
THE UNIVERSITY
of ADELAIDE
Web and Database Computing
adelaide.edu.au
Server Architecture and Routes: Cookies & Sessions

Managing State

State in a stateless protocol

Recall that HTTP is **stateless**

- Each request is unrelated to any previous requests.
- The server doesn't retain infomation about individual requests.

But **sometimes we need state**; how can we address this problem?

- On the server, we can store state in databases, but how do we connect this data to new requests?
  - e.g. if a user has a shopping cart and we want to show them the contents that was in their cart when they last visited our page.
- What if we want to store short term information (only since the user logged in) that we don't want to store long term in the database.
  - e.g. Pages visited since logging in?
- We *could* make users login to identify themselves?
  - But then we can't have anonymous shopping or browsing.
  - Login data would need to sent with **every request**

Cookies

A **HTTP cookie** is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing.
The browser sends the cookie in the header of each new HTTP request to that website.

Cookies were created to allow the server (e.g. express) to store state on the client (e.g. browser)

- Widely used.
- Stored on client-side.
- Can be viewed and modified both client-side and server side.
- Can be used to store **small** pieces of data (up to 4,096 bytes) that need to be sent to the server with each request.
- Can be used to uniquely identify different clients, avoiding need to log in with each request.
- Can be set to expire at a specific date/time (persistent). If no expiry provided, will be cleared by the browser when closed (sessional/transient).

Cookies in Express

Express-generator provides the cookie-parser middleware to automatically handle cookies.

Sending a cookie to the client:

```
res.cookie('cookie_name', 'cookie_value');
```

- Do this before your `res.send` or `res.json`
- Results in this cookie on the client:
- Which will be present in susequent requests to the server:

Cookies in Express

Accessing a cookie sent by the client:

```
req.cookies.cookie_name;
```

Additional sending options:

We can set additional options with the options object

```
res.cookie('cookie_name', { domain: '.example.com', path: '/admin', secure: true });
```

Additional options include:

- **expires** Expiry date of the cookie in GMT. If not specified or set to 0, creates a session cookie.
- **maxAge** Convenient option for setting the expiry time relative to the current time in milliseconds.
- **path** Path for the cookie. Defaults to “/”.
- **secure** If true the cookie may only be used with HTTPS.

Some references for Cookies in express:

Basic tutorial: https://www.codementor.io/nodejs/tutorial/cookie-management-in-express-js

Express API for reading cookies https://expressjs.com/en/resources/middleware/cookie-parser.html

Express API for setting cookies http://expressjs.com/en/4x/api.html#res.cookie

State on the Server

Server Sessions

Assume a user has logged in. How do we track information about the user since logging in?

- We could just store all the data we want to keep as cookies which expire when the browser is closed, **BUT**
- All of this information is visible to the user; they can look at their cookies.
- All of this information has to be sent to the server with every request.

This approach can expose information about the internal workings of your web application (product id numbers, user id numbers, etc) which can be a security risk – if I edit the cookie and change the user id can I get someone else's information?

Server Sessions allow this temporary user information to be stored at the server instead of the client.

Server Sessions & Session Tokens

Cookies can be used to uniquely identify clients, avoiding the need to send login data with each request.

- The sessions software sets a cookie at the client browser with a cryptographically unique session id, known as a **session token**.
- Whenever the browser sends requests to this website, it sends incldues this data (because cookies)
- The server uses the unique session id to associate that client with data that it stores.
  - We can store as much data as we like against the session.
  - This data is stored server-side (secure)
- When the session token expires or the server shuts down the session data is lost.

Set Up Server Sessions in express

Need to setup the express-session middleware in `app.js` (not included by default)

```
...

var session = require('express-session');    // THIS CODE //

var app = express();

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());

app.use(session({                             //           //
  secret: 'a string of your choice',          //           //
  resave: false,                              // THIS CODE //
  saveUninitialized: true,                    //           //
  cookie: { secure: false }                   //           //
}));                                          //           //

app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);

module.exports = app;
```

Using Server Sessions in express

The session middleware will automatically handle most of this for us:

- Creates the sessions and issues Session Token to the client
- Stores session data on the server and maps it to for you and maps it the Session Token of incoming requests.
- Makes this stored data available for each request using `req.session` variable.

You can set and read session variables using `req.session` in your routes in `index.js`

```
req.session.variableName = 5;
console.log(req.session.variableName);
```

It is also possible to store your sessions in your database to make them survive the server shutting down

- https://www.npmjs.com/package/express-mysql-session
- This is beyond the scope of our course, but provided for y'all to have fun with.

Cookies vs Server Sessions

Cookies or Sessions?

Duration

- Both will be kept according to the expiry date on the cookie/session token.
- If no expiry date is present, these will usually be lost when the browser closes.

Storage Capacity

- Cookies can be as large as 4,096 bytes in size.
- A session token itself is just a small cookie, but since the data is stored on the server, there is no limit on the amount of data that can be stored against a server session.

Security

- Cookie data is stored on the client side and can be fully accessed/modified by the client.
- While the Session Token itself is stored on the client, its contents is useless to the client.
- The Session data itself is stored on the server and never exposed to the client.
- A Session Token ID is cryptographically generated so it is effectively impossible to forge.

Summary

- HTTP is a stateless protocol, but we often need to maintain state between requests.
- Cookies allow storing small pieces of data on the client that are sent with each client request.
- Server sessions simply use a special cookie known as a Session Token to associate different requests from the same client with data stored on the server.

THE UNIVERSITY
of ADELAIDE