2020

ASP.NET Core Blazor

Skrevet af: Mark Svendstrup Knudsen
Vejleder: Henrik Kryger Høltzer
Skole: Zealand Erhvervsakademi
Uddannelse: Datamatiker, 4. semester
Afleveret: 29-05-2020
Anslag: 23199 / 24000

Indholdsfortegnelse:

Introduktion

Web Applications er indenfor de sidste par år blevet en af de helt store spillere, når man vil udvikle eller lave en Applikation. Grunden til det er, at smartphones har haft en stor teknologisk udvikling, hvilke har resulteret i at brugen af browsere på smartphones er steget voldsomt, og når man ville udvikle en Web Applikation, som har funktioner der skal afvikles i browsere på klient siden, så har det indtil for nyligt kun været muligt via JavaScript baseret frameworks, som Angular, React og Vue.

Så for nogle år siden fik en mand ved navn Steve Sanderson, som arbejder for Microsoft, en god ide. Kunne det være muligt at køre .NET via WebAssembly, som er den nye standard i de fleste browser, så man f.eks. kan udskrive C# i en browser. Det virkede og det blev så til det Steve Sanderson kaldte for Blazor (navnet er en form for sammensætning mellem browser og razor).[1]

Jeg har tænkt mig i denne synopsis at finde ud af hvad Blazor er og hvordan det fungere. Jeg vil specielt fokusere på hvordan man laver en Web Applikation i Blazor som kører på klient siden, det vil sige at den fungere lige præcis som f.eks. en Angular Web Applikation. En ny og bedre måde at udvikling Web Development er for mig altid udfordrende og spændende, og efter min mening er Blazor det mest lovende og spændende inden for Webudvikling. Personligt syntes jeg at Blazor lyder meget interessant og har efter min mening et stort potentiale i fremtiden, men da det også er så nyt og uprøvet, så kan jeg ikke vide mig sikker på, at alt i mit forløb vil foregå succesfuldt. Det er både spændende og udfordrende, og det er også en af grunde til at jeg har valgt at undersøge dette emne i min synopsis.

[1] https://devchat.tv/adventures-in-dotnet/net-003-blazor-with-daniel-roth/ - starter fra 12:53 til 14:50

Problemformulering

Hvordan laver man en Web Applikation i Blazor, som kører på klient siden:

- Hvad er Blazor og hvordan fungere den?
- Kan man bruge Blazor til at lave Web Applications?
- Kan Blazor konkurrere med f.eks. React eller Angular?
- Og hvordan ser fremtidsmulighederne ud for Blazor?

Metoder

Her vil jeg klargøre hvilke metoder jeg har tænkt mig at benytte for at svare på min problemformulering. Jeg vil benytte mig af følgende metoder:

- **Research:** Da Blazor er så nyt som det er, så er research delen nok det jeg kommer til at bruge mest tid på. Jeg vil læse diverse brugbare materialer, som artikler og vejledninger. Jeg vil se videoer og tutorials for at få en bedre indlæring i hvordan man koder i Blazor. Hvis jeg finder brugbare lydoptagelser som f.eks. podcast så vil jeg også benytte mig af denne metode, da jeg føler at en af de bedste læringsprocesser som fungere for mig, er at lytte til folk som kan forklare det på en nem og direkte måde.

- **Test og prototyper:** Jeg vil eksperimentere mig med at udvikle prototyper af en Blazor Web Application, dette er en vigtig del af en læringsproces, og jeg vil nok bruge tid på at lave mange forskellige typer Blazor projekter, hvis jeg føler det nyttigt.

- **Skrivning af synopsen:** Jeg vil aflægge tid til at skrive min synopsis. Dette er min dokumentation på alt det jeg har lavet og researchet, jeg vil også forklare diverse elementer som jeg har kommet frem til via min research, samt fremvise kode fra mit arbejde.

Planlægning

I dette kapitel vil jeg beskrive, hvordan jeg har tænkt mig at planlægge mit forløb. De 2 overordnet fokusområder i dette forløb er at finde svarene til de spørgsmål jeg har stillet, og så senere hen nedskrive det ind i synopsen.

Jeg har tænkt mig så vidt muligt at overholde nedenstående skema, men vil ikke udelukke at undtagelser kan indtræffe, hvor jeg f.eks. skriver lidt i synopsen under min research eller i weekenden.

Uge	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag
18	Research af Blazor, samt teste og eksperimentere med at udvikle Blazor applikation.					weekend	weekend
19	Research af Blazor, samt teste og eksperimentere med at udvikle Blazor WebAssembly applikation.					weekend	weekend
20	Teste og eksperimentere med at udvikle Blazor WebAssembly applikation.					weekend	weekend
21	Teste og eksperimentere med at udvikle Blazor WebAssembly applikation.					weekend	weekend
22	Skrivning hvad jeg har researchet og udviklet ind i synopsen.				Aflevering inden kl. 11:00	weekend	weekend

