Christoffer Kalmar, Jon Andersen, Erik Dam, Nickolaj Sørensen

Javascript & Frameworks - Data Integration
2. Semester
Første projekt

Indholdsfortegnelse

Indledning

Rapporten vedrører brugen af Node.js og XML, samt samspillet derimellem. Rapporten kommer først ind på hvilke metodeovervejelser der er gjort i forbindelse med udarbejdelse af projektet. Derudover beskrives også hvordan vi har lavet research til projektet, samt brugen deraf. Der vil i rapporten også blive analyseret på specielt det at bruge node.js og xml - var der f.eks noget vi kunne have gjort anderledes og hvorfor? Og hvorfor har vi gjort, som vi har gjort? Konstruktionen af koden vil også blive berørt. Der vil også blive evalueret på processen - har blandt andet arbejdsgangene været optimale? Kunne vi have gjort det bedre? Hvis ja, hvordan? Slutteligt vil der blive konkluderet på projektet som helhed.

Problemformulering

Hvordan kan vi sammenkoble Node.js med et XML dokument sådanne at vi både kan vise, redigere og gemme indhold i XML dokumentet på et live site.
Delopgaver:
- Hvordan får vi applikationen til at modtage og opdatere data.
- Hvordan får vi data til at blive vist i mere end en sortering.

Metodeovervejelser

sammen med Node.js sådanne at vi både kunne validere samt læse og opdatere data.

Yderligere skulle sitets forside indeholde menupunkter der skulle indeholde både funktionalitet til fremvisning og vedligeholdelse samt oprettelse og redigering af data.

Det var her vores to delopgaver træder i kraft, og vores virkelige opgaver kommer i spil:
1: Hvordan får vi applikationen til at modtage og opdatere data.
2: Hvordan får vi data til at blive vist i mere end en sortering.

Vi gik derfor igang via Visual studio tilføjelsen Live Share, så vi alle kunne oprette dokumenterne sammen, så vi alle havde et overblik, og kunne kode sammen. Vi observerede at man skal være påpasselig når man arbejder i Live Share. Hvis man lukker filmappen man arbejder i, som host, smider man samtlige deltagere ud af samarbejdet og skal genstarte forbindelsen. Dog fordelen i at kunne bearbejde et projekt i samme live fil-mappe og gennem terminalen i Visual studio, gjorde at vi kunne spare væsentlig tid i deling af projektet. Frem for at skulle uploade, downloade og gennemse ændringer individuelt, kunne vi nu arbejde i projektet med samme øjne, hvilket var til stor fordel.

Research

Meget af researchen - specielt i forbindelse med XML, har været fra det tilgængelige undervisningsmateriale. Altså undervisningsmaterialet fra dkexit. Derudover har vi spurgt Niels om hjælp via emails. udover det har Christoffer via en bekendt back-end udvikler som han kender fra en gaming server spurgt om hjælp i håb om at kunne være til assistance med det vi sad fast med, han gav det et godt forsøg men da han var mere back-end, så var XSL og XML ikke ligefrem hans aller stærkeste side i form af kodning. Så vi kommer altså ikke videre efter vi havde siddet med det i et godt stykke tid. Så var det Ellers tilbage til at prøve at søge frem til et svar, og her har vi så benyttet os af w3schools og Mozilla Developer Network.

Analyse

XML og XSL

XML simplificerer ting, såsom data-deling, data-transport, data -tilgængelighed.
For at kunne bruge XSL skal du kunne have grundlæggende forståelse for HTML og XML.
Hvor HTML har CSS, så har XML, XSL.
Og dette tilfælde kan vi se det på vores Sorting(bilag 1), da vi tager data fra vores XML dokument som har et parent element (Catalog) og inde i det har vi et child element (cd) og inde i den cd har vi så nogle forskellige elementer, alt fra title til year.
Og nu hvor vi har en XML dokument med data, kan vi så oprette et XSL dokument hvor vi så kan få tilføjet noget styling og noget sortering(bilag 2). Her får vi lavet noget styling i toppen hvor vi giver den en h2, og derefter får vi lavet et table hvor vi kan gemme vores data, vi får so lavet nok felter til det antal elementer vi skal bruge.
Når det er gjort så laver vi så vores for each, og det den gør er at den går ind i vores catalog og cd for at finde de elementer som vi kan sortere ud fra i dette tilfælde så bruger vi “price”, den vil så automatisk gå ind og se hvilket pris som er højest siden vi ikke har defineret at det skal være ascending eller descending hvilket betyder at descending så tager man det højeste tal først og går nedad, og ascending er så det modsatte.
Efter det så tildeler vi resten af elementerne en værdi.

