Lab 4: Cached Blog Static

Due Date: Before Lab on Week 6

Overview

In this lab, we'll be creating a fake (or real) blog. Writing is one of the most important ways to get your name out into the world, whether it's to anyone with a browser or your colleagues at work. It's also one of the best parts of the web that your generation missed along with Neopets and Runescape.

Objectives

By the end of this lab, you should be able to

1. Write Promise based code to interact with the filesystem
2. Send files in response to HTTP Requests
3. Write CSS to make your blog look pretty
4. Implement in-memory caching

Task 1: Create the Request Router

Given an http.Request, look at the pathname and choose the handler to use. This should be pretty similar to what you've done with Lab 2/3. You'll need to create the following routes:

1. "/" which maps to Routes.home
2. "/public": which maps to Routes.public
3. "/about": which maps to Routes.about

You can see this mapping done in the RoutingTable object. Make sure that you know how to communicate a route that was Not Found back to a calling function. One solution you might use is to return a function which can handle the specified route or null if no route matches.

Task 2: Start the server

You should use the http.createServer function to start the server. Then, actually call your routeRequest function. I suggest attaching the staticFilesDirectory to the request object. It's common to attach a property like req.app which has app wide information handed to each request.

Task 3: Fill in the routes/js files

For the home.js and about.js files, read the corresponding files (index.html and about.html) respectively and serve back the content. These two functions should be roughly similar.

Once you're comfortable with those two functions, you should return any file from the public directory. It's a similar function, but you'll have to figure out the path of the file requested by the user. In this case, the request file may not exist, so return an appropriate response.

Task 4: Edit the static files served.

This server serves some CSS files and an about.html file. Fill in the commented section of the about.html and index.html files. Because we're just starting CSS, make sure you read through the CSS file and play around with it.

Task 5: Add Caching

In your server.js file, we're going to add a cache. These files aren't going to change often so one way to speed up the response to the user is to store things in memory instead of always reading a file.

Create a cache object at the top level of the server.js file. Then, in your request handler, check if the cache object has an entry which matches the URL the user is requesting. If the cache is "fresh" enough, return the cached version. If not, you'll need to have all your functions add cached responses to the cache object. So it's recommended to pass the cache object around similar to how you pass the staticFilesDirectory string around.

Task 6: Submitting

For this lab, you'll receive 15 points for demoing to the TA. You can do this during lab or office hours. If you can't make it to those, email the TA to coordinate something.

Similar to your last couple labs, remove node_modules and zip everything before uploading to Camino.