Updated Service Layer Design

Capstone Project

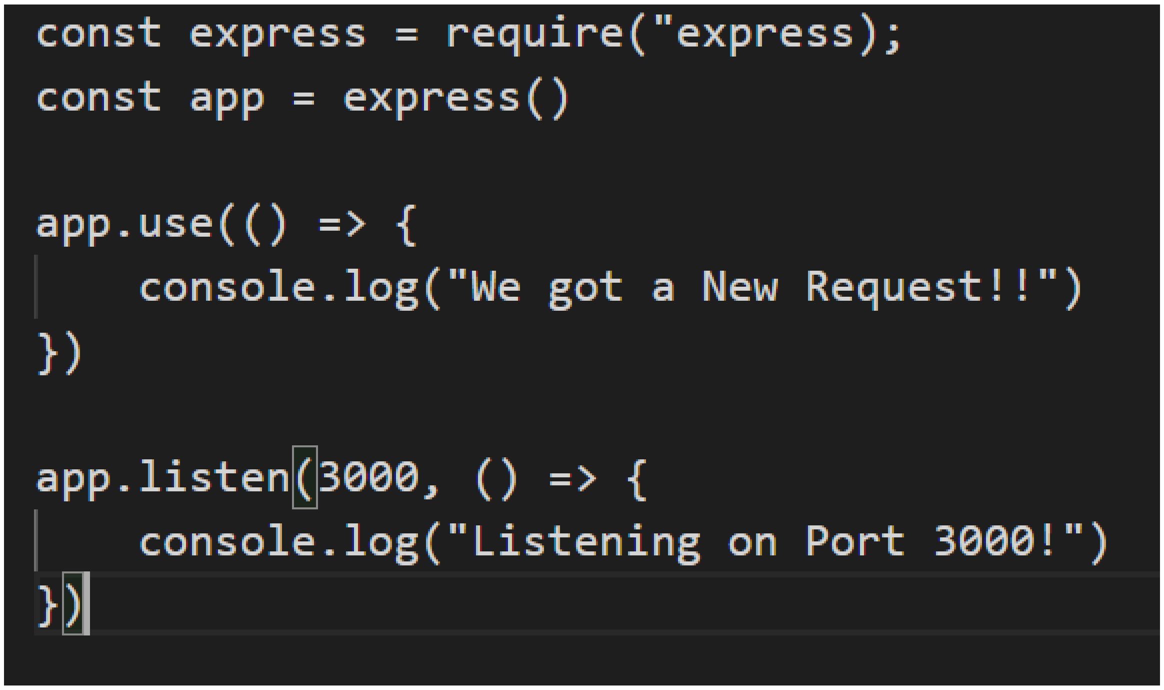
Brandon Beaudry

For my project I am using Mongodb, Mongoose, and Express. Express is a node package and is a web development framework that helps to make web applications. It helps to get the servers up and running with Node. It is designed for building web applications and Application Programming Interfaces (API's). API's is a software intermediary that allows two application to talk to each other.

Express helps with the listening of incoming requests and figures out what is being asked and what it is searching for. It also checks for thinks like the username and password and makes sure they are valid entries. Often it checks things from the database or saves something to the database. Then we build a response and sent that back to the client. Some of the most important part of what express does is it helps us to start up a server to listen for requests. It will parse incoming requests because HTTP requests are really just text information, not JavaScript objects or anything. It turns requests into objects and helps match those requests so that we could write code that only runs when requesting. It helps us craft our response and set the status code as well as many other things.

We install Express with NPM. After I download Express, I can make a file and we need to require express in order to use it. We then need to execute Express.

Below is an example of code used to get the app to begin to listen (using port 3000).



```
const express = require("express);
const app = express()

app.use(() => {
    console.log("We got a New Request!!")
})

app.listen(3000, () => {
    console.log("Listening on Port 3000!")
})
```

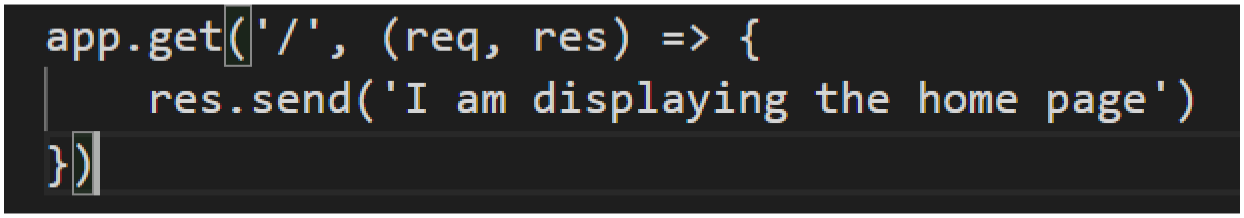
Then if I open my browser and type in localhost:3000 and hit enter with the above code The app.use(() section above will print out, "We got a New Request" in the terminal even though it isn't showing in the browser for the user. Every time a request hits our server when I hit enter it will print out, "We got a

New Request!!" in the terminal. However, the above does not respond in the client even though we got verification in the terminal. We need to require Express and execute it by saving it to a variable (app in this case) then call methods on that app including listen which is what starts the server.

In order to respond with content, we need two different parameters added to the function that are automatically passed in. The first is an object that represents request (req). The second is an object that represents the outgoing response (res). To see a list of request methods, see http://expressjs.com/en/4x/api.html#req. To see a list of the response methods, see http://expressjs.com/en/4x/api.html#res.