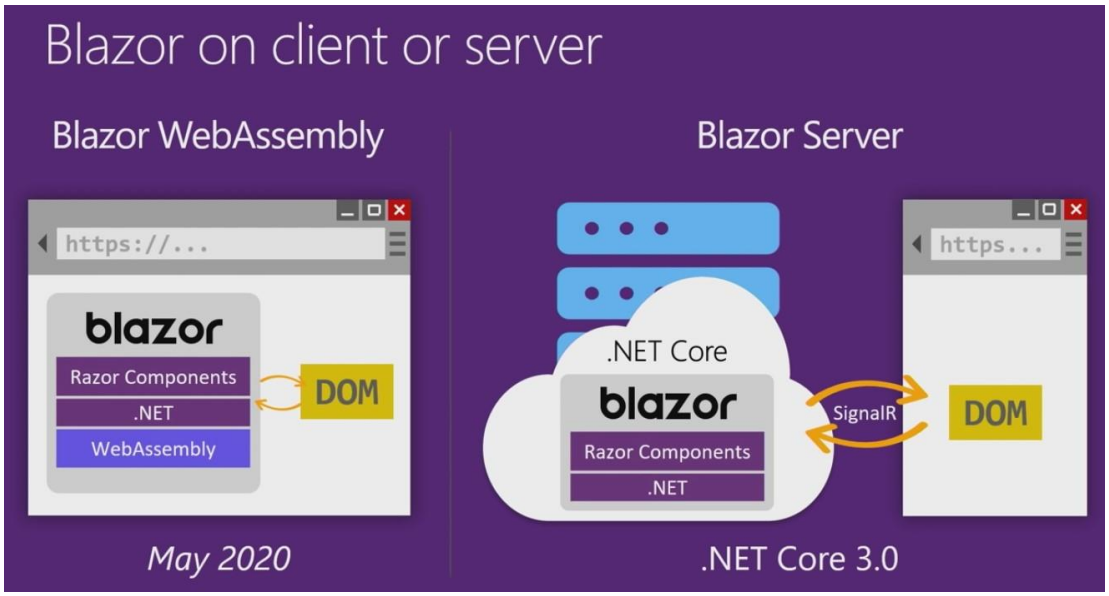
Research

Nu skal jeg i gang med min research. I de næste par sider vil jeg fokusere på at beskrive diverse elementer og funktioner i Blazor, samt hvad Blazor.

Hvad er Blazor?

ASP.NET Core Blazor er et nyt SPA framework fra Microsoft, der bruges til at bygge interaktive Web Applications på klientsiden. Traditionelt har man lavet Web applikationer ved at man programmeret serveren i f.eks. C# eller Java, og så var klientsiden programmeret i JavaScript frameworks som Angular eller React. Her har Blazor sin første fordel ved at både serversiden og klientsiden kan programmeres i C#, der er et programmeringssprog til forskel fra JavaScript der er et scripting-sprog, dette gør at man kan lave Full Stack Development ved kun at have forståelse for .NET og C#.

En anden fordel er at man kan tilgå og bruge alle de Libraries som .net tilbyder, hvilke kan vise sig at være meget brugbart når man vil udvikle store applikationer. Blazor findes indtil videre i 2 forskellige udgaver med hver deres Hosting Models, Blazor Server og Blazor WebAssembly. Begge kan arbejdes og eksperimenteres med lige nu.[2]



Blazor Hosting Models[3]

[2] https://www.udemy.com/course/aspnet-core-blazor-the-big-picture/ - ca. 1:10 I videoen.

[3] https://www.infoq.com/news/2020/02/blazor-webassembly-preview/

Blazor Server

Ved lanceringen af ASP.NET core 3.0 blev Blazor Server en integreret del, og kan lige nu bruges til at lave applikationer med i Visual Studio eller Visual Studio Code. Modsat Blazor WebAssembly bliver ændringer af UI i en applikation udført ved hjælp af en SignalR forbindelse, det vil sige at kommunikationen mellem browseren og serveren sker via SignalR.

Nogle af fordelene ved at benytte Blazor Server overfor Blazor WebAssembly, er at den har mindre filer den skal downloade og derfor er hurtig at loade, den behøver ikke bruge WebAssembly og så kan Serveren håndtere hemmeligheder som f.eks. passwords, hvilket altid er fortrukket i Serveren så igen kan opfange den selv om de skulle være krypterede.[4]

Blazor WebAssembly

Hvad er WebAssembly?

Som navnet angiver benyttes WebAssembly i Blazor WebAssembly, og selv om man skulle tro det, så er WebAssembly ikke et Assembly sprog.[5] WebAssembly eller også kaldt WASM som er et binært instruktionsformat der kører på en portable virtuelt stack maskine. High-level kodesprog som f.eks. C, C++, C# eller Rust bliver Compiled til bytecode, som så kan køres via internettet på en klient eller server applikation. Alle de store browser i dag som Chrome, Firefox og Edge kan køre WebAssembly.[6]

Hvordan fungere Blazor WebAssembly?

Blazor WebAssembly er nok den mest spændende af de 2 udgaver, da dette er en konkurrent til f.eks. Angular eller React. Blazor WebAssembly er indtil videre lige nu i mit forløb i preview mode, men planen er at den vil blive integreret en gang i maj 2020.

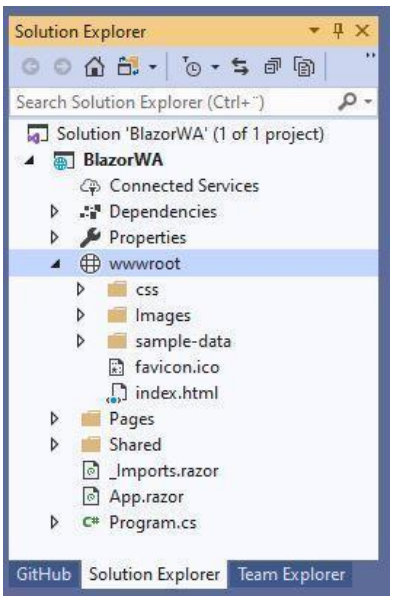
Blazor WebAssembly foregår ved at man downloader en version af Mono .NET runtime, samt applikationens DLL'er og Dependencies. Når det så er gjort, bliver Mono .NET runtime bootstrapped og derefter loades og udførtes dll'erne. Det er kun Mono .NET runtime der bliver compilet indtil WebAssembly, men udviklerne af Blazor er i gang med at kigge på hvordan man eventuelt kan give udvikler muligheden for at få sin app compilet om til WebAssembly.

I forhold til størrelserne på hvad der downloades, så fylder runtimen ca. 2,4 mb hvilket egentlig er ret imponerende få en .NET runtime, men til gengæld ikke så imponerende når man sammenligner med JavaScript. Man håber fra udviklernes side at når Blazer bliver udgivet til maj, så vil den fylde betydeligt mindre.[7]

[4] https://docs.microsoft.com/en-us/aspnet/core/release-notes/aspnetcore-3.0?view=aspnetcore-3.1
[5] https://cynicaldeveloper.com/podcast/122/ - starter fra 3:13 og slutter 4:55
[6] https://webassembly.org/
[7] https://stackoverflow.blog/2020/02/26/whats-behind-the-hype-about-blazor/

Hvordan Blazor ser ud

For at få et indblik af hvordan Blazor WebAssembly fungere, så kan man kigge i Solution Explorer for at se hvordan projektet er opdelt. Blazor benytter komponenter ligesom React eller Angular, og Blazor bruger Razor Components som man kan se på endelsen af alle filerne. Razor indeholder flere ting end implementeret her i Blazor, men det væsentlige her i Blazor er at Razor muliggøre at HTML og C# kan sammenkobles i filerne.



