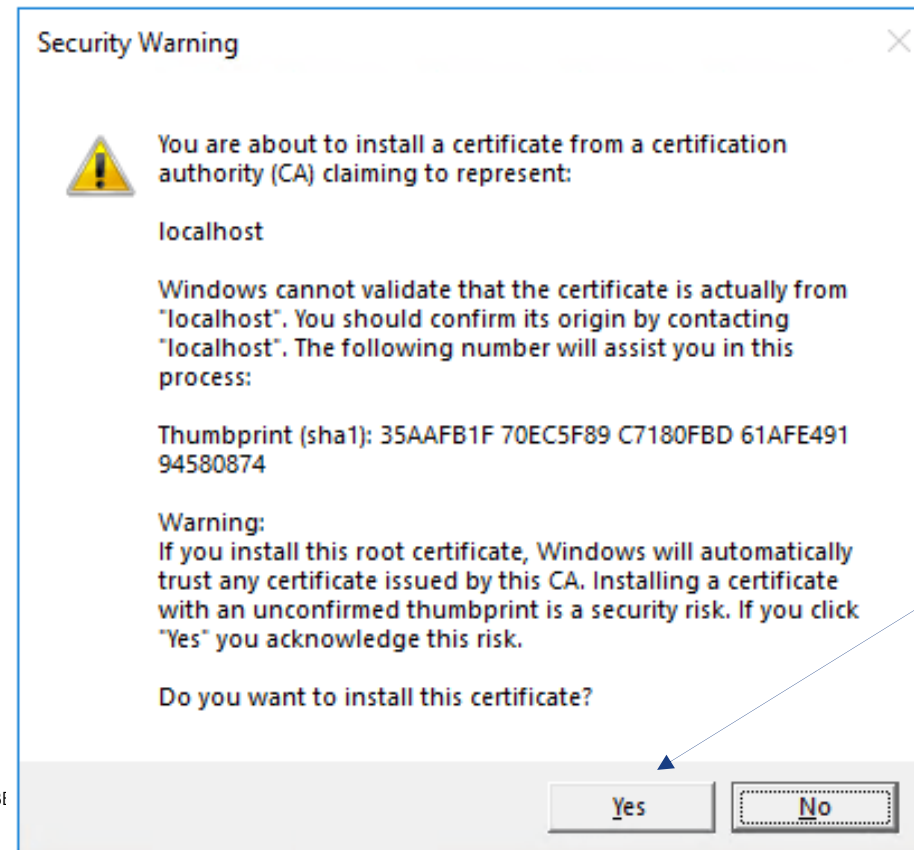
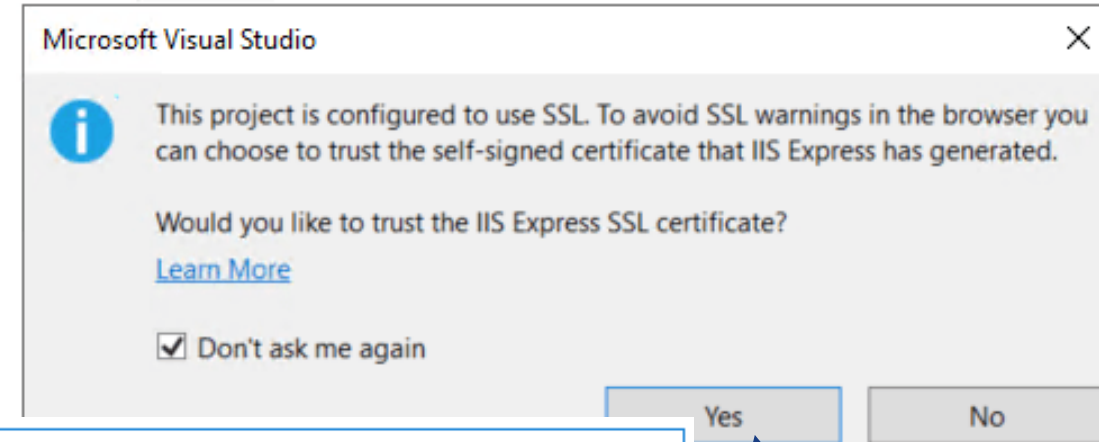


<https://dotnettutorials.net>

CREATING A PROJECT

In Visual Studio

1. In the *File* menu choose *New->Project*
2. Search for Web API and select **'ASP.NET Core Web API'**
3. In the Additional information menu
 1. Make sure you use .NET 6.0 or higher
 2. Make sure you *Use Controllers* are selected
4. Run the project.



