CMSC335

Web Application Development with JavaScript

Router, Cookies, Sessions

Department of Computer Science

University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

REST (Representational State Transfer)

- **An architectural style**; not a protocol
  - Designed to operate with resource-oriented services (locate/manipulate resource)
  - Allow different data formats (e.g., html, text, JSON)
  - Advantages: Fast, language, and platform independent
- **Resources are represented by URLs**
  - Resource - document, person, location
  - Each resource has a unique URI
  - Each resource can be dynamically generated instead of having an actual page/document
- Operations are performed via HTTP methods (GET, POST, PUT, DELETE) with resources
- Video: https://www.youtube.com/watch?v=-MTSQjw5DrM&t=173s
  - Up to time marker 2:29
- Alternative: GraphQL
  - Video: https://www.youtube.com/watch?v=eIQh02xuVw4

Router

- To manage the complexity of many routes in your main app, you can define files that take care of some of them. In the file you create, a router will take care of some of those routes
- The Router object (that can handle .get, .post, etc.) is created using express.Router()
- Here is an example of a router for requests that start with **/building**

```
/* Code in file building.js */
const express = require('express');
const router = express.Router();
router.get("/", (request, response) => { response.send("/ in building.js") });
router.get("/iribe", (request, response) => { response.send("/iribe in building.js")
});
```

- For the above code in the main app we have

  app.use("/buildings", buildings);

- **Example:** Router

Cookies

- Cookie - small piece of information sent by a server and stored either in the browser's memory or as a small file in the hard drive. Acceptance of the cookie depends on the client
- **Browser sends the cookie back with every request to the server that sent the cookie**
- Cookie - contains a name/value pair. This is how the cookie information may look like when sent by the server in the http header

  Set-Cookie: automobile=nelyota; path=/; domain=notRealCars.comr
- Setting a cookie - associating a value with a name
- Getting a cookie - getting the value associated with a name
- Cookie size - 4KB per cookie

Cookies with an Expiration Date

- Cookies without an expiration date will expire when the browser is closed
- You can see cookies in Chrome by right-click on the page and selecting “Application” and under “Storage” expanding “Cookies”
- In Node we need the cookie-parser module
  - We use **response.cookie** to set a cookie
  - We use **response.cookies** to access cookies
- **Example:** Cookies (in Node)
  - Type: http://localhost:3000/
  - Type: http://localhost:3000/check
  - Type: http://localhost:3000/terps

Sessions

- **Session** - time period during which a person views several different web pages in a browser and then quits
- **What would you like**
  - To keep track of information throughout the session. For example, keeping track of color preferences, usernames, data selection, etc.
- **What is the problem?**
  - http (the protocol that makes possible the communication between browsers and web servers) is stateless
  - Stateless - every page request is independent
- **Solution** - Sessions
- In Node we can use the express-session module to have session support
  - **request.session** used to store variables
    - » E.g., request.session.name = “Mary”;
  - Use **request.session.save()** to save session variables
  - Use **request.session.destroy()** to destroy a session
  - Sessions rely on cookies

Sessions

- **Example:** Sessions/app.js
  - Using Insomnia
    - » Send a POST request http://localhost:3000/login
      - You will get an “Invalid user” message
    - » Using Insomnia, send a **POST** request to http://localhost:3000/login using “Form”→“Form URL Encoded” and the parameters **user** and **password**. The values will be ***peter*** and ***terps***, respectively.
      - You will get a “User has logged in” message
    - » Send a **GET** request http://localhost:3000/browse
      - You will get a “Welcome back peter, browse” message

Sessions

- **Example:** Sessions/app.js
  - Using Insomnia
    - » Send a **POST** request http://localhost:3000/buy using "Form"→"Form URL Encoded" and the parameter **item** with the value **tv**
      - You will get an "tv added to your cart" message
    - » Send a **POST** request http://localhost:3000/checkout
      - You will get an "Items you are buying are tv" message
    - » Send a **POST** request http://localhost:3000/logout
    - » You will get a "You have logged out" message
    - » Try to add buy another article after logging out
  - Let's see the cookie with session information

Express Application Generator

- Link: https://expressjs.com/en/starter/generator.html