Blazor ligner lidt hvad man måske kender til og kan måske sammenlignes med f.eks. MVC, og Blazor har mappen Pages, hvor man lægger komponenter som kan tilkobles diverse sider på en hjemmeside. Derudover har vi mappen Shared, som indeholder de generelle komponenter som NavMenu.razor, hvor man indsætter en route og MainLayout.razor der har med sidens primære layout. For at få et indblik i hvordan Blazer virker så kan man kigge i filen Counter.razor. Man har 3 opdelinger i filen, Routing, HTML og C# koden. Oppe i toppen defineres routing ved @Page, @ indikerer at man bruger razor og betyder at man kan bruge C# kode iblandt HTML, der findes visse definerede keywords som page, if eller foreach.

I Blazor kan man bruge Razor på 3 forskellige måder, først kan man bruge en Implicit Expression, det er hvor man referere til f.eks. en enkelt variabel som har en værdi som kan bruges. Så kan man bruge en Explicit Expression, som benytter parenteser så man kan have mellemrum og lave mere end kun at referere til en variabel, og sidst har vi Code Block. Code Block er der hvor der kodes mest i C#, og her laver jeg f.eks. som eksemplet viser funktionerne, og til forskel til implicit og Explicit så renderes intet her.[8]

```
@page "/counter"

<h1>Counter</h1>

<p>Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
        private int currentCount = 0;

        private void IncrementCount()

        {

            currentCount++;

        }
}
```

[8] https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-3.1

Lifecycle Metoder

I Blazor findes der det man kalder Lifecycle metoder, som man bl.a. også kender fra andre JavaScript frameworks. Det er forskellige synkrone og asynkrone metoder der udarbejder forskellige arbejde på forskellige komponenter når de initialiseres eller renderes, og der er indtil videre 8 Lifecycle metoder i Blazor:

```
1. OnInitialized()
2. OnInitializedAsync()
3. SetParametersAsync(ParameterView parameters)
4. OnParametersSetAsync()
5. OnParametersSet()
6. OnAfterRenderAsync(bool firstRender)
7. OnAfterRender(bool firstRender)
8. ShouldRender()
```

Metoderne gør forskellige ting f.eks. så bruges `OnInitialized` og `OnInitializedAsync` når komponenter initialiseres efter at have modtaget parametre fra sin Parent komponenten, forskellen på de 2 metoder er som navnet angiver at `OnInitializedAsync` udføres asynkront.[9]

Det generelle i Blazor Lifecycle er, at først så kaldes de indskudte services, parametre og underkomponenter læses derefter og så til sidst kan komponenten renderes. Hvis der så sker ændringer med komponenten, så genereres den på ny. Når komponenten ikke bruges, så bortskaffes den indtil den så kan blive kaldt på ny og genskabes.[10]

Data Bindings i Blazor

I Blazor er der 3 forskellige data bindings man kan benytte, one-way binding, two-way binding og så event binding. Man bruger data binding til at forbinde mock-up delen med kode delen, f.eks. hvis man skal bruge en int eller string til at få udført et specifikt arbejde, eller hvis der skal ske noget når man trykker på en knap.

One-way binding, bruges når der kun er en envejs forbindelse modsat two-way binding, hvor data 'en er forbundet begge veje. Man bruger event binding til at udføre funktioner.[11]

[9] https://docs.microsoft.com/en-us/aspnet/core/blazor/lifecycle?view=aspnetcore-3.1

[10] https://www.linkedin.com/learning/blazor-first-look/component-life-cycle

[11] https://www.udemy.com/course/aspnet-core-blazor-the-big-picture/

Eksperimentering af Blazor WebAssembly

Installation og opstart af Blazor

Lige nu er det kun Blazor server som er produktions dygtig via .NET Core 3.1 LTS, men planen er at Microsoft ASP.NET teamet vil udgive Blazor WebAssembly en gang i maj i år, hvilket er midt inde i min synopsis så det bliver spændende at følge med i.

Hvis man så vil gå i gang med at arbejde med en Blazor WebAssembly app, så kan man godt det, man skal bare installere den seneste version af .NET Core 3.1 SDK (version 3.1.201 eller nyere), samt en Blazor WebAssembly 3.2.0 Preview (jeg benytter preview 5).

Først kan man se hvilken version af .NET Core SDK man har installeret ved at skrive nedenstående i cmd.

```
dotnet --version
```

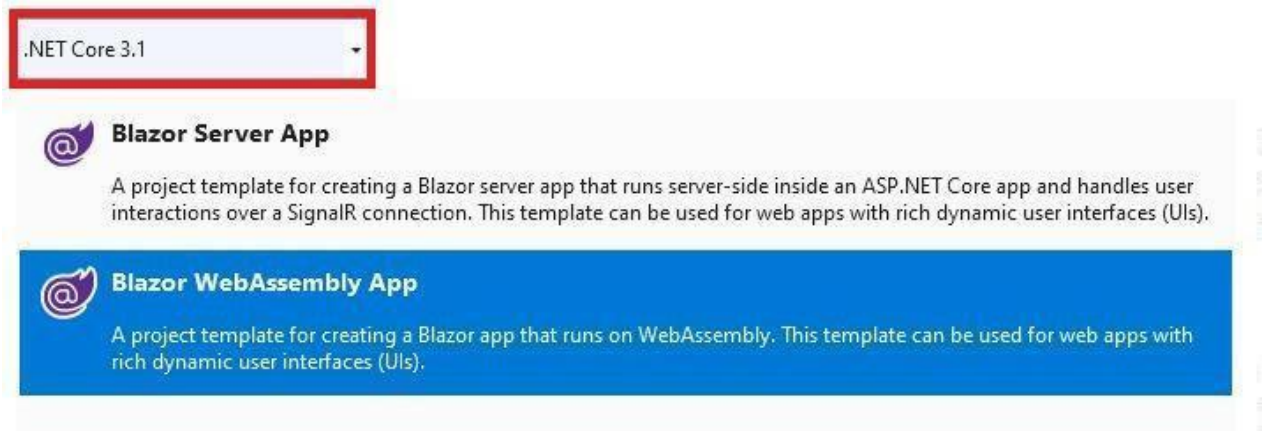
Efter man har verificeret at man har den rigtige SDK, så skriver man følgende i cmd:

```
dotnet new -i Microsoft.AspNetCore.Components.WebAssembly.Templates::3.2.0-preview5.20216.8
```