Node.js

module.exports (*bilag 3*)
Clienten sender nogle request til serveren om hvilke filer der skal hentes og vises.
Dette gøres via router.js hvor den så sender klienten igennem handlers.js hvor den sender et svar tilbage igennem module.exports ifølge de ønskede requests. (request

and respond).
Denne respons kan så være i form af billeder, tekst, filer osv.
xml2js.module er et modul der gør det muligt at parse mellem XML og JSON for at man kan konvertere data mellem JSON og XML.

server (bilag 4)
starter med at definere modulet så det kan exporteres, dernæst kører den router fra router.js med servercall'et i. Herefter laver den et body array og pusher det givne data til body'en, som nu kan blive display'et. slutterligt laver den en server.listen, som logger hvor serveren er startet. i dette tilfælde localhost:3000

Konstruktion

Principielt så er hele konstruktionen fiktionel, da vi ikke kunne sammensætte vores dokumenter. Grunden var (*se bilag 5*) at vi ikke kunne sammenkoble vores xsl stylesheet til vores xml.
Først og fremmest måtte vi starte vores arkitektur: et HTML dokument som forside, med links til undersider. Nogle javascript moduler samt de opgivne *books.xml* og *authors.xml*. Disse skulle så indeholde vores grund informationer som i denne sammenhæng skulle være indhold(værdi) af bøger, og dertil diverse forfattere. Disse to informationskilder skulle så kunne kobles sammen og vises. Men vi valgte i første omgang valgte vi at kreere et samlet database dokument i XML format med et catalog parrent med cd elementer hvori vi definerede title, artist, country, company, price, year.
Dette xml dokument skulle så læses i en funktion og parses til en JSON string hvor vi kunne manipulere ved at ændre, fjerne eller tilføje både elementer og objekter, før vi brugte xml2js.modul til at konvertere JSON string tilbage til XML fil i vores databases.xml.

Her ville vi så kombinere denne funktion med vores JavaScript form, for både at kunne fremvise, ændre og slette objekterne. Samt senere tilføje en funktion der kunne oprette et cd objekt. Senere var idéen at ændre filerne så vi havde en fil til artister og udgivne cd'er.

Evaluering af proces

Vi startede ud med at lægge en plan for ugen ved hjælp af trello (bilag 6). Det gik dog relativt hurtigt op for os, at vi allerede om tirsdagen var bagud med opgaverne, som skulle have været lavet om mandagen. Sådan fortsatte det ellers ugen ud, og vi endte derfor med at miste overblikket over hvad der skulle laves - og specielt hvordan det skulle laves. Det lægger sig nok også til, at vi ikke tidligere har lavet projekter, som omhandler netop XML og node. Det var derfor svært for os, at tidsestimere på projektet som helhed. Med andre ord fandt vi det svært, at gå til en konkret task der skulle løses, og vurdere hvor lang tid den tager. Dette er givetvis også vist i form at både rapporten og produktet. Begge dele kunne være mere gennemarbejdet, men på grund af tidspresset kom vi ikke helt i mål med det.

Projektet blev derfor hurtigt så komplekst for os, at vi efterhånden sad med forskellige mapper for at teste koden af - på trods af at vi jo, som tidligere nævnt, havde muligheden for at sidde med den samme kode og programmere det sammen. Dette gjorde, at da vi skulle forsøge at kombinere de forskellige mapper og/eller filer, blev det (selvfølgelig) meget mere komplekst end det allerede var i forvejen, hvilket nok også var medvirkende til, at vi ikke kom i mål med projektet. Vi havde altså mange af de funktioner der skulle bruges, men var ikke i stand til at kombinere dem under en samlet server ved hjælp af node.js.

