All things u wrote as ovning 2 is ovningar for lek 3 not lek 2

**Laborationer - Express & Template Engines**

* [Express.js | Using Template Engines](https://expressjs.com/en/guide/using-template-engines.html)
* [PugJS.org](https://pugjs.org/api/getting-started.html)

1. Skapa en express-server som har en route / som levererar en index.html via sendFile. Servern kan lyssna på valfri port. Starta servern med nodemon.
2. Från din index.html, lägg till så att du har en main.css-fil länkad via index.html där du stylar din sida. Du ska även på sidan visa upp en valfri bild som du serverar till din server lokalt. Här behöver du alltså använda static. [Serving static files in express](https://expressjs.com/en/starter/static-files.html)
3. Utveckla din express-server genom att ta emot parametrar i requesten (exempelvis userID) och visa upp olika resultat beroende på parameter. Hämta gärna din data från en json-fil, och nöj dig med att skicka svaret som ett json-objekt, då med metoden res.json.  
   Kika mer på <https://webapplog.com/url-parameters-and-routing-in-express-js/>
4. Gör ungefär samma sak, fast denna gång ska du inte ta emot en parameter, utan en query.
5. Installera modulen pug med npm install --save pug. Skapa en index.pug som du lägger i mappen views. Sätt view engine till pug och rendera ut din index med res.render. Innehållet i index.pug ska vara samma som din index.html från övning 1 & 2.
6. Dela upp din index.pug i två olika delar. En layout.pug som är din standardlayout för samtliga sidor. Använd sedan extend för att förlänga denna layout till index.pug.

views

layout.pug

index.pug

1. Skapa sedan två nya routes förutom /. Den ena ska gå till en "om oss"-sida och den andra ska gå till en kontaktsida. Samtliga sidor ska förlängas från layout.pug. Försök att återanvända så mycket kod som möjligt.

views

layout.pug

index.pug

about.pug

contact.pug

1. Skicka med extra information till varje view via res.render() som byter ut innehållet baserat på vilken view man går till. Du måste alltså använda variabler i dina pug-filer.
2. Bygg ut din server så att du kan ta emot någon form av information via parametrar och querys (som i övning 3 och 4).
3. Utöka övning 8 genom att servern serverar två (eller fler) sidor, och se till att logga info på alla, men lägg också två separata middleware-funktioner för varje sida. Observera i vilken ordning du “kedjar” funktionerna till sidorna.
4. Använd länken:<https://yhjust16.herokuapp.com/nobel> för att hämta den informationen i din app.js och spara hela arrayen som en varibel i app.js, logga ut (console.log) variabeln för att se om du fick ner informationen. Du kan här som tidigare använda https.get() eller testa använda paketet [request](https://github.com/request/request) som har samma funktionalitet men mer utökat.
5. Skicka med variabeln från föregående övning till index.pug och skriv ut varje pristagares namn i en lista. Här måste du använda: [Pug.js: iteration](https://pugjs.org/language/iteration.html).
6. Skapa en express-server där du skriver egen middleware för att logga vem som besöker sidan. Använd request-objektet för att ta reda på det du vill ha.   
   <http://expressjs.com/en/4x/api.html#req>
7. Använd samma information som från övning 11. Men nu ska du skriva ut en lista som bara visar pristagare som är från **USA**. Sorteringen ska ske i pug med [conditionals](https://pugjs.org/language/conditionals.html)
8. Återskapa samma layout och funktionalitet som tidigare fast använd en annan **Template Engine**. Få det att t.ex. funka med [Handlebars.js](http://handlebarsjs.com/). **Använd någon av dessa paket till Express för att köra handlebars:** [**hbs**](https://github.com/pillarjs/hbs), [handlebars-express](https://github.com/ericf/express-handlebars)