Så er man klar til at lave en Blazor WebAssembly applikation fra Visual Studio 2019 eller Visual Studio Code hvis man foretrækker det.[12]

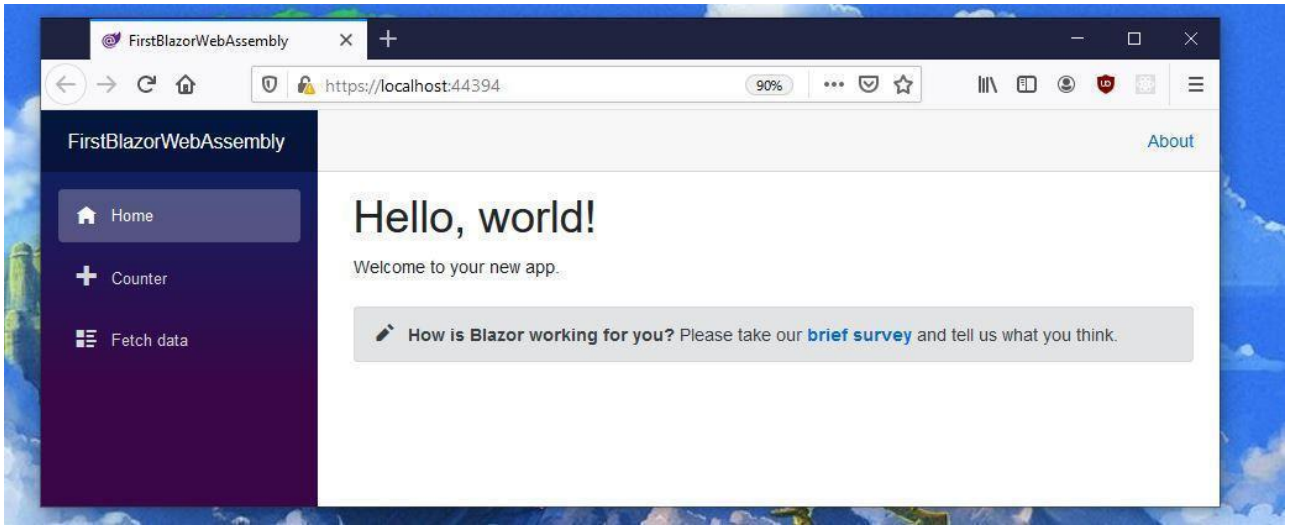
Opret et Blazor WebAssembly Projekt

Når man har valgt Blazor App, går man videre og vælger Blazor WebAssembly App, det er vigtigt at man husker at se hvilken version af .NET Core man har valgt, den skal sættes til .NET Core 3.1.



[12] https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-preview-5-release-now-available/

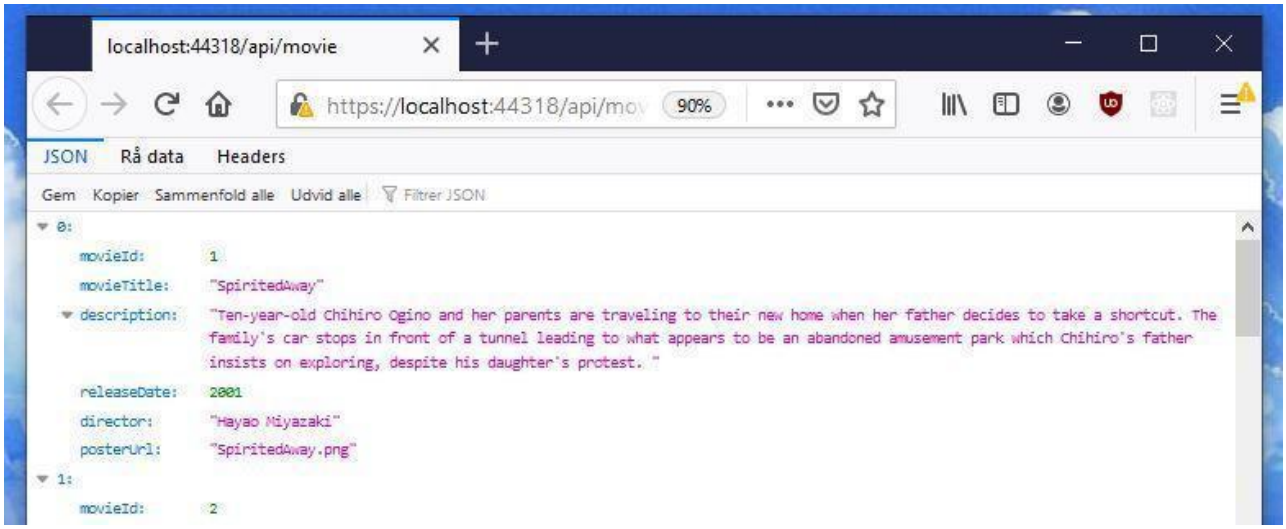
Når projektet er færdigt med at blive genereret, så kan man teste at projekt virker ved at køre projektet og åbne den i en browser.



Ligesom Angular eller React så er der udarbejdet en template. Her ser man hvordan Blazor er opsat med en sidemenu hvor der er tre knapper, Home der er forsiden, Counter som går til allerede beskrevet komponent Counter og så Fetch data som demonstrere en visning af hentet vejrdata. Til højre findes Main Body, det er her hvor Razor Pages/komponenter vises.