Derudover har hele corona-situationen nok også besværliggjort processen, i og med det uden tvivl er mere effektivt at sidde fysisk med personer, som man laver projekter og skal skrive kode sammen med. Man undgår i højere misforståelser og situationer hvor folk ikke ved, hvad de skal gå i gang med og hvordan.

Hvad kan så gøres anderledes til næste gang? Store dele af processen, er svaret nok desværre.

Overblik

Vi skal først og fremmest være hurtigere og bedre til at få skabt et overblik over,

hvad det egentlig kræver at løse opgaven. Både ift. tid, men også hvad der faktisk skal laves. Det skulle gerne gøre, at vi så vidt muligt kan lave de forskellige tasks i den mest optimale rækkefølge. Vi var nemlig aldrig helt klar over, hvad der skulle programmeres og hvorfor. Vi sprang direkte på koden, uden at have det fulde overblik over, hvad der skulle laves udover den fil vi sad og arbejdede i. Det medførte, at vi mange gange skulle rette rigtig meget i koden efterfølgende, da der måske ikke var tænkt nok over, hvad koden skulle bruges til. Dette gør netop, at vi kommer til at tage ét skridt frem og mindst ét tilbage, hvilket gjorde, at processen i stedet for at være en lige linje frem til målet, blev ad utallige omveje som endte et andet sted end det skulle.

Starte på rapporten tidligere

Vi kom givetvis for sent i gang med rapporten. Havde vi startet på rapporten tidligere, gerne mandag, og dermed også tvunget os selv til at beskrive de ting vi lavede mens vi lavede dem, havde forståelsen for projektet også været højere. Som tidligere nævnt kom vi flere gange til at lave noget, hvor vi ikke til fulde forstod hvorfor det skulle laves, og var derfor ikke i stand til at lave det.

Konklusion

Der er nu redegjort for hvilke metoder der er benyttet og hvordan vi har researchet. Derudover har rapporten berørt hvordan vi har benyttet xml og node.js, samt konstruktionen deraf. Slutteligt er der også evalueret på processen.

På trods af at vi stødte ind i utallige udfordringer, fik vi trods alt lavet noget brugbart. Vi kom desværre ikke i mål med det vi ville, altså det, som projektbeskrivelsen lød på. Dog føler vi, at vi har fået nogle redskaber, som vi kan tage med os til fremtidige projekter og/eller opgaver. Her tænkes specielt på processen, der som tidligere nævnt ikke har været optimal. Derudover har vi også fået arbejdet mere med node.js og xml/xsl, som har givet os en bedre forståelse for de forskellige redskaber i forhold til da vi startede på projektet. Vi fik ikke løst opgaven til fulde, men i og med at vi har lavet nogle fejl, så lærer vi af dem til næste projekt og gør det anderledes, og bedre, end denne gang.

Referencer

http://dkexit.eu/webdev/site/ch42s03.html
http://dkexit.eu/webdev/site/ch42s04.html
https://www.w3schools.com/xml/xsl_sort.asp
http://dkexit.eu/webdev/site/ch15s03.html
http://dkexit.eu/webdev/site/ch16.html
http://dkexit.eu/webdev/site/ch17.html
https://attacomsian.com/blog/nodjs-edit-xml-file

Bilag 1

```
<cd>
  <title>Appetite for destruction</title>
  <artist>Guns n Roses</artist>
  <country>USA</country>
  <company>Geffen Records</company>
  <price>9.90</price>
  <year>1987</year>
</cd>
```

Bilag 2

```
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>title</th>
        <th>artist</th>
        <th>country</th>
        <th>company</th>
        <th>price</th>
        <th>year</th>

      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:sort select="price"/>
    <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
        <td><xsl:value-of select="country"/></td>
        <td><xsl:value-of select="company"/></td>
        <td><xsl:value-of select="price"/></td>
        <td><xsl:value-of select="year"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Bilag 3

