Webprogrammering || Node.js & Dataintegration

af Rainbow Warriors

Indholdsfortegnelse

Indledning og problemformulering

Som vores første projekt på andet semester har vi fået en opgave, hvor vi skal benytte nye såvel som gamle færdigheder, vi har lært på uddannelsen indtil videre. Opgaven lyder på at lave en hjemmeside, hvor man kan finde en liste over bøger, forfattere og deres tilhørende informationer, samt udvikle en mulighed for, at man selv kan indtaste bøger eller forfattere derinde. Der skal primært bruges færdigheder, vi har lært på 2. semester, hvilket i dette tilfælde vil sige XML og Node. Dertil har vi udarbejdet følgende problemformulering:

Hvordan kan vi lave en dynamisk hjemmeside ved hjælp af Native Node og XML, der kan modtage og gemme brugerinputs og printe dem ud?

Metoder og research

Umiddelbart lige efter, vi hørte om opgaven, begyndte vi vores research fase. Her satte vi os ned og gennemgik noget af det kode, vi har arbejdet med indtil videre, med fokus på det, vi har lært om XML og Node, for at finde noget inspiration. Derefter skrev og snakkede vi sammen for at gå igang med at tackle opgaven.

Kommunikation gennem Microsoft Teams

Et af vores første skridt var, at lægge en plan for, hvordan vi bedst muligt skulle håndtere og løse opgaverne i løbet af en uge. Det gjorde vi ved at have et møde på Microsoft Teams for at snakke det hele igennem, imens vi lavede en Google Drive mappe til projektet, hvor vi også oprettede de dokumenter, vi skulle bruge - herunder bl.a. et skema med en tidsplan og en model over, hvad vores sider skulle indeholde. Grunden til, at vi her valgte at primært benytte Microsoft Teams og Google Drive er, at det gjorde det meget lettere for os at dele vores tanker og ideer med hinanden, samtidig med at vi kunne skrive på de samme dokumenter og tale sammen.

Google Drive er derudover også brugt til at holde styr på alle vores filer og dokumenter, som ikke direkte er vores kode - den havde vi liggende i et GitHub repository i stedet, ligesom vi plejer fra undervisningen.

Oprettelse af en træstruktur

Som noget af det første gik vi ind og lavede en træstruktur for de dokumenter, som vi vidste, vi skulle oprette. Det har gjort, at vi tidligt har fået et overblik over, hvordan dokumenterne skulle opbygges og deres indhold. Det har derfor gjort det nemt for os, da vi gik ind og begyndte og kode.

Live Sharing i Visual Studio Code

Til at programmere vores projekt har vi alle benyttet Visual Studio Code, mere eller mindre. Simpelthen bare fordi, det er det program, de fleste af os er vant til, og fordi det har mange fordele i form af bl.a. plugins. Plugins som Live Share, som gør, at man kan sidde og kode i det samme projekt og se ændringerne fra alle medlemmer live. På den måde har vi undgået mange problemer, der normalt kan opstå ved at lave mange push eller pull commands igennem Git, såsom meget trælse merge problemer.

Live Share kombineret med Microsoft Teams' mulighed for skærmdeling har derfor virkelig været nogle stærke midler at bruge for os, da vi sparede en del tid og frustrerende situationer - selvom vi dog stadig havde en del af sidstnævnte. Andet ville jo næsten være kedeligt.

Fejlfinding/linting

I løbet af arbejdsprocessen har vi også benyttet os meget af linting for at løse fejl, der opstod undervejs. Til dette formål har vi både benyttet vores editors' indbyggede fejlfindingssystem, samt tjekket vores logs i terminalen, hver eneste gang, vi afprøvede vores Node server igennem den. Vi afprøvede serveren utallige gange, da det var vigtigt for at finde og rette de fejl, der måtte være. Faktisk havde det nok næsten været umuligt at løse opgaven, hvis vi ikke havde husket at teste vores server så meget, da det har hjulpet os til at løse vores problemer effektivt og hurtigt, så vi nåede det meste inden for vores bestemte tidsrum.

Især moduler og fejl i stier gav os problemer, da der var mange forskellige filer at holde styr på til sidst.

XML

XML står for Extended Markup Language, er en måde at formatere teksten på og tilføre HTML nogle flere egenskaber. XML er lavet til at strukturere data.
Indholdet af en XML-fil kan formateres med et stylesheet, så det ligner en hjemmeside.

I XML definerer man selv tags, som angiver præcist, hvilke informationer XML-filen skal indeholde. I eksemplet nedenfor har vi lavet et lille stykke XML kode som fortæller os, at vi har række data i form af forfatter og lidt info om ham.

```
<author>
  <authorname>
    <firstname>Ernest</firstname>
    <lastname>Hemingway</lastname>
  </authorname>
  <birthyear>1899</birthyear>
  <deathyear>1961</deathyear>
  <birthplace>Oak Park, Illinois</birthplace>
  <country>USA</country>
  <language>American English</language>
  <bio>Ernest Miller Hemingway was an American novelist,
</author>
```

Hvis man sendte dette til en XML applikation, så ville denne applikation kunne læse XML filen og trække netop denne data om forfatter eller bøgerne ud. Systemet, der modtager filen ved hvilken data der findes i hvert element.

Eksemplet ovenfor ville man f.eks. på forhånd have aftalt, at i <author></author> skulle forfatter fornavn og efternavn indsættes og i <birthyear></birthyear> skulle fødselsdatoen indsættes. For at være sikre på, at der er den rigtige data og data-formater, der anvendes i hvert element, anvendes der typisk en XSD (XML Schema Definition).

Validering af xml med xsd validering

En anden form for metode, som også benytter sig af linting, er validering af et xml dokument, der skal sættes op mod et xsd schema. Her sørger man for, at man benytter sig af de samme elementer med de samme navne og fortæller hvilken slags typer, der kan være f.eks. strings eller integers. Den kigger også efter hvor mange gange et bestemt element har lov til at optræde, og om de gør det rigtigt i forhold til skemaet.

Node.js

Node.js er open-source, som gør det muligt at bruge JavaScript server-side. JavaScript er i stand til at foretage en masse asynkrone kald, hvilket gør at det kan færdiggøre opgaver sideløbende med hinanden. Dette gør ydeevne meget hurtigere end de fleste nuværende populære programmeringssprog.

I vores projekt har vi benyttet Node.js til at lave en server, som vi har brugt til både at teste og køre vores kode. Den er altså også brugt som en forbindelse mellem vores browser og server, når vi har skulle tjekke resultatet af vores arbejde. En af de største fordele ved at benytte Node.js i vores projekt har været, at den ikke blokerer input og output.

Evaluering af process

Til at starte med fulgte vi vores tidsplan rimelig tæt, og følte os godt på vej i projektet. Dog blev det svært at overholde den senere på ugen, idet vi kæmpede med en del problemer, da vi bl.a. skulle få vores side til at fremvise vores indtastede oplysninger, altså inputs fra brugere, m.m. Sådan kan det jo gå med programmering, og det har klart været en udfordring at få tiden til at passe, men med blot en uge at gøre godt med, måtte vi bare gøre vores bedste og hænge i så godt, vi kunne.

Ud over at projektet har givet os erfaring med Dataintegration og at arbejde med Node, så har vi i hvert fald også lært lidt om at måske sætte mere tid af til eventuelle kode-relaterede udfordringer, der kan tage meget lang tid at komme igennem.

Mandag - 15	Tirsdag - 16	Onsdag - 17	Torsdag - 18	Fredag - 19
9:00-14:00	9:00-14:00	9:00-14:00	9:00 - 14:00	9:00
samlet web-app	author.xml books.xml	Input felter		fremlæggelse
samling af books.xml		Opdatering af XML med JSON	rapport	
skema til author	rapport	rapport	forberedelse til fremlæggelse	

Konklusion

Hvad kan projektet bruges til?

Ideen bag en opgave som denne, er at udvikle en database, som kan gemme og opbevare præcis det, man vil have den til, igennem en brugers inputs. Det er en procedure, man kan se ret mange steder i en computer-orienteret verden i dag, fx hos et biblioteks bogsystem eller en webshop. Eksempelvis benytter man sig af noget lignende, når man vil gemme noget på sin ønskeliste på en webshop. Det her er altså et projekt, som er meget brugbart og kan komme til at have mange forskellige funktioner alt efter, hvor og hvordan det bruges.

Vi har forsøgt at skabe netop sådan en database ved brug af vores nye færdigheder inden for bl.a. XML og Node, og nåede derfor frem til et resultat, vi er (nogenlunde) tilfredse med, samt et svar på vores problemstilling.

Referencer

Niels 

4. Selecting and Traversing - XSLT Cookbook [Book] (n.d.) available from <https://www.oreilly.com/library/view/xslt-cookbook/0596003722/ch04.html> [18 February 2021]

C# - How to Convert JSON to XML or XML to JSON? (n.d.) available from <https://stackoverflow.com/questions/814001/how-to-convert-json-to-xml-or-xml-to-json> [18 February 2021]

How to Edit an XML File with Node.Js (2020) available from <https://attacomsian.com/blog/nodjs-edit-xml-file> [18 February 2021]

Javascript - Reading XML File in Node.Js - Stack Overflow (n.d.) available from <https://stackoverflow.com/questions/32873100/reading-xml-file-in-node-js?fbclid=IwAR0hYWEtrTz_R6l1feFMz6XqrYtcP4LOfP540ExzO3453XeUsOVDNrEPliQ> [18 February 2021]

Web Programming II, Node.Js (n.d.) available from <http://dkexit.eu/webdev/site/apbs02.html?fbclid=IwAR31P5WGLQHxYh9ejO

M9Nkqf5ZtN_lJsyeUVGCe_bgc5DAYPlsAM8nNvpFc#assNode4M> [18 February 2021a]

Web Programming II, Node.Js (n.d.) available from <http://dkexit.eu/webdev/site/apbs02.html?fbclid=IwAR1U6HJgQY-IfmjHmUIjfayYdx-l7X9KXpYgzsFCRy_ybVODqhh-__XVTzw#assNode4M> [18 February 2021b]

Xml-Js (n.d.) available from <https://www.npmjs.com/package/xml-js> [18 February 2021]