Lav en Blazor WebAssembly der henter og viser data

Noget af det som jeg tænker at jeg ville prøve at udvikle i Blazor, er nok det mest essentielle man kan forestille sig nemlig http request til en web api. I denne del af min eksperimentering vil jeg benytte mig af en web api, som jeg allerede har udviklet på forhånd. Jeg vil gøre brug af min MovieWebApi, som indeholder en statisk liste af movie objekter som jeg kan tilgå og ændre.



Efter jeg har genereret et nyt projekt, så starter jeg med at lave en ny Razer Component, som jeg kalder Movies. For at kunne hente mine Movie objekter fra min api, så skal jeg implementere HttpClient. Hvis ikke HttpClient allerede en implementeret i Blazor så tilføjes den i program.cs filen. Jeg henter HttpClient ved brug af inject, jeg skriver følgende under @page "/movies":[13]

```
@inject HttpClient Http
```

Jeg laver en ny klasse Movie, og laver en variabel af den og tildeler den så mine Movie objekter. Jeg benytter mig af en Lifecycle metode til at lave en Get request til min api. Jeg skriver følgende kode inde i @code:

```
@code{
    private Movie[] movies;
    override async Task OnInitializedAsync()
    {
        movies = await Http.GetFromJsonsync<Movie[]>("https://localhost:44318/api/movie");
    }
}
```

Nu hvor min Movies.razor komponent for lavet en instans af mine Movie objekter så skal jeg have dem vist på siden, jeg bruger et foreach loop. Jeg skriver følgende ovenover @code:

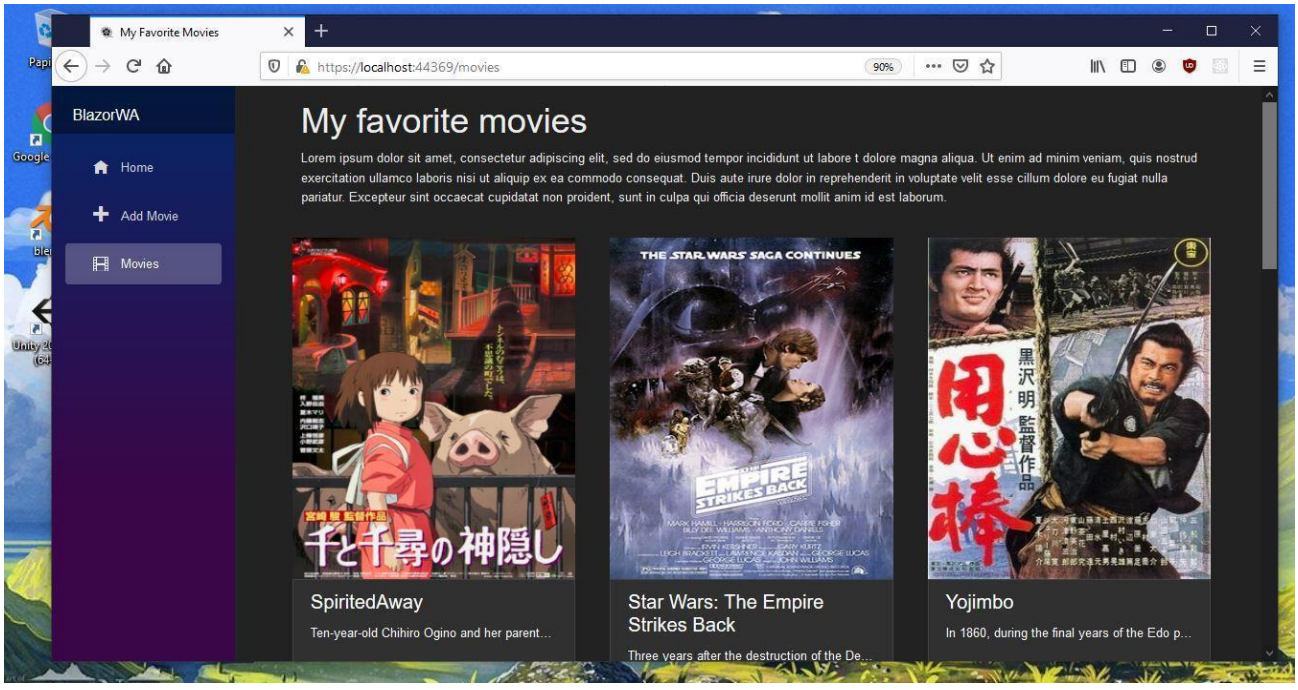
```
<div class="card-deck">
        @foreach (var movie in movies)
        {...
        ...}
</div>
```

[13] https://docs.microsoft.com/en-us/aspnet/core/blazor/call-web-api?view=aspnetcore-3.1#httpclient-and-json-helpers

Nu gennemløber jeg mit array af Movie objekter og kan så nu udskrive noget efter behov. Jeg skriver følgende inde i foreach loopet:

```
<div class="card-body border-primary col-lg-4 col-md-3 col-sm-12 div-card">
  <div style="cursor: pointer;">
    <img src="Images/MoviePoster/@movie.PosterUrl.ToString()" width="100%" style="height:
        400px;">
  </div>
  <div class="list-group-item" style="min-height: 200px; cursor: pointer;">
    <h4 class="card-title">@movie.MovieTitle</h4>
    <p class="card-text" style="white-space: nowrap; overflow: hidden; text-overflow: ellipsis;">
      @movie.Description
    </p>
  </div>
</div>
```

