Lappeenrannan teknillinen yliopisto
School of Business and Management

Software Development Skills

Marta Xiaoyang Moraga Hernández, 001101267

LEARNING DIARY, FULL-STACK 2022-23 MODULE

24/12/2022

I accessed the course and read the necessary information. As it is quite similar to the Front-End course, I started setting up everything. I decided to work on Windows using Visual Studio Code.

Creating the repository was quite simple, because it was the same process as last time for the other course.

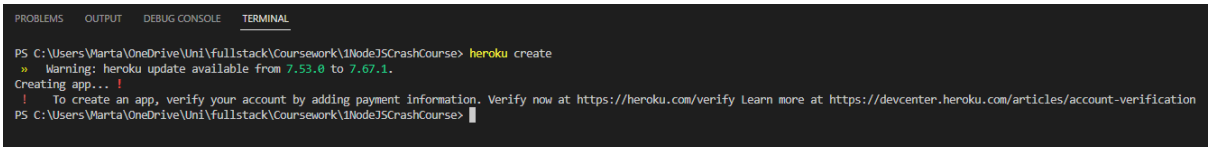
Node.js Crash Course

The start was quite simple to follow, as the Advanced Web Applications course also uses Node.js.

26/12/2022

The tutorial is easy to follow, though I ran into some issues due to spelling. The short rundown of the modules was quite useful. Learning about event, fs, http, os, path and url modules made me understand why a certain module would be required in different situations. Moreover, working with Express will be much easier after learning more about Node.js.

As it is necessary to add a payment method to Heroku, I did not deploy my application. However, I followed every step and noted down how the process would be if I used Heroku.



MongoDB Crash Course

To download MongoDB and use the shell, MongoDB Community Server https://www.mongodb.com/try/download/community and MongoDB Shell https://www.mongodb.com/try/download/shell are necessary.

When I was watching the tutorial and had downloaded MongoDB Community Server, I was not able to find mongo.exe on my computer. After looking up the solution online, I followed the first answer to this stackoverflow question https://stackoverflow.com/questions/73081708/mongo-exe-not-installed-in-version-6-0-0, which solved the issue.

After trying to insert an object the warning: *DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite* appeared. I will use those commands in the future.

I also learnt that I can start or stop the MongoDB service throughout the Services console.

28/12/2022

It seems that *db.posts.find()* and *db.posts.find().pretty()* show similar results in the current version of mongosh.

When trying to update the message *MongoInvalidArgumentError: Update document requires atomic operator appears*. It seems that the functionality described in the tutorial has changed. Update can target individual properties of the DB document, while replace does change everything as the tutorial stated. Update cannot be used without an atomic operator like $set or $inc.

After trying to insert an object the warning: *DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite* appeared. Similar to the insert method. I will use those commands in the future.

I ran into some issues connecting to the cluster as PowerShell did not recognize mongosh as a command. I used this tutorial https://databasefaqs.com/mongodb-is-not-recognized-as-an-internal-or-external-command/ to solve them. It seems I had forgotten to add the path.

I was using MongoBD from its shell application Mongosh by running mongosh.exe, however, it could also be run from the PowerShell. That may be the reason for some differences between what I was doing and the tutorial.

Finally, using MongDBCompass, I exported the collection 'posts' as a json to the Coursework folder. I also added a screenshot of the collection 'todos' that was on the cluster.

Express JS Crash Course

I was already familiar with Express, as it is necessary in the Advances Web Applications course. This tutorial deepened my understanding. After learning about Node.js, Express becomes more understandable. In Express, there is less to handle but now I understand the logic behind it.

The tutorial taught how to create get, post, put and delete methods to check, update and delete information, among other things like the basis of Express. I found middleware creation especially interesting.

When using the Handlebars Middleware I ran into a small issue. I copied the specified code from the Handlebars github as the tutorial stated::

```
app.engine('handlebars', exphbs({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');
```

Which resulted in *TypeError: exphbs is not a function*. I solved this by reading this thread https://stackoverflow.com/questions/69959820/typeerror-exphbs-is-not-a-function. I just had to add *.engine*.

Working with Handlebars and Bootstrap was a good introduction to views.

Angular: Tour of Heroes application and tutorial

Setting up the project went smoothly, except I did not expect a new folder to be created so I ended up with a couple redundant folders that had to be deleted. The Angular application is much heavier than what we were working with when using Node.js or Express.

The serve command has all the necessary functions to work on the application. It reminds me of the nodemon and tsc commands combined, but much more complex. I learnt about the app.component.html, app.components.ts and styles.css files. It felt similar to working with views in express.

Directives and bindings are very powerful tools.

29/12/2022

App-routing.module.ts was already generated, since Angular asked when the application was generated.

I was already familiar with Model–view–controller (MVC) due to object oriented programming, therefore Angular workflow did not surprise me too much.

Following the tutorial was simple as it shows the necessary code and it states where to paste it. However, understanding is a different story as there are a lot of concepts to learn. It was especially hard for me to set up the 'pseudo server' that let the app simulate fetching data from a server.

My favourite part was how angular handles the views. Everything is very modular, which is ideal to show different things that do not depend on each other.

30/12/2022

Angular: Building a template-driven form

I generated the app and the hero-form component with the commands from the last tutorial. I was able to recognize some expressions from the last tutorial, like binding with [(ngModel)] or the directive *ngFor.

I had some doubts where to put some of the code, for instance, where *const myHero* (an example of the hero class) went. I assumed it was just an example and left it for later. It truly was an example.

As the html was explained with excerpts and not the full code, my hero-form.component.html ended up slightly different from the example. For example, the main form and div that shows what you submitted are in different divs, the div with class="container" and the one with [hidden]="submitted" are the same one, and I forgot to add a warning message for "Power is required."

The end result looks the same and this is not an html focussed course, but I changed it anyways. I deleted power.pristine, because the warning will never show otherwise. Since you cannot select an invalid option, it will either be pristine or valid.

[hidden]="submitted" reminded me of the Front-end Course, where JS was used to add “show” to the classes and css to hide/show said classes. I suppose the logic is similar, but the syntax is quite different.

Things like (click)="newHero()” makes linking a method to an element quicker, as it erases the need to assign a document element to a variable.