Web and Mobile App Development

Sessions and User Control

Material created by:
David Augenblick, Bill Mongan, Dan Ziegler, Samantha Bewley, and Matt Burlick

Sessions and User Control

- There's two more common things we want for our dynamic sites:
  1. The ability to control what users can do/see by having login/logout functionality and permissions.
  2. The ability to keep track of information between pages
     - In particular the current user and his/her permissions.

Login / Logout Considerations

- Our users will typically be kept in some database.
- And there will typically be at least one “superadmin”
- So first thing’s first…
  - Let’s create a database table and insert an initial “admin” user
  - Our database should have
    - UserID – Autoincrement, Key
    - Username – Unique string, non-null
    - Password – Non-null

Login / Logout Considerations

- So in our MYSQL command line client let’s do the following
  - `CREATE DATABASE <dbnam>;`
  - `USE <dbname>;`
  - `CREATE TABLE users(userid INT KEY AUTO_INCREMENT, username VARCHAR(20) UNIQUE NOT NULL, password VARCHAR(100) NOT NULL, type CHAR(1) DEFAULT=1);`
  - `INSERT INTO users (username, password, type) VALUES(‘myname’, PASSWORD(‘mypass’), 2);`
- Note the use of the MYSQL PASSWORD function
  - This creates a hash of the password so we don’t store it in plaintext!

Login / Logout Considerations

- Now let’s make a bunch of endpoints!
  1. Starting Page
  2. Login page
  3. Protected “landing” page with links to
     - Logout page
     - Administer users page
- We’ll use a mixture of serving content directly within the server and routing to other Nodejs scripts based on complexity.

Starting Page

- Let’s just have a simple page with a username and password fields and a button to login.
- Now we have several choices
  - Do we do everything via ajax right in this page?
  - Or do we move between URLs?
- Probably a UX decision….
- In the following slide we’ll also make use of the `redirect` method of the response object:

```
res.redirect(<whereto?>);
```

Node Server

```
var database = require('./controllers/database');
var db = new database();

app.get('/', function (req, res){
    res.write(`<html>
        <body>
        <form method=post action='/login'>
        <input type=text name=username>
        <input type=password name=password>
        <input type=submit value=Login>
        </form>
        </body>
        </html>`);
    res.end();
});
```

```
app.post('/login', function (req, res){
    db.once('loggedin', function(msg){
        if(msg==1){
            return res.redirect('/getUsers');
        }
        else{
            return res.redirect('/');
        }
    });
    db.login(req.body.username, req.body.password);
});
```

Node Server

```
app.get('/getUsers', function(req,res){
    db.once('usertable',function(rows){
        var str = "<table><th>User</th><th>Permissions</th>";
        for(var i=0; i < rows.length; i++)
            str += "<tr><td>"+rows[i].username +
                "</td><td>"+rows[i].type+"</td></tr>";

        str +="</table>";
        str +=`<br>Add User
            <form method=post action='/addUser'>
            Username: <input name=username>
            Password: <input name=pass>
            Type <select> name = type
                <option value=1>User</option>
                <option value=2>Admin</option>
            </select>
            <submit value='Add User'>
            </form>`;
        res.write('<html><body>' + str+'</body></html>');
        res.end();
    });
    db.getUserTable();

});
```

Database Class

```
'use strict'

var EventEmitter = require('events').EventEmitter;
var mysql = require('mysql');

var dbinfo = require('../Passwords/databaseinfo.json');

var con = mysql.createConnection(dbinfo);
con.connect(function(err)  {
     if (err)  {
          console.log('Error connecting to database');
     }
     else  {
            console.log('Database successfully connected');
     }
 });

class Database extends EventEmitter{
     constructor(){super();}
     login(username,password){
          //next slide
     }
     getUserTable(){
          //next slide
     }
}
exports.Database = Database
```

Database Class

```
    login (username,password){
        var str = "SELECT type FROM users WHERE username="+con.escape(username)
            + " AND password=PASSWORD("+ con.escape(password) +")";
        var self = this;
        con.query(str,
            function(err, rows, fields){
                if(err){
                    console.log('Error');
                    return 0;
                }
                else{
                    if(rows.length>0)
                        self.emit('loggedin',1);
                    else
                        self.emit('loggedin',0);
                }
            }
        );
    }
```

Database Class

```
    getUserTable (){
        var str = 'SELECT username, type FROM users order by username';
        var self = this;
        con.query(str,
            function(err, rows, fields){
                if(err){
                    console.log('Error');
                    return 0;
                }
                else{
                    self.emit('usertable',rows);
                }
            }
        )
    }
```

Sessions

- But who’s to stop anyone from just going to the endpoint /getUsers
- We’d like to check the status of the current user to see if he/she has the credentials to view this page
  - And/or modify it’s content based on the user’s permission.
- A common way to do this is to store *session* information.
- Sessions enable you to keep track of information between pages as they pertain to a particular visitor or your site.
- As with cookies (which are stored on the user's browser), sessions can keep users authenticated (logged in) as they re-load a page
- We’ll use the `client-sessions` module for this
  - `npm install client-sessions`

Sessions

- We just need to load the client-sessions module and bind to our express app the configured session.
  - Including a secret key which is like an ID for our app
- Now we can get/set session variables in the request object

```
var express = require('express');
var app = express();
var session = require('client-sessions');
app.use(session({
  cookieName: 'session',
  secret: 'asdfasdf23423',    //we could load all this in from an external file
  duration: 30 * 60 * 1000,
  activeDuration: 5 * 60 * 1000,  //if timeout, but active, extend timeout by this much
}));

// create routes and apply sessions to them
app.get('/helloWorld', function(req, res) {
    if(req.session.lastpage) {
        res.write('Last page: ' + req.session.lastPage + '. ');
    }
    req.session.lastPage = '/helloWorld';
    res.write('Hello World. ‘);
    res.end();
});
```

Sessions

- We also may want to reset a session, destroy it, or delete individual things from a session.
- We can do that by
  - `req.session.destroy();`
  - `req.session.reset();`
  - `delete req.session.<sessionitem>;`

Log-In Sessions

- Ok let's add some sessioning to our app!
- When the user logs in (successfully) we set some session info
    - Username
    - Type
- We'll also use the session to store messages
    - So if the person didn't log in we can yell at them!
- And we can use the stored session information to decide if a visitor should see a particular endpoint
    - And/or customize it based on their permission.

Logging In…

```
app.post('/login', function (req, res){
    db.once('loggedin', function(msg){
        if(msg==1){
            req.session.userid=req.body.username;
            return res.redirect('/getUsers');
        }
        else{
            req.session.msg = "Invalid login";
            return res.redirect('/');
        }
    });


    db.login(req.body.username, req.body.password);
});
```

Verifying Login

```
app.get('/getUsers', function(req,res){
    if(!req.session.userid){
        req.session.msg = 'Not allowed there';
        return res.redirect('/');
    }

    // what to do if logged in…

});
```

```
app.get('/', function (req, res){
    res.write(`<html><body>`);
    if(req.session.msg){
        res.write(req.session.msg);
        delete req.session.msg;
    }
    res.write(`
        <form method=post action='/login'>
        <input type=text name=username>
        <input type=password name=password>
        <input type=submit value=Login>
        </form>
        </body>
        </html>`);
    res.end();
});
```

Logging Out

```
app.get('/logout', function (req, res){
    req.session.reset();
    req.session.msg = 'You logged out';
    return res.redirect('/');
});
```