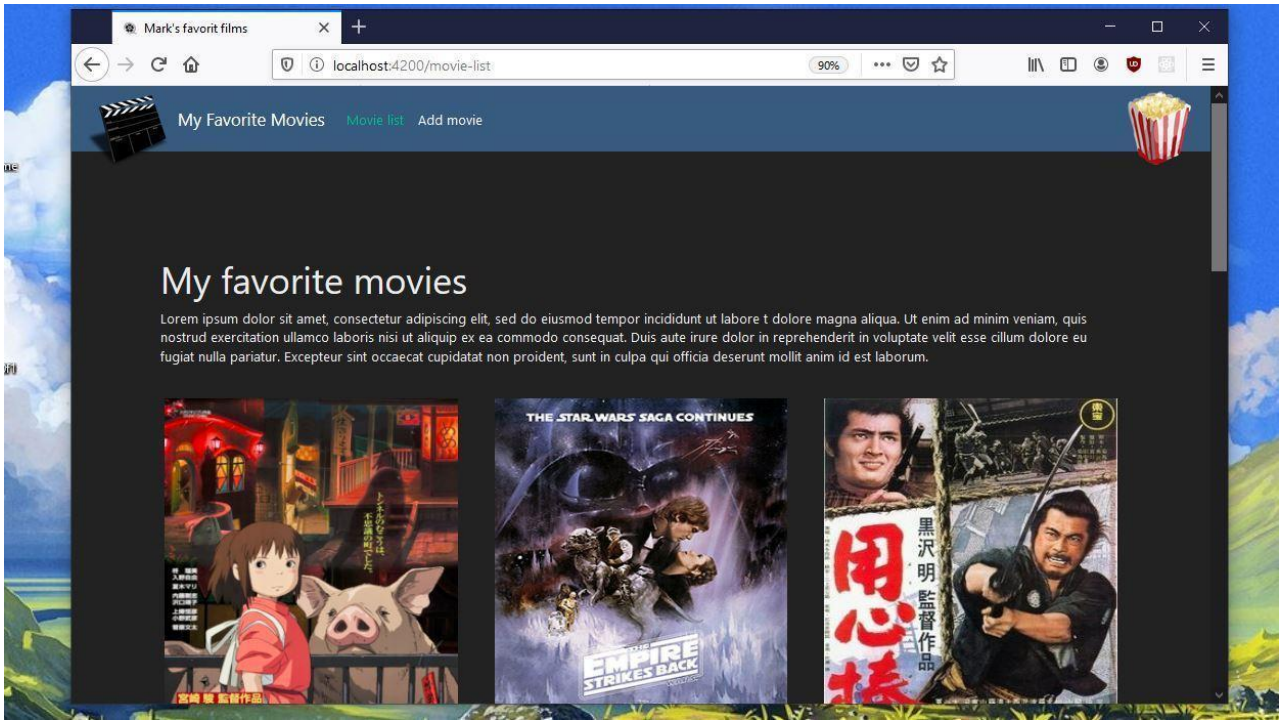
Jeg benytter mig af bootstrap til at lave en nogenlunde fin opstilling. Og med det, så har jeg lavet mit første http request i Blazor. Resultatet af min Movies.razor komponent vist i browseren:[14]



[14] https://www.linkedin.com/learning/blazor-first-look/basic-components

Sammenligning med Angular Web Application

For at se lighederne så vil jeg sammenligne med et Angular projekt som viser det samme. I min Angular web app henter jeg Movie objekter fra samme api MovieWebApi, og viser dem ligesom min Blazor App i Cards via bootstrap.



Som vist så kan Blazor vise nøjagtigt det sammen som Angular og når Blazor WebAssemly bliver produktions dygtig, så vil den helt klart kunne konkurrere med de populære JavaScript frameworks som Angular.

Blazor har den fordel overfor JavaScript at klienten og serveren kan være under samme projekt, til gengæld hvis vi kigger på Blazor server så er den afhængig af JavaScript til at lave SignalR forbindelser. En anden ting er, at hvis man har udviklet nogle gode metoder eller klasser, så kan de blive delt mellem klienten og serveren så man kun skal lave dem en gang og rette et sted hvis det bliver nødvendigt.[15]

[15] https://www.telerik.com/blogs/why-you-should-use-blazor-over-javascript-frameworks-to-build-your-single-page-application

Mislykkede forsøg med Blazor

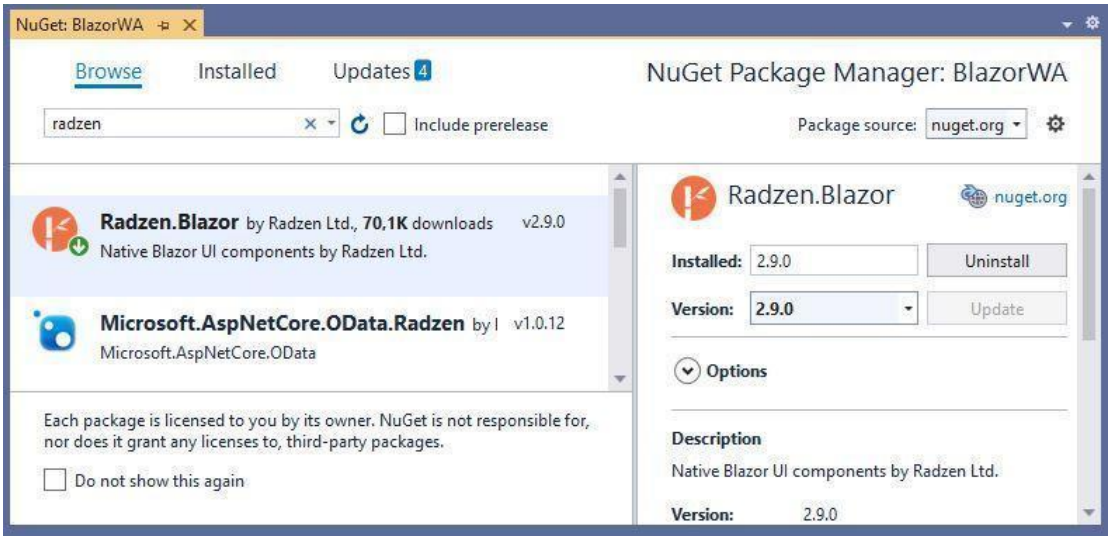
I denne sektion af min synopsis vil jeg forklare nogle af de ting som jeg har eksperimenteret med og prøvet at implementere i min Blazor applikation, men som jeg ikke har kunne fået til at fungere enten fordi jeg ikke har fået den fulde forståelse for givet emne eller har valgt at fokusere på noget andet.