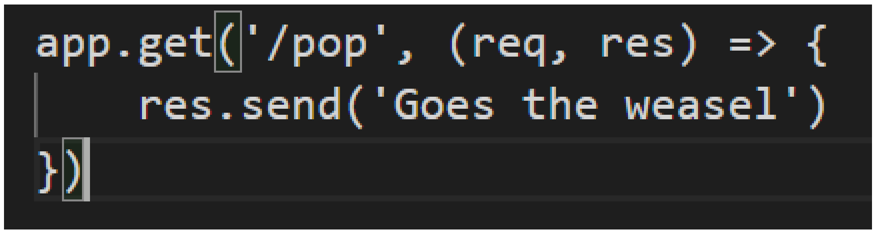
Under the response there is one for res.send([body]), which sends the http response. It will accept different types of data so we can respond with a string and it will respond with plain text. We could also respond with a JavaScript object and turn that into JSON as part of the response.

If we want to route a path, we can do the following to route to the homepage which when performed would display "I am displaying the home page":



```
app.get('/', (req, res) => {
    res.send('I am displaying the home page')
})
```

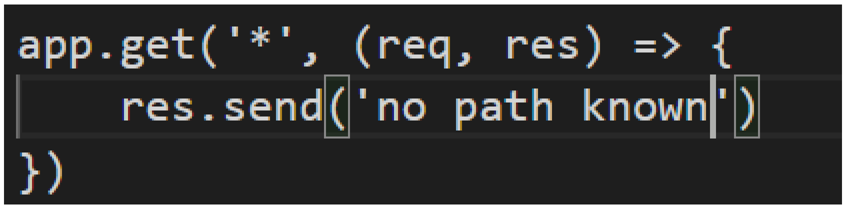
We could also create another path. For the below I made the example of /pop and then had it go to a separate page and display, "Goes the weasel". So the /pop would be added after the initial homepage address and it would route it to the page associated with /pop.



```
app.get('/pop', (req, res) => {
    res.send('Goes the weasel')
})
```

The above two examples only respond to get requests but we can also do post, put, or delete. To see the different options, we have for Express see http://expressjs.com/en/4x/api.html#app.

We can also do a route to display "no path known" for anything not already with a path established by using the following.

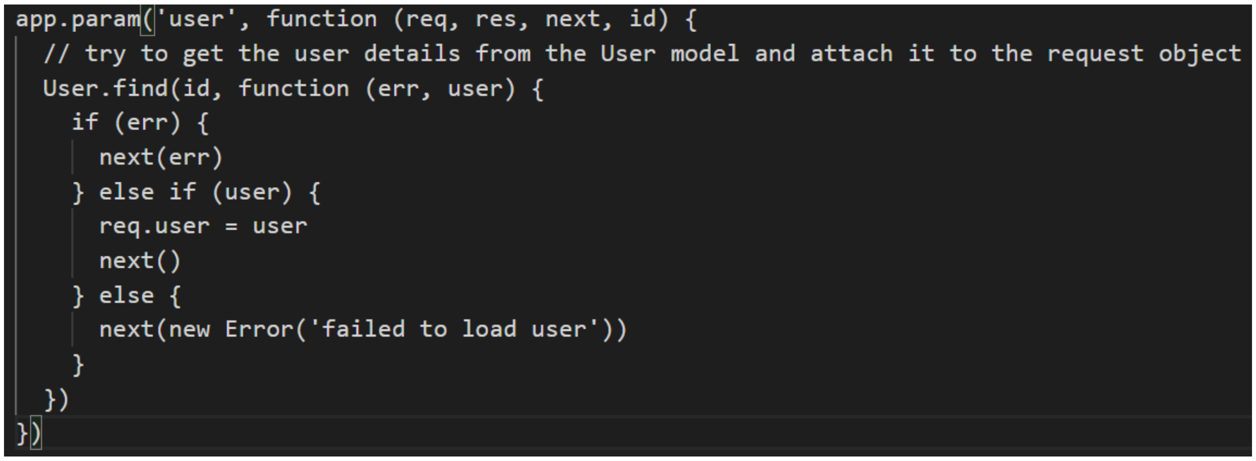


```
app.get('*', (req, res) => {
    res.send('no path known')
})
```

However, the above example must be done last because routes are done in order so if this is input before our other paths then everything we request will display “no path known” even if we try the path for /pop as an example.

We can route to parameters where the name is the name of the parameter or an array of them, and callback function. The parameters of the callback function are the request object, the response object, the next middleware, the value of the parameter and the name of the parameter in that order.

An example of a parameter is the following:



```
app.param('user', function (req, res, next, id) {
  // try to get the user details from the User model and attach it to the request object
  User.find(id, function (err, user) {
    if (err) {
      next(err)
    } else if (user) {
      req.user = user
      next()
    } else {
      next(new Error('failed to load user'))
    }
  })
})
```

<u>Examples of the updated explanation to show the http response.</u>

Scenario	The Request	Situation	Response
Add a destination	Request to add a destination	Successfully found add destination page	http display add destination page
	“ ”	Destination page not found	Error: destination page could not be found

Scenario	Request	Situation	Response
Delete destination	Request to delete destination	Warning: Make sure you wish to delete page	HTTP display Are you sure you want to delete destination? Yes/no
	“ ”	Successfully deleted destination	HTTP display destination deletion successful
	“ ”	Could not Delete destination	HTTP display Error unable to delete destination

Scenario	Request	Situation	Response
Add picture of destination	Request to add picture of destination	Successfully added picture	HTTP display picture successful
	“ ”	Could not add picture	HTTP display destination deletion successful

Scenario	Request	Situation	Response
Delete picture of destination	Request to delete picture	Warning: Make sure you wish to delete picture	HTTP display Are you sure you want to delete picture? Yes/no
	“ ”	Successfully deleted picture	HTTP display picture deletion successful
	“ ”	Could not Delete picture	HTTP display Error unable to delete picture