GENERATED CODE

WeatherForecast.cs

Controllers/

WeatherForecastController.cs

Program.cs



GENERATED CODE - MODEL

WeatherForecast.cs

Controllers/

WeatherForecastController.cs

Program.cs

```
public class WeatherForecast
{
    public DateOnly Date { get; set; }
    public int TemperatureC { get; set; }
    public int TemperatureF => 32 +
(int)(TemperatureC / 0.5556);
    public string? Summary { get; set; }
}
```



GENERATED CODE

WeatherForecast.cs

Controllers/

WeatherForecastController.cs

Program.cs

```
[ApiController]
```

```
[Route("[controller]")]
```

```
public class WeatherForecastController :
```

```
ControllerBase {
```

```
[HttpGet(Name = "GetWeatherForecast")]
```

```
public IEnumerable<WeatherForecast> Get() {
```

```
return Enumerable.Range(1, 5).Select(index => new
```

```
WeatherForecast{
```

```
    Date = DateOnly.FromDateTime(
```

```
        DateTime.Now.AddDays(index)),
```

```
    TemperatureC = Random.Shared.Next(-20, 55),
```

```
    Summary = Summaries[
```

```
        Random.Shared.Next(Summaries.Length)]
```

```
    })
```

```
.ToArray();
```

```
}
```

```
}
```



GENERATED CODE

WeatherForecast.cs

Controllers/

WeatherForecastController.cs

Program.cs

```
var builder = WebApplication.CreateBuilder(args);  
// Add services to the container.  
builder.Services.AddControllers();  
builder.Services.AddEndpointsApiExplorer();  
builder.Services.AddSwaggerGen();  
var app = builder.Build();  
// Configure the HTTP request pipeline.  
if (app.Environment.IsDevelopment()) {  
    app.UseSwagger();  
    app.UseSwaggerUI();  
}  
app.UseHttpsRedirection();  
app.UseAuthorization();  
app.MapControllers();  
app.Run();
```



EXERCISES

Solve 1 and 2

Jump to Bonus when done



REQUESTS

- HTTP Verbs
- **GET**
- POST
- PUT
- PATCH
- DELETE

```
[HttpGet("{id}")]  
public async Task<ActionResult<TodoItem>> GetTodoItem(long  
id) {  
    // GET todoItem  
    // TodoItem todoItem = ...  
    if (todoItem == null) {  
        return NotFound();  
    }  
    return todoItem;  
}
```

Used to return resources from a web server API (one or many)

TASK + ASYNC/AWAIT

You will see this in much greater details in SW4SWD

- Task is a lightweight thread.
 - “All” methods that contains ‘async’ in name returns a task.
- Task<T> can contain data.
- `await` unwraps (among other things) inner data
 - `string stringData = await httpClient.GetStringAsync(URL);`
- If you uses await in a method body, the body needs to be decorated with `async` and return a Task

REQUESTS

- HTTP Verbs
- GET
- **POST**
- PUT
- PATCH
- DELETE

[HttpPost]

```
public async Task<ActionResult<TodoItem>>
```

```
PostTodoItem(TodoItem todoItem)
```

```
{
```

```
    return Created("todoitem", todoItem);
```

```
    //return CreatedAtAction(nameof(GetTodoItem), new { id =  
todoItem.Id }, todoItem);
```

```
}
```

Used to create new resources on the web server

REQUESTS

- HTTP Verbs
- GET
- POST
- **PUT**
- PATCH
- DELETE

```
[HttpPut("{id}")]  
public async Task<IActionResult> PutTodoItem(long id,  
TodoItem todoItem)  
{  
    if (id != todoItem.Id) {  
        return BadRequest();  
    }  
    return NoContent();  
}
```

