STATE MANAGEMENT

Stephen Schaub

State Management Options

- Global
- Per-Request
- Cookies
- Session

Globally Shared State

- Data to be shared by all users can be stored in global script variables
- To expose a global object throughout an express app, add it as a property of app.locals
  - app.locals.db = mydatabase;
- Once set, a property persists in app.locals for the lifetime of the application
- Use for:
  - Constants and objects needed throughout app

Per-Request State

- Data needed for the duration of a single request can be stored in the res.locals collection:
  - res.locals.productId = req.params.id;
- This collection does not persist between requests
- Use this to pass information along Node.js middleware chain

Cookies

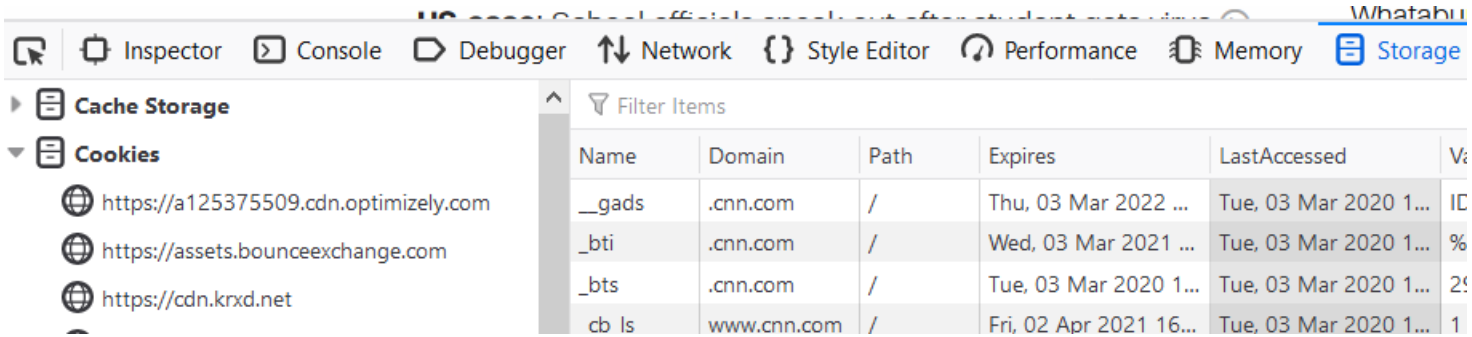
- Use cookies for data that needs to persist between requests
- Set a cookie in middleware or route callback using res.cookie()

  res.cookie('rememberme', '1', { maxAge: 900000 });
  - Must set cookies before using res.send() or res.render()
- res.cookie() sends cookie to browser using Set-cookie: HTTP header

  Set-Cookie: rememberme=1; Expires=Wed, 21 Oct 2015 07:28:00 GMT
  - See https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie
- Browser sends cookie back to server on subsequent requests until cookie expires
- Access in middleware or route callback using req.cookie.*cookiename*
  - Requires cookie-parser middleware
- Example: webapps/cookiedemo

Viewing Cookies in the Browser

- Use browser developer tools



Session

- Cookies give us a way to store per-user state in the browser
- Web applications often need a way to maintain per-user state on the server
- Use express-session module to get session state capabilities
- Example: webapps/sessiondemo

How express-session Works

- Maintains a collection of session objects in memory
- Each session object is associated with a separate browser that has recently accessed the app
- A small cookie associates remote browser with a session object
- express-session middleware checks for the session cookie
  - If found, lookup session object associated with cookie and populate req.session with the object
  - If not found, create new session cookie and associate with new session object
- For production, must pick an appropriate session storage mechanism
  - Default express session store does not expire sessions, causing memory leak

Cookies vs. Session

Cookies

- Store limited amount of state
- Increase size of HTTP request/response
- Cookie data subject to modification by client

Session

- Larger storage capacity
- In-memory sessions limit scalability
- Not subject to client tampering

Passing Data Between Pages

- Query Strings
- Hidden Form Fields
- Session
- Cookies

- See examples/webapps/register_validation

Navigating Between Pages

- Multi-page applications need to navigate between pages
- Two techniques:
  - Client-side redirect
  - Server-side transfer
- Issues:
  - Does browser URL reflect current page?
    - Related: Will user bookmark / refresh the resulting page?
  - Data transfer from source to target page
  - Performance

Page Navigation Techniques Compared

Client-Side Redirect

- Use res.redirect("/dest/page")
  - Outputs 302 to browser
  - Browser sends GET request for /dest/page
- Transfer data between pages using cookie, session or query string

Server-Side Transfer

- Use res.render("dest/page")
- Pass data from source page directly to target template

Flash Messages

- When transferring from one page to another with a redirect, want a convenient way for the source page to set a message to be displayed on the destination page
- connect-flash middleware uses the session to do this
- https://github.com/jaredhanson/connect-flash