Tværfagligt projekt uge 7
PBWEB Februar 2021

Nadia Skau
Morten Højrup Kristensen
Morten Due

Indledning

Vi har i undervisningen gennemgået Node.js og XML samt XSLT.
Vores projekt i uge 7 har til formål at bygge webapplikationer på server-siden, hvor vi tidligere har bygget client-side applikationer gennem browseren. Til dette har vi anvendt teknologier fra undervisningen, såsom Node.js.
Udover renderering af JavaScript, har vi arbejdet med håndtering og transformering af data i XML format.

Problemformulering

En bruger skal kunne indtaste en ny bog eller en ny forfatter via en formular i browseren.
Brugeren skal have mulighed for at opdatere information om bogen eller forfatteren
samt slette bogen/forfatteren.
Desuden skal brugeren kunne få fremvist en liste af bøger, hvor der er mulighed for sortering.
Dataen skal være gemt i et JSON format, der transformeres til XML vha. Native node.js.

Hvordan kan vi vedligeholde to XML-filer?
Hvordan får vi fremvist informationen overskueligt til brugeren med forskellige sorteringer?

Research

Vi har haft problemstillingen, at vores data skulle opbevares i XML og samtidig skulle der være mulighed for at redigere denne data. Det er muligt at indlæse XML data og skrive til en XML-fil, men det er ikke muligt at redigere denne data. Derfor har vi benyttet de to følgende artikler til at konvertere XML data til JSON data:
(*How to Convert XML to JSON in Node.Js* 2020)
(*How to Edit an XML File with Node.Js* 2020)

Vi har brugt følgende dokumentation til at researche options for xml2js-modulet, da vi havde problemer med skrivningen til XML-filen:
(*Xml2js* n.d.)
(*Package - Xml2js* n.d.)

Analyse og metodeovervejelser

Visning af data

I vores visning af 'books.xml'[1] indlæser vi XML-filen på server-side vha. modulet 'fs' (*File-System - Npm* n.d.). Herefter konverterer vi det til JSON vha 'xml2js' (*Xml2js* n.d.)
I første omgang har vi slet ikke tænkt på at transformere XML'en vha. XSLT, så derfor har vi brugt denne metode. Her har det været svært at få fat i alle elementerne, da objekterne har nestede objekter. Det havde været hurtigere at transformere det vha. XSLT.
I vores nuværende visning af books er sortering også besværlig. Her har vi et array af JSON-objekter, og det kræver en funktion til at gå ind og sortere på en specifik key. Her er XSL'en noget nemmere at bruge til sortering, hvilket vi gør brug af i vores visning af authors.

I vores view af 'author.xml' benytter vi netop XSLT til at transformere XML'en til HTML.
Her har vi blot sat vores router og server op til at håndtere XML- og XSL-filer.
Modulet 'xml2js' skaber dog problemer, når vi benytter denne til at skrive til XML'en: vi mister tagget, hvori vi referer til XSL-filen. Når dataen transformeres til JSON, mister vi headeren i XML'en. Builder() funktionen tilføjer standarden igen, men dette er altså uden tagget til stylesheetet (*Xml2js* n.d.).

I vores sortering af 'Authors' har vi brugt 2 XML- og XSL-filer, hvor hver XSL-fil sorterer forskelligt. Dette er ikke den mest optimale metode at gøre det på, dette vil skabe mange ekstra filer, hvis der skulle tilføjes nye sorterings måder. Her ville det være bedst at genbruge den originale XML-fil og tilknytte forskellige XSL-filer afhængig af ønsket layout/sortering.
Vi har dog ikke kunne finde en måde, hvorpå vi kunne løse det sådan, derfor har vi valgt at arbejde med flere sæt filer.

[1] Se fil 'printBooks.js'

Vi kunne have løst det ved at indlæse vores data client-side med AJaX og her brugt XSLT-processor til at importere og tilknytte forskellige XSL-filer.
Vi valgte dog at arbejde server-side, da dette er fokus på nuværende tidspunkt i undervisningen.

Redigering af data

Når man skal slette en bog[2], skal brugeren fremsøge på ISBN-nummer, da dette er unikt. Herefter indlæser vi XML-filen, konverter til JSON og fremsøger objektet med tilhørende ISBN-nummer. Derefter fjerner vi det fremsøgte objekt. Derefter konverter vi tilbage til XML vha 'xml2js' og skriver filen på ny. En udfordring, vi stødte på her, var som nævnt ovenfor, at 'xml2js' skriver filen uden reference til XSL'en. Denne ulempe går igen, når vi skal opdatere vores data.

Når vi opdaterer data for en bog, bruger vi samme metode som ovenfor med at fremsøge objektet vha. et ISBN-nummer. Her havde vi store udfordringer med async/await, idet vi skulle indlæse data, før vi kunne skrive dem i vores HTML-formular.
Vi bruger 'xml2js' til at konvertere formular-dataen over til et XML-format, og hvis et inputfelt er tomt, bliver det skrevet med følgende syntaks: "<element/>". Det kunne altså potentielt give en del tomme tags. Vi kunne afhjælpe det ved at tjekke på, om det er tomt og lade være med at sætte det ind i vores objekt.