Routing med parameter

Jeg har prøvet at arbejde med parameter i routing, Jeg ville gerne kunne vælge og trykke på en knap, så jeg kommer over til en anden side hvor jeg kunne få set informationer om det valgte objekt. Jeg har tilføjet til mit foreach loop en knap, som man kunne trykke på og så gav jeg titlen fra det valgte Movie objekt som parameter, så jeg ud fra det kunne lave et nyt Get Request når man instantiere siden der skulle vise et objekt.

Genanvendelige Blazor UI Components

Allerede nu findes der mange forskellige leverandører af UI komponenter til Blazor, mit fokus har været at kigge på en af dem som jeg vil eksperimentere med. Radzen er en helt gratis måde at for tilgang til UI komponenter, man kan lave alt fra pop-up bokse og notifikationer til grafer og charts. For at få tilgang til Radzen så skal man installere en NuGet Package.[16]



Jeg har kun kort prøvet at implementere en Radzen komponent. Min fornemmelse er at det ikke skulle være alt for kompliceret at bruge, men det har i den korte tid hvor jeg har undersøgt den ikke lykkes mig at få det til at fungere.

[16] https://blazor.radzen.com/get-started

Fremtiden for Blazor

Blazor WebAssembly er endelig kommet. Fra og med d. 19. maj i år er Blazor WebAssembly blevet implementeret i .NET Core 3.1, og er nu produktions dygtig til at lave web applications.[17] Det betyder at man nu kan arbejde med 2 udgaver af Blazor som er supported, og Microsoft er allerede i gang med at kigge på andre muligheder for at bruge Blazor.

Fremtiden for Blazor kan være svær at specificere, men Blazor er helt sikkert kommet for at blive. Og selvom JavaScript er det mest udbredte sprog (ca. 69 %), så tror jeg at med tiden så kan Blazor blive en af de mest populære web frameworks på markedet.

Det næste som Microsoft er i gang med at eksperimentere med, er noget der hedder Blazor Electron og Mobile Blazor. Mobile Blazor er som navnet angiver tiltænkt, så man kan udvikle Blazor applikationer til mobiltelefoner. Blazor Elektron er til udvikling af desktop applikationer til computer og lignende.[18]

[17] https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-now-available/

[18] https://fossbytes.com/javascript-most-popular-programming-language-stack-overflow/

Konklusion

Efter 5 ugers undersøgelse og eksperimentering af ASP.NET Core Blazor, så vil jeg kort lave en konklusion af hvad jeg har fundet ud af og besvare min problemformulering. Jeg havde følgende spørgsmål jeg ville besvare:

1. Hvad er Blazor og hvordan fungere den?
2. Kan man bruge Blazor til at lave Web Applications?
3. Kan Blazor konkurrere med f.eks. React eller Angular?
4. Og hvordan ser fremtidsmulighederne ud for Blazor?

1) ASP.Net Core Blazor er et nyt web application framework som kan bruges til at udvikle Web Applications i Browseren. Microsoft giver to muligheder for at udvikle Web Applications. Den ene med, Blazor Server og den anden med Blazor WebAssembly.

I stedet for at dele en Full Stack Development proces op i to, hvor serveren skal laves i et programmeringssprog (f.eks. C#) og så skal klientdelen laves i et JavaScript framework, så kan Blazor samle alt i et programmeringssprog nemlig C#. Dette faktum er nok det mest interessante i mine øjne, da arbejdsgiver fremover nu kan optimere processen når nye medarbejder skal oplæres, hvor der nu kan spares ressourcer og tid.

2) I forhold til mit arbejde med Blazor WebAssembly, så har det for det meste kunne fungere, men fordi Blazor både har været i en preview udgave og det er et nyt arbejdsværktøj til softwareudvikling, så har der været visse ting jeg har haft problemer med. Det er bl.a. Routing med parameter, hvor jeg ville vælge en Movie objekt og så ud fra en titel eller id kunne få vist informationer om netop det valgte objekt.

3) Kan Blazor konkurrere med JavaScript? Med tiden så er svaret helt klart ja, men indtil videre så er JavaScript det mest udbredte sprog indenfor webudvikling, og så har Blazor Server stadig brug for JavaScript for at fungere. Den største positive melding med Blazor er at man kan udvikle Full Stack Development ved brug af kun et sprog, og der er noget som jeg tror virksomheder godt kan se kan være nyttigt og med til at gøre produktionen nemmere og hurtigere.

4) Fremtiden for Blazor ser lys ud. Fra og med d. 19 maj så er både Blazor Server og Blazor WebAssembly produktions dygtig og bliver vedligeholdt. Det næste skridt for Blazor er at kunne komme ind på mobilen og ens desktop, og hvis alt går som det har gjort hidtil så vil Microsoft også levere her med Blazor Elektron og Mobile Blazor.

Efter dette undersøgende forløb af Blazor og med den fornemmelse og viden jeg havde om Blazor inden jeg gik i gang med synopsen, så har Blazor givet mig et indtryk at der både sker forandringer og forbedringer indenfor webudvikling. En ting er sikkert, det er ikke sidste gang jeg har beskæftiget mig med Blazor.

Refleksion

Til sidst vil jeg reflektere over hvordan mit arbejdsforløb er gået og hvilke udfordringer jeg har haft, samt fortælle om hvad der har fungeret godt for mig.

I forhold til min problemformulering så syntes jeg det var lidt svært at finde de helt rigtige ord til at beskrive hvad jeg skulle finde ud af, da Blazor netop er så nyt og for mange uudforsket. Overordnet har det for mig været at lære noget nyt.

De metodevalg jeg har truffet, har for mig været gode, jeg har researchet vidt og bredt, og har lært meget af alt det jeg har udforsket og eksperimenteret med. En af de ting som jeg har fået bevist igen, er at informative kilder og tutorials er en af de bedste læringsprocesser som fungere for mig. At se andre mennesker forklare dedikeret om enkelte elementer er både inspirerende og intuitivt, og derfor har jeg fået rigtig meget ud af at se diverse tutorials og læse artikler.

