Föreläsning 4

Express views

NORTH LINK

Express views

Express vyer är ett hjälpmedel vi kan använda för att dynamiskt bygga HTML på backenden. Express vyer använder sig av någon form av template system. Vi kommer att testa pug.

Setup pug

För att använda pug template motorn behöver vi installera biblioteket samt berätta för express att vi vill använda det.

```
> npm install pug
```

```
app.set('view engine', 'pug');
```

View folder

För att använda vyer behöver vi en views mapp.

Vi berättar för express var vår mapp med vyer ligger

```
app.set('views', './views');
```

I denna mappen skapar vi `.pug` filer. Namnet vi ger dem blir namnet på vyn.

Använda en vy

Att svara på en request med en vy är väldigt enkelt.

```
app.get('/', function(req, res) {
    res.render('index');
});
```

Denna koden kollar om det finns en index.pug fil i views mappen (för vi har konfigurerat det så). Om den finns så kommer den att applicera template motorn på filen och svara till klienten med resultatet.

Kort intro till pug

Pug är en fristående template motor (vi behöver inte använda det med express), som ofta används tillsammans med express.js

Dokumentationen finns på https://pugjs.org/api/getting-started.html

Språket ser väldigt snarlikt ut till HTML (det är HTML vi får när vi har kompilerat). Men språket låter oss skriva enkel kod för att jobba med HTML, t.ex. conditionals och loopar.

Parameters

Det går att skicka parametrar från node till pug genom att ange ett objekt som andra argument i `render` metoden.

```
app.get('/', (req, res) => {
    res.render('index', {title: 'hello'});
});
```

Nycklarna kommer finnas som globala variabler

```
h1= title
```

Conditionals

Conditionals ser i princip likadant i pug som i JS. Notera att vi inte har brackets i pug utan istället skapar man block med indentering.

```
body
    if loading
        p
            Loading
    else
        p
            Done
```

Loopar

I pug har vi `each` och `while`. Each fungerar snarlikt till for...of

```
ul
    each val in [‘banana’, ‘apple’, ‘pear’]
        li= val
```

Notera att likhetstecknet innebär att p kommer innehålla värdet på variablen val

Includes

Det går att inkludera in en annan pug fil. Det är användbart om man t.ex. har en gemensam header mellan flera sidor.

// header.pug

```
h1 Hello
```

// index.pug

```
header

    include header.pug
```

Mer info

https://pugjs.org/api/getting-started.html

Jag rekommenderar att ni tittar igenom hela dokumentationen (läs igenom alla avsnittet som finns i högermenyn under language reference)

Övning - lista med frukter

Skapa ett enkelt REST API där vi kan lägga till frukter.

Skapa därefter en route som listar alla skapade frukter med hjälp av en pug template.

Lägg gärna på pagination

Formulär

Man kan skapa HTTP requests även med HTML formulär.

```
<form method=”POST” href=”/login”>
    <input type=”text” name=”email” />
    <input type=”password” name=”password” />
</form>
```

Formulär

- name nyckeln är den nyckeln vi kommer ha datan på i backenden
- formulär data skickas som `application/x-www-form-urlencoded`
- Med `express.urlencoded` middlewaren kan vi enkelt lägga till formulär parsing

Övning - Frukt formulär

På API:et du precis gjorde, lägg till en route som renderer ett formulär.

När man submittar formuläret ska en resurs skapas på servern - man ska redirectas till listvyn och ska kunna se den nya resursen.

Det ska finnas länkar mellan de två vyerna.

Lös allting med vyer och formulär, använd ingen JSON, AJAX eller frontend kod.

Static middlewaren

Det finns en inbyggd middleware `express.static` i express som hjälper oss att leverera statiska resurser. Saker som bilder, CSS, osv.

Ofta har man en mapp `public` som innehåller alla statiska filer (namnet public för denna mappen bli öppen för alla).

Därefter man kan skriva

```
app.use(express.static('public', options));
```

https://expressjs.com/en/4x/api.html#express.static

Övning - Static

Skapa en public mapp, lägg till några bilder och CSS som du sedan inkluderar i din listvy och formulärvy.