Oprettelse af en bog og forfatter fungerer ens. Formular-dataen sættes ind i et objekt. Herefter indlæser vi vores XML-fil og konverterer dataen til JSON, hvor vi kan skubbe det nye objekt ind. Det transformeres tilbage til XML og filen skrives på ny.

MongoDB

Det var et krav i opgaven at opbevare dataen i XML, men det er ikke muligt at manipulere XML-data direkte. Det gør det nødvendigt at konvertere dataen frem og tilbage, hvilket vi kunne undvære ved brug af en Mongo Database.
Mongo Databasen har nemlig dataen i et JSON format, så det kan vi trække direkte ud og manipulere og sende tilbage i samme format uden konvertering. På denne måde er man fri for, at ens data bliver konverteret til et uønsket format eller der er datatab.

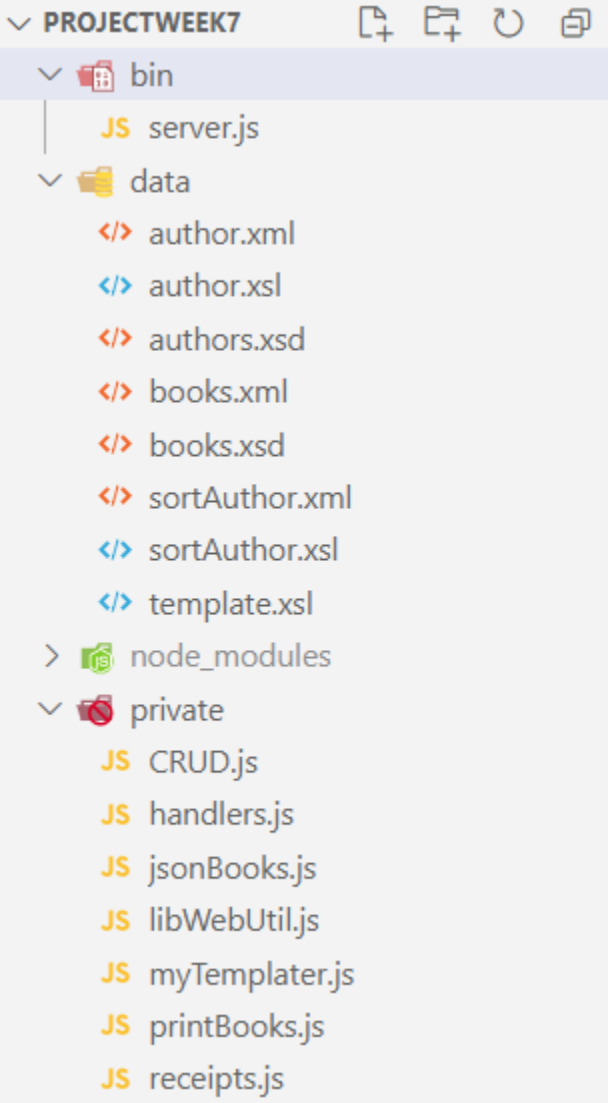
XML er dog utrolig nemt at transformere til HTML vha. XSL, og her sparer man altså noget tid og kodelinjer fremfor at opbygge HTML-viewet manuelt fra et JSON-objekt med JavaScript.

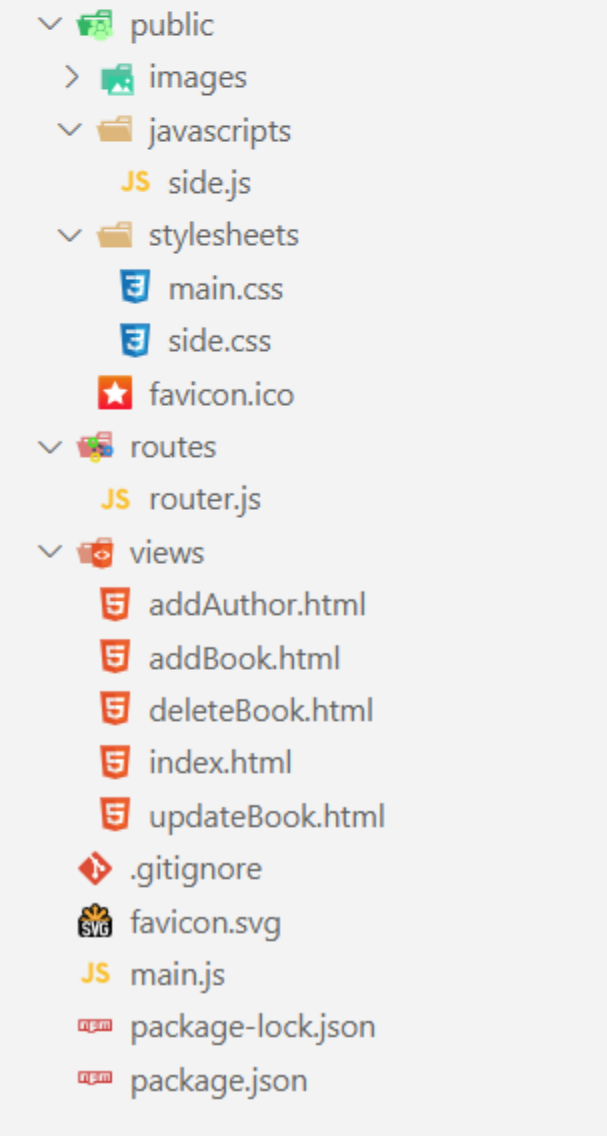
[2] Se fil 'CRUD.js'

