Node Labs

Over the course of the next two labs, you will be in charge of creating a Node server that will communicate with our front-end we had built during the Angular lab. This will be a precursor to a full-stack application.

The back-end will be built in 3 stages:

1. Creating the server
2. Setting up the server's routing
3. Connecting to a database

Node Lab Part One

Task: Construct a Node server that will be able to send information from a module on your back-end to your front-end.

What does the application do?

1. When the user navigates to localhost:3000, it will display a random item from a list of tasks they must complete.

Build Specifications:

1. The server file must require an external module that you have created.
2. The external module will consist of an array of tasks that you need to complete, which must be exported.
3. Has to send a random item from the list of tasks to the front-end.

Node Lab Part Two

Task: Convert the Node server to an Express server. Create a module that contains routes for your front-end to communicate with. Test the endpoints with Postman. Once the endpoints are working with Postman, write the HTTP requests within your service(front-end) to handle these communications.

The front-end of this application is going to be your Angular lab!

What does the application do?

1. The back-end will now have routes for GET, POST, PUT, and DELETE which allows our front-end to communicate with our server. Each route will be handling the following functionality:
    a. GET: retrieves rows of tasks from the table within your database
    b. POST: adds a new task to the table within your database
    c. PUT: allows us to update a task from the table within your database.
    d. DELETE: deletes a task from the table within your database.
2. The front-end will now have HTTP requests for GET, POST, PUT, and DELETE.

Build Specifications:

1. Use Express to create your server.
2. Create a public folder that will house your front-end files. Adjust the server.js file accordingly to ensure Express is going to be using the public folder.
3. Require the module that will contain the routes you have created.
4. Test your endpoints using Postman.
5. Write four HTTP requests in your Angular service that will handle making a GET, POST, PUT, and DELETE request.
6. Upon successful request, log a message to the console.
    a. Example: GET request will log “GET request successful” to both your terminal and your browser’s console.
    b. Example: DELETE request will log “DELETE request successful” to both your terminal and your browser’s console.

SQL LAB

TASK:
Practice writing SQL statements on the Northwind database.

SETUP:

- In pgAdmin3, create a database called northwind.
- Open up a SQL window. Copy-paste and run this file... https://raw.githubusercontent.com/pthom/northwind_psql/master/northwind.sql

DETAILS:
Write SQL queries to do the following tasks. Record these queries in a text document so you can repeat them in class.

1. Select all the records from the "Customers" table.
2. Get distinct countries from the Customers table.
3. Get all the records from the table Customers where the Customer's ID starts with "BL".
4. Get the first 100 records of the orders table.
5. Get all customers that live in the postal codes 1010, 3012, 12209, and 05023.
6. Get all orders where the ShipRegion is not equal to NULL.
7. Get all customers ordered by the country, then by the city.
8. Add a new customer to the customers table. You can use whatever values/
9. Update all ShipRegion to the value 'EuroZone' in the Orders table, where the ShipCountry is equal to France.
10. Delete all orders from `order_details` that have quantity of 1.
11. Calculate the average, max, and min of the quantity at the `order details` table.
12. Calculate the average, max, and min of the quantity at the `order details` table, grouped by the orderid.
13. Find the CustomerID that placed order 10290 (orders table)

BONUS:

14. Do an inner join, left join, right join on orders and customers tables.
15. Get employees' firstname for all employees who report to no one.
16. Get employees' firstname for all employees who report to Andrew.

Node Lab Part Three

Task: Create a database for your Angular TODO list. Connect the front-end to the back-end.

What does the application do?

1. The application will now have a consistent database to hold and retrieve information from, allowing the user to keep their list of tasks.

Build Specifications:

1. In pgAdmin, create a database called “TodoDB” that has a table called “Todos”.
2. Construct the pg-connection-pool.js file that will contain all of the information allowing the server to communicate with the database.
3. Adjust your GET, POST, PUT, and DELETE requests in your routes module to include the appropriate queries for each of the four requests.
4. Test your endpoints with Postman to make sure the routing is set up.