Used to update existing resources on the server

REQUESTS

- HTTP Verbs
- GET
- POST
- PUT
- PATCH
- **DELETE**

```
[HttpDelete("{id}")]  
public async Task<IActionResult> DeleteTodoItem(long id)  
{  
    var todoItem = null; // Find resources somewhere  
    if (todoItem == null) {  
        return NotFound();  
    }  
    return NoContent();  
}
```

Used to delete resources from the

REQUESTS

- HTTP Verbs
- GET
- POST
- PUT
- **PATCH**
- DELETE

Used to update resources from partial documents

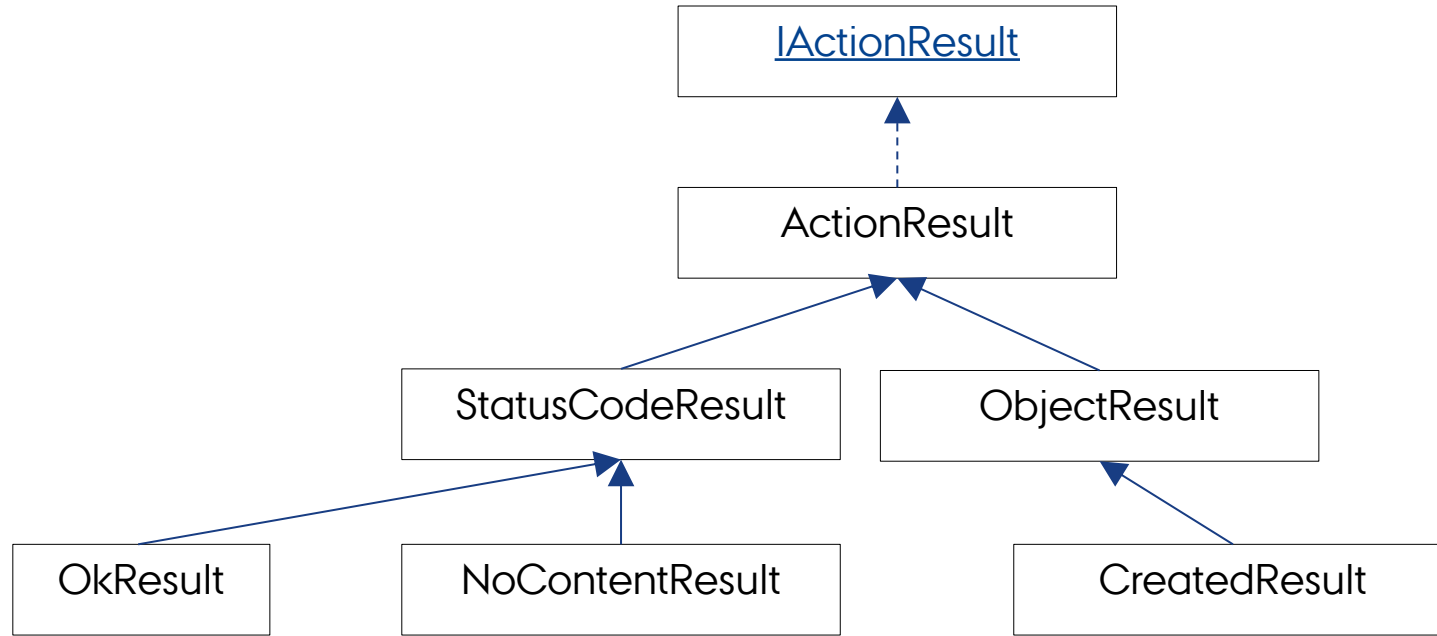
HTTP RESPONSE CODES

1. [Informational responses](#) (100 – 199)
2. [Successful responses](#) (200 – 299)
 1. 200 OK
 2. 201 CREATED
 3. 204 NO CONTENT
3. [Redirection messages](#) (300 – 399)
 1. NOT MODIFIED
4. [Client error responses](#) (400 – 499)
 1. 403 FORBIDDEN
 2. 404 NOT FOUND
5. [Server error responses](#) (500 – 599)
 1. 500 INTERNAL SERVER ERROR

IActionResult

Helper methods for creating Results

Ok(), NoContent(), etc...



EXERCISES

Continue on **1** and **2**

Solve **3**->

Jump to Bonus when
done





AARHUS
UNIVERSITY