Konstruktion

Vi har samlet vores XML, XSL og XSD-filer i datamappen. Alle vores scripts er samlet i private-mappen pånær vores 'main.js', der starter vores server og router.
Serveren ligger i bin-mappen, og routeren ligger i routes-mappen. Disse holdes adskilt fra de andre scriptfiler for at give et bedre overblik, og sikre at filerne udelukkende har et formål.
I vores public-mappe har vi vores css stylesheets og eventuelle billeder. I views-mappen har vi vores HTML-sider.

Vores handlers kalder andre moduler, hvis der skal ske mere en blot indlæsning af en statisk fil. Så undgår vi at handler-filen bliver for lang, og man har muligheden for at redigere i disse moduler uden at ødelægge handler-filen.





Evaluering af proces

Vi har valgt at arbejde synkront, da vi alle gerne vil have mest mulig læring ud af vores projekter. Det er naturligvis tidskrævende, men det giver et godt udbytte læringsmæssigt.
Det betyder dog, at vores applikation ikke har fået den store styling, da vi har lagt kræfterne i kodningen.

Vi har ikke fået planlagt, hvordan vi ville gribe det til værks. Vi startede blot med visningen af books uden at tænke over, hvordan vi ville gøre det. Vi brugte 'xml2js', fordi det stod i vores opgavebeskrivelse - og vi nærlæste ikke, at det var til manipuleringen af data.
Set i bakspejlet ville vi have valgt at opbygge visningen vha. XSL transformering, da det er nemmere. Vores fremgangsmåde er meget at prøve os frem, og til tider bør vi stoppe op og tænke over de valg, vi træffer.

Vi har brugt GitHub som KanBan-board (*Langehk/Projectweek7* n.d.), og her kunne vi godt have været bedre til at kigge på det. I et større projekt ville der klart være behov for mere planlægning og struktur i forhold til arbejdsopgaver.

Konklusion

Vi har udviklet en webapplikation, hvor brugeren har mulighed for at se listen af bøger og forfattere. Man har også mulighed for at opdatere information omkring en bog eller slette den ved at fremsøge den på ISBN-nummer. Der er mulighed for at oprette både en bog og en forfatter.

Vores visninger er opbygget på to forskellige måder, da vi i udviklingsforløbet fandt en bedre metode. Derfor er bøger fremvist vha. JSON og JavaScript, hvor forfattere er fremvist vha XSLT. Det er en del nemmere at transformere XML vha. XSLT fremfor at konvertere XML til JSON og så opbygge visningen i JavaScript. Det gav os nogle udfordringer, hvor visningen af forfattere gik en del nemmere.

Både når man opdaterer, sletter eller opretter en bog, indlæser applikationen XML-filen og opdaterer dennes indhold og skriver filen på ny. Således bliver den vedligeholdt uanset om man sletter, opretter eller opdaterer information.
Når man opretter en ny forfatter fungerer det på samme måde som ovenfor, og denne bliver derfor også vedligeholdt. Her har vi to XML- og XSL-filer, hvor vi har sørget for, der bliver skrevet til begge XML-filer, således dataen bliver vedligeholdt begge steder.

Vores arbejdsproces har bestemt været produktiv, men i næste projekt vil vi planlægge og tænke noget mere, inden vi begynder at kode vores løsning, således vi får det kodet hensigtsmæssigt.
Hvis vi havde haft mere tid, ville vi have forsøgt at finde en mere hensigtsmæssig måde at sortere på forskellige måder med XSLT.

Referencer

File-System - Npm (n.d.) available from <https://www.npmjs.com/package/file-system> [17 February 2021]
How to Convert XML to JSON in Node.Js (2020) available from <https://attacomsian.com/blog/nodejs-convert-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]
Langehk/Projectweek7 (n.d.) available from <https://github.com/langehk/projectweek7> [17 February 2021]
Package - Xml2js (n.d.) available from <https://developer.aliyun.com/mirror/npm/package/xml2js> [18 February 2021]
Xml2js (n.d.) available from <https://www.npmjs.com/package/xml2js> [17 February 2021]

Repository: https://github.com/langehk/projectweek7.git