Min eksperimenteringsfase gik også nogenlunde som jeg forventede, men i visse tilfælde brugte jeg nok for meget tid på enkelte ting, som Routing hvor jeg havde problemer med at få vist enkelte objekter fra min Web APi. Det gjorde at min planlægning blev forskudt lidt ved at jeg har inddraget lidt af min weekend til at skrive på min synopsis, så jeg havde mere tid til at programmere og eksperimentere i Blazor. Det har dog ikke være et helt så stort problem, da jeg på forhånd havde forudset at sådan noget kunne ske.

En anden vigtig refleksion jeg har haft er noget der kan relateres til den situation vores land sammen med hele verden er vidne til, nemlig corona-virussen. Det, at landet i de seneste uger og måneder har været lukket ned, har gjort at jeg ikke har været så meget udenfor mit hjem. Det har påvirket mig sådan, at jeg efter mange dages computerarbejde fik ondt i kroppen, og jeg skulle have tænkt mere på mit velfærd. Dette er er nok den vigtigste refleksion, da det ikke er noget man lige sådan tænker over, men som kan have en stor betydning for ens arbejdsproces.

Litteraturliste:

Artikler:

Daniel Roth *Blazor WebAssembly 3.2.0 Preview 5 release now available*, https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-preview-5-release-now-available/, Besøgt d. 29/4/20 (Internet)

Brugt til at komme i gang med Blazor, hvor jeg installere SDK og preview.

Chris Sainty *What's behind the hype about Blazor?*, https://stackoverflow.blog/2020/02/26/whats-behind-the-hype-about-blazor/, Besøgt d. 2/5/20 (Internet)

Brugt til at give en kort forståelse for hvad Blazor WebAssebly er for noget.

Daniel Roth *Blazor WebAssembly 3.2.0 now available*, https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-now-available/, Besøgt d. 29/4/20 (Internet)

Brugt til at fortælle om Blazors nuværende tilstand.

Matthew MacDonald *The Future of Blazor is Native Apps*, https://medium.com/young-coder/the-future-of-blazor-is-native-apps-9f4a510acdd5, Besøgt d. 22/5/20 (Internet)

Brugt til at få et blik på Blazors fremtidsplaner.

Adarsh Verma *JavaScript Is The Most Popular Programming Language: Stack Overflow Survey,* https://fossbytes.com/javascript-most-popular-programming-language-stack-overflow/, Besøgt d. 22/5/20 (Internet)

Brugt til at fortælle anvendelsens procenten af JavaScript.

David Grace *Why You Should Use Blazor over JavaScript Frameworks to Build Your Single-Page Application,* https://www.telerik.com/blogs/why-you-should-use-blazor-over-javascript-frameworks-to-build-your-single-page-application, Besøgt d. 23/5/20 (Internet)

Brugt til at sammenligne Blazor med JavaScript frameworks.

Hjemmesider:

Microsoft *What's new in ASP.NET Core 3.0,* https://docs.microsoft.com/en-us/aspnet/core/release-notes/aspnetcore-3.0?view=aspnetcore-3.1, Besøgt d. 2/5/20 (Internet)

kort beskrivelse af Blazor Server er taget fra denne beskrivelse.

Microsoft *ASP.NET Core Blazor lifecycle,* https://docs.microsoft.com/en-us/aspnet/core/blazor/lifecycle?view=aspnetcore-3.1, Besøgt d. 12/5/20 (Internet)

Brugt denne hjemmeside fra Microsoft til at forklare Blazor Lifecycle Methods.

https://webassembly.org/, Besøgt d. 27/4/20 (Internet)

Jeg har brugt denne siden til at kort at fortælle hvad WebAssembly er.

https://blazor.radzen.com/get-started, Besøgt d. 23/5/20 (Internet)

Brugt denne hjemmeside til at få en forståelse for Blazor Radzen.

Microsoft *HttpClient and JSON helpers,* https://docs.microsoft.com/en-us/aspnet/core/blazor/call-web-api?view=aspnetcore-3.1#httpclient-and-json-helpers

Brugt til at lave HTTP request in Blazor.

Podcast:

Daniel Roth Adventures In Dotnet *.NET 003: Blazor with Daniel Roth*, https://devchat.tv/adventures-in-dotnet/net-003-blazor-with-daniel-roth/, Besøgt d. 28/4/20 (Internet) - starter fra 12:53 til 14:50

Jeg har brugt denne podcast, hvor Daniel Roth der arbejder for Microsoft, fortæller kort om Blazor og dens historie.

Guy Royse *what is WebAssembly,* https://cynicaldeveloper.com/podcast/122/, Besøgt d. 3/5/20 (Internet) starter fra 3:13 og slutter 4:55

Kort brugt til at forklare WebAssembly.

Videoer:

Ervis Trupja *AspNet Core Blazor: The Big Picture*, *What is Asp.Net Core Blazor?*, https://www.udemy.com/course/aspnet-core-blazor-the-big-picture/, Besøgt d. 30/4/20 (Internet)

Jeg har brugt denne gratis kursus på Udemy til at få en kort og klar forståelse over, hvad Blazor er og hvordan den fungere.

Ervis Trupja *AspNet Core Blazor: The Big Picture*, *AspNet Core Blazor Data Binding*, https://www.udemy.com/course/aspnet-core-blazor-the-big-picture/, Besøgt d. 26/5/20 (Internet)

Brugt denne video til at forstå data binding i Blazor.

Richard Goforth *blazor first look, razor-overview,* https://www.linkedin/learning, Besøgt d. 5/5/20 (Internet)

Brugt til at forklare Razor, og hvordan Blazor benytter det.

Richard Goforth *blazor first look, basic-components,* https://www.linkedin/learning, Besøgt d. 25/5/20 (Internet)

Brugt til at forklare components i Blazor.

Billeder:

https://xamariners.com/community/building-100-c-and-html-web-assembly-apps-with-blazor-bye-bye-javascript/

Jeg har brugt billedet til forsiden.

https://www.udemy.com/course/aspnet-core-blazor-the-big-picture/ - fra ca. 1:10 I videoen

Brugt til at vise Blazor Hosting Models.