Week 3 Workshop

NodeJS Express MongoDB

Agenda

Activity	Time
Get Prepared: Log in to Nucamp Learning Portal • Slack • Screenshare	10 minutes
Check-In	10 minutes
Week Recap	40 minutes
Task 1 & 2	60 minutes
BREAK	15 minutes
Tasks 3 & 4 - Leave time for testing!	90 minutes
Check-Out	15 minutes

Check-In

- How was this week? Any particular challenges or accomplishments?
- Did you understand the Exercises and were you able to complete them?
- You must complete all Exercises before beginning the Workshop Assignment.

Week 3 Recap - Overview

New Concepts This Week

• Basic Authentication • Cookies • Sessions • Passport	• Token-Based Authentication • JSON Web Tokens • Mongoose Population

Next slides will review these concepts

Basic Authentication

- Simplest way to authenticate
- Set up a gatekeeping middleware function in your Express app placed *before* access to the resources you want to protect, and check the incoming request for an authorization header
- If there is no authorization header, send back a challenge to the client via setting this response header:
  - `res.setHeader('WWW-Authenticate', 'Basic');`
- If there is an authorization header, validate it against a stored username/password

Basic Authentication

- Basic Authorization header will be a string with the word "Basic" followed by a space, then the username and password separated by a colon & encoded in Base64
- Discuss: What is the difference between encoding and encrypting? Ask for a student to answer.

Basic Authentication

- Something that is encoded can be easily decoded (UTF-8 is another kind of encoding).
- Encoding is not meant to add anything to security, whereas encryption is meant to make something more secure.

Basic Authentication

- With Basic Authentication, the client must send the username/password back to the server every time it requests a protected resource
- Communication is "stateless" meaning server does not retain any information about the client, must be reminded again every time the client makes a request

Cookies

- Cookies are a way of adding a level of "state" to client-server communication – server can remember who client is
- Support for cookies is built into HTTP
- Once a user authenticates with valid credentials, server sends client a cookie with a small bit of information (e.g. username)
- Client then sends that cookie back in subsequent requests to the same server for authentication instead of sending username/password again each time, and server will recognize cookie as being from itself
- Small, fixed size, can only contain a little information
- Cookie-parser middleware both parses cookies from header and enables signing cookies to guard against tampering

Sessions

- Express Sessions is the library/module typically used in Express to handle session-based authentication
- Uses a cookie to store a session identifier on the client
- Server stores data about the client and the current authenticated session on server side, typically in a database or local file storage
- When client sends cookie in header to server, server uses session id from cookie to access server-side data about the client
- Not limited to cookie size, can store more information about client

Passport

- Passport is a popular Express authentication middleware that simplifies the hassles of authenticating
- Has 500+ plugins (called Strategies) for different ways to authenticate
- Local Strategy is just having the username/password stored locally, there are also many different third party strategies to authenticate with third party services (Facebook, Twitter, etc)

Passport

- Passport is able to use session-based or token-based authentication
- <u>**Discuss:**</u> What are two major drawbacks of sessions, reasons why you might want to use tokens instead? (answers next slide, but request student answers first)

Drawbacks of Sessions

- Two major drawbacks of sessions:
  - Not easily scalable as server must store and maintain information on users, if there are 1000s or more of users, then this can be hard on the server
  - Doesn't work well with mobile applications

Token-Based Authentication

- Token-based authentication does not store any data about the client on the server, thus is more scalable
- Mobile app-friendly
- Enables sharing authentication across applications
- Token data can be sent from the client in request header, request body, or as URL query parameter (part of the URL)

JSON Web Tokens (JWTs)

- Standards for JSON Web Tokens are maintained in IETF RFC 7519
- Most commonly used form of tokens for authentication
- Three parts: **Header**, **Payload**, **Signature**
- Each part is separately encoded with Base64Url encoding then joined together with two periods
- NPM library jsonwebtoken provides Node API
- passport-jwt library provides Passport Strategy

Mongoose Population

- MongoDB is a NoSQL document database
- SQL relational databases are built around records being able to reference each other
- NoSQL databases do not have a lot of functionality for referencing/linking each other
- Mongoose Population provides a way to reference documents in one collection with documents from another collection:
  - Set type of field to mongoose.SchemaTypes.ObjectID
  - Set Ref property of field to name of model for other collection
  - When needed, use .populate() method to populate data from other collection

Workshop Assignment

- It's time to start the workshop assignment!
- Be sure to have watched the assignment video to understand how to create an admin user!
- Leave yourselves time for testing with Postman after Task 4.
- Break out into groups of 2-3. Sit near your workshop partner(s).
  - Your instructor may assign partners, or have you choose.
- Work closely with each other.
  - 10-minute rule does *not* apply to talking to your partner(s). You should consult each other throughout.
- Follow the workshop instructions very closely.
  - Both the video and written instructions. Pay careful attention to any screenshots in the written instructions.
- Talk to your instructor if any of the instructions are unclear to you.

Check-Out

- Submit to the learning portal one of the following options:
  - Either: a zip file of your entire nucampsiteServer folder with your updated files, *excluding* the node_modules folder,
  - Or: a text file that contains the link to a public online Git repository for the nucampsiteServer folder.
- Wrap up – Retrospective
  - What went well
  - What could improve
  - Action items
- Start Week 4 or work on your Portfolio Project.
- If everyone is done early, then take time to go over the Code Challenges and Challenge Questions from this week – for each one, a student volunteer who has completed the challenge may explain their answer to the class.