```
module.exports = {
    home(req, res) {
        let path = req.url;
        if (path === "/" || path === "/start") {
            path = "/index";
        }
        path = "views" + path + ".html";
        let content = "text/html; charset=utf-8";
        getAndServe(res, path, content);
    },

    bookData(req, res) {
        readWrite.read("public/xml/booksCanon.xml", res)
        readWrite.read("public/xml/browsere2.xsl", res)
    },

    js(req, res) {
        let path = "public/javascripts" + req.url;
        let content = "application/javascript; charset=utf-8";
        getAndServe(res, path, content);
    },
    css(req, res) {
        let path = "public/stylesheets" + req.url;
        let content = "text/css; charset=utf-8";
        getAndServe(res, path, content);
    },
    png(req, res) {
        let path = "public/images" + req.url;
        let content = "image/png";
        getAndServe(res, path, content);
    },
    ico(req, res) {
        let path = "public" + req.url;
        let content = "image/x-icon";
        getAndServe(res, path, content);
    },

    notfound(req, res) {
        console.log(`Handler 'notfound' was called for route ${req.url}`);
        res.end();
    },
```

Bilag 4

```
"use strict";
/*
 *  new server.js adds request body data
 */
const http = require("http");                   // http module
const lib = require("../private/libWebUtil");   // home grown utilities
const hostname = "localhost";
const port = Number(process.argv[2]) || 3000;

module.exports = {
    start(router) {
        const server = http.createServer();

        server.on("request", function (req, res) {      // eventhandler for "request"
            console.log(lib.makeLogEntry(req));         // home made utility for logging
            let body = [];
            req.on("data", function (bodyData) {        // eventhandling for data reception
                body.push(bodyData);                    // bodyData is an object
            });
            req.on("end", function () {                 // eventhandling for end-of-data
                body = Buffer.concat(body).toString();  // body2string
                router.route(req, res, body);           // pass to router
            });
        });

        server.listen(port, hostname, function () {
            console.log(`Log: Server started on http://${hostname}:${port}/`);
        });
    }
}
```

Bilag 5

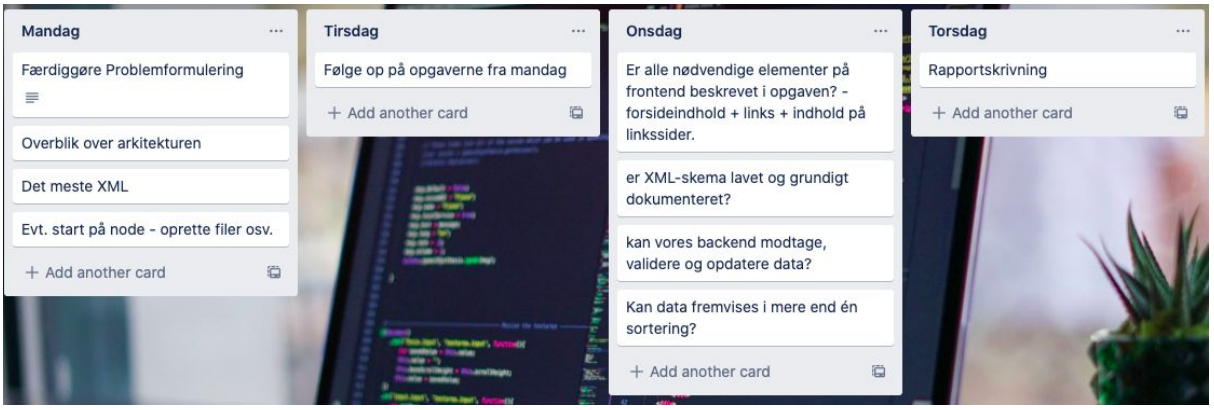
Observering:

```
result.catalog.cd[2].name = '<?xml-stylesheet type="text/xsl"
href="browsere2.xsl"?>';
```

Ovenstående "hjemmelavede" funktion skulle hjælpe os med at kombinere et stylesheet til en xml fil til nedenstående funktion. Til stor forundring kan vi observere at < og > bliver konverteret til `<` og `>`. Dette sker med stor forundring hos os grundet at det er i en string, og denne string egentlig bør blive udskrevet som den er.

```
<name>&lt;?xml-stylesheet type="text/xsl" href="browsere2.xsl"?&gt;</name>
```

Bilag 6



bilag 1